System Cache v5.0

LogiCORE IP Product Guide

Vivado Design Suite

PG118 (v5.0) November 5, 2021

Xilinx is creating an environment where employees, customers, and partners feel welcome and included. To that end, we're removing non-inclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You may still find examples of non-inclusive language in our older products as we work to make these changes and align with evolving industry standards. Follow this link for more information.

Table of Contents

Chapter 1: Introduction **4**
- Features 4
- IP Facts 5

Chapter 2: Overview **6**
- Navigating Content by Design Process 7
- Applications 7
- Unsupported Features 17
- Licensing and Ordering 18

Chapter 3: Product Specification **20**
- CCIX RA Master Port Cache Coherency 20
- CHI RN Master Port Cache Coherency 22
- Address Translation Service and Cache 24
- AXI Master 26
- ACE Master Port Cache Coherency 28
- Optimized Ports Cache Coherency 28
- Exclusive Monitor 29
- Cache Memory 29
- Error Handling 30
- Non-Secure Handling 32
- Control and Statistics 33
- Standards 35
- Performance 36
- Resource Use 41
- Port Descriptions 42
- Register Space 46

Chapter 4: Designing with the Core **101**
- System Cache Design 101
- Transaction Properties 102
- General Design Guidelines 115

Back-Door DMA..... 122
Clocking..... 125
Resets..... 125
Protocol Description..... 125

Chapter 5: Design Flow Steps..... 132
Customizing and Generating the Core..... 132
Constraining the Core..... 162
Simulation..... 163
Synthesis and Implementation..... 163

Appendix A: Upgrading..... 164
Migrating to the Vivado Design Suite..... 164
Upgrading in the Vivado Design Suite..... 164
Functionality Changes..... 164
Port and Parameter Changes..... 166

Appendix B: Debugging..... 168
Finding Help on Xilinx.com..... 168
Debug Tools..... 169
Simulation Debug..... 170
Hardware Debug..... 171
Interface Debug..... 171

Appendix C: Additional Resources and Legal Notices..... 175
Xilinx Resources..... 175
Documentation Navigator and Design Hubs..... 175
References..... 175
Revision History..... 176
Please Read: Important Legal Notices..... 178

XILINX

Chapter 1

Introduction

The LogiCORE™ System Cache IP core provides system level caching capability to an AMBA® AXI4 system.

Features

- Support for Cache Coherent Interconnect for Accelerators (CCIX®) Request Agent (RA) coherency protocol with one port and one link
- Support for CHI interface with one Request Node (RN)
- Address Translation Cache (ATC) with the PCIe® Address Translation Service (ATS) protocol
- Dedicated AXI4 slave ports for a MicroBlaze™ processor
- Up to 16 generic AXI4 slave ports for other AXI4 masters, limited to four ports with CCIX master coherency
- Up to 16 generic AXI4 slave ports for other AXI4 masters
- Optional cache coherency on dedicated MicroBlaze processor ports with AXI Coherency Extension (ACE)
- Optional support for exclusive access with non-coherent configuration
- Optional cache coherency on master port for Zynq® UltraScale+™ MPSoC connection
- Optional support for Non-Secure transactions
- Optional support for AXI error handling
- AXI4 master port connecting the external memory controller
- Highly configurable cache—2 or 4 set associative cache of up to 4 MB in size
- Optional AXI4-Lite Statistics and Control port
- Supports up to 64-bit AXI4 address width
- Optional support for 64-bit AXI4 Virtual Address via ATS and PRI over PCIe
- Optional support for PCIe Advanced Error reporting (AER) error handling, when ATS is in use

IP Facts

LogiCORE™ IP Facts Table	
Core Specifics	
Supported Device Family[1]	UltraScale+™ Virtex® UltraScale+™ HBM UltraScale™ Zynq®-7000 SoC Zynq® UltraScale+™ MPSoC 7 series Versal®
Supported User Interfaces	AXI4, ACE, AXI4-Lite, AXI4-Stream (ATS), CXS (CCIX), CHI
Resources	Performance and Resource Use web page
Provided with Core	
Design Files	Vivado® RTL
Example Design	See the CCIX lounge and the Versal CCIX lounge (registration required)
Test Bench	Not Provided
Constraints File	Not Provided
Simulation Model	Not Provided
Supported S/W Driver	N/A
Tested Design Flows	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Release Notes and Known Issues	Master Answer Record: 54452
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For a complete list of supported devices, see the Vivado® IP catalog.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The System Cache core can be added to an AXI4 system to improve overall system computing performance, for accesses to external memory.

With cache coherency, efficient multi-processor systems can be implemented and the workload distributed between multiple processors or accelerators, with simple and safe data sharing. The coherency is managed on a hardware level with minimal software handling required.

The System Cache core can provide improved system performance for:

- Applications with repeated access of data occupying a certain address range, for example, when external memory is used to buffer data during computations. In particular, performance improvements are achieved when the data set exceeds the capacity of the MicroBlaze™ processor internal data cache.
- Systems with small MicroBlaze processor caches, for example, when the MicroBlaze processor implementation is tuned to achieve as high frequency as possible. In this case, the increased system frequency contributes to the performance improvements, and the System Cache core alleviates the performance loss incurred by the reduced size of the MicroBlaze processor internal caches.
- Accelerators working on data sets that are shared between multiple accelerators and the Application Processing Unit (APU) in the Zynq® UltraScale+™ MPSoC. The cache coherency ensures all participating units can share data safely and efficiently.
- Accelerators working on data sets that are shared between multiple accelerators and a remote PCIe® host using the Cache Coherent Interconnect for Accelerators (CCIX®) cache coherency protocol. The cache coherency ensures all participating units share data safely and efficiently. PCIe Address Translation Services, ATS, with optional Process Address Space ID (PASID) support (technology dependent option), is provided to give accelerators the possibility to use virtual memory synchronized with the host.
- Accelerators connecting to System Cache in Versal® using a CHI connection to the Versal CCIX PCIe Module (CPM) for local coherency. The coherency domain can be extended from the CPM to include other devices by utilizing CCIX. System wide address translation is performed by ATS with PASID support.. This solution provides support for both local and remote memory.

Navigating Content by Design Process

Xilinx® documentation is organized around a set of standard design processes to help you find relevant content for your current development task. All Versal® ACAP design process Design Hubs and the Design Flow Assistant materials can be found on the Xilinx.com website. This document covers the following design processes:

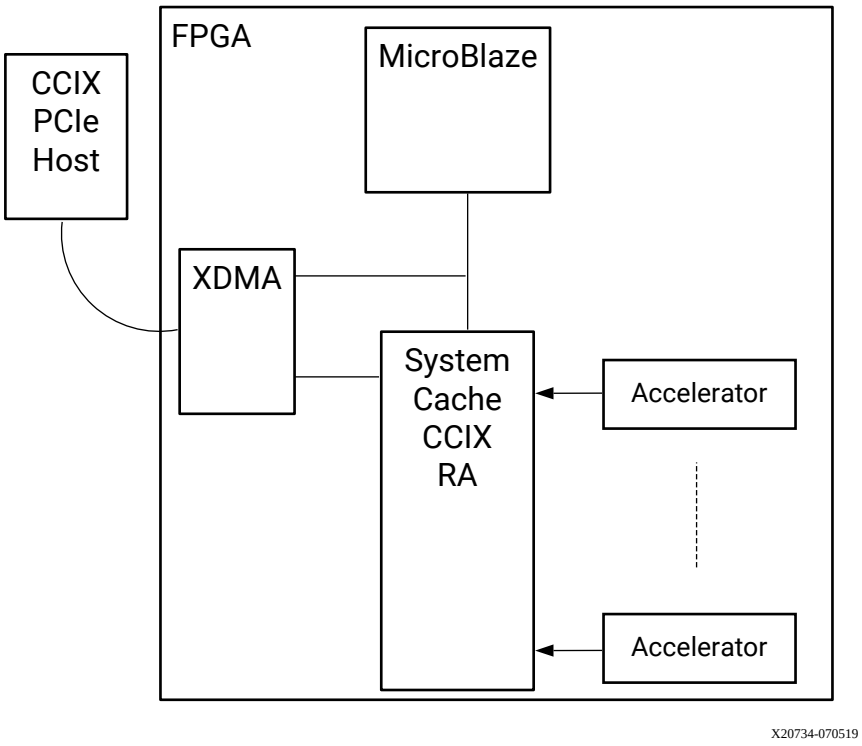
- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, functional simulation, and evaluating the Vivado® timing, resource use, and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
  - Port Descriptions
  - Customizing and Generating the Core
- **System Integration and Validation:** Integrating and validating the system functional performance, including timing, resource use, and power closure. Topics in this document that apply to this design process include:
  - Performance
  - Resource Use
  - Frequency and Hit Rate
  - Bandwidth

Applications

CCIX Example

System Cache configured in CCIX XDMA mode enables cache coherency with memory distributed throughout the system all controlled by a PCIe host. In this use case the accelerators connect to System Cache using AXI4 and System Cache provides a local cache function coherent with the rest of the CCIX memory system, without any need for the accelerators to handle the coherency protocol.

Figure 1: **CCIX Request Agent (RA) XDMA System**



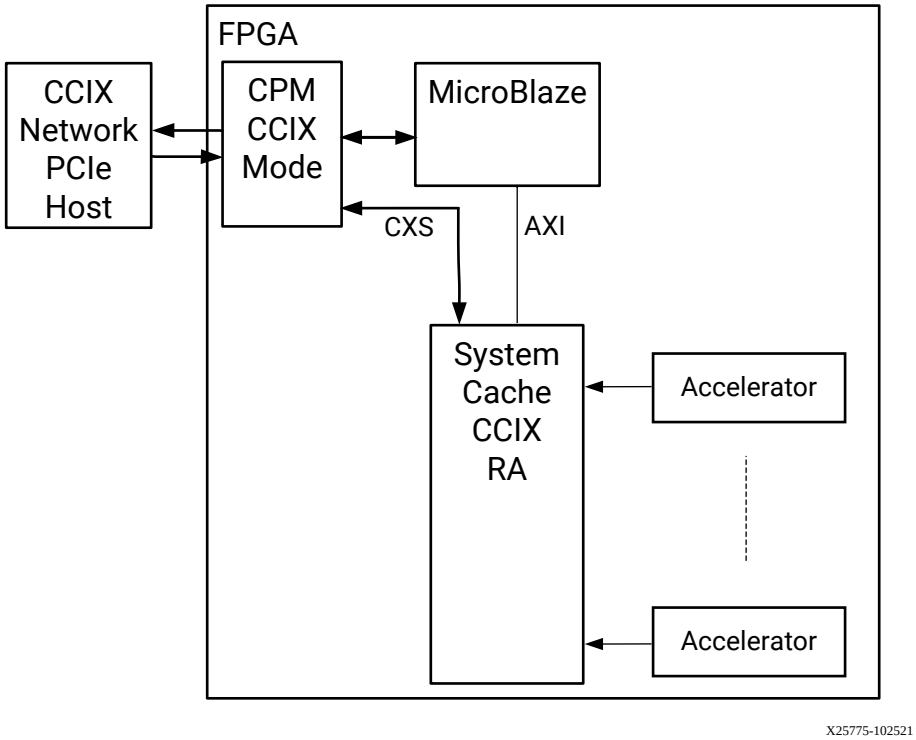
In the CCIX configuration up to four accelerators/kernels can be used, connected to System Cache with ordinary AXI4 interfaces. The system can be configured to use Shared Virtual Memory, SVM, via the Address Translation Services, ATS. Coherent traffic is connected to PCIe using CXS interfaces, while the optional ATS support uses AXI4-Stream. A MicroBlaze processor subsystem handles all the configuration and maintenance to make System Cache operate as a full member of the CCIX network.

Key building blocks in this solution are:

- PCIe Host with CCIX Home Agent
- XDMA IP core handling the PCIe connection for the CCIX communication protocol
- System Cache configured as a Request Agent using the CCIX protocol to keep the local cache coherent with the rest of the CCIX memory system, and the Address Translation Cache (ATC) to handle virtual address translation
- MicroBlaze processor based sub-system controlling System Cache to provide Designated Vendor-Specific Capability, DVSEC, to support the CCIX protocol
- Up to four accelerators connected via AXI4 to System Cache

System Cache can also be connected to the CPM in Versal Premium devices, which can extend the coherency domain to enable cache coherency with memory distributed throughout the system, all controlled by a CCIX enabled PCIe host. The CPM has the ability to connect to one System Cache core with CCIX enabled.

Figure 2: **CCIX RA CPM System**



The CPM configuration is similar to the XDMA example with the following extensions:

- The system can be configured to use Shared Virtual Memory, SVM, via the Address Translation Services, ATS.
- System Cache and CPM supports the ability to use Host propagated PASID, which enables each accelerator to use a unique ID when requesting address translation via ATS.

Figure 3: **CCIX XDMA Component Overview**

Front end (FE), handles arbitration between accelerators

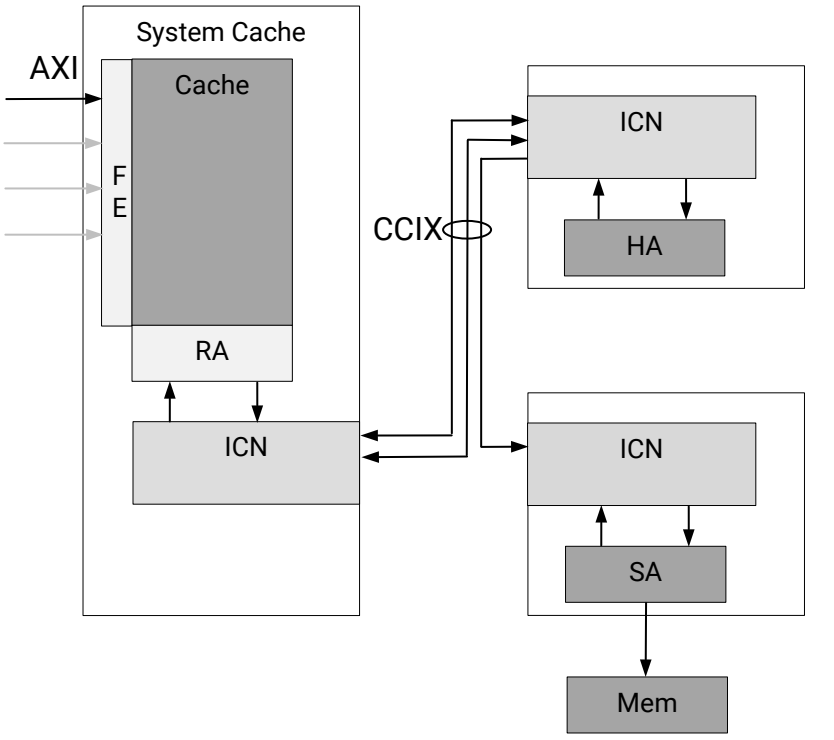
Request Agent (RA), performs read and write transactions to different addresses

Home Agent (HA), is responsible for memory address range in system

Slave Agent (SA), provides additional memory in system. Always accessed through Home Agent

Memory (Mem)

Coherent Interconnect (ICN)



X23069-080219

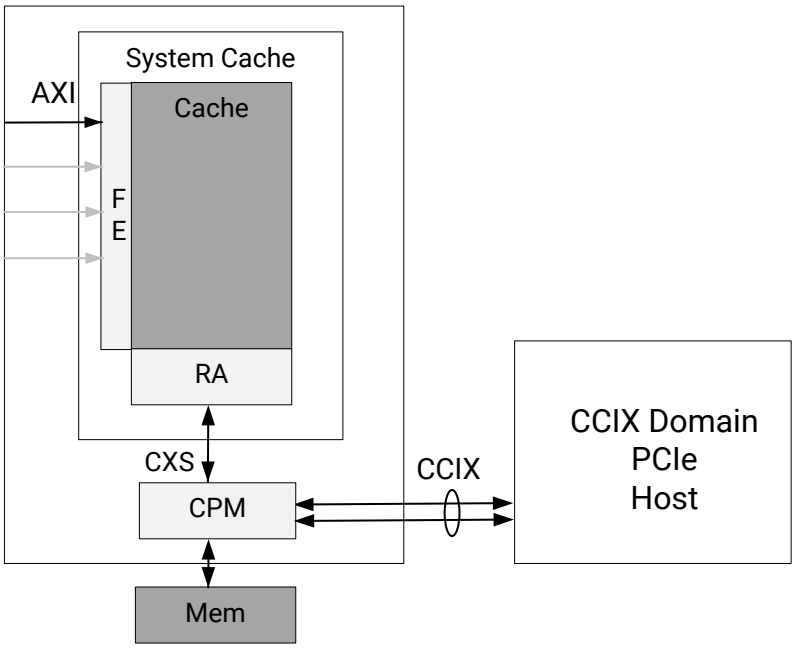
Figure 4: **CCIX CPM Component Overview**

Front end (FE), handles arbitration between accelerators

Request Agent, performs read and write transactions to different addresses

CPM, gateway between internal and external CCIX Domain

Memory (Mem)

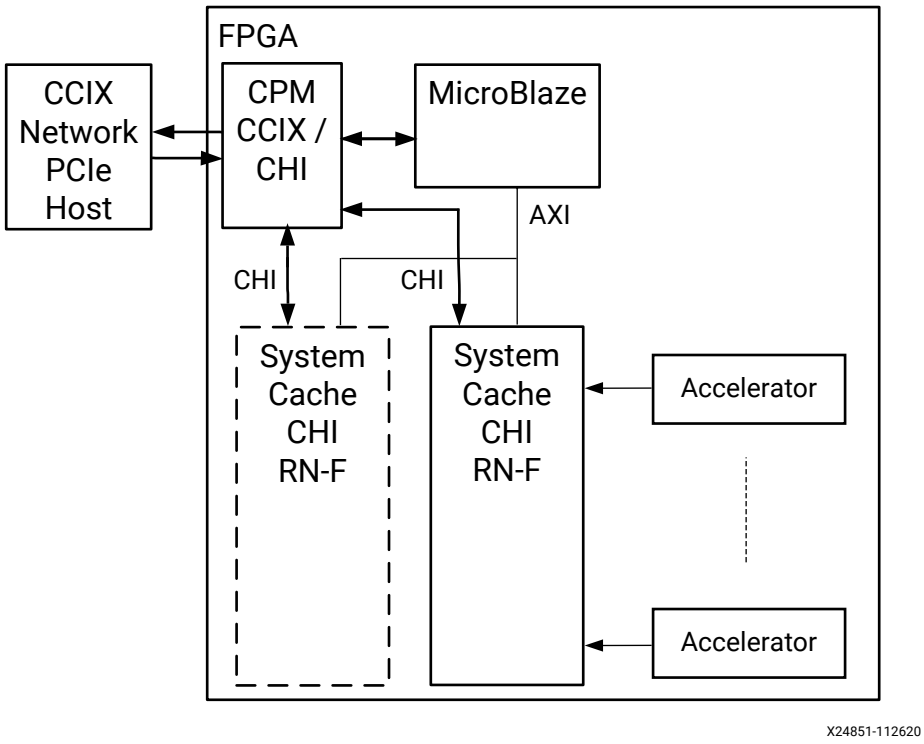


X25791-102521

CHI Example

System Cache configured to use CHI Master Coherency enables cache coherency similar to CCIX. In this case System Cache is connected to CPM in Versal, which has the ability to extend the coherency domain to enable cache coherency with memory distributed throughout the system, all controlled by a PCIe host. In this use case the accelerators connect to System Cache using AXI4 and System Cache provides a local cache function coherent with the rest of the CHI/CCIX memory systems, without any need for the accelerators to handle the coherency protocol. CPM has the ability to connect to two System Cache cores with CHI enabled.

Figure 5: **CHI Request Node (RN-F) System**



In the CHI configuration up to four accelerators/kernels can be used, connected to System Cache with ordinary AXI4 interfaces. The system can be configured to use Shared Virtual Memory, SVM, via the Address Translation Services, ATS. CPM supports the ability to use PASID, which enables each accelerator to use a unique ID when requesting address translation. A MicroBlaze processor subsystem handles all the configuration and maintenance to make System Cache operate as a full member of the CHI network.

Key building blocks in this solution are:

- Optional PCIe Host with CCIX Home Agent
- Versal CPM for CHI connection and potential connection to CCIX domain
- One or two System Cache cores configured as Request Node using the CHI protocol to keep the local cache coherent with the rest of the memory system, and the Address Translation Cache (ATC) to handle virtual address translation with PASID
- MicroBlaze processor based sub-system controlling System Cache to provide configuration and control connection to rest of the system
- Up to four accelerators connected via AXI4 to each System Cache

Figure 6: **CHI Component Overview**

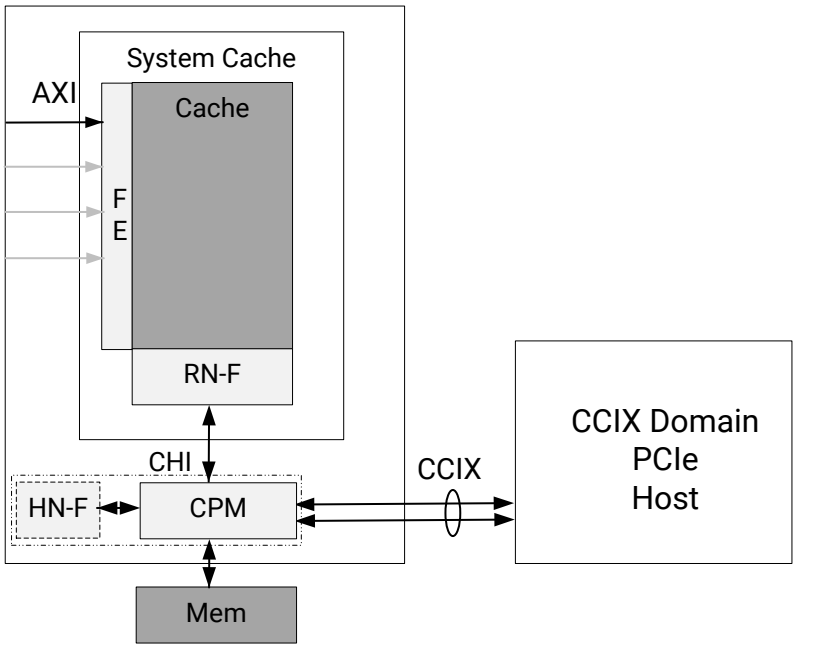
Front end (FE), handles arbitration between accelerators

Request Node (RN-F), performs read and write transactions to different addresses

CPM, gateway between CHI and CCIX Domain

Optional HA, (HN-F), CPM configuration, is responsible for memory address range in system

Memory (Mem)



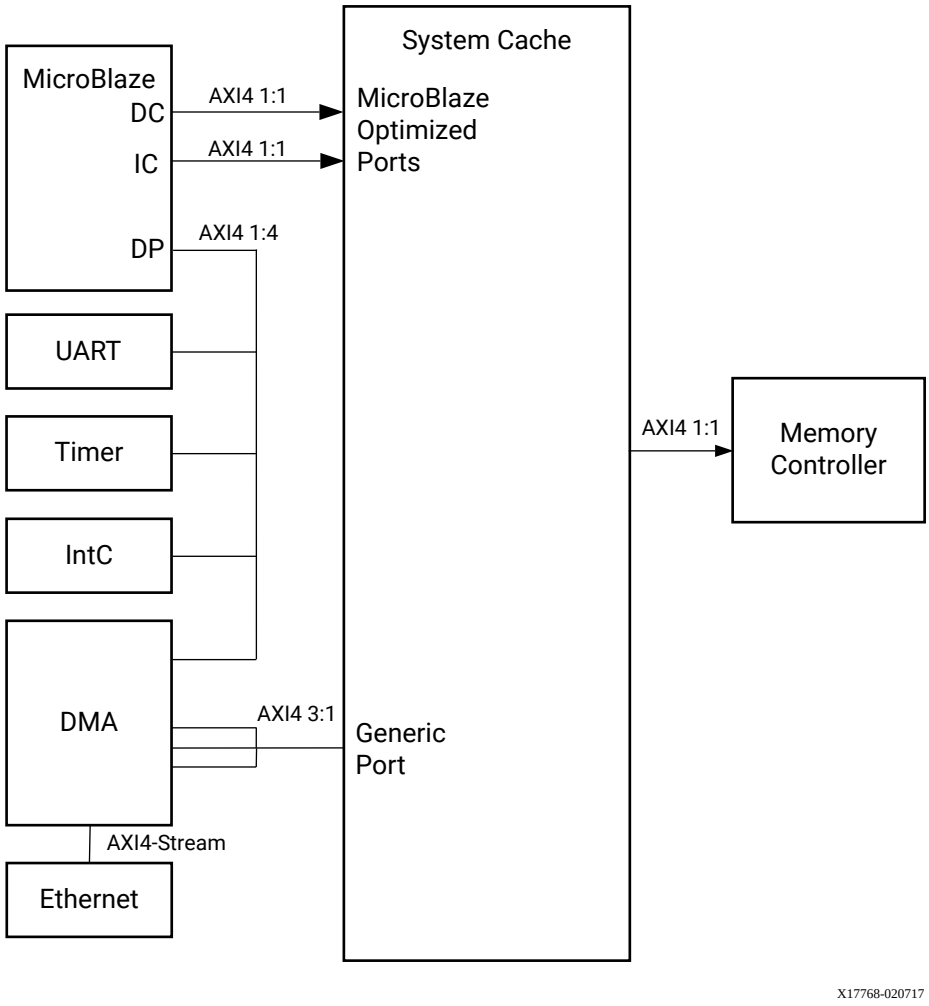
X24852-102521

Pure AXI Example

An Ethernet communication system example is shown in the following figure. The system consists of a MicroBlaze processor connected point-to-point to two optimized ports of the System Cache core. A DMA controller is connected to a generic port on the System Cache core through a 3:1 CCIX Component Overview interconnect, because the DMA controller has three master ports. The DMA in turn is connected to the Ethernet IP core using an AXI4-Stream MicroBlaze processor peripheral data port (`M_AXI_DP`) for register configuration and control interface. Standard peripheral functions such as a UART, timer, interrupt controller as well as the DMA controller control port are connected to the MicroBlaze processor peripheral data port (`M_AXI_DP`) for register configuration and control.

With this partitioning the bandwidth critical interfaces are connected directly to the System Cache core and kept completely separated from the AXI4-Lite based configuration and control connections. This system is used as an AXI-specific example throughout the documentation.

Figure 7: **Ethernet Systems**



In this example, the MicroBlaze processor is configured for high performance while still being able to reach a high maximum frequency. The MicroBlaze processor frequency is mainly improved due to small cache sizes, implemented using distributed RAM.

The lower hit rate from small caches is mitigated by the higher system frequency and the use of the System Cache core. The decreased hit rate in the MicroBlaze processor caches is compensated by cache hits in the System Cache core, which incur less penalty than accesses to external memory.

Write-through data cache is enabled in the MicroBlaze processor which, in the majority of cases, gives higher performance than using write-back cache when MicroBlaze processor L1 caches are small. The reverse is usually true when there is no System Cache core, or when MicroBlaze processor L1 caches are large. Finally, victim cache is enabled for the MicroBlaze processor instruction cache, which improves the hit rate by storing the most recently discarded cache lines.

All AXI4 data widths on the System Cache core ports are matched to the AXI4 data widths of the connecting modules to avoid data width conversions, which minimizes the AXI4 interconnect area overhead. The AXI4 1:1 connections are only implemented as routing without any logic in this case. All AXI4 ports are clocked using the same clock, which means that there is no need for clock conversion within the AXI4 interconnects. Avoiding clock conversion gives minimal area and latency for the AXI4 interconnects. The parameter settings for the MicroBlaze processor and the System Cache core can be found in the following tables.

Table 1: **MicroBlaze Processor Parameter Settings for the Ethernet System**

Parameter	Value
C_CACHE_BYTE_SIZE	512
C_ICACHE_ALWAYS_USED	1
C_ICACHE_LINE_LEN	8
C_ICACHE_STREAMS	0
C_ICACHE_VICTIMS	8
C_DCACHE_BYTE_SIZE	512
C_DCACHE_ALWAYS_USED	1
C_DCACHE_LINE_LEN	8
C_DCACHE_USE_WRITEBACK	0
C_DCACHE_VICTIMS	0

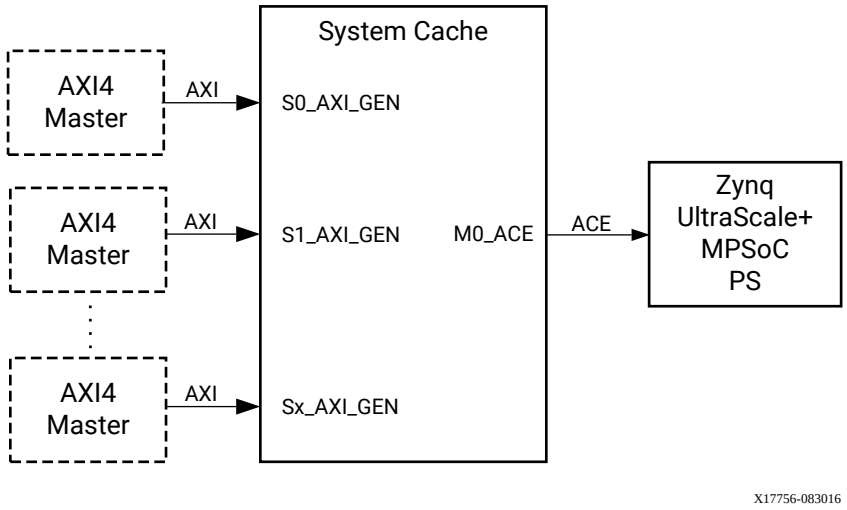
Table 2: **System Cache Parameter Settings for the Ethernet System**

Parameter	Value
C_NUM_OPTIMIZED_PORTS	2
C_NUM_GENERIC_PORTS	1
C_NUM_WAYS	4
C_CACHE_SIZE	65536
C_M_AXI_DATA_WIDTH	32

ACE Example

Another example use case, shown in the following figure, is a set of accelerators connected through the System Cache core to the ACE port on a Zynq UltraScale+ MPSoC. To fully take advantage of the System Cache, AXI transactions from the accelerators should be set up as Write-Back memory type (`ARCACHE` and `AWCACHE`), preferably Write-back Read and Write-allocate. If it is not possible to directly control this from an accelerator, it is possible to override some of the `AxCACHE` function through parameters such as `C_Sx_AXI_GEN_FORCE_WRITE_ALLOCATE` on a per port basis. This override functionality is available on all ports including the optimized ports.

Figure 8: **Example of a System with Two or More Accelerators**



It is possible to connect one or more MicroBlaze processor caches to the optimized ports, but they will not be cache coherent with the Zynq UltraScale+ MPSoC Processing System (PS) so manual cache maintenance with WIC and WDC type instructions is needed to observe data.

Example parameters for the System Cache core in this kind of configuration can be found in the following table. `Sx_AXI_GEN_*` should be configured for all active ports.

Table 3: **Example System Cache Parameters for Accelerator Configuration**

Parameter	Value
C_NUM_OPTIMIZED_PORTS	0
C_NUM_GENERIC_PORTS	2 or more
C_NUM_WAYS	4
C_CACHE_SIZE	131072
C_M_AXI_DATA_WIDTH	128
C_ENABLE_COHERENCY	2
C_ENABLE_NON_SECURE	1
C_ENABLE_ERROR_HANDLING	1
C_Sx_AXI_GEN_DATA_WIDTH	128
C_Sx_AXI_GEN_FORCE_READ_ALLOCATE	1
C_Sx_AXI_GEN_PROHIBIT_READ_ALLOCATE	0
C_Sx_AXI_GEN_FORCE_WRITE_ALLOCATE	1
C_Sx_AXI_GEN_PROHIBIT_WRITE_ALLOCATE	0
C_Sx_AXI_GEN_FORCE_READ_BUFFER	1
C_Sx_AXI_GEN_PROHIBIT_READ_BUFFER	0
C_Sx_AXI_GEN_FORCE_WRITE_BUFFER	1
C_Sx_AXI_GEN_PROHIBIT_WRITE_BUFFER	0

All the C_Sx_AXI_PROHIBIT_WRITE_ALLOCATE/ C_Sx_AXI_GEN_PROHIBIT_WRITE_ALLOCATE parameters are cleared by default and need to be set to disable allocation on Write Miss. This is not backwards compatible with earlier System Cache versions.

Unsupported Features

The following features of the standards are not supported in the core:

CCIX RA Master Port Cache Coherency

When CCIX Master Port coherency is selected the Optimized slave ports are disabled. The Generic ports are also limited to a maximum of four ports.

CCIX optional CompAck removal is not supported.

CCIX optional support for 128 byte cache lines is not available, only 64 byte cache lines are supported.

CCIX optional Partial Cache States are not supported.

CHI Master Port Cache Coherency

When CHI Master Port coherency is selected the Optimized slave ports are disabled. The Generic ports are also limited to a maximum of four ports.

CHI optional Stashing feature is not supported.

CHI DVM feature is not supported. No DVM requests are generated, and SnpDVM is terminated but not acted upon.

CHI Partial Cache States are not supported.

AXI/ACE Slave Address Space

Fixed burst is not supported on any AXI or ACE port. System Cache only deals with regular memory handling, fixed burst is more relevant in a queue context.

Most significant address bit is not available dynamically, only to set upper or lower half statically, i.e. up to 63 address bits can be used freely from AXI Masters.

ACE Master Port Cache Coherency

When master port cache coherency is enabled coherency on the optimized ports is not supported. In this case, any MicroBlaze processor that is connected must perform manual cache maintenance operation with WIC and WDC instructions (usually through BSP function calls) in order to work reliably with the coherent domain. This usually also includes proper communication between the participants in the coherency domain and any MicroBlaze processor to synchronize for safe data exchange.

Non-Coherent Implementation

The System Cache core provides no support for coherency between the MicroBlaze processor internal caches when cache coherency is disabled. This means that software must ensure coherency for data exchanged between the processors. When the MicroBlaze processors use write-back data caches, all processors need to flush their caches to ensure that correct data is being exchanged. For write-through caches, it is only the processors reading data that need to invalidate their caches to ensure that correct data is being exchanged.

Optimized Port Cache Coherent Implementation

When optimized port cache coherency is enabled, cached masters connected through the generic AXI4 slave ports are not included in the coherency domain. The reason for this is that the connection is pure AXI and not ACE, so it is not possible to snoop any master connected to a generic port.

All writes from a generic port remove corresponding line(s) from any MicroBlaze processor cache connected to an optimized port so that the new data is visible to the MicroBlaze processor. A read gets a snapshot of the current value of the coherency domain; if this value is stored locally (cached) in the AXI master it is the responsibility of that master to perform proper cache maintenance to remain coherent.

When cache coherency is enabled, exclusive transactions from the generic ports are disabled, and treated as normal transactions. The reason for this is that only the ACE transaction-based method with snoop messages is supported when cache coherency is enabled.

Licensing and Ordering

This Xilinx® LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License.

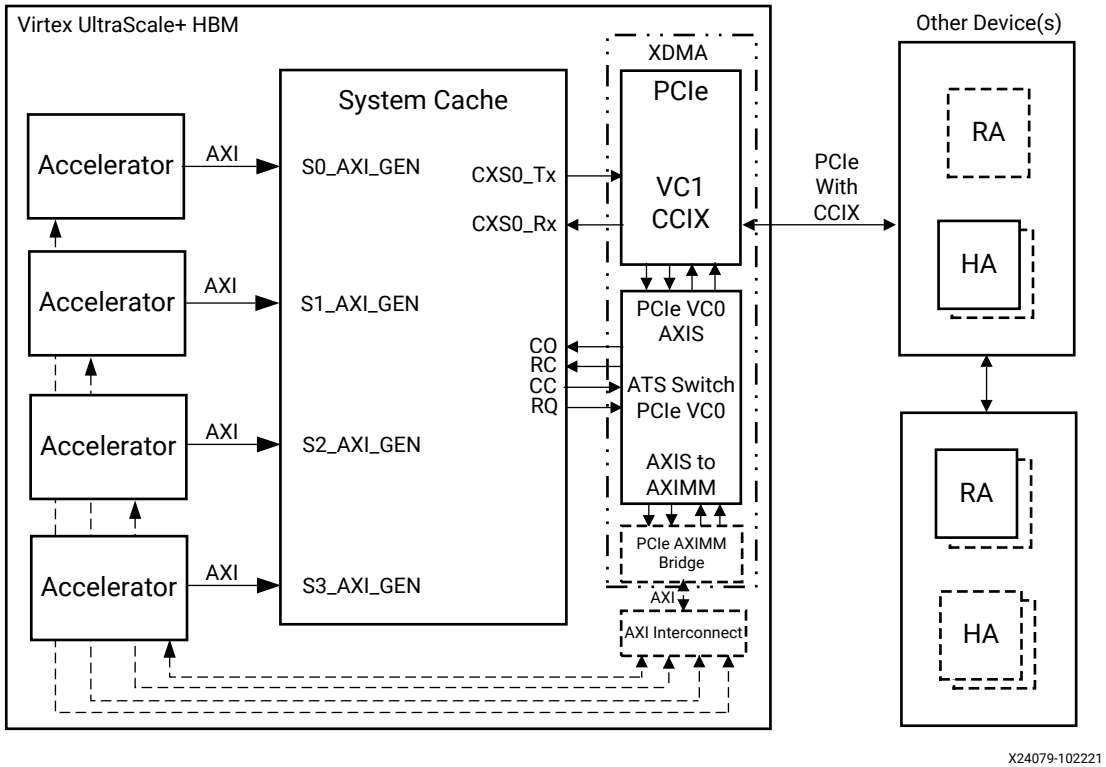
Information about other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

CCIX RA Master Port Cache Coherency

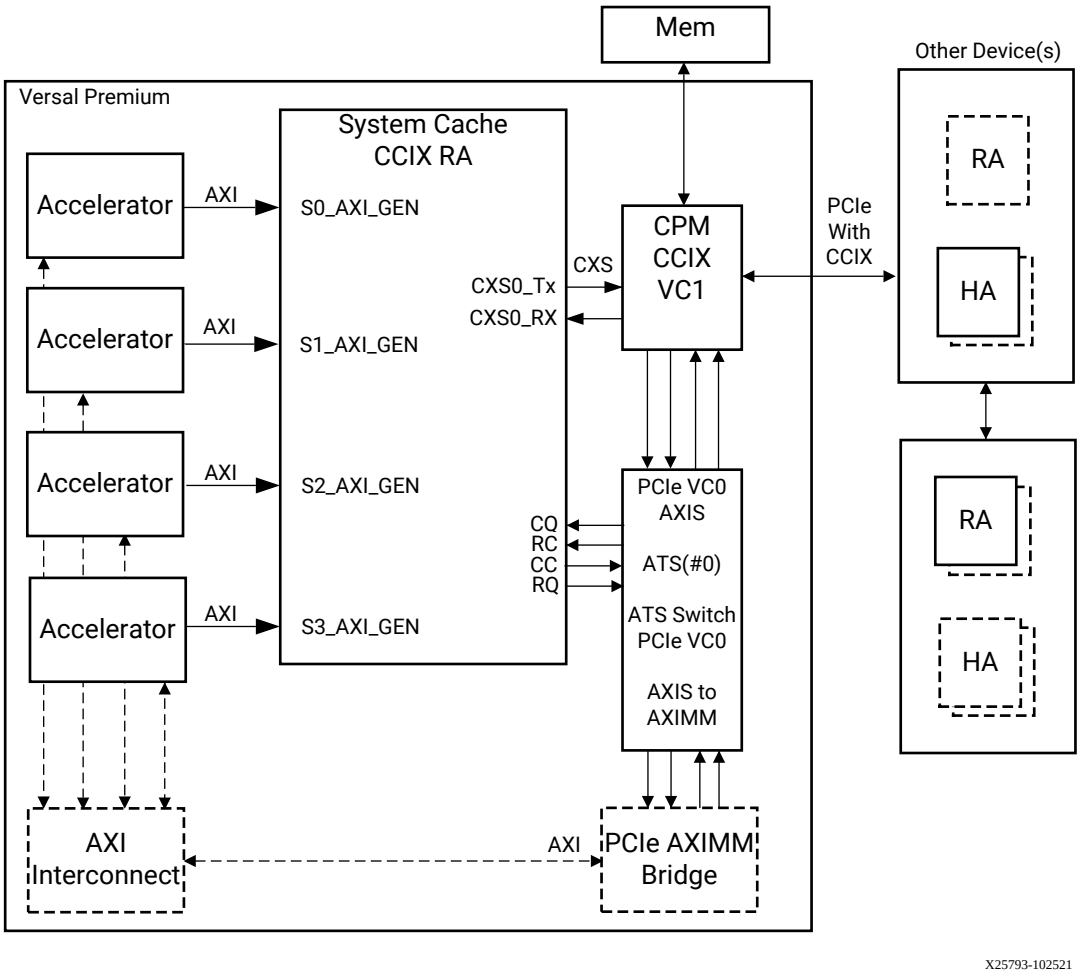
Cache Coherent Interconnect for Accelerators (CCIX®) cache coherency is used for multiple devices with system wide cache coherency. System Cache provides one Request Agent (RA) that can access multiple Home Agents (HA). It provides up to four AXI4 interfaces for accelerators to connect to as shown in the following figure.

Figure 9: **Typical CCIX RA XDMA Master Port Cache Coherency System**



System Cache connects to the PCIe® block with one CXS point-to-point interface in each direction and with four AXI4-Stream interfaces when ATS address translation is enabled. The FPGA is connected to other devices via PCIe, which will tunnel CCIX traffic on a Virtual Channel (VC). These devices must contain one or more HAs, and possibly even internal RAs, as well as connect to other devices with additional RAs and/or HAs.

Figure 10: **Typical CCIX RA CPM Cache Coherency System**

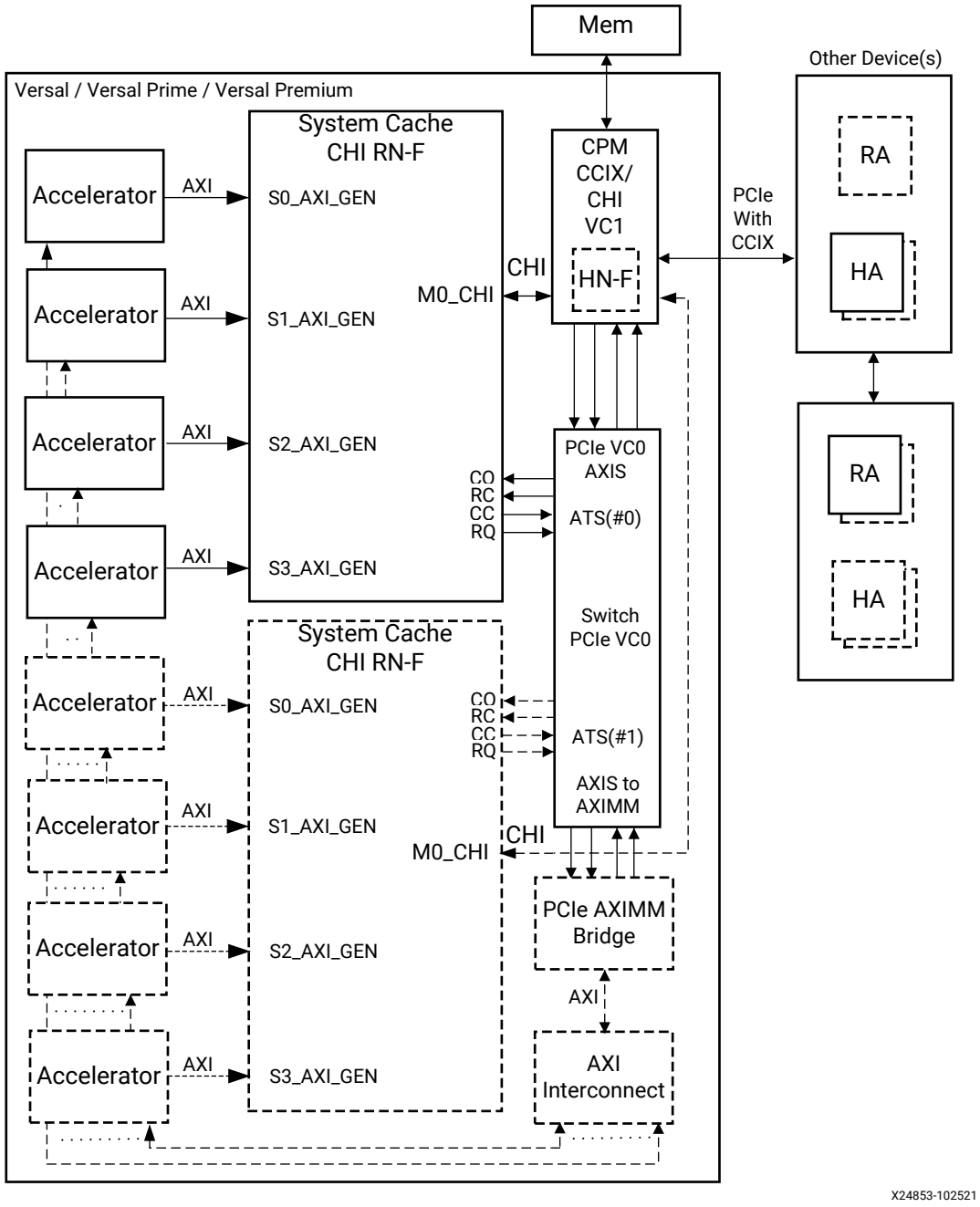


System Cache connects to the PCIe with a CXS/CCIX point-to-point interface similar to CCIX XDMA, with replacement of the XDMA PCIe block by the CCIX enabled CPM PCIe connectivity extension and with four AXI4-Stream interfaces when ATS address translation is enabled. Furthermore, the PASID Host propagation is supported, if enabled by Host support. An ATS switch is included to distribute ATS traffic to the System Cache instance, with dual clock configuration to allow System Cache single clock usage, even with configurable asynchronous PCIe clock domain.

CHI RN Master Port Cache Coherency

CHI is used for connection to local cache coherency in a device or system-wide cache coherency. System Cache provides one Request Node (RN-F) that can access the local Home Node (HN) in the Versal CPM and multiple Home Agents (HA) when CCIX is enabled externally in the CPM. Up to four AXI4 interfaces are provided per System Cache instance for accelerator connections as shown in the following figure.

Figure 11: **Typical CHI RN Master Port Cache Coherency System**



System Cache connects to the CPM block with one CHI point-to-point interface and with four AXI4-Stream interfaces when ATS address translation is enabled. The CPM has two CHI ports so that two System Caches can be connected to the same CPM. An ATS switch is included to distribute ATS traffic to all System Cache instances. The FPGA is connected to other devices via PCIe, which tunnels CCIX traffic on a Virtual Channel (VC). These devices must contain one or more HAs, and may possibly even have internal RAs, as well as connect to other devices with additional RAs and/or HAs.

Address Translation Service and Cache

There are various reasons for enabling System Cache address translation, including:

- Avoiding host device driver and letting accelerators work directly with addresses provided by the host application
- Limiting the impact of “memory leakage” or an incorrectly programmed endpoint
- Address space conversion (smaller endpoint address range to larger system virtual address space)
- Providing scatter/gather functionality
- Virtualization support

The System Cache includes an ATC function with companion ATS to support virtual address handling. The function is split into three parts:

- **ATC TLB:** ATC Translation Lookaside Buffer, located in each AXI4 Slave port interface, which holds the most recently used translations for AR and AW channel transactions respectively.
- **ATC Table:** ATC Translation Lookaside Buffer Cache, one common table with recently cached local copies of the Host TA Virtual to Physical Address map, with a default size of 256 entries.
- **ATS:** Address Translation Service, protocol based message service over PCIe Virtual Channel 0 (VC0), including message structures to support maintenance of ATC Table synchronization with the Host TA.

To enable ATS, configure System Cache with optional `C_ENABLE_ADDRESS_TRANSLATION` set. The ATS function is activated through the System Cache Capability registers and the PCIe configuration structure, mirrored in the System Cache register space.

If ATS is enabled, but not activated, System Cache will not issue any ATS Translation Requests and will use the “untranslated” physical addresses for all read and write requests.

If System Cache is configured with address translation disabled, the ATS AXI4-Stream interfaces are not visible on the System Cache IP core.

The System Cache translation process, using the ATC Table and the AR and AW channel ATC TLB for each of the AXI4 Slave ports provides:

- The ability to reduce look-up latency by distributing address translation caching
- Reduced probability of “thrashing” within the Host TA memory management entries
- Improved system performance by ATC TLB search, ATC Table locality, and reduced dependency on the Host TA latency
- Reduced latency by less frequent requests to the Host TA for missing address mappings

ATS uses a request-completion protocol between an endpoint, System Cache and the Host TA, to provide the translation service. ATS capabilities also include handling the Page Request Interface (PRI), to request the Host TA to map requested pages.

Because System Cache only expects ATS/PRI messages, any other messages on VC0 must be filtered out outside System Cache. Unexpected messages will be silently discarded, to avoid inconsistency or message blocking, causing data to be lost for any memory or I/O accesses.

The System Cache ATS interface consists of independent Request/Completion streams, two for Requester and two for Completer, each having programmable parity protection and checking on the AXI4-Stream interfaces.

- Incoming Completer Request (CQ) interface through which ATS Invalidation requests and PRI Response from Host TA are sent to the System Cache
- Outgoing Completer Completion (CC) interface through which the System Cache sends back responses to the completer requests (including AER error handling)
- Outgoing Requester Request (RQ) interface through which the System Cache generates translation requests to the Host TA, ATS Invalidation completions and PRI requests
- Incoming Requester Completion (RC) interface through which the translation completions are received from the Host TA in response to System Cache ATS TA requests

To allow ATC Table entries to use unique virtual address spaces, System Cache optionally allows a Process Address Space ID (PASID) extension.

