System Cache v4.0

LogiCORE IP Product Guide

Vivado Design Suite

PG118 April 5, 2017

XILINX
ALL PROGRAMMABLE™

Table of Contents

Appendix B: Debugging

Appendix C: Additional Resources and Legal Notices

Introduction

The LogiCORE™ System Cache IP core provides system level caching capability to an AMBA® AXI4 system. The System Cache core resides in front of the external memory controller and is seen as a Level 2 cache from the MicroBlaze™ I and D caches. It can also be used to connect fabric accelerators to the UltraScale+ MPSoC ACE port.

Features

- Dedicated AXI4 slave ports for a MicroBlaze processor
- Connects up to 16 MicroBlaze processor cache ports, normally eight processors
- 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 512kB in size
- Optional AXI4-Lite Statistics and Control port
- Supports up to 64 bit AXI4 address width

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	UltraScale+™ UltraScale™ Zynq®-7000 All Programmable SoC Zynq® UltraScale+™ MPSoC 7 series
Supported User Interfaces	AXI4, ACE, AXI4-Lite
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	Vivado: RTL
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Not Provided
Simulation Model	Not Provided
Supported S/W Driver	N/A
Tested Design Tools(2)	
Design Entry Tools	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide
Synthesis Tools	Vivado Synthesis
Support	
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For a complete list of supported derivative 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 IP core can be added to an AXI4 system to improve overall system computing performance, for accesses to external memory. There are two principal configurations where the System Cache is used. The first configuration is a MicroBlaze processor-based system where it serves as an L2 cache for up to eight fully connected processors or up to 16 individual cache ports. The second configuration is as a coherent cache for accelerators connected to a Zynq® UltraScale+™ MPSoC.

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.

Applications

An Ethernet communication system example is shown in Figure 1-1. 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 AXI4 interconnect, because the DMA controller has three AXI4 master ports. The DMA in turn is connected to the Ethernet IP core using an AXI4-Stream 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 example throughout the documentation.

