TECHNISCHE UNIVERSITÄT DRESDEN

Vodafone Chair for Mobile Communications Systems

Technical University Dresden
Fakultät Elektrotechnik und Informationstechnik

Integration and Analysis of a Vector Processor in a Multiprocessor System-on-a-Chip

Master Thesis

Autor: Chen Zehao
Matrikelnummer: 5049404

Course of Study:	Nanoelectronic Systems
Examiner:	Prof. Dr. Gerhard Fettweis Dr. Emil Matúš
Supervisor:	Dipl.-Ing. Viktor Razilov Dr.-Ing. Mattis Hasler
Processing Period:	02. January 2024 - 11. June 2024
Date of Submission:	11. June 2024

Selbständigkeitserklärung

Statement of Authorship

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich reiche sie erstmals als Prüfungsleistung ein. Mir ist bekannt, dass eine Täuschung mit der Note „nicht ausreichend“ (5,0) geahndet wird.

I hereby declare that I have written this thesis independently and have listed all used sources and aids. I am submitting this thesis for the first time as part of an examination. I understand that attempted deceit will result in the failing grade „not sufficient“ (5.0).

Name / *Last Name*: Chen

Vorname / *First Name*: Zehao

Matrikelnummer / *Matriculation Number*: 5049404

02.06.2024 — Datum / *Date*

Chen Zehao — Unterschrift / *Signature*

Abstract

Nowadays, advanced micro electric system becomes more complicate. Some specific processing units are design to fit applications, such as GPU for dealing with Graphic, NPU for accelerating AI programs and TPU for machine learning. Therefore, heterogeneous systems are developed to integrate these micro-hardware more easily. The current M^3 system[5] is a advanced software/hardware co-design heterogeneous system which is already integrated with RISC-V cores. However, for current cores do not have the instruction set architecture (ISA) extension to deal with vector instructions. So we integrate the RISC-V RVV1.0 extension supporting core Ara[11] into the system. Analyzing the area cost and performance of Ara to get a matching configuration for specific user case.

Also, as the system becomes multicore and a problem is rising: cache coherence. We study the current cache coherence system BlackParrot[19] to understand the principle of cache coherence and propose a cache coherence architecture for M^3 hardware system. Also some analysis are came up with the feasibility and performance.

Contents

List of Figures

List of Tables

1 INTRODUCTION

Nowadays, embedded systems are become rather complex. They must be specifically designed to adequately fulfill the execution of these specific computational processes while meeting application requirements related to functional behavior, response time or throughput, energy consumption, geometric dimensions, price, and other attributes. The latest progress in nanoelectronic technology and its associated new applications has created a scenario where increasingly complex embedded systems require real-time computations to be carried out on extremely tight schedules, meeting high demands for energy, power consumption, area, and cost efficiency. Furthermore, many embedded systems need to be sufficiently flexible to allow for extensive reuse between different product versions, adapt to market changes, adhere to evolving standards or user requirements, and undergo simple modifications during their development or usage processes [15].

All of these factors lead to significant design and development challenges, such as optimizing multi-objective MPSoCs or new instruction set architecture (ISA) like RISC-V.

Multiprocessor System-on-a-Chip (MPSoC)

MPSoCs are complex integrated circuit systems that combine multiple processor cores onto a chip [12]. MPSoCs are designed to do perform the unique benefits of each core. In MPSoCs, processor cores include general-purpose processing unit (CPU), graphics processing unit (GPU), digital signal processor (DSP), and integrated circuit for specific applications(ASICs).

RISC-V and "V" Vector Extension

RISC-V is an ISA for customizable processors. The basic instruction set of RISC-V is 32 bit or 64bit Integer set, however, it has various extensions including Multiply and Divide instruction Extension (M), Atomic instruction extension (A), Floating-Point instruction extension (F).

The RISC-V Vector Extension (RVV) is an important part of the RISC-V ISA. It is designed for tasks which can benefit from Single Instruction Multiple Data (SIMD) architectures.

Aim of the Work

The M^3 MPSoC system [5] is developed trying to lower the complexity of integration of CPUs and accelerators for specific applications. There already two type of RISC-V processors: one is 5-stage in-order scalar RV64IMAFD RISC-V core Rocket [3] and another is out-of-order machine RV64GC RISC-V core BOOM [27]. To make the M^3 hardware platform to support vector instructions, we choose RVV compatible RISC-V Ara SoC [18] to integrate in the M^3 hardware.

Also, the advanced MPSoC system has multi-processors working together. So the cache coherence problem is a critical factor of MPSoC performance. To explore the cache coherence applicable solutions on the M^3 hardware system, cache coherence implications are mentioned in the thesis. To get possible coherence model and analyzed performance of M^3 hardware system.

2 BACKGROUND

The thesis is based on several important hardware platform, which are M^3 hardware MPSoC [21], Ara 64-bit Vector Unit SoC [11], BlackParrot RISC-V multicore platform [19] and Openpiton manycore framework [8].

2.1 The M^3 Hardware/Software Platform

The M^3 hardware/software platform aims to provide an MPSoC system based on hardware/software co-design, which can simplify the integration of CPUs and accelerators for specific applications. In the M^3 system, tiles (comprising cores or accelerators) are partitioned and allocated to different applications. This feature enables specific applications to run exclusively on each tile. Consequently, it optimizes resource utilization and exhibits improved computational efficiency [5].

And in M^3 system, communications are achieved by a custom per-tile hardware component known as the Data Transfer Unit (DTU). The DTU component can provide a uniform interface to all tiles, which can simplify the integration of heterogeneous systems. And in advanced version of M^3, the DTU is also referred as Trusted Communication Unit (TCU) [4], which is mention in chapter 2.2.

2.1.1 The M^3 Operating System(OS)

M^3Linux enables running Linux on an isolated partition within M^3. It consists of the Linux operating system, riscv-pk [6] with a bbl bootloader, and various applications. These applications possess the capability to interface with both M^3 and Linux, effectively bridging the gap between the two systems [17].

2.1.2 The M^3 Hardware System

The M^3 hardware MPSoC architecture consists of multiple heterogeneous blocks interconnected by a Network-on-Chip (NoC). Each block may contain a RISC-V processor or accelerators referred to as Processing Modules (PMs). Additionally, blocks can interface with multiple I/O peripherals (network blocks) or off-chip memories (storage blocks). TCUs can isolate different blocks and establish dedicated communication channels. Only when the M^3 core is running a processor within the selected block does it have the authority to configure the communication channels [21].

The NoC is a packet-switched network which contains four NoC routers, connections between tiles and routers. Those routers can be connected by various typologies, such as mesh, hexagonal or octagonal to accommodate the expected traffic patterns.

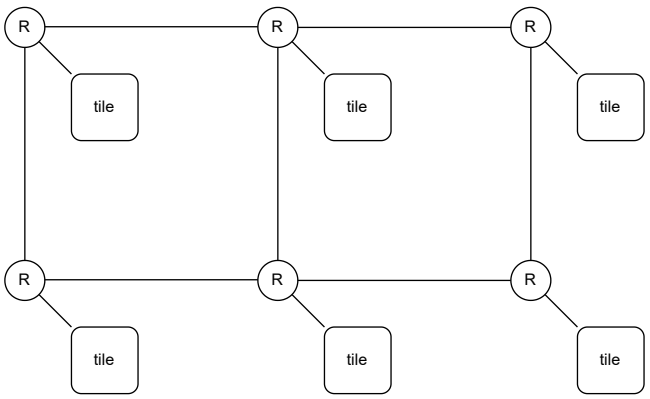

Figure 2.1: The Network-on-chip system, mesh typology

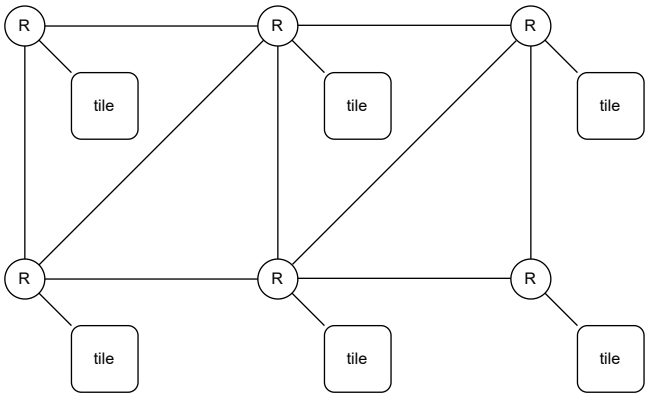

Figure 2.2: The Network-on-chip system, mesh with diagonal links

2.2 Trusted Communication Unit

The TCU is a fundamental component to build secure hardware system [4]. In M^3 system, the TCU, the interconnect, and the kernel tiles are trusted, other user tiles are untrusted.

Figure 2.3 illustrates the internal architecture of one tile. The CU is connected to the TCU and can access the TCU's registers via memory-mapped input/output (MMIO). By configuring TCU regfiles, the tile can change its communication channel or kernel privilege.

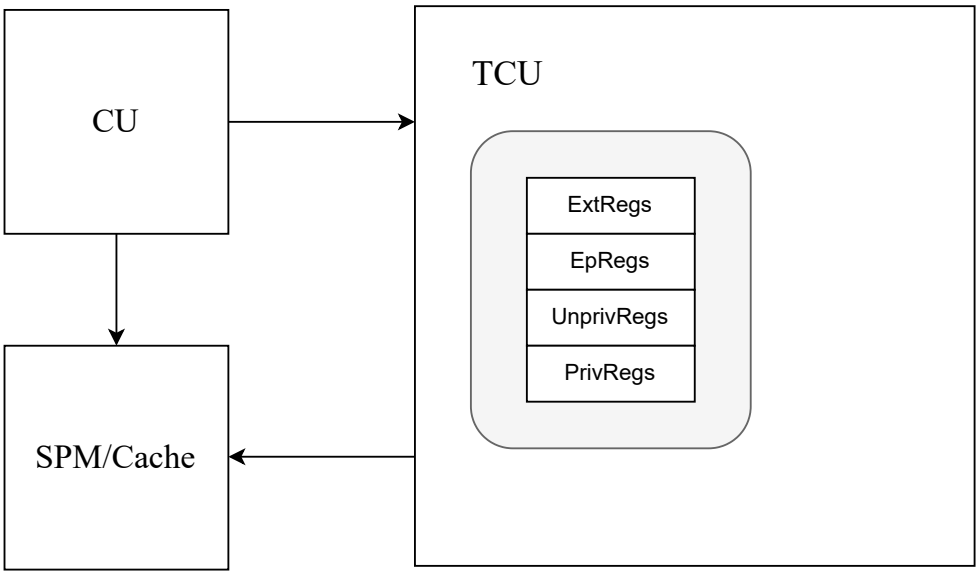

Figure 2.3: The internal organization of a tile, and TCU contains four regfiles: Endpoints regfile, Unprivileged regfile, Privileged regfile and External regfile [4].

The TCU plays a crucial rule in managing tile isolation and cross-tile communication. It has two key advantages. First, it simplifies the management and utilization of heterogeneous tiles, such as coordinating interactions between cores and accelerators through a common defined interface. Secondly, this architecture can work with OS, ensure that most of cores and accelerators remain outside the trusted computing base. And this setup can allow cores running applications to full access to all OS features without requiring other parts of the system to trust core or the software on it.

2.3 Ara: A RISC-V SoC with 1.0 RVV support

Ara contains of two main components. One is a 6-stage RISC-V CPU processor Ariane [25] and another one is the 64-bit vector RISC-V co-processor Ara. Ariane can support RV64IAMC and is a 64-bit, single issue, in-order core fabricated using 22-nm FD-SOI technology. Also, Ariane contains several important features, including privilege instructions and interrupt.

Ara co-processor is based on 1.0 RVV. And its micro architecture is scalable, containing configurable number of lanes, each comprising part of the processor's vector register file and functional units.

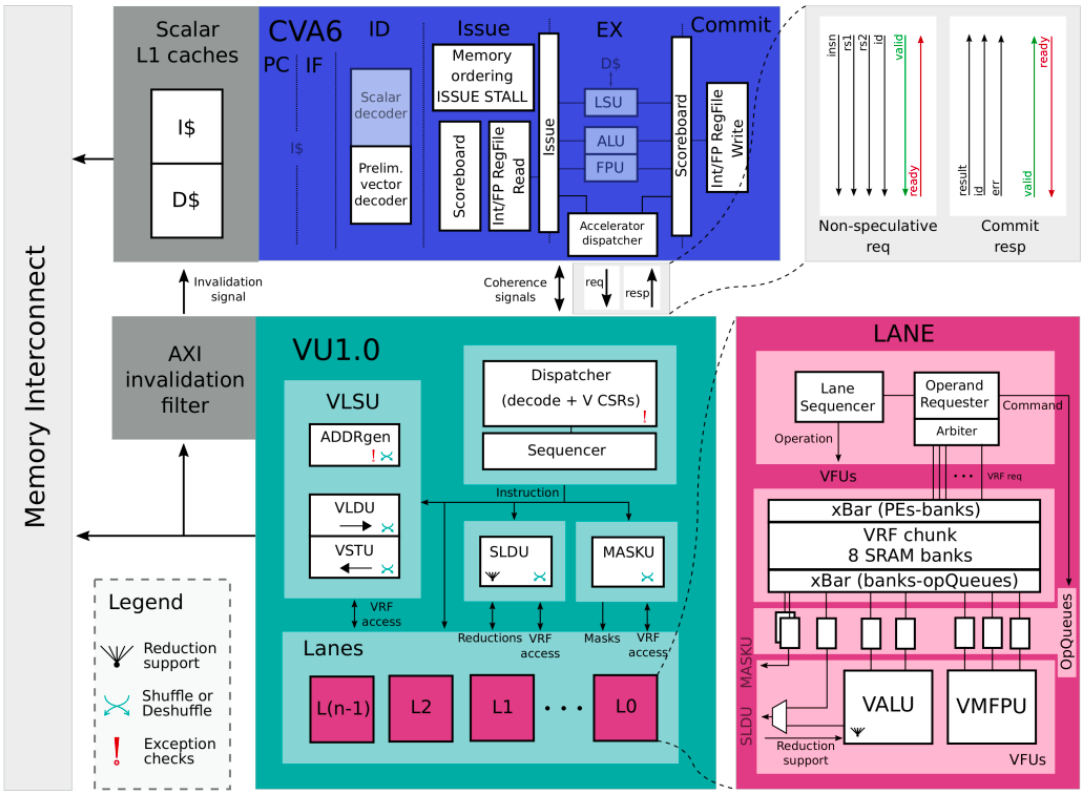

Figure 2.4: Ara's top level diagram, which contains two processors: RISC-V RV-IMC Ariane and RVV co-processor Ara [18].

When Ara executes a 256×256 double-precision matrix multiplication across 16 lanes, Ara can achieve up to 97% Floating-point Unit (FPU) utilization. Then Ara can execute at over 1GHz under typical conditions(TT/0.80 V/25°C), and it can have a performance of up to 33 DP-GFLOPS. Ara can attain up to 41 DP-GFLOPS/W under the same conditions.

2.4 BlackParrot: A RISC-V Multicore Platform

The BlackParrot platform is a SoC to integrate with accelerators and linux-capable 64 bit RISC-V cores. It is a multi-core platform which focuses on solving coherence problem. BlackParrot is based a distributed directory-based cache coherence protocol BedRock and can realize MSI and MESI. And the platform is also scalable and heterogeneous, which is shown in Figure 2.6. The BlackParrot chip is validated by a GlobalFoundries 12-nm FinFET tapeout.

The MSI and MESI protocol are the finite-state machine based coherence protocol. There are 4 states as the coherence states, which are I (invalid), S (shared), E (exclusive), M (modified). The difference between MSI and MESI protocol is that MESI protocol's E state can reduce the bus traffic caused by writes of data blocks that only exist in single cache. The finite-state machine of MESI is shown in Figure 2.5. All caches will monitor all the transactions on the bus. The state of the cache block is changed according to the MESI protocol state machine.

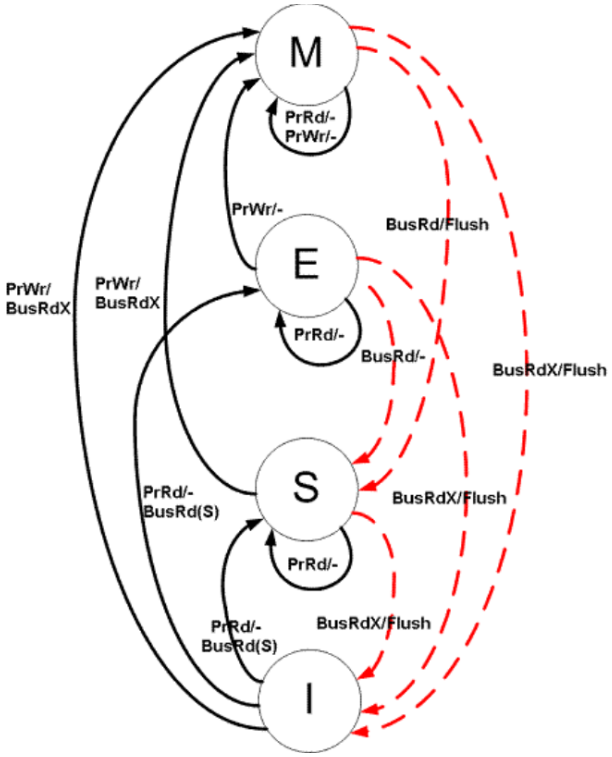

Figure 2.5: The finite-state machine of MESI protocol [23]. Red line indicates the bus transactions and black kine indicates the processor transactions.