With CCIX protocol address translations, using a Host TA without PASID propagation, the virtualization with the PASID extension is added to the AXI4 Slave ports to allow different processes to use dedicated translation mappings in the ATC over time, which will not propagate PASID on the AXI4-Stream interfaces.

With CHI protocol address translation and CCIX protocol address translation for supported technologies, the added PASID extension also propagates the PASID to the Host TA via the AXI4-Stream interfaces.

When Host TA PASID propagation is enabled the page attributes `Privileged` and `Execute` options are propagated from Host TA together with normal Read/Write attributes, as well as the translated page attributes `Global Mapping` (same TA page mapping regardless of PASID) and `Global Invalidate` (invalidate all TA pages in ATC table regardless of PASID for given TA).

AXI Master

In a typical system with one MicroBlaze™ processor, as shown in the following figures, the instruction and data cache interfaces (`M_AXI_IC` and `M_AXI_DC`) are connected to dedicated AXI4 interfaces optimized for MicroBlaze on the System Cache core. The System Cache core often makes it possible to reduce the MicroBlaze internal cache sizes without reducing system performance. Non-MicroBlaze AXI4 interface masters are connected to one or more of the generic AXI4 slave interfaces of the System Cache core either through an AXI4 interconnect or directly as shown in the following figures.

Figure 12: **Typical System with a Single Processor**

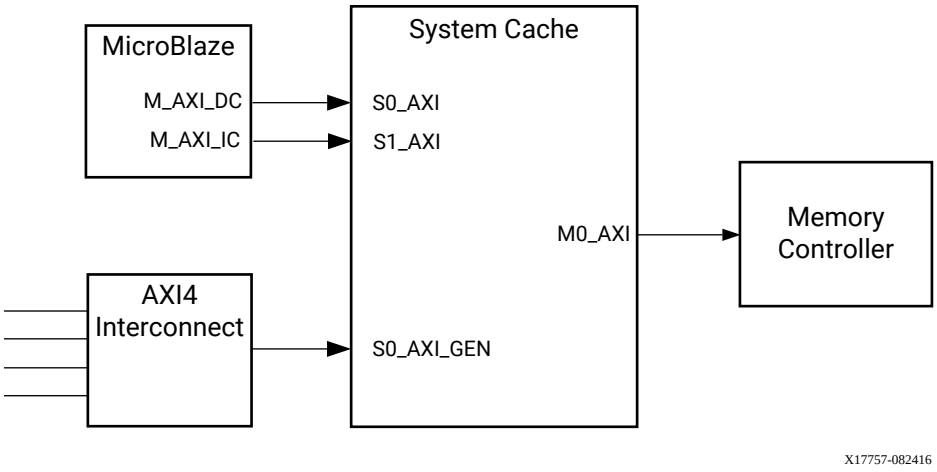


Figure 13: **Typical System with a Single Processor**

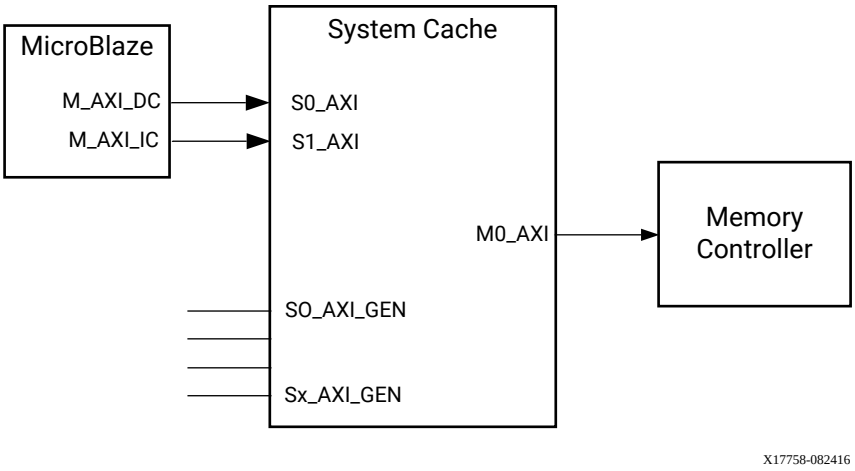
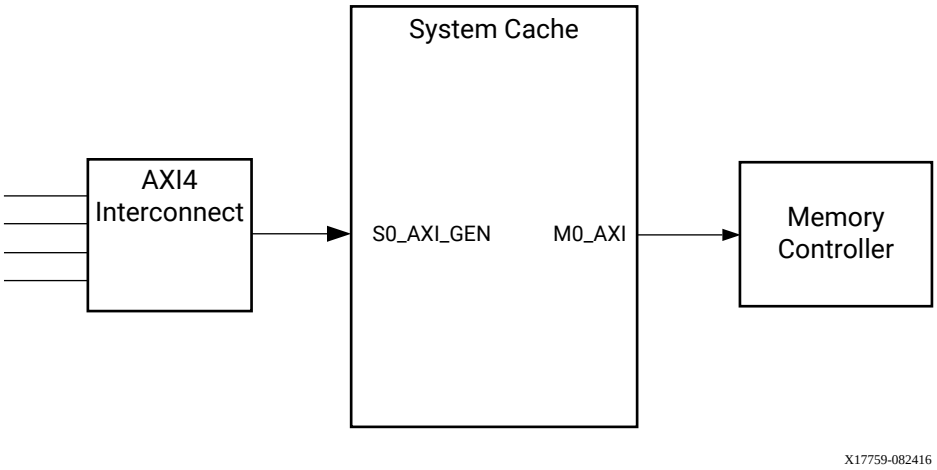
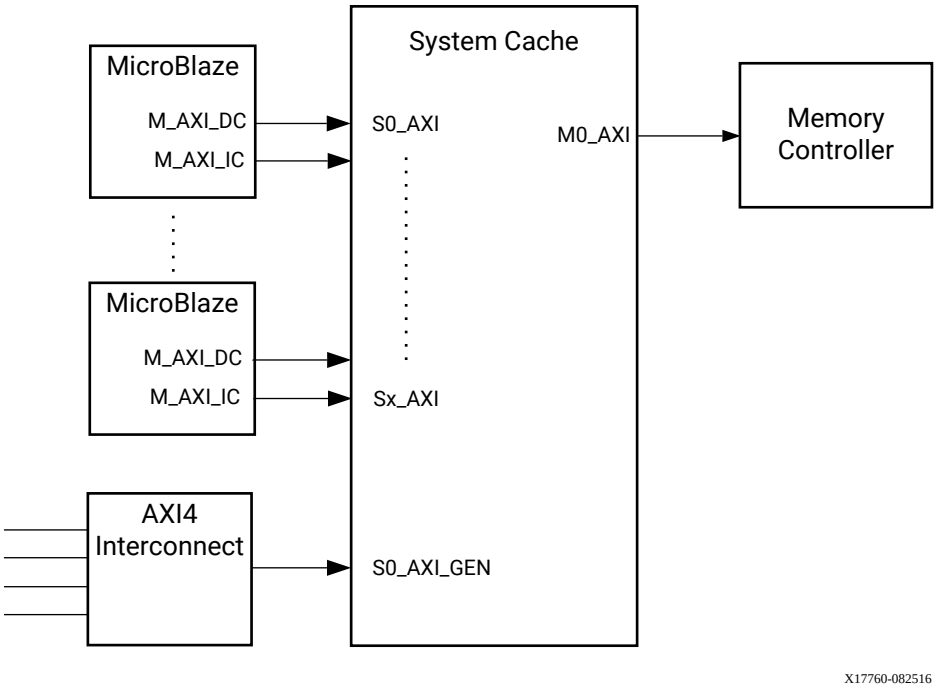


Figure 14: **System without Processor**



The System Cache core has 16 cache interfaces optimized for MicroBlaze, enabling direct connection of up to eight MicroBlaze processors, as shown in the following figure.

Figure 15: **Typical System with Multiple MicroBlaze Processors**



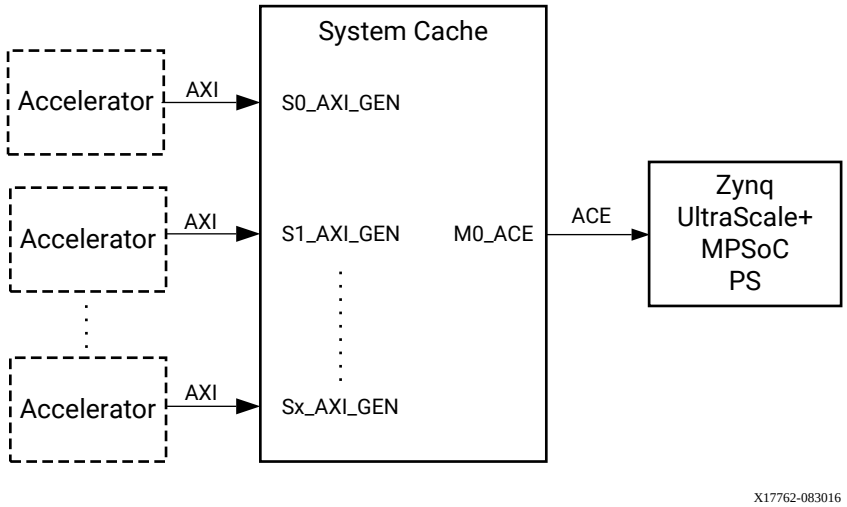
ACE Master Port Cache Coherency

ACE Master port cache coherency is used to support an arbitrary AXI4 master, primarily intended for accelerators, with a system cache that is coherent to the Zynq® UltraScale+™ MPSoC PS caches. All coherency is handled seamlessly from the point of view of the AXI master. The only software required is the regular set up of memory types and modes in the APU.

The R5 real-time processors need software interaction to accurately see the latest data because only the A53s have hardware support for cache coherency to the programmable logic (PL).

Up to 16 accelerators can be connected directly to the System Cache core which provides fast access to memory that is coherent to the PS caches in the Zynq UltraScale+ MPSoC, as shown in the following figure.

Figure 16: **Typical System with Multiple Accelerators**



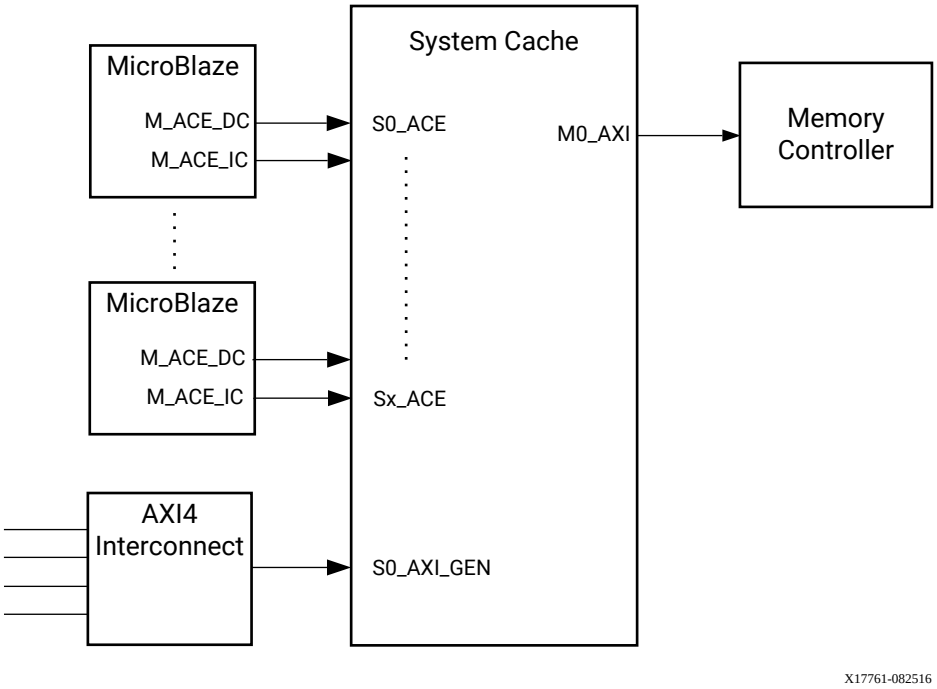
Optimized Ports Cache Coherency

Optimized port cache coherency support is used to enable data and instruction cache coherency for multiple MicroBlaze cores. It provides reliable exclusive transactions to implement software spinlocks and simplifies multi-processor (MP) systems where data is shared among the processors. Any data that is updated from one MicroBlaze is guaranteed to be seen from its peer processors without any special software interactions other than ordinary single MicroBlaze rules

for handling self-modifying code. This data manipulation information exchange is handled by the snooping mechanism provided by the AXI Coherency Extension (ACE) (see the *AMBA AXI and ACE Protocol Specification*). Distributed Virtual Memory (DVM) messages are also available with ACE to ensure that memory management units (MMU) and branch target caches are updated across the system when related changes are performed by any of the connected processors.

The 16 cache interfaces optimized for MicroBlaze provide support for up to eight cache coherent MicroBlaze processors, as shown in the following figure.

Figure 17: **Typical System with Multiple Coherent MicroBlaze Processors**



Exclusive Monitor

The optional exclusive monitor provides full support for exclusive transactions when cache coherency is disabled. It supports exclusive transactions from both generic and optimized ports.

Cache Memory

The cache memory provides the actual cache functionality in the core. The cache is configurable in terms of size and associativity.

The cache size can be configured with the parameter `C_CACHE_SIZE` according to User Parameters. The selected size is a trade-off between performance and resource usage, in particular the number of on-chip RAMs.

The associativity can be configured with the parameter `C_NUM_WAYS` according to User Parameters. Increased associativity generally provides better hit rate, which gives better performance but requires more area resources.

The type of memory used for Tags, Data and LRU can be configured with the parameters `C_CACHE_TAG_MEMORY_TYPE`, `C_CACHE_DATA_MEMORY_TYPE` and `C_CACHE_LRU_MEMORY_TYPE` respectively. The possible values are `Automatic`, `LUTRAM`, `BRAM` and `URAM`, except that `LUTRAM` cannot be selected for Data. Additionally, it is only possible to select `URAM` for UltraScale+ devices. The default configuration, `Automatic`, is generally recommended, but for larger cache sizes selecting `URAM` for Data (if possible) will provide better resource utilization.

The correspondence between selected parameters and on-chip RAMs used can be found in Performance and Resource Utilization.

Related Information

User Parameters

Error Handling

Different levels of support for error handling are available depending on the configuration, all the way from feature-rich for CCIX down to bare minimum or none for pure AXI4.

With the Address translation option, the PCIe AER can be enabled extending AXI4-Stream data integrity.

CCIX Error Handling

CCIX error handling expands on the AXI error handling and extends it to full Reliability, Availability and Serviceability (RAS) support, according to the CCIX protocol. It has all the filtering and control options in order to tailor the kind of errors that need to be handled.

The Protocol Error Reporting (PER) messages sent and error logs created depend on the configuration defined by control registers.

CHI Error Handling

CHI Error handling is based on CCIX, with the same structure used for PER Log but without the PER message part. In addition the CHI specific error handling is used.

ATS/ATC Error Handling

ATS/ATC error handling expands on the PCIe, Requester and Completer AXI4-Stream, error handling and extends it to full AER support. Generated AER messages and created error logs depend on the configuration defined by control registers.

Internal ATC memory structures are protected by optional parity. All internal ATC memory parity errors are reported as internal ATC correctable errors via PER logging, which is propagated in CCIX configurations.

- Protocol errors reported via the Advanced Error Reporting Extended Capability Structure:
  1. Unsupported Request (UR), Completer Abort (CA), or Unexpected Completion: AER is propagated by the ATS interface
  2. AXI4-Stream parity error: When parity check is enabled, AER is propagated by the ATS interface and the TLP is aborted
  3. Malformed TLP detected by the PCIe core: AER is reported by the core and the ongoing translation results in Decode Error on the generic AXI port BRESP
  4. Illegal TAG usage, unsuccessful or erroneous completions detected by the PCIe core: AER is reported by the core and the ongoing translation results in Decode Error on the generic AXI port BRESP
  5. Requester channel timeout detected by the PCIe core: AER is reported by the core
  6. Protocol Completer channel timeouts: AER is propagated by the ATS interface and the ongoing translation results in Decode Error on the generic AXI port BRESP
- Protocol Error not reported via AER:
  1. Receiver overflow: Error should avoided by configuration and when full or not ready the ATS interface does not accept additional data on the AXI4-Stream interface
  2. ECRC Check Failed: ECRC is not supported and System Cache must be configured with the TD bit cleared
  3. Header Discontinue, Poisoned TLP Received: Transaction is aborted and no AER is propagated by the ATS interface

The ATS/ATC error handling is enabled with the `C_ATS0_CQ_CC_ENABLE_AER` parameter for Completer channels and `C_ATS0_RQ_RC_ENABLE_AER` for Requester channels.

When ATS AER handling is enabled a detected error will be logged and propagated to the PCIe Host and the Cmd/Cpl (header) and/or Payload (data) affected will be discarded.

In case CA/UR, Sequence and TLP incompleteness are detected on the RC channel, errors signaled over AXI4-Stream are logged and reported, but error handling relies on the PCIe core reporting of AER external to System Cache for complementary error logs.

The ATC integrity error handling feature is controlled by the configuration parameter `C_ENABLE_INTEGRITY`, also used for all other System Cache integrity functionality.

In case of ATC integrity error detection, caused by a Single Event Upset (SEU) or by integrity function fault injection, the affected ATC table entry is silently invalidated to avoid undetectable multiple error accumulation in the memory.

ATC Table look-up, in case of an ATC TLB miss, blocks any attempt to use table entries with parity errors. The automatic background scrubbing or manually initiated scrubbing then removes any pending errors from the ATC Table, with silent invalidation.

Also note that any normal Host TA commanded invalidation, for any address range, causes pending errors to be removed.

Integrity error detection and removal is logged and reported via PER.

ATC integrity handling uses the same structure on CHI as for CCIX, with PER Log but without the PER message propagation.

With CHI, the interrupt event notifies the software error manager to poll the PER log

An ATC integrity error is correctable on system level, since the PCIe host TA holds all translations cached by the ATC Table. Any invalidation due to an integrity error will cause a new ATS TA request.

See the PCI Express® Base Specification for the AER structure definition, and CCIX® Base Specification for the PER message definition.

AXI4/ACE Error Handling

The optional AXI4/ACE error handling is enabled with the `C_ENABLE_ERROR_HANDLING` parameter. It allows the refusal of the allocation of a line if there is an decode or slave error when a line is fetched from external memory.

Non-Secure Handling

The optional handling of Secure/Non-Secure can be enabled with the `C_ENABLE_NON_SECURE` parameter. When active the `AxPROT[1]` bit is treated as an extra address bit to provide a distinction between the two modes. This also means that the same address can be cached as both Secure and Non-Secure at the same time.

Additional control registers are available when this feature is enabled to allow command and control of the System Cache core, distinguishing between the different modes.

Note: When enabling Address Translation, the same Host TA Address map will be used. The need to keep a unique mapping per mode must be resolved on the system level in host TA, by invalidation and address remapping.

Control and Statistics

A minimal set of control and statistics is always available. When CCIX is enabled the set of available control and statistics features is expanded.

CCIX Control

Typically CCIX is controlled by firmware executing on a dedicated MicroBlaze processor. Features that can be controlled are:

- Request Agent on/off
- Link on/off including the entire sequence of bringing up and taking down the link
- Port on/off
- Snoop response behavior
- Cache Maintenance Operations (CMO) event properties

CHI Control

Like CCIX, CHI is typically controlled by firmware executing on a dedicated MicroBlaze processor. Features that can be controlled are:

- Request Node on/off
- Link on/off including the entire sequence of bringing up and taking down the link
- System Coherency on/off
- Snoop response behavior
- Cache Maintenance Operations (CMO) event properties

ATS/ATC Control

When Address Translation with ATS is enabled, for CHI and CCIX, the set of available control and statistics features is expanded to configure, control and gather statistics.

ATS control is needed to configure, control and report status to the PCIe root complex (host TA), to simplify system configuration and bring up; normally all PCIe endpoint configuration, enumeration and operational control are handled by firmware.

Features available for control are:

- ATS propagation on/off
- PRI propagation on/off
- PRI Retry capability on/off
- PRI Retry threshold level, if PRI Retry capability on
- TAG range support
- Memory attributes on/off (propagated from Host TA override defaults)
- PASID on/off (when enabled - both CCIX and CHI)
- Page attributes Privilege Enable/ Execute Enable on/off (with Host TA PASID propagation enabled)
- Global Invalidate Enabled on/off (with Host TA PASID propagation enabled)

AXI4/ACE Control

The following types of control and configuration information are available:

- Control registers for Flush and Clear cache maintenance
- Version Registers with System Cache core configuration

Statistics

The optional Statistics and Control block can be used to collect cache statistics such as cache hit rate and access latency. The statistics are primarily intended for internal Xilinx use, but can also be used to tailor the configuration of the System Cache core to meet the needs of a specific application. The following types of statistics are collected:

- Port statistics for each slave interface
  - Total Read and Write transaction counts
  - Port queue usage for the six transaction queues associated with each port
  - Cache hit rates for read and write
  - Read and Write transaction latency
  - Total ATC Read and Write transaction counts
  - ATC TLB Address hit rates for read and write

    - ATC TLB Read and write latency
    - ATC TLB Read and write LRU replacement ratio
- Arbitration statistics
- Functional unit statistics
    - Stall cycles
    - Internal queue usage
- Port statistics for the master interface
    - Read and write latency
- CCIX Backend
    - Message count
    - TLP count
    - Credit count
- CHI Backend
    - Message count
    - Credit count
- ATS
    - ATC Table Address hit rates for read and write
    - ATC Table Read and write latency
    - ATC Table Read and write LRW replacement ratio
    - ATC Table PRI transaction Retry counts
    - ATC Table PRI transaction TimeOut counts
    - ATC Table TA transaction Fail counts
    - ATC Table Invalidation transaction counts

Standards

The System Cache core adheres to:

- AMBA® AXI4 and ACE interface standard (see Arm® AMBA AXI Protocol Specification, Version 2.0 ARM IHI 0022E)

- AMBA AXI4-Stream interface standard (see Arm AMBA 4 AXI4-Stream Protocol Specification, Version 1.0 ARM IHI 0051A)
- AMBA® CHI interface standard (see AMBA® 5 CHI Architecture Specification, Issue B ARM IHI 0050B)
- AMBA CXS interface standard (see Arm AMBA CXS Protocol Specification, ARM IHI 0079A)
- CCIX Cache Coherent Interconnect for Accelerators standard (see CCIX Base Specification Revision 1.1 Version 1.0)
- PCI Express® standard (see PCI Express Base Specification Revision 5.0 Version 1.0)
- PCI Express ATS Memory Attributes (see ATS Memory Attributes ECN, Revision A)

Performance

The perceived performance is dependent on many factors such as frequency, latency and throughput. Which factor has the dominating effect is application-specific. There is also a correlation between the performance factors; for example, achieving high frequency can add latency and also wide datapaths for throughput can adversely affect frequency.

Read latency is defined as the clock cycle from the read address is accepted by the System Cache core to the cycle when first read data is available.

Write latency is defined as the clock cycle from the write address is accepted by the System Cache core to the cycle when the BRESP is valid. These calculations assume that the start of the write data is aligned to the transaction address.

Snoop latency is defined as the time from the clock cycle a snoop request is accepted by the System Cache core to the cycle when CRRESP or CDDATA is valid, whichever is last. Not all snoops result in a CDDATA transaction.

Maximum Frequencies

For details about performance, visit Performance and Resource Utilization.

CCIX Cache Latency

CCIX latency calculations are in principle defined in the same way as above in the introduction, but the time in flight on the PCIe bus is not included. Also, no additional delays due to ATS/ATC virtual address translation is included.

For the different types of transactions, this means:

- Read: From AXI ARVALID to start of TLP request + start of TLP response to AXI RRESP
- Write: From AXI AWVALID to start of TLP request + start of TLP response to AXI BRESP

- Snoop: From start of TLP request to start of TLP response
- Scrub (automatic): From timer initiation of the scrub to completion
- Scrub (manual): From AXI control write initiation of the scrub to completion

The latency depends on many factors such as traffic from other ports and conflict with earlier transactions. The numbers in the following table assume a completely idle System Cache core.

Table 4: **System Cache CCIX Latency**

Type	CCIX Latency
Read Hit	16
Read Miss	45 + round-trip delay on PCIe for request
Read Miss Dirty	max of: 45 + round-trip delay on PCIe for read request 39 + round-trip delay on PCIe for write request
Write Hit	16
Write Miss	44 + round-trip delay on PCIe for request
Write Miss Dirty	max of: 45 + round-trip delay on PCIe for read request 39 + round-trip delay on PCIe for write request
Snoop (missing broadcast)	23
Snoop	26
Snoop with data	33
Scrub (automatic)	11
Scrub (manual)	17

CHI Cache Latency

CHI latency calculations are defined in a similar way as CCIX, but the time in flight in the CHI domain is not included. Also, no additional delays due to ATS/ATC virtual address translation is included.

For the different types of transactions, this means:

- Read: From AXI ARVALID to request FLIT + response FLIT to AXI RRESP
- Write: From AXI AWVALID to request FLIT + response FLIT to AXI BRESP
- Snoop: From start of Request FLIT to start of TLP response
- Scrub (automatic): From timer initiation of the scrub to completion
- Scrub (manual): From AXI control write initiation of the scrub to completion

The latency depends on many factors such as traffic from other ports and conflict with earlier transactions. The numbers in the following table assume a completely idle System Cache core.

Table 5: **System Cache CHI Latency**

Type	CHI Latency
Read Hit	16
Read Miss	34 + round-trip delay on CHI for request
Read Miss Dirty	max of: 32 + round-trip delay on CHI for read request 31 + round-trip delay on CHI for write request
Write Hit	16
Write Miss	32 + round-trip delay on CHI for request
Write Miss Dirty	max of: 32 + round-trip delay on CHI for read request 31 + round-trip delay on CHI for write request
Snoop (missing broadcast)	23
Snoop	26
Snoop with data	33
Scrub (automatic)	11
Scrub (manual)	17

ATS/ATC Latency

The inclusion of the Address Translation Service, ATS and the ATC TLB in the AXI port interfaces will add latency to the above AXI4/ACE Cache Latency, in most cases the locality will have minimal hit latency but miss latency are expected in restarted systems with no accumulated translation and in case of locality context change.

Read latency is defined from the read address is accepted by the System Cache core to the cycle when first read data is available via the Address Translation service.

Write latency is defined from the write address is accepted by the System Cache core to start of the write data is aligned to the transaction address via the Address Translation service.

For the transaction best case latency it is assumed that previous accesses have already used the address range, hence both read and write will result in ATC TLB hits.

When a miss occurs in the ATC TLB, it is assumed that the ATC Table has a copy of a valid Translation, latency is added with best-case and worst-case latency due to ATS search.

***Note*:** The worst-case latency is related to ATS Tables size and is depth dependent – here the default size of 256 entries is used.

The average expected for ATC Search latency depends upon temporal locality of addresses in use in the ATC Table, where the entries of the n last mapped translations are cached, and ATC search hits will be within these n translations.

In the case of ATC table miss, the latency will be extended with PCIe Root Complex, Host TA, translation latency, which are system level defined latency – outside System Cache definition scope.

The host TA best-case translation time is the round trip transaction latency from request to response plus the host TA lookup time. The time can be extended with one or more page request round trips plus host page management latencies, and finally another retry translation round trip host TA lookup.

ATC table lookup latency is two clock cycles best case, 256+2 clock cycles worst case, and n/2+2 clock cycles on average (assuming locality for the n last accesses and hit latency within n entries).

Table 6: **System Cache Core Latencies with Address Translation**

Type	AXI4 Port Latency with Address Translation Latency (See CHI/ CCIX Master port Read/Write Latency)
Read Hit/Miss, ATC TLB Hit	2 + Master port Read latency (Hit/Miss)
Read Hit/Miss, ATC TLB Miss, ATC Table Hit	3 + ATC table lookup + Master port Read latency (Hit/Miss)
Read Hit/Miss, ATC TLB Miss and ATC Table Miss	3 + ATC table lookup Worst + latency added by PCIe ATS lookup + Master port Read latency (Hit/Miss)
Write Hit/Miss, ATC TLB Hit	2 + Master port Write burst latency (Hit/Miss)
Write Hit/Miss, ATC TLB Miss, ATC Table Hit	3 + ATC table lookup + Master port Write burst latency (Hit/Miss)
Write Hit/Miss, ATC TLB Miss, ATS Table Miss	3 + ATC table lookup Worst + latency added by PCIe ATS lookup + Master port Write burst latency (Hit/Miss)

AXI4/ACE Cache Latency

Here latency is used as described in the introduction.

The latency depends on many factors such as traffic from other ports and conflict with earlier transactions. The numbers in the following table assume a completely idle System Cache core and no write data delay for transactions on one of the optimized ports. For transactions using a generic AXI4 port an additional two clock cycle latency is added.

Table 7: **System Cache Core Latencies for Optimized Port**

Type	Optimized Port Latency
Read Hit	6
Read Miss	7 + latency added by memory subsystem
Read Miss Dirty	Maximum of: 7 + latency added by memory subsystem 7 + latency added for evicting dirty data (cache line length * 32 / M_AXI Data Width)
Write Hit	3 + burst length
Write Miss	Non-bufferable transaction: 7 + latency added by memory subsystem for writing data Bufferable transaction: Same as Write Hit

Enabling optimized port cache coherency affects the latency and also introduces new types of transaction latencies. The numbers in the following table assume a completely idle System Cache core and no write data delay for transactions on one of the optimized ports. Transactions from a generic port still have two cycles of extra latency.

Table 8: **System Cache Core Latencies for Cache Coherent Optimized Port**

Type	Coherent Optimized Port Latency
DVM Message	9 + latency added by snooped masters
DVM Sync	12 + latency added by snooped masters
Read Hit	9 + latency added by snooped masters
Read Miss	10 + latency added by snooped masters + latency added by memory subsystem
Read Miss Dirty	Maximum of: 10 + latency added by snooped masters + latency added by memory subsystem 10 + latency added by snooped masters + latency added for evicting dirty data (cache line length * 32 / M_AXI Data Width)
Write Hit	Maximum of: 3 + burst length 6 + latency added by snooped masters
Write Miss	Non-bufferable transaction: 10 + latency added by snooped masters + latency added by memory subsystem for writing data Bufferable transaction: same as Write Hit

When master port cache coherency is enabled the System Cache core provides CRRESP and potential data as quickly as possible, but the response time varies according to the current state and transactions in flight, both internally and externally, as long as they have an effect on the System Cache state. See the following table for latency values.

Table 9: **Core Latency Values for Master Port Cache Coherency**

Type	Master Port Snoop Latency
Snoop Miss	3 + latency of any preceding snoop blocking progress 4 + latency of any preceding snoop blocking progress (if hazard with pipelined access) 5 + latency of any preceding snoop blocking progress + latency to compete active write with hazard
Snoop Hit	4 + latency to acquire data access + latency of any preceding snoop blocking progress 5 + latency of any preceding snoop blocking progress (if hazard with pipelined access) 5 + latency of any preceding snoop blocking progress + latency to complete active write with hazard

The numbers for an actual MicroBlaze application vary depending on access patterns, hit/miss ratio and other factors. Example values from a system (see Typical System with a Single Processor above) running the iperf network testing tool with the LWIP TCP/IP stack in raw mode are shown in the following four tables, where the first contains the hit rate for transactions from all ports, and the remaining show per port hit rate and latencies for the three active ports.

Table 10: **Application Total Hit Rate**

Type	Hit Rate
Read	99.82%
Write	92.93%

Table 11: **System Cache Hit Rate and Latencies for MicroBlaze D-Side Port**

Type	Hit Rate	Min	Max	Average	Standard Deviation
Read	99.68%	6	290	8	3
Write	96.63%	4	31	4	1

Table 12: **System Cache Hit Rate and Latencies for MicroBlaze I-Side Port**

Type	Hit Rate	Min	Max	Average	Standard Deviation
Read	9.96%	5	568	6	2
Write	N/A	N/A	N/A	N/A	N/A

Table 13: **System Cache Hit Rate and Latencies for Generic Port**

Type	Hit Rate	Min	Max	Average	Standard Deviation
Read	76.68%	7	388	18	13
Write	9.78%	6	112	24	5

Throughput

The System Cache core is fully pipelined and can have a theoretical maximum transaction rate of one read or write hit data concurrent with one read and one write miss data per clock cycle when there are no conflicts with earlier transactions.

This theoretical limit is subject to memory subsystem bandwidth, intra-transaction conflicts and cache hit detection overhead, which reduce the achieved throughput to less than three data beats per clock cycle.

Resource Use

The System Cache core uses instantiated block RAM or UltraRAM; how many varies with configuration. Distributed RAM is inferred by the RTL code and is also dependent on the configuration.

For full details about performance and resource use, visit the Performance and Resource Use web page.

Port Descriptions

CCIX Port Descriptions

The block diagram of the System Cache core configured for CCIX is shown in the following figure. All interfaces are compliant to AXI4, AXI4-Stream or CXS where applicable. The input signals ACLK and ARESETN implement clock and reset for the entire System Cache core.

Figure 18: **CCIX Block Diagram**

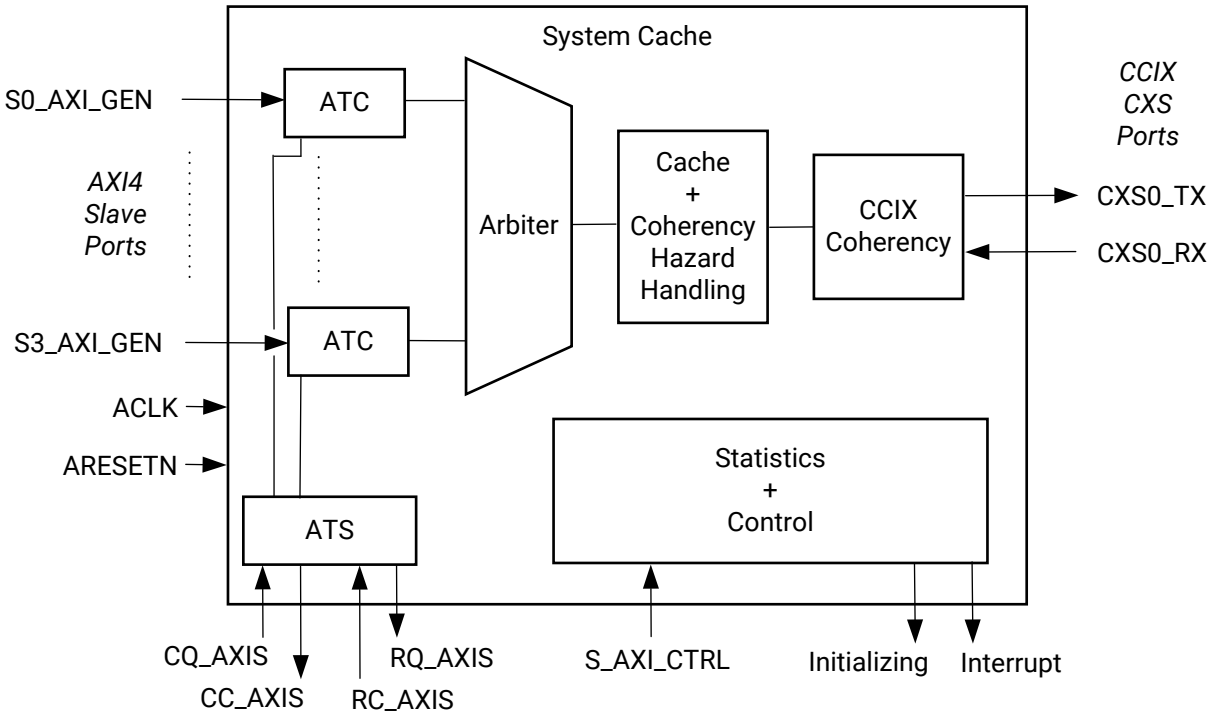


Table 14: **CCIX Interfaces**

Name	Type	Description
ACLK	Input	Core clock
ARESETN	Input	Synchronous reset of core
Initializing	Output	Core is initializing after reset
Sx_AXI_GEN[1]	AXI4 Slave	Generic cache port

Table 14: **CCIX Interfaces** *(cont'd)*

Name	Type	Description
S_AXI_CTRL	AXI4-Lite Slave	Control port
CXS0_RX	CXS Receive	CXS interface
CXS0_TX	CXS Transmit	CXS Interface
CQ_AXIS	AXI4-Stream Slave	ATS CQ Interface
CC_AXIS	AXI4-Stream Master	ATS CC Interface
RC_AXIS	AXI4-Stream Slave	ATS RC Interface
RQ_AXIS	AXI4-Stream Master	ATS RQ Interface
Interrupt	Output	Control interrupt output

Notes:
1. x = 0 to 3

CHI Port Descriptions

The block diagram of the System Cache core configured for CHI is shown in the following figure. All interfaces are compliant to AXI4, AXI4-Stream or CHI where applicable. The input signals `ACLK` and `ARESETN` implement clock and reset for the entire System Cache core. The M0_CHI interface contains all 6 channels and the separate signals to handle link and system coherency handshake.

Figure 19: **CHI Block Diagram**

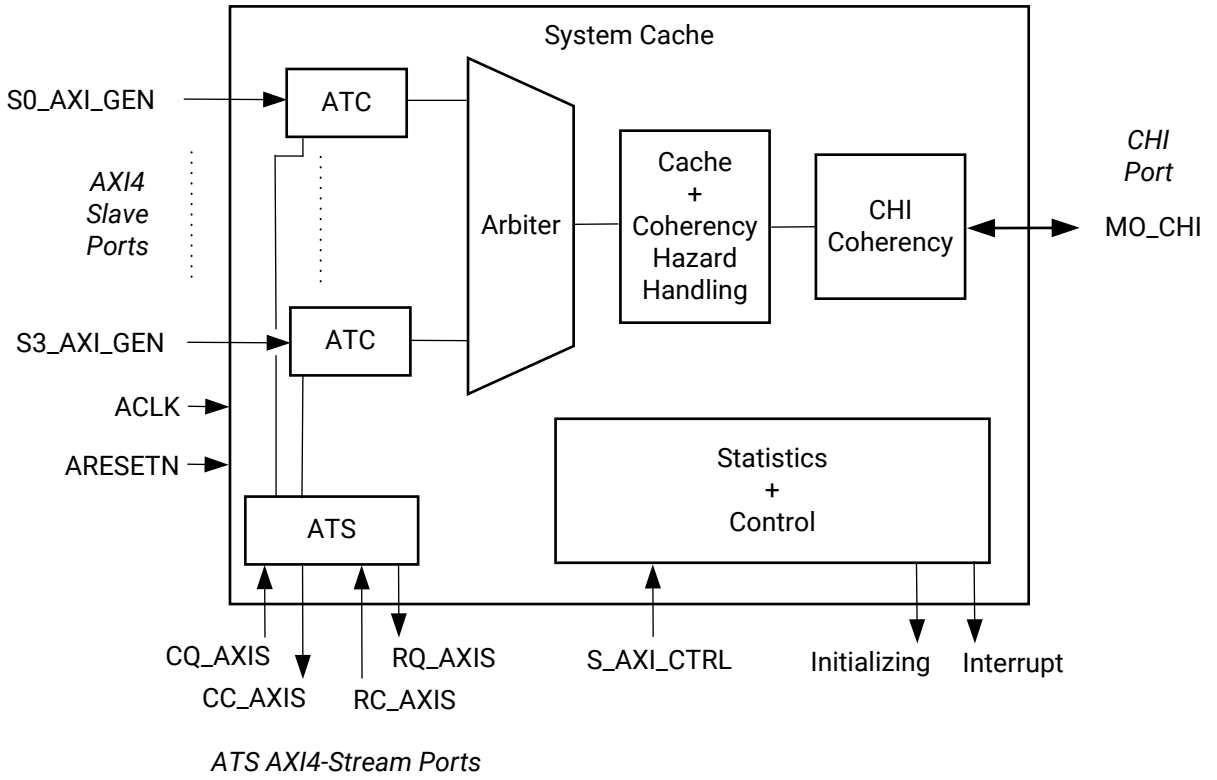


Table 15: **CHI Interfaces**

Name	Type	Description
ACLK	Input	Core clock
ARESETN	Input	Synchronous reset of core
Initializing	Output	Core is initializing after reset
Sx_AXI_GEN[1]	AXI4 Slave	Generic cache port
S_AXI_CTRL	AXI4-Lite Slave	Control port
M0_CHI	CHI	CHI Interface containing all 6 channels and extra signals
CQ_AXIS	AXI4-Stream Slave	ATS CQ Interface
CC_AXIS	AXI4-Stream Master	ATS CC Interface
RC_AXIS	AXI4-Stream Slave	ATS RC Interface
RQ_AXIS	AXI4-Stream Master	ATS RQ Interface
Interrupt	Output	Control interrupt output

Notes:

1. x = 0 to 3

AXI4/ACE Port Descriptions

The block diagram for the System Cache core is shown in the following figure. All System Cache core interfaces are compliant with AXI4. The input signals `ACLK` and `ARESETN` implement clock and reset for the entire System Cache core.

Figure 20: **AXI/ACE Block Diagram**

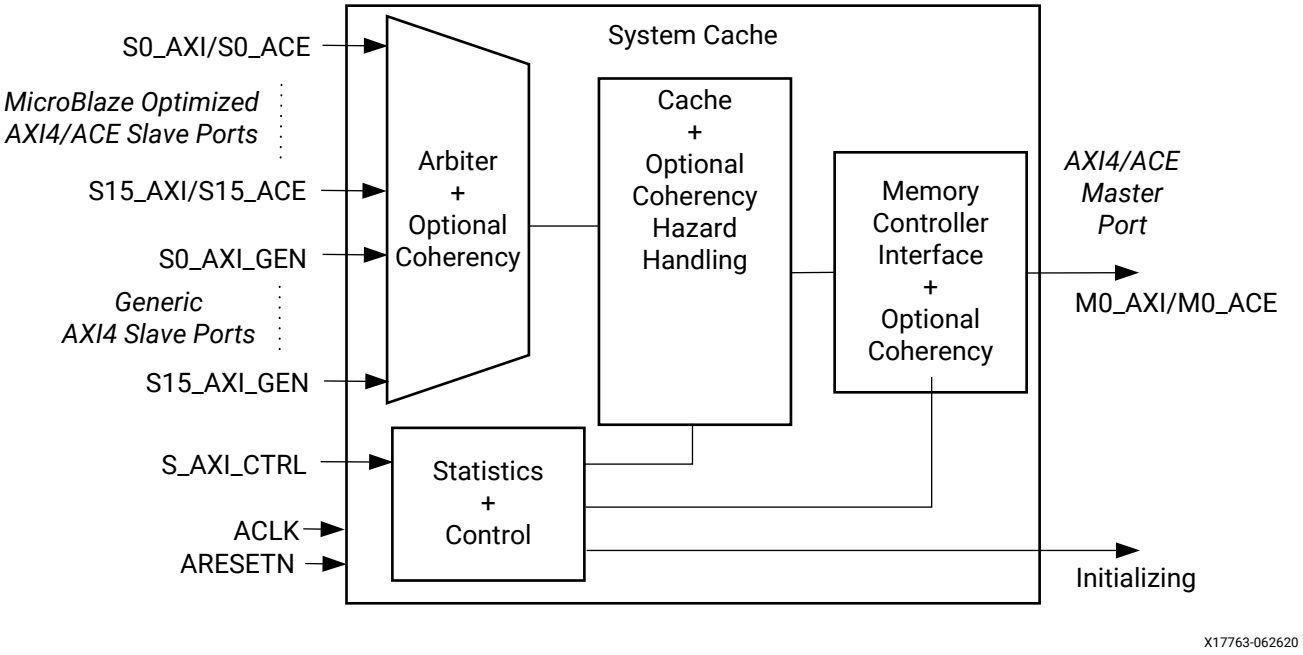


Table 16: **I/O Interfaces**

Name	Type	Description
ACLK	Input	Core clock
ARESETN	Input	Synchronous reset of core
Initializing	Output	Core is initializing after reset
Sx_AXI[1], [2]	AXI4 Slave	MicroBlaze optimized cache port
Sx_ACE[1], [2]	ACE Slave	MicroBlaze optimized cache coherent port
Sx_AXI_GEN[1]	AXI4 Slave	Generic cache port
M0_AXI[3]	AXI4 Master	Memory controller master port
M0_ACE[3]	ACE Master	Master interface to ACE port on PS
S_AXI_CTRL	AXI4-Lite Slave	Control port

Notes:

1. x = 0–15
2. Sx_AXI and Sx_ACE are mutually exclusive.
3. M0_AXI and M0_ACE are mutually exclusive.

Register Space

In the following tables Access is indicated by R for read-only, W for write-only and R/W for Read/Write.

All registers in the optional Statistics and Control module are 64 bits wide. The address map structure is shown in the following table. The data width of the control interface is configurable to be 32 (default) or 64 bits. For the 32-bit interface a write to a control-type register takes effect on the write to the lower 32-bits. For a 64-bit interface the control register write always takes effect immediately.

Category

Table 17: **Category Field**

Address (binary)	Description
0_00yy_yyxx_xxxx_xx00	Statistics Functionality for Ports (Optimized ports)
0_01yy_yyxx_xxxx_xx00	Statistics Functionality for Ports (Generic ports)
0_1000_00xx_xxxx_xx00	Statistics Field for Arbiter
0_1100_00xx_xxxx_xx00	Statistics Field for Access
0_1100_01xx_xxxx_xx00	Statistics Field for ATS
1_0000_00xx_xxxx_xx00	Statistics Field for Lookup
1_0100_00xx_xxxx_xx00	Statistics Field for Update
1_1000_00xx_xxxx_xx00	Statistics Field for Backend AXI/ACE
1_1001_00xx_xxxx_xx00	Reserved
1_1001_10xx_xxxx_xx00	Reserved
1_1010_00xx_xxxx_xx00	Statistics Field for Backend CCIX/CHI
1_1100_00xx_xxxx_xx00	Field for Control

Functionality

Statistics Registers

Statistics Field for Ports

Table 18: **Statistics Fields for Optimized Port 0**

Offset[1]	Register Name	Access	Format[2]	Description
0x0_0000	RdSegO0	R	COUNT	Segments Per Read Transaction
0x0_0020	WrSegO0	R	COUNT	Segments Per Write Transaction
0x0_0040	RdInfoQueueO0	R	QUEUE	Read Information Port Queue Statistics

Table 18: **Statistics Fields for Optimized Port 0** *(cont'd)*

Offset[1]	Register Name	Access	Format[2]	Description
0x0_0060	RdDataQueueO0	R	QUEUE	Read Data Port Queue Statistics
0x0_0080	BIPInfoQueuO0	R	QUEUE	BRESP Information Port Queue Statistics
0x0_00A0	BPQueueO0	R	QUEUE	BRESP Port Queue Statistics
0x0_00C0	WrInfoQueueO0	R	QUEUE	Write Information Port Queue Statistics
0x0_00E0	WrDataQueueO0	R	QUEUE	Write Data Port Queue Statistics
0x0_0100	RdBlockO0	R	COUNT	Read Blocked from Arbitration
0x0_0120	WrHitO0	R	SCOUNT	Write Hits
0x0_0140	WrMissO0	R	SCOUNT	Write Miss
0x0_0160	WrMissDirtyO0	R	SCOUNT	Write Miss Dirty
0x0_0180	RdHitO0	R	SCOUNT	Read Hit
0x0_01A0	RdMissO0	R	SCOUNT	Read Miss
0x0_01C0	RdMissDirtyO0	R	SCOUNT	Read Miss Dirty
0x0_01E0	LockWrHitO0	R	SCOUNT	Locked Write Hit
0x0_0200	LockRdHitO0	R	SCOUNT	Locked Read Hit
0x0_0220	FirstWrHitO0	R	SCOUNT	First Write Hit
0x0_0240	RdLatencyO0	R	COUNT	Read Latency
0x0_0260	WrLatencyO0	R	COUNT	Write Latency
0x0_0280	CRLSO0	R/W	LONGINT	Configuration for read latency statistics
0x0_02A0	CWLSO0	R/W	LONGINT	Configuration for write latency statistics

Notes:

1. Address offset is given for Optimized Port 0. For offset to other per port registers use the formulas:
   - Optimized Port x Offset = IP Offset + x * 0x400, x = 1 to 15
   - Generic Port x Offset = IP Offset + x * 0x400 + 0x4000, x = 1 to 15
2. See section Register Records and Formats for details.