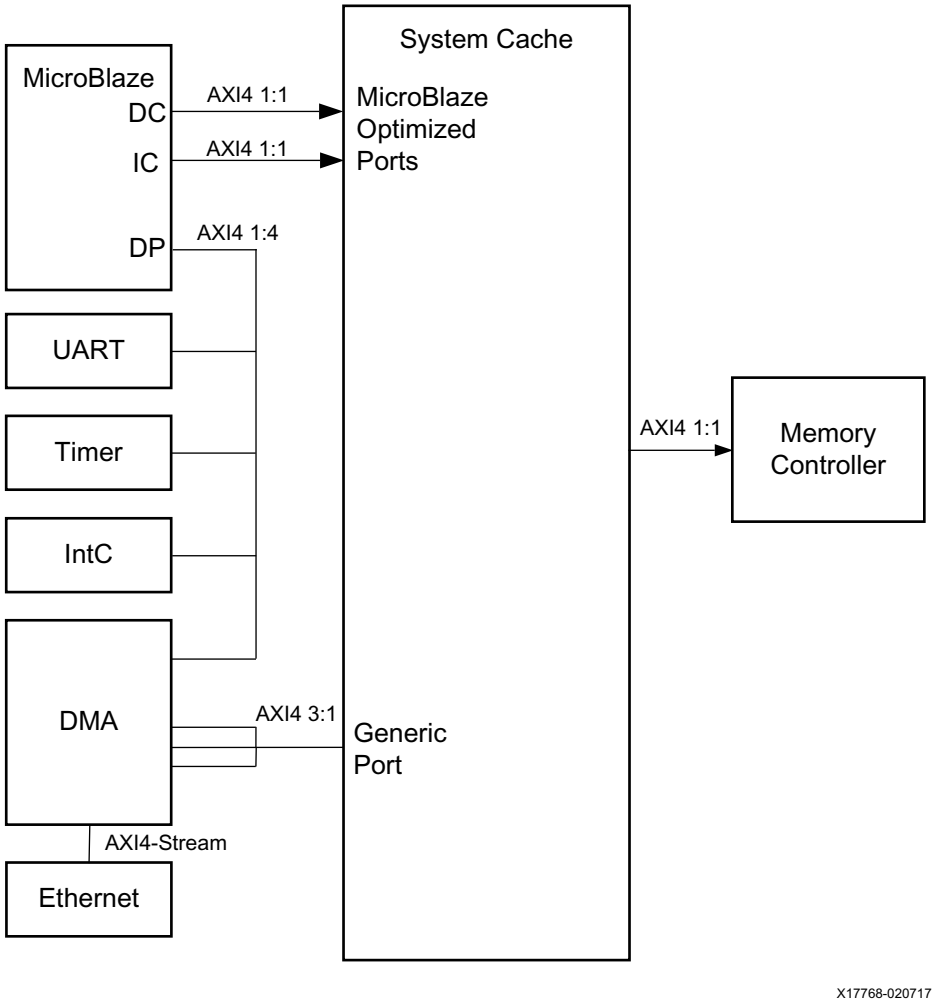

Figure 1-1: **Ethernet System**

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 Table 1-1 and Table 1-2, respectively.

Table 1-1: **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 1-2: **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

Another example use case, shown in Figure 1-2, 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.

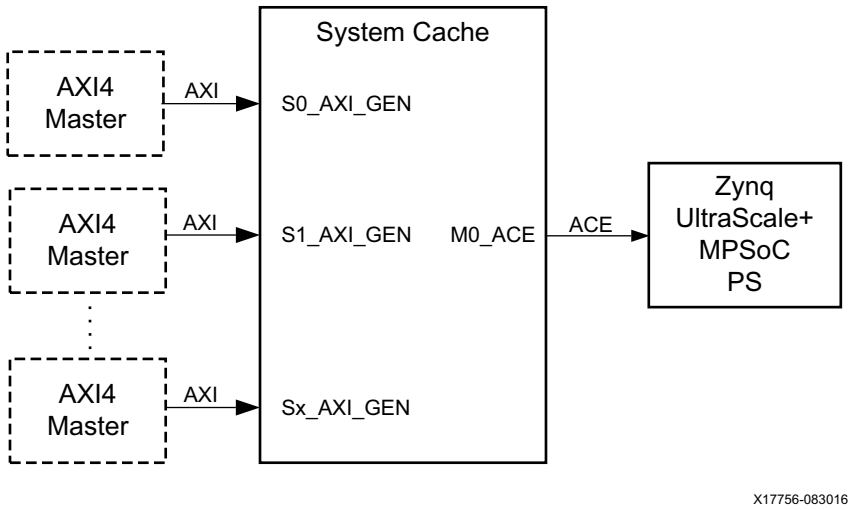

Figure 1-2: **Example of a System With Two or More Accelerators**

It is possible to connect one or more MicroBlaze processors 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 Table 1-3. Sx_AXI_GEN_* should be configured for all active ports.

Table 1-3: **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

Table 1-3: **Example System Cache Parameters for Accelerator Configuration *(Cont'd)***

Parameter	Value
C_Sx_AXI_GEN_PROHIBIT_READ_BUFFER	0
C_Sx_AXI_GEN_FORCE_WRITE_BUFFER	1
C_Sx_AXI_GEN_PROHIBIT_WRITE_BUFFER	0

For backwards compatibility all the C_Sx_AXI_PROHIBIT_WRITE_ALLOCATE/ C_Sx_AXI_GEN_PROHIBIT_WRITE_ALLOCATE parameters are set by default and need to be cleared to enable allocation on Write Miss.

Unsupported Features

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.

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.

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 this and 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

In a typical system with one MicroBlaze processor, as shown in Figure 2-1 and Figure 2-2, 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 (Figure 2-1) or directly (Figure 2-2).