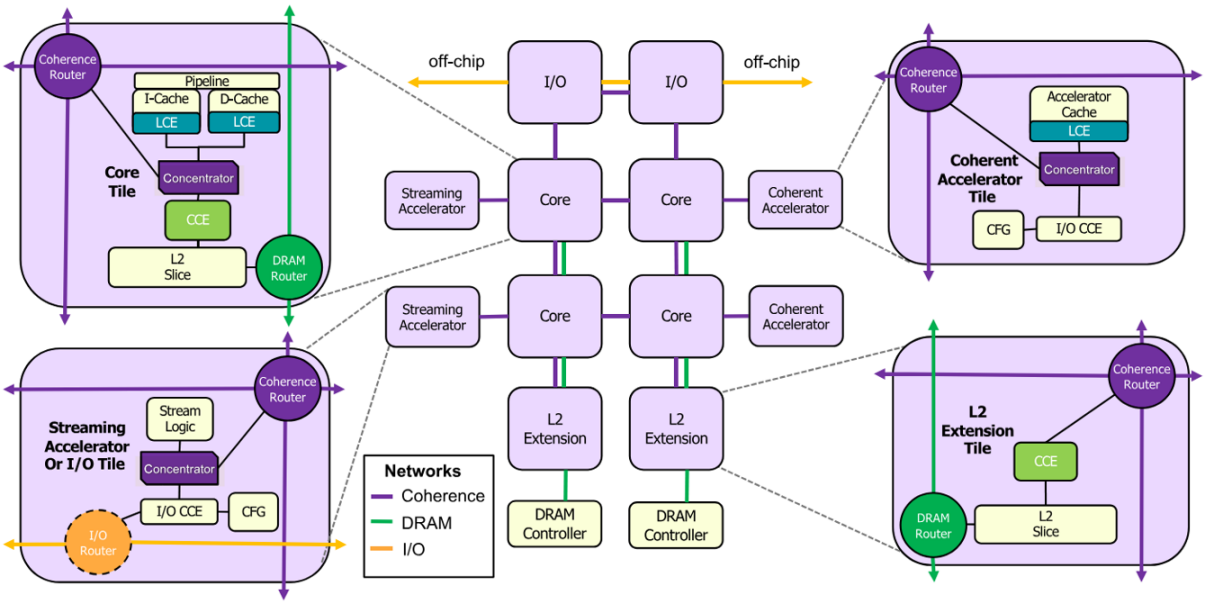

Figure 2.6: The top-level of BlackParrot multi-core platform system [19].

The microarchitecture contains 3 NoC networks:

- Coherence network (BedRock Network): The BedRock network is a cache-coherent interconnect fabric linking all the tiles in a BlackParrot system. It acts as the connection hub for all Local Cache Engines (LCEs) and Central Cache Engines (CCEs) within the system.
- DRAM Network: The DRAM network connects CCEs to devices capable of servicing memory requests, such as DRAM, Flash, or on-chip ROMs.
- IO Network: The I/O network connects a BlackParrot processor to peripherals, including serial ports, PCIe controllers, I/O devices, and also debug interfaces.

2.4.1 Write-through and Write-back cache

The type of cache influences the strategy of the cache storing data, and also is closed to cache coherence system. In write-through caches, data is updated to cache and memory at the same time. The write-through caches will not have the inconsistency problem and often used in no frequency writes to cache case. And for write-back caches, data is only updated when only the cache line is going to be replaced.

2.5 OpenPiton: A Tiled Manycore Framework

OpenPiton is general purpose and multi-threading manycore processor. This scalable tiled manycore framework ranges from one to half a billion cores. Built on a 64-bit architecture, it uses the Ariane core and features a distributed directory-based cache coherence protocol across its networks. Both core and other components can be highly configurable in OpenPiton system. It has been validated in ASIC and multiple Xilinx FPGA prototypes running full-stack Debian Linux. The microarchitecture of OpenPiton is shown in Figure 2.7.

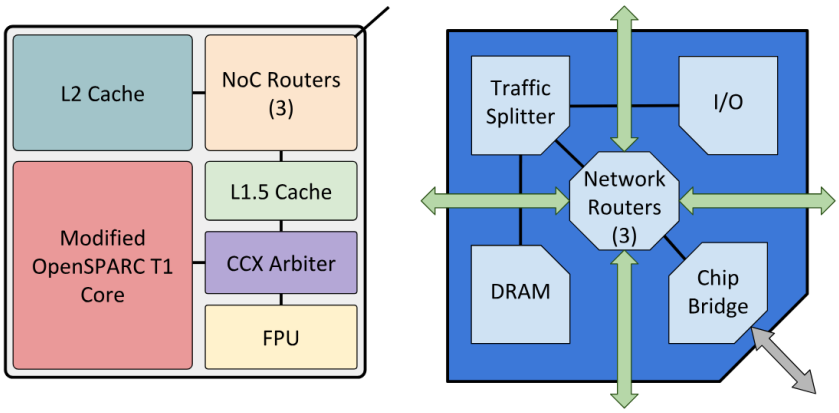

Figure 2.7: The top-level of OpenPiton Manycore Framework, left shows the tile, right shows the chipset.

L1 cache in OpenPiton is a normal write-through cache. The L1.5 cache is a write-back cache and is used as the glue logic between L1 cache to coherence NoC networks. And L2 cache is a distributed write-back cache which is shared by all tiles.

OpenPiton is built on a directory-based MESI protocol. L1.5 caches and L2 caches can communicate coherence messages with each other by NoC, which can be ensured to be deadlock-free.

3 SYSTEM INTEGRATION

The MPSoC system integration is based on the M^3 hardware system. There is a burst need in parallel computing in modern advanced digital system. However, in the original M^3 hardware system, due to Rocket and BOOM RISC-V core lack of the ability to deal with vector instructions, Ara is chosen to integrate into new M^3 hardware system. Compared with other RISC-V processor with vector instruction extension, Ara has an excellent performance of up to 33 DP-GFLOPS. Also, Ara uses a standard protocol AMBA [2], which is a standard handshake protocol developed by ARM, Inc. And it means that Ara is fit into our system and easier to plug in.

As the digital system architecture becomes more and more complex nowadays, architecture of MPSoC system needs to be divided into three levels. On the one hand, it will make the system is easier to understand and on the other hand it will also make digital IP easier to maintain.

First level is the top level of MPSoC system, which is connected by Network-on-chip. The top system's behavior is easy to be observed in this level by decoding the NoC packet. And the data packet is transmitted to outside the system by serial interface like Ethernet and UART. Because of the easy debugging nature of data packets, the software programmers will feel much easier when develop OS applications.

The second level of system architecture is the Ara tile level. And in this level, the Ara is wrapped into a black box. TCU plays the important role in set up communication between Ara tile and other tiles. For example, if Ara needs fetch instructions from DDR4 SDRAM tile, it needs to tell the TCU correct address and proper read access request. Otherwise the request will be rejected.

The lowest level is the Ara level. And in this level, the system is divided into several master and slave IPs. The master and slave IPs build the whole SoC system on a same system bus, which is the AXI general X-Bbar. This is also the most complex level, because hardware developer needs to take consider of every module's function to make sure there will not be any address conflict or data conflict.

3.1 MPSoC System Level

3.1.1 MPSoC system architecture

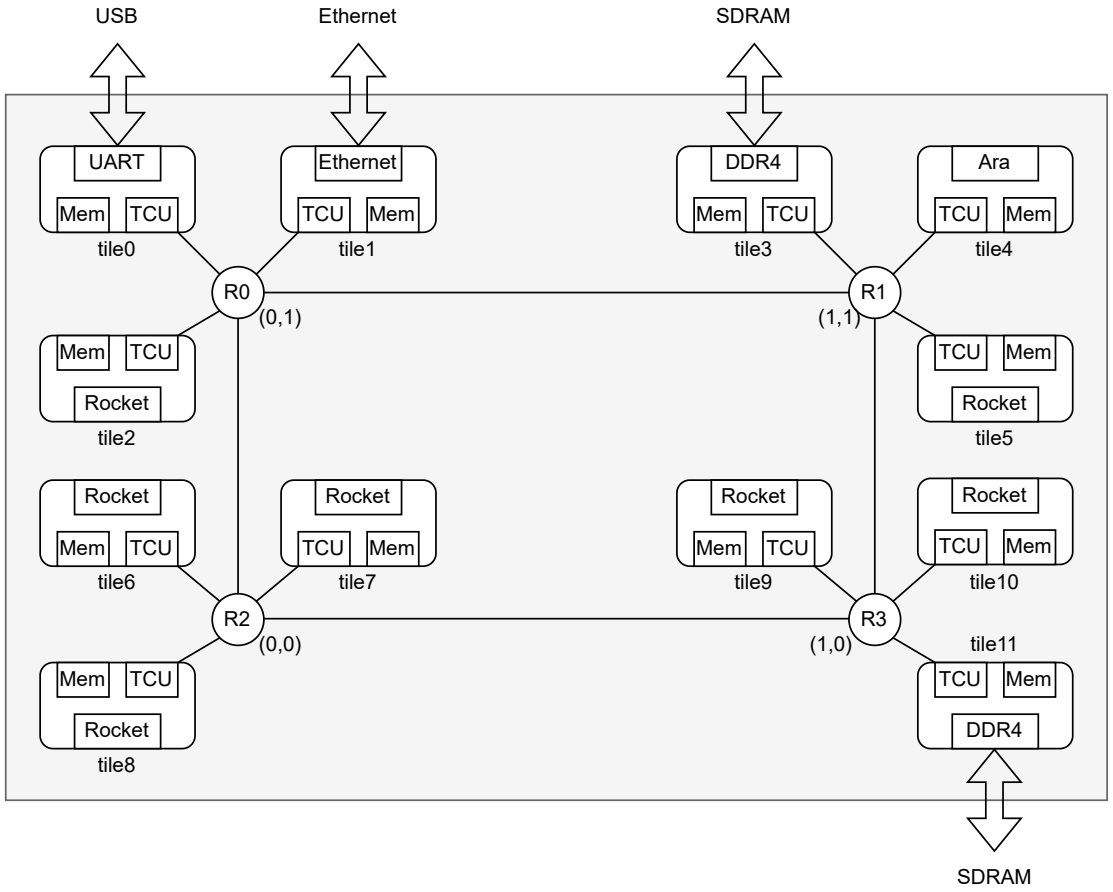

Figure 3.1: The top-level of MPSoC with Ara tile (xyz-coordinate = 1, 1, 1), and other tiles are rocket cores.

The current MPSoC is implemented on a Xilinx VCU118 Virtex UltraScale+ FPGA [22]. The top-level design is a 2×2 star mesh NoC with four routers, and each router connects three tiles. And here are 12 tiles in the MPSoC in total:

- 7x Rocket Processor
- Ara Processor
- 2x DDR4 Memory Interface

- Gbit Ethernet Interface [13]
- UART Interface

In a NoC system, four routers are located at positions (0,0), (0,1), (1,0), and (1,1), respectively. As shown in Figure 3.1, these four routers form a 2×2 star-mesh network. Additionally, routers can be defined by their chip and link numbers in the third dimension (z-coordinate). For instance, the router at (0,0) has chips (z=0, 1, 2) and is connected to the routers at (0,1) and (1,0) with two links (z=3,4). Each chip is associated with a unique module ID (modid), which is composed of the xyz coordinates [21].

The standard routing algorithm for transmitting data packets from the source router to the destination router implements an xy routing method. First, the data packet is transmitted in the x-direction from the source router until it reaches the corresponding x-value of the destination router. Then, the data packet is transmitted in the y-direction until it reaches the destination router. Routing is implemented in each router through a lookup table (LUT). The algorithm compares the destination module ID with the current router's module ID and forwards the data packet to the external link based on the stored LUT values [21].

Data is transported through the NoC by packets. For Each packets, there could be one or several flits. A packet containing one flit can transfer 64-bit data (selected by bsel byte) to the address of the target module (trg-chipid, trg-modid). The source tile is specified by src-chipid and src-modid. The mode defines the type of the packet. By using burst transfer, 128-bit data can be transmitted in one flit. The burst length indicates the number of valid payload flits, which are 16 byte flit. The burst flag indicates a burst packet. All burst transmission units (except the last one) are transmitted with burst=1, while the last transmission unit of the burst packet is indicated by burst=0. The exact structure of standard flit can be referred in the Hardware Document [21].

3.1.2 MPSoC System Sub-modules Description

Rocket Processor

The Rocket core is a RISC-V core generated by the UCB-Chipyard Framework [20], which support RV64 instructions. And the current Rocket supports the following features:

- L1-level 16-kB instruction cache and data cache
- 512-kB L2-level cache
- PTW and TLB contained MMU
- Double-precision FPU

- two external interrupt inputs
- MMIO interface connecting core and TCU registers
- Memory interface connecting core cache and TCU-Bypass
- Debugging JTAG interface
- Instruction Tracer

Ara Processor

The Ara processor consists of Ariane RISC-V CPU and Ara RVV1.0 co-processor, which is detailed described in Chapter 3.2 and Chapter 3.3.

DDR4 Memory Interface

The VCU118 board is equipped with two 16-bit DRAM components, each with a capacity of 512MB. Therefore, a total of 2x 2.5GB of external memory is available. The SDRAM devices (Micron MT40A256M16GE-083E [1]) can reach a maximum data rate of 196 Gbit/s, which are connected via DDR4-2400. Due to the inability to efficiently map the 80-bit data width of all five DRAM components to byte addresses in hardware, only four components are used, resulting in a data width of 64 bits and a total capacity of 2x 2.0GB.

Gbit Ethernet Interface

The VCU118 board includes a self-built Ethernet PHY with an RJ45 connector. This Ethernet interface is used for controlling and debugging purposes by connecting the MPSoC platform with a host PC. The communication can reach a maximum data rate of 1 Gbit/s by using the UDP/IP protocol.

3.2 Ara Tile Level

3.2.1 Ara Tile Architecture

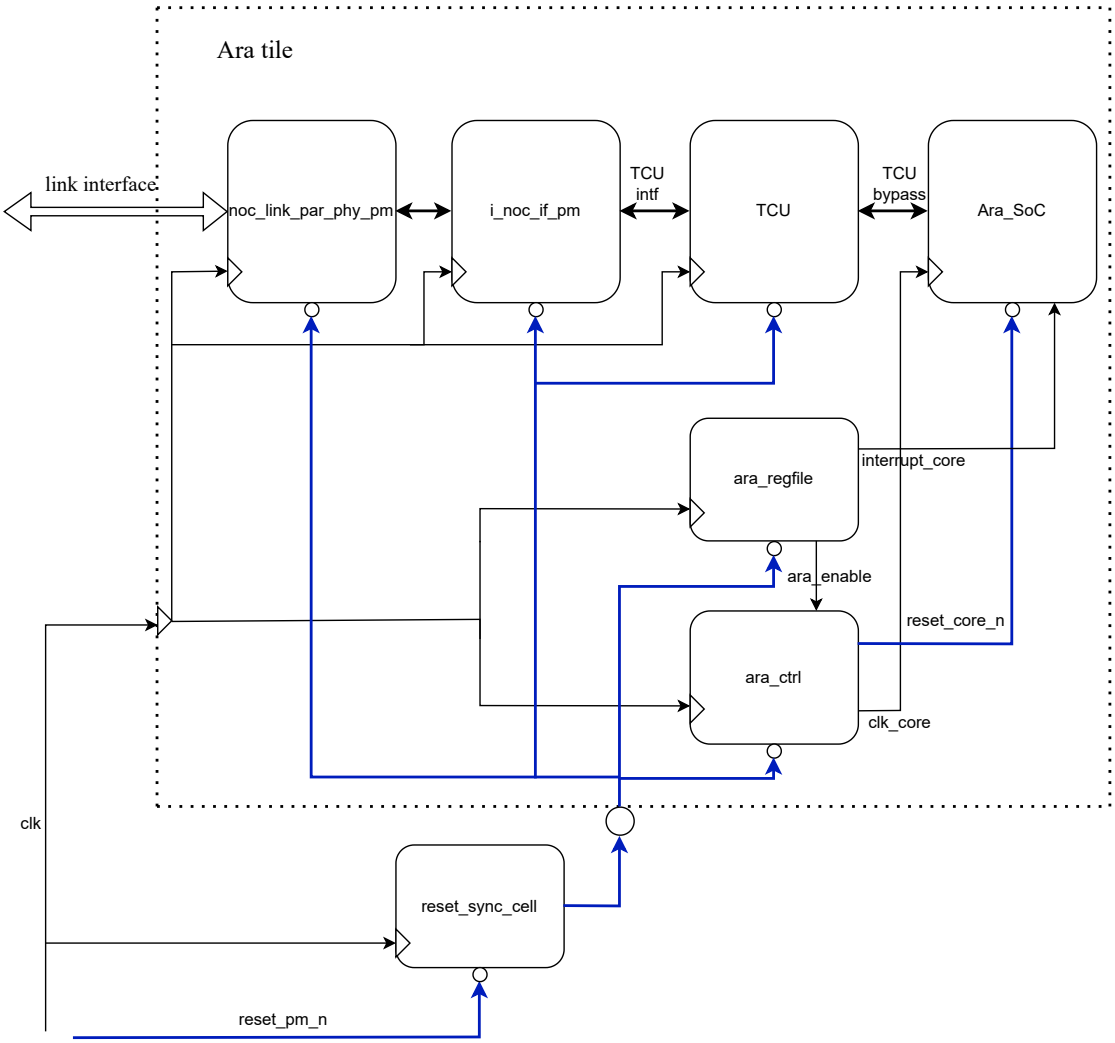

Figure 3.2: The top-level of Ara tile with clock tree and reset tree.