Statistics Field for Arbiter

Table 19: **Statistics Field for Arbiter**

Offset	Register Name	Access	Format[1]	Description
0x0_8000	ValidArbit	R	COUNT	Arbiter Transaction Clock Cycles after arbitration
0x0_8020	ConcurrentAccess	R	COUNT	Arbiter Transactions available to select when arbitrating

Notes:

1. See section Register Records and Formats for details.

Statistics Field for Access

Table 20: **Statistics Field for Access**

Offset	Register Name	Access	Format[1]	Description
0x0_C000	AccessValid	R	COUNT	Access Transaction Clock Cycles after access stage
0x0_C020	AccessFetchStall	R	COUNT	Access Time snoop fetch stall because of conflicts
0x0_C040	AccessReqStall	R	COUNT	Access Time snoop request stalls because of conflicts
0x0_C060	AccessActionStall	R	COUNT	Access Time snoop action stalls because of conflicts

Notes:

1. See section Register Records and Formats for details.

Statistics Field for Lookup

Table 21: **Statistics Field for Lookup**

Offset	Register Name	Access	Format[1]	Description
0x1_0020	LUFetchStall	R	COUNT	Lookup Time fetch stall because of conflict
0x1_0040	LUMemStall	R	COUNT	Lookup Time memory stalls because of conflict
0x1_0060	LUDataStall	R	COUNT	Lookup Time stalled due to memory access
0x1_0080	LUDataHitStall	R	COUNT	Lookup Time stalled due to conflict
0x1_00A0	LUDataMissStall	R	COUNT	Lookup Time stalled due to full buffers

Notes:

1. See section Register Records and Formats for details.

Statistics Field for Update

Table 22: **Statistics Field for Update**

Offset	Register Name	Access	Format[1]	Description
0x1_4000	UDStall	R	COUNT	Update Cycles transactions are stalled
0x1_4020	UDTagFree	R	COUNT	Update Cycles tag interface is free
0x1_4040	UDDataFree	R	COUNT	Update Cycles data interface is free
0x1_4060	UDRdInfo	R	QUEUE	Update Queue statistics for read transactions
0x1_4080	UDRdData	R	QUEUE	Update Queue statistics for read data
0x1_40A0	UDEvict	R	QUEUE	Update Queue statistics for evict information
	UDBRespSrc	R	QUEUE	Update Queue statistics for BRESP source information
0x1_40E0	UDWrMiss	R	QUEUE	Update Queue statistics for write miss information

Table 22: **Statistics Field for Update** *(cont'd)*

Offset	Register Name	Access	Format[1]	Description
0x1_4100	UDWrMissAlloc	R	QUEUE	Update Queue statistics for allocated write miss information

Notes:
1. See section Register Records and Formats for details.

Statistics Field for Backend AXI/ACE

Table 23: **Statistics Field for Backend AXI/ACE**

Offset	Register Name	Access	Format[1]	Description
0x1_8000	BEWrAddr	R	QUEUE	Backend Queue statistics for write address channel information
0x1_8020	BEWrData	R	QUEUE	Backend Queue statistics for write channel data
0x1_8040	BERdAddr	R	QUEUE	Backend Queue statistics for read address channel information
0x1_8060	BESearchDep	R	COUNT	Backend Transaction search depth for read access before released
0x1_8080	BERdStall	R	COUNT	Backend Cycles stall due to search
0x1_80A0	BERdProtStall	R	COUNT	Backend Cycles stall due to conflicts
0x1_80C0	BERdLatency	R	COUNT	Backend Read latency for external transaction to memory
0x1_80C0	BEWrLatency	R	COUNT	Backend Write latency for external trasaction to memory

Notes:
1. See section Register Records and Formats for details.

Statistics Field for ATS/ATC

Table 24: **Statistics Fields for ATS/ATC**

Offset[1]	Register Name	Access	Format[2]	Description
0x0_42C0	ATCWrHitO0	R	SCOUNT	ATC Generic Port 0 Write Hit
0x0_42E0	ATCWrMissO0	R	SCOUNT	ATC Generic Port 0 Write Miss
0x0_4300	ATCLRUWrMissO0	R	SCOUNT	ATC Generic Port 0 LRU Write Miss
0x0_4320	ATCRdHitO0	R	SCOUNT	ATC Generic Port 0 Read Hit
0x0_4340	ATCRdMissO0	R	SCOUNT	ATC Generic Port 0 Read Miss
0x0_4360	ATCLRURdMissO0	R	SCOUNT	ATC Generic Port 0 LRU Read Miss
0x0_4380	ATCRdLatencyO0	R	COUNT	ATC Generic Port 0 Read Latency
0x0_43A0	ATCWrLatencyO0	R	COUNT	ATC Generic Port 0 Write Latency
0x0_43E0	ATCCRLSO0	R/W	LONGINT	Configuration for ATC Generic Port 0 Read latency statistics.
0x0_43E8	ATCCWLSO0	R/W	LONGINT	Configuration for ATC Generic Port 0 Write latency statistics

Table 24: **Statistics Fields for ATS/ATC** *(cont'd)*

Offset[1]	Register Name	Access	Format[2]	Description
0x0_C200	ATSRdSegments	R	COUNT	ATS Segments per Read Transaction
0x0_C220	ATSWrSegments	R	COUNT	ATS Segments per Write Transaction
0x0_C240	ATSPRIRetry	R	ECOUNT	ATS Transaction PRI Retry
0x0_C260	ATSPRITimeout	R	ECOUNT	ATS Transaction PRI Timeout
0x0_C280	ATSTAFail	R	ECOUNT	ATS Transaction TA Fail
0x0_C2A0	ATSInvalTimeout	R	ECOUNT	ATS Transaction Invalidation Timeout
0x0_C2C0	ATSInval	R	SCOUNT	ATS Transaction Invalidation
0x0_C2E0	ATSWrHit	R	SCOUNT	ATS Write Hit
0x0_C300	ATSWrMiss	R	SCOUNT	ATS Write Miss
0x0_C320	ATSLRWWrMiss	R	SCOUNT	ATS LRW Write Miss
0x0_C340	ATSRdHit	R	SCOUNT	ATS Read Hit
0x0_C360	ATSRdMiss	R	SCOUNT	ATS Read Miss
0x0_C380	ATSLRWRdMiss	R	SCOUNT	ATS LRW Read Miss
0x0_C3A0	ATSRdLatency	R	COUNT	ATS Read Latency
0x0_C3C0	ATSWrLatency	R	COUNT	ATS Write Latency
0x0_C3E0	ATSCRLS	R/W	LONGINT	Configuration for ATS Read latency statistics
0x0_C3E8	ATSCWLS	R/W	LONGINT	Configuration for ATS Write latency statistics

Notes:

1. Address offset is given for Generic Port 0. For offset to other per port registers use the formulas:
   - Generic Port x Offset + x * 0x400, x = 1 to 15
2. See section Register Records and Formats for details.

Statistics Field for Backend CCIX

Table 25: **Statistics Field for Backend CCIX**

Offset	Register Name	Access	Format[1]	Description
0x1_9000	TxReqRd	R	ECOUNT	TX Request types ReadUnique, ReadShared,ReadClean, ReadNotSharedDirty
0x1_9020	TxReqRdNoSnp	R	ECOUNT	TX Request type ReadNoSnp
0x1_9040	TxReqRdOnce	R	ECOUNT	TX Request types ReadOnce, ReadOnceCleanInvalid, ReadOnceMakeInvalid
0x1_9060	TxReqWrNoSnp	R	ECOUNT	TX Request types WriteNoSnpPtl, WriteNoSnpFull
0x1_9080	TxReqWrUniq	R	ECOUNT	TX Request types WriteUniquePtl, WriteUniqueFull
0x1_90A0	TxReqWrCB	R	ECOUNT	TX Request types WriteBackPtl, WriteBackFullUD, WriteBackFullSD, WriteCleanFullSD, WriteEvictFull

Table 25: **Statistics Field for Backend CCIX** *(cont'd)*

Offset	Register Name	Access	Format[1]	Description
0x1_90E0	TxReqAtom	R	ECOUNT	TX Request types AtomicStore, AtomicLoad, AtomicSwap, AtomicCompare
0x1_9100	TxReqPerm	R	ECOUNT	TX Request types CleanUnique, MakeUnique
0x1_9120	TxReqDataless	R	ECOUNT	TX Request types CleanShared, CleanSharedPersist, CleanInvalid,MakeInvalid
0x1_9140	TxEvict	R	ECOUNT	TX Request type Evict
0x1_9160	TxRspCrdExch	R	ECOUNT	TX Request CreditGrant
0x1_9180	TxReqCrdRet	R	ECOUNT	TX Request CreditReturn
0x1_91A0	TxRspCompAck	R	ECOUNT	TX Snoop Response CompAck
0x1_91C0	TxRspSnp	R	ECOUNT	TX Snoop Responses SnpResp_I, SnpResp_UC, SnpResp_SC
0x1_9200	TxRspSnpMiss	R	ECOUNT	TX Snoop Response SnpRespMiss
0x1_9220	TxMscCrdRet	R	ECOUNT	TX Misc CreditReturn
0x1_9240	TxDatCrdExch	R	ECOUNT	TX Data CreditGrant
0x1_9260	TxDatCrdRet	R	ECOUNT	TX Data CreditReturn
0x1_9280	TxDatCompDat	R	ECOUNT	TX Request Response CompData_UC, CompData_SC, CompData_UD_PD, CompData_SD_PD
0x1_92A0	TxDatSnp	R	ECOUNT	TX Snoop Response all SnpRespData and SnpRespDataPtl
0x1_9320	TxSnpCrdExch	R	ECOUNT	TX Snoop CreditGrant
0x1_9340	TxSnpCrdRet	R	ECOUNT	TX Snoop CreditReturn
0x1_9360	TxReqSnp	R	ECOUNT	TX Snoop SnpToAny, SnpToC, SnpToS, SnpToSC, SnpToI,SnpMakeI
0x1_9380	TxRspComp	R	ECOUNT	TX Request Response Comp
0x1_93A0	TxNumTLP	R	ECOUNT	TX total number of TLPs
0x1_93C0	TxNumSrc	R	ECOUNT	TX number of message types per TLP
0x1_9400	RxReqRd	R	ECOUNT	RX Request types ReadUnique, ReadShared, ReadClean, ReadNotSharedDirty
0x1_9420	RxReqRdNoSnp	R	ECOUNT	RX Request type ReadNoSnp
0x1_9440	RxReqRdOnce	R	ECOUNT	RX Request types ReadOnce, ReadOnceCleanInvalid, ReadOnceMakeInvalid
0x1_9460	RxReqWrNoSnp	R	ECOUNT	RX Request types WriteNoSnpPtl, WriteNoSnpFull
0x1_9480	RxReqWrUniq	R	ECOUNT	RX Request types WriteUniquePtl, WriteUniqueFull
0x1_94A0	RxReqWrCB	R	ECOUNT	RX Request types WriteBackPtl, WriteBackFullUD, WriteBackFullSD, WriteCleanFullSD, WriteEvictFull
0x1_94E0	RxReqAtom	R	ECOUNT	RX Request types AtomicStore, AtomicLoad, AtomicSwap, AtomicCompare

Table 25: **Statistics Field for Backend CCIX** *(cont'd)*

Offset	Register Name	Access	Format[1]	Description
0x1_9500	RxReqPerm	R	ECOUNT	RX Request types CleanUnique, MakeUnique
0x1_9520	RxReqDataless	R	ECOUNT	RX Request types CleanShared, CleanSharedPersist, CleanInvalid, MakeInvalid
0x1_9540	RxReqEvict	R	ECOUNT	RX Request type Evict
0x1_9560	RxReqCrdExch	R	ECOUNT	RX Request CreditGrant
0x1_9580	RxReqCrdRet	R	ECOUNT	RX Request CreditReturn
0x1_95A0	RxRspCmpAck	R	ECOUNT	RX Response type CompAck
0x1_95C0	RxRspSnp	R	ECOUNT	RX Snoop Responses SnpResp_I, SnpResp_UC, SnpResp_SC
0x1_9600	RxRspSnpMiss	R	ECOUNT	RX Snoop Response SnpRespMiss
0x1_9620	RxMiscCrdExch	R	ECOUNT	RX Misc CreditGrant
0x1_9640	RxDatCrdExch	R	ECOUNT	RX Data CreditGrant
0x1_9660	RxDatCrdRet	R	ECOUNT	RX Data CreditReturn
0x1_9680	RxDatCompDat	R	ECOUNT	RX Request Response CompData_UC, CompData_SC, CompData_UD_PD, CompData_SD_PD
0x1_96A0	RxDatSnp	R	ECOUNT	RX Snoop Response all SnpRespData and SnpRespDataPtl
0x1_9720	RxSnpCrdExch	R	ECOUNT	RX Snoop CreditGrant
0x1_9740	RxSnpCrdRet	R	ECOUNT	RX Snoop CreditReturn
0x1_9760	RxReqSnp	R	ECOUNT	RX Snoop SnpToAny, SnpToC, SnpToS, SnpToSC, SnpToI, SnpMakeI
0x1_9780	RxRspComp	R	ECOUNT	RX Response Comp Event
0x1_97A0	RxNumTlp	R	ECOUNT	RX total number of TLPs

Notes:

1. See section Register Records and Formats for details.

Statistics Field for Backend CHI

Table 26: **Statistics Field for Backend CHI**

Offset	Register Name	Access	Format[1]	Description
0x1_9000	TxReqRd	R	RCOUNT	TX Request types ReadUnique, ReadShared,ReadClean, ReadNotSharedDirty
0x1_9020	TxReqRdNoSnp	R	RCOUNT	TX Request type ReadNoSnp
0x1_9040	TxReqRdOnce	R	RCOUNT	TX Request types ReadOnce, ReadOnceCleanInvalid, ReadOnceMakeInvalid
0x1_9060	TxReqWrNoSnp	R	RCOUNT	TX Request types WriteNoSnpPtl, WriteNoSnpFull

Table 26: **Statistics Field for Backend CHI** *(cont'd)*

Offset	Register Name	Access	Format[1]	Description
0x1_9080	TxReqWrUniq	R	RCOUNT	TX Request types WriteUniquePtl, WriteUniqueFull
0x1_90A0	TxReqWrCB	R	RCOUNT	TX Request types WriteBackPtl, WriteBackFullUD, WriteBackFullSD, WriteCleanFullSD, WriteEvictFull
0x1_90C0	Reserved	R	RCOUNT	Reserved
0x1_90E0	TxReqAtom	R	RCOUNT	TX Request types AtomicStore, AtomicLoad, AtomicSwap, AtomicCompare
0x1_9100	TxReqPerm	R	RCOUNT	TX Request types CleanUnique, MakeUnique
0x1_9120	TxReqDataless	R	RCOUNT	TX Request types CleanShared, CleanSharedPersist, CleanInvalid,MakeInvalid
0x1_9140	TxReqPCredRet	R	RCOUNT	TX Request types PCreditReturn
0x1_9160	TxReqAll	R	RCOUNT	All TX Request types
0x1_9180	TxReqCrdRet	R	ECOUNT	TX Request PCreditReturn
0x1_91A0	TxRspCompAck	R	ECOUNT	TX Snoop Response CompAck
0x1_91C0	TxRspSnp	R	ECOUNT	TX Snoop Responses SnpResp_I, SnpResp_UC, SnpResp_SC
0x1_91E0	TxRspSnpFwd	R	ECOUNT	TX Snoop Forward Responses types SnpResp_I_Fwded_I, SnpResp_I_Fwded_SC, SnpResp_I_Fwded_UC, SnpResp_I_Fwded_UD_PD, SnpResp_I_Fwded_SD_PD, SnpResp_SC_Fwded_I, SnpResp_SC_Fwded_SC, SnpResp_SC_Fwded_SD_PD, SnpResp_UC_Fwded_I, SnpResp_UD_Fwded_I, SnpResp_SD_Fwded_I, SnpResp_SD_Fwded_SC
0x1_9200	TxRspSnpMiss	R	ECOUNT	TX Snoop Response SnpRespMiss
0x1_9220	TxDatAll	R	ECOUNT	TX All Data type
0x1_9240	TxDatCB	R	ECOUNT	TX Data type CopyBackWrData_I, CopyBackWrData_UC, CopyBackWrData_SC,CopyBackWrData_UD_PD, CopyBackWrData_SD_PD
0x1_9260	TxDatNonCB	R	ECOUNT	TX Data type NonCopyBackWrData
0x1_9280	TxDatCompDat	R	ECOUNT	TX Request Response CompData_UC, CompData_SC, CompData_UD_PD, CompData_SD_PD
0x1_92A0	TxDatSnp	R	ECOUNT	TX Snoop Non-Forward Response types SnpRespData_I, SnpRespData_UC, SnpRespData_UD, SnpRespData_SC, SnpRespData_SD, SnpRespData_I_PD, SnpRespData_UC_PD, SnpRespData_SC_PD

Table 26: **Statistics Field for Backend CHI** *(cont'd)*

Offset	Register Name	Access	Format[1]	Description
0x1_92C0	TxDatSnpFwd	R	SCOUNT	TX Snoop Data Forward Response types SnpRespData_I_Fwded_SC, SnpRespData_I_Fwded_SD_PD, SnpRespData_SC_Fwded_SC, SnpRespData_SC_Fwded_SD_PD, SnpRespData_SD_Fwded_SC, SnpRespData_I_PD_Fwded_I, SnpRespData_I_PD_Fwded_SC, SnpRespData_SC_PD_Fwded_I, SnpRespData_SC_PD_Fwded_SC
0x1_92E0	TxDatSnpPtl	R	SCOUNT	TX Snoop Response SnpRespDataPtl_I_PD and SnpRespDataPtl_UD
0x1_9300	TxDatAll	R	SCOUNT	TX all Data responses
0x1_9400	RxCore	R	ECOUNT	Number of times a snoop data has been extracted from core.
0x1_9420	RxCBRefresh	R	ECOUNT	Number of time a CopyBack has ts line state altered by a snoop
0x1_9560	RxRspRetryAck	R	ECOUNT	RX Response RetryAck
0x1_9580	RxRspPGrnt	R	ECOUNT	RX Response PCreditGrant
0x1_95A0	RxRspCmpAck	R	ECOUNT	RX Response type CompAck
0x1_95C0	RxRspSnp	R	ECOUNT	RX Snoop Responses SnpResp_I, SnpResp_UC, SnpResp_SC
0x1_9620	RxRspAll	R	ECOUNT	RX All responses
0x1_9640	RxRspCompDBID	R	ECOUNT	RX resposne RespCompDBID
0x1_9660	RxRspDBID	R	ECOUNT	RX response RespDBID
0x1_9680	RxDatCompDat	R	ECOUNT	RX Request Response CompData_UC, CompData_SC, CompData_UD_PD, CompData_SD_PD
0x1_96A0	RxDatSnp	R	ECOUNT	RX Snoop Response all SnpRespData and SnpRespDataPtl
0x1_9700	RxDatAll	R	ECOUNT	RX all Data response types
0x1_9720	RxSnpFwd	R	ECOUNT	RX all snoop forward types
0x1_9740	RxSnpStash	R	ECOUNT	RX all snoop stash types
0x1_9760	RxReqSnp	R	ECOUNT	RX Snoop SnpToAny, SnpToC, SnpToS, SnpToSC, SnpToI, SnpMakeI
0x1_9780	RxRspComp	R	ECOUNT	RX Response Comp Event
0x1_97A0	RxRspRec	R	ECOUNT	RX response RespReadReceipt type
0x1_97C0	RxSnpAll	R	ECOUNT	RX all snoop request types

Notes:

1. See section Register Records and Formats for details.

Register Records and Formats

COUNT Format Register Record

Table 27: **COUNT Format Register Record**

Address Offset	Name	Reset Value	Access	Description
0x00	COUNTEVENT	0	R	Number of times the event has been triggered
0x08	COUNTMINMAX	0	R	Min, max and status information
0x10	COUNTSUM	0	R	Sum of measured data
0x18	COUNTSUM2	0	R	Sum of measured data squared

SCOUNT Format Register Record

Table 28: **SCOUNT Format Register Record**

Address Offset	Name	Reset Value	Access	Description
0x00	COUNTEVENT	0	R	Number of times the event has been triggered
0x08				Reserved
0x10	COUNTSUM	0	R	Sum of measured data
0x18	COUNTSUM2	0	R	Sum of measured data squared

ECOUNT Format Register Record

Table 29: **ECOUNT Format Register Record**

Address Offset	Name	Reset Value	Access	Description
0x00	COUNTEVENT	0	R	Number of times the event has been triggered
0x08				Reserved
0x10				Reserved
0x18				Reserved

RCOUNT Format Register Record

Table 30: **RCOUNT Format Register Record**

Address Offset	Name	Reset Value	Access	Description
0x00	COUNTEVENT	0	R	Number of request
0x08				Reserved
0x10	COUNTSUM	0	R	Number of unique requests, i.e., COUNTSUM is less than or equal to COUNTEVENT

Table 30: **RCOUNT Format Register Record** *(cont'd)*

Address Offset	Name	Reset Value	Access	Description
0x18		0	R	Reserved

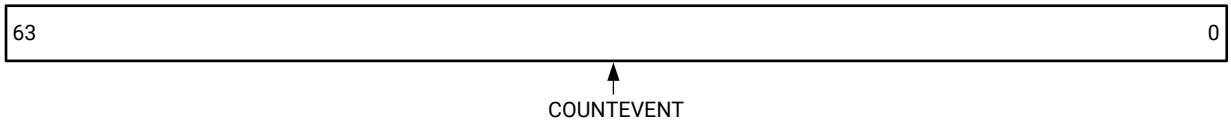
QUEUE Format Register Record

Table 31: **QUEUE Format Register Record**

Address Offset	Name	Reset Value	Access	Description
0x00	QUEUEEMPTY	0	R	Clock cycles queue has been empty
0x08	QUEUEUPDATE	0	R	Number of times updated with push or pop
0x10	QUEUEMAX	0	R	Max depth of queue
0x18	QUEUESUM	0	R	Sum of queue depth when updated

COUNTEVENT Format Register

Figure 21: **COUNTEVENT Format Register**



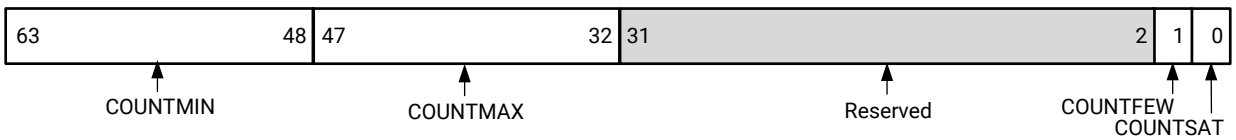
X20803-073119

Table 32: **COUNTEVENT Format Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:0	COUNTEVENT	0	R	Number of times the event has been triggered

COUNTMINMAX Format Register

Figure 22: **COUNTMINMAX Format Register**



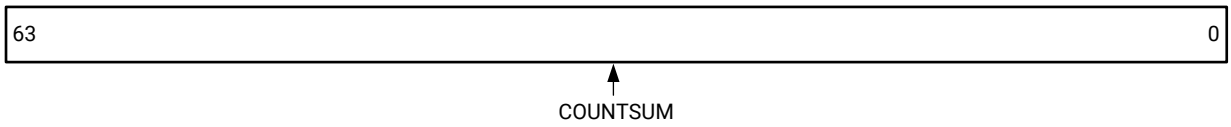
X20804-073119

Table 33: **COUNTMINMAX Format Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:48	COUNTMIN	0xFFFF	R	Minimum observed counted value as unsigned
47:32	COUNTMAX	0x0000	R	Maximum observed counted value as unsigned, saturates when 0xFFFF is reached
31:2				Reserved
1	COUNTFEW	0		Flag if number of concurrent events of the measured type has been reached, indicating that the resulting statistics are inaccurate
0	COUNTSAT	0		Flag if measurement have been saturated; this means that the statistics results are less accurate. Both average and standard deviation measurements will be lower than actual value.

COUNTSUM Format Register

Figure 23: **COUNTSUM Format Register**



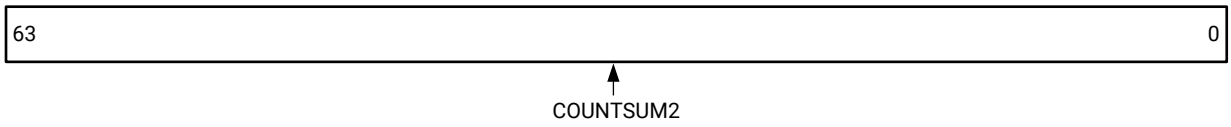
X20805-073119

Table 34: **COUNTSUM Format Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:0	COUNTSUM	0	R	Sum of measured data

COUNTSUM2 Format Register

Figure 24: **COUNTSUM2 Format Register**



X20806-073119

Table 35: **COUNTSUM2 Format Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:0	COUNTSUM2	0	R	Sum of measured data squared

QUEUEEMPTY Format Register

Figure 25: **QUEUEEMPTY Format Register**

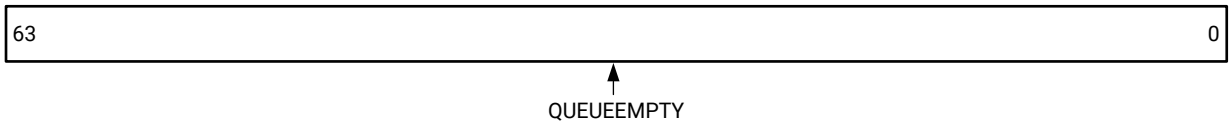


Table 36: **QUEUEEMPTY Format Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:0	QUEUEEMPTY	0	R	Clock cycles the queue has been empty

QUEUEUPDATE Format Register

Figure 26: **QUEUEUPDATE Format Register**

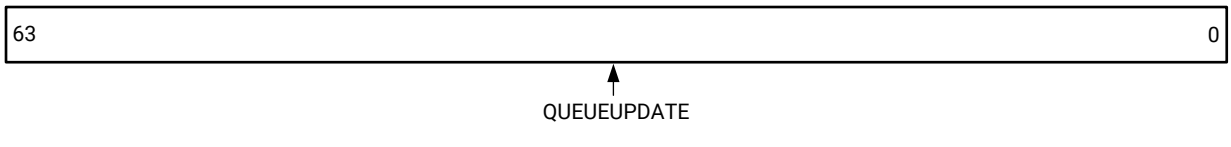


Table 37: **QUEUEUPDATE Format Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:0	QUEUEUPDATE	0	R	Number of times queue updated with push or pop

QUEUEMAX Format Register

Figure 27: **QUEUEMAX Format Register**

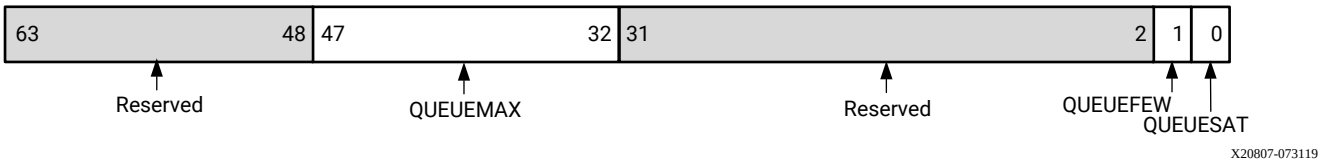


Table 38: **QUEUEMAX Format Bit Definitions**

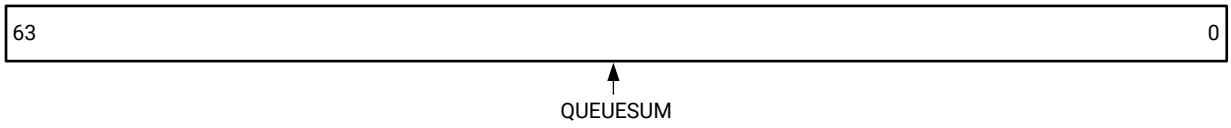
Bits	Name	Reset Value	Access	Description
63:48				Reserved

Table 38: **QUEUEMAX Format Bit Definitions** *(cont'd)*

Bits	Name	Reset Value	Access	Description
47:32	QUEUEMAX	0	R	Maximum depth for queue as unsigned measurement encountered, saturates when 0xFFFF reached
31:2				Reserved
1	QUEUEFEW	0	R	Flag if number of concurrent events of the measured type has been reached, indicating that the resulting statistics are inaccurate
0	QUEUESAT	0	R	Flag if measurement has been saturated; this means that the statistics results are less accurate. Both average and standard deviation measurements will be lower than the actual value.

QUEUESUM Format Register

Figure 28: **QUEUESUM Format Register**



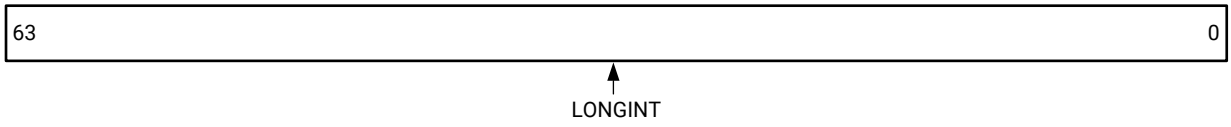
X20802-073119

Table 39: **QUEUESUM Format Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:0	QUEUESUM	0	R	Sum of queue depth when updated

LONGINT Format Register

Figure 29: **LONGINT Format Register**



X24116-061120

Table 40: **LONGINT Format Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:0	LONGINT	0	R/W	64 bit value

Backend CCIX and CHI Registers

Backend CCIX and CHI Registers

In the nominal use case the registers below are handled by firmware executing in the MicroBlaze processor sub-system, and there is nothing the user needs to handle.

Fields denoted with FW: in the descriptions below are updated by the firmware.

Unused CCIX registers are reserved in CHI context, and will return Reserved default value if read.

Table 41: **Backend CCIX Address Map**

Offset	Register Name	Access	Format	Description
0x1_A200	CCIXCCSI0	R	32	Primary Port 0 Common Capability & Status I
0x1_A204	CCIXCCSII0	R	32	Primary Port 0 Common Capability & Status II
0x1_A208	CCIXCCSIII0	R	32	Primary Port 0 Common Capability & Status III
0x1_A210	CCIXCCI0	R/W	32	Primary Port 0 Common Control I
0x1_A214	CCIXCCII0	R/W	32	Primary Port 0 Common Control II
0x1_A218	CCIXDEVECS0	R/W	32	CCIX Device Error Control & Status
0x1_A21C	CCIXSCPCIE0	R/W	32	Primary Port 0 SC PCIe & CCIX Common
0x1_A280	CCIXCSI0	R	32	Port 0 Capability & Status I
0x1_A284	CCIXCSII0	R	32	Port 0 Capability & Status II
0x1_A288	CCIXCSIII0	R	32	Port 0 Capability & Status III
0x1_A290	CCIXCS00	R	32	Port 0 Error Control & Status 0
0x1_A294	CCIXCS10	R	32	Port 0 Error Control & Status 1
0x1_A298	CCIXCI0	R/W	32	Port 0 Control I
0x1_A29C	CCIXTPIDM0	R/W	32	Port 0 Source TransportID Map
0x1_A300	CCIXL0TXS0	R	64	Port 0 Link 0 System Cache Tx Status
0x1_A380	CCIXL0RXS0	R	64	Port 0 Link 0 System Cache Rx Status
0x1_A680	CCIXL0BFCV0	R/W	64	Port 0 Link 0 BFCV0 + BFCV1
0x1_B000	CCIXL0CI0	R/W	32	Port 0 Link 0 Control I
0x1_B004	CCIXL0CII0	R/W	32	Port 0 Link 0 Control II
0x1_B008	CCIXL0CIII0	R/W	32	Port 0 Link 0 Control III
0x1_B00C	CCIXL0CIV0	R/W	32	Port 0 Link 0 Control IV
0x1_B010	CCIXL0ECS00	R/W	32	Port 0 Link 0 Error Control & Status 0
0x1_B014	CCIXL0ECS01	R/W	32	Port 0 Link 0 Error Control & Status 1
0x1_B018	CCIXL0DTPIDM0	R/W	32	Port 0 Link 0 Destination TransportID Map
0x1_B01C	CCIXL0PLSPEC0	R/W	32	Port 0 Link 0 SC Port/Link Specific
0x1_B200	CCIXLCSI0	R	32	Port 0 Link Capability and Status I
0x1_B204	CCIXLCSII0	R	32	Port 0 Link Capability and Status II
0x1_B208	CCIXLCSIII0	R	32	Port 0 Link Capability and Status III
0x1_B20C	CCIXLCSIV	R	32	Port 0 Link Capability and Status IV

Table 41: **Backend CCIX Address Map** *(cont'd)*

Offset	Register Name	Access	Format	Description
0x1_B300	CCIXTRA0CS1	R	32	RA0 Capability and Status 1
0x1_B304	CCIXRA0CI	R/W	32	RA0 Control I
0x1_B310	CCIXRA0ECS0	R/W	32	RA0 Error Control & Status 0
0x1_B314	CCIXRA0ECS1	R/W	32	RA0 Error Control & Status 1
0x1_B318	CCIXRA0SCSPE	R/W	64	SC RA0 Specific

Table 42: **Backend CHI Address Map**

Offset	Register Name	Access	Format	Description
0x1_A300	CCIXL0TXS0	R	64	Port 0 Link 0 System Cache Tx Status
0x1_A380	CCIXL0RXS0	R	64	Port 0 Link 0 System Cache Rx Status
0x1_B01C	CCIXL0PLSPEC0	R/W	32	Port 0 Link 0 SC Port/Link Specific
0x1_B300	CCIXTRA0CS1	R	32	RA0 Capability and Status 1
0x1_B304	CCIXRA0CI	R/W	32	RA0 Control I
0x1_B310	CCIXRA0ECS0	R/W	32	RA0 Error Control & Status 0
0x1_B314	CCIXRA0ECS1	R/W	32	RA0 Error Control & Status 1
0x1_B318	CCIXRA0SCSPE	R/W	64	SC RA0 Specific

Primary Common Capabilities & Status I (DVSEC) Register

Figure 30: **Primary Common Capabilities & Status I (DVSEC) Register**

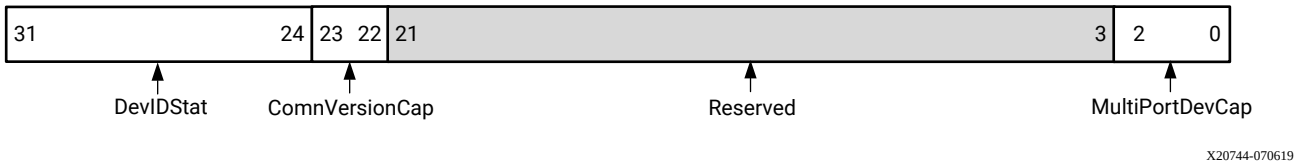
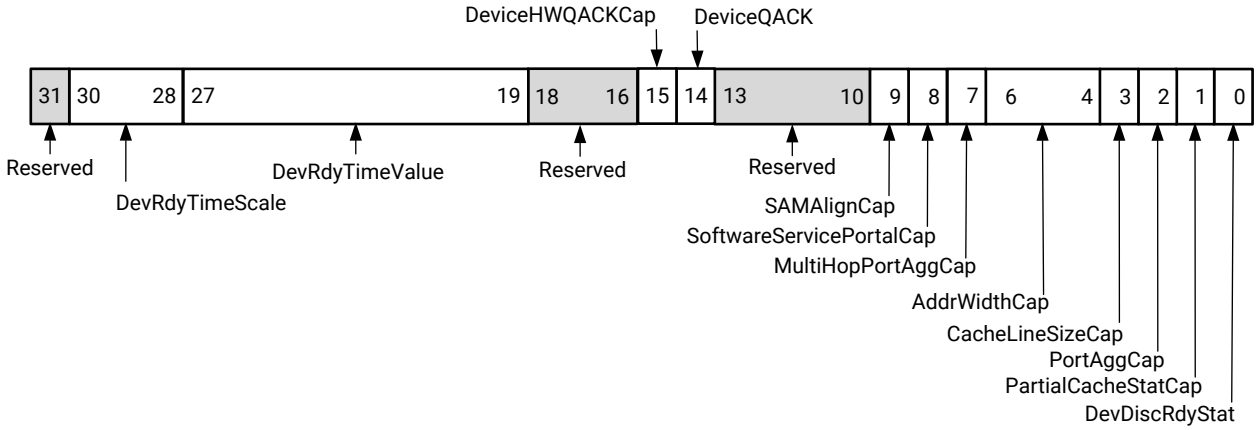


Table 43: **Primary Common Capabilities & Status I (DVSEC) Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:24	DevIDStat	0	R	FW: Device ID Status
23:22	ComnVersionCap	0	R	FW: Common Version Capability
21:3				Reserved
2:0	MultiportDevCap	0	R	MultiPort Device Capability

Primary Common Capabilities & Status II (DVSEC) Register

Figure 31: **Primary Common Capabilities & Status II (DVSEC) Register**



X20745-070619

Table 44: **Primary Common Capabilities & Status II (DVSEC) Bit Definitions**

Bits	Name	Reset Value	Access	Description
31				Reserved
30:28	DevRdyTimeScale	0	R	FW: Readiness Time Scale
27:19	DevRdyTimeValue	0	R	FW: Readiness Time Value
18:16				Reserved
15	DeviceHWQACKCap	0	R	CCIX Device HW QACK Capability
14	DeviceQACK	0	R	CCIX Device HW QACK
13:10				Reserved
9	SAMAlignCap	0	R	SAM Alignment Capability
8	SoftwareServicePortalCap	0	R	FW: CCIX Software Service Portal Capability
7	MultiHopPortAggrCap	0	R	Multi-Hop Port Aggregation Capability
6:4	AddrWidthCap	0	R	Address Width Capability
3	CacheLineSizeCap	0	R	Cache Line Size Capability
2	PortAggCap	0	R	Port Aggregation Capability
1	PartialCacheStatesCap	0	R	Partial Cache State Capability
0	DevDiscRdyStat	1	R	Device Discovery Register Status

Primary Common Capabilities & Status III (DVSEC) Register

Figure 32: **Primary Common Capabilities & Status III (DVSEC) Register**

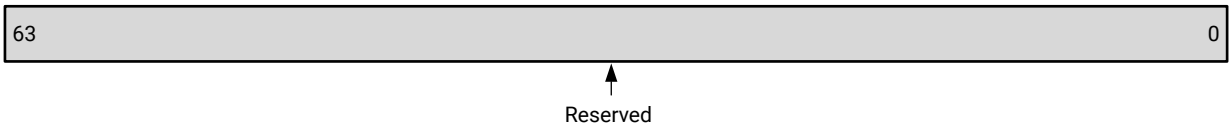


Table 45: **Primary Common Capabilities & Status III (DVSEC) Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:0				Reserved

Primary Common Control I (DVSEC) Register

Figure 33: **Primary Common Control I (DVSEC) Register**

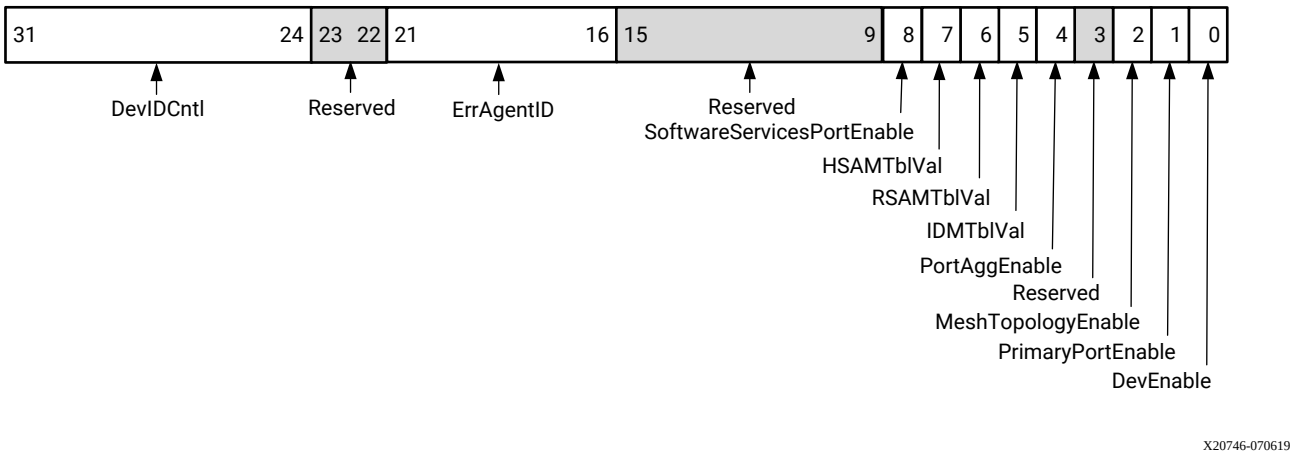


Table 46: **Primary Common Control I (DVSEC) Bit Definitions**

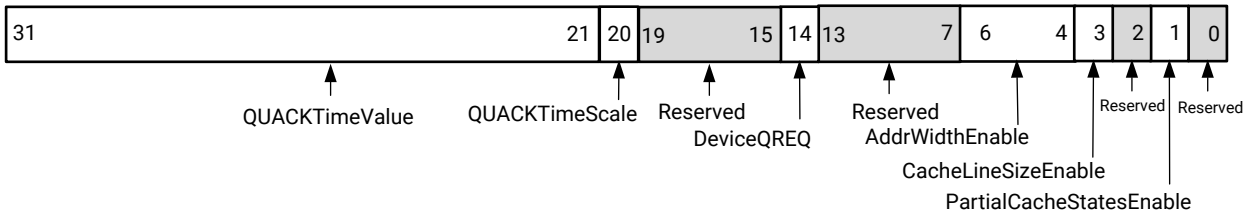
Bits	Name	Reset Value	Access	Description
31:24	DevIDCntl	0	R/W	FW: DIDC
23:22				Reserved
21:16	ErrAgentID	0	R/W	Error Agent ID
15:9				Reserved
8	SoftwareServicesPortEnable	0	R/W	FW: Software Service Portal Enable
7	HSAMTblVal	0	R/W	HSAM Table Valid
6	RSAMTblVal	0	R/W	RSAM Table Valid
5	IDMTblVal	0	R/W	FW: IDM Table Valid

Table 46: **Primary Common Control I (DVSEC) Bit Definitions** *(cont'd)*

Bits	Name	Reset Value	Access	Description
4	PortAggEnable	0	R/W	Port Aggregation Enable
3				Reserved
2	MeshTopologyEnable	0	R/W	Mesh Topology Enable
1	PrimaryPortEnable	0	R/W	Primary Port Enable
0	DevEnable	0	R/W	Device Enable

Primary Common Control II (DVSEC) Register

Figure 34: **Primary Common Control II (DVSEC) Register**



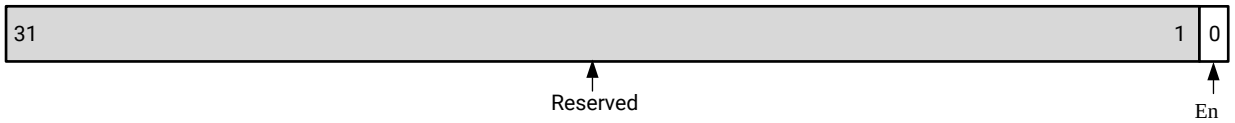
X20747-070619

Table 47: **Primary Common Control II (DVSEC) Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:21	QUACKTIMEValue	0	R/W	QUACK Time Value
20	QUACKTimeScale	0	R/W	QUACK Time Scale
19:15				Reserved
14	DeviceQREQ	0	R/W	CCIX Device QREQ
13:7				Reserved
6:4	AddrWidthEnable	0	R/W	Address Width Enable
3	CacheLineSizeEnable	0	R/W	Cacheline Size Enable
2				Reserved
1	PartialCacheStatesEnable	0	R/W	Partial Cache State Enable
0				Reserved

CCIX Device Error Control & Status Register

Figure 35: **CCIX Device Error Control & Status Register**



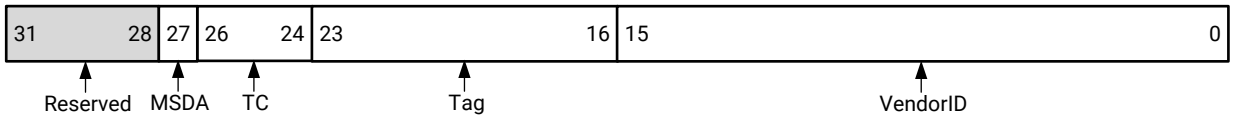
X22827-070619

Table 48: **CCIX Device Error Control & Status Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:1				Reserved
0	EN	0	R/W	Error Reporting Enable

Primary PCIe & CCIX Common Register

Figure 36: **Primary PCI & CCIX Common Register**



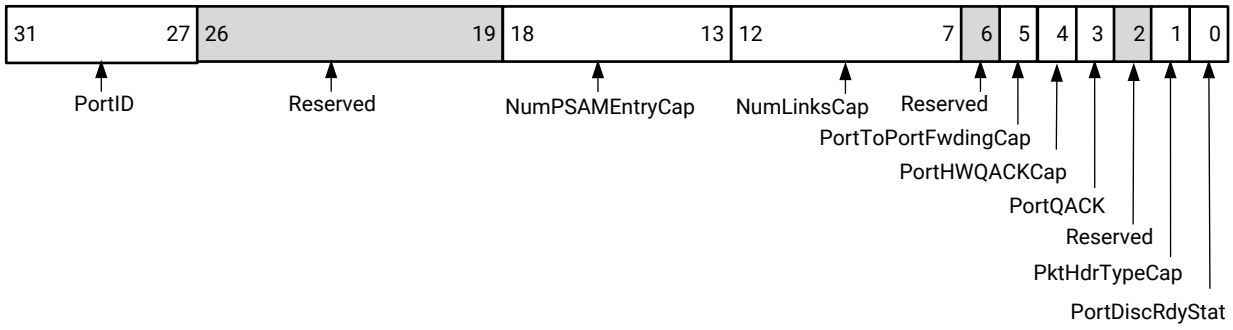
X20748-070619

Table 49: **Primary PCI & CCIX Common Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:28				Reserved
27	MSDA	0	R/W	Move to SD Allowed
26:24	TC	0	R/W	TC
23:16	Tag	0	R/W	Tag
15:0	VendorID	0	R/W	Vendor ID

Port Capability & Status I Register

Figure 37: **Port Capability & Status I Register**



X20749-070619

Table 50: **Port Capability & Status I Bit Definitions**

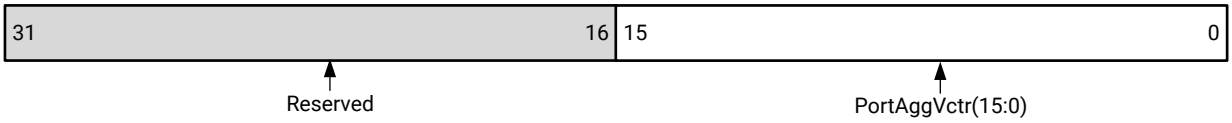
Bits	Name	Reset Value	Access	Description
31:27	PortÍD	0	R	CCIX Port ID

Table 50: **Port Capability & Status I Bit Definitions** *(cont'd)*

Bits	Name	Reset Value	Access	Description
26:19				Reserved
18:13	NumPSAMEntryCap	0x2	R	FW: Number of PSAM Entries Capabilities
12:7	NumLinksCap	0x1	R	Number of Links Capability
6				Reserved
5	PortToPortFwdingCap	0	R	Port-to-Port Forward Capability
4	PortHWQACKCap	0	R	Port HW QACK Capability
3	PortQAck	0	R	Port QUACK
2				Reserved
1	PktHdrTypeCap	0	R	Package Header Type Capability (Optimized)
0	PortDiscRdyStat	1	R	CCIX Port Discover Status

Port Capability & Status II Register

Figure 38: **Port Capability & Status II Register**



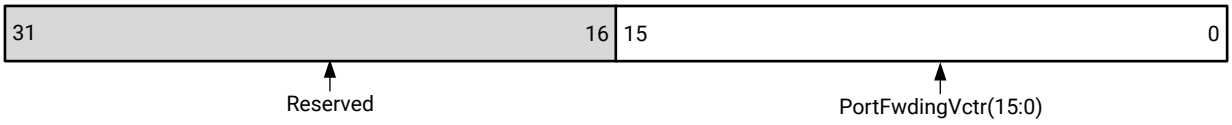
X20750-070619

Table 51: **Port Capability & Status II Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:16				Reserved
15:0	PortAggVctr	0	R	FW: Aggregation with PortID15-PortID0

Port Capability & Status III Register

Figure 39: **Port Capability & Status III Register**



X20751-070619

Table 52: **Port Capability & Status III Bit Definitions**

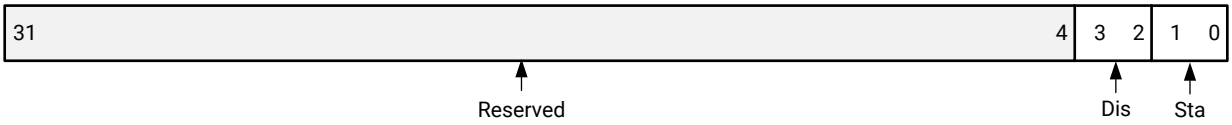
Bits	Name	Reset Value	Access	Description
31:16				Reserved

Table 52: **Port Capability & Status III Bit Definitions** *(cont'd)*

Bits	Name	Reset Value	Access	Description
15:0	PortFwdingvVctr	0	R	FW: Forwarding to PortID15-PortID0

Port Error Control & Status 0 Register

Figure 40: **Port Error Control & Status 0 Register**



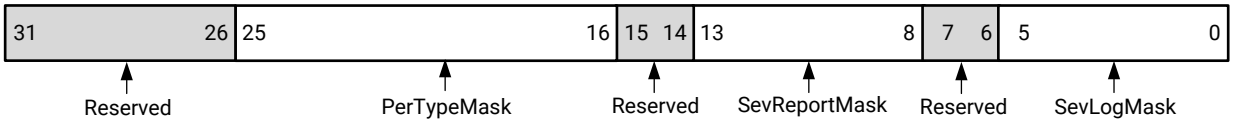
X20760-032420

Table 53: **Port Error Control & Status 0 Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:4				Reserved
3:2	Dis	0	R/W	PER Disable
1:0	Sta	0	R/W	Error Status

Port Error Control & Status 1 Register

Figure 41: **Port Error Control & Status 1 Register**



X20761-032420

Table 54: **Port Error Control & Status 1 Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:26				Reserved
25:16	PerTypeMask	0	R/W	PER Type Mask
15:14				Reserved
13:8	SevReportMask	0	R/W	Severity Reporting Mask
7:6				Reserved
5:0	SevLogMask	0	R/W	Severity Logging Mask

Port Control I Register

Figure 42: **Port Control I Register**

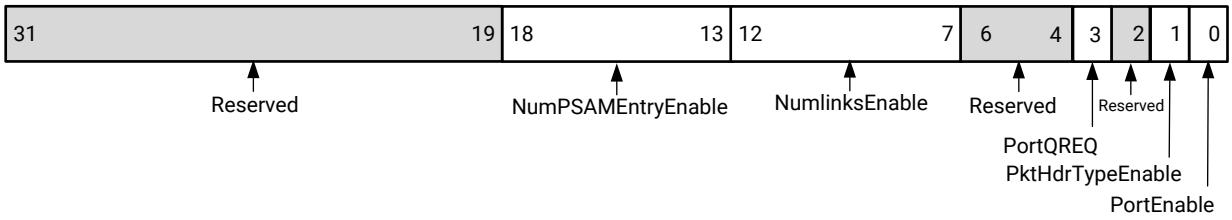


Table 55: **Port Control I Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:19				Reserved
18:13	NumPSMAEntryEnable	0	R/W	FW: Number of PSAM Entries Enable
12:7	NumLinksEnable	0	R/W	Number of Links Enable
6:4				Reserved
3	PortQREQ	0	R/W	Port QREQ
2				Reserved
1	PktHdrTypeEnable	0	R/W	Optimized Packet Header Type Enable
0	PortEnable	0	R/W	Port Enable

Port Source Transport ID Map Register

Figure 43: **Port Source Transport ID Map Register**

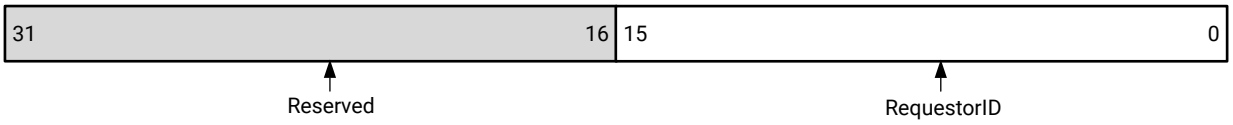
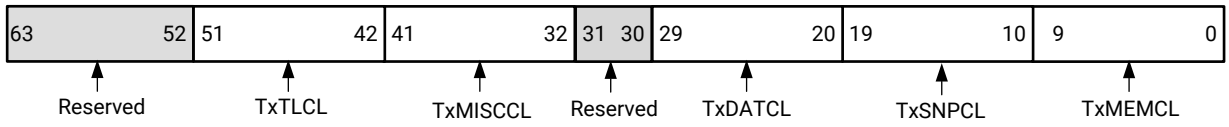


Table 56: **Port Source Transport ID Map Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:16				Reserved
15:0	RequestorID	0	R/W	Requestor ID

Port Link Tx Status Register

Figure 44: **Port Link Tx Status Register**



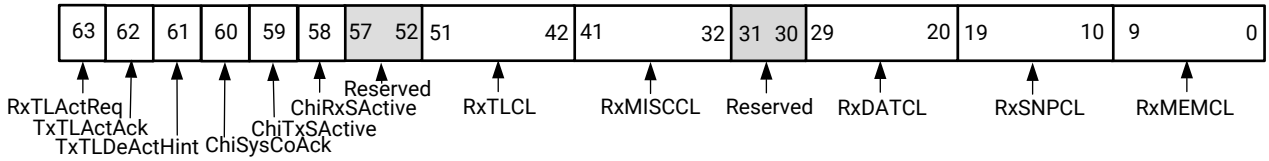
X20754-062220

Table 57: **Port Link Tx Status Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:52				Reserved
51:42	TxTLCL	0	R	CCIX: CXS Transaction Layer Credit Level available for use / CHI: Reserved
41:32	TxMISCCL	0	R	CCIX: Misc Credit Level available for use / CHI: TX Rsp Credit Level available for use
31:30				Reserved
29:20	TxDATCL	0	R	CCIX: Data Credit Level available for use / CHI: TX Data Credit Level available for use
19:10	TxSNPCL	0	R	CCIX: Snoop Credit Level available for use / CHI: RX Snoop Credit Level available for use
9:0	TxMEMCL	0	R	CCIX: Memory Request Credit Level available for use / CHI: TX Req Credit Level available for use

Port Link Rx Status Register

Figure 45: **Port Link Rx Status Register**



X23019-072021

Table 58: **Port Link Rx Status Bit Definitions**

Bits	Name	Reset Value	Access	Description
63	RxTLActReq	0	R	CCIX: CXS0_ACTIVE_REQ_RX value / CHI: M0_CHI_RXLINKACTIVEREQ value
62	TxTLActAck	0	R	CCIX: CXS0_ACTIVE_ACK_TX value / CHI: M0_CHI_TXLINKACTIVEACK value

Table 58: **Port Link Rx Status Bit Definitions** *(cont'd)*

Bits	Name	Reset Value	Access	Description
61	TxTLDeActHint	0	R	CCIX: CXS0_DEACT_HINT_TX value / CHI: RXLINK goes to Deactivate state
60	ChiSysCoAck	0	R	CCIX: Reserved / CHI: M0_CHI_SYSCOACK value
59	ChiTxSActive	0	R	CCIX: Reserved / CHI: M0_CHI_TXSACTIVE value
58	ChiRxSActive	0	R	CCIX: Reserved / CHI: M0_CHI_RXSACTIVE value
57:52				Reserved
51:42	RxTLCL	0xF	R	CCIX: CXS Transaction Layer Credit Level that can be granted / CHI: Reserved
41:32	RxMISCCL	0	R	CCIX: Misc Credit Level that can be granted / CHI: RX Rsp Credit Level that can be granted
31:30				Reserved
29:20	RxDATCL	0	R	CCIX Data Credit Level that can be granted / CHI: RX Data Credit Level that can be granted
19:10	RxSNPCL	0	R	CCIX: Snoop Credit Level that can be granted / CHI: RX Snoop Credit Level that can be granted
9:0	RxMEMCL	0	R	CCIX: Memory Request Credit Level that can be granted / CHI: Reserved

Port Link BFCV0+BFCV1 Register

Figure 46: **Port Link BFCV0+BFCV1 Register**

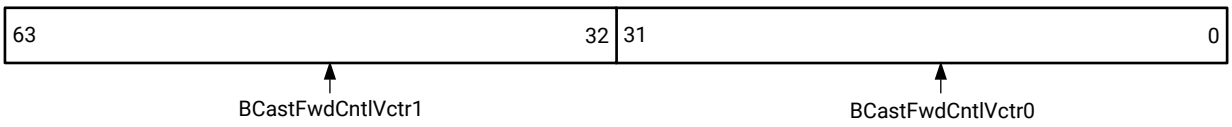
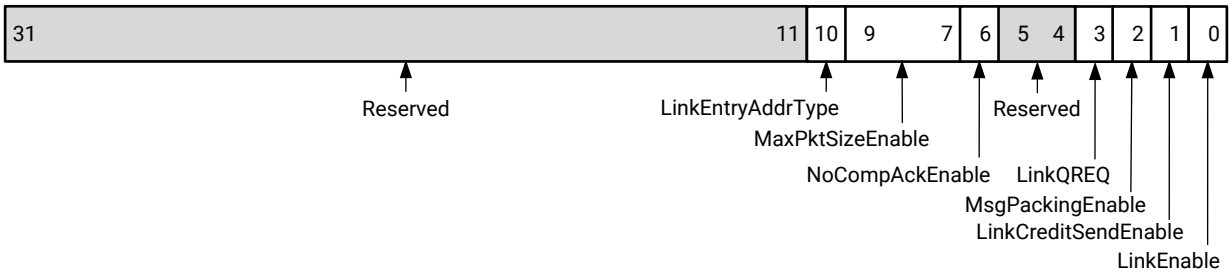


Table 59: **Port Link BFCV0+BFCV1 Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:32	BCastFwdCntlVctr1	0	R/W	FW: Broadcast Forward Control Vector1
31:0	BCastFwdCntlVctr0	0	R/W	FW: Broadcast Forward Control Vector 0

Port Link Control I Register

Figure 47: **Port Link Control I Register**



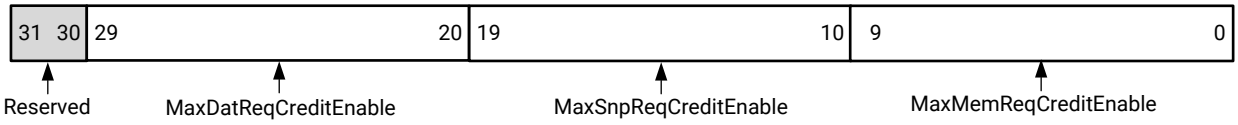
X20755-070619

Table 60: **Port Link Control I Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:11				Reserved
10	LinkEntryAddrType	0	R/W	FW: Link Entry Address Type
9:7	MaxPktSizeEnable	0	R/W	Max Package Size Enable
6	NoCompAckEnable	0	R/W	NoCompAck Enable
5:4				Reserved
3	LinkQREQ	0	R/W	Link Quiescent Request
2	MsgPackingEnable	0	R/W	Message Packing Entry
1	LinkCreditSendEnable	0	R/W	CCIX Link Credit Send Enable
0	LinkEnable	0	R/W	CCIX Link Enable

Port Link Control II Register

Figure 48: **Port Link Control II Register**



X20757-070719

Table 61: **Port Link Control II Bit Definitions**

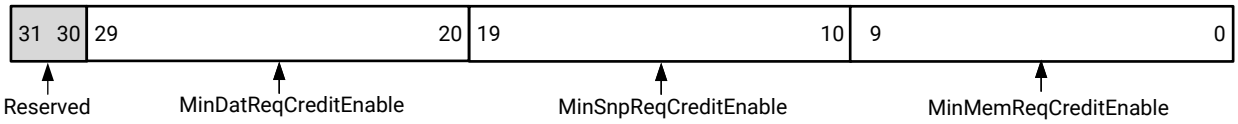
Bits	Name	Reset Value	Access	Description
31:30				Reserved
29:20	MaxDatReqCreditEnable	0	R/W	Maximum Data Request Credit Enable
19:10	MaxSnpReqCreditEnable	0	R/W	Maximum Snoop Request Credit Enable

Table 61: **Port Link Control II Bit Definitions** *(cont'd)*

Bits	Name	Reset Value	Access	Description
9:0	MaxMemReqCreditEnable	0	R/W	Maximum Memory Request Credit Enable

Port Link Control III Register

Figure 49: **Port Link Control III Register**



X20758-070719

Table 62: **Port Link Control III Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:30				Reserved
29:20	MinDatReqCreditEnable	0	R/W	Minimum Data Request Credit Enable
19:10	MinSnpReqCreditEnable	0	R/W	Minimum Snoop Request Credit Enable
9:0	MinMemReqCreditEnable	0	R/W	Minimum Memory Request Credit Enable

Port Link Control IV Register

Figure 50: **Port Link Control IV Register**

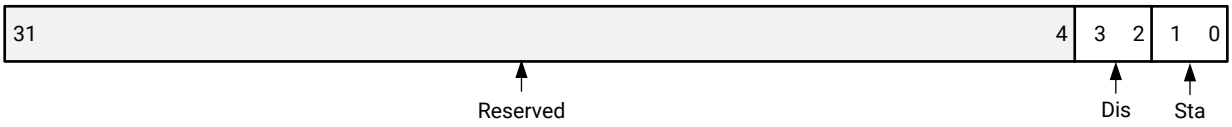
X20759-070719

Table 63: **Port Link Control IV Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:20				Reserved
19:10	MinMiscReqCreditEnable	0	R/W	Minimum Request Credit Enable
9:0	MaxMiscReqCreditEnable	0	R/W	Maximum Misc Request Credit Enable

Port Link Error Control & Status 0 Register

Figure 51: **Port Link Error Control & Status 0 Register**



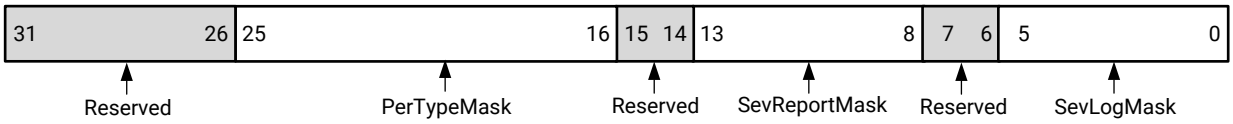
X20760-032420

Table 64: **Port Link Error Control & Status 0 Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:4				Reserved
3:2	Dis	0	R/W	PER Disable
1:0	Sta	0	R/W	Error Status

Port Link Error Control & Status 1 Register

Figure 52: **Port Link Error Control & Status 1 Register**



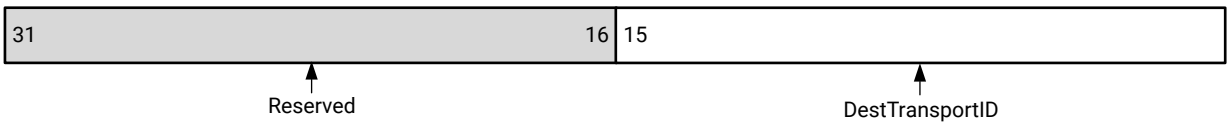
X20761-032420

Table 65: **Port Link Error Control & Status 1 Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:26				Reserved
25:16	PerTypeMask	0	R/W	PER Type Mask
15:14				Reserved
13:8	SevReportMask	0	R/W	Severity Reporting Mask
7:6				Reserved
5:0	SevLogMask	0	R/W	Severity Logging Mask

Port Link Destination TransportID Map Register

Figure 53: **Port Link Destination TransportID Map Register**



X20762-070719

Table 66: **Port Link Destination TransportID Map Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:16				Reserved
15:0	DestTransportID	0	R/W	Destination Transport ID

SC Specific Port Link Specific Register

Figure 54: **SC Specific Port Link Specific Register**

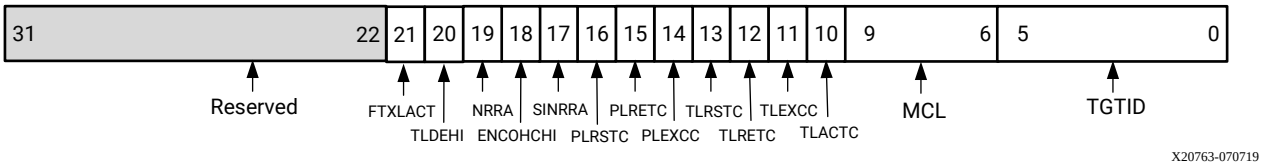


Table 67: **SC Specific Port Link Specific Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:22				Reserved
21	FTXLACT	0	R/W	Reserved (Force TX Link activation)
20	TLDEHI	0	R/W	CCIX: Transaction Layer Deactivation Hint / CHI: Reserved
19	NRRA	0	R/W	Reserved (No Remote RA)
18	ENCOHCHI	0	R/W	CCIX: Reserved / CHI: Enable coherency by using the System Coherency handshake
17	SNRRA	0	R/W	Reserved (Single Remote RA)
16	PLRSTC	0	R/W	CCIX: Protocol Layer Reset Credits / CHI: Reserved
15	PLRETC	0	R/W	CCIX: Protocol Layer Return Credits / CHI: Reserved
14	PLEXCC	0	R/W	CCIX: Protocol Layer Exchange Credits / CHI: Reserved
13	TLRSTC	0	R/W	Transaction Layer Reset Credits
12	TLRETC	0	R/W	Transaction Layer Return Credits
11	TLEXCC	0	R/W	Transaction Layer Exchange Credits
10	TLACTC	0	R/W	Transaction Layer Activate Connection
9:6	MCL	0	R/W	Reserved (Max Chain Length)
5:0	TGTID	0	R/W	CCIX: TgtID, when Sending Credit Exchange / CHI: Reserved

Port Link Capability & Status I Register

Figure 55: **Port Link Capability & Status I Register**

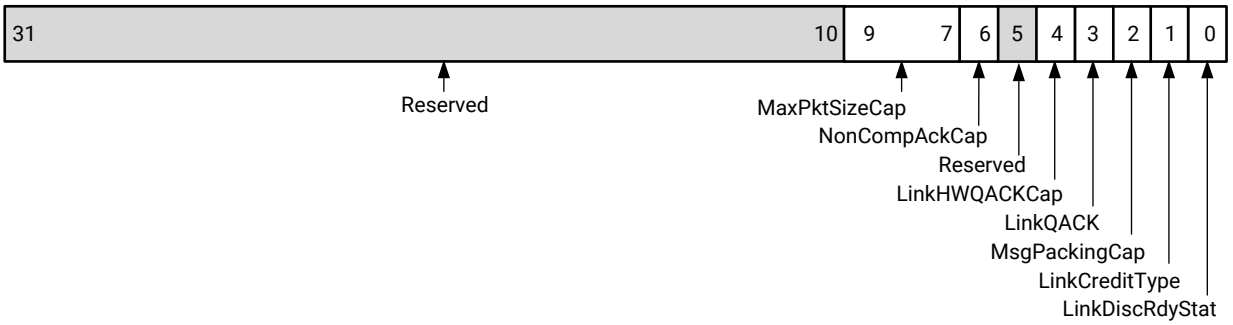


Table 68: **Port Link Capability & Status I Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:10				Reserved
9:7	MaxPktSizeCap	0	R	Max Package Size Capability
6	NoCompAckCap	0	R	NoCompAck Capability
5				Reserved
4	LinkHWQACKCap	0	R	Link's Hardware Quiesce Acknowledgment Capability
3	LinkQACK	0	R	Link's Quiesce Acknowledgment status
2	MsgPackingCap	0	R	Message Packing Capability
1	LinkCreditType	0	R	CCIX Link Credit Type
0	LinkDiscRdyStat	1	R	CCIX Link Discovery Status

Port Link Capability & Status II Register

Figure 56: **Port Link Capability & Status II Register**

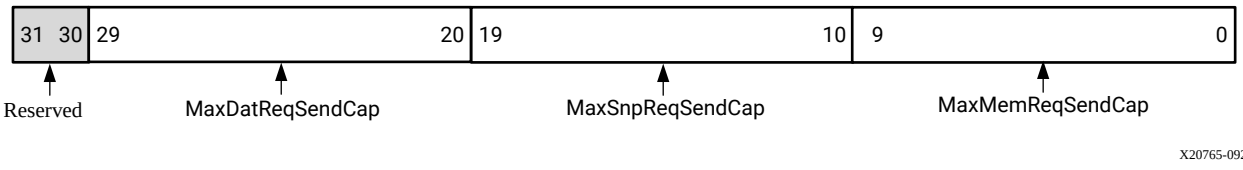


Table 69: **Port Link Capability & Status II Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:30				Reserved
29:20	MaxDatReqSendCap	N[1]	R	Maximum Data Request Send Capability
19:10	MaxSnpReqSendCap	M[2]	R	Maximum Snoop Request Send Capability

Table 69: **Port Link Capability & Status II Bit Definitions** *(cont'd)*

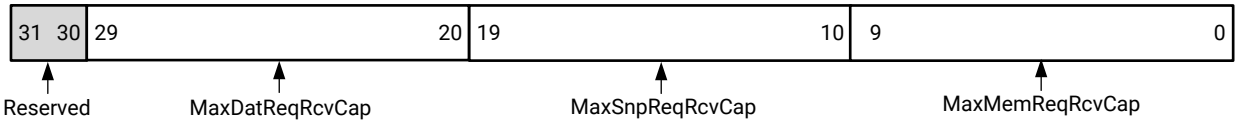
Bits	Name	Reset Value	Access	Description
9:0	MaxMemReqSendCap	N[1]	R	Maximum Memory Request Send Capability

Notes:

1. N = C_NUM_SLAVE_TRANSACTIONS * (C_NUM_OPTIMIZED_PORTS + C_NUM_GENERIC_PORTS)
2. M = C_NUM_SNOOP_TRANSACTIONS

Port Link Capability & Status III Register

Figure 57: **Port Link Capability & Status III Register**



X20766-092419

Table 70: **Port Link Capability & Status III Bit Definitions**

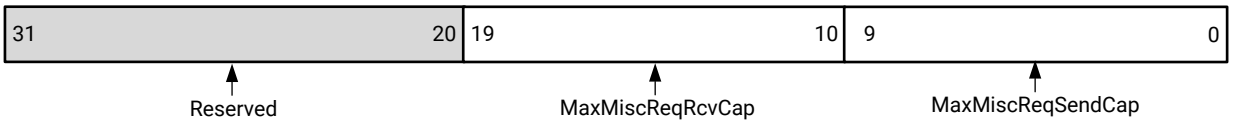
Bits	Name	Reset Value	Access	Description
31:30				Reserved
29:20	MaxDatReqRcvCap	N[1]	R	Maximum Data Request Receive Capability
19:10	MaxSnpReqRcvCap	M[2]	R	Maximum Snoop Request Receive Capability
9:0	MaxMemReqRcvCap	N[1]	R	Maximum Memory Request Receive Capability

Notes:

1. N = C_NUM_MASTER_TRANSACTIONS / C_NUM_LINKS
2. M = C_NUM_SNOOP_TRANSACTIONS / C_NUM_LINKS

Port Link Capability & Status IV Register

Figure 58: **Port Link Capability & Status IV Register**



X20767-070719

Table 71: **Port Link Capability & Status IV Bit Definitions**

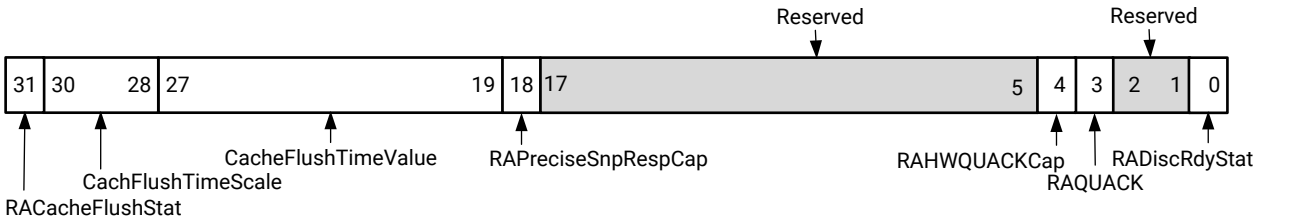
Bits	Name	Reset Value	Access	Description
31:20				Reserved
19:10	MaxMiscReqRcvCap	0x10	R	Maximum Misc Request Send Capability

Table 71: **Port Link Capability & Status IV Bit Definitions** *(cont'd)*

Bits	Name	Reset Value	Access	Description
9:0	MaxMiscReqSendCap	0x10	R	Maximum Misc Request Receive Capability

RA Capability & Status I Register

Figure 59: **RA Capability & Status I Register**



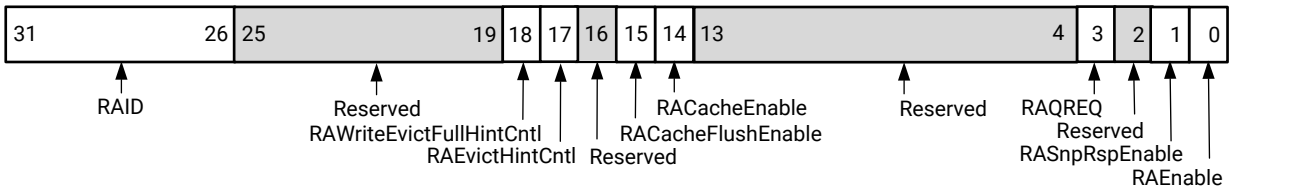
X20775-072919

Table 72: **RA Capability & Status I Bit Definitions**

Bits	Name	Reset Value	Access	Description
31	RACacherFlushStat	0	R	Cache Flush Status
30:28	CacheFlushTimeScale	0	R	Cache Flush Completion Time Scale
27:19	CacheFlushTimeValue	0	R	Cache Flush Completion Time Value
18	RAPreciseSnpRespCap	0	R	RA Precise Snp Resp Capability
17:5				Reserved
4	RAHWQUACKCap	0	R	RA HW QUACK Capability
3	RAQUACK	0	R	RA QACK
2:1				Reserved
0	RADiscRdyStat	1	R	Request Agent Discovery Status

RA Control I Register

Figure 60: **RA Control I Register**



X20777-072919

Table 73: **RA Control I Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:26	RAID	0	R/W	Request Agent ID
25:19				Reserved
18	RAWriteEvictFullHintCntl	0	R/W	RA WriteEvictFull Hint Control
17	RAEvictHintCntl	0	R/W	RA Evict Hint Control
16				Reserved
15	RACacheFlushEnable	0	R/W	Cache Flush Enable
14	RACacheEnable	0	R/W	Cache Enable
13:4				Reserved
3	RAQREQ	0	R/W	RA QREQ
2				Reserved
1	RASnpRspEnable	0	R/W	RA Snoop Response Enable
0	RAEnable	0	R/W	Request Agent Enable

RA Error Control Status 0 Register

Figure 61: **RA Error Control Status 0 Register**

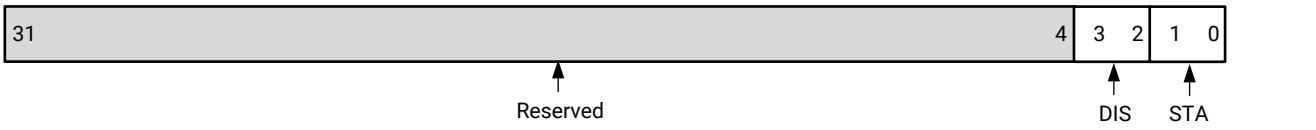


Table 74: **RA Error Control Status 0 Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:4				Reserved
3:2	DIS	0	R/W	PER Disable (LogDis and Dis)
1:0	STA	0	R/W	Error Status (Sta)

RA Error Control Status 1 Register

Figure 62: **RA Error Control Status 1 Register**

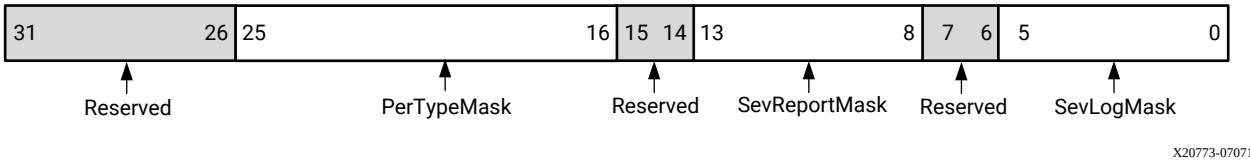


Table 75: **RA Error Control Status 1 Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:26				Reserved
25:16	PerTypeMask	0	R/W	PER Type Mask
15:14				Reserved
13:8	SevReportMask	0	R/W	Severity Reporting Mask
7:6				Reserved
5:0	SevLogMask	0	R/W	Severity Logging Mask

RA System Cache Specific Register

Figure 63: **RA System Cache Specific Register**

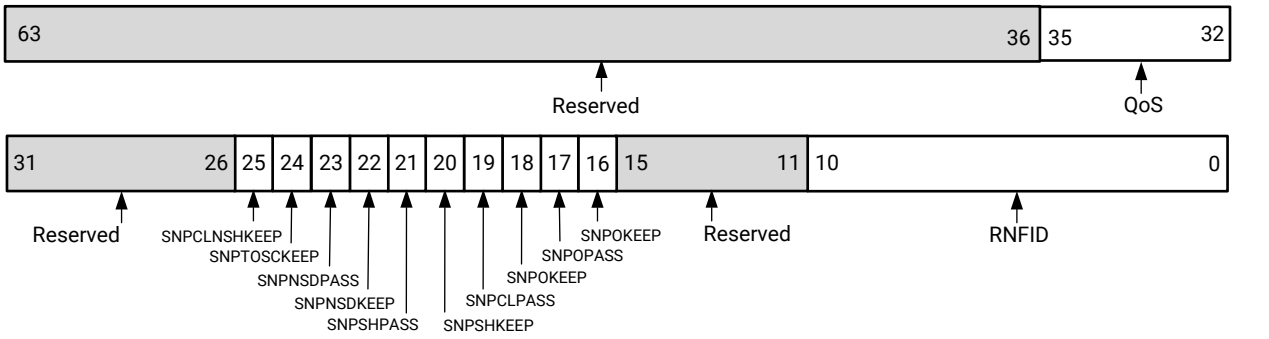


Table 76: **RA System Cache Specific Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:45				Reserved
44	SNPNSDFWDPASS	0	R/W	SnpNSDFwd Pass Dirty
43	SNPNSDFWDKEEP	0	R/W	SnpNSDFwd Keep Line
42	SNPSHFWDHOME	0	R/W	SnpSharedFwd Dirty Home, move dirty data to Home instead of requesting RN
41	SNPSHFWDPASS	0	R/W	SnpSharedFwd Pass Dirty
40	SNPSHFWDKEEP	0	R/W	SnpSharedFwd Keep Line
39	SNPCLFWDPASS	0	R/W	SnpCleanFwd Pass Dirty
38	SNPCLFWDKEEP	0	R/W	SnpCleanFwd Keep Line
37	SNPOFWDPASS	0	R/W	SnpOnceFwd Pass Dirty
36	SNPOFWDKEEP	1	R/W	SnpOnceFwd Keep Line
35:32	QoS	0x10	R/W	Quality of Service
31:26				Reserved
25	SNPCLNSHKEEP	0	R/W	SnpCleanShared Keep Line
24	SNPTOSCKEEP	0	R/W	SnpToSC Keep Line

Table 76: **RA System Cache Specific Bit Definitions** *(cont'd)*

Bits	Name	Reset Value	Access	Description
23	SNPNSDPASS	0	R/W	SnpNSD Pass Dirty
22	SNPNSDKEEP	0	R/W	SnpNSD Keep Line
21	SNPSHPASS	0	R/W	SnpShared Pass Dirty
20	SNPSHKEEP	0	R/W	SnpShared Keep Line
19	SNPCLPASS	0	R/W	SnpClean Pass Dirty
18	SNPCLKEEP	0	R/W	SnpClean Keep Line
17	SNPOPASS	0	R/W	SnpOnce Pass Dirty
16	SNPOKEEP	1	R/W	SnpOnce Keep Line
15:11				Reserved
10:0	RNFID	0	R/W	RN-F NodeID (CHI only)

Backend ATS Registers

Backend ATS Address Map

In the nominal use case with CHI or CCIX protocol context the following registers are handled by firmware executing in the MicroBlaze processor sub-system, and there is nothing the user needs to handle.

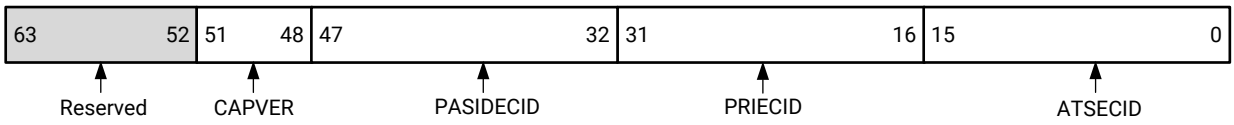
Table 77: **Backend ATS Address Map**

Offset	Register Name	Format	Description
0x1_9800	PASIDATSEXTCAP	64	PASID & ATS Extended Capability Header Note: Currently unused, the external processor handles PCIe configuration in FW.
0x1_9808	PASIDATSCAP	64	PASID & ATS Capability Header
0x1_9810	ATSPAGEREQCAP	64	ATS Outstanding Page Request Capacity
0x1_9818	ATSPAGEREQALL	64	ATS Outstanding Page Request Allocation
0x1_9820	ATSPRICTRL	64	ATS & PRI Control
0x1_9828	PASIDCTRL	64	PASID Control
0x1_9840	ATSPAGEREQSTAT	64	ATS Page Request Status
0x1_9860	ATSPCIECTRL	64	ATS PCIe Control
0x1_9870	ATSPCIEEXTCTRL	64	ATS PCIe Extended Control

PASID & ATS Extended Capability Header Register

This register holds the System Cache PCIe configuration capabilities. Firmware replaces this information with locally defined PCIe endpoint capabilities.

Figure 64: **PASID & ATS Extended Capability Header Register**



X20736-073019

Table 78: **PASID & ATS Extended Capability Header Bit Definitions**

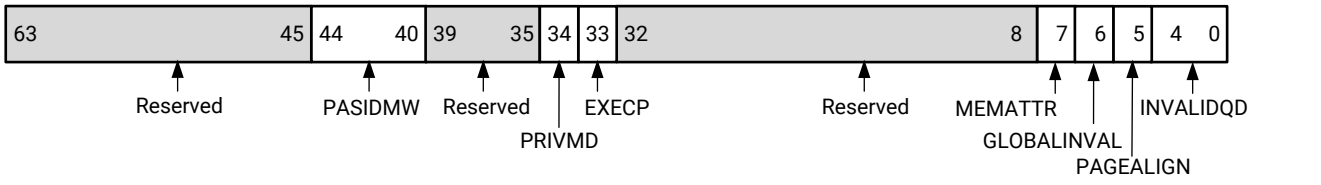
Bits	Name	Reset Value	Access	Description
63:52				Reserved
51:48	CAPVER	0	R	Capability version
47:32	PASIDECID	0	R	PASID Extended Capability ID
31:16	PRIECID	0	R	PRI Extended Capability ID
15:0	ATSCID	0	R	PASID & ATS Extended Capability Header ID

PASID & ATS Capability Header Register

This register holds the System Cache PCIe PASID and ATS configuration capabilities. Firmware reads this information and combines it into a System Cache capabilities PCIe Configuration response.

***Note*:** System Cache uses bit 7 for MEMATTR as originally proposed in earlier PCI Express ATS expansions, whereas the PCI Express Base Specification Revision 5.0 Version 1.0 implies that bit 8 should be used – as bit 7 is used for Relaxed Ordering Supported (not used by System Cache) in the PCI Express specification.

Figure 65: **PASID & ATS Capability Header Register**



X20737-073019

Table 79: **PASID & ATS Capability Header Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:45				Reserved
44:40	PASIDMW	0x14	R	Max PASID Width (Used when CHI PASID enabled)
39:35				Reserved
34	PRIVMD	0	R	Privileged Mode Supported (Used when CHI PASID enabled)

Table 79: **PASID & ATS Capability Header Bit Definitions** *(cont'd)*

Bits	Name	Reset Value	Access	Description
33	EXECP	0	R	Execute Permission Supported (Used when CHI PASID enabled)
32:8				Reserved
7	MEMATTR	1	R	Memory Attributes Supported
6	GLOBALINVAL	1	R	Global Invalidate Supported (Used when CHI PASID enabled)
5	PAGEALIGN	1	R	Page Align Request
4:0	INVALIDQD	1	R	Invalidate Queue Depth

ATS Outstanding Page Request Capacity Register

This register holds the System Cache ATS PRG configuration capabilities. Firmware reads this information and combines it into a System Cache capabilities PCIe Configuration response.

Figure 66: **ATS Outstanding Page Request Capacity Register**

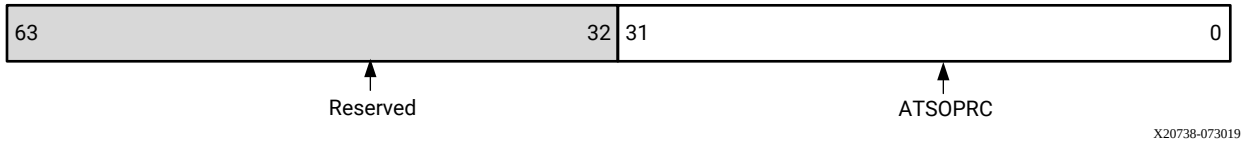


Table 80: **ATS Outstanding Page Request Capacity Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:32				Reserved
31:0	ATSOPRC	0xF	R	ATS Outstanding Page Request Capacity

ATS Outstanding Page Request Allocation Register

This register holds the System Cache ATS PRG Allocation. Firmware writes this information during configuration, but it is allowed to be changed after System Cache initialization.

Figure 67: **ATS Outstanding Page Request Allocation Register**

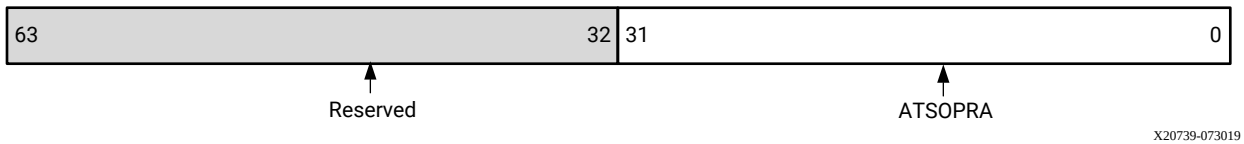


Table 81: **ATS Outstanding Page Request Allocation Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:32				Reserved
31:0	ATSOPRA	0	R/W	ATS Outstanding Page Request Allocation

ATS & PRI Control Register

This register holds the System Cache ATS and PRI control. Firmware writes this information during configuration, but it is allowed to be changed after System Cache initialization.

If ATS Enable is changed from 0 to 1, a silent complete invalidation of the ATC Table is performed.

Figure 68: **ATS & PRI Control Register**

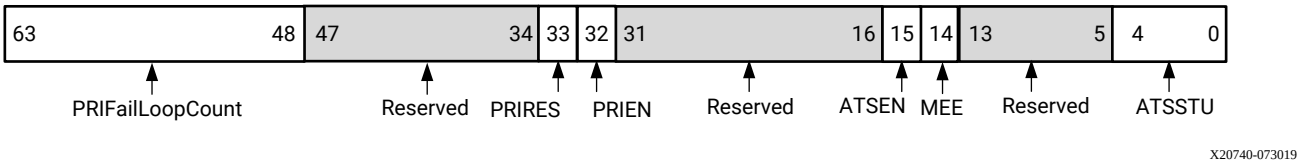


Table 82: **ATS & PRI Control Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:48	PRIFailLoopCount	0	R/W	Fail Loop Count
47:34				Reserved
33	PRIRES	0	R/W	PRI Reset
32	PRIEN	0	R/W	PRI Enable
31:16				Reserved
15	ATSEN	0	R/W	ATS Enable
14	MEE	0	R/W	Mem Attributes Enable
13:5				Reserved
4:0	ATSSTU	0	R/W	ATS Smallest Translation Unit

PASID Control Register

This register holds the System Cache PASID control. Firmware writes this information as a result of the PCIe enumeration and configuration phase. It should not be changed after System Cache initialization.

Figure 69: **PASID Control Register**

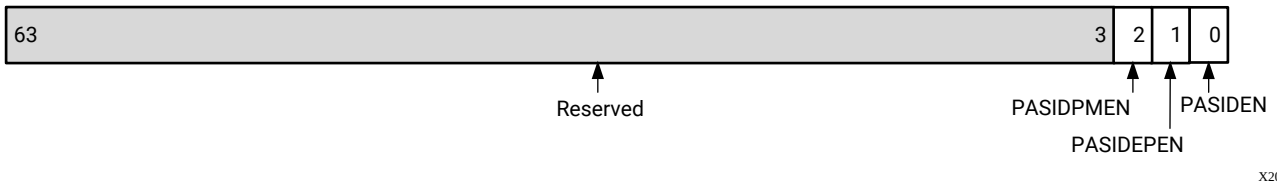


Table 83: **PASID Control Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:3				Reserved
2	PASIDPMEN	0	R/W	PASID Privilege Mode Enable (Valid in CHI PASID context, otherwise Reserved)
1	PASIEPEN	0	R/W	PASID Execute Permission Enable (Valid in CHI PASID context, otherwise Reserved)
0	PASIDEN	0	R/W	PASID Enable (Valid in CHI PASID context, otherwise Reserved)

ATS Page Request Status Register

This register holds the System Cache ATS PRG status. Firmware writes this information during configuration, but it is allowed to be changed after System Cache initialization.

Figure 70: **ATS Page Request Status Register**

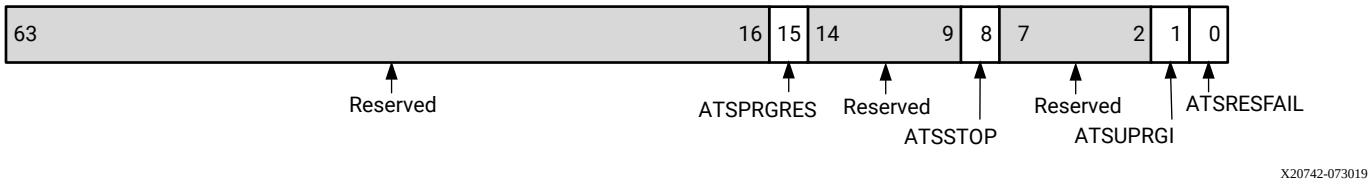


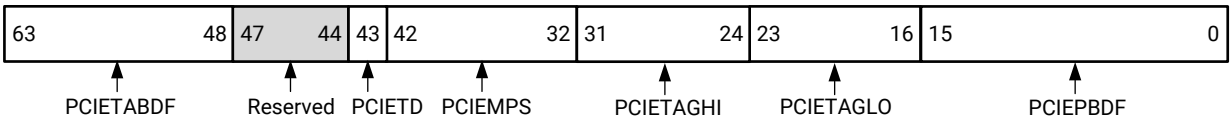
Table 84: **ATS Page Request Status Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:16				Reserved
15	ATSPRGRES	0	R	PRG Response PASID Required (Valid in CHI PASID context, otherwise Reset Value)
14:9				Reserved
8	ATSSTOP	1	R	PRG Stop
7:2				Reserved
1	ATSUPRGI	0	R/W	Unexpected Page Request Group Index
0	ATSRESFAIL	0	R/W	Response Failure

ATS PCIe Control Register

This register holds the System Cache ATS PCIe configuration control. Currently the system-level processor firmware writes this information as a result of the PCIe enumeration and configuration phase. It should not be changed after System Cache initialization.

Figure 71: **ATS PCIe Control Register**



X20743-073019

Table 85: **ATS PCIe Control Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:48	PCIETABDF	0	R/W	PCIe TA BDF, Root Complex BDF (used by ATS/CCIX)
47:44				Reserved
43	PCIETD	0	R/W	PCIe TD, Optional ECRC Enable
42:32	PCIEMPS	0	R/W	PCIe MPS, (used by ATS/CCIX)
31:24	PCIETAGHI	0xFF	R/W	PCIe 8bits Tag High, limited unique range if dual System Cache in EP
23:16	PCIETAGLO	0	R/W	PCIe 8bits Tag Low
15:0	PCIEEPBDF	0	R/W	PCIe EP BDF, (used by ATS/CCIX)

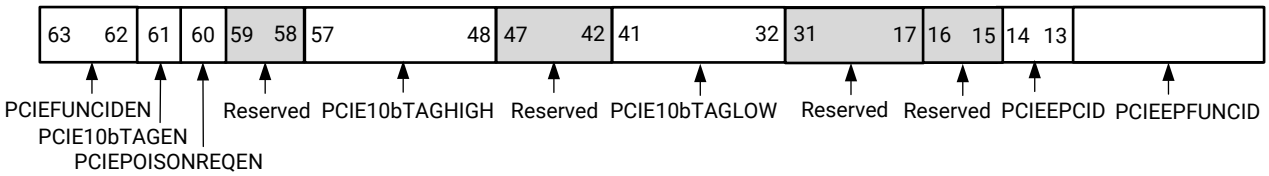
ATS PCIe Extended Control Register

This register holds the System Cache ATS PCIe Extended configuration control. Currently, the system level processor firmware writes this additional information as a result of the PCIe enumeration and configuration phase. The extended configuration allows none, partial, full, or enhanced selection of FunctionID for self identification. In addition to legacy EP BDF, the register captures EP FunctionID for EP identification – usage of EP BDF / EP FuntionID depends upon PCIEFUNCIDEN settings.

Optional 10 bits TAG support now enables System Cache to allow the 10 bits TAG option with Host capabilities support (PCIeRev4.0 onwards - System Cache present refer to PCIeRev5.0). With 10 bits TAG mode enabled, this register holds TAG Low and TAG High ranges for 10bits TAG, and supersedes 8bits TAG Low and TAG High ranges in ATS PCIe Control Register. Settings in this register should not be changed after System Cache initialization.

Note: System Cache expects selection by PCIEFUNCIDEN, to enable EPBDF override with FUNCTIONID/COMPONENTID - None/Partial/Full/Enhanced, based upon target technology. User firmware is expected to set PCIEFUNCIDEN, PCIEEPBDF and FUNCTIONID/COMPONENTID to the correct system values during the configuration phase. For backwards compatibility, PCIEFUNCIDEN with default Reset value, `0b00`, will select the PCIEEPBDF settings.

Figure 72: **ATS PCIe Extended Control Register**



X25467-072021

Table 86: **ATS PCIe Extended Control Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:62	PCIEFUNCIDEN	0	R/W	PCIe Function ID Enable, (used by ATS/CCIX) • `0b00` = Disable EP FUNCTIONID and COMPONENT ID, use EP BDF at all TLPs on Completer ID/Requester ID/Device ID PCIe Configuration register. • `0b01` = Enable EP Partial FUNCTIONID at TLPs on Completer ID/Requester ID, EP Self identification Device ID = EP BDF (PCIe Configuration register. • `0b10` = Enable EP FUNCTIONID at all TLPs on Completer ID/Requester ID/Device ID, EP BDF (PCIe Configuration register) and COMPONENT ID unused. • `0b11` = Reserved
61	PCIE10bTAGEN	0	R/W	PCIe 10bit Tag Enable, (used by ATS/CCIX) • `0` = 8bit Tag – overloaded T9/T8 bits normal mode • `1` = Enable 10bit Tag extension on Completer and Requester Interface
60	PCIEPOISONREQEN	0	R/W	PCIe Poison Request Enable (extend the Poison control if 10bits TAG mode are supported, as field are consumed), (used by ATS/CCIX) bit unused in 8bits TAG mode • `0` = Disable Completer poison completion(CC) / Requester poison request(RQ) upon NonCorr Payload Error, if AER Disabled on CC/RQ • `1` = Enable Completer poison completion(CC) / Requester poison request(RQ) upon NonCorr Payload Error, if AER Enabled on CC/RQ
59:58				Reserved
57:48	PCIE10bTAGHIGH	0x3FF	R/W	PCIe 10bits Tag High, override 8bits Tag Low - if 10bit Tag Enable bit is set. Limited unique range if dual System Cache in EP
47:42				Reserved
41:32	PCIE10bTAGLOW	0	R/W	PCIe 10bits Tag Low, override 8bits Tag Low - if 10bit Tag Enable bit is set
31:15				Reserved (EP BAR ID, PF# presence)
14:13	PCIEEPCID	0	R/W	PCIe EP Component ID. Reserved - should always be set to 0.
12:0	PCIEEPFUNCID	0	R/W	PCIe EP FunctionID, (used in partial/full and enhanced mode by ATS/CCIX)

Control Registers

Table 87: **Control Register Address Map**

Offset	Register Name	Access	Format	Description
0x1_C000	Command	R/W	COMMAND	Command register
0x1_C008	StatEn	W	STATEN	Statistics Enable
0x1_C010	CMONonSecClean	W	CMOADDR	CMO - NonSecure Clean
0x1_C018	CMONonSecFlush	W	CMOADDR	CMO - NonSecure Flush
0x1_C020	Version0	R	VERSION0	Version Register 0, Basic
0x1_C024	Version1	R	VERSION1	Version Register 1, Extended
0x1_C030	DVMNonSecFirst	W	DVMDATA	DVM - NonSecure First Beat
0x1_C038	DVMNonSecSecond	W	DVMDATA	DVM - NonSecure Second Beat
0x1_C040	CMONonSecMem	W	CMOADDR	Barrier - NonSecure Memory
0x1_C048	CMONonSecSynch	W	CMOADDR	Barrier - NonSecure Synchronization
0x1_C050	CMOSecClean	W	CMOADDR	CMO - Secure Clean
0x1_C058	CMOSecFlush	W	CMOADDR	CMO - Secure Flush
0x1_C060	DVMSecFirst	W	DVMDATA	DVM - Secure First Beat
0x1_C068	DVMSecSecond	W	DVMDATA	DVM - Secure Second Beat
0x1_C070	CMOBarSecmem	W	CMOADDR	Barrier - Secure Memory
0x1_C078	CMOBarSecSynch	W	CMOADDR	Barrier - Secure Synchronization
0x1_C080	CMOSecCleanSh	W	CMOADDR	CMO - Secure CleanShared
0x1_C088	CMONonSecClnSh	W	CMOADDR	CMO - NonSecure CleanShared
0x1_C090	Reserved	W	CMOADDR	Reserved
0x1_C098	Reserved	W	CMOADDR	Reserved
0x1_C0C0	IRQStatus	R/W	IRQ	Interrupt Status
0x1_C0C8	IRQEnable	R/W	IRQ	Interrupt Enable
0x1_C0D0	IRQPending	R	IRQ	Interrupt Pending
0x1_C0E0	IntegCmd	R/W	INTEGCMD	Integrity Config - Scrub/Inject Command
0x1_C0E8	IntegInjectCfg	W	INTEGINJECTCFG	Integrity Config - Inject Configuration
0x1_C0F0	IntegScrub	R/W	INTEGSCRUB	Integrity Config - Scrub Timer
0x1_C100	IntegTagOnOff	R/W	INTEGONOFF	Integrity Status - Tag Integrity On/Off
0x1_C104	IntegTagStatus	R/W	INTEGSTATUS	Integrity Status - Tag Integrity Status
0x1_C108	IntegTagCECnt	R/W	INTEGERRCNT	Integrity Status - Tag CE Count
0x1_C10C	IntegTagUECnt	R/W	INTEGERRCNT	Integrity Status - Tag UE Count
0x1_C110	IntegTagCEFFA	R	INTEGFIRSTFAIL	Integrity Status - Tag CE First Failing Address
0x1_C118	IntegTagUEFFA	R	INTEGFIRSTFAIL	Integrity Status - Tag UE First Failing Address
0x1_C120	IntegDataOnOff	R/W	INTEGONOFF	Integrity Status - Data Integrity On/Off
0x1_C124	IntegDataStatus	R/W	INTEGSTATUS	Integrity Status - Data Integrity Status
0x1_C12C	IntegDataUECnt	R/W	INTEGERRCNT	Integrity Status - Data UE Count

Table 87: **Control Register Address Map** *(cont'd)*

Offset	Register Name	Access	Format	Description
0x1_C138	IntegDataFFA	R	INTEGFIRSTFAIL	Integrity Status - Data UE First Failing Address
0x1_C140	IntegATCOnOff	R/W	INTEGONOFF	Integrity Status - ATC Integrity On/Off
0x1_C144	IntegATCStatus	R/W	INTEGSTATUS	Integrity Status - ATC Integrity Status
0x1_C148	IntegATCCECnt	R/W	INTEGERRCNT	Integrity Status - ATC CE Count
0x1_C150	IntegATCFFA	R	INTEGFIRSTFAIL	Integrity Status - ATC CE First Failing Address

COMMAND Register

The Command register is primarily used for full cache events and some global setting. It provide the full cache manipulation functions, such as FCMakeInvalid to make sure the full cache is made invalid. When FCMakeInvalid or FCCleanInvalid is commanded and the corresponding configuration bit MakeInvalid or CleanInvalid is set, the external signal Initializing is asserted during the operation.