The Ara tile is number four tile in MPSoC, and its module ID (modid) is 25 (xyz-coordinate = 1, 1, 1) The top-level of Ara tile is shown in Figure 3.2, and it has 5 types of common modules and 2 configurable black-box module TCU and Ara:

- Ara Register File

- Ara Control Unit
- NoC Link Physical
- NoC Interface
- 3x Synchronization Unit (tile synchronous reset unit, tile synchronous home-id unit and tile synchronous host-id unit)
- Trust Communication Unit (TCU)
- Ara System-on-a-Chip (SoC)

3.2.2 Ara Tile Sub-modules Description

Ara Register File

Ara regfile is used to configure Ara's behavior. It can be either read or written, there are 3 main configure registers, their function and address is shown in Table 3.1. The main purpose of the Ara regfile is that it will make software programmers take control of the core more easier. The software programmers can determine when to boot up the core and when send it a external interrupt.

Table 3.1: Ara Register File Mapping Table

Register Name	Address	Write Data	Description
REG_ARA_EN	0x00000000	0x0000000(1/0)	Enable the core clk and reset (1) or Enable the core clk and reset (0).
REG_ARA_EXT_INT1	0x00000008	0x0000000(1/0)	Enable the core external INT1 (1) or Disable the core external INT1 (0).
REG_ARA_EXT_INT2	0x00000010	0x0000000(1/0)	Enable the core external INT2 (1) or Disable the core external INT2 (0).

Ara Control Unit

The Ara control unit is used to control Ara's clock and reset. Only when the control unit receives the enable signal from the Ara register file, then it will send out the core clock and reset the core. The unit also has the function of syncing clock and reset. It will make sure that there is no meta-stability when switching input clock or releasing asynchronous reset.

NoC Link Physical

The NoC Link Physical module is the physical of the Ara tile. It receives the header and payload packets from the transmitter (TX) of NoC Interface module, and encodes them into link data to noc router. Also, it receives the link data from noc router and decodes the data into the header and payload packets to receiver (RX) of NoC Interface. The NoC Link Physical module encodes header and payload into link data is shown in Waveform 3.3.

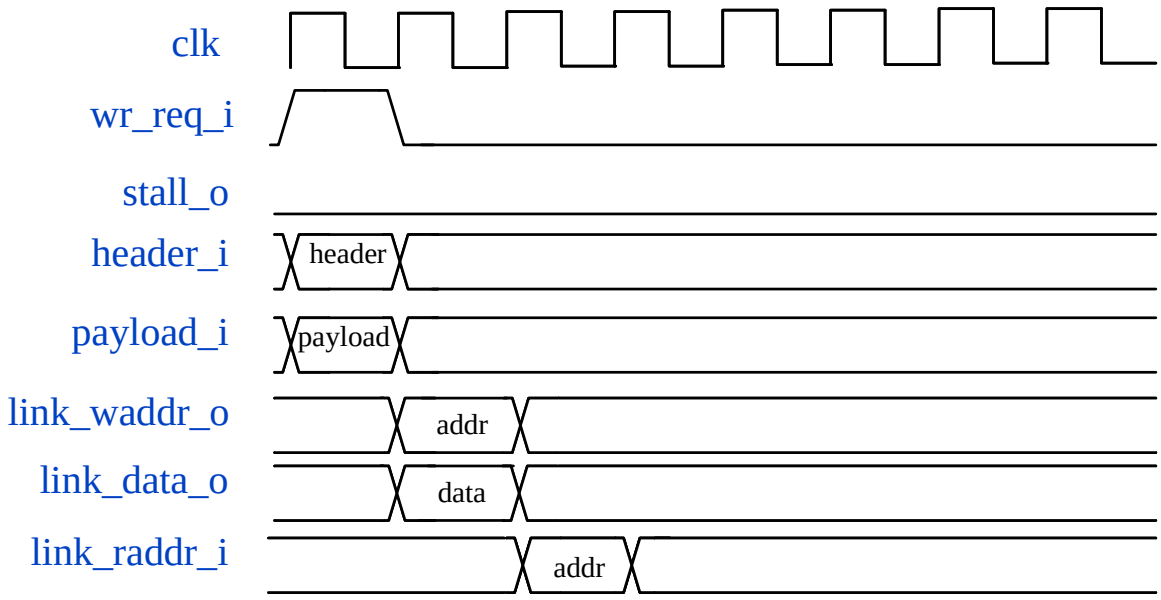

Figure 3.3: The waveform of link physical encodes header and payload into link data.

NoC Interface

The NoC Interface module connects to TCU module. The protocol between the NoC Interface module and TCU is a AMBA-like protocol, which means the module support burst data transfer mode. When TX of TCU sends out a write or read data request and NoC Interface module is ready, the NoC Interface module will encodes the data, address, modid and chipid into header and payload packets. Also, when TX of NoC Interface module sends out a write or read data request and TCU is ready, it will decode the header and payload packets into protocol data to TCU. The NoC Interface module encodes mod data into header and payload is shown in Waveform 3.4.

Synchronization Unit

There are 3 synchronization unit modules in Ara tile. The first unit is to synchronize the reset from the NoC, and the second unit is used to synchronize the host chip ID. The last

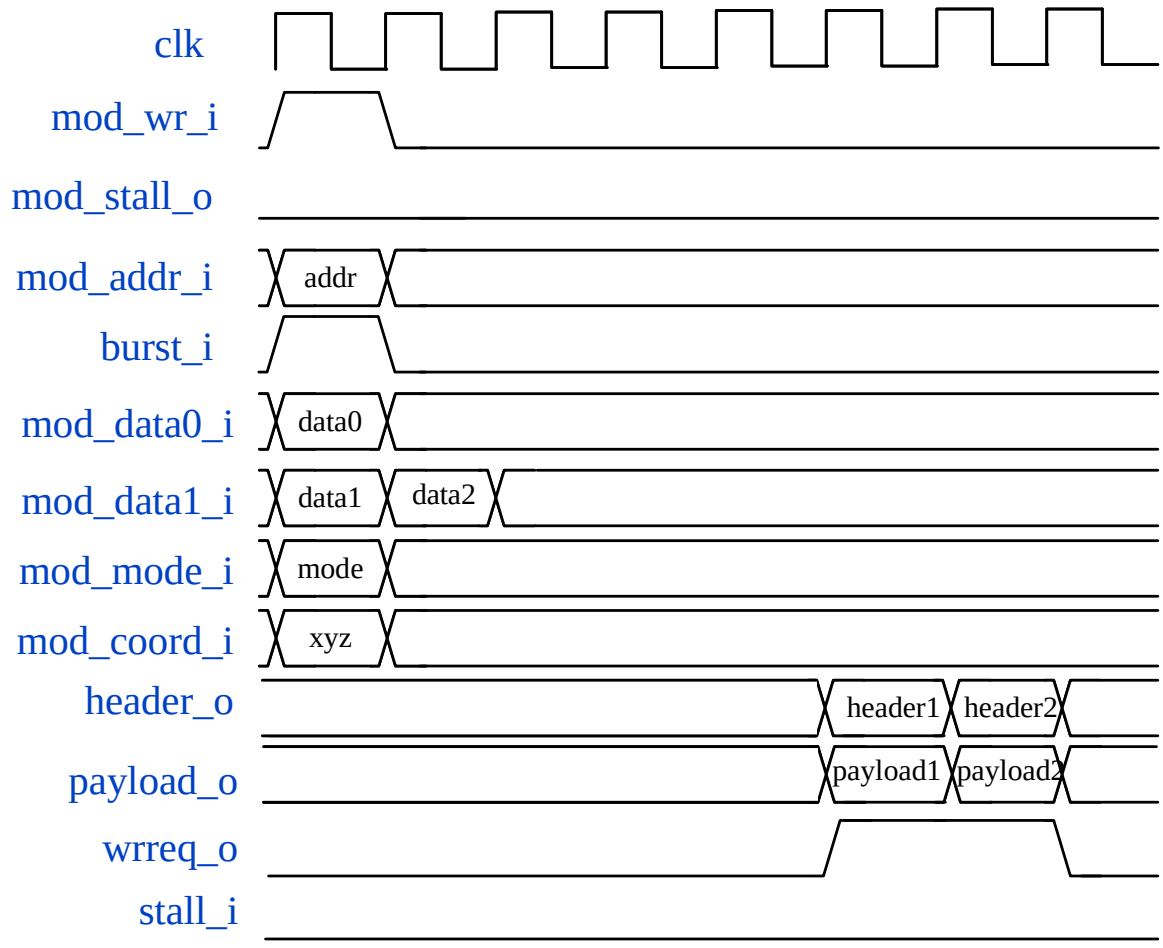

Figure 3.4: The waveform of NoC interface encodes mod data into header and payload.

one is similar to the second synchronization unit, which is use to synchronize home chip ID. All of the units have the function of preventing the Clock Domain Crossing (CDC) errors, which may cause meta-stability in hardware.

Trusted Communication Unit

TCU is treated as a customizable black-box module in Ara tile. The developer can modify the TCU's configuration without knowing the details. And TCU supports the following important features in Ara tile:

- Direct Memory Access: byte-level memory transfer.
- Message Transfer: memory transport with flow control (credit system in NoC).
- Memory Mapping: common interface between the NoC Interface module, Processing Cores, and Local Memory (SPM).

- Extensive Interface: enabling privileged features and extensions by external interface.
- Virtual Addressing: virtualization by a 32-entry Translation Lookaside Buffer (TLB).
- VPE Context Switching: virtual PEs (VPEs) context switching ability.
- Physical Memory Protection: protecting the physical memory by controlling external memory access.
- Debug Log: 2^{16} entries of debug log.
- Debug Printing: the function of distinct debug print.

Initialization of memory transfer and message sending is provided by non-privileged commands. External interfaces are used to enable privileged modes (kernel mode), virtual memory (VM), and context switching (CTXSW) extensions, as well as to execute commands from other Tiles. Support for VM and CTXSW is achieved through privileged interfaces that access the core requests and TLB [21].

The TCU provides a common interface that connects the tile to NoC. And the NoC packets which should be handled by the TCU are sent by the common interface, which is connect to the NoC Interface module. Also, there is a TCU-bypass interface in TCU that forwards the NoC packets which should be handle by other components in the tile. In Ara tile, the TCU-bypass interface is connected with Ara. There is a multiplexer in the TCU that it can distinguish whether to bypass the NoC packet to Ara.

To configure the TCU in Ara, there is some parameters we must consider, which is shown in Table 3.2. The TCU parameters are close related to Ara parameters in Table 3.3. The most important ones are CORE_DMEM_DATA_SIZE, CORE_IMEM_DATA_SIZE and IMEM_DATA_SIZE. Those parameters have close relations with the core data and instruction data width.

Ara

Ara is the CU of the tile and it is treated as a customizable black-box processor in Ara tile. The advantages of wrapping the processor is that the developer can modify the core configuration by needs. Also, the developer do not need to know the core's detail. The parameters of Ara is shown in Table 3.3. The most important parts of these parameters is FPUSupport and NrLanes, the developer can change the FPUSupport according to what kind of Floating-point unit (FPU) and vector instructions they need to fit programs better.

Table 3.2: TCU Parameters

Parameter Name	**Default Value**	**Description**
HOME_MODID	0	The module ID of Ara tile.
CLKFREQ_MHZ	80	The clock frequency of Ara tile, which is presented in MHz.
CORE_DMEM_DATA_SIZE	64	The TCU MMIO regfile interface data width, is the same with ARA_REG_DATA_SIZE parameter in Ara parameter in Ara tile.
CORE_DMEM_ADDR_SIZE	32	The TCU MMIO regfile interface address width, is the same with ARA_REG_ADDR_SIZE parameter in Ara parameter in Ara tile.
CORE_DMEM_BSEL_SIZE	8	The TCU MMIO regfile interface byte-select data width, set as CORE_DMEM_DATA_SIZE/8.
CORE_IMEM_DATA_SIZE	128	The TCU acceptable instruction data width from the core.
CORE_IMEM_ADDR_SIZE	32	The TCU acceptable instruction address width from the core.
CORE_IMEM_BSEL_SIZE	16	The TCU acceptable instruction byte-select width, set as CORE_IMEM_DATA_SIZE/8.
DMEM_DATA_SIZE	64	The data SPM data width.
DMEM_ADDR_SIZE	17	The data SPM address width.
DMEM_BSEL_SIZE	8	The data SPM byte-select data width, set as DMEM_DATA_SIZE/8.
IMEM_DATA_SIZE	128	The instruction SPM data width, set as the same with ARA_MEM_DATA_SIZE in Ara tile.
IMEM_ADDR_SIZE	17	The instruction SPM address width.
IMEM_BSEL_SIZE	16	The instruction SPM byte-select data width, set as IMEM_DATA_SIZE/8.

Table 3.3: Ara Parameters

Parameter Name	**Default Value**	**Description**
FPUSupport	FPUSupportNone	The type of vector instruction of Floating Point Unit of Ara co-processor supporting, it could be set as FPUSupportNone (000), FPUSupportHalf (001), FPUSupportSingle (010), FPUSupportHalfSingle (011), FPUSupportDouble (100), FPUSupportSingleDouble (110) and FPUSupportHalfSingleDouble (111).
NrLanes	4	The number of Lanes of Ara co-processor, which is related to Ara co-processor's arithmetic intensity.
AxiDataWidth	128	The AXI bus data width of Ara, which equals to $NrLanes \times 32$ bits.
AxiAddrWidth	64	The AXI bus address width of Ara, usually set as 64 bits.
AxiUserWidth	1	The AXI bus user width of Ara, usually set as 1 bits.
AxiIdWidth	5	The AXI bus id width of Ara, usually set as 5 bits.
ARA_MEM_DATA_SIZE	128	The Ara and TCU memory interface data width, set as 128 bit.
ARA_MEM_BSEL_SIZE	16	The Ara and TCU memory interface byte-select data width, set as ARA_MEM_DATA_SIZE/8.
ARA_MEM_ADDR_SIZE	64	The Ara and TCU memory interface address width, usually set as 64 bits.
ARA_REG_DATA_SIZE	64	The Ara and TCU MMIO regfile interface data width.
ARA_REG_BSEL_SIZE	8	The Ara and TCU MMIO regfile interface byte-select data width, set as ARA_REG_DATA_SIZE/8.
ARA_REG_ADDR_SIZE	32	The Ara and TCU MMIO regfile interface address width.
ARA_USE_LOCAL_MEM	0	If the Ara uses local memory (1) or DDR4 memory (0).

3.3 Ara Level

3.3.1 Ara Architecture

The Ara is the computing unit of Ara tile. It supports RISC-V 64-bit IMC and 1.0 RVV extension instruction. There are two feasible types of Ara architectures, which are architecture A shown in Figure 3.5 and architecture B in Figure 3.6. And Ara main consists of 3 master modules, 5 slave modules, 4 common modules (converter and multiplexer) and system bus:

- AXI-Xbar (System Bus)
- Ariane RISC-V IMC Processor (Master Module)
- Ara 1.0 RVV Co-processor (Master Module)
- TCU to AXI(Master Module)
- BOOTROM (Slave Module)
- AXI to NoC (Slave Module)
- AXI to Reg (Slave Module)
- AXI to Mem (Slave Module)
- Control Register (Slave Module)
- AXI Data Width Converter (Common Module)
- AXI to AXI-Lite Converter (Common Module)
- AXI Multiplexer (Common Module)
- Invalidation Filter (Common Module)

All the modules are connected by AXI-4 protocol, which can simplify the development of the design SoC with multiprocessors and large number of slave modules (controllers and peripherals). AXI-4 protocol can provide efficient IP reuse, flexibility, compatibility and well support for IPs. Also, the bandwidth and latency of AXI4 is also a big advantage compared with other system buses [14]. The AXI4 uses a valid-ready protocol, which can ensures the stability of data transfer and low latency. And it also supports important data channel features like burst transfer and interleaving, which make sure high bus performance.

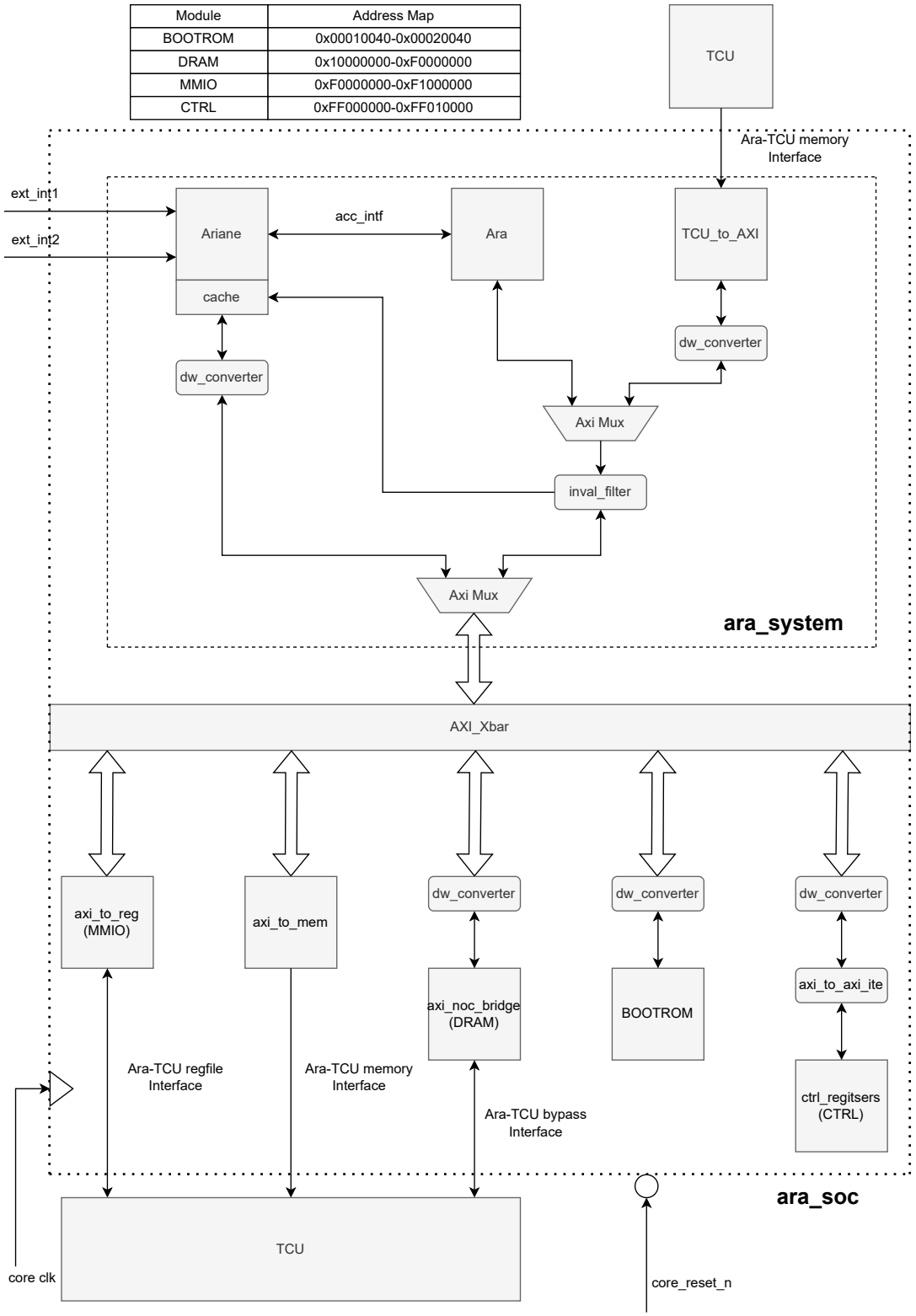

Figure 3.5: The top-level of Ara architecture A with one actual Master in AXI-Bar.