For the soft reset feature all bits should be set at the same time even if there are separate ones for major modules.

***Note*:** When using commands to initialize the full cache, all traffic on the external interfaces should be quiescent.

Figure 73: **COMMAND Register**

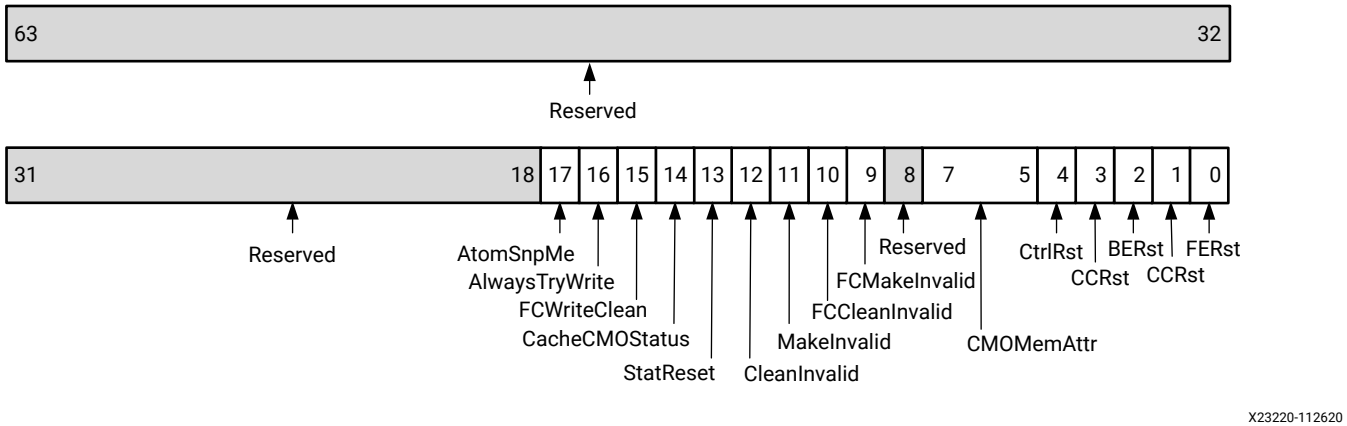


Table 88: **COMMAND Register Bit Definitions**

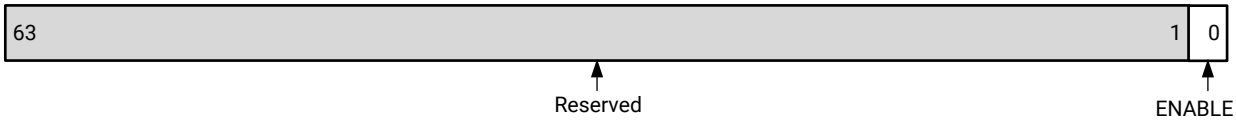
Bits	Name	Reset Value	Access	Description
63:18				Reserved
17	AtomSnpMe	-	W	Atomic SnpMe Attribute is set regardless of transaction and state evaluation. CCIX/CHI specific.

Table 88: **COMMAND Register Bit Definitions** *(cont'd)*

Bits	Name	Reset Value	Access	Description
16	AlwaysTryWrite	-	W	AlwaysTryWrite, always try to push data downstream to the memory (try using WriteEvictFull). CCIX specific.
15	FCWriteClean	-	W	Full Cache WriteClean class (remove dirty data using WriteCleanFull). Dirty lines always goes to SC state. CCIX/CHI specific.
14	CacheCMOStatus	-	W	Cache Maintenance Operations or initalization status (set while initialization or full cache is being invalidated)
13	StatReset	-	W	Statistics Reset
12	CleanInvalid	0	R/W	Soft CleanInvalid seen as initialization. CCIX/CHI specific.
11	MakeInvalid	0	R/W	Soft MakeInvalid seen as initialization. CCIX/CHI specific.
10	FCCleanInvalid	-	W	Full Cache CleanInvalid class, dirty data is written. CCIX/CHI specific.
9	FCMakeInvalid	-	W	Full Cache MakeInvalid class. CCIX/CHI specific.
8				Reserved
7:5	CMOMemAttr	-	W	Cache Maintenance Operations (CMO) Memory Attributes. CCIX specific.
4	CtrlRst	-	W	Soft Control Register Reset
3	ATSRst	-	W	Soft ATS Reset
2	BERst	-	W	Soft Back End (BE) Reset
1	CCRst	-	W	Soft Cache Core (CC) Reset
0	FERst	-	W	Soft Front End (FE) Reset

Statistics Enable Register, STATEN

Figure 74: **Statistics Enable Register, STATEN**



X20783-091619

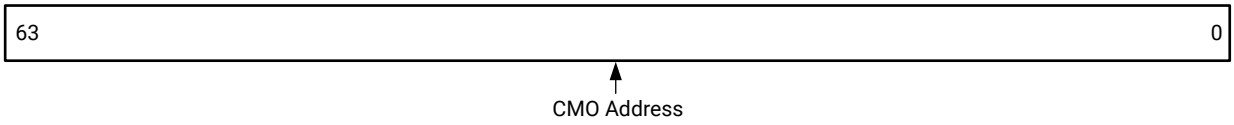
Table 89: **Statistics Enable Register, STATEN, Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:1				Reserved
0	Enable	-	W	Statistics Enable

CMO -Address Register, CMOADDR

Register used for Cache Maintenance Commands. Before a CMO operation is issued, the traffic should be quiesced to avoid conflicts, thus the core CCIX/CHI traffic will be in idle when the CMO operation is performed.

Figure 75: **CMO - Address Register, CMOADDR**



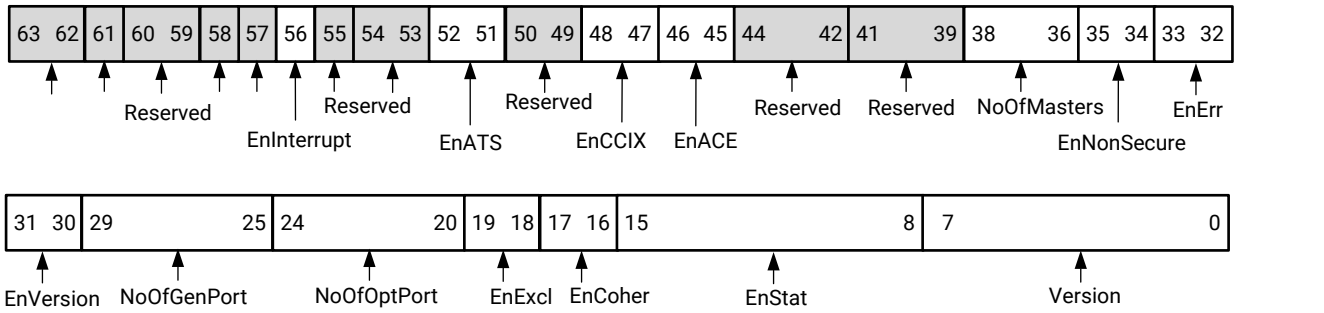
X20784-091619

Table 90: **CMO Address Register, CMOADDR, Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:0		-	W	Cache Maintenance Operations (CMO) Address

Version Register 0, VERSION0

Figure 76: **Version Register 0, VERSION0**



X20786-091719

Table 91: **Version Register 0, VERSION0, Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:62				Reserved
61				Reserved
60:59				Reserved
58				Reserved
57				Reserved
56	EnInterrupt	C_ENABLE_INTERRUPT	R	Enable Interrupt
55				Reserved
54:53				Reserved
52:51	EnATS	C_ENABLE_ADDRESS_TRANSLATION	R	Enable Address Translation

Table 91: **Version Register 0, VERSION0, Bit Definitions** *(cont'd)*

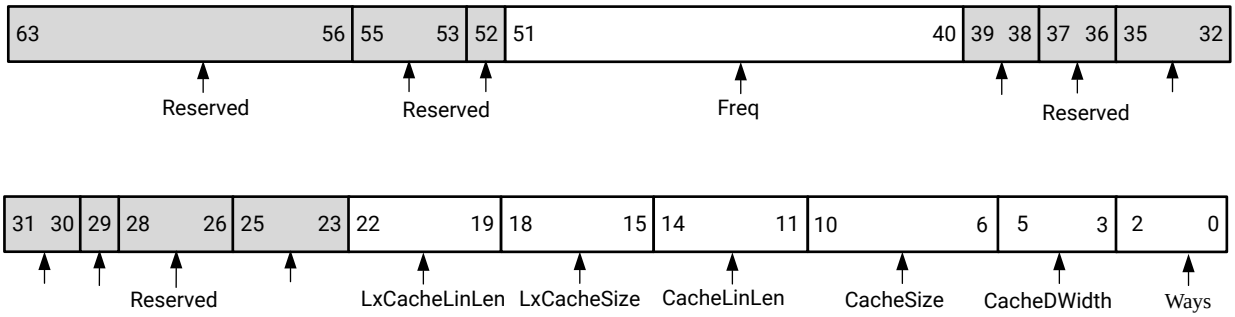
Bits	Name	Reset Value	Access	Description
50:49				Reserved
48:47	EnCCIX	C_ENABLE_CCIX_PROTOCOL	R	Enable CCIX protocol
46:45	EnACE	C_ENABLE_ACE_PROTOCOL	R	Enable ACE protocol
44:42				Reserved
41:39				Reserved
38:36	NoOfMasters	C_NUM_MASTER_PORTS	R	Number of Masters
35:34	EnNonSecure	C_ENABLE_NON_SECURE	R	Enable Security Handling
33:32	EnErr	C_ENABLE_ERROR_HANDLING	R	Enable Error Handling
31:30	EnVersion	C_ENABLE_VERSION_REGISTER	R	Version registers available: 0 - Basic register set, only this register 1 - Full register set, this and register1, see Version Register 1 Bit Field Definition. 2 and 3 are reserved
29:25	NoOfGenPort	C_NUM_GENERIC_PORTS	R	Generic Number of generic port implemented: 0 - All disabled 1 to 16 - Number of ports implemented 17 to 31 are reserved
24:20	NoOfOptPort	C_NUM_OPTIMIZED_PORTS	R	Number of optimized port implemented: 0 - All disabled 1 to 16- Number of ports implemented 17 to 31 are reserved
19:18	InExcl	C_ENABLE_EXCLUSIVE	R	Internal Exclusive monitor implementation: 0 - Disabled 1 - Enabled 2 to 3 are reserved
17:16	EnCoher	C_ENABLE_COHERENCY	R	Cache coherency implementation: 0 - Disabled 1 - Optimized port cache coherency 2 - Master port cache coherency 3 - reserved
15:8	EnStat	C_ENABLE_STATISTICS	R	Enabled statistics block, binary encoded with multiple selected simultaneously: xxxx_xxx1 - Optimized ports xxxx_xx1x - Generic port xxxx_x1xx - Arbiter xxxx_1xxx - Access xxx1_xxxx - Lookup xx1x_xxxx - Update x1xx_xxxx - Backend 1xxx_xxxx - ATC

Table 91: **Version Register 0, VERSION0, Bit Definitions** *(cont'd)*

Bits	Name	Reset Value	Access	Description
7:0	Version	15	R	Core Version: 0 - System Cache version 2.00a 1 - System Cache version 3.0 2 - System Cache version 2.00b 3 - System Cache version 3.1 4 - System Cache version 4.0 5-7 - Reserved 8 - System Cache version 5.0 9 - System Cache version 5.0.1 10 - System Cache version 5.0.2 11 - System Cache version 5.0.3 12 - System Cache version 5.0.4 13 - System Cache version 5.0.5 14 - System Cache version 5.0.6 15 - System Cache version 5.0.7 16-255 Reserved

Version Register 1, VERSION1

Figure 77: **Version Register 1, VERSION1**



X20787-061520

Table 92: **Version Register 1, VERSION1, Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:56				Reserved
55:53				Reserved
52				Reserved
51:40	Freq	C_FREQ/1000000	R	Frequency in MHz
39:38				Reserved
37:36				Reserved
35:30				Reserved
29				Reserved

Table 92: **Version Register 1, VERSION1, Bit Definitions** *(cont'd)*

Bits	Name	Reset Value	Access	Description
28:26				Reserved
25:23				Reserved
22:19	LxCacheLineLen	Log2(C_Lx_CACHE_LINE_LENGTH/4)	R	Cache line length of connected masters on the optimized port, see Cache Line Length Definitions below.
18:15	LxCacheSize	Log2(C_Lx_CACHE_SIZE/64)	R	Cache size of connected masters on the optimized port, see Cache Size below.
14:11	CacheLineLen	Log2(C_CACHE_LINE_LENGTH/4)	R	Internal cache line length, see Cache Line Length Definitions below.
10:6	CacheSize	Log2(C_CACHE_SIZE/64)	R	Internal cache size, see Cache Size below.
5:3	CacheDWidth	Log2(C_CACHE_DATA_WIDTH/2)	R	Internal cache data width: 0 - 8-bit data interface and path 1 - 16-bit data interface and path 2 - 32-bit data interface and path 3 - 64-bit data interface and path 4 - 128-bit data interface and path 5 - 256-bit data interface and path 6 - 512-bit data interface and path 7 - 1024-bit data interface and path
2:0	Ways	Log2(C_NUM_WAYS/2)	R	Number of associative sets: 0 = 2-way set associative 1 = 4-way set associative 2 to 7 = Reserved

Table 93: **Cache Size**

Value	Description	Value	Description
0	64 byte cache size	9	32K byte cache size
1	128 byte cache size	10	64K byte cache size
2	256 byte cache size	11	128K byte cache size
3	512 byte cache size	12	256K byte cache size
4	1K byte cache size	13	512K byte cache size
5	2K byte cache size	14	1M byte cache size
6	4K byte cache size	15	2M byte cache size
7	8K byte cache size	16	4M byte cache size
8	16K byte cache size	17 - 31	Reserved

Table 94: **Cache Line Length Definitions**

Value	Description	Value	Description
0	4 word cache line	5	128 word cache line
1	8 word cache line	6	256 word cache line
2	16 word cache line	7	512 word cache line

Table 94: **Cache Line Length Definitions** *(cont'd)*

Value	Description	Value	Description
3	32 word cache line	8	1024 word cache line
4	64 word cache line	9 - 15	Reserved

DVM Data Register, DVMDATA

Distributed Virtual Memory (DVM) messages are only available with Master ACE Coherency. Any arbitrary DVM message, single or two part, can be insert with the DVM register functionality. The format is described in the AMBA AXI and ACE Protocol Specification in the "Distributed Virtual Memory Transactions" chapter.

The insertion is triggered by writing to DVMNonSecFirst or DVMSecFirst, i.e. for a two part message it is required that the second part is written before the first to make sure it is available when message generation is triggered by writing the first part.

Table 95: **DVM Data Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
n-1:0	"Various"		W	Encoding depends on DVM message, see AMBA AXI and ACE Protocol Specification

IRQ Status Register

Figure 78: **IRQ Status Register**

Table 96: **IRQ Status Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:60				Reserved
59:56			R/W	SAM S3 to S0 interrupt status. Write 1 to acknowledge an interrupt. 56 - S0, SAM no match 59:57 - S3 to S1, Reserved
55:51				Reserved

Table 96: **IRQ Status Register Bit Definitions** *(cont'd)*

Bits	Name	Reset Value	Access	Description
50:32	Backend	0	R/W	Backend B18 to B0 interrupt status. Write 1 to acknowledge an interrupt. 37:32 - B5 to B0, Reserved 38 - B6, Comp/CompData with DatErr/NonDatErr (HA response tagged with Data or Non-Data Error) 50:39 - B18 to B7, Reserved
31:23				Reserved
22:16	Core	0	R/W	Core C6 to C0 interrupt status. Write 1 to acknowledge an interrupt. 16 - C0, Tag CE 17 - C1, Tag UE 18 - C2, Data Miss Clean/Dirty CE 19 - C3, Data Hit Clean/Dirty CE 20 - C4, Data Miss Clean/Dirty UE 21 - C5, Data Hit Clean UE 22 - C6, Data Hit Dirty UE
15				Reserved
14:8	Translation	0	R/W	Translation T6 - T0 interrupt status. Write 1 to acknowledge an interrupt. 13:8 - T5 to T0, Reserved 14 - T6, ATS Table Address, Page, PASID CE
7:1				Reserved
0	Frontend	0	R/W	Frontend F0 interrupt status. Write 1 to acknowledge an interrupt. 0 - F0, Reserved

IRQ Enable Register

Figure 79: **IRQ Enable Register**

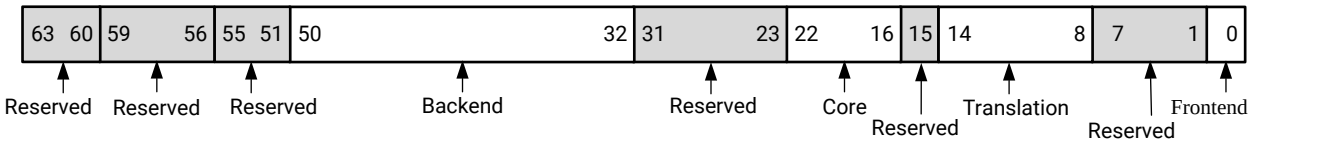


Table 97: **IRQ Enable Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:0	IRQ Enable	0	R/W	Enable interrupts. All bit positions are mapped to the corresponding positions in the IRQ Status Register.

IRQ Pending Register

Figure 80: **IRQ Pending Register**

Table 98: **IRQ Pending Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:0	IRQ Pending		R	Pending interrupts. All bit positions are mapped to the corresponding positions in the IRQ Status Register.

Integrity Config Scrub/Inject Command Register, INTEGCMD

Figure 81: **Integrity Config Scrub/Inject Command Register, INTEGCMD**

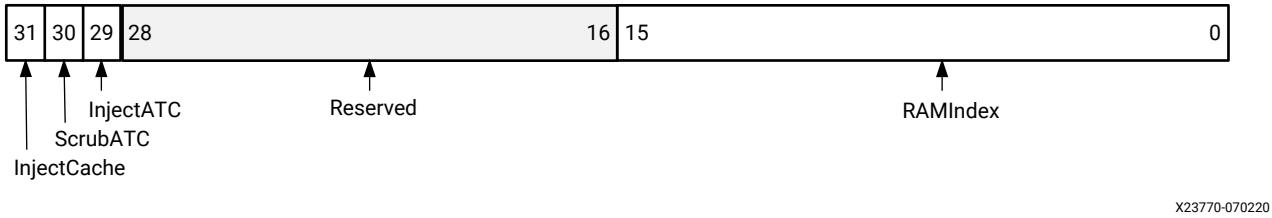


Table 99: **Integrity Config Scrub/Inject Command Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
31	InjectCache	0	R/W	Inject Pending Cache Fault pattern
30	ScrubATC	0	R/W	Scrub ATC
29	InjectATC	0	R/W	Inject Pending ATC Fault
28:16				Reserved
15:0	RAMIndex	0	R/W	Scrub/Inject RAM Index

Integrity Config Inject Configuration Register, INTEGINJECTCFG

Figure 82: **Integrity Config Inject Configuration Register, INTEGINJECTCFG**

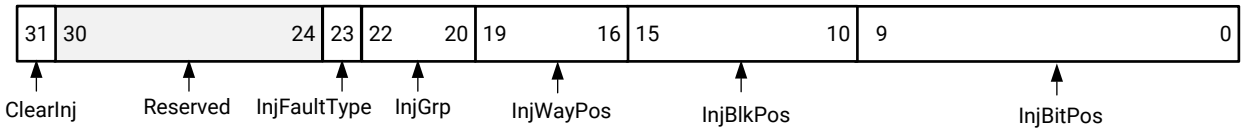


Table 100: **Integrity Config Inject Configuration Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
31	ClearInj	-	W	Clear Inject Fault pattern
30:24				Reserved
23	InjFaultType	-	W	Inject Fault Type: 0 - Information bit (actual data or tag contents) 1 - Integrity (actual extra bits for ECC or parity)
22:20	InjGrp	-	W	Inject Group 0 - Tag 1 - Data 2 - ATC 3-7 - Reserved
19:16	InjWayPos	-	W	Inject Way Position (number of Way in Set)
15:10	InjBlkPos	-	W	Inject Block Position (block position within cache line)
9:0	InjBitPos	-	W	Inject Bit Position (bit position within a block)

Integrity Config Scrub Timer Register, INTEGSCRUB

Figure 83: **Integrity Config Scrub Timer Register, INTEGSCRUB**

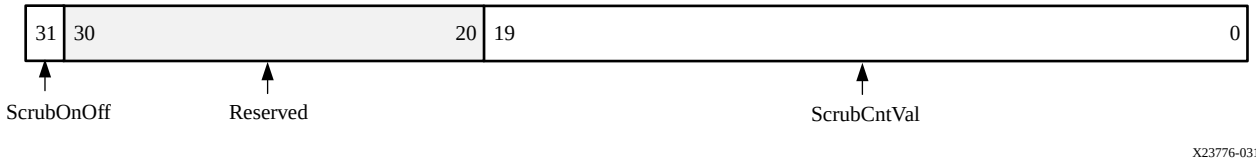


Table 101: **Integrity Config Scrub Timer Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
31	ScrubOnOff	1	R/W	Scrubbing: 0 - Off 1 - On
30:20				Reserved
19:0	ScrubCntVal	0XFFFFF	R/W	Scrubbing Event Counter Value (Number of clock cycles until scrubbing next RAM Index)

Integrity On/Off Register, INTEGONOFF

Figure 84: **Integrity On/Off Register, INTEGONOFF**

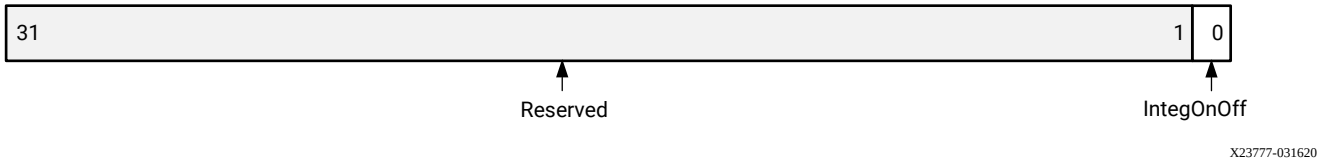
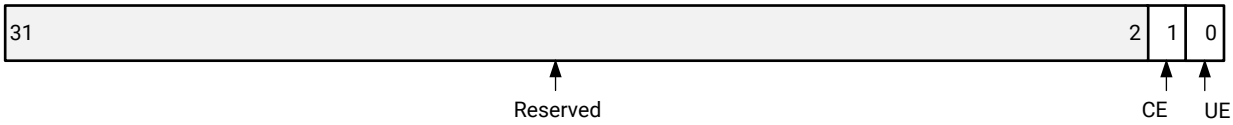


Table 102: **Integrity On/Off Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:1				Reserved
0	IntegOnOff	1	R/W	Integrity: 0 - Off 1 - On

Integrity Status Register, INTEGSTATUS

Figure 85: **Integrity Status Register, INTEGSTATUS**



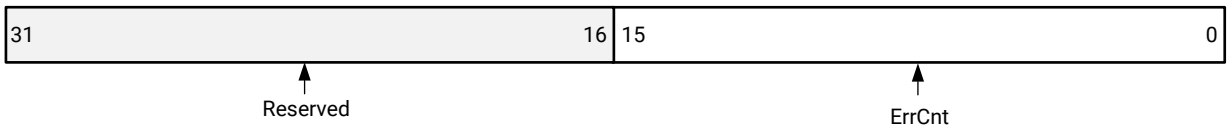
X23778-031620

Table 103: **Integrity Status Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
31:2				Reserved
1	CE	0	R/W	Correctable Error Status
0	UE	0	R/W	Uncorrectable Error Status

Integrity Error Count Register, INTEGERRCNT

Figure 86: **Integrity Error Count Register, INTEGERRCNT**



X23779-031620

Table 104: **Integrity Error Count Register Bit Definitions**

Bits	Name	Reset Value	Access	Description:
31:16				Reserved
15:0	ErrCnt	0	R/W	Error Count, either correctable or uncorrectable depending on type

Integrity First Fail Register, INTEGFIRSTFAIL

Figure 87: **Integrity First Fail Register, INTEGFIRSTFAIL**

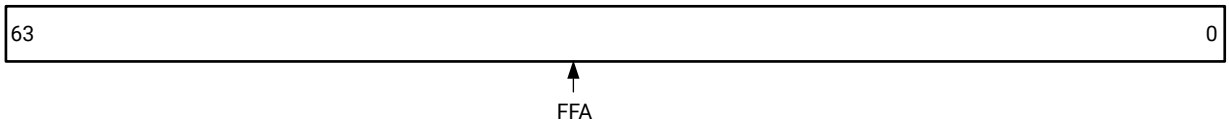


Table 105: **Integrity First Fail Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
63:0	FFA	0		First Failing Address, physical address or ATC table entry

READ Latency

Table 106: **Read Latency Measurement Mode Definitions**

Value	Description
0	AR channel valid until first data is acknowledged.
1	AR channel acknowledged until first data is acknowledged.
2	AR channel valid until last data is acknowledged.
3	AR channel acknowledged until last data is acknowledged.

WRITE Latency

Table 107: **Write Latency Measurement Mode Definitions**

Value	Description
0	AW channel valid until first data is written.
1	AW channel acknowledged until first data is written.
2	AW channel valid until last data is written.
3	AW channel acknowledged until last data is written.
4	AW channel valid until BRESP is acknowledged.
5	AW channel acknowledged until BRESP is acknowledged.

Statistics Enable Configuration

Table 108: **Statistics Enable Configuration**

Value	Description
0	Statistics collection disabled.
1	Statistics collection enabled.

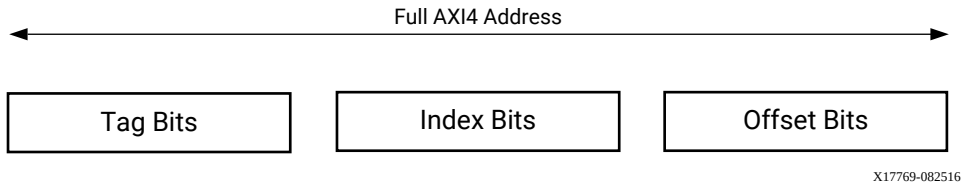
Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the core. The descriptions in this chapter related to AXI4 and ACE are MicroBlaze™ processor-centric, but the end result can usually be achieved in a similar way by any master that can behave like a MicroBlaze processor from a bus transaction point of view.

System Cache Design

The System Cache core is a write-back cache with configurable size and set associativity. The configuration determines how the address range for the System Cache core is divided and used, see the following figure.

Figure 88: **Address Bit Usage**



Cache Lines

The cache size is divided into cache lines, the number of cache lines is determined by cache size (`C_CACHE_SIZE`) divided by line length in bytes (`4 * C_CACHE_LINE_LENGTH`). In the System Cache core storage, data inside a cache line is accessed by the offset bits of an address.

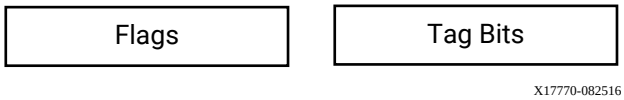
Sets

Several cache lines, determined by `C_NUM_WAYS`, are grouped to form a set. Each set is accessed by the index bits part of the address, see the previous figure. An address can be mapped to any of the cache lines, also called ways, in a set. The way that is used when allocating a cache line is firstly any free way in a set, secondly already allocated lines replaced by a Least Recently Used (LRU) algorithm.

Tags

A cache line tag consists of the tag bits part of the address and flag bits that determine the status of the cache line, for example if it is valid, unique or dirty. See the following figure for tag contents.

Figure 89: **Cache Line Tag Contents**



Transaction Properties

Each slave port has ARCACHE bits for the read channel, and AWCACHE bits for the write channel that determine how to handle a transaction. In a Master Coherent configuration AXI transactions are also converted to ACE/CCIX transactions according to ARCACHE and AWCACHE, as well as the current cache line state.

AXI Memory Properties

The following tables show the ARCACHE and AWCACHE bus contents for Read and Write channels respectively, according to the AMBA® AXI and ACE Protocol Specification.

Table 109: **ARCACHE Bit Field**

Other Allocate	Read Allocate	Modifiable	Bufferable
3	2	1	0

Table 110: **AWCACHE Bit Field**

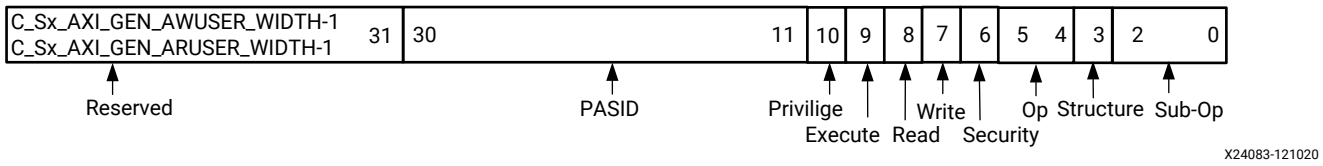
Write Allocate	Other Allocate	Modifiable	Bufferable
3	2	1	0

AXI User Sideband Properties

To control advanced transaction behavior the AXI User bits ARUSER and AWUSER are utilized to define a sideband protocol. The format is the same for both ARUSER and AWUSER but differs in available transactions.

The format define several fields as shown in following figure and table. Reserved bits are intended for future extensions.

Figure 90: **AxUSER Bit Field Definition**



Op defines the group of operations that are used, Structure selects any structure/layout settings and Sub-op defines a more precise operation, if available.

Currently this functionality is used to provide support for Atomic operations for CCIX and CHI coherency.

The ATS Page defines the group of hints and overrides sideband signals contain page setting information to be used when requesting translations.

The optional PASID sub-field must be applied when PASID is enabled, to allow unique translation for unpropagated Host TA translation requests.

Atomics

The atomic operations are divided into four groups: AtomicStore, AtomicLoad, AtomicSwap and AtomicCompare. All atomic transactions except AtomicStore use both the AXI AW and AR channels.

There are a number of general rules that must be followed:

- Atomic operations that use both channels must have the same setting on both channels
- Multiple simultaneous operations must be issued in the same order on both channels

Operation 0 is used for normal operation, while the remainder are used for various Atomic operations.

The operations are defined in the following table.

Table 111: **Atomics Sideband Operations**

Op [5:4]	Type	Description
00	Normal	Normal transaction with translation hints in Sub-Op and Structure bit fields
01	AtomicStore	Operation has AtomicStore category with Sub-Op defining exactly how the data is handled. Structure bit defines Endianness.
10	AtomicLoad	Operation has AtomicLoad category with Sub-Op defining exactly how the data is handled. Structure bit defines Endianness.
11	AtomicSwap	Operation has AtomicSwap category, and the distinction from AtomicCompare is defined by Sub-Op
11	AtomicCompare	Operation has AtomicCompare category, and the distinction from AtomicSwap is defined by Sub-Op

If cache line is allocated when not expected the SnpMe attribute of the Atomics operation will be set.

AtomicLoad and AtomicStore both have the same set of eight sub-operations, with the only difference that AtomicLoad returns the value prior to the atomic operation.

Table 112: **AtomicLoad and AtomicStore Sub-Operation Definition**

Sub-op [2:0]	Type	Description
000	ADD	Add
001	CLR	Bitwise clear
010	EOR	Bitwise Exclusive OR
011	SET	Bitwise set
100	SMAX	Signed Max
101	SMIN	Signed Min
110	UMAX	Unsigned Max
111	UMIN	Unsigned Min

The Sub-Op Add, SMax, SMin, UMax and UMin also use the structure bit to define Endianness.

Table 113: **AtomicLoad and AtomicStore Structure Definition**

Structure [3]	Type	Description
0	Little	Little Endian
1	Big	Big Endian

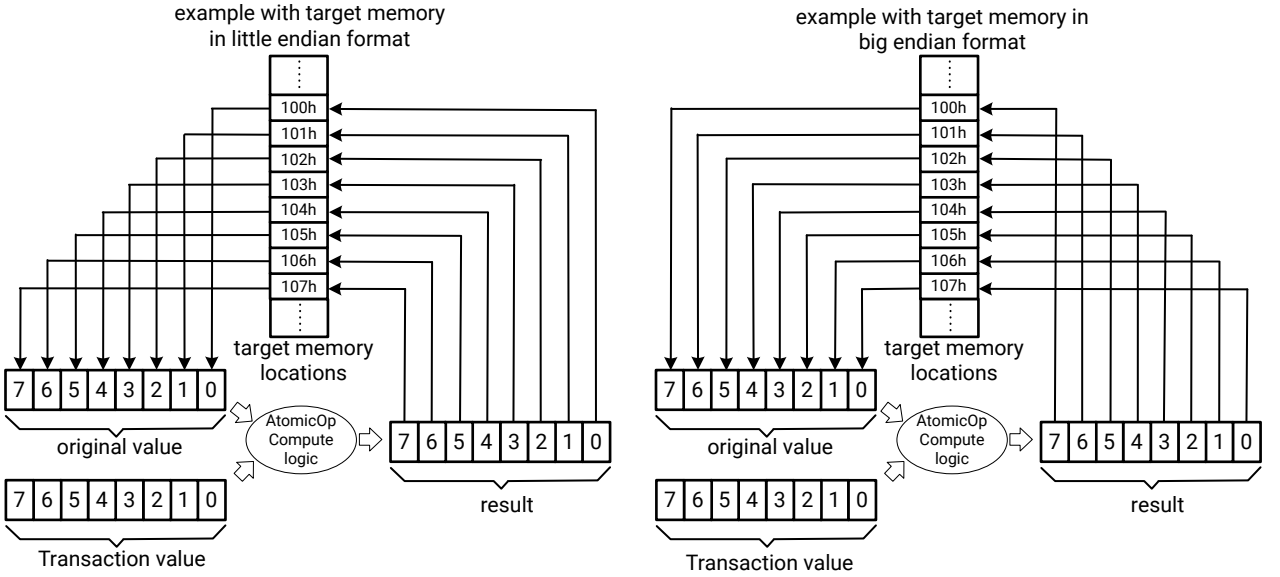
AtomicSwap and AtomicCompare both have the same Op value but use different Sub-Op values

Table 114: **AtomicSwap and AtomicCompare Sub-operation Definition**

Sub-op [2:0]	Type	Description
000	Swap	AtomicSwap
001	Compare	AtomicCompare

The Endianness bit selects how the individual bytes are handled in arithmetic operations:

Figure 91: **Atomic Endianess Mapping**



X24123-061220

The coherency protocol imposes a number of rules on the length of the atomic operations.

Table 115: **Allowed Atomic Lengths**

Operation	Lengths
AtomicStore	1, 2, 4, or 8 bytes
AtomicLoad	1, 2, 4, or 8 bytes
AtomicSwap	1, 2, 4, or 8 bytes
AtomicCompare	2, 4, 8, 16, or 32 bytes. Returned value is half length.

AtomicCompare has both a Compare and a Swap value, which makes it twice as large as the others.

There is a lot of flexibility in creating the atomic operations from AXI, although some rules must be followed:

- AtomicLoad, AtomicStore, and AtomicSwap
  - Incr and Wrap can be used to build the transaction, with the total length of the operation determined by AxSIZE (number of beats) * AxLEN (beat size):
    - Incr must be aligned to the total length
    - Wrap must be aligned to AxSIZE (be aware that the data used for the operation is the unwrapped data)

- Both AR and AW must have the same settings, excluding AtomicStore which lacks the AR phase

- AtomicCompare (CAS)
  - Incr and Wrap can be used, with the total length of the operation determined by AxSIZE (number of beats) * AxLEN (beat size):
    - AR must have half the length of AW
    - Incr must be aligned to the total length of AW
    - Wrap must be aligned to the total length of AR
  - Both AR and AW must use the same address

A common method for AtomicCompare is to use a Wrap transaction with the same size (AxSIZE) on both AR and AW, and just use half the length (ARLEN) for AR.

ATS Page Information

The sideband signals containing Address translation page setting information to be used when requesting Gen/Opt Port transaction.

Table 116: **ATS Page Properties Definition**

Page Related Bit field [30:6]	Type	Description
[6]	Security	Page Security setting: 0 - Secure 1 - NonSecure
[7]	Write	Page Write permission: 0 - Write prohibited 1 - Write permitted
[8]	Read	Page Read permission: 0 - Read prohibited 1 - Read permitted
[9]	Execute	Page Execute permission: (Valid with CHI PASID enabled) 0 - Execute prohibited 1 - Execute permitted
[10]	Privilege	Page Permission setting: (Valid with CHI PASID enabled) 0 - Normal access 1 - Privilege access
[30:11]	PASID	Process Address Space ID (Valid, propagates to Host TA with CHI PASID enabled, used as extended Translation ID with CCIX PASID enabled)

Property Override

To achieve the expected transaction behavior all Optimized and Generic slave ports have parameters to override the original transaction settings for all the Allocate bits and the Bufferable bit. They can be used to set the properties for AXI Masters that have non-programmable ARCACHE and/or AWCACHE configurations that do not match the desired transaction properties.

Parameters such as `C_Sx_AXI_[GEN_]FORCE_[READ|WRITE]_[ALLOCATE|BUFFER]` set the corresponding bit to 1, whereas parameters such as `C_Sx_AXI_[GEN_]PROHIBIT_[READ|WRITE]_[ALLOCATE|BUFFER]` clear the bit to 0.

The following tables show the relationship between the corresponding parameters and the ARCACHE and AWCACHE bits respectively for an Optimized port. An identical mapping applies to the Generic `S_AXI_GEN` ports. The Modifiable bit is not applicable.

Table 117: **Parameter to ARCACHE Bit Mapping**

Other Allocate	Read Allocate	Bufferable
C_Sx_AXI_FORCE_WRITE_ALLOCATE	C_Sx_AXI_FORCE_READ_ALLOCATE	C_Sx_AXI_FORCE_READ_BUFFER
C_Sx_AXI_PROHIBIT_WRITE_ALLOCATE	C_Sx_AXI_PROHIBIT_READ_ALLOCATE	C_Sx_AXI_PROHIBIT_READ_BUFFER

Table 118: **Parameter to AWCACHE Bit Mapping**

Write Allocate	Other Allocate	Bufferable
C_Sx_AXI_FORCE_WRITE_ALLOCATE	C_Sx_AXI_FORCE_READ_ALLOCATE	C_Sx_AXI_FORCE_WRITE_BUFFER
C_Sx_AXI_PROHIBIT_WRITE_ALLOCATE	C_Sx_AXI_PROHIBIT_READ_ALLOCATE	C_Sx_AXI_PROHIBIT_WRITE_BUFFER

Cache Handling

A read transaction allocates a cache line, unless already allocated, when the read Allocation bit is set for a Write-Back configuration. A cache line remains allocated regardless of the read transaction properties.

Similarly a write transaction allocates a cache line if the Write Allocate bit is set, unless already allocated. A write to an already allocated line remains allocated if at least one of the Allocate bits is set, as well as Bufferable and Modifiable, otherwise the cache line is deallocated. The new data and any dirty data in the cache line is written downstream.

A Write Miss with Allocate enabled always fetches the entire cache line from a downstream cache or memory before merging the new data into the cache line. Additional transactions to this cache line will stall until the merge has completed.

ACE Property Translation

All transactions in the Master Coherent configuration always have ARDOMAIN and AWDOMAIN set to InnerShareable, unless the parameter `C_DEFAULT_DOMAIN` is changed from its default value.

There are three possible sources for transactions on the Master ACE port:

- DVM inserted from the control interface as well as automatic DVM sync responses.
- Inserted Barriers from the control interface.
- Traffic related to cache events directly from transactions or indirectly as a flush from the control interface.

The following table show the transaction settings for DVM type operations.

Table 119: **ARCACHE and ARSNOOP Settings for DVM Transactions**

Transaction	M0_AXI_ARCACHE	M0_AXI_ARSNOOP
DVM Message or DVM Sync	0010	1111
DVM Complete[1]	0010	1110

Notes:
1. Automatic response to DVM Sync

The following tables show the transactions settings for Barrier type operations on read and write channels respectively.

Table 120: **ARCACHE and ARSNOOP Settings for Barrier Transactions**

Transaction	M0_AXI_ARCACHE	M0_AXI_ARSNOOP
Read Barrier	0010	0000

Table 121: **AWCACHE and AWSNOOP Settings for Barrier Transactions**

Transaction	M0_AXI_AWCACHE	M0_AXI_AWSNOOP
Write Barrier	0011	000

Incoming transactions on slave ports, cache maintenance operation on the Ctrl port as well as the current cache line state determines the kind of transactions that will be seen on the Master ACE interface. The following tables show all types of events. Some of the write related events actually appear on the master read channel.

Table 122: **ARCACHE and ARSNOOP Settings for Cache Originating Transactions**

Event	Transaction	M0_AXI_ARCACHE	M0_AXI_ARSNOOP
Read Miss not Allocating	ReadOnce	1111	0000
Read Miss Allocating	ReadShared	1111	0001
Read Hit (any type)	No bus event		
Write Miss Allocating	ReadUnique	1111	0111
Write Hit Shared	CleanUnique	1111	1011

Table 123: **AWCACHE and AWSNOOP Settings for Cache Originating Transactions**

Event	Transaction	M0_AXI_AWCACHE	M0_AXI_AWSNOOP
Write Miss not Allocating	WriteUnique	0011	000
Evicting Dirty Line[1]	WriteBack	0011	011
Write Hit Unique	No bus event		

Notes:

1. Reusing line or Flushing

Complete ACE Transaction Flow

The following tables show the complete transaction flow and conversion from the point the transaction arrives at a Generic or Optimized port where the ARCACHE and AWCACHE values in the tables are the end result of any manipulation by the FORCE or PROHIBIT parameters.

Table 124: **Complete Slave ARCACHE to Cache Event and Potential Master Transaction Mapping**

Slave ARCACHE	Cache Event	Cache Action	Master ACE Transaction	
00xx	Read Miss	Bypass Cache	ReadOnce	ARCACHE = 1111
	Read Hit	Use cached line	N/A	N/A
x100 x110	Read Miss	Bypass cache	ReadOnce	ARCACHE = 1111
	Read Hit	Use cached line	N/A	N/A
x101 x111	Read Miss	Allocate cache line and forward data	ReadShared	ARCACHE = 1111
	Read Hit	Use cached line	N/A	N/A
10xx	Read Miss	Bypass cache	ReadOnce	ARCACHE = 1111
	Read Hit	Use cached line	N/A	N/A

Table 125: **Complete Slave AWCACHE to Cache Event and Potential Master Transaction Mapping**

Slave AWCACHE	Cache Event	Cache Action	Master ACE Transaction	
00xx	Write Miss	Bypass cache	WriteUnique	AWCACHE = 0011
	Write Hit Shared	Request write permission, evict dirty cache line after write	CleanUnique WriteBack	ARCACHE = 1111 AWCACHE = 0011
	Write Hit Unique	Evict dirty cache line after write	WriteBack	AWCACHE = 0011
0100 0101 0110	Write Miss	Bypass cache	WriteUnique	AWCACHE = 0011
	Write Hit Shared	Request write permission, evict dirty cache line after write	CleanUnique WriteBack	ARCACHE = 1111 AWCACHE = 0011
	Write Hit Unique	Evict dirty cache line after write	WriteBack	AWCACHE = 0011
0111	Write Miss	Bypass cache	WriteUnique	AWCACHE = 0011
	Write Hit Shared	Request write permission	CleanUnique	ARCACHE = 1111
	Write Hit Unique	Update cache line	N/A	N/A
1x00 1x01 1x10	Write Miss	Bypass cache	WriteUnique	AWCACHE = 0011
	Write Hit Shared	Request write permission, evict dirty cache line after write	CleanUnique WriteBack	ARCACHE = 1111 AWCACHE = 0011
	Write Hit Unique	Evict dirty cache line after write	WriteBack	AWCACHE = 0011
1x11	Write Miss	Allocate cache line	ReadUnique	ARCACHE = 1111
	Write Hit Shared	Request write permission	CleanUnique	ARCACHE = 1111
	Write Hit Unique	Update cache line	N/A	N/A

CCIX Property Translation

Incoming transactions on slave ports, cache maintenance operation via the control port, as well as the current cache line state determine the kind of transactions output on the CCIX interface, if any at all. The following tables show all types of events. Note that some of the write related events actually appear on the read channel.

Table 126: **Read Request and Memory Type for Cache Originating Transactions**

Event	Request	Memory	Request
Read Miss not Allocating	ReadNoSnp	Device-nRnE Device-nRE Non-Cached	0x00
Read Miss not Allocating (coherent)	ReadOnce	WBnA	0x01
Read Miss Allocating	ReadShared	WBA	0x07
Read Hit (any type)	No bus event		
Write Miss Allocating	ReadUnique	WBnA/WBA	0x04
Write Hit Shared	CleanUnique	WBnA/WBA	0x10

Table 127: **Write Request and Memory Type for Cache Originating Transactions**

Event	Request	Memory	Request
Write Miss not Allocating	WriteNoSnp	Device-nRnE Device-nRE Non-Cached	0x20
Write Miss not Allocating (coherent)	WriteUnique	WBnA	0x22
Evicting Dirty Line[1]	WriteBack	WBA	0x27
Write Hit Unique	No bus event		

Notes:

1. Reusing line or flushing

Complete CCIX Transaction Flow

Like ACE Master Port coherency CCIX relies on AxCACHE and the cache state to determine the resulting events in the CCIX domain.

Table 128: **Complete Slave ARCACHE to Cache Event and Potential CCIX Transaction Mapping**

Slave ARCACHE	Cache Event	Cache Action	CCIX Transaction	
00xx	Read Miss	Bypass Cache	ReadNoSnp	ReqAttr=Device-nRnE ReqAttr=Device-nRE ReqAttr=Non-Cached
	Read Hit	Use cached line	N/A	N/A
x100 x110	Read Miss	Bypass cache	ReadOnce	ReqAttr=WBnA
	Read Hit	Use cached line	N/A	N/A
0101 0111	Read Miss	Allocate cache line and forward data	ReadShared	ReqAttr=WBA
	Read Hit	Use cached line	N/A	N/A

Table 128: **Complete Slave ARCACHE to Cache Event and Potential CCIX Transaction Mapping** *(cont'd)*

Slave ARCACHE	Cache Event	Cache Action	CCIX Transaction	
1101 1111	Read Miss	Allocate cache line and forward data	ReadUnique	ReqAttr=WBA
	Read Hit	Use cached line	N/A	N/A
10xx	Read Miss	Bypass cache	ReadOnce	ReqAttr=WBnA
	Read Hit	Use cached line	N/A	N/A

Table 129: **Complete Slave AWCACHE to Cache Event and Potential CCIX Transaction Mapping**

Slave AWCACHE	Cache Event	Cache Action	CCIX Transaction	
00xx	Write Miss	Bypass cache	WriteUnique	ReqAttr=WBnA
	Write Hit Shared	Request write permission, evict dirty cache line after write	CleanUnique WriteBack	ReqAttr=WBA
	Write Hit Unique	Evict dirty cache line after write	WriteBack	ReqAttr=WBA
0100 0101 0110	Write Miss	Bypass cache	WriteUnique	ReqAttr=WBnA
	Write Hit Shared	Request write permission, evict dirty cache line after write	CleanUnique WriteBack	ReqAttr=WBnA
	Write Hit Unique	Evict dirty cache line after write	WriteBack	ReqAttr=WBnA
0111	Write Miss	Bypass cache	WriteUnique	ReqAttr=WBA
	Write Hit Shared	Request write permission	CleanUnique	ReqAttr=WBnA
	Write Hit Unique	Update cache line	N/A	N/A
1x00 1x01 1x10	Write Miss	Bypass cache	WriteUnique	ReqAttr=WBA
	Write Hit Shared	Request write permission, evict dirty cache line after write	CleanUnique WriteBack	ReqAttr=WBA
	Write Hit Unique	Evict dirty cache line after write	WriteBack	ReqAttr=WBA
1x11	Write Miss	Allocate cache line	ReadUnique	ReqAttr=WBA
	Write Hit Shared	Request write permission	CleanUnique	ReqAttr=WBA
	Write Hit Unique	Update cache line	N/A	N/A

CHI Property Translation

Incoming transactions on slave ports, cache maintenance operation via the control port, as well as the current cache line state determine the kind of transactions output on the CHI interface, if any at all. The following tables show all types of events. Note that some of the write related events actually appear on the read channel.