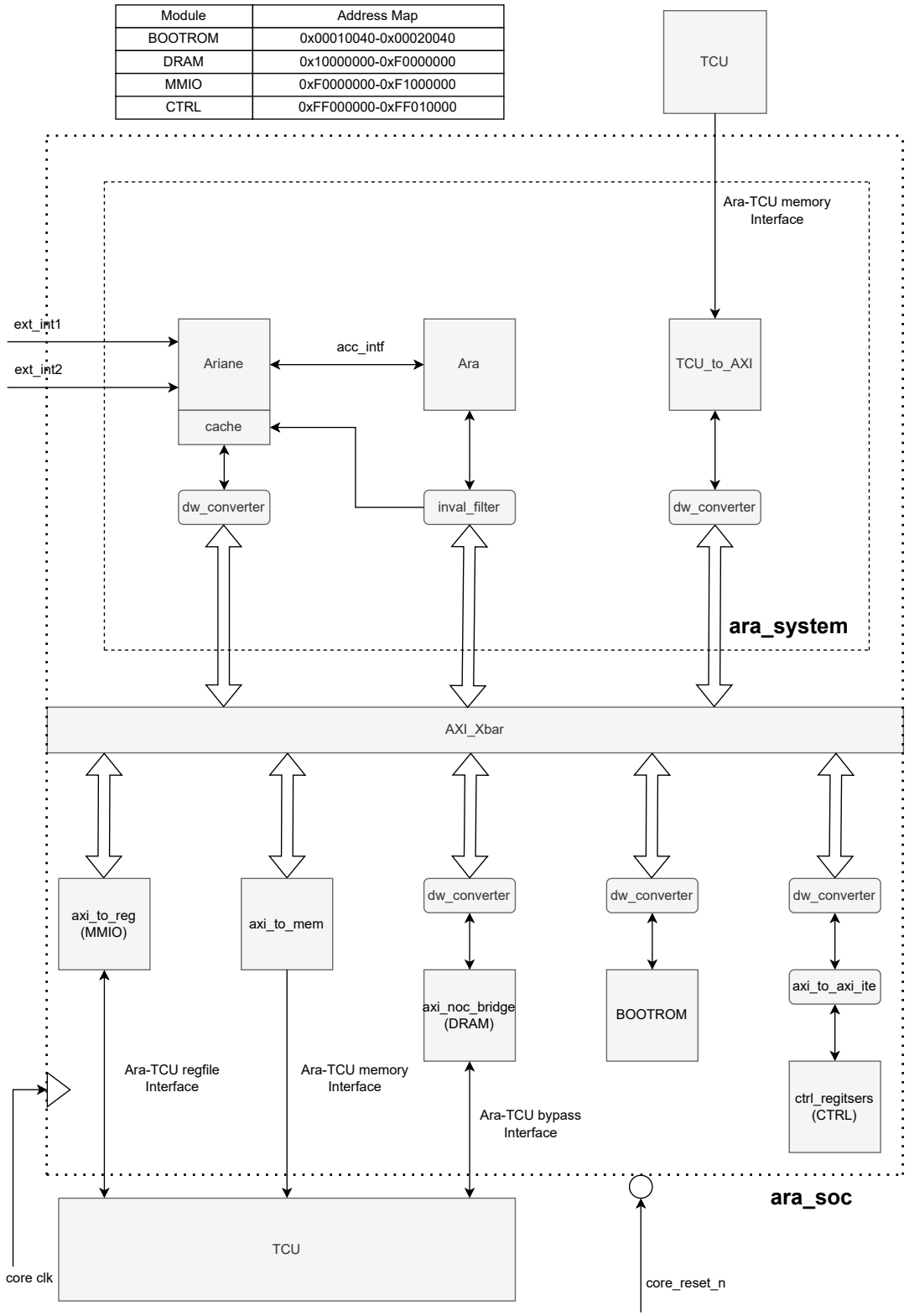

Figure 3.6: The top-level of Ara architecture B with three actual Master in AXI-XBar.

For each architecture A and B, their AXI-Slave modules have the same configuration. The main difference is the Ara system, which is the AXI-Master modules. In Architecture A, only one real master module is connected to the system bus (AXI-Xbar). The three master modules (Ariane, Ara, TCU) AXI-channel are selected by two AXI-multiplexer modules. Because of the Round-Robin Arbiter in AXI multiplexer, the three modules have the equal priority. By this way, three modules can access the system bus in turns and will not be blocked. For architecture B, master modules are directly connected to system bus. This means three AXI-channels in X-BAR, and the performance and bandwidth of X-BAR will be enhanced in this way. However, this would increase extra area for AXI-demux module in X-BAR. Compared with AXI-multiplexer, the area cost of AXI-demux and AXI-decerr is much more considerable, which can referred in Figure 3.7. Also, by the function limit, the three master modules will more likely to access the same slave module at the same time. And the requests from master modules are also needed to be selected in the same AXI-multiplexer. By those consideration, we choose the architecture A of final Ara architecture.

3.3.2 Ara Sub-modules Description

AXI-XBar

The AXI-Xbar is composed of AXI-demux, AXI-mux and AXI-decerr. It is a fully connected crossbar switch, which means that each master module connected to the crossbar slave port has a direct line connection to all slave modules connected to the crossbar master port. The number of slave ports and master ports of the crossbar can be configurable. The master port ID width is wider than the slave port ID width. The internal multiplexer uses additional ID bits to route responses. The ID width of the master port is AxiSlaveIdWidth $+ \log_2(\text{Number of slaves})$. The Figure 3.7 illuminates the configure of AXI-Xbar [7]. The area cost of AXI-XBar is mainly on AXI-demux and AXI-decerr module. AXI-mux costs relatively small compared with the other two modules.

The AXI-Xbar is routed by address mapping rules. The address map is shared by all master ports. The address map contains an arbitrary number of rules (but at least one). Each rule maps an address range to a master port. When there is not matching rule in the address map for the transaction from master port, the transaction will routed to AXI-decerr module. The AXI-decerr module will absorb the transaction and respond with errors.

Also there is some parameters needs to be set in AXI-Xbar module, which are shown in Table 3.4. The LatencyMode parameter is needed to be take cared, because usually the

system critical path is on the AXI-Xbar AXI channel. We can set as CUT_ALL_PORTS to increase the system frequency.

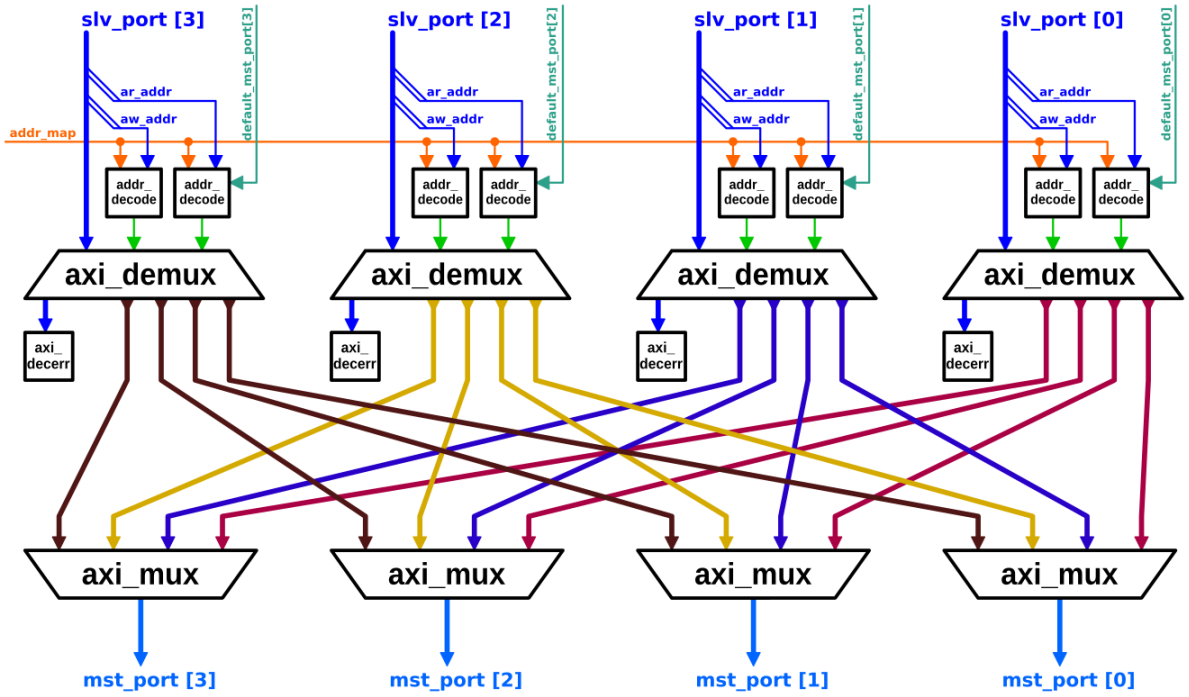

Figure 3.7: The block-diagram of the AXI-Xbar in Ara, which has a configurable number of master and slave ports [7].

Table 3.4: AXI-Xbar Parameters

Parameter Name	**Default Value**	**Description**
NoSlvPorts	5	AXI slave ports actual number of the crossbar.
NoMstPorts	1	AXI master ports actual number of the crossbar.
LatencyMode	CUT_ALL_PORTS	The latency mode (pipelined or not) of every AXI channel in crossbar, usually is set as NO_LATENCY or CUT_ALL_PORTS.
AxiIdWidthSlvPorts	5	Slave ports AXI ID width.
AxiIdUsedSlvPorts	5	Usually the same with AxiIdWidthSlvPorts.
AxiAddrWidth	64	The AXI address width.
AxiDataWidth	64	The AXI data width of Ariane, which is different from Ara.
NoAddrRules	5	The number of slave ports address mapping rules, usually the same with NoSlvPorts.

Ariane RISC-V IMC Processor

The Ariane RISC-V IMC Processor supports RV64IAMC instructions. It is a 6-stage pipelined CPU, its architecture is shown in Figure 3.8. The processor has the following import features:

- In-order issue, out-of-order write-back and in-order instructions commit.
- L1-level 16-kB instruction cache and 32-kB data cache
- PTW and TLB contained MMU.
- Branch Prediction with 2-Depth RAS, 32-Entries BTB and 128-Entries BHT.
- Two external interrupt inputs and one software interrupt input.
- Vector instructions accelerate interface with Ara co-processor.
- Instruction Tracer.
- Linux Capable.

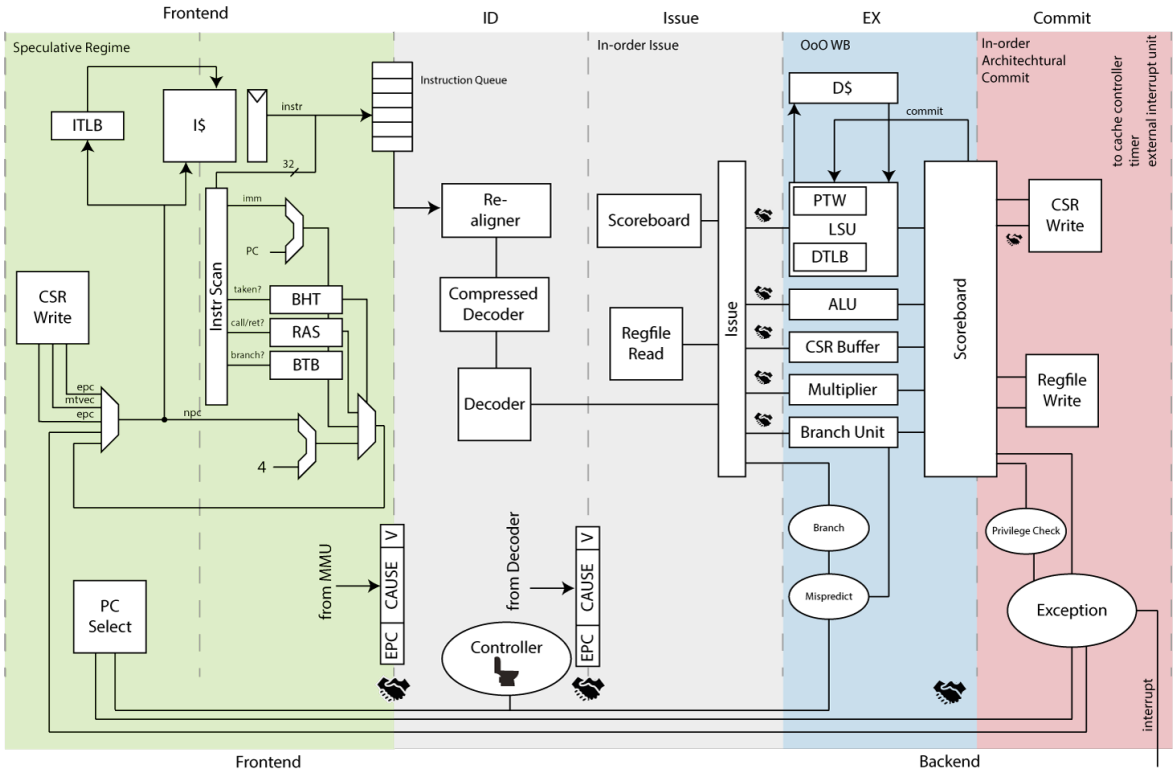

Figure 3.8: The architecture of 6-stage pipelined in-order RISC-V CPU Ariane [26].

Also, the Ariane's configuration can be set within some parameters, which is shown in Table 3.5. The CachedRegionAddrBase and CachedRegionLength need to be care of, because the two parameters affect the AXI-AW and AXI-AW channel. And the Ariane core will boot from the base of BootROMBase, which is 0x00010040. Also, some important parameters are in 'ariane_pkg.sv' file, which can configure the supporting extensions on Ariane RISC-V. And it can also config the instruction and data cache of Ariane.

Table 3.5: Ariane Parameters

Parameter Name	**Default Value**	**Description**
DramBase	0x10000000	The base address of DRAM.
DramLength	0xE0000000	The length of DRAM.
BootROMBase	0x00010040	The base address of BOOTROM.
BootROMLength	0x00010000	The length address of BOOTROM.
DmBaseAddress	0x00000000	The base address of Debug Module.
DmLength	0x00001000	The length of Debug Module.
CachedRegionAddrBase	0x00010000	The address base of Ariane cache could cache, which will affect AWCACHE and ARCACHE signal.
CachedRegionLength	0x80000000	The length of Ariane cache could cache, which will affect AWCACHE and ARCACHE signal.
RVV/RVF/RVD/RVA	1	If enable the V/F/D/A extension on Ariane. Set 0 as disable and 1 as enable.
CONFIG_L1I(D)_SIZE	16*1024(32*1024)	The L1 I(D)-CACHE size.
I(D)CACHE_SET_ASSOC	4(8)	The associativity of L1 I(D)-CACHE, must set between 4 to 64.
I(D)CACHE_INDEX_WIDTH	12	The index width of L1 I(D)-CACHE, which is set as $\log_2\left(\frac{\text{CONFIG\_L1I(D)\_SIZE}}{\text{I(D)CACHE\_SET\_ASSOC}}\right)$.
I(D)CACHE_TAG_WIDTH	44	The tag width of L1 I(D)-CACHE, which is set as PLEN - I(D)CACHE_TAG_WIDTH.
I(D)CACHE_LINE_WIDTH	256(128)	The I(D)-CACHE line width.

Ara 1.0 RVV Co-processor

The Ara 1.0 RVV is the co-processor of Ariane RISC-V, which deals with vector instructions. When Ariane decodes vector instructions, it will send the accelerate request to Ara by vector accelerate interface. Then Ara will deal with the vector instructions and send back the result by AXI-4 interface. One thing needs to be take care of is when Ara sends back the result by AXI-4, it will also invalidate the Ariane cache by Invalidation Filter. It has configurable number of Lanes and vector instruction length.

The Ara Arithmetic intensity is decided by the number of Lane. The researchers Matheus Cavalcante et al.in ETHz did some benchmarks about matrix multiplication and convolutions, the results are shown in Figure 3.9. The left one shows the performance of matrix multiplication $C = A \times B + C$, with different number of lanes. The right one shows the Convolutions performance results for three considered benchmarks AXPY (vector length of 256), MATMUL (between matrices of size 256×256) and CONV (use GoogLeNet's sizes) [11].

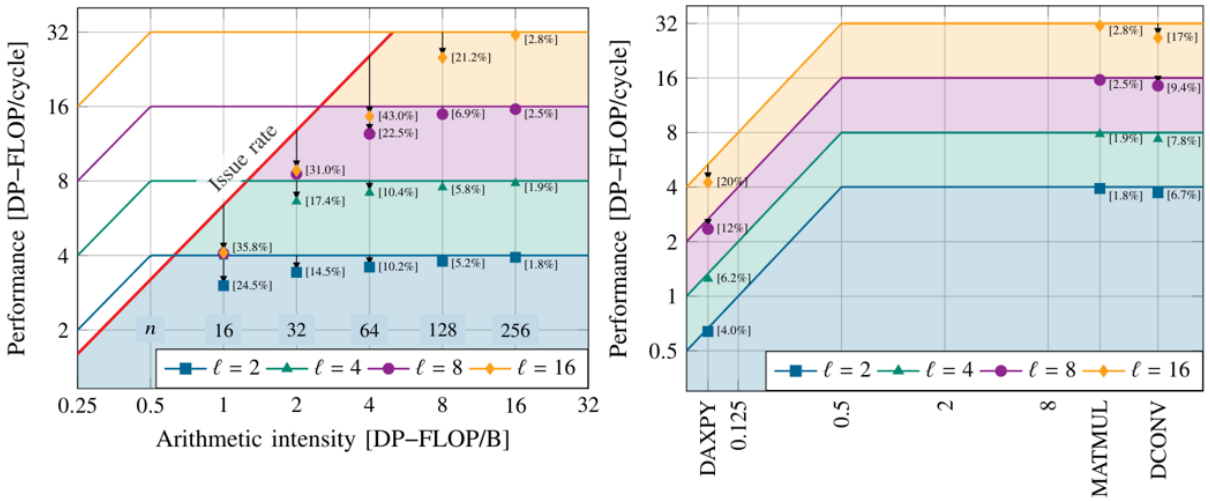

Figure 3.9: The arithmetic intensity of Ara co-processor in different benchmarks (matrix multiplication-left and CONV-right) [11]

.

To configure Ara, there are some parameters need to be set, which are shown in Table 3.6. Also, some parameters are in 'ara_pkg.sv' file, which can config the ara vector register length and vector elements length.

Table 3.6: Ara Parameters

Parameter Name	Default Value	Description
NrLanes	4	The number of Lanes in Ara.
FPUSupport	FPUSupportNone	The type of vector instruction of Float- ing Point Unit of Ara co-processor sup- porting, which can be also referred in Table 3.3.
AxiDataWidth	128	The AXI bus data width of Ara, which equals to $NrLanes \times 32$ bits.
AxiAddrWidth	64	The AXI bus address width of Ara, usually set as 64 bit.
ELEN	64	The vector elements length, up to 64 bits.
ELENB	8	Maximum size of a single vector element, in bytes.
VLEN	512	The vector register length, up to 64 bits.
VLENB	64	Maximum size of vector register length, in bytes.
MAXVL	512	Maximum vector length (in elements), which is set as the same with VLEN as defalut.
NrVInsn	8	Number of vector instructions that can run in parallel.

TCU to AXI

The TCU to AXI module is a bridge connects TCU-master with system bus. The TCU master can write to slave modules by the bridge through the memory interface. The parameters are shown in Table 3.7:

Table 3.7: TCU to AXI Parameters

Parameter Name	Default Value	Description
AXI_ID_WIDTH	3	The AXI ID width of TCU. Due to AXI-mux, the ID width of TCU is AxiSOCIdWidth (5) $-$ $\log_2$(NrMaster (3)).
AXI_ADDR_WIDTH	32	The AXI address width of TCU
AXI_DATA_WIDTH	64	The AXI data width of TCU.

BOOTROM

The BOOTROM module contains the boot instructions of Ariane. The data in BOOTROM cannot be modified, and it is read-only data. Ariane actually starts to fetch the instruc-

tions from address 0x00010040, and the assemble codes of BOOTROM mainly operates the CSR regfile in Ariane. It will enable the Machine Interrupt Enable (MIE) in mstatus and set the Machine Trap-Vector Base-Address Register (mtvec) as 0x10000000, which is the DRAM base address.

Also it set the Machine Software Interrupt Enbale (MSIE) as 1 in mie register. When there is a machine software interrupt, the PC will jump to address 0x10000000.

AXI to Mem/Reg

The AXI to Mem/Reg module is a bridge connects system bus with TCU-slave. The AXI-Xbar can write to the TCU-slave by the bridge through the memory/register interface. And the module supports burst write or read features. The parameters is shown in the Table 3.8:

Table 3.8: AXI to Mem/Reg bridge Parameters

Parameter Name	Default Value	Description
AXI_ID_WIDTH	5	The AXI ID width of AXI to Mem/Reg bridge.
AXI_ADDR_WIDTH	32	The AXI address width of AXI to Mem/Reg bridge
AXI_DATA_WIDTH	128	The AXI data width of AXI to Mem/Reg bridge.

AXI to NoC

The AXI to NoC bridge connects the system bus with TCU-bypass interface. The master modules can write or read data from DRAM through the bridge. Also, the bridge supports burst write or read features. The block diagram is shown in Figure 3.10.

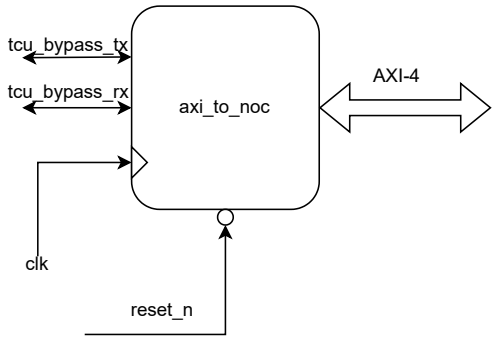

Figure 3.10: The block diagram of axi_to_noc bridge.

Control Register

The control regfile in Ara connects to system bus as a slave module by AXI-Lite interface. It is mainly used to modify the DRAM base and end address. However, it is not used in Ara for current configuration. The block diagram is shown in Figure 3.11.

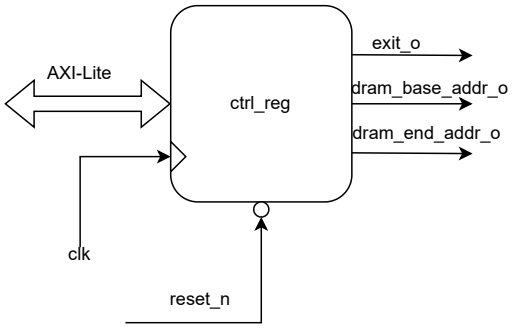

Figure 3.11: The block diagram of ctrl_reg.

AXI Converter

The AXI data width converter connects two AXI data channels with different data width or protocol. For data width converter, it must be the same AXI-protocol. By the direction from master port to slave port, it could be a up-size converter or down-size converter. AS for AXI-Lite converter, it could convert AXI-4 master port to AXI-Lite slave port. The block diagram is shown in Figure 3.12.

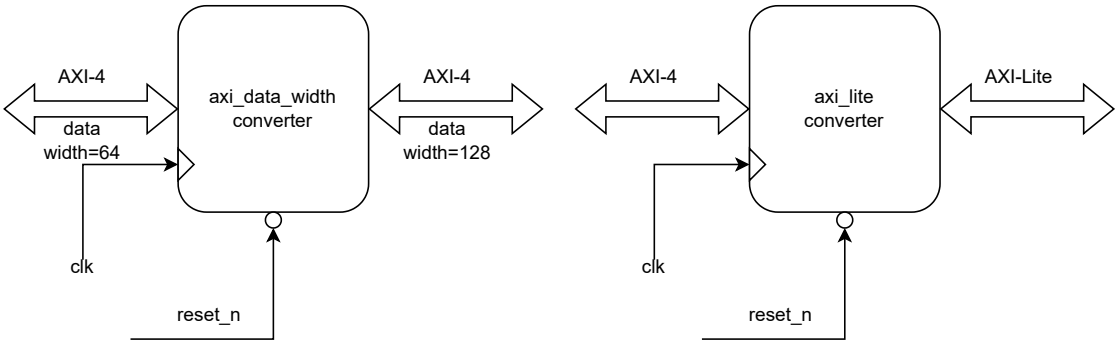

Figure 3.12: The block diagram of axi_converter.

AXI Multiplexer

The AXI-mux is a multiplexer specially used for AXI-4 protocol based data channel. The number of slave ports can be configured to choose one master port. The choose algorithm is based on Round-Robin algorithm and every slave ports have the equal

Table 3.9: AXI Multiplexer Parameters

Parameter Name	Default Value	Description
AXI_ID_WIDTH	3	The AXI ID width of AXI multiplexer slave ports.
SpillAr	1	Pipeline the AXI-AR channel (1) or not (0).
SpillR	1	Pipeline the AXI-R channel (1) or not (0).
SpillAw	1	Pipeline the AXI-AW channel (1) or not (0).
SpillW	1	Pipeline the AXI-W channel (1) or not (0).
SpillB	1	Pipeline the AXI-B channel (1) or not (0).

priority to write to or read from master port. Also, the AXI-mux could cut the critical path for the data channel. The parameters is shown in Table 3.9.

Invalidation Filter

The invalidation filter is used to send out invalid signal the cache of Ariane core when Ara co-processor wants to write data to memory. Because if Ara co-processor writes data to memory, the data in Ariane's cache is out-of-data. For convince, the invalidation filter will invalid the cache of Ariane core.

4 SYSTEM ANALYSIS

The system analysis is based on Xilinx VCU118 Virtex UltraScale+ FPGA platform, which provides 2,586K system logic cells, 6840 DSP slices, 345.9Mb memory and 832 I/Os [22].

According the complexity of system, we focus on two levels: Ara and the M^3 system. The first level is mainly focus on the configuration of Ara, especially on the Ara co-processor. The second level compares different types of RISC-V cores, about their area and performance. These data can help with the developers choose the matching core and the configuration of Ara according to their applications.

In ASICs, the area is usually measured by cell number. However, the area of a FPGA system is usually measured by LUTs, Flip-Flops(FFs), RAMS and DSP blocks. Also, the processors uses specific benchmarks as performance tests. We count the cycles that processors finish the benchmarks as the performance measurements.

4.1 Ara Area Cost Analysis

The area cost of Ara is depending on several main factors, including the number of Lanes of Ara co-processor, VLEN and the type of FPU Support. All the Ara is synthesised in FPGA with a fixed frequency 80 MHz, and by different configurations. And last we compare the different types of RISC-V core area cost.

The Table 4.1 and Figure 4.1 illuminates the sub-module area costs when $l = 4$, VLEN = 512 , and FPU supporting half, single, double data type. From the data we can conclude that the main area cost of Ara is from the Ara co-processor, which approximately cost 75% area of Ara.

Table 4.1: Ara Area Cost of Sub-module when $l = 4$, VLEN = 512 and FPUSupportHalfSingleDouble

Module Name	CLB LUTs	CLB FFs	BlockRAM	DSP blcoks
Ara	316583	94680	120	236
X-Bar	1431	1252	0	0
Ariane RISC-V	40624	24228	44	28
Ara co-processor	245031	55487	70	196
TCU-top	13753	6582	4	12
axi4_to_noc	299	90	0	0
axi4_to_reg	275	147	0	0
axi4_to_mem	263	147	0	0
BOOTROM	46	4	1	0
ctrl_register	91	200	0	0

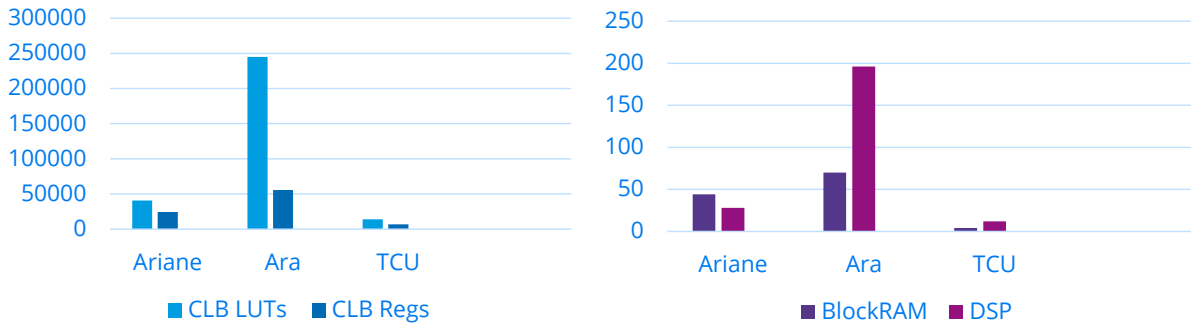

Figure 4.1: Area Cost of Ara three master modules.

4.1.1 Number of Lanes

The Ara's area is synthesised with different number of lanes l, however with fix core frequency 80 MHz, 512 bit width VLEN, and FPU supporting half, single, double data type. The area cost of Ara within different number of lanes l is shown in Table 4.3 and Figure 4.3. By area cost data, we can conclude that when we increase one number of lanes l, the area cost will increase about 41500 LUTs, 12500 FFs, 16 BlockRAMs and 49 DSPs. Because the number of lanes l will affect the arithmetic intensity of Ara co-processor, and the FPU is relatively complex and cost a lot LUT and FF to synthesis. Also, due to the vector registers and multipliers in lane, the BlockRAM and DSPs will also increase a considerable number.

Table 4.2: Area Cost of Ara for different lanes l when VLEN = 512 and FPUSupportHalfSingleDouble

Number of Lanes l	**CLB LUTs**	**CLB FFs**	**BlockRAM**	**DSP blcoks**
2	186982	70336	84	138
4	316583	94680	116	236
8	684057	151861	180	432
16	1326259	261637	308	824

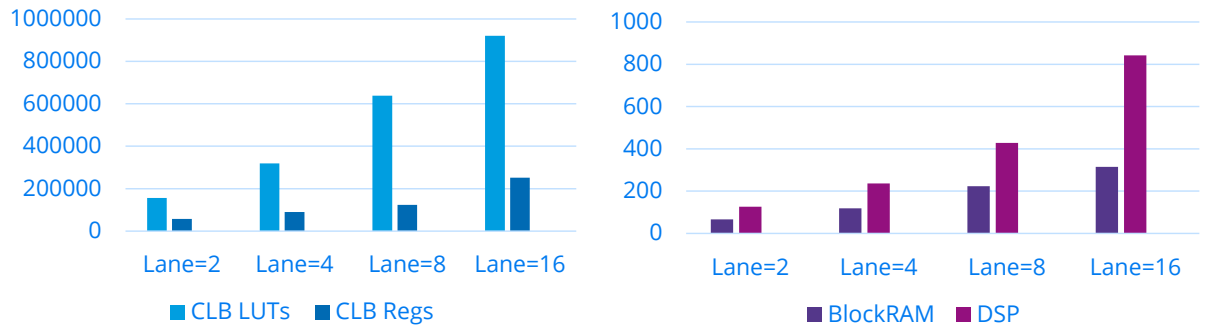

Figure 4.2: Area Cost of Ara for different lanes l when VLEN = 512 and FPUSupportHalfSingleDouble.

4.1.2 VLEN

The Ara's area is synthesised with different VLEN, however with fix core frequency 80 MHz, lane number l=4, and FPU supporting half, single, double data type. The area cost of Ara within different VLEN is shown in Table 4.3 and Figure 4.3. We can conclude from the data that with the increase of VLEN, the number of Flip-Flops and BlockRAM will increase. It is because that with the increasing of VLEN, Ara needs more registers and BlockRAM to store vector instructions and data. However, the area cost does not increase so much, the possible reason might be the BlockRAM in FPGA is relatively large to adapt to the increase of VLEN. However, the cost of DSPs will not increase, so increase VLEN will not increase compute complexity.

Table 4.3: Area Cost of Ara for different VLEN when Lane Number l=4 and FPUSupport-HalfSingleDouble

VLEN	CLB LUTs	CLB FFs	BlockRAM	DSP blcoks
64	319753	91325	106	236
128	317654	93524	108	236
256	315648	94238	112	236
512	316583	94680	116	236

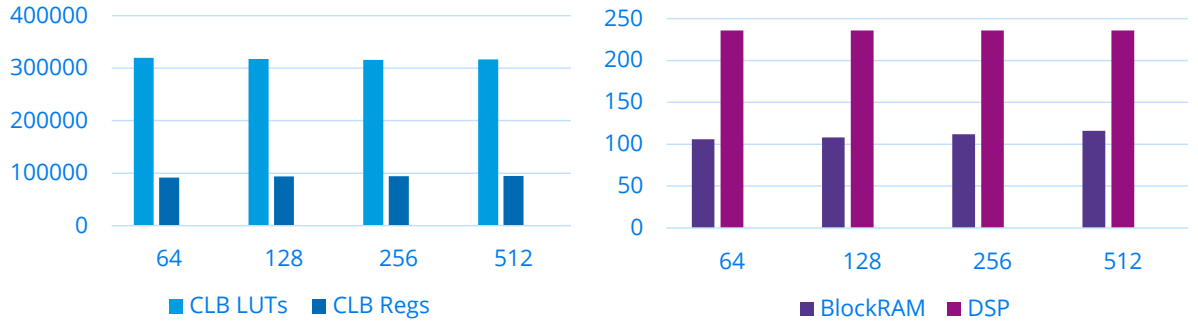

Figure 4.3: Area Cost of Ara for different VLEN when Lane Number l=4 and FPUSupportHalfSingleDouble.