Table 130: **Read Request and Memory Type for Cache Originating Transactions**

Event	Request	Memory	Request
Read Miss not Allocating	ReadNoSnp	Device nRnE Device nRE Non-Cacheable	0x4
Read Miss not Allocating (coherent)	ReadOnce	Snoopable WriteBack No-Allocate	0x3
Read Miss Allocating	ReadShared	Snoopable WriteBack Allocate	0x1
Read Hit (any type)	No bus event		
Write Miss Allocating	ReadUnique	Snoopable WriteBack No-Allocate/Allocate	0x7
Write Hit Shared	CleanUnique	Snoopable WriteBack No-Allocate/Allocate	0xB

Table 131: **Write Request and Memory Type for Cache Originating Transactions**

Event	Request	Memory	Request
Write Miss not Allocating	WriteNoSnpPtl	Device nRnE Device nRE Non-Cacheable	0x1C
Write Miss not Allocating (coherent)	WriteUniquePtl	Snoopable WriteBack No-Allocate	0x18
Evicting Dirty Line[1]	WriteBackFull	Snoopable WriteBack Allocate	0x1B
Write Hit Unique	No bus event		

Notes:

1. Reusing line or flushing

Complete CHI Transaction Flow

Like ACE Master Port coherency CHI relies on AxCACHE and the cache state to determine the resulting events in the CHI domain.

Table 132: **Complete Slave ARCACHE to Cache Event and Potential CHI Transaction Mapping**

Slave ARCACHE	Cache Event	Cache Action	CHI Transaction	
00xx	Read Miss	Bypass Cache	ReadNoSnp	MemAttr=Device-nRnE MemAttr=Device-nRE MemAttr=Non-Cacheable
	Read Hit	Use cached line	N/A	N/A

Table 132: **Complete Slave ARCACHE to Cache Event and Potential CHI Transaction Mapping** *(cont'd)*

Slave ARCACHE	Cache Event	Cache Action	CHI Transaction	
x100 x110	Read Miss	Bypass cache	ReadOnce	MemAttr=Snoopable WriteBack No-Allocate
	Read Hit	Use cached line	N/A	N/A
0101 0111	Read Miss	Allocate cache line and forward data	ReadShared	MemAttr=Snoopable WriteBack Allocate
	Read Hit	Use cached line	N/A	N/A
1101 1111	Read Miss	Allocate cache line and forward data	ReadUnique	MemAttr=Snoopable WriteBack Allocate
	Read Hit	Use cached line	N/A	N/A
10xx	Read Miss	Bypass cache	ReadOnce	MemAttr=Snoopable WriteBack No-Allocate
	Read Hit	Use cached line	N/A	N/A

Table 133: **Complete Slave AWCACHE to Cache Event and Potential CHI Transaction Mapping**

Slave AWCACHE	Cache Event	Cache Action	CHI Transaction	
00xx	Write Miss	Bypass cache	WriteUniquePtl	MemAttr=Snoopable WriteBack No-Allocate
	Write Hit Shared	Request write permission, evict dirty cache line after write	CleanUnique WriteBackFull	MemAttr=Snoopable WriteBack Allocate
	Write Hit Unique	Evict dirty cache line after write	WriteBackFull	
0100 0101 0110	Write Miss	Bypass cache	WriteUniquePtl	MemAttr=Snoopable WriteBack No-Allocate
	Write Hit Shared	Request write permission, evict dirty cache line after write	CleanUnique WriteBackFull	MemAttr=Snoopable WriteBack No-Allocate
	Write Hit Unique	Evict dirty cache line after write	WriteBackFull	MemAttr=Snoopable WriteBack No-Allocate
0111	Write Miss	Bypass cache	WriteUniquePtl	ReqAttr=WBA
	Write Hit Shared	Request write permission	CleanUnique	MemAttr=Snoopable WriteBack No-Allocate
	Write Hit Unique	Update cache line	N/A	N/A
1x00 1x01 1x10	Write Miss	Bypass cache	WriteUniquePtl	MemAttr=Snoopable WriteBack Allocate
	Write Hit Shared	Request write permission, evict dirty cache line after write	CleanUnique WriteBackFull	MemAttr=Snoopable WriteBack Allocate
	Write Hit Unique	Evict dirty cache line after write	WriteBackFull	MemAttr=Snoopable WriteBack Allocate
1x11	Write Miss	Allocate cache line	ReadUnique	MemAttr=Snoopable WriteBack Allocate
	Write Hit Shared	Request write permission	CleanUnique	MemAttr=Snoopable WriteBack Allocate
	Write Hit Unique	Update cache line	N/A	N/A

Snoop Effect

The effect a snoop transaction has on an allocated cache line varies with the type of snoop as well as the state of the line. The effect can also be controlled for a particular snoop type within the allowed responses. These settings are intended for advanced users.

For each type of snoop transaction it is possible to select whether to keep the line allocated or not, if the transaction allows it. If the line is kept it is also possible to select how to treat Dirty data in some cases, if write responsibility is passed on or if it remains.

These configurations are used to define desired behavior, but it is not always guaranteed that it can be honored.

Table 134: **Snoop and Parameter Relation**

Parameter	CCIX	CHI	ACE
C_SNOOP_KEEP_READ_ONCE	SnpToAny	SnpOnce/SnpOnceFwd	ReadOnce
C_SNOOP_PASS_READ_ONCE	SnpToAny	SnpOnce/SnpOnceFwd	ReadOnce
C_SNOOP_KEEP_READ_SHARED	SnpToS	SnpShared/SnpSharedFwd	ReadShared
C_SNOOP_PASS_READ_SHARED	SnpToS	SnpShared/SnpSharedFwd	ReadShared
C_SNOOP_KEEP_READ_CLEAN	N/A	SnpClean/SnpCleanFwd	ReadClean
C_SNOOP_PASS_READ_CLEAN	N/A	SnpClean/SnpCleanFwd	ReadClean
C_SNOOP_KEEP_READ_NSD	N/A	SnpNotSharedDirty/SnpNotSharedDirtyFwd	ReadNotSharedDirty
C_SNOOP_PASS_READ_NSD	N/A	SnpNotSharedDirty/SnpNotSharedDirtyFwd	ReadNotSharedDirty
C_SNOOP_KEEP_CLEAN_SHARED	SnpToC	SnpCleanShared	CleanShared
C_SNOOP_KEEP_SNPTOSC	SnpToSC	N/A	N/A

General Design Guidelines

The are no golden settings to achieve maximum performance for all cases, as performance is application and system dependent. This section contains general guidelines that should be considered when configuring the System Cache core and other IP cores to improve performance.

AXI4 Data Widths

AXI4 Data widths should match wherever possible. Matching widths results in minimal area overhead and latency for the AXI4 interconnects.

AXI4 Clocking

The System Cache core is fully synchronous. Using the same clock for all the AXI4 ports removes the need for clock conversion blocks and results in minimal area overhead and latency for the AXI4 interconnects.

Frequency and Hit Rate

MicroBlaze Based Designs

The system cache size should be configured to be larger than the connected L1 caches to achieve any improvements. Increasing the system cache size increases hit rates and has a positive effect on performance. The downside of increasing the system cache size is an increased number of FPGA resources being used. Higher set associativity usually increases the hit rate and the application performance.

For maximum performance the MicroBlaze™ processor should be configured to match the target frequency of the System Cache core and the rest of the system. Depending on how high the target frequency is the MicroBlaze processor configuration might need to be tweaked to achieve this goal.

There are two primary alternatives that should be considered first. With MicroBlaze v10.0 and later there is a new frequency-optimized 8-stage pipeline that has a slightly higher inter-process clocks per instruction (CPI) but is free from parameter configuration frequency dependencies. It always matches the frequency of the System Cache core. Note that the longer pipeline also results in increased resource use.

If a MicroBlaze processor with a 5-stage pipeline is used there are a number of factors that can be changed to increase the frequency. These techniques also apply to a 3-stage pipeline when the options are supported. The maximum frequency of the MicroBlaze processor is affected by its cache sizes. Smaller MicroBlaze processor cache sizes usually means that the MicroBlaze processor can meet higher frequency targets, but at the cost of reduced L1 hit rates. The optimum point for the frequency versus cache size trade-off using the System Cache core occurs when the MicroBlaze processor caches are set to either 256 or 512 bytes (dependent on other MicroBlaze configuration settings). For improved frequency, implement the MicroBlaze cache tags with distributed RAM.

Enabling the MicroBlaze branch target cache can improve performance but might reduce the maximum obtainable frequency for 5-stage pipeline (when using an 8-stage pipeline, BTC should always be enabled with maximum configuration, if resources are available). Depending on the rest of the MicroBlaze processor configuration, smaller BTC sizes (for example, 32 entries (C_BRANCH_TARGET_CACHE_SIZE = 3)) could be considered.

MicroBlaze processor advanced cache features can be used to tweak performance but they are only available in non-coherent configurations. Enabling MicroBlaze processor victim caches increases MicroBlaze processor cache hit rates, with improved performance as a result. Enabling victim caches can however reduce the MicroBlaze processor maximum frequency in some cases. Instruction stream cache should be disabled, because it reduces performance when connected to the System Cache core. MicroBlaze processor performance is often improved by using 8-word cache lines on the Instruction Cache and Data Cache.

CCIX Based Designs

In the CCIX case there is no upstream cache that can be configure in relation to System Cache, and accelerators or kernels do not usually offer many parametrization options. However, there are still a number of options that can be tweaked in System Cache, such as size and transaction limits. The CCIX CPM (Versal Premium) use case has the same configuration possibilities as CCIX XDMA, but with an asynchronous interface to the CPM, which enables System Cache to use a single clock domain for CCIX and ATS independent of the PCIe frequency.

When CCIX designs are configured with ATS, the ATS data width must be configured according to the CCIX/PCIe data width, which can be 256 or 512 bits, as well as 1024 bits in Versal Premium devices.

The parameter configuration for maximum achievable performance is unique for all application and system configuration; the optimum settings for one case is not necessarily the same for a different case.

Another property of CCIX based designs is that frequency is fixed to 250 MHz for System Cache. Since System Cache with CCIX is a quite complex and large IP core, used in large designs, it is often necessary to use advanced Vivado implementation strategies in addition to adjusting System Cache parameters to achieve the frequency target. For the most difficult designs it might be necessary to try different strategies to find the one that provides the best result for a particular design, but in many cases it is sufficient to use the strategy Performance_NetDelay_high. Additional improvements can be achieved by enabling Post-Route Phys Opt Design with the directive Explore and advanced Vivado implementation strategies.

See also *UltraFast Design Methodology Timing Closure Quick Reference Guide* (UG1292) and *Vivado Design Suite User Guide: Design Analysis and Closure Techniques* (UG906) for additional information and resources for achieving timing closure.

CHI Based Design

CHI designs have the same configuration possibilities as CCIX, but with an asynchronous interface to the CPM and fixed Data width of 512 bits.

When CHI designs are configured with ATS, the ATS data width must be configured according to the PCIe data width, independent of CHI Data width, and must use the same derived PCIe Clock and frequency. When CHI designs with ATS cannot use the same clock and frequency as PCIe the ATS AXI4-Stream interfaces must use an asynchronous connection. (alternative external ATS Switch with dual clock support) In both configurations ATS interface must be configured with the PCIe data width.

CHI designs are also large and can benefit from advanced Vivado implementation strategies.

Bandwidth

Using wider AXI4 interfaces increases data bandwidth, but also increases FPGA resource usage. Using the widest possible common AXI4 data width between the System Cache AXI4 Master and the external memory gives the highest possible bandwidth. This also applies to the AXI4 connection between MicroBlaze™ processor caches and the System Cache core. The widest possible common width gives the highest bandwidth.

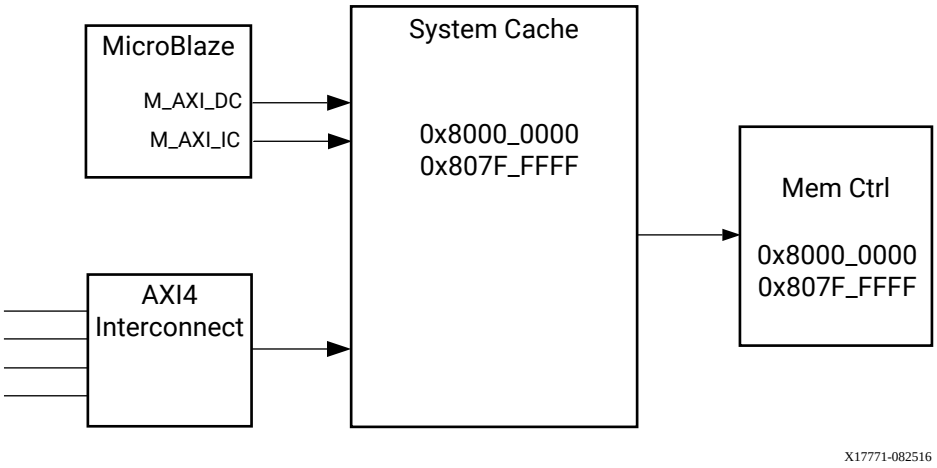
Arbitration

The System Cache core arbitration scheme is round-robin. When the selected port does not have a pending transaction, the first port with an available transaction is scheduled, taking the optimized ports in ascending numeric order and then the generic ports in ascending numeric order.

Address Map

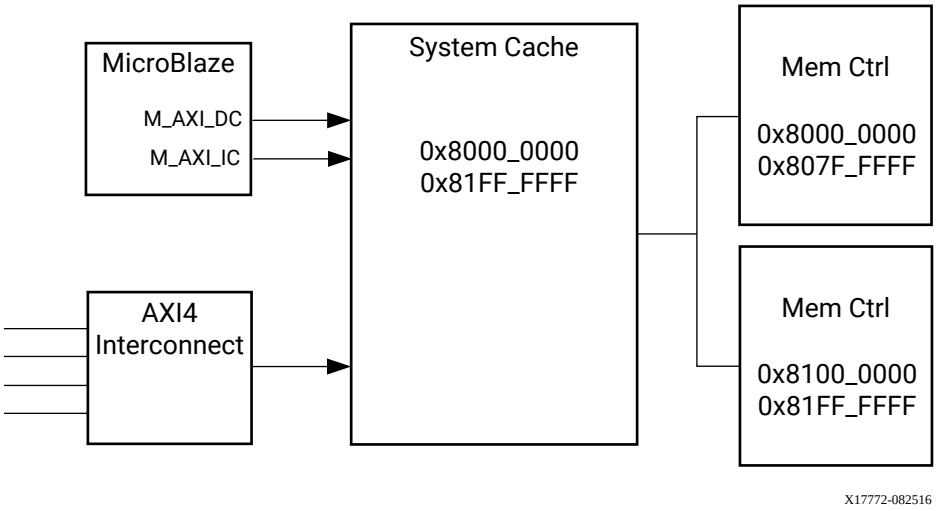
The most common case is when the System Cache core is connected to one memory controller. In this case the System Cache core should have the exact same address range as the memory controller. An example with single memory controller is shown in the following figure.

Figure 92: **Single Memory Controller Configuration**



When the System Cache core is connected to multiple slaves it normally has an address range that spans the complete range of all the slaves. Depending on how many slaves there are and their individual address ranges this could leave holes in the address map, which the software designer must be aware of to avoid program issues. An example with a dual memory controller having a hidden address hole is shown in the following figure.

Figure 93: **Dual Memory Controller Configuration with Hole**



CCIX System Address Map

The CCIX System Address Map (SAM) is responsible for translating an address to a Target ID (TgtID) in the coherent domain. All available registers for configuration are outlined in the register section. Up to 32 SAM entries are supported, but usually much fewer than that are used.

If an address is provided that is not covered by a SAM entry, the transaction will end with Decode Error on AXI.

SAM entries are either configured in Range or Mask mode:

- **Range:** Base register is the lower address and Mask register is the upper address of the defined address range
- **Mask:** Mask register defines which address bits are compared with the corresponding bits in the Base register (intended for advanced users, not used by firmware)

Typically all Request Agents are assigned an ID during the PCIe/CCIX negotiations via the firmware, so the SAM requires no configuration.

It is possible for specify default values used after reset for the first four entries, using the parameters listed in the following table:

Table 135: **SAM Parameter Mapping**

Parameter[1]	Description
C_SAMx_VALID	Valid entry
C_SAMx_LOCAL	Reserved for future use, set to 0
C_SAMx_ID	Agent ID of Home Agent
C_SAMx_LINK	Reserved for future use, set to 0
C_SAMx_BASEADDR	Lower address of range / Base for mask
C_SAMx_HIGHADDR	Upper address of range / Mask

Notes:
1. x = 0 - 3

Note: CHI systems also have SAM for address to ID translation but in this case it is located in the CPM instead of System Cache.

Optimized Port Cache Coherency

Enable optimized port cache coherency when a MicroBlaze™ multi-processor system is expected to work on the same data, and hardware support for cache coherence is desired for efficiency and safety. This handles all coherency support that is required for branch target cache and MMU as well as instruction and data caches. Without the hardware cache coherency support all this has to be handled through software.

ACE Master Port Cache Coherency

Enabling master port cache coherency with accelerators in the Zynq® UltraScale+™ MPSoC PL improves memory management by providing a local cache that is hardware cache coherent to the APU caches. With a correctly sized cache the majority of the accelerator AXI traffic can be terminated in the System Cache core to reduce the impact on the PS internal bandwidth. Snoop traffic generated from the PS is also handled efficiently by the System Cache core.

CCIX Master Port Cache Coherency

On Virtex UltraScale+ HBM devices CCIX is available in System Cache to create a coherent cache for accelerators. This allows transparent data sharing between the accelerators, a server host and external interface attached memory.

On Versal (Premium) devices CCIX is available in System Cache to create a coherent cache for accelerators by connecting to the CPM. Coherency either remains local or the CPM is configured to use CCIX, which allows transparent data sharing between the node and agents.

The system solution is more complex than ACE, but firmware handles all the negotiations and configuration to create a seamless experience from the accelerator point of view. When all firmware negotiations are complete the CCIX link will be activated and programmed with the correct values.

CCIX Start

Example of programming order is as follows:

- Setup SAM
- Setup RA
- Setup Link
- Setup Port
- Activate Link
  - Set credit for TL and PL levels
  - Enable TL and PL Credit Exchange as well as activate Link
- Enable Device

CCIX Stop

The recommended procedure to take down the CCIX link is:

- Optionally disable all Agents
- Switch from PL Credit Exchange to Credit Return, when all are returned move to next stage

- Switch from TL Credit Exchange to Credit Return
- Deactivate Link
- Optionally disable Device

CHI Master Port Coherency

On Versal devices CHI is available in System Cache to create a coherent cache for accelerators by connecting to CPM. Either coherency remains local or CPM uses CCIX, which allows transparent data sharing between the node and agents.

Similar to the CCIX only case, firmware is used to control all the negotiations in the system of choice.

CHI Start

Example of programming order is as follows:

- Setup RN-F
- Enable SYSCOREQ for coherency
- Enable TL Credit Exchange

CHI Stop

The recommended procedure to take down the CHI link is:

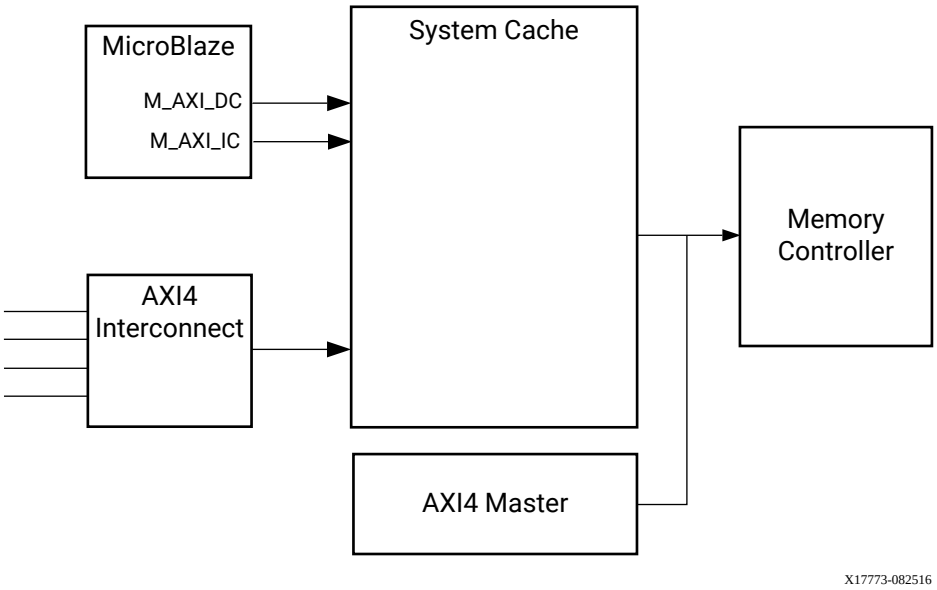
- Disable all Accelerators
- Disable RN-F
- Switch from TL Credit Exchange to Credit Return

Back-Door DMA

When the System Cache core shares the memory controller(s) with at least one other master, a back-door DMA connection is opened up. The following figure shows a configuration with one additional master. This back-door connection creates issues with data visibility; if an address is allocated in the System Cache core it hides data newly updated by the MicroBlaze™ processor from the other masters that access the memory directly, because the System Cache core uses a write-back storage policy. Similarly, updates from other masters to main memory are hidden from the MicroBlaze processor due to the system cache allocation, which provides the obsolete data that has been previously allocated.

There are multiple solutions to the issues that arise from this system design, depending on the System Cache core configuration. All methods are designed to make sure that an address is not allocated in any of the cache lines in a set.

Figure 94: **Back-Door DMA Connection with One Additional Master**



With Optimized Port Cache Coherency

Optimized port cache coherency enables communication directly from the MicroBlaze™ processor to the System Cache core with sideband information. In this case the MicroBlaze processor code can use the `WDC.EXT` instructions. There are two types of cache maintenance operations, flush or clear, that can be used depending on how dirty data in the System Cache core for this memory region is handled.

Flush is used when the old data should be retained. This is useful for keeping the old data when the memory region is only partially or sparsely updated from the other master. A flush operation must be used before the other master writes new data or new data might be lost when flushing the system cache. Flush is also used to make sure data written from a MicroBlaze processor is also visible to the other back-door masters.

Clear should be used when the old data is no longer needed or before the entire memory region is completely updated. Clear operations are typically faster than flush because they do not add transactions on the `M_AXI` interface of the System Cache core, which have the potential of introducing stall conditions depending on utilization level.

With ACE Master Port Cache Coherency

When master port cache coherency is enabled the System Cache core is only coherent to the caches in the APU. Other masters such as the R5 real time processors need to use customary cache maintenance actions to ensure that they can see data from the coherent domain when required. This includes operations on its own caches, as well as ensuring that all dirty data has been moved downstream to the final destination in order to be visible.

With Control Interface

The control interface can be used when cache coherency is not implemented or if a master not connected to one of the optimized ACE ports should handle the cache maintenance. Both Flush and Clear type of operations are available by writing to registers on the control interface. The rules for when Flush or Clear should be used are the same as for the cache coherent case.

The cache maintenance operations from the control port have lower priority than ordinary transactions from the optimized and generic ports. Due to this they can be slower than the equivalent cache coherent counterpart if the cache load remains high during cache maintenance.

When Non-Secure handling is enabled there are two separate sets of registers for Secure and Non-Secure Clear and Flush.

Without Cache Coherency and Control Interface

When neither cache maintenance, through the Optimized ACE ports, nor Control interface are available, cache lines can still be evicted but with less control and more overhead. This eviction is achieved by reading enough dummy data to ensure that an address is evicted from any of the Cache Lines in a Set that the address can be mapped to. The amount of dummy data that needs to be read equals the set associativity that is implemented.

Preferably the dummy data should be located a multiple of the System Cache core cache size away from the memory region that needs to be cleared. From this point the dummy reads should be performed with a distance of the system cache size (`C_CACHE_SIZE`) divided by number of sets association (`C_NUM_WAYS`). This has to be repeated for each cache line that is covered by the memory region that should be cleared.

For large memory regions it is probably easiest to read data in a system cache sized memory region with a step size of the system cache line length in bytes (`4 * C_CACHE_LINE_LENGTH`). As this method is equivalent to a flush cache maintenance operation it has to be performed before another master updates the memory locations in question, otherwise old dirty data could potentially overwrite the new data as cache lines are evicted.

Clocking

The System Cache core is fully synchronous with all interfaces and the internal function clocked by the ACLK input signal. It is advisable to avoid asynchronous clock transitions in the system as they add latency and consumes area resources.

Resets

The System Cache core is reset by the ARESETN input signal. ARESETN is synchronous to ACLK and needs be asserted one ACLK cycle to take affect. The System Cache core is ready for statistic register operation three ACLK cycles after ARESETN is deasserted. Before the System Cache core is available for general data access the entire memory is cleared (all previous content is discarded). The time it takes to clear the cache depends on the configuration; the approximate time is 2*C_CACHE_SIZE/(4*C_CACHE_LINE_LENGTH) clock cycles.

When Address Translation is enabled for the System Cache core, the ATC table is silently invalidated after reset (all previous content is discarded). The time it takes to clear the ATC table depends on the configuration; the approximate time is C_ATC_SIZE+2 (default 258) clock cycles.

The ATC Table can also be silently invalidated in the same way when reset via the System Cache control register, or if the ATS EN field in ATS & PRI Control register changes from Clear to Set.

Protocol Description

All interfaces to the System Cache core adhere to the AXI4, AXI4-Lite, AXI4-Stream, ACE, and CXS protocols with limitations.

MicroBlaze Processor Optimized AXI4 Slave Interface

The System Cache core has up to 16 AXI4 interfaces optimized for accesses performed by the cache interfaces on the MicroBlaze™ processor. Because the MicroBlaze processor has one AXI4 interface for the instruction cache and one for the data cache, systems with up to eight MicroBlaze processors can be fully connected.

By using a 1:1 AXI4 interconnect to directly connect the MicroBlaze processor and the System Cache core, access latency for MicroBlaze processor cache misses is reduced, which improves performance. The optimization to only handle the types of AXI4 accesses issued by the MicroBlaze processor simplifies the implementation, saving area resources as well as improving performance. The data widths of the MicroBlaze processor optimized interfaces are parameterized to match the data widths of the connected MicroBlaze processors. With wide interfaces the MicroBlaze processor cache line length normally determines the data width.

The Optimized AXI4 slave interfaces are compliant to a subset of the AXI4 interface specification. The interface includes the subsequent features and exceptions:

- Support for 32-, 128-, 256-, and 512-bit data widths
- Support for some AXI4 burst types and sizes
  - No support for FIXED bursts
  - WRAP bursts corresponding to the MicroBlaze processor cache line length (either 4, 8, or 16 beats)
  - Single beat INCR burst, or either 4, 8, or 16 beats corresponding to the MicroBlaze processor cache line length
  - Exclusive accesses are treated as a normal accesses, never returning EXOKAY, unless the internal exclusive monitor is enabled
  - Optional support for Secure/Non-Secure handling
  - Only support for native transaction size (same as data width for the port)
- Only support for burst transactions that are contained within a single cache line in the System Cache core
- AXI4 user signals are not supported
- All transactions executed in order regardless of thread ID value. No read reordering or write reordering is implemented.

MicroBlaze Processor Optimized ACE Slave Interface

The optimized AXI4 interfaces are replaced by ACE point-to-point connections when cache coherency is enabled. They are optimized for accesses performed by the cache interfaces on the MicroBlaze™ processor. Eight MicroBlaze processors are supported with coherency enabled.

The Optimized ACE slave interfaces are compliant to a subset of the ACE interface specification. The interface includes the subsequent features and exceptions:

- Support for 32-bit data width
- Support for some ACE burst types and sizes
  - No support for FIXED bursts

- Optional support for Secure/Non-Secure handling
- Only sharable transactions are supported
- WRAP bursts corresponding to the WRAP bursts corresponding to the MicroBlaze™ processor cache line length (either 4, 8, or 16 beats) processor cache line length (either 4, 8, or 16 beats)
- Single beat INCR burst, or either 4, 8, or 16 beats corresponding to the MicroBlaze™ processor cache line length
- The AR channel supports `ReadOnce`, `ReadClean`, `CleanUnique`, `CleanInvalid`, `MakeInvalid` and `DVM` transactions
- Support for propagation of `CleanInvalid` and `MakeInvalid` to peer or downstream caches
- The AW channel supports `WriteUnique` only
- Exclusive accesses are only supported for sharable transactions (with ACE transactions exchange)
- Only support for native transaction size bursts (same as data width for the port). Single beat support for all sizes
- The AC channel can generate `ReadClean`, `ReadOnce`, `CleanUnique`, `CleanInvalid`, `MakeInvalid` and `DVM` transactions
- CD channel data is not used

- Only support for burst transactions that are contained within a single cache line in the System Cache core
- Only support for Write-Through cache in the MicroBlaze™ processor
- AXI4 user signals are not supported
- All transactions executed in order regardless of thread ID value. No read reordering or write reordering is implemented.

Generic AXI4 Slave Interface

To handle several AXI4 masters in a system two methods are available: either an AXI4 interconnect is used to share the single generic AXI4 slave interface on the System Cache core or, alternatively, multiple generic AXI4 interfaces are used (most commonly one per master that needs access). The generic AXI4 interface has a configurable data width to efficiently match the connected AXI4 masters. This ensures that both the system area and the AXI4 access latency are reduced.

The generic AXI4 slave interface is compliant to the full AXI4 interface specification. The interface includes the subsequent features and exceptions:

- Support for 32-, 64-, 128-, 256-, and 512-bit data widths

- Support for all burst types and sizes
  - No support for FIXED bursts
  - Up to 16 beats for WRAP bursts
  - Up to 256 beats for INCR burst
  - Optional support for Secure/Non-Secure handling
  - Exclusive accesses are treated as a normal accesses, never returning EXOKAY, unless the internal exclusive monitor is enabled
- Support for burst sizes that are less than the data width (narrow bursts)
- AXI4 user signals are supported with CCIX Master Coherency enabled, to provide atomic transactions
- Out-of-order transactions based on thread ID value are supported with CCIX Master Coherency enabled. Otherwise all transactions are executed in order regardless of thread ID value, and no read reordering or write reordering is implemented.

AXI4-Lite Control Slave Interface

The AXI4-Lite control slave interface is compliant with the AXI4-Lite interface specification. The interface includes the following features and options:

- Support for 32- and 64-bit data bus widths
- Requires a 128K byte address range
- Write strobes are ignored and all write accesses are treated as being the full data bus width, because the optional WSTRB signal is omitted from the interface

***Note*:** Register writes with no write strobes asserted are performed as if they were the full data bus width.

AXI4 Master Interface

The AXI4 master interface is used to connect the external memory controller. The data width of the interface can be parameterized to match the data width of the AXI4 slave interface on the memory controller. For best performance and resource usage, the parameters on the interface and the Memory Controller should match.

The AXI4 master interface is compliant to the AXI4 interface specification. The interface includes the following features:

- Support for 32-, 64-, 128-, 256-, and 512-bit data widths
- Generates the following AXI4 burst types and sizes
  - 2 - 16 beats for WRAP bursts

    - 1 - 64 beats for INCR burst
- AXI4 user signals are not provided
- A single thread ID value is generated

ACE Master Interface

The ACE master interface is compliant to the ACE interface specification. The interface includes the following features:

- Only 128-bit data widths
- Generates the following ACE burst types and sizes
    - 2 - 16 beats for WRAP bursts
    - 1 - 64 beats for INCR burst
- ACE user signals are not provided
- Two thread IDs are used, one for data transactions and one for `DVM`
- Any `DVM` message can be inserted through the control interface
- Memory or Synchronization barriers
- AR channel can generate the following types of data accesses
    - `ReadOnce`, `ReadShared`, `ReadUnique`, `CleanUnique`, `CleanInvalid` and `MakeInvalid`
- AW channel can generate the following types of data accesses
    - `WriteBack` and `WriteUnique`
- All snoop transactions are supported on the AC channel
- An internal exclusive monitor is included to track exclusive transactions

CCIX Master Interface

The CCIX master interface is compliant to the CCIX Base Specification Revision 1.1 Version 1.0. The following features are supported:

- CXS interface 256 and 512 bits data width, and 1024 bits data width for Versal Premium devices
- Optimized and Compatible header
- Packing enabled or disabled
    - Chaining of memory requests

- One CCIX Port
  - One CCIX link per port
- One Request Agent (RA)
- 64 byte cache line
  - All available sizes are supported for Read and Write requests
- All address widths (48, 52, 56, 60 and 64-bit)
- 128, 256 and 512 byte Maximum Packet Size (MPS)
- The following transactions are generated under normal conditions:
  - `ReadNoSnp`, `ReadOnce`, `ReadUnique`, `ReadShared`, `CleanShared`, `CleanUnique`, `CleanInvalid`, `MakeInvalid`, `WriteNoSnp`, `WriteUnique`, `WriteBackFullSD`, `WriteBackFullUD`
- The following transactions can be generated when Atomics is enabled:
  - `AtomicStore_ADD`, `AtomicStore_CLR`, `AtomicStore_EOR`, `AtomicStore_SET`, `AtomicStore_SMAX`, `AtomicStore_SMIN`, `AtomicStore_UMAX`, `AtomicStore_UMIN`, `AtomicLoad_ADD`, `AtomicLoad_CLR`, `Atomicoad_EOR`, `AtomicLoad_SET`, `AtomicLoad_SMAX`, `AtomicLoad_SMIN`, `AtomicLoad_UMAX`, `AtomicLoad_UMIN`, `AtomicSwap`, `AtomicCompare`
  - SnpME variants of Atomics
- All snoop combinations

CHI Master Interface

The CHI master interface is compliant to Arm AMBA-5 CHI Architecture Specification, Arm IHI0050B. The following features are supported:

- One Request Node (RN-F)
- Data width 512 bits
- Address width 48 bits
- 64 byte cache line
  - All available sizes are supported for Read and Write requests
- The following transactions are generated under normal conditions:
  - `ReadNoSnp`, `ReadOnce`, `ReadUnique`, `ReadShared`, `CleanShared`, `CleanUnique`, `CleanInvalid`, `MakeInvalid`, `WriteNoSnp`, `WriteUnique`, `WriteBackFullSD`, `WriteBackFullUD`

- The following transactions can be generated when atomic transactions are enabled:
  - `AtomicStore_ADD`, `AtomicStore_CLR`, `AtomicStore_EOR`, `AtomicStore_SET`, `AtomicStore_SMAX`, `AtomicStore_SMIN`, `AtomicStore_UMAX`, `AtomicStore_UMIN`, `AtomicLoad_ADD`, `AtomicLoad_CLR`, `Atomicoad_EOR`, `AtomicLoad_SET`, `AtomicLoad_SMAX`, `AtomicLoad_SMIN`, `AtomicLoad_UMAX`, `AtomicLoad_UMIN`, `AtomicSwap`, `AtomicCompare`
- All normal and forward snoop combinations

AXI4-Stream ATS Interfaces

The ATS interface uses four AXI4-Stream channels: CC, CQ, RC and RQ. These channels are connected to PCIe, VC0, when used together with CCIX or CHI. The following features are supported:

- Interface width 256 or 512 bits, and also 1024 bits for Versal Premium devices, must be the same width as the CXS width with CCIX
- Either 256 or 512-bit (1024-bit for Versal Premium devices) wide interfaces can be used with the CHI, independent of the CHI data width (must be same as PCIe VC0)
- Per channel user signals are supported with CCIX on UltraScale™ and UltraScale+™ devices according to the definition in *Integrated Block for PCI Express* (PG213)
- Per channel user signals are supported with CHI on Versal ACAP devices according to the definition in *Versal ACAP Integrated Block for PCI Express* (PG343)
- Per channel user signal width, definition depends on data width and optional PASID width for CHI (Versal) and CHI/CCIX (Versal Premium)
- Burst sizes via header PCIe packet length, limited by Message type and system MPS configured by firmware in the System Cache ATS PCIe Control register
- Packet delimiter by AXI4-Stream protocol or by straddle user signals when the option is selected
- Single and Burst multibeat transactions qualified by AXI4-Stream protocol or by straddle user signals when the option is selected
- Slave side back pressure allowed
- Parity protected, per byte odd parity bit, when the option is selected
- Supports abortion and rejection of header and payload via user signals

Design Flow Steps

This section describes customizing and generating the core, constraining the core, and the simulation, synthesis, and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
- *Vivado Design Suite User Guide: Designing with IP* (UG896)
- *Vivado Design Suite User Guide: Getting Started* (UG910)
- *Vivado Design Suite User Guide: Logic Simulation* (UG900)

Customizing and Generating the Core

This section includes information about using Xilinx® tools to customize and generate the core in the Vivado® Design Suite.

If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl console.

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) and the *Vivado Design Suite User Guide: Getting Started* (UG910).

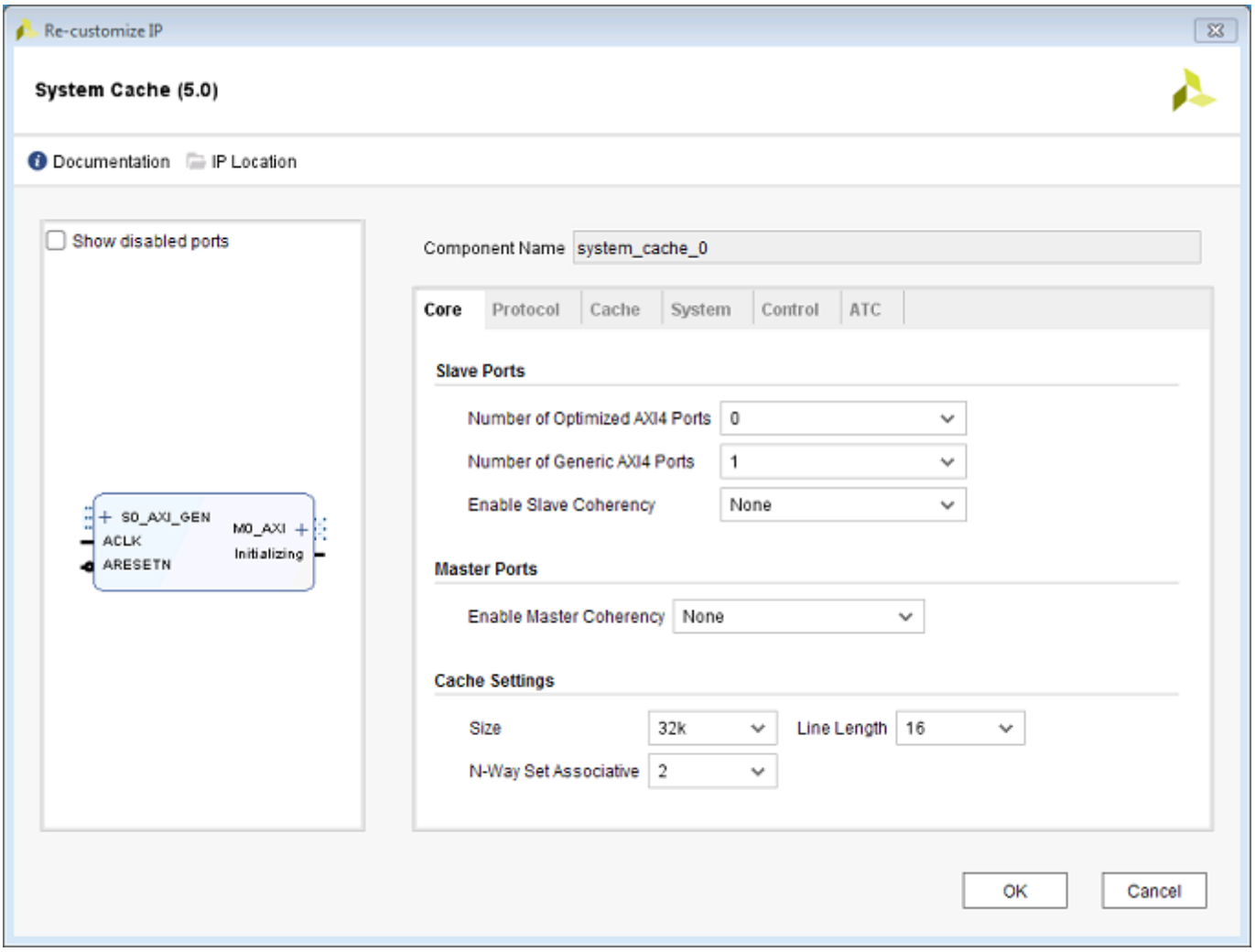
The System Cache core parameters are divided into six categories: core, protocol, cache, system, control and ATC. Additionally, when ACE, CCIX or CHI Master Coherency is enabled, the coherency category is also shown, and when CCIX Master Coherency is enabled the SAM category is shown. See the following table for allowed values..

Figures in this chapter are illustrations of the Vivado IDE. The layout depicted here might vary from the current version.

Core Parameters Tab

The Core Parameters Tab is shown in the following figure.

Figure 95: **Core Parameters Tab**



- **Number of Optimized AXI4 Ports:** Sets the number of optimized ports that are available to connect to a MicroBlaze processor or equivalent IP in terms of AXI4/ACE transaction support.

- **Number of Generic AXI4 Ports:** Set the number of generic AXI4 ports that are available for IP cores not adhering to the AXI4 subset required for an optimized port, such as DMA.
- **Enable Slave Coherency:** Set slave port cache coherency mode to None or ACE Coherency Protocol.
- **Enable Master Coherency:** Set master port cache coherency mode to None, ACE Coherency Protocol, CCIX Coherency Protocol or CHI Coherency Protocol.
- **Size:** Sets the size of the system cache in bytes from 32k to 4M.
- **N-Way Set Associative:** Specifies 2- or 4-way set associative cache.
- **Line Length:** Set the cache line length in 32-bit words from 16 to 1024. For CCIX, CHI and ACE master coherency the cache line length is fixed to 16.

Figure 96: **Core Parameters Tab with CCIX Coherency Protocol**

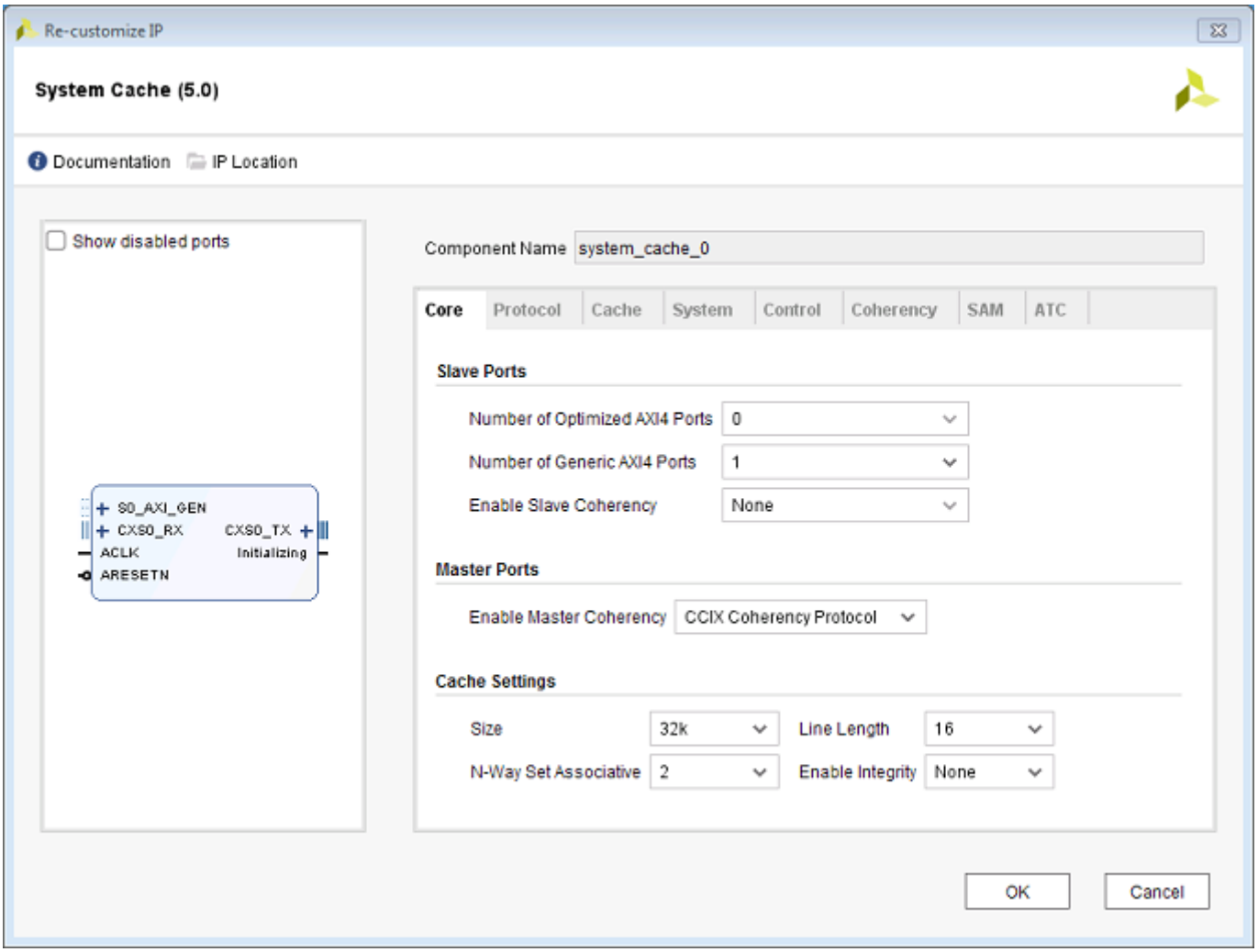
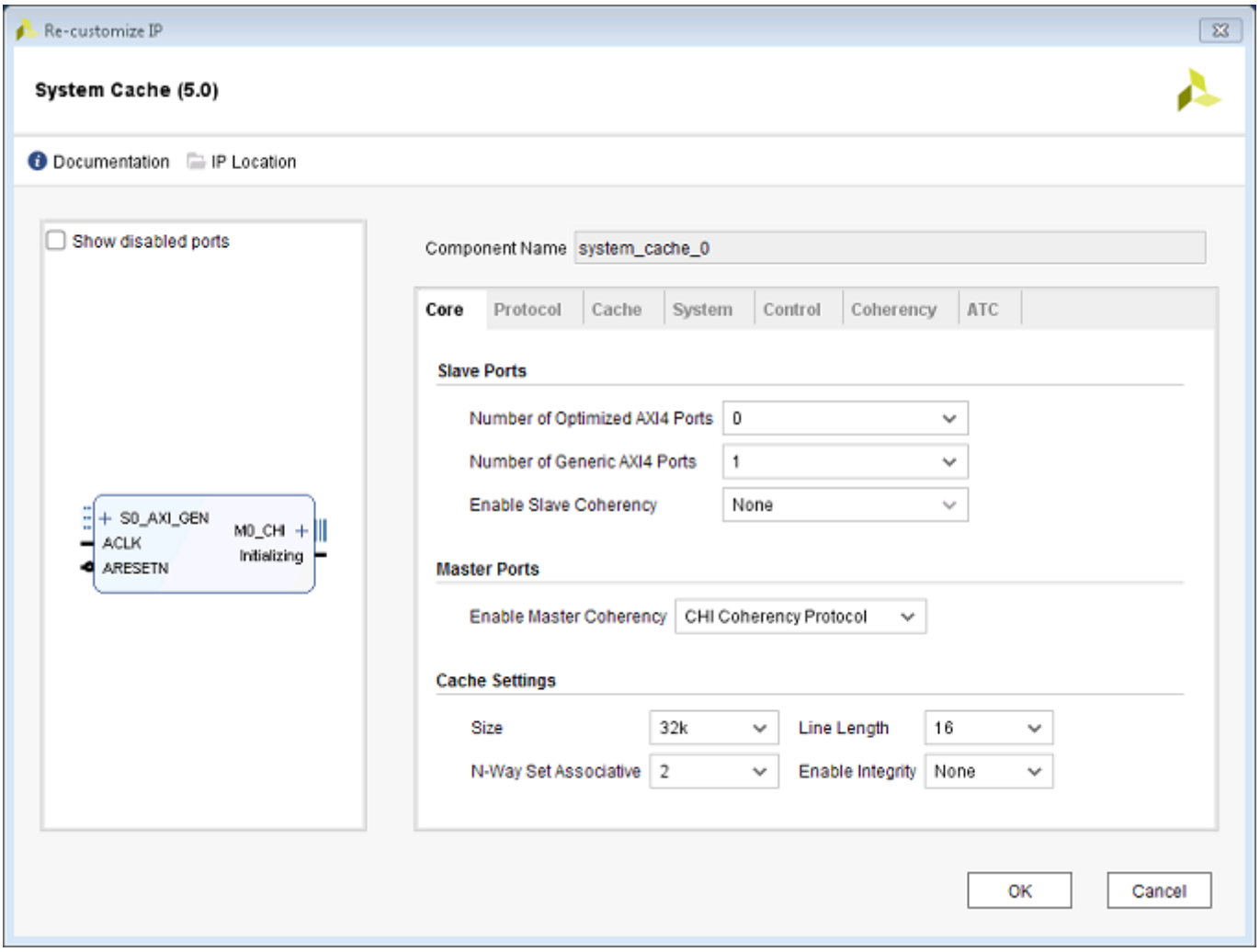


Figure 97: **Core Parameters Tab with CHI Coherency Protocol**



- **Enable Integrity:** Enable Error Correcting Codes (ECC) for the cache tag, and parity for cache data. If address translation is used, ATC parity is also enabled.

Protocol Parameters Tab

The Protocol Parameters Tab is shown in the following figure.