4.1.3 FPU Support

The Ara's area is synthesised with different types of FPU support, however with fix core frequency 80 MHz, lane number l=4 and 512 bit width VLEN. The FPU support means the type of the data precision of FPU supporting like half, single and double. The data of different type FPU support is shown in Table 4.4, Figure 4.4 and Figure 4.5. By analysing the data, we can have a conclusion that when FPU Support is none FPU supporting, which means that none data type of FPU supporting. So the area cost is much smaller compared with other types of FPU Support. Also, FPU supporting half, single, double data type costs most area, it is because this type supports Half, Single and Double vector data type. For DSP cost, Double data type is relatively large than Half and Single data type. This is because Double data type have precision than Single and Half data type, it needs more DSPs to do calculation. However, the Block-RAM cost keep the same except none FPU support. The data shows that the storage elements have nothing to do with the FPU Support.

Table 4.4: Area Cost of Ara for different types of FPUSupport when Lane Number l=4 and VLEN=512

FPUSupport	**CLB LUTs**	**CLB FFs**	**BlockRAM**	**DSP blcoks**
FPUSupportNone	71552	39193	52	40
FPUSupportHalf	283187	83897	116	184
FPUSupportSingle	265495	81621	116	184
FPUSupportHalfSingle	297639	88901	116	200
FPUSupportDouble	256695	80573	116	204
FPUSupportSingleDouble	283102	87333	116	220
FPUSupportHalfSingleDouble	316583	94680	116	236

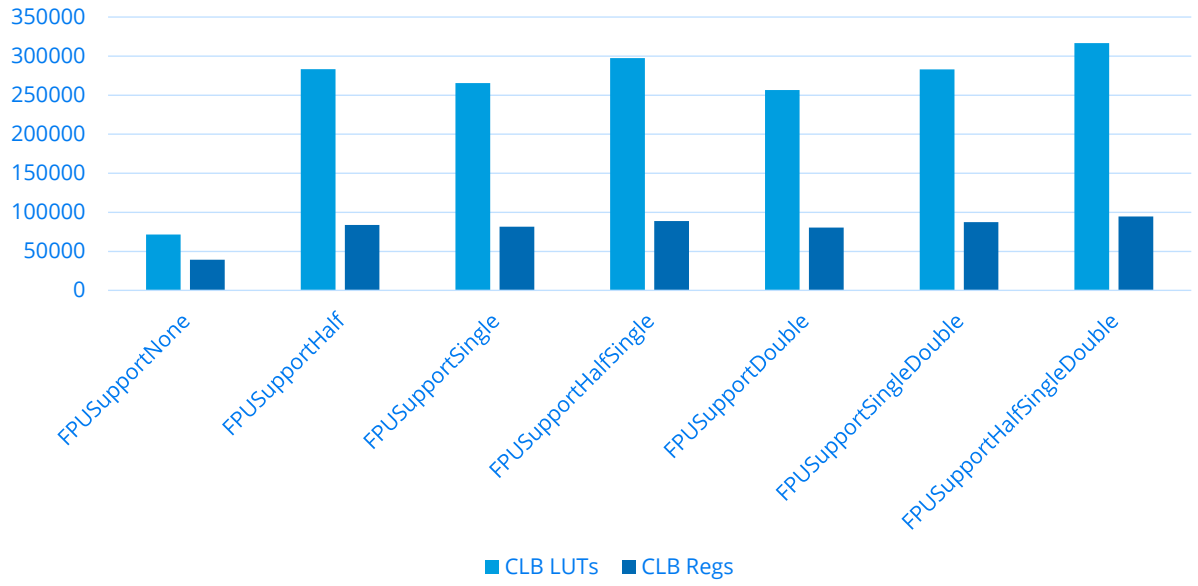

Figure 4.4: LUTs and FFs of Ara for different FPUSupport when lane $l = 4$ and VLEN = 512.

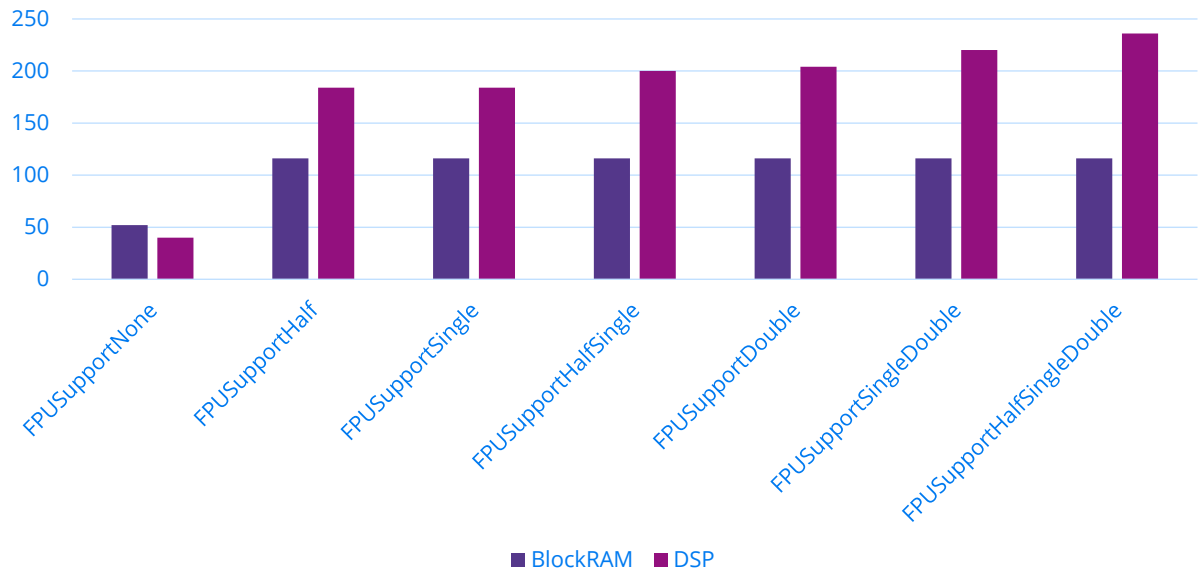

Figure 4.5: BlockRAMs and DSPs of Ara for different FPUSupport when lane $l = 4$ and VLEN $= 512$.

4.1.4 RISC-V core area comparison

To compare area cost with different RISC-V cores, we choose Rocket and BOOM RISC-V core to synthesis within the same frequency 100 MHz. And the original Ariane core and Ara is synthesised with the configuration $l = 4$, VLEN $= 512$, and FPU supporting half, single, double data type. The data is shown in Table 4.5 and Figure 4.6. The Rocket is in-order RISC-V core and BOOM is out-of-order RISC-V core. Compared with Rocket and BOOM, the original Ariane cost less area. However, for the Ara supporting vector instructions, the area cost is much larger than Rocket and BOOM. So supporting vector instructions for RISC-V cores will greatly increase the area complexity.

Table 4.5: The Area of different RISC-V cores compared with Ara ($l = 4$, VLEN $= 512$ and FPUSupportHalfSingleDouble).

Number of Lanes l	**CLB LUTs**	**CLB FFs**	**BlockRAM**	**DSP blocks**
Rocket	66337	30991	157	27
BOOM	179869	91474	167	48
Ariane RISC-V	40624	24228	44	28
Ara (l=4)	316583	94680	120	236

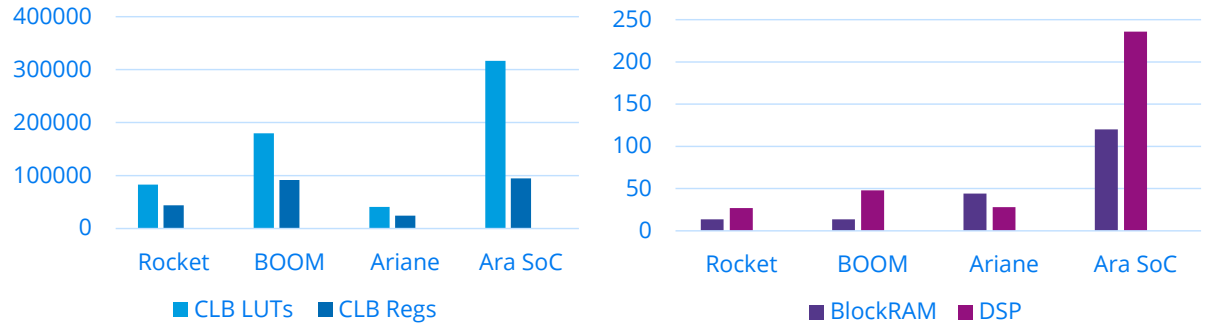

Figure 4.6: The Area of different RISC-V cores compared with Ara ($l = 4$, VLEN = 512 and FPUSupportHalfSingleDouble).

4.2 The M^3 Hardware System Area Cost Analysis and FPGA Implementation

The M^3 hardware system is synthesised to get the complete system data. The architecture of system is shown in Figure 3.1, which contains the following tiles:

- 7x Rocket Processor
- Ara Processor
- 2x DDR4 Memory Interface
- Gbit Ethernet Interface
- UART Interface

The Rocket Processor, DDR4 Memory Interface, Gbit Ethernet Interface and UART Interface is synthesised with the frequency of 100 MHz, and the Ara is synthesised with the frequency of 80 MHz, $l = 4$, VLEN = 512, and FPU supporting half, single, double data type. The area cost of the whole system is shown in Table 4.6.

Table 4.6: The M^3 hardware Area Cost of different tiles by synthesis

Module Name	**CLB LUTs**	**CLB FFs**	**BlockRAM**	**DSP blocks**
The M^3 hardware	862400	372337	1300	431
Rocket tile $\times 7$	66337 $\times 7$	30991 $\times 7$	157 $\times 7$	27 $\times 7$
Ara tile	320673	94651	116	236
DDR4 Interface $\times 2$	25271 $\times 2$	22277 $\times 2$	36 $\times 2$	3 $\times 2$
Gbit Ethernet Interface	13174	7671	13	0
NoC Network	15141	8949	0	0

The percentage of the FPGA's overall resource is shown in Table 4.7. In the table, we can see that the M^3 system costs more than half of LUTs and BlockRAM. And in theory, there could be 3 Ara tile fit into VCU118 FPGA, otherwise the LUTs resource is not enough.

Table 4.7: The M^3 hardware Area Cost percentage of synthesis

Resource Type	**Used**	**Available**	**Utilization%**
CLB LUTs	862400	1182240	72.95%
CLB FFs	372337	2364480	15.75%
BlockRAM	1362	2160	63.06%
DSP blocks	431	6840	6.30%

After the synthesis, we implement the M^3 Hardware System on FPGA. After place and routing optimization steps on FPGA, the area cost is shown is Table 4.8, which does not have a big difference with synthesis model. And the place and routing area cost on FPGA is shown on Figure 4.7. The yellow part stands for Rocket tile, green for Ara tile, red for NoC network, purple for Gbit Ethernet Interface, blue for DDR4 Interface and cyan for other logic like router.

Table 4.8: The M^3 hardware Area Cost of different tiles after place and routing

Module Name	**CLB LUTs**	**CLB FFs**	**BlockRAM**	**DSP blocks**
The M^3 hardware	846048	371881	1300	431
Rocket tile $\times 7$	65412 $\times 7$	30991 $\times 7$	157 $\times 7$	27 $\times 7$
Ara tile	315579	94715	116	236
DDR4 Interface $\times 2$	23138 $\times 2$	21655 $\times 2$	36 $\times 2$	3 $\times 2$
Gbit Ethernet Interface	12969	7649	13	0
NoC Network	14751	8935	0	0

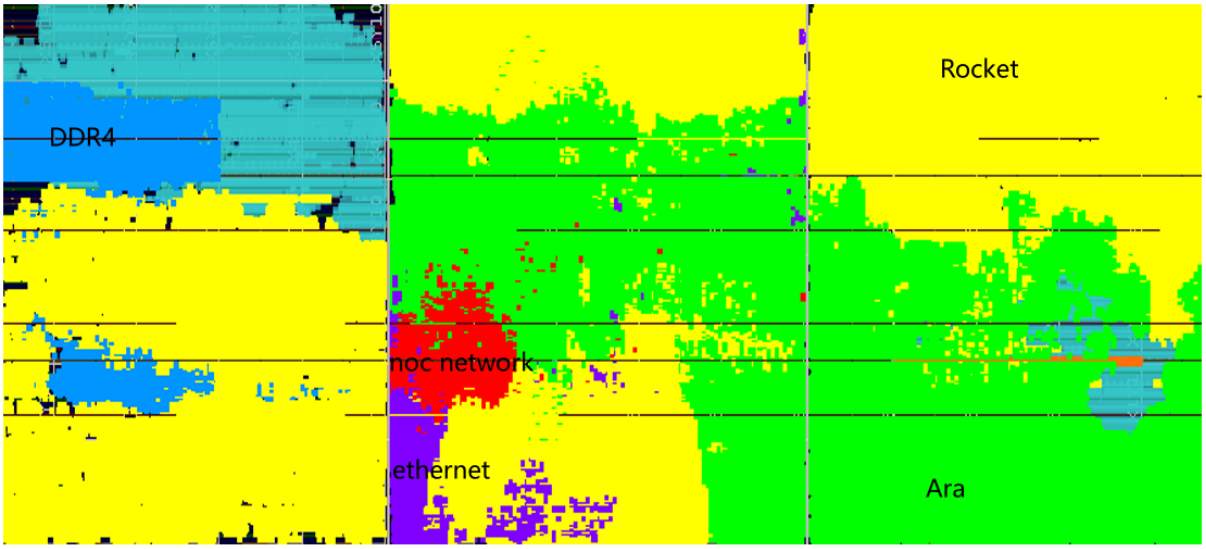

Figure 4.7: The M^3 hardware Area Cost of different tiles after place and routing.

4.3 Ara Performance Analysis

To analyze the performance of Ara, a simple test "foocoo" is used as the benchmark test. The only thing of this test is to write data 0xdeadbeef to address 0xf0000c00 as result. The foocoo original C file is compiled into assembly by RISC-V GCC tool-chain. Then the assembly code is translated into hex which can be recognized as the instruction. The waveform of foocoo test is shown in Figure 4.8. And all the simulates are done with the frequency 100 MHz, clock period 10 ns.

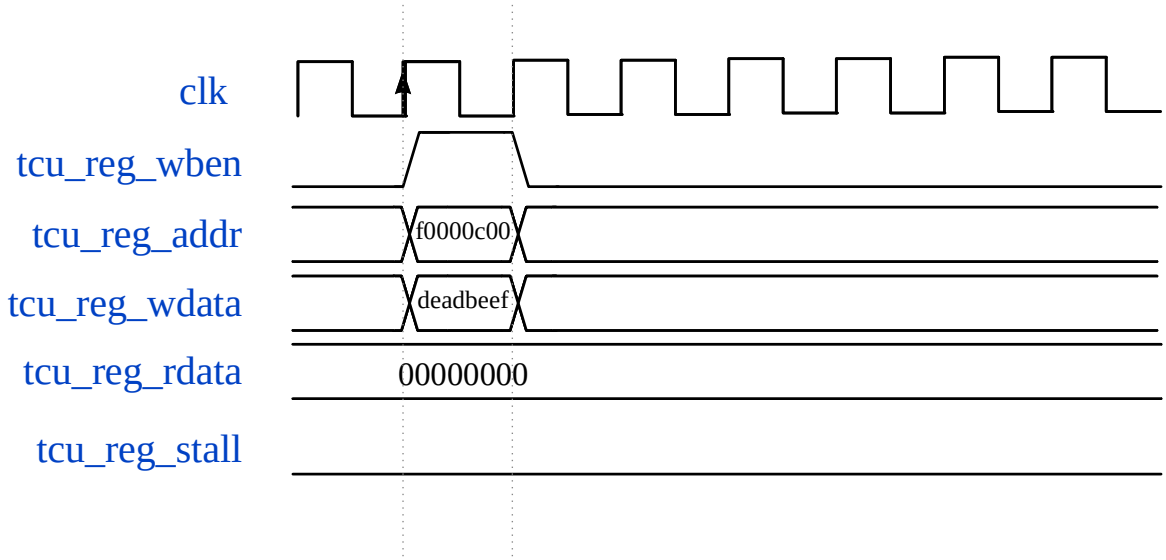

Figure 4.8: The foocoo RISC-V testbench waveform.

We count the simulate cycles for the cores finishing the foocoo test. In order to have a comparison with other RISC-V cores, the simulate is also done on Rocket core. The result is shown in the Table 4.9.

Table 4.9: The number of cycles that RISC-V cores finish foocoo test.

Core Name	**Cycles**
Rocket core	160700
Ara core	420609

From the cycle data, we can see that for the current configuration of Ara, it does not have advantages compared with Rocket running normal RISC-V benchmarks like foocoo. This happens because various reasons, one is the improper configuration of cache of Ariane. The Ariane core always wants to get data from DRAM rather than use the cache. One

possible reason is that the D-cache's cache line size is 16 byte, which is relatively small (Rocket D-cache line size is 64 byte). However, the D-cache's cache line size cannot be changed in FPGA, otherwise the D-cache will store the wrong data. So the cache settings needs to be optimized in further work. By theory, Ara's normal benchmarks should be better than Rocket. And the main benefits of Ara is that it can accelerate the vector instructions benchmarks, but it needs the compiler supporting.

5 CACHE COHERENCE IMPLICATION

For multicore system like M^3 hardware system, there could be the problem with the consistency and synchronization of data which is stored in different caches. So the cache coherence implication is suggested for this problem.