Figure 98: **Protocol Parameters Tab**

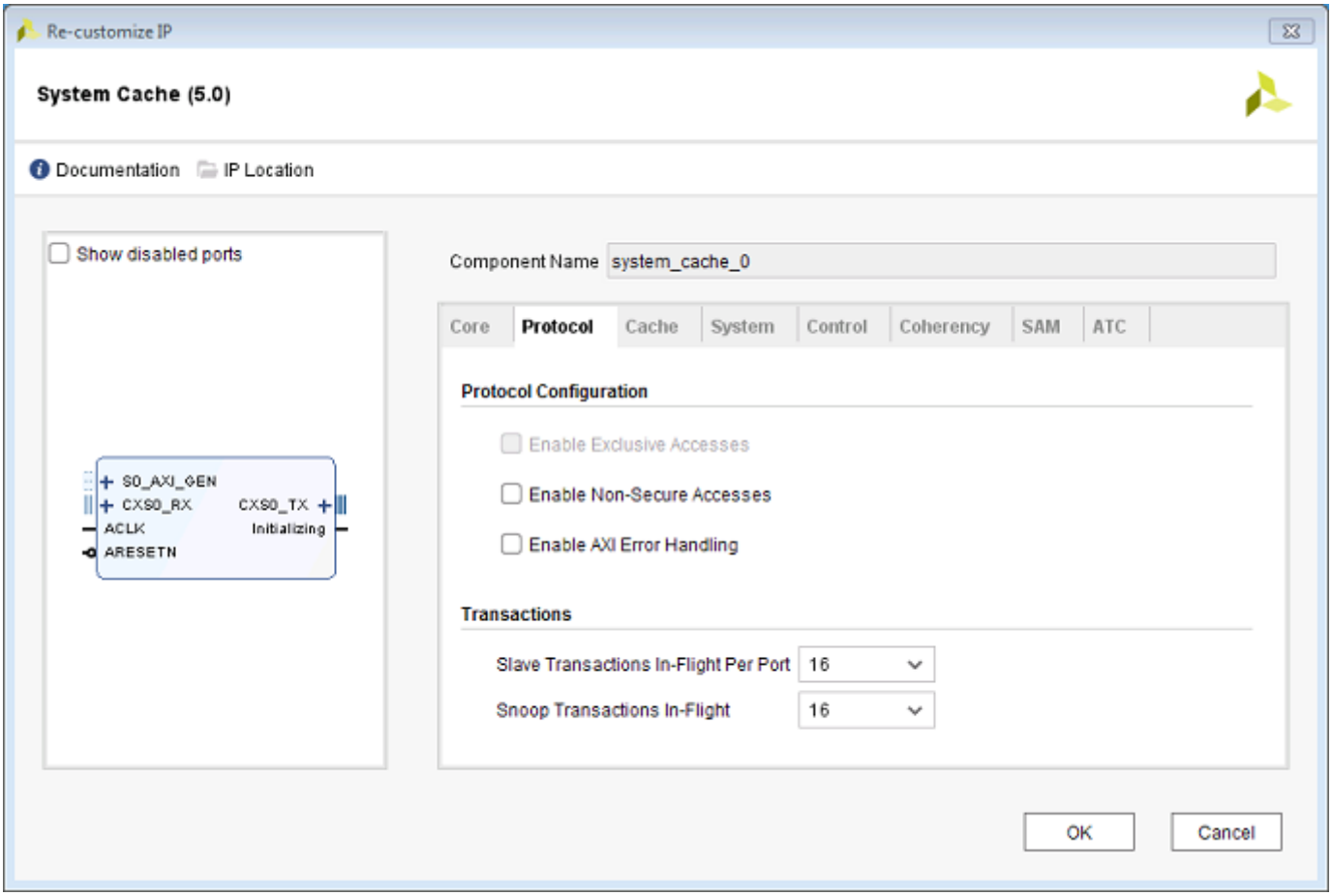
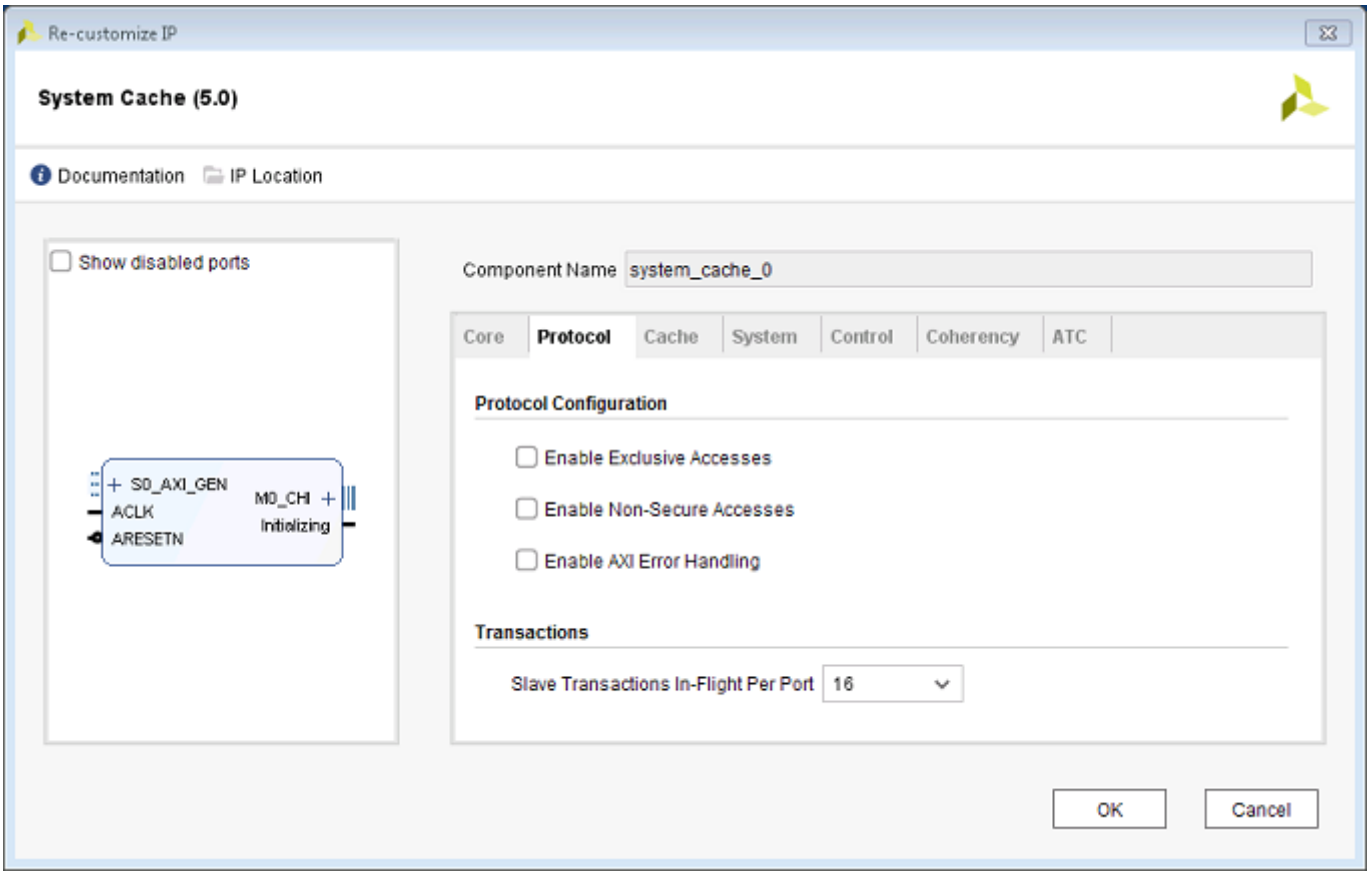


Figure 99: **Protocol Parameters Tab with CHI Coherency Protocol**

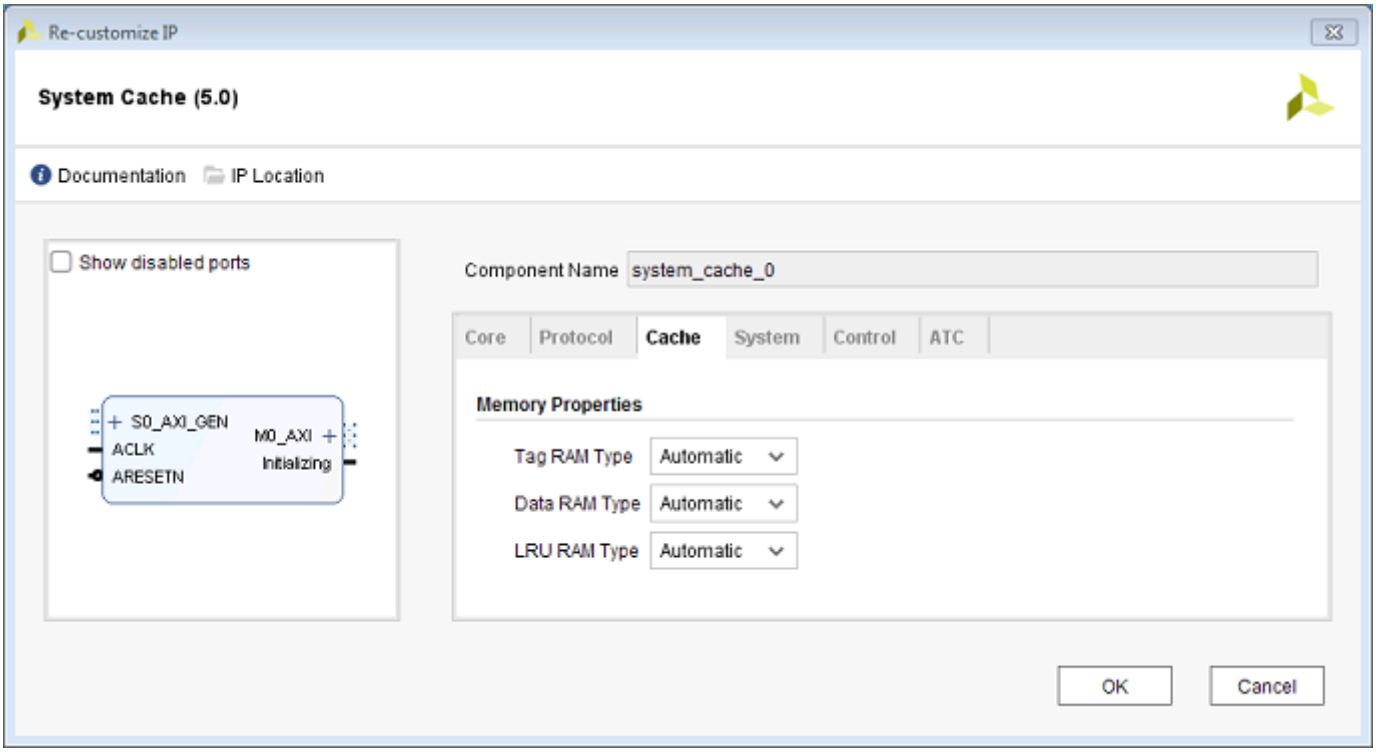


- **Enable Exclusive Access:** Enables Exclusive handling for non-coherent implementation. Disabled with ACE slave coherency protocol and CCIX master coherency protocol
- **Enable Non-Secure Access:** Enable feature to distinguish between and cache both Secure and Non-Secure transactions.
- **Enable AXI Error Handling:** Enable feature to make AXI errors prevent allocation of line.
- **Slave Transactions In-Flight Per Port:** Maximum number of transactions in flight per channel for each slave port. Only available with CCIX and CHI master coherency protocol.
- **Snoop Transactions In-Flight:** Maximum number of snoop requests handled simultaneously. Only available with CCIX master coherency protocol.

Cache Parameters Tab

The Cache Parameters Tab is shown in the following figure.

Figure 100: **Cache Parameters Tab**

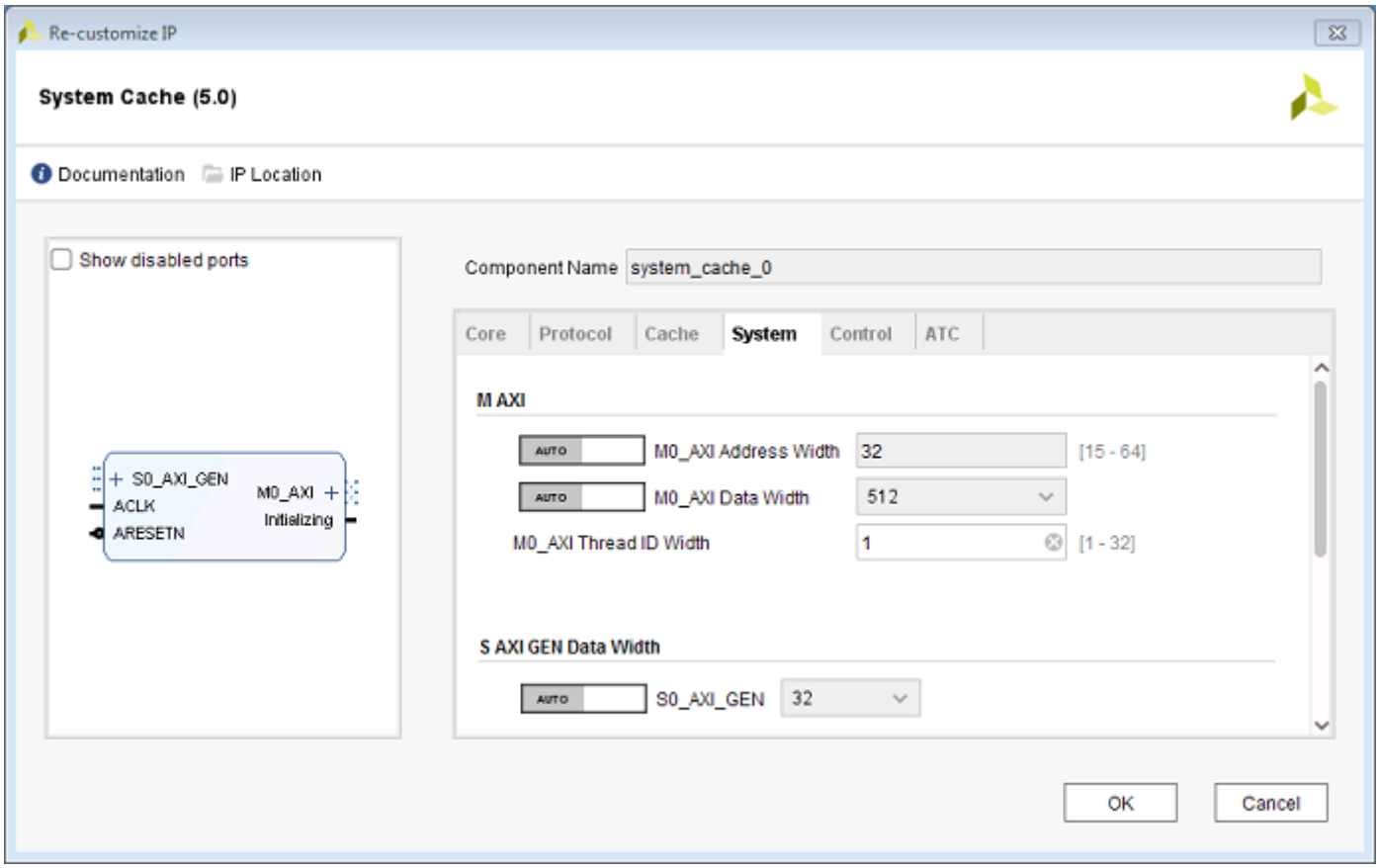


- **Tag RAM Type:** Select memory type used for cache tags. Automatic is normally recommended, but a specific memory type can be selected if necessary.
- **Data RAM Type:** Select memory type used for cache data. Automatic is normally recommended, but a specific memory type can be selected if necessary.
- **LRU RAM Type:** Select memory type used for Least Recently Used (LRU) flags. Automatic is normally recommended, but a specific memory type can be selected if necessary.

System Parameters Tab

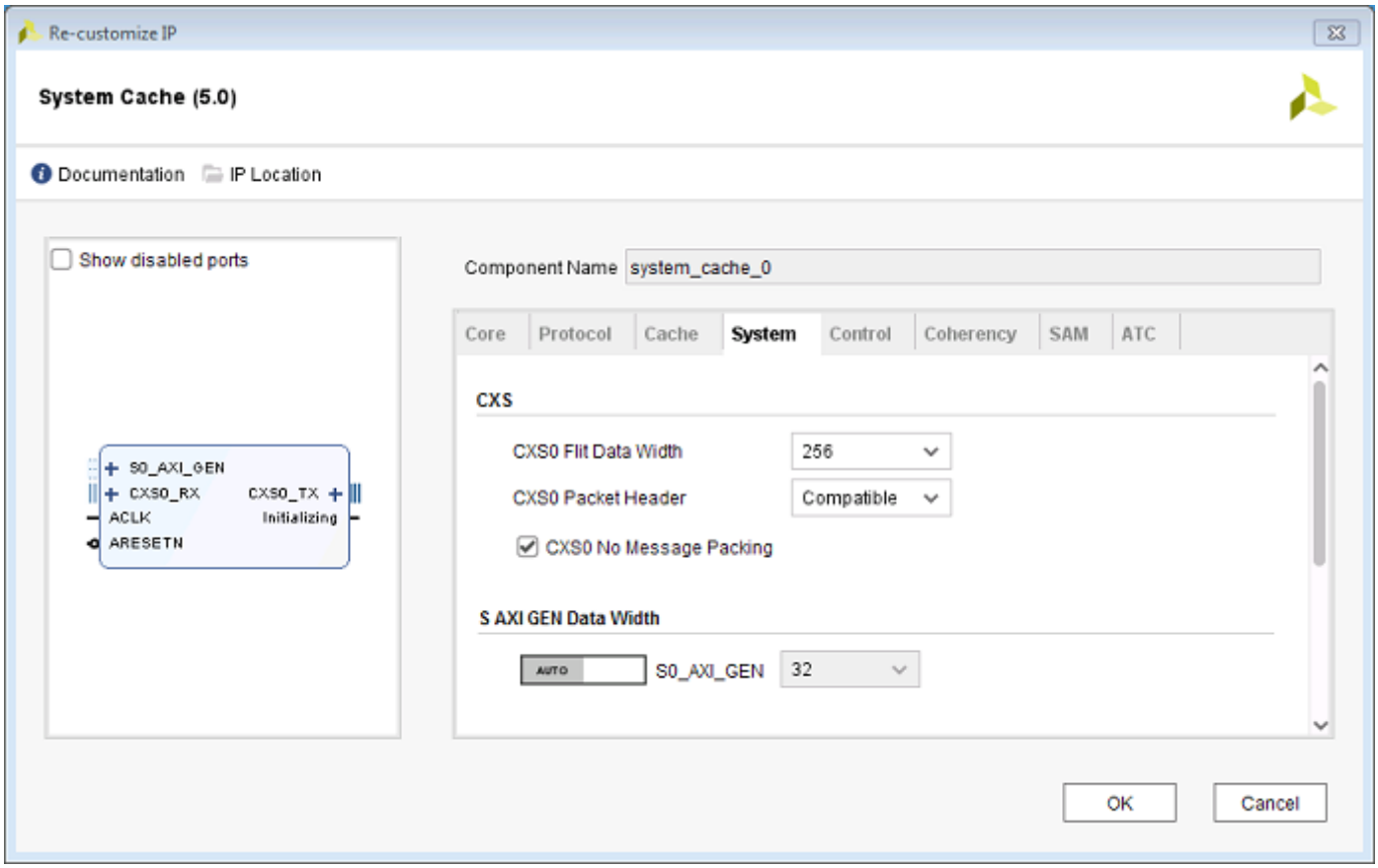
The system parameter tab for AXI, CCIX and CHI master interfaces is shown in the following figures with the data width parameters visible for the slave interface.

Figure 101: **System Parameter Tab with AXI Master Interface**



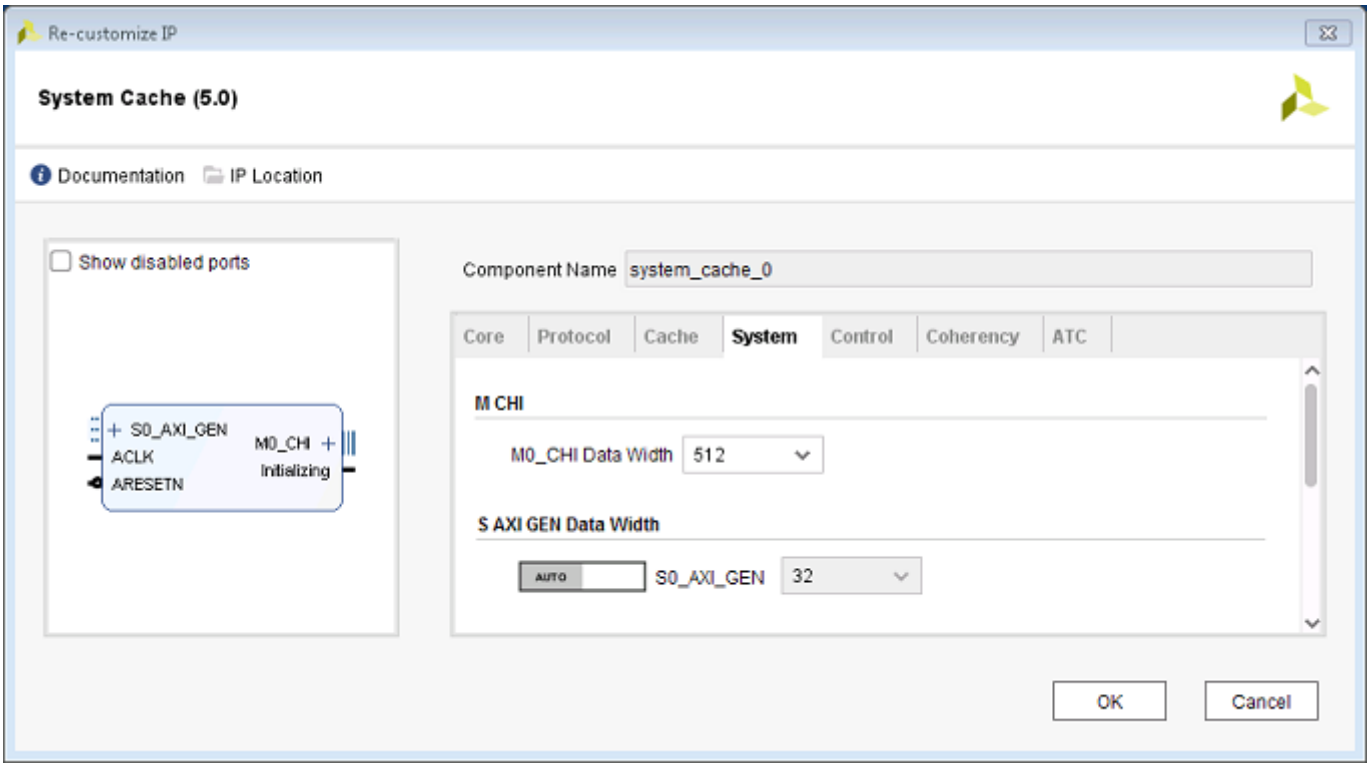
- **M_AXI Address Width:** Sets the address width of the master interface that is connected to the memory subsystem.
- **M_AXI Data Width:** Sets the data width of the master interface that is connected to the memory subsystem.
- **M_AXI Thread ID Width:** Sets the ID width of the master interface that is connected to the memory subsystem.
- **Sx_AXI_GEN Data Width:** Sets the data width of the generic ports individually.

Figure 102: **System Parameter Tab with CCIX Master Interface**



- **CXS0 Flit Data Width:** Sets the data width of the CXS interface flit to 256 or 512 bits, and also 1024 bits for Versal Premium devices.
- **CXS0 Packet Header:** Defines the CCIX packet header format, either Compatible or Optimized.
- **CXS0 No Message Packing:** Set if only a single message is packed into each CCIX packet.
- **Sx_AXI_GEN Data Width:** Sets the data width of the generic ports individually.

Figure 103: **System Parameter Tab with CHI Master Interface**

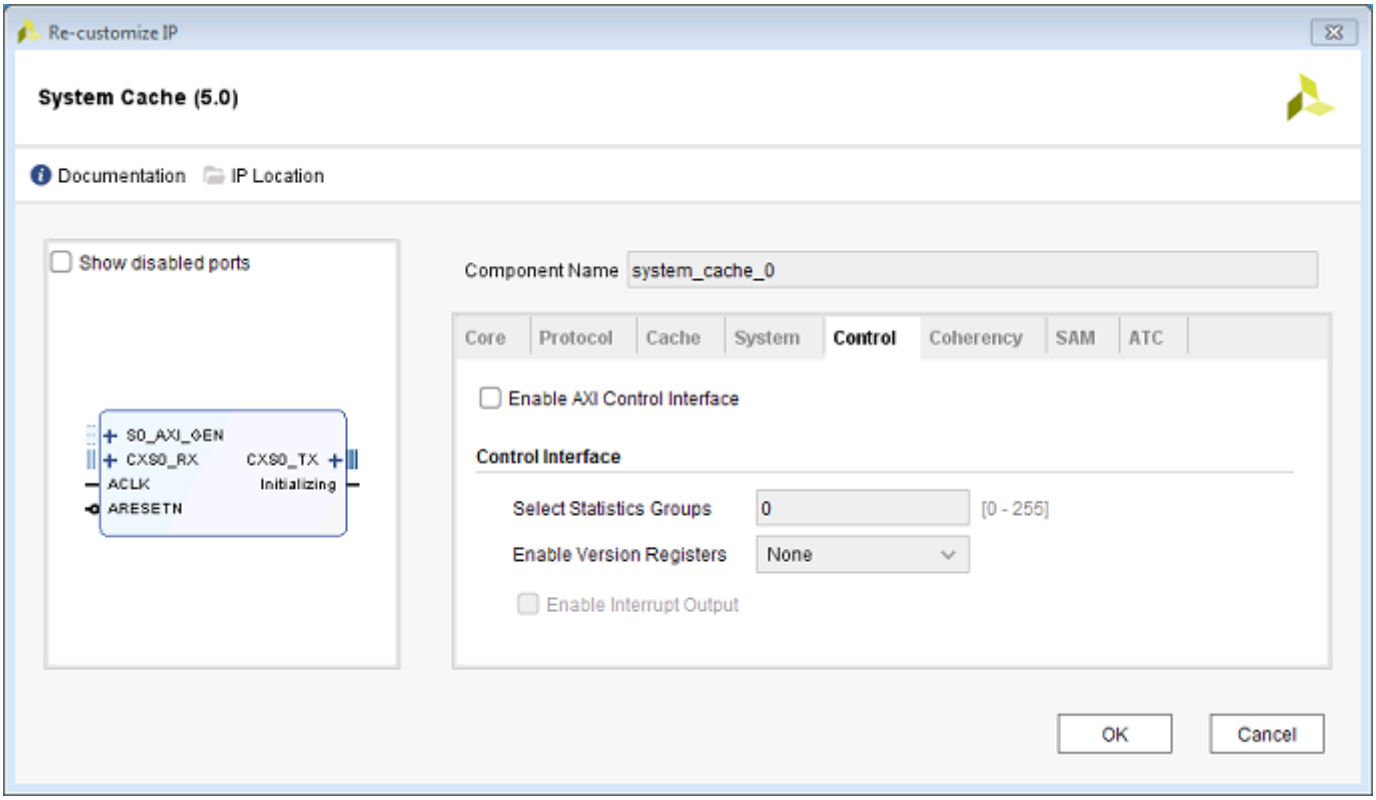


- **M0_CHI Data Width:** Sets the data width of the CHI interface.
- **Sx_AXI_GEN Data Width:** Sets the data width of the generic ports individually.

Control Parameters Tab

The Control Parameters Tab is shown in the following figure.

Figure 104: **Control Parameters Tab**



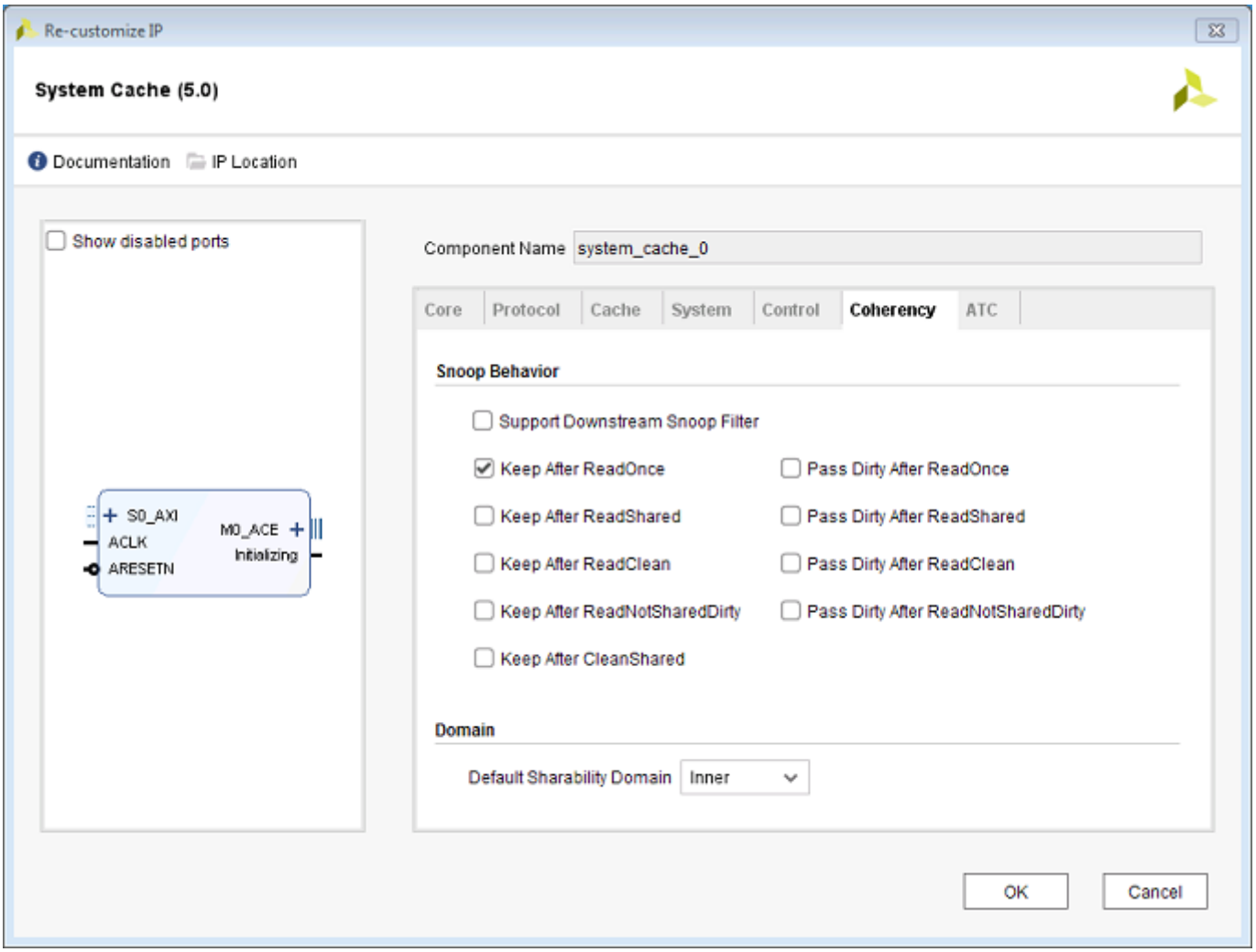
- **Enable AXI Control Interface:** Set if control interface is available.
- **Select Statistics Groups:** Bit mask that determines which statistics groups are included.
- **Enable Version Registers:** Set level of Version registers that should be included.
- **Enable Interrupt Output:** Enable the interrupt output and corresponding control registers. Only available with CCIX and CHI master coherency protocol.

Coherency Parameters Tab

The Coherency Parameters Tab is shown in the following figures.

Note: This tab is only shown when ACE, CCIX or CHI Master Coherency is enabled.

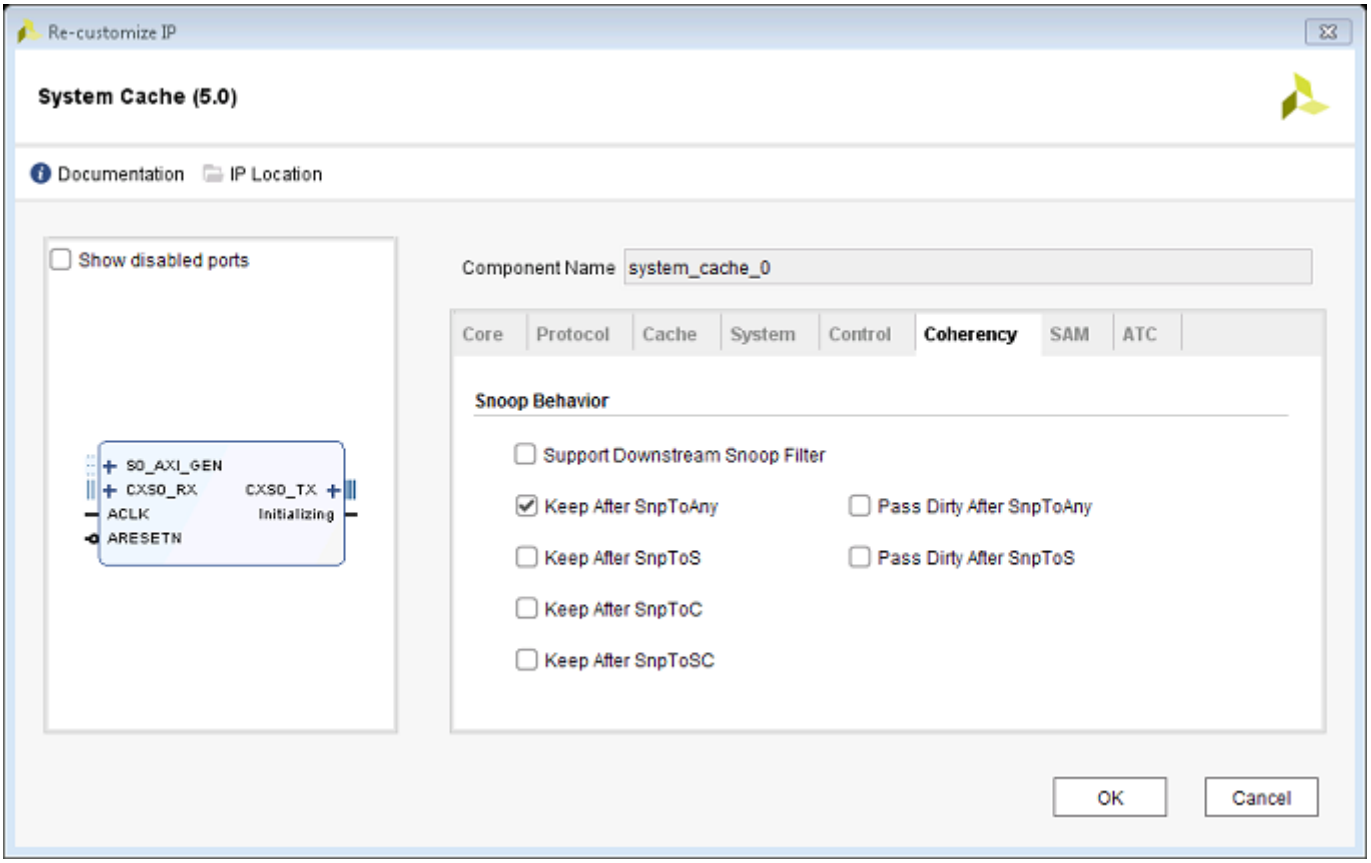
Figure 105: **Coherency Parameters Tab with ACE Coherency Protocol**



- **Support Downstream Snoop Filter:** Generate transactions to support a downstream snoop filter to improve performance.
- **Keep After:** Preference is to keep the allocation in the cache after a snoop of the indicated type. This may increase performance, but it is application dependent.
- **Pass Dirty After:** Always pass write responsibility with data for a snoop of the indicated type. This may increase performance, but it is application dependent.

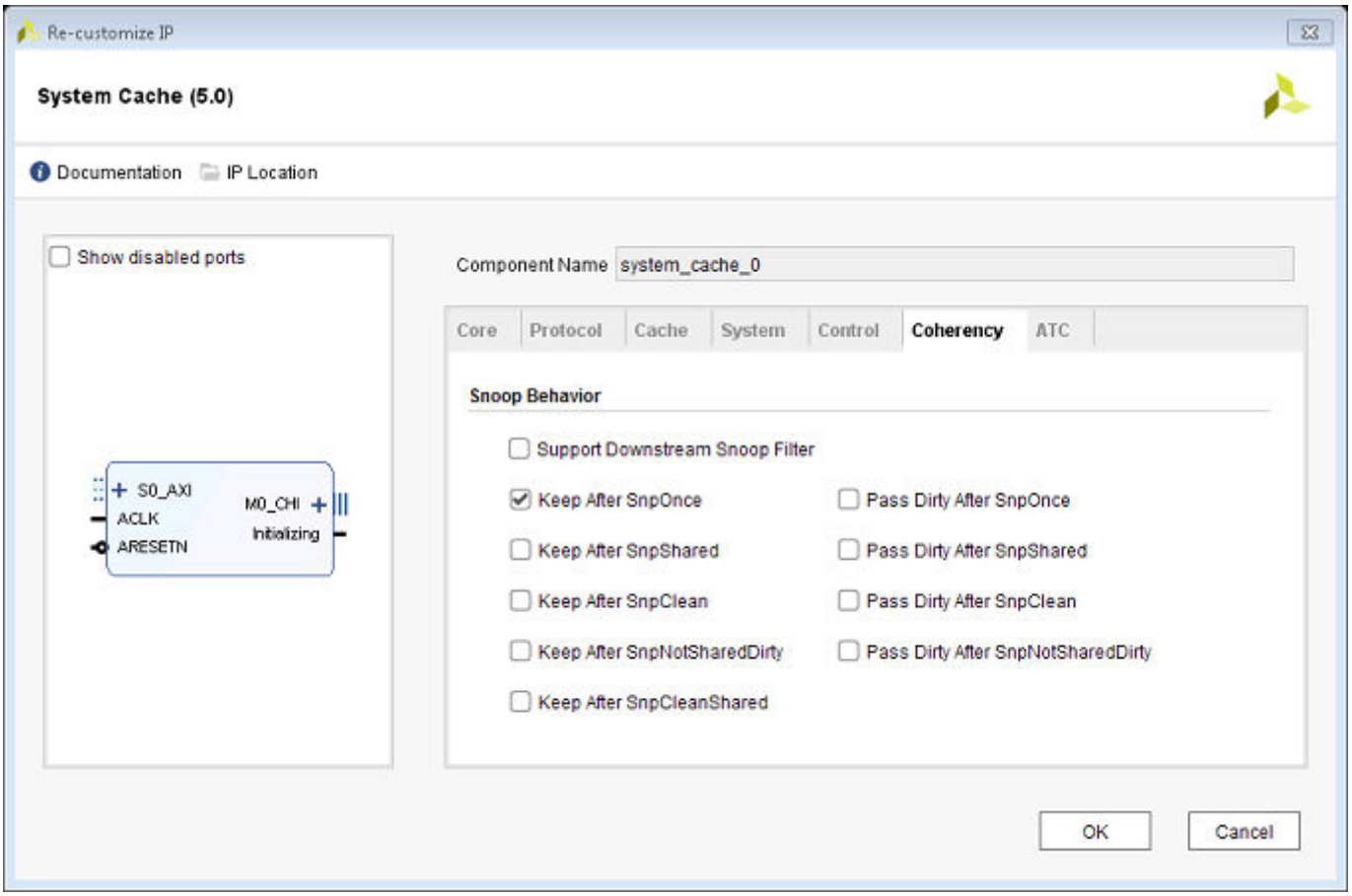
- **Default Sharability Domain:** Set default sharability domain to Inner or Outer. Only available with ACE master coherency protocol.

Figure 106: **Coherency Parameters Tab with CCIX Coherency Protocol**



- **Support Downstream Snoop Filter:** Generate transactions to support a downstream snoop filter to improve performance.
- **Keep After:** Preference is to keep the allocation in the cache after a snoop of the indicated type. This may increase performance, but it is application dependent.
- **Pass Dirty After:** Always pass write responsibility with data for a snoop of the indicated type. This may increase performance, but it is application dependent.

Figure 107: **Coherency Parameters Tab with CHI Coherency Protocol**



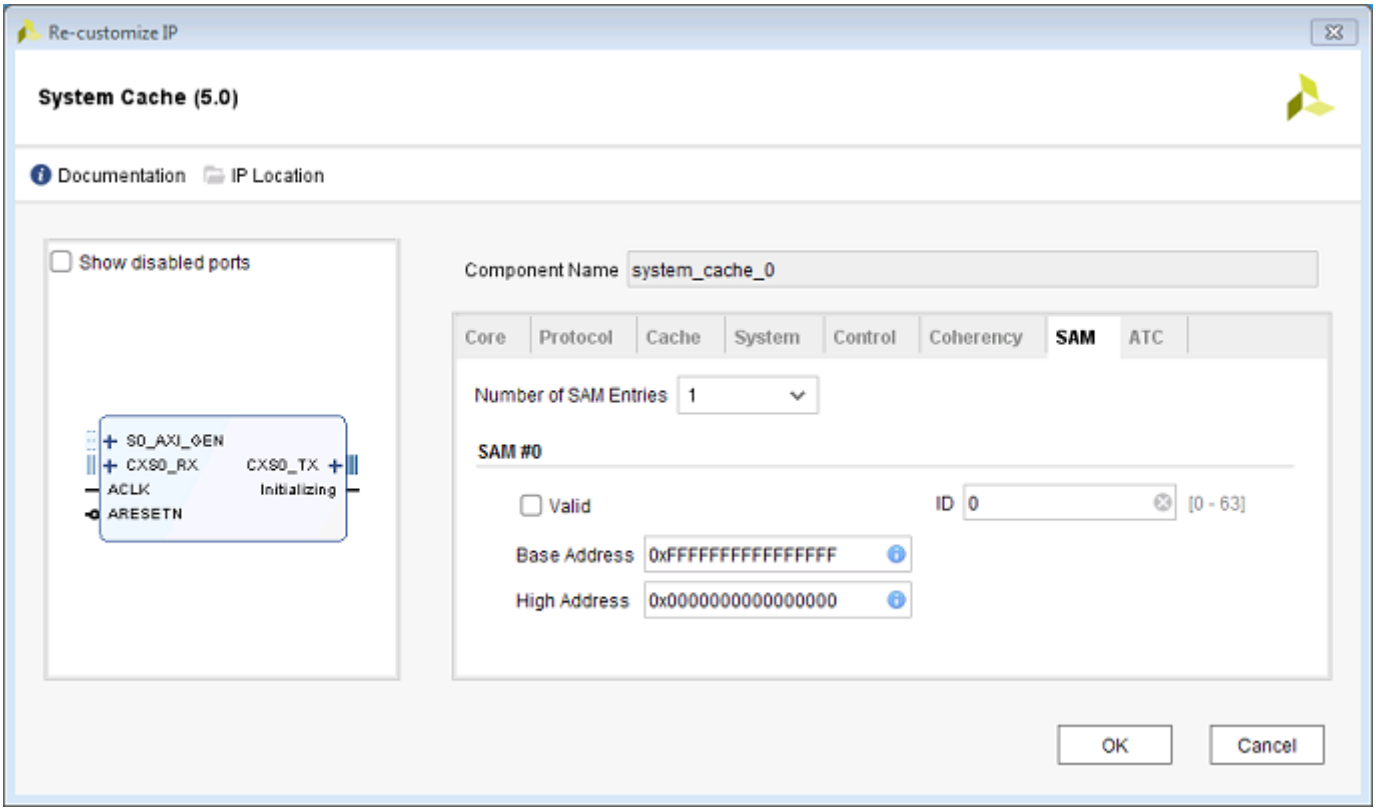
- **Support Downstream Snoop Filter:** Generate transactions to support a downstream snoop filter to improve performance.
- **Keep After:** Preference is to keep the allocation in the cache after a snoop of the indicated type. This may increase performance, but it is application dependent.
- **Pass Dirty After:** Always pass write responsibility with data for a snoop of the indicated type. This may increase performance, but it is application dependent.

SAM Parameters Tab

The System Address Map (SAM) Parameters Tab is shown in the following figure.

Note: This tab is only shown when CCIX Master Coherency is enabled.

Figure 108: **SAM Parameters Tab**



- **Number of SAM Entries:** Select number of System Address Map (SAM) entries. The values of the first four entries can be defined here.

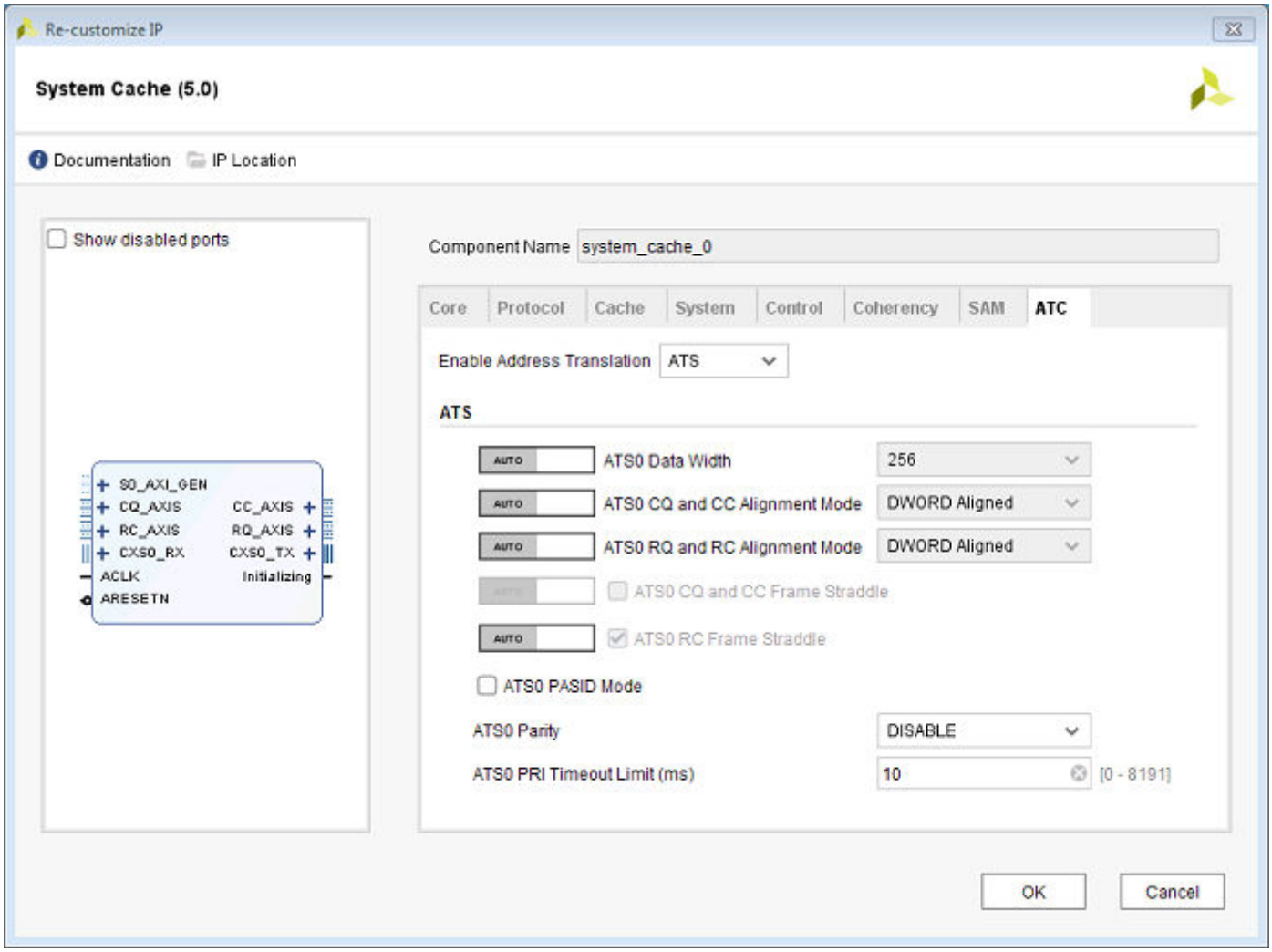
Each of the four entries has the following settings:

- **Valid:** System Address Map (SAM) entry is mapping a valid memory region.
- **ID:** Agent ID for Home Agent that owns this memory region.
- **Base Address:** Base address for memory region.
- **High Address:** High address or mask for memory region.

ATC Parameters Tab

The Address Translation Cache (ATC) Parameters Tab is shown in the following figure.

Figure 109: **ATC Parameters Tab**



- **Enable Address Translation:** Enable support for Address Translation Cache (ATC) with the PCIe Address Translation Service (ATS). The default is None.
- **ATS0 Data Width:** AXI4-Stream interface data width. Set this value to match the PCIe AXI4-Stream interface.
- **ATS0 CQ and CC Alignment Mode:** AXI4-Stream CQ and CC interface alignment mode. Set this value to DWORD or Address aligned to match the PCIe AXI4-Stream interface.
- **ATS0 RQ and RC Alignment Mode:** AXI4-Stream RQ and RC interface alignment mode. Set this value to DWORD or Address aligned to match the PCIe AXI4-Stream interface.