The cache coherence implication is based on the BlackParrot platform. The original microarchitecture is shown in Figure 2.6. The Coherence network (Bedrock), DRAM network and I/O network are independent with each other, each network has its own router to router the data packet. The coherence network is the key to implicate the cache coherence on multi-core system. The coherence network (BedRock Network) is consists of 4 networks: Request network, Command network, Fill network and Response network. The BedRock Network is shown in Figure 5.1. All the coherence networks have no specific ordering properties.

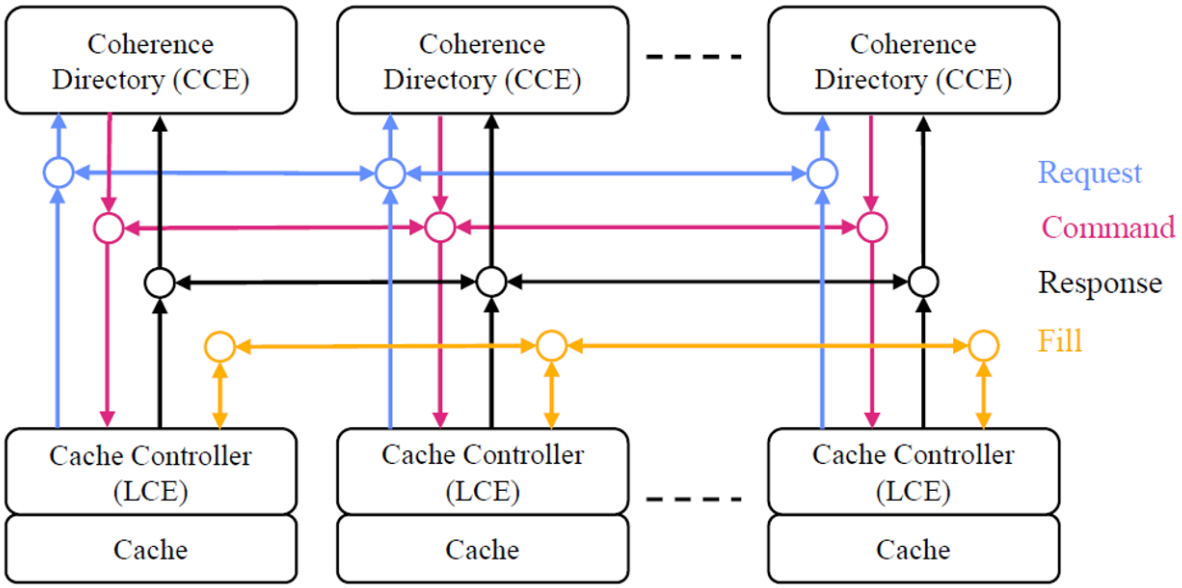

Figure 5.1: The overview of BedRock Networks [9].

And the defines of the four networks:

- Request Network: The Request network sends out messages from the cache controller to the coherence directory.

- Command Network: The Command network sends out coherence commands from the coherence directory to the cache controllers.
- Fill Network: The Fill network sends out inter-cache data transfers between the cache controllers.
- Response Network: The Response network sends out messages from the cache controller to the coherence directory in reaction to commands issued by the directory or relayed from another cache controller.

The cache controller or directory deal with network packets from high to low priority is Response, Fill, Command and Request. When cache controller or directory receive a message, it will send out message by the priority. For example, Commands will be sent due to Requests, and cache controllers will send out Fills or Responses when receive Command. Also, cache controller will send Response when it receives Fills. By the priority, the network can avoid to cause deadlock [24].

The cache controllers are also called Local Cache Engines (LCEs) in Bedrock protocol. It manages coherence messages for a single cache. The cache can be a private or shared cache, however, it should be a write-back cache but not a write-through cache. When there is a write or read miss occurs, it will send out a new request. Also, it will response to coherence commands from Command network. The permissions of block which associated to the cache will keep unchanged until the controller sends out a new request. Also, the cache controller need to response to coherence commands in time. It cannot stall the responses to command or fill messages. However, the request messages can be stalled when processing command or fill messages.

Central Cache Engines (CCEs) is referred to the BedRock coherence directory. The system could have one or more directories, the address space will be divided equally by all directories if there are multiple directories. To prevent deadlock, the directory need to deal with responses in time. The cache directory deals with coherence requests with the order of the arrive sequence. If space is needed for a newly requested block and a cache block with dirty data is returned, a write-back operation will be performed and sent to memory. Subsequently, if the request type demands it, all other caches in the shared state containing that block will be invalidated. Then the directory either initiates a memory read, commands a cache-to-cache transfer (with possible writeback), or responses with update permissions if the cache has the block. It then waits for a coherence acknowledgment to complete the transaction [24].

5.1 Coherence System Architecture

To fit the BlackParrot coherence protocols into the M^3 hardware system, some changes needs to be done on the microarchitecture of M^3 hardware. There are three NoC networks (coherence network, I/O network and DRAM network) in BlackParrot microarchitecture. There are two networks (DRAM Network and Coherence Network) connect to a core tile, which is shown in Figure 5.2. And DRAM network and Coherence network uses the similar encode principles. So the TCU and NOC Interface can both encode messages from DRAM network and Coherence network by theory. The important part is the modid of two kind of messages are different.

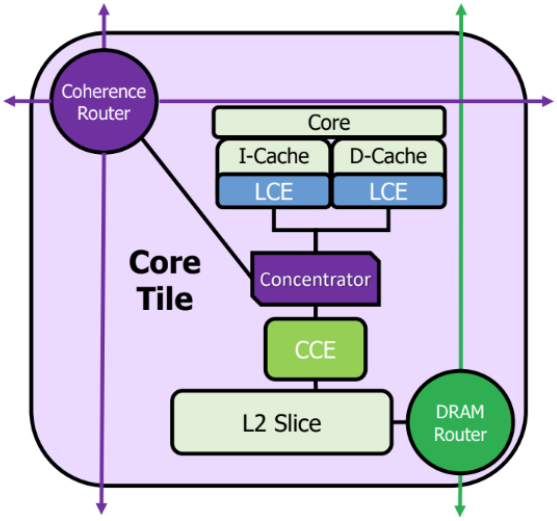

Figure 5.2: The overview of core tile in BlackParrot microarchitecture. [10].

The original BlackParrot platform routers and M^3 hardware routers between tiles have different protocol, so the M^3 hardware changes are mainly on the data path between core tile and routers. Also some changes will also happen on TCU, which will be explained later. An assumed coherence model microarchitecture fitted for the M^3 hardware is assumed, which is shown in Figure 5.3.

The core tile treated as black-box will send out five types of packets (Request packets, Response packets, Fill packets, Command packets and Mem packets) which are the corresponding Coherence network and DRAM network in BlackParrot. Since coherence packets need to be arbitrated in a specific priority. So the packet-arbiter module needs to arbitrate those four packets(Request, Response, Fill and Command). Since Mem packets is independent of the four packets, however, in order to not block the memory read. The high to low priority I recommend is: Response, Fill, Command, Mem and Request.

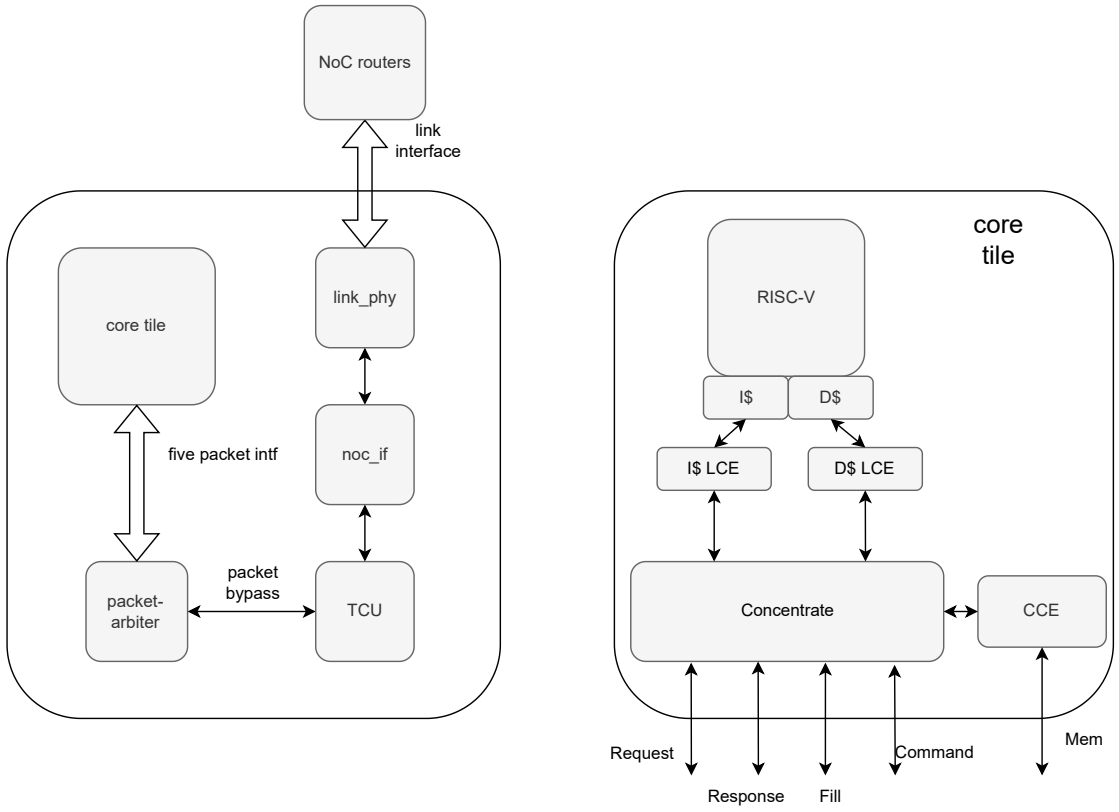

Figure 5.3: The assumed microarchitecure of M^3 hardware fitted coherence model, left is the overview architecture, right is the core tile.

And here the format of five packets. There three main parts: 1-bit valid, 128-bit data and 60-75 bit header (not fixed, depend on packet types). The packets format is shown in Figure 5.4 and the waveform is shown in Figure 5.5, also with their defines:

- data: data is the cache coherence data, which can be burst of 2 or 4 transferred.
- valid: valid[0] stands for valid bit for I-cache, valid[1] stands for valid bit for D-cache.
- msg_type: msg_type is a union of the LCE-CCE and Mem message types. Including req, cmd, fill, resp, fwd and rev.
- subop: specifies the type of store. example: e_bedrock_store, e_bedrock_amolr and e_bedrock_amosc.
- addr: addr is the address used by the message.
- size: size indicates the size in bytes of the message using.

- payload: payload is an opaque field to the network, that is network specific (Req, Cmd, Fill, Resp, Mem) and each endpoint will interpret the field as appropriate.

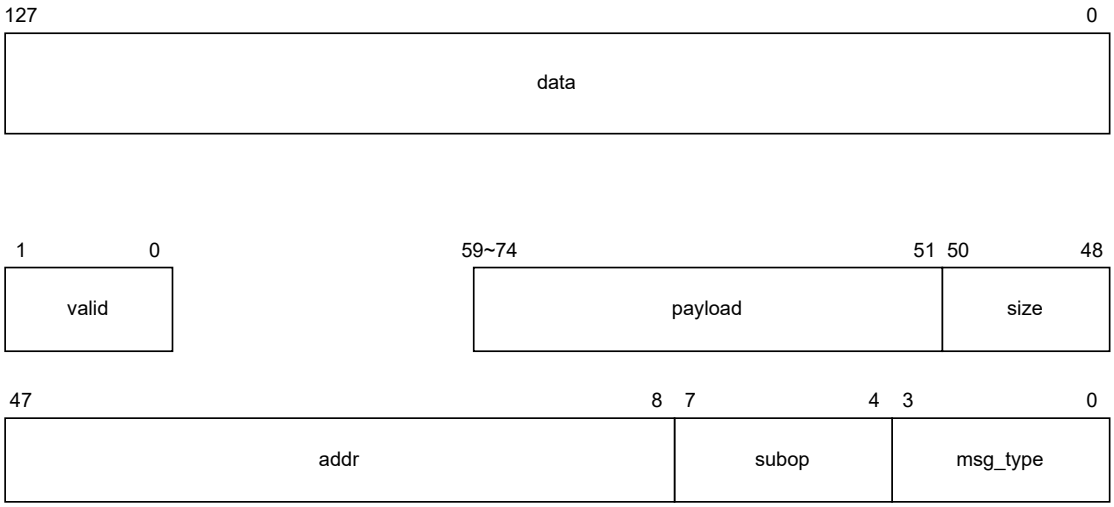

Figure 5.4: The format of five packets (Request packets, Response packets, Fill packets, Command packets and Mem packets)

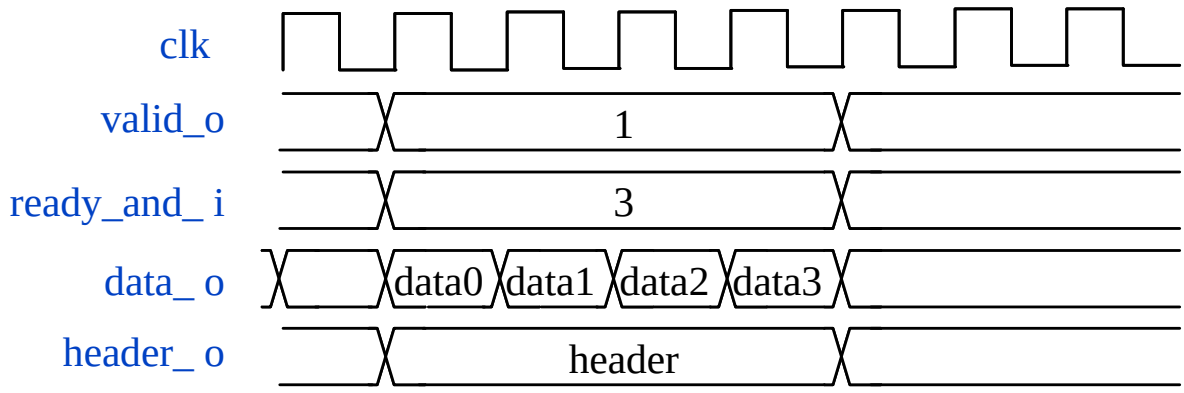

Figure 5.5: The packet data waveform, send by burst of 4

And also there are five types payload, which is shown in Figure 5.6, and take defines of Mem payload as example:

- lce_did: the ID of LCE that sent the initial request to the CCE.
- way_id: the way ID within the cache miss address target set to fill the data in to.
- prefetch: if the request is a prefetch from LCE.
- uncached: if the request is an uncached request from LCE.

- speculative: if the request is issued speculatively by the CCE.

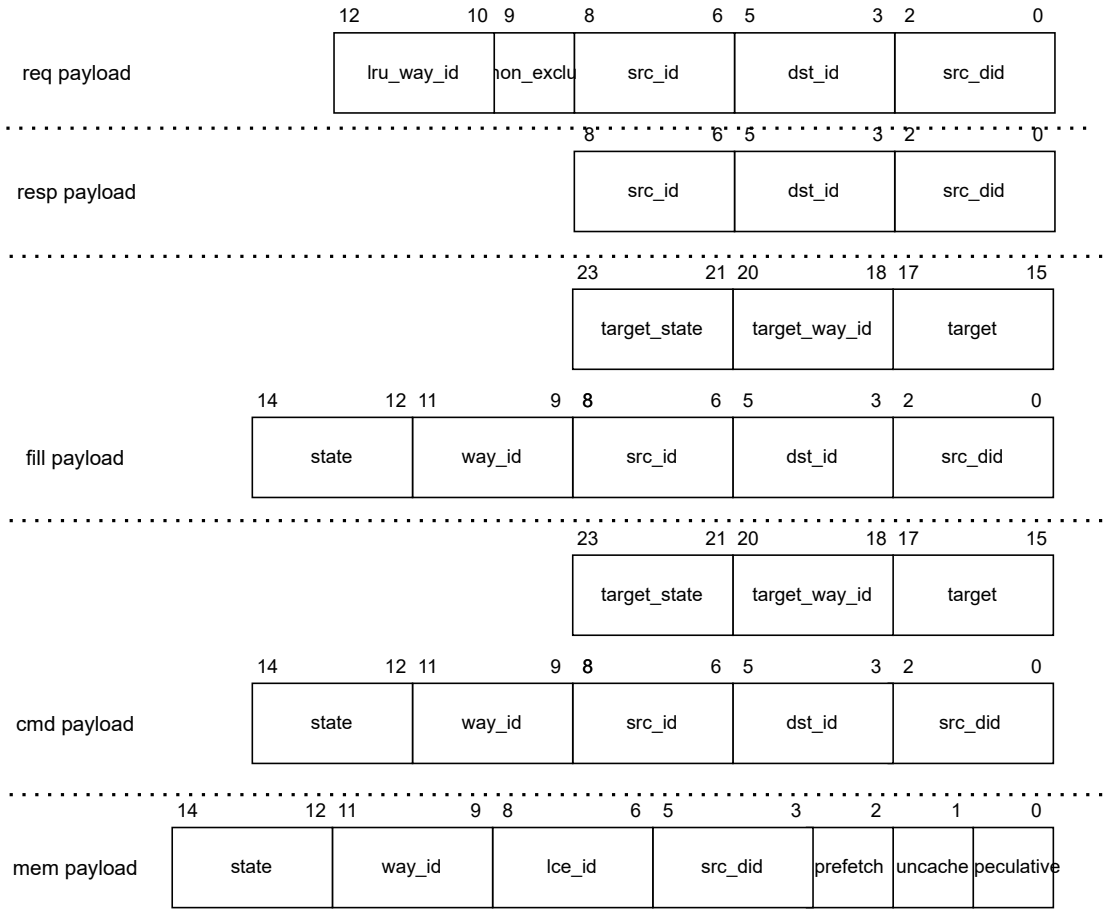

Figure 5.6: The payload of packet, with five types

To fit the data type to TCU, the packet arbiter will receive the five data packet and send a general data packet for TCU. To fit in data width of TCU, the addr bits in original packet header in Figure 5.4 is get out and set a independent address channel. And take 24-bit data width as the payload width. Also, the 2-bit valid bits is merged into the first burst data0. The data format of first burst data0 is shown in Figure 5.7. Also, the 128-bit data bits in original packet is divided into two 64-bit data1 and data2. So the 59-74 header and 128-bit data is sent by data0, data1 and data2 in Figure 5.8.

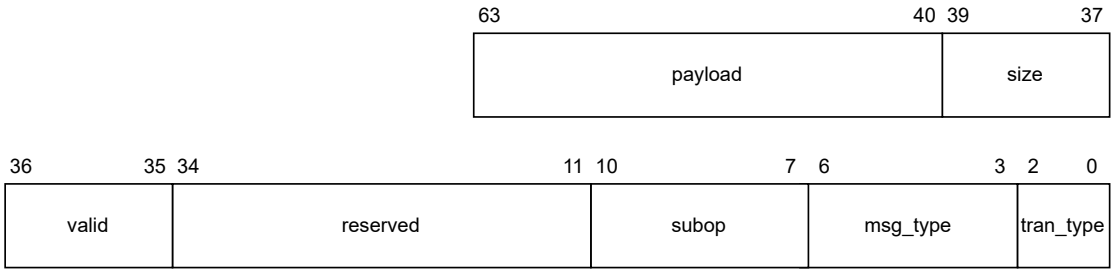

Figure 5.7: The first burst data of general packet

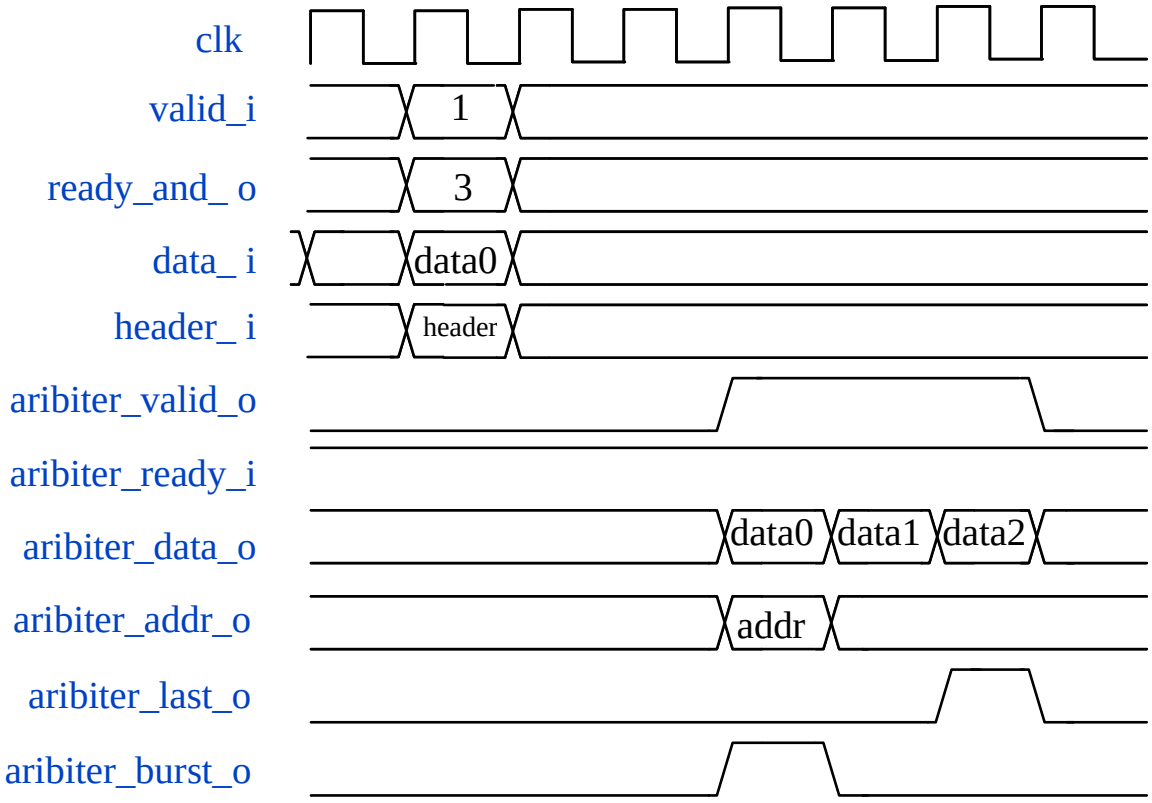

Figure 5.8: The waveform of arbiter receives 5 packets and generate the general packet for TCU

Because the current TCU does not have the interface to receive the data packet, so TCU needs to add the extra logic to encode the general packet into packets which is fit into M^3 system. Also, the logic can be enabled with the modid which can recognize the coherence tile.

5.2 Cache Coherence Protocol

The cache coherence protocol of the coherence M^3 hardware system can be referred from BedRock Coherence Protocol Specification [24]. There are 6 states as the coherence states, which are I (invalid), S (shared), E (exclusive), M (modified), O(owned), F (forwarded), adopting the terminology in the book [16]. There are three types of table indicates the full cache coherence protocol:

- The BedRock Network Message Table describes the network messages and its main function. There are four Message tables(BedRock Request, Command, Fill and Response Network Messages). For those messages are associated with BedRock Cache Protocol table and BedRock Coherence Next State table.
- The BedRock Cache Controller Protocol Table stands for the controller's behaviors (Action/State) after receiving Cache Action or Coherence Message events. For the BedRock Coherence Directory Protocol Table, the (Action/State) behavior is a little different. It would indicate a coherence state which is associated with that message or composite message components.
- The BedRock Cache Controller Next State Table indicates the next state of the controller is going to be when a cache or coherence event happened. Also with BedRock Coherence Directory Next State Table, which will indicate both the cache directory and the cache controller that actives the event.

5.3 The Coherence System Performance Analysis

Since the coherence implication has not integrated to the M^3 hardware system, the analysis on the performance mainly on theory. To analyze to performance the coherence system, we check the waveform of five networks in original BlackParrot platform, by counting the data packets in the networks in certain cycles. Because the M^3 coherence hardware system has only one packet interface to TCU, which is arbitrated by the packet arbiter module. We can count if there are some packets overlap to judge if the arbiter block some packets and have impact on hardware system performance.

The BlackParrot platform has two cores implemented and runs a "helloworld" program, the waveform of one core's networks are shown in Figure 5.9. The system simulating frequency is 1-GHz, which means clock period is 1ns. The whole processing waveform can be separated into 3 stages according the waveform features:

- Stage 0: Stage 0 is from program start to 68400 cycles, in which Request networks, Command networks and Mem networks send out and receive data packets.
- Stage 1: Stage 1 is from 68400 cycles to about 88000 cycles to , in which only Request networks and Command networks, send out and receive data packets.
- Stage 2: Stage 2 is from 88000 cycles to program end , in which basically every network sends out and receives data packets.

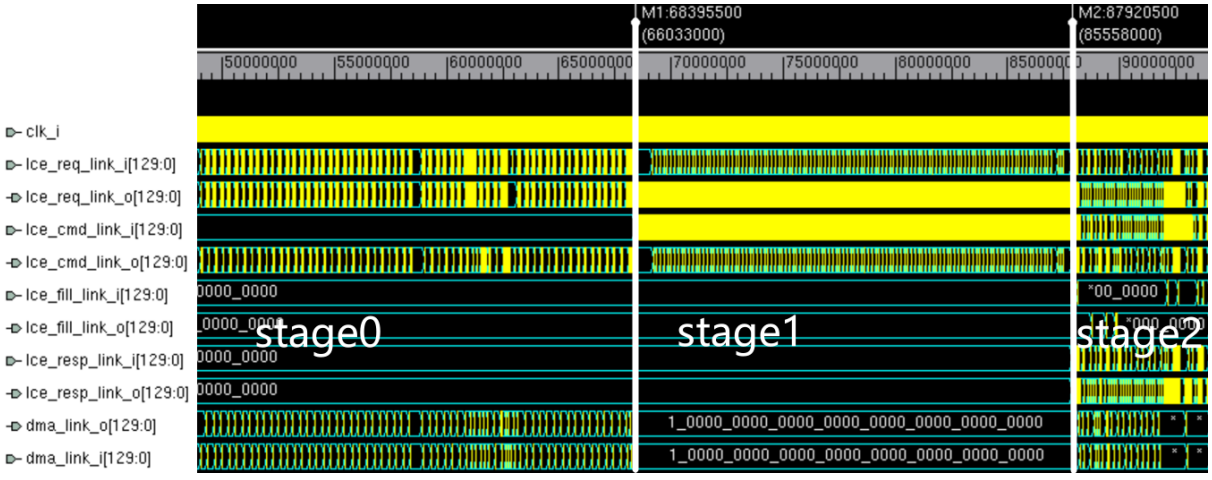

Figure 5.9: The waveform of core's five networks on a "helloworld" program

And for each stages we sample 1000 cycles period to count the data packets number for every network. For stage0, the time period is 2000ns - 3000ns (2000cycle - 3000cycle), which is shown in Table 5.1. For stage1, the time period is 70000ns - 71000ns (70000cycle - 71000cycle), which is shown in Figure 5.2. For stage2, the time period is 88000ns - 89000ns (88000cycle - 89000cycle), which is shown in Table 5.3.

Table 5.1: Number of packets in 2000ns - 3000ns (2000cycle - 3000cycle) in stage0.

Packet Network Name	**Packet number**
req_link_in	41
req_link_out	16
cmd_link_in	0
cmd_link_out	24
fill_link_in	0
fill_link_out	0
resp_link_in	0
resp_link_out	0
mem_link_in	6
mem_link_out	3
input packets overlap	0
output packets overlap	0

Table 5.2: Number of packets in 70000ns - 71000ns (70000cycle - 71000cycle) in stage1.

Packet Network Name	**Packet number**
req_link_in	12
req_link_out	55
cmd_link_in	88
cmd_link_out	7
fill_link_in	0
fill_link_out	0
resp_link_in	0
resp_link_out	0
mem_link_in	0
mem_link_out	0
input packets overlap	0
output packets overlap	0

Table 5.3: Number of packets in 88000ns - 89000ns (88000cycle - 89000cycle) in stage2.

Packet Network Name	**Packet number**
req_link_in	18
req_link_out	13
cmd_link_in	35
cmd_link_out	51
fill_link_in	5
fill_link_out	0
resp_link_in	15
resp_link_out	0
mem_link_in	8
mem_link_out	8
input packets overlap	0
output packets overlap	0

As the analytic conclusion, the overlap is rarely happen on the five packets. So the packet arbiter will have limit performance impact on the whole coherence system since all coherence networks require no ordering properties. So the packet arbiter is applicable for the coherence system.

6 CONCLUSION

In the paper, we presented the integrated architecture for Ara integrated M^3 hardware system. For the microarchitecture, we choose a area-saving plan for Ara. Also, from the area cost data, we concluded that area of Ara is close related to the number of Lanes l in Ara co-processor and FPU supporting types. The area cost of Ara is much bigger than the normal RISC-V cores, like Rocket and BOOM. However, due to current settings of cache in Ariane, the processor cannot have the full performance.

Also, one assumed cache coherence microarchitecture was proposed for M^3 hardware system. We analyzed the feasibility and performance of coherence architecture. And we concluded that with the implement of packet-arbiter module, the coherence function can be realized without too much performance cost.

7 DISCUSSION AND FUTURE WORK

For the current system, there are some improvements is worthy need to be done:

- Optimize the cache configuration: due to the limit of FPGA platform, the data cache line size is fixed into 16 byte, which is relative small. And it might cause Ara keep reading data from DRAM rather than cache.
- Test more benchmarks: more benchmarks are worth to try for Ara to test its performance and stability.
- Design the Packet-Arbiter module: to realize to M^3 hardware based cache coherence system, the packet-arbiter hardware module need to be designed and implemented into hardware.

8 ACKNOWLEDGMENTS

First, I would like to thank my thesis supervisors Viktor Razilov and Mattis Hasler, who give me support and suggestions when I stuck at some certain difficult. Also, I want to give sincere thanks to Prof.Gerhard Fettweis, who is not only a knowledgeable professor, but also a interesting and humorous person giving us lively academic lectures.

Bibliography

[1] *4Gb: x4, x8, x16 DDR4 SDRAM Features*. Micron. 2024. URL: https://www.micron.com/products/dram/ddr4-sdram/part-catalog/mt40a256m16ge-083e-it..

[2] *AMBA® AXI Protocol Specification*. ARM, Inc. 2023. URL: https://developer.arm.com/Architectures/AMBA.

[3] Krste Asanović et al. *The Rocket Chip Generator*. Tech. rep. UCB/EECS-2016-17. Apr. 2016. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html.

[4] Nils Asmussen. *Trusted Communication Unit – Specification*. Barkhausen Institut. 2023.

[5] Nils Asmussen et al. "M3: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores." In: ASPLOS '16. Atlanta, Georgia, USA: Association for Computing Machinery, 2016, pp. 189–203. ISBN: 9781450340915. DOI: 10.1145/2872362.2872371. URL: https://doi.org/10.1145/2872362.2872371.

[6] aswaterman. *RISC-V Proxy Kernel and Boot Loader*. 2023. URL: https://github.com/riscv-software-src/riscv-pk.

[7] *AXI4+ATOP Fully-Connected Crossbar*. pulp-platform. 2024. URL: https://github.com/pulp-platform/axi/blob/master/doc/axi_xbar.md.

[8] Jonathan Balkind et al. "OpenPiton: An Open Source Manycore Research Framework." In: ASPLOS '16. Atlanta, Georgia, USA: Association for Computing Machinery, 2016, pp. 217–232. ISBN: 9781450340915. DOI: 10.1145/2872362.2872414. URL: https://doi.org/10.1145/2872362.2872414.

[9] *BlackParrot at FOSDEM 2020*. University of Washington and Boston University. 2020. URL: https://github.com/black-parrot/black-parrot/tree/master?tab=readme-ov-file.

[10] *BlackParrot at the December 2020 RISC-V Summit*. University of Washington and Boston University. 2020. URL: https://github.com/black-parrot/black-parrot/tree/master?tab=readme-ov-file.

[11] Matheus Cavalcante et al. “Ara: A 1-GHz+ Scalable and Energy-Efficient RISC-V Vector Processor With Multiprecision Floating-Point Support in 22-nm FD-SOI.” In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28.2 (2020), pp. 530–543. DOI: 10.1109/TVLSI.2019.2950087.

[12] Raimarius Delgado, Jaeho Park, Byoung Wook Choi. “MPSoC: The Low-cost Approach to Real-time Hardware Simulations for Power and Energy Systems.” In: *IFAC-PapersOnLine* 52.4 (2019). IFAC Workshop on Control of Smart Grid and Renewable Energy Systems CSGRES 2019, pp. 57–62. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2019.08.155. URL: https://www.sciencedirect.com/science/article/pii/S2405896319304914.

[13] Alex Forencich. *Verilog Ethernet*. URL: https://github.com/alexforencich/verilog-ethernet.

[14] *Introduction to AMBA AXI4*. Issue 0101. ARM, Inc. 2020. URL: https://www.arm.com/.

[15] Lech Jóźwiak, Menno Lindwer. “Issues and Challenges in Development of Massively-Parallel Heterogeneous MPSoCs Based on Adaptable ASIPs.” In: *2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing*. 2011, pp. 483–487. DOI: 10.1109/PDP.2011.55.

[16] Vijay Nagarajan et al. *A Primer on Memory Consistency and Cache Coherence*. 2nd. Morgan & Claypool Publishers, 2020. ISBN: 1681737094.

[17] *Official repository of M3: microkernel-based system for heterogeneous manycores*. Barkhausen Institut gGmbH. 2024. URL: https://github.com/Barkhausen-Institut/M3.

[18] Matteo Perotti et al. “A “New Ara” for Vector Computing: An Open Source Highly Efficient RISC-V V 1.0 Vector Processor Design.” In: *2022 IEEE 33rd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. 2022, pp. 43–51. DOI: 10.1109/ASAP54787.2022.00017.

[19] Daniel Petrisko et al. “BlackParrot: An Agile Open-Source RISC-V Multicore for Accelerator SoCs.” In: *IEEE Micro* 40.4 (2020), pp. 93–102. DOI: 10.1109/MM.2020.2996145.

[20] *Rocket Chipyard Framework*. University of California, Berkeley. 2024. URL: https://github.com/ucb-bar/chipyard.

[21] *SCoHa Hardware Documentation*. Barkhausen Institut. 2022.

[22] *VCU118 Evaluation Board: User Guide*. Xilinx, Inc. 2018. URL: https://www.xilinx.com/support/documentation/boards_and_kits/vcu118/ug1224-vcu118-eval-bd.pdf.

[23] wikipedia. *MESI protocol*. 2024. URL: https://en.wikipedia.org/wiki/MESI_protocol.

[24] Mark Wyse. *The BedRock Cache Coherence Protocol and System*. 1.1. University of Washington. 2022. URL: https://github.com/black-parrot/black-parrot/blob/master/docs/bedrock_uarch_guide.md.

[25] Florian Zaruba, Luca Benini. "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology." In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27.11 (2019), pp. 2629–2640. DOI: 10.1109/TVLSI.2019.2926114.

[26] zarubaf. *ariane-architecture*. 2019. URL: https://github.com/openhwgroup/cva6/tree/v4.2.0.

[27] Jerry Zhao et al. "SonicBOOM: The 3rd Generation Berkeley Out-of-Order Machine." In: (May 2020).