- **ATS0 CQ and CC Frame Straddle:** AXI4-Stream CQ and CC interface frame straddle when using 512-bit data width, or 1024 bit data width for Versal Premium devices, and DWORD alignment. Enable this to match the PCIe AXI4-Stream interface. Allows up to four packets to be transferred in the same beat on CQ and CC with Versal Premium devices.
- :
- **ATS0 RC Frame Straddle:** AXI4-Stream RC interface frame straddle. Enable this to match the PCIe AXI4-Stream interface. Used with 256-bit data width and DWORD alignment. Allows up to two packets to be transferred in the same beat.
- **ATS0 RQ and RC Frame Straddle:** AXI4-Stream RQ and RC interface frame straddle. Enable this to match the PCIe AXI4-Stream interface. Used with 512-bit, data width or 1024 bit data width for Versal Premium devices and DWORD alignment. Allows up to four packets, or eight packets with Versal Premium devices, to be transferred in the same beat on RC. Allows up to two packets, or four packets on Versal Premium devices, to be transferred in the same beat on RQ. Not visible in the previous figure.
- **ATS0 PASID Mode:** Enable handling of PCIe Process Address Space ID (PASID). ATC uses PASID unique translation mapping and global, PASID independent, invalidation. Multiple unique ATC translations and PCIe host relations to PASID requests must be handled by other means. PASID mode is only selectable if the target technology supports the feature and it is activated with support by the host and accelerators.
- **ATS0 Parity:** Enable parity on AXI4-Stream interfaces. Set to GENERATE to generate parity. Set to GENERATE CHECK to generate and check parity. This is independent of the integrity setting, and only applies to AXI4-Stream parity. If parity checking is disabled, packets with parity errors will be consumed unless they have other errors.
- **ATS0 PRI Timeout Limit:** Define the timeout limit for PCIe Page Request Interface (PRI). Setting the value to 0 disables the timeout.

Parameter Values

Certain parameters are only available in some configurations, others impose restrictions that IP cores connected to the System Cache core need to adhere to. All these restrictions are enforced by design rule checks to guarantee a valid configuration. The following tables describe the System Cache core parameters.

The parameter restrictions are:

- Internal cache data width must either be 32 or a multiple of the cache line length of masters connected to the optimized ports (`C_CACHE_DATA_WIDTH` = 32 or `C_CACHE_DATA_WIDTH` = `n` * 32 * `C_Lx_CACHE_LINE_LENGTH`).
- All optimized slave port data widths must be less than or equal to the internal cache data width (`C_Sx_AXI_DATA_WIDTH` ≤ `C_CACHE_DATA_WIDTH`).
- All generic slave port data widths must be less than or equal to the internal cache data width (`C_Sx_AXI_GEN_DATA_WIDTH` ≤ `C_CACHE_DATA_WIDTH`).

- The master port data width must be greater then or equal to the internal cache data width (C_CACHE_DATA_WIDTH ≤ C_M_AXI_DATA_WIDTH).
- The internal cache line length must be greater than or equal to the corresponding cache line length of the AXI4 masters connected to the optimized port (C_CACHE_LINE_LENGTH ≥ C_Lx_CACHE_LINE_LENGTH).
- With optimized port cache coherency enabled, only 32-bit data is supported for all parts of the datapath.

  (C_Sx_AXI_DATA_WIDTH = C_CACHE_DATA_WIDTH = C_M_AXI_DATA_WIDTH = 32).
- With master port cache coherency, data widths are limited to 128 (C_Sx_AXI_DATA_WIDTH ≤ C_CACHE_DATA_WIDTH = C_M_AXI_DATA_WIDTH = 128).

System Cache Parameters

Table 136: **System Cache Parameters**

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_FAMILY[4]	FPGA Architecture	Supported architectures	virtex7	string
C_INSTANCE[4]	Instance Name	Any instance name	system_cache	string
C_FREQ[4]	System Cache clock frequency	Any valid frequency for the device	0	natural
C_BASEADDR[4]	Cacheable area base address		0xFFFF FFFF FFFF FFFF	std_logic_vector
C_HIGHADDR[4]	Cacheable area high address. Minimum size is 32KB		0x0000 0000 0000 0000	std_logic_vector
C_ENABLE_COHERENCY[1]	Enable implementation of cache coherent optimized ports	0, 1, 2	0	natural
C_ENABLE_SLAVE_COHERENCY	Enable implementation of cache coherent slave ports	0, 1	0	natural
C_ENABLE_MASTER_COHERENCY	Enable implementation of cache coherent master ports	0, 1, 2, 3	0	natural
C_ENABLE_ACE_PROTOCOL[1]	Enable ACE protocol	0, 1	0	natural
C_ENABLE_CCIX_PROTOCOL[1]	Enable CCIX protocol	0, 1	0	natural
C_ENABLE_CHI_PROTOCOL[1]	Enable CHI protocol	0, 1	0	natural
C_ENABLE_INTEGRITY	Enable Integrity	0, 1	0	natural
C_ENABLE_EXCLUSIVE	Enable implementation of exclusive monitor for non-coherent implementation	0, 1	0	natural
C_ENABLE_NON_SECURE	Enable distinction between Secure and Non-Secure transactions	0,1	0	natural

Table 136: **System Cache Parameters** *(cont'd)*

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_ENABLE_ERROR_HANDLING	Make RRESP with any error value drop an allocation attempt	0,1	0	natural
C_ENABLE_CTRL	Enable implementation of Statistics and Control function	0, 1	0	natural
C_ENABLE_STATISTICS	Bit mask for which statistics groups implemented when Control Interface is enabled: xxxx_xxx1 - Optimized Ports xxxx_xx1x - Generic Port xxxx_x1xx - Arbiter xxxx_1xxx - Access xxx1_xxxx - Lookup xx1x_xxxx - Update x1xx_xxxx - Backend 1xxx_xxxx - ATC	0-255	255	natural
C_ENABLE_VERSION_REGISTER	Level of Version Register to include: 0 - None 1 - Basic 2 - Full	0, 1, 2	0	natural
C_ENABLE_INTERRUPT	Enable control interface interrupt output	0, 1	0	natural
C_ENABLE_ADDRESS_TRANSLATION	Enable address translation cache: 0 - None 1 - ATS	0, 1	0	natural
C_NUM_OPTIMIZED_PORTS	Number of ports optimized for MicroBlaze processor cache connection	0-16[2]	1	natural
C_NUM_GENERIC_PORTS	Number of ports supporting full AXI4	0-16[3]	0	natural
C_NUM_MASTER_PORTS	Number of master ports	0, 1	1	natural
C_NUM_WAYS	Cache associativity	2, 4	2	natural
C_NUM_SLAVE_TRANSACTIONS[4]	Slave transactions in-flight per port	16, 32, 64, 128, 256	16	natural
C:_NUM_MASTER_TRANSACTIONS[4]	Master transactions in-flight	16, 32, 64, 128, 256	16	natural
C_NUM_SNOOP_TRANSACTIONS[4]	Snoop transactions in-flight	16, 32, 64, 128, 256	16	natural
C_NUM_OOO_CHANNELS[4]	Number of out-of-order channels	0 - 16	0 (AXI, ACE) 1 (CCIX, CHI)	natural
C_CACHE_DATA_WIDTH	Cache data width used internally. Automatically calculated to match AXI4 master interface	32, 64, 128, 256, 512	32	natural

Table 136: **System Cache Parameters** *(cont'd)*

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_CACHE_LINE_LENGTH	Cache line length. Constant value for ACE, CCIX. and CHI	16, 32, 64, 128, 256, 512, 1024	16	natural
C_CACHE_SIZE	Cache size in bytes	32768, 65536, 131072, 262144, 524288, 1048576, 2097152, 4194304	32768	natural
C_CACHE_TAG_MEMORY_TYPE	Cache tag memory type	0, 1, 2, 3	0	natural
C_CACHE_DATA_MEMORY_TYPE	Cache data memory type	0, 2, 3	0	natural
C_CACHE_LRU_MEMORY_TYPE	Cache Least Recently Used (LRU) memory type	0, 1, 2, 3	0	natural
C_Lx_CACHE_LINE_LENGTH[1]	Cache line length on masters connected to optimized ports. Automatically assigned with manual override	4, 8, 16	4	natural
C_Lx_CACHE_SIZE[1]	Cache size on masters connected to optimized ports. Automatically assigned with manual override	64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536	1024	natural
C_SUPPORT_SNOOP_FILTER	Support downstream snoop filter	0, 1	0	natural
C_SNOOP_KEEP_READ_ONCE	Keep allocation in cache after snoop of type ReadOnce (ACE) or SnpToAny (CCIX, CHI)	0, 1	0	natural
C_SNOOP_KEEP_READ_SHARED	Keep allocation in cache after snoop of type ReadShared (ACE) or SnpToS (CCIX, CHI)	0, 1	0	natural
C_SNOOP_KEEP_READ_CLEAN	Keep allocation in cache after snoop of type ReadClean (ACE)	0, 1	0	natural
C_SNOOP_KEEP_READ_NSD	Keep allocation in cache after ReadNotSharedDirty (ACE)	0, 1	0	natural
C_SNOOP_KEEP_CLEAN_SHARED	Keep allocation in cache after CleanShared (ACE) or SnpToC (CCIX, CHI)	0, 1	0	natural
C_SNOOP_KEEP_SNPTOSC	Keep allocation in cache after SnpToSC (CCIX)	0, 1	0	natural
C_SNOOP_PASS_READ_ONCE	Always pass write responsibility for snoop of type ReadOnce (ACE) or SnpToAny (CCIX, CHI)	0, 1	0	natural
C_SNOOP_PASS_READ_SHARED	Always pass write responsibility for snoop of type ReadShared (ACE) or SnpToS (CCIX, CHI)	0, 1	0	natural
C_SNOOP_PASS_READ_CLEAN	Always pass write responsibility for snoop of type ReadClean (ACE)	0, 1	0	natural

Table 136: **System Cache Parameters** *(cont'd)*

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_SNOOP_PASS_READ_NSD	Always pass write responsibility for snoop of type ReadNotSharedDirty (ACE)	0, 1	0	natural
C_DEFAULT_DOMAIN	Set default sharability domain for ACE Master Coherency: 0 - Inner 1 - Outer	0, 1	0	natural

Notes:

1. Automatically set. Not available in the Customize IP dialog box.
2. Optimized ports are not available with CCIX or CHI coherency protocol.
3. Generic ports are limited to 1 - 4 with CCIX or CHI coherency protocol.
4. Not available in the Customize IP dialog box.

MicroBlaze Processor Cache Optimized AXI4 Slave Interface Parameters

Table 137: **MicroBlaze Processor Cache Optimized AXI4 Slave Interface Parameters**

Parameter Name[1]	Feature/Description	Allowable Values	Default Value	VHDL Type
C_Sx_AXI_ADDR_WIDTH[2]	Address width	15-64	32	natural
C_Sx_AXI_DATA_WIDTH	Data width	32, 128, 256, 512	32	natural
C_Sx_AXI_RRESP_WIDTH[2]	Width of RRESP. Automatically assigned.	2, 4	2	natural
C_Sx_AXI_ID_WIDTH[2]	ID width. Automatically assigned.	1-32	1	natural
C_Sx_AXI_AWUSER_WIDTH[2]	Width of AWUSER. Automatically assigned.	1-64	1	natural
C_Sx_AXI_ARUSER_WIDTH[2]	Width of ARUSER. Automatically assigned.	1-64	1	natural
C_Sx_AXI_SUPPORT_UNIQUE[2]	Reserved	0	0	natural
C_Sx_AXI_SUPPORT_DIRTY[2]	Reserved	0	0	natural
C_Sx_AXI_FORCE_READ_ALLOCATE[2]	Force read transactions to use read allocate, also function as override value for other allocate on write channel	0, 1	0	natural
C_Sx_AXI_PROHIBIT_READ_ALLOCATE[2]	Prohibit read transactions from using read allocate, also function as override value for other allocate on write channel	0, 1	0	natural
C_Sx_AXI_FORCE_WRITE_ALLOCATE[2]	Force write transactions to use write allocate, also function as override value for other allocate on read channel	0, 1	0	natural

Table 137: **MicroBlaze Processor Cache Optimized AXI4 Slave Interface Parameters** *(cont'd)*

Parameter Name[1]	Feature/Description	Allowable Values	Default Value	VHDL Type
C_Sx_AXI_PROHIBIT_WRITE_ALLOCATE[2]	Prohibit write transactions from using write allocate, also function as override value for other allocate on read channel	0, 1	0	natural
C_Sx_AXI_FORCE_READ_BUFFER[2]	Force read transactions to use buffer bit	0, 1	0	natural
C_Sx_AXI_PROHIBIT_READ_BUFFER[2]	Prohibit read transactions from using buffer bit	0, 1	0	natural
C_Sx_AXI_FORCE_WRITE_BUFFER[2]	Force write transactions to use buffer bit	0, 1	0	natural
C_Sx_AXI_PROHIBIT_WRITE_BUFFER[2]	Prohibit write transactions from using buffer bit	0, 1	0	natural
C_Sx_AXI_ENABLE_ATOMIC[2]	Enable atomic sideband signals	0, 1	0	natural
C_OPTx_READ_RA_TYPE[2]	Read atomic default transaction: 3 - ReadUnique 4 - ReadClean 5 - ReadNotSharedDirty 6 - ReadShared	3, 4, 5, 6	6	natural
C_OPTx_READ_WA_RA_TYPE[2]	Write atomic + Read atomic default transaction: 3 - ReadUnique 4 - ReadClean 5 - ReadNotSharedDirty 6 - ReadShared	3, 4, 5, 6	3	natural
C_OPTx_READ_WA_TYPE[2]	Write atomic default transaction: 0 - ReadOnce 1 - ReadOnceCleanInvalid 2 - ReadOnceMakeInvalid	0, 1, 2	0	natural
C_OPTx_ENABLE_FULL_WRITE[2]	Enable write unique full	0, 1	0	natural
C_OPTx_ENABLE_ATC[2]	Enable address translation cache	0, 1	1	natural
C_OPTx_PROHIBIT_ATC_OVERRIDE[2]	Prohibit transaction type override	0, 1	1	natural
C_OPTx_ATC_SIZE[2]	Address translation cache size	1, 2, 4, 8	1	natural

Notes:

1. x = 0 - 15
2. Not available in the Customize IP dialog box.

Generic AXI4 Slave Interface Parameters

Table 138: **Generic AXI4 Slave Interface Parameters**

Parameter Name[1]	Feature/Description	Allowable Values	Default Value	VHDL Type
C_Sx_AXI_GEN_ADDR_WIDTH[2]	Address Width	15-64	32	natural

Table 138: **Generic AXI4 Slave Interface Parameters** *(cont'd)*

Parameter Name[1]	Feature/Description	Allowable Values	Default Value	VHDL Type
C_Sx_AXI_GEN_DATA_WIDTH	Data Width	32, 64, 128, 256, 512	32	natural
C_Sx_AXI_GEN_ID_WIDTH[2]	ID width. Automatically assigned.	1-32	1	natural
C_Sx_AXI_GEN_AWUSER_WIDTH[2]	Width of read address user bits. Automatically assigned.	1-64	1	natural
C_Sx_AXI_GEN_ARUSER_WIDTH[2]	Width of write address user bits. Automatically assigned.	1-64	1	natural
C_Sx_AXI_GEN_SUPPORT_UNIQUE[2]	Reserved	0	0	natural
C_Sx_AXI_GEN_SUPPORT_DIRTY[2]	Reserved	0	0	natural
C_Sx_AXI_GEN_FORCE_READ_ALLOCATE[2]	Force read transactions to use read allocate, also function as override value for other allocate on write channel	0, 1	0	natural
C_Sx_AXI_GEN_PROHIBIT_READ_ALLOCATE[2]	Prohibit read transactions from using read allocate, also function as override value for other allocate on write channel	0, 1	0	natural
C_Sx_AXI_GEN_FORCE_WRITE_ALLOCATE[2]	Force write transactions to use write allocate, also function as override value for other allocate on read channel	0, 1	0	natural
C_Sx_AXI_GEN_PROHIBIT_WRITE_ALLOCATE[2]	Prohibit write transactions from using write allocate, also function as override value for other allocate on read channel	0, 1	0	natural
C_Sx_AXI_GEN_FORCE_READ_BUFFER[2]	Force read transactions to use buffer bit	0, 1	0	natural
C_Sx_AXI_GEN_PROHIBIT_READ_BUFFER[2]	Prohibit read transactions from using buffer bit	0, 1	0	natural
C_Sx_AXI_GEN_FORCE_WRITE_BUFFER[2]	Force write transactions to use buffer bit	0, 1	0	natural
C_Sx_AXI_GEN_PROHIBIT_WRITE_BUFFER[2]	Prohibit write transactions from using buffer bit	0, 1	0	natural
C_Sx_AXI_GEN_ENABLE_ATOMIC[2]	Enable atomic sideband signals	0, 1	0	natural
C_GENx_READ_RA_TYPE[2]	Read atomic default transaction: 3 - ReadUnique 4 - ReadClean 5 - ReadNotSharedDirty 6 - ReadShared	3, 4, 5, 6	6	natural
C_GENx_READ_WA_RA_TYPE[2]	Write atomic + Read atomic default transaction: 3 - ReadUnique 4 - ReadClean 5 - ReadNotSharedDirty 6 - ReadShared	3, 4, 5, 6	3	natural

Table 138: **Generic AXI4 Slave Interface Parameters** *(cont'd)*

Parameter Name[1]	Feature/Description	Allowable Values	Default Value	VHDL Type
C_GENx_READ_WA_TYPE[2]	Write atomic default transaction: 0 - ReadOnce 1 - ReadOnceCleanInvalid 2 - ReadOnceMakeInvalid	0, 1, 2	0	natural
C_GENx_ENABLE_FULL_WRITE[2]	Enable write unique full	0, 1	0	natural
C_GENx_ENABLE_ATC[2]	Enable address translation cache	0, 1	1	natural
C_GENx_PROHIBIT_ATC_OVERRIDE[2]	Prohibit transaction type override	0, 1	1	natural
C_GENx_ATC_SIZE[2]	Address translation cache size	1, 2, 4, 8	1	natural

Notes:

1. x = 0 - 15
2. Not available in the Customize IP dialog box.

Statistics and Control AXI4-Lite Slave Interface Parameters

Table 139: **Statistics and Control AXI4-Lite Slave Interface Parameters**

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_S_AXI_CTRL_BASEADDR	Control area base address		0xFFFFFFFF	std_logic_ vector
C_S_AXI_CTRL_HIGHADDR	Control area high address. Minimum size is 128KB		0x00000000	std_logic_ vector
C_S_AXI_CTRL_ADDR_WIDTH	Address Width	17-64	32	natural
C_S_AXI_CTRL_DATA_WIDTH	Data Width	32, 64	32	natural

Memory Controller AXI4 Master Interface Parameters

Table 140: **Memory Controller AXI4 Master Interface Parameters**

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_M0_AXI_ADDR_WIDTH	Address Width. Constant value.	32	32	natural
C_M0_AXI_DATA_WIDTH	Data Width	32, 64, 128, 256, 512	32	natural
C_M0_AXI_THREAD_ID_WIDTH	ID width. Automatically assigned with manual override	1-32	1	natural
C_M0_AXI_RRESP_WIDTH	Width of RRESP. Automatically assigned.	2, 4	2	natural

CCIX, SAM, and CXS Interface

Table 141: **CCIX, SAM, and CXS Interface Parameters**

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_CXS0_DATA_FLIT_WIDTH	Flit Data Width	256, 512, 1024	256	natural
C_CXS0_RX_CREDIT_LIMIT[1]	Receiver credit limit	1 - 255	15	natural
C_CXS0_MAX_PKT_PER_FLIT[1]	Packets per flit	2, 3, 4	2	natural
C_CXS0_ERROR_FULL_DATA[1]	Full packet length violation error. Unused.	0, 1	0	natural
C_CXS0_CONTINUOUS_DATA[1]	Continuous data. Unused.	0, 1	0	natural
C_CXS0_REPLICATION[1]	Flow signal replication. Unused	NONE, DUPLICATE, TRIPLICATE	NONE	string
C_CXS0_DATACHECK[1]	Data & control integrity:	NONE, PARITY, SECDED	NONE	string
C_CCIX0_PKT_HEADER	CCIX Packet Header: 0 - Compatible 1 - Optimized	0, 1	0	natural
C_CCIX0_NO_MESSAGE_PACK	Do not allow CCIX message packing	0, 1	1	natural
C_CCIX0_NO_COMP_ACK[1]	No completion acknowledge	0, 1	0	natural
C_CCIX0_CACHE_LINE_SIZE[1]	Cache line size	64, 128	64	natural
C_CCIX0_ADDR_WIDTH[1]	Address width	48 - 64	48	natural
C_CCIX0_MAX_PACKET_SIZE[1]	Maximum packet size	128, 256, 512, 1024	128	natural
C_MSG_CHAIN_LENGTH[1]	Message chain length	1, 2, 3, 4	1	natural
C_COHERENT_ID_WIDTH[1]	Coherent ID width	6 - 11	6	natural
C_DYNAMIC_PORT_PROPERTY_OVERRIDE[1]	Dynamic change AXI override	0, 1	0	natural
C_DEFAULT_REQx_ID[1][2]	Requester ID	0	0	natural
C_NUM_SAM_ENTRIES	Number of SAM entries	1 - 32	1	natural
C_SAM_VALID	SAM entry valid	0, 1	0	natural
C_SAM_ID	SAM identifier	0 - 2047	0	natural
C_SAM_BASEADDR	SAM base address		0xFFFF FFFF FFFF FFFF	std_logic_vector
C_SAM_HIGHADDR	SAM high address or mask		0x0000 0000 0000 0000	std_logic_vector

Notes:

1. Not available in the Customize IP dialog box.
2. x = 1 to 4.

CHI and CHI Interface

Table 142: **CHI and CHI Interface Parameters**

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_ENABLE_CHI_DATACHECK_ERROR[1]	Enable CHI Data Check Error	0, 1	1	natural
C_ENABLE_CHI_POISON_ERROR[1]	Enable CHI Poison Error	0, 1	1	natural
C_M0_CHI_PROTOCOL[1]	M0_CHI Protocol	CHI, CHI-E	CHI	string
C_M0_CHI_ATOMIC_TRANSACTIONS[1]	M0_CHI Atomic Transactions	0, 1	0	natural
C_M0_CHI_CACHE_STASH_TRANSACTIONS[1]	M0_CHI Cache Stash Transactions	0	0	natural
C_M0_CHI_DIRECT_MEMORY_TRANSFER[1]	M0_CHI Direct Memory Transfer	0, 1	0	natural
C_M0_CHI_DATA_POISON[1]	M0_CHI Data Poison	0, 1	0	natural
C_M0_CHI_DATA_CHECK[1]	M0_CHI Data Check	FALSE, ODD_PARITY	FALSE	string
C_M0_CHI_CCF_WRAP_ORDER[1]	M0_CHI CCF Wrap Order	0, 1	0	natural
C_M0_CHI_REQ_ADDR_WIDTH[1]	M0_CHI REQ Address Width	44 - 52	48	natural
C_M0_CHI_NODEID_WIDTH[1]	M0_CHI Node ID Width	7 - 11	7	natural
C_M0_CHI_DATA_WIDTH	M0_CHI Data Width	128, 256, 512	128	natural
C_M0_CHI_BARRIER_TRANSACTIONS[1]	M0_CHI Barrier Transactions	0	0	natural
C_M0_CHI_ENHANCED_FEATURES[1]	M0_CHI Enhanced Features	0, 1	1	natural

Notes:

1. Not available in the Customize IP dialog box.

ATS Interface

Table 143: **ATS Interface Parameters**

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_ATC_SIZE[1]	Address Translation Table Size	1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384	256	natural
C_ATS0_DATA_WIDTH	Data Width	256, 512, 1024	256	natural

Table 143: **ATS Interface Parameters** *(cont'd)*

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_ATS0_CQ_CC_ALIGNMENT_MODE	CQ/CC alignment mode: 0 - DWORD Aligned 1 - Address Aligned	0, 1	0	natural
C_ATS0_RQ_RC_ALIGNMENT_MODE	RQ/RC alignment mode: 0 - DWORD Aligned 1 - Address Aligned	0, 1	0	natural
C_ATS0_CQ_CC_STRADDLE	CQ/CC frame straddle	0, 1	0	natural
C_ATS0_RQ_RC_STRADDLE	RC or RQ/RC frame straddle	0, 1	1	natural
C_ATS0_PASID_MODE	PASID mode	0, 1	0	natural
C_ATS0_PARITY	Parity: 0 - None 1 - Generate 2 - Generate Check	0, 1, 2	0	natural
C_ATS0_PRI_TIMEOUT_LIMIT	PRI timeout limit (ms)	0 - 8191	10	natural
C_ATS0_CQ_TUSER_WIDTH[1]	CQ TUSER width. Automatically assigned.	88, 108, 183, 229, 465	88	natural
C_ATS0_CC_TUSER_WIDTH[1]	CC TUSER width. Automatically assigned.	33, 81, 165	33	natural
C_ATS0_RQ_TUSER_WIDTH[1]	RQ TUSER width. Automatically assigned.	62, 85, 137, 183, 373	62	natural
C_ATS0_RC_TUSER_WIDTH[1]	RC TUSER width. Automatically assigned.	75, 161, 337	75	natural
C_ATS0_CQ_CC_ENABLE_AER[1]	Enable Completer Channels AER	0, 1	1	natural
C_ATS0_RQ_RC_ENABLE_AER[1]	Enable Requester Channels AER	0, 1	0	natural

Notes:

1. Not available in the Customize IP dialog box. Allowable values are technology dependent.

User Parameters

The following table shows the relationship between the fields in the Vivado® IDE and the User Parameters (which can be viewed in the Tcl Console).

Table 144: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE Parameter/Value[1]	User Parameter/Value[1]	Default Value
Number of Optimized AXI4 Ports	C_NUM_OPTIMIZED_PORTS	1
Number of Generic AXI4 Ports	C_NUM_GENERIC_PORTS	0
Enable Slave Coherency	C_ENABLE_SLAVE_COHERENCY	0
None	0	

Table 144: **Vivado IDE Parameter to User Parameter Relationship** *(cont'd)*

Vivado IDE Parameter/Value[1]	User Parameter/Value[1]	Default Value
ACE Coherency Protocol	1	
Enable Master Coherency	C_ENABLE_MASTER_COHERENCY	0
None	0	
ACE Coherency Protocol	1	
CCIX Coherency Protocol	2	
CHI Coherency Protocol	3	
Enable Integrity	C_ENABLE_INTEGRITY	0
Enable Exclusive Accesses	C_ENABLE_EXCLUSIVE	0
N-Way Set Associative	C_NUM_WAYS	2
Line Length	C_CACHE_LINE_LENGTH	16
Size	C_CACHE_SIZE	32k
32k	32768	
64k	65536	
128k	131072	
256k	262144	
512k	524288	
1M	1048576	
2M	2097152	
4M	4194304	
Enable Exclusive Access	C_ENABLE_EXCLUSIVE	0
Enable Non-Secure Access	C_ENABLE_NON_SECURE	0
Enable AXI Error Handling	C_ENABLE_ERROR_HANDLING	0
Tag RAM Type	C_CACHE_TAG_MEMORY_TYPE	0
Automatic	0	
LUTRAM	1	
BRAM	2	
URAM	3	
Data RAM Type	C_CACHE_DATA_MEMORY_TYPE	0
Automatic	0	
BRAM	2	
URAM	3	
LRU RAM Type	C_CACHE_LRU_MEMORY_TYPE	0
Automatic	0	
LUTRAM	1	
BRAM	2	
URAM	3	
M0_AXI Address Width	C_M0_AXI_ADDR_WIDTH	32
M0_AXI Data Width	C_M0_AXI_DATA_WIDTH	32

Table 144: **Vivado IDE Parameter to User Parameter Relationship** *(cont'd)*

Vivado IDE Parameter/Value[1]	User Parameter/Value[1]	Default Value
M0_AXI Thread ID Width	C_M0_AXI_THREAD_ID_WIDTH	1
CXS0 Flit Data Width	C_CXS0_FLIT_DATA_WIDTH	256
CXS0 Packet Header	C_CCIX0_PKT_HEADER	0
Compatible	0	
Optimized	1	
CXS0 No Message Packing	C_CCIX0_NO_MSG_PACK	1
Sx_AXI[2]	C_Sx_AXI_DATA_WIDTH[2]	32
Sx_AXI_GEN[2]	C_Sx_AXI_GEN_DATA_WIDTH[2]	32
M0_CHI Data Width	C_M0_CHI_DATA_WIDTH	512
Enable AXI Control Interface	C_ENABLE_CTRL	0
Select Statistics Groups	C_ENABLE_STATISTICS	0
Enable Version Registers	C_ENABLE_VERSION_REGISTER	None
None	0	
Basic	1	
Full	2	
Enable Interrupt Output	C_ENABLE_INTERRUPT	0
Support Downstream Snoop Filter	C_SUPPORT_SNOOP_FILTER	0
Keep After ReadOnce	C_SNOOP_KEEP_READ_ONCE	1
Keep After ReadShared	C_SNOOP_KEEP_READ_SHARED	0
Keep After ReadClean	C_SNOOP_KEEP_READ_CLEAN	0
Keep After ReadNotSharedDirty	C_SNOOP_KEEP_READ_NSD	0
Keep After CleanShared	C_SNOOP_KEEP_CLEAN_SHARED	0
Pass Dirty After ReadOnce	C_SNOOP_PASS_READ_ONCE	0
Pass Dirty After ReadShared	C_SNOOP_PASS_READ_SHARED	0
Pass Dirty After ReadClean	C_SNOOP_PASS_READ_CLEAN	0
Pass Dirty After ReadNotSharedDirty	C_SNOOP_PASS_READ_NSD	0
Default Sharability Domain	C_DEFAULT_DOMAIN	0
Inner	0	
Outer	1	
Keep After SnpToAny	C_SNOOP_KEEP_READ_ONCE	1
Keep After SnpToS	C_SNOOP_KEEP_READ_SHARED	0
Keep After SnpToC	C_SNOOP_KEEP_READ_CLEAN	0
Keep After SnpToSC	C_SNOOP_KEEP_SNPTOSC	0
Pass Dirty After SnpToAny	C_SNOOP_PASS_READ_ONCE	0
Pass Dirty After SnpToS	C_SNOOP_PASS_READ_SHARED	0
SnpOnce_Keep_Line	C_SNOOP_KEEP_READ_ONCE	1
SnpClean_Keep_Line	C_SNOOP_KEEP_READ_CLEAN	0
SnpShared_Keep_Line	C_SNOOP_KEEP_READ_SHARED	0

Table 144: **Vivado IDE Parameter to User Parameter Relationship** *(cont'd)*

Vivado IDE Parameter/Value[1]	User Parameter/Value[1]	Default Value
SnpNotSharedDirty_Keep_Line	C_SNOOP_KEEP_READ_NSD	0
SnpCleanShared_Keep_Line	C_SNOOP_KEEP_CLEAN_SHARED	0
SnpOnce_Pass_Dirty	C_SNOOP_PASS_READ_ONCE	0
SnpClean_Pass_Dirty	C_SNOOP_PASS_READ_CLEAN	0
SnpShared_Pass_Dirty	C_SNOOP_PASS_READ_SHARED	0
SnpNotSharedDirty_Pass_Dirty	C_SNOOP_PASS_READ_NSD	0
Number of SAM Entries	C_NUM_SAM_ENTRIES	1
Valid	C_SAMx_VALID[3]	0
ID	C_SAMx_ID[3]	0
Base Address	C_:SAMx_BASEADDR[3]	0xFFFFFFFFFFFFFFFF
High Address	C_SAMx_HIGHADDR[3]	0x0000000000000000
Enable Address Translation	C_ENABLE_ADDRESS_TRANSLATION	0
None	0	
ATS	1	
ATS0 Data Width	C_ATS0_DATA_WIDTH	256
ATS0 CQ and CC Alignment Mode	C_ATS0_CQ_CC_ALIGNMENT_MODE	0
DWORD Aligned	0	
Address Aligned	1	
ATS0 RQ and RC Alignment Mode	C_ATS0_RQ_RC_ALIGNMENT_MODE	0
DWORD Aligned	0	
Address Aligned	1	
ATS0 CQ and CC Frame Straddle[5]	C_ATS0_CQ_CC_STRADDLE	0
ATS0 RC Frame Straddle[4]	C_ATS0_RQ_RC_STRADDLE	1
ATS0 RQ and RC Frame Straddle[4]	C_ATS0_RQ_RC_STRADDLE	1
ATS0 PASID Mode	C_ATS0_PASID_MODE	0
ATS0 Parity	C_ATS0_PARITY	0
DISABLE	0	
GENERATE	1	
GENERATE CHECK	2	
ATS0 PRI Timeout Limit (ms)	C_ATS0_PRI_TIMEOUT_LIMIT	10

Notes:

1. Parameter values are listed in the table where the Vivado IDE parameter value differs from the user parameter value. Such values are shown in this table as indented below the associated parameter.
2. x = 0 to 15.
3. x = 0 to 3.
4. ATS0 RQ and RC Frame Straddle is used for 512-bit and 1024-bit (Versal Premium) data widths. ATS0 RC Frame Straddle is used for 256-bit data width.
5. ATS0 CQ and CC Frame Straddle is allowed for 512-bit and 1024 bit (Versal Premium) data widths, but is disabled by default.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896).

Constraining the Core

Required Constraints

This section is not applicable for this IP core.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

This section is not applicable for this IP core.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

Simulation

For comprehensive information about Vivado® simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900).

IMPORTANT! *For cores targeting 7 series or Zynq-7000 devices, UNIFAST libraries are not supported. Xilinx IP is tested and qualified with UNISIM libraries only.*

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896).

Appendix A

Upgrading

This appendix contains information about migrating a design from ISE® to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details are provided (where applicable) about any port changes and other impact to user logic are included.

Migrating to the Vivado Design Suite

For information on migrating to the Vivado® Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911).

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado® Design Suite.

Functionality Changes

Version 1.01a, 1.01b and 1.01c

The following describes changes between the ISE version of the core and the current version:

- The supported cache sizes has been increased to support 256 KB and 512 KB.
- Hit and miss statistics are changed to be on a per port basis instead of total.
- Added support for simultaneously ongoing read for each channel.

Version 2.00a and 3.0

- Added cache coherency support on optimized ports.
- Added statistics related to cache coherency.
- Added optional exclusive monitor for non-coherent cache implementation.
- Added version and cache maintenance registers.

Version 3.1

- Increased number of optimized ports from 8 to 16.
- Increased number of generic ports from 1 to 16.
- Added support for 16 word cacheline when coherency is enabled.
- Added support for up to 64-bit wide addresses.

Version 4.0

- Added cache coherency support for Master port
- Added optional support for Non-Secure transactions
- Added optional AXI4 error processing

Version 5.0

2019.2 Release

- Increased available cache sizes
- Increased available cache line lengths
- Added fine-grained control over RAM utilization
- Changed Version Register fields

2020.1 Release

- Added support for CCIX coherency protocol
- Added support for Atomic transactions
- Added ATS support for address translation
- Added support for cache maintenance transactions on slave ports
- Added fine-grained control of AXI to coherency translations
- Added fine-grained control of snoop behavior

- Added ECC and parity protection for cache memory when using CCIX protocol

2020.2 Release

- Added support for CHI coherency protocol

2021.1 Release

- Added support for PCIe 10bits TAG, requires Host Capabilities support and announcement for activation
- Added support for Versal FunctionID/ComponetID to supersede EP BDF for self identification (None/Partial/Full)
- Updated Port Link Tx/Rx Status with CCIX/CHI context definition and CXS/CHI Link status signal inspection

2021.2 Release

- Added support for CCIX coherency protocol with CPM (Versal Premium)
  - Extended CXS to support 1024-bit data width
  - Added Host TA propagated PASID support for CCIX
- Extended ATS AXI4-Stream to support 1024-bit data width (Versal Premium)
  - Added ATS AXIS CQ/CC/RQ/RC Sideband protocol for 1024 data width
  - Extended ATS AXIS CQ/CC/RQ Straddle mode, up to 4 Headers per beat
  - Extended ATS AXIS RC Straddle mode, up to 8 Headers per beat

Port and Parameter Changes

Version 3.1

- Increased number of optimized ports from 8 to 16
- Increased number of generic ports from 1 to 16

Version 4.0

- Renamed C_NUM_SETS to C_NUM_WAYS
- Renamed M_AXI interface and signals to M0_AXI
- Added ACE extra signals and channels to M0_AXI as well as providing the M0_ACE interface

- Added AXI ARCACHE/AWCACHE override parameters to each port

Version 5.0

2019.2 Release

- Added AXI user signals reserved for future us
- Added Initializing output signal

2020.1 Release

- Changed default value of C_Sx_AXI_PROHIBIT_WRITE_ALLOCATE to 0
- Changed default value of C_Sx_AXI_GEN_PROHIBIT_WRITE_ALLOCATE to 0
- Added CXS interface for CCIX coherency, and related parameters
- Added AXI4-Stream interfaces for ATS address translation, and related parameters
- Added interrupt output signal

2020.2 Release

- Added CHI interface for CHI coherency, and related parameters

Appendix B

Debugging

This appendix includes details about resources available on the Xilinx® Support website and debugging tools.

If the IP requires a license key, the key must be verified. The Vivado® design tools have several license checkpoints for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with an error. License checkpoints are enforced by the following tools:

- Vivado Synthesis
- Vivado Implementation
- write_bitstream (Tcl command)

IMPORTANT! *IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.*

Finding Help on Xilinx.com

To help in the design and debug process when using the core, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support. The Xilinx Community Forums are also available where members can learn, participate, share, and ask questions about Xilinx solutions.

Documentation

This product guide is the main document associated with the core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx® Documentation Navigator. Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use keywords such as:

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the System Cache Core

AR 54452.

Technical Support

Xilinx provides technical support on the Xilinx Community Forums for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To ask questions, navigate to the Xilinx Community Forums.

Debug Tools

There are many tools available to address System Cache core design issues. It is important to know which tools are useful for debugging various situations.

Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx® devices.

The Vivado logic analyzer is used to interact with the logic debug LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908).

Reference Boards

All Xilinx® development boards for 7 series, UltraScale™, UltraScale+™ and Versal ACAP devices support the System Cache core. These boards can be used to prototype designs and establish that the core can communicate with the system.

Simulation Debug

The simulation debug flow for Mentor Graphics Questa Advanced Simulator is described below. A similar approach can be used with other simulators.

- Check for the latest supported versions of Questa Advanced Simulator in the Xilinx® Design Tools: Release Notes Guide. Is this version being used? If not, update to this version.
- If using Verilog, do you have a mixed mode simulation license? If not, obtain a mixed-mode license.
- Ensure that the proper libraries are compiled and mapped. In the Vivado® Design Suite this is done within the tool using **Flow → Simulation Settings**.
- Have you associated the intended software program for all connected MicroBlaze™ processors with the simulation? Use **Tools → Associate ELF Files** in the Vivado® Design Suite to do this.
- When observing the traffic on any of the AXI4 interfaces connected to the System Cache core, see the *AMBA® AXI and ACE Protocol Specification* for the AXI4 timing.

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The Vivado® debug feature is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the debug feature for debugging the specific problems.

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the `locked` port.

Interface Debug

Optimized AXI4 Interfaces

Only the number of ports specified by `C_NUM_OPTIMIZED_PORTS` are available. There are no registers to read, but basic functionality is tested by writing data and then reading it back. Output `S<x>_AXI_AWREADY` asserts when the write address is used, `S<x>_AXI_WREADY` asserts when the write data is used, and output `S<x>_AXI_BVALID` asserts when the write response is valid. Output `S<x>_AXI_ARREADY` asserts when the read address is used, and output `S<x>_AXI_RVALID` asserts when the read data/response is valid. If the interface is unresponsive, ensure that the following conditions are met:

- The `ACLK` input is connected and toggling.
- The interface is not being held in reset, and `ARESETN` is an active-Low reset.
- Ensure the accessed Optimized port is activated.
- If the simulation has been run, verify in simulation and/or a Vivado® debugging tool capture that the waveform is correct for accessing the AXI4 interface.

Generic AXI4 Interfaces

Only the number of ports specified by C_NUM_GENERIC_PORTS are available. There are no registers to read, but basic functionality is tested by writing data and then reading it back. Output S<x>_AXI_GEN_AWREADY asserts when the write address is used, S<x>_AXI_GEN_WREADY asserts when the write data is used, and output S<x>_AXI_GEN_BVALID asserts when the write response is valid. Output S<x>_AXI_GEN_ARREADY asserts when the read address is used, and output S<x>_AXI_GEN_RVALID asserts when the read data/response is valid. If the interface is unresponsive, ensure that the following conditions are met:

- The ACLK input is connected and toggling.
- The interface is not being held in reset, and ARESETN is an active-Low reset.
- Ensure the accessed generic port is activated.
- If the simulation has been run, verify in simulation and/or a Vivado® debugging tool capture that the waveform is correct for accessing the AXI4 interface.

AXI4-Lite Control Interface

The AXI4-Lite interface is only available when the Control interface is enabled with C_ENABLE_CTRL. Read from a register that does not have all 0s as a default to verify that the interface is functional. Output S_AXI_CTRL_ARREADY asserts when the read address is used, and output S_AXI_CTRL_RVALID asserts when the read data/response is valid. If the interface is unresponsive, ensure that the following conditions are met:

- The ACLK input is connected and toggling.
- The interface is not being held in reset, and ARESETN is an active-Low reset.
- If the simulation has been run, verify in simulation and/or a Vivado® debugging tool capture that the waveform is correct for accessing the AXI4 interface.

CXS Interfaces

The outbound TLP packets on CXS contain CCIX messages that are events based on what has occurred on any of the AXI slave ports, or as a response to the CXS inbound messages. The first are primarily requests and the second are snoop responses or CompAck.

The CXS links work the same independent of direction: CXS0_VALID_TX/CXS0_VALID_RX is asserted when valid information is available on CXS0_DATA_TX/CXS0_DATA_RX. Framing of data is defined in CXS0_CNTL_TX/CXS0_CNTL_RX, for example start and stop of packets.

Since CXS is credit based, there are no ready signals to indicate when a valid FLIT is consumed, it is implicitly only valid for a single clock cycle. To distribute credits the receiver side, which is the owner of the credits, asserts `CXS0_CRDGNT_TX/CXS0_CRDGNT_RX` one cycle per credit granted to the transmit side. This mean that credits need to be distributed before any traffic can occur. The credit distribution is usually initiated when a CXS link is activated with a request/acknowledge handshake, such as `CXS0_ACTIVE_REQ_TX/CXS0_ACTIVE_ACK_TX`.

For more detailed information of CXS framing see Arm AMBA CXS Protocol Specification, ARM IHI 0079A , and for the actual embedded messages see CCIX Base Specification Revision 1.1 Version 1.0.

CHI Interface

In the CHI case, traffic is transmitted on six channels that are grouped into Tx and Rx links with three channels each. Some additional signals are also used to handle link status and system coherency.

Active FLITs on all channels are qualified with a valid signal, for example `M0_CHI_TXREQFLITV` for `M0_CHI_TXREQFLIT` in case of the request channel. Channels receive the credits with a dedicated signal, which is called `M0_CHI_TXREQLCRDV` in the request example.

Before debugging a channel, verify that the corresponding Link is in RUN state, i.e. `M0_CHI_TXLINKACTIVEREQ` and `M0_CHI_TXLINKACTIVEACK` are set to 1.

For coherent traffic the system coherency should be in RUN state with both `M0_CHI_SYSCOREQ` and `M0_CHI_SYSCOACK` set to 1.

For more details on CHI behavior and FLIT bit mapping, see Arm® *AMBA 5 CHI Architecture Specification* (ARM IHI 0050B).

AXI4-Stream Interfaces

These interfaces are only available when ATS is enabled with `C_ENABLE_ADDRESS_TRANSLATION`.

AXI4-Stream Slaves (CQ/RC):

The input `ATS0_S_AXIS_xx_VALID` is asserted when address, data/response, and user channel input `ATS0_S_AXIS_xx_TUSER` are valid.

Normally, the input `ATS0_S_AXIS_xx_TLAST` is asserted when the last data/response in a beat is valid, simultaneously with `ATS0_S_AXIS_xx_VALID`.

However, when the AXI4-Stream straddle option is enabled, the TLP to decode consist of start/end pointer(s) in the `ATS0_S_AXIS_xx_TUSER` user channel input, indicating when the last data in a beat is valid, and the input `ATS0_S_AXIS_xx_TLAST` is inactive.

The interface response output ATS0_S_AXIS_xx_READY is asserted when the data/response is accepted. If the System Cache is not ready, the master side should wait while keeping all inputs unchanged.

The user channel input ATS0_S_AXIS_xx_TUSER encodes information regarding framing, byte alignment, parity, and the start/end pointers mentioned above. Decoding depends on the data width and PASID mode, see *Versal ACAP Integrated Block for PCI Express LogiCORE IP Product Guide* (PG343) for details.

AXI4-Stream Masters: (CC/RQ):

The output ATS0_M_AXIS_xx_VALID is asserted when address, data/response, and user channel output ATS0_M_AXIS_xx_TUSER are valid.

Normally, the output ATS0_M_AXIS_xx_TLAST is asserted when the last data/response in a beat is valid, simultaneously with ATS0_M_AXIS_xx_VALID.

However, when the AXI4-Stream straddle option is enabled, the TLP to encode consist of start/end pointer(s) in the ATS0_M_AXIS_xx_TUSER user channel output, indicating when the last data in a beat is valid, and the output ATS0_M_AXIS_xx_TLAST is inactive.

The interface response input ATS0_M_AXIS_xx_READY should be asserted when the data/response is accepted. If the slave side is not ready, System Cache waits while keeping all outputs unchanged.

The user channel output ATS0_M_AXIS_xx_TUSER encodes information regarding framing, byte alignment, parity, and the start/end pointers mentioned above. Encoding depends on the data width and PASID mode, see *Versal ACAP Integrated Block for PCI Express LogiCORE IP Product Guide* (PG343) for details.

If the interface is unresponsive, ensure that the following conditions are met:

- The ACLK input is connected and toggling.
- The interface is not being held in reset, and ARESETN is an active-Low reset.
- If the simulation has been run, verify in simulation and/or a Vivado® debugging tool capture that the waveform is correct for accessing the AXI4-Stream interface.

XILINX

Appendix C

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the Vivado® IDE, select **Help → Documentation and Tutorials**.
- On Windows, select **Start → All Programs → Xilinx Design Tools → DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on DocNav, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this guide:

1. *AMBA® AXI and ACE Protocol Specification* (ARM IHI 0022E)
2. *AMBA 4 AXI4-Stream Protocol Specification* (ARM IHI 0051A)
3. *AMBA CXS Protocol Specification* (ARM IHI 0079A)
4. *Cache Coherent Interconnect for Accelerators* (CCIX Base Specification Revision 1.1 Version 1.0)
5. *PCI Express®* (PCI Express Base Specification Revision 5.0 Version 1.0)
6. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
7. *Vivado Design Suite User Guide: Designing with IP* (UG896)
8. *Vivado Design Suite User Guide: Getting Started* (UG910)
9. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
10. *ISE to Vivado Design Suite Migration Guide* (UG911)
11. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
12. *Vivado Design Suite User Guide: Design Analysis and Closure Techniques* (UG906)
13. *UltraFast Design Methodology Timing Closure Quick Reference Guide* (UG1292)
14. *MicroBlaze Processor Reference Guide* (UG984)
15. *UltraScale+ Devices Integrated Block for PCI Express LogiCORE IP Product Guide* (PG213)
16. *AMBA 5 CHI Architecture Specification* (ARM IHI 0050B)
17. *Versal ACAP Integrated Block for PCI Express LogiCORE IP Product Guide* (PG343)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
11/05/2021 Version 5.0	
General Updates	• Added support for CCIX CPM mode, Versal Premium, CXS (data width 256, 512, and 1024 bits) • Updated CCIX PASID support, Host TA PASID propagation, Versal Premium • Extended support for ATS AXIS (data width 1024 bits), Versal Premium

Section	Revision Summary
08/06/2021 Version 5.0	
General Updates	• Added support for PCIe 10bits TAG, requires Host Capabilities support and announcement for activation • Added support for Versal FunctionID/ComponetID to supersede EP BDF for self identification (None/Partial/Full) • Updated Port Link Tx/Rx Status with CCIX/CHI context definition and CXS/CHI Link status signal inspection
12/11/2020 Version 5.0	
General Updates	• Added optional support for CHI • Added optional support for ATS PASID
07/08/2020 Version 5.0	
General Updates	• Added optional support for CCIX • Added optional support for ATS • Added Interrupt port • Added support for snoop effect configuration • Added optional support for integrity in the CCIX case
10/30/2019 Version 5.0	
General Updates	• Increased supported cache size • Added fine-grained control over RAM utilization • Added Initialization port • Increased supported cache line lengths
04/05/2017 Version 4.0	
Property Translation	Added section for AXI transaction properties and translation
10/05/2016 Version 4.0	
General Updates	• Added optional support for master port cache coherency • Added optional support of Non-Secure transactions • Added optional support for AXI4 error handling • Added per slave port allocate and buffer override functionality • Updated Xilinx
11/18/2015 Version 3.1	
General Updates	Added support for UltraScale+™ families
06/24/2015 Version 3.1	
General Updates	Moved performance and resource utilization data to the web
04/01/2015 Version 3.1	
General Updates	• Increased optimized ports to 16 • Increased generic ports to 16 • Added support for 16 word cache line when coherent • Resource tables updated • Support of up to 64-bit address width

Section	Revision Summary
10/02/2013 Version 3.0	
General Updates	• Revision number advanced to 3.0 to align with core version number. • Description of new reset behavior • Resource tables updated • Updated latency number for write transactions
03/20/2013 Version 1.0	
Initial release.	Replaces PG031. No documentation changes for this release.

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING

OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright

© Copyright 2013-2021 Xilinx, Inc. Xilinx, the Xilinx logo, Alveo, Artix, Kintex, Kria, Spartan, Versal, Vitis, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. PCI, PCIe, and PCI Express are trademarks of PCI-SIG and used under license. AMBA, AMBA Designer, Arm, ARM1176JZ-S, CoreSight, Cortex, PrimeCell, Mali, and MPCore are trademarks of Arm Limited in the EU and other countries. All other trademarks are the property of their respective owners.