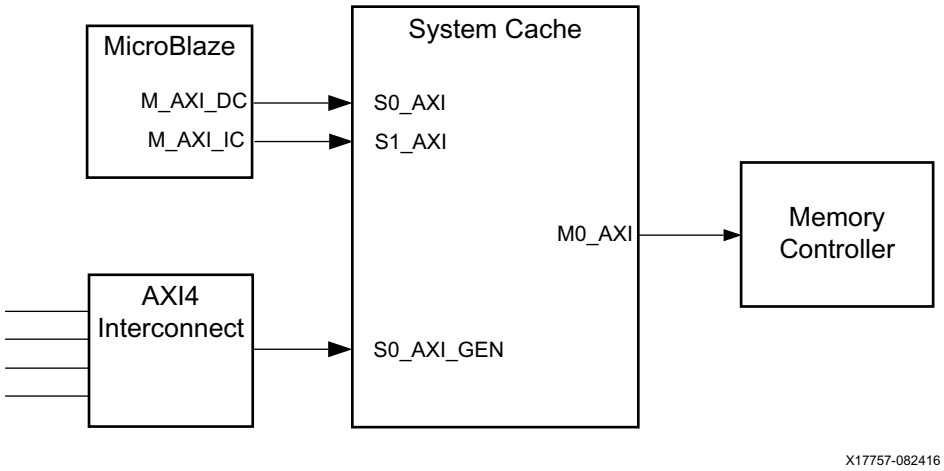

Figure 2-1: **Typical System With a Single Processor**

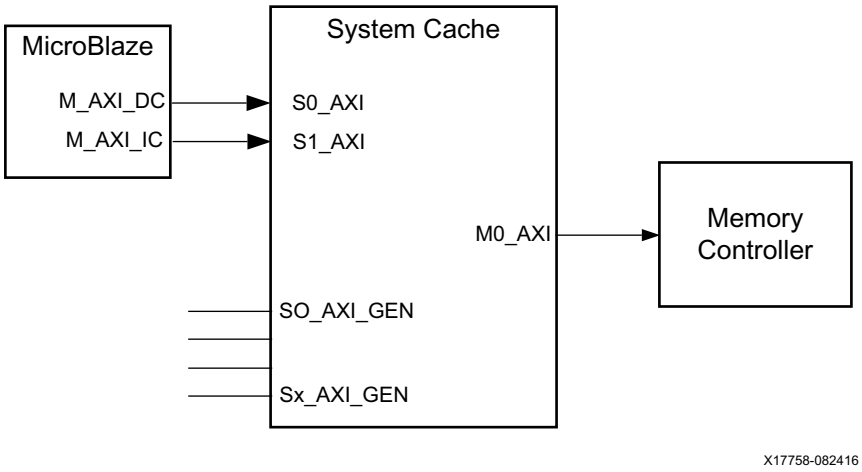

Figure 2-2: **Typical System With a Single Processor**

The System Cache core can also be used in a system without any MicroBlaze processor, as shown in Figure 2-3. For this case, it is also possible to have multiple generic ports.

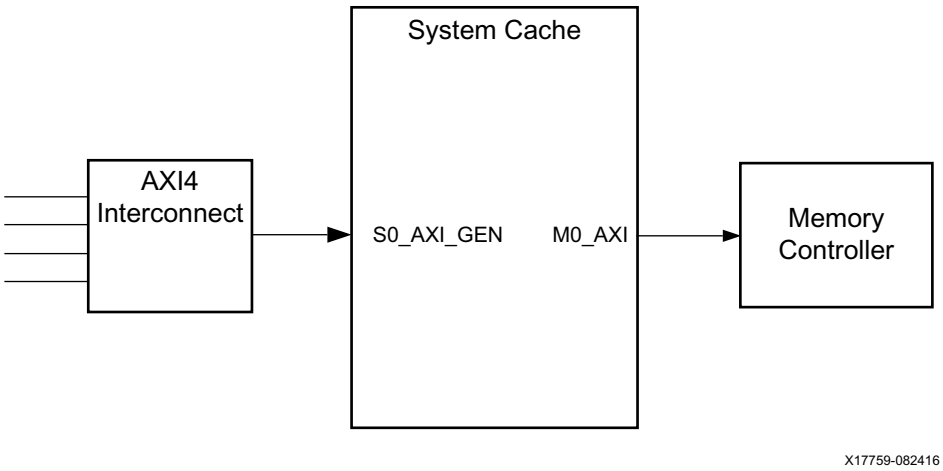

Figure 2-3: **System Without Processor**

The System Cache core has 16 cache interfaces optimized for MicroBlaze, enabling direct connection of up to eight MicroBlaze processors, as shown in Figure 2-4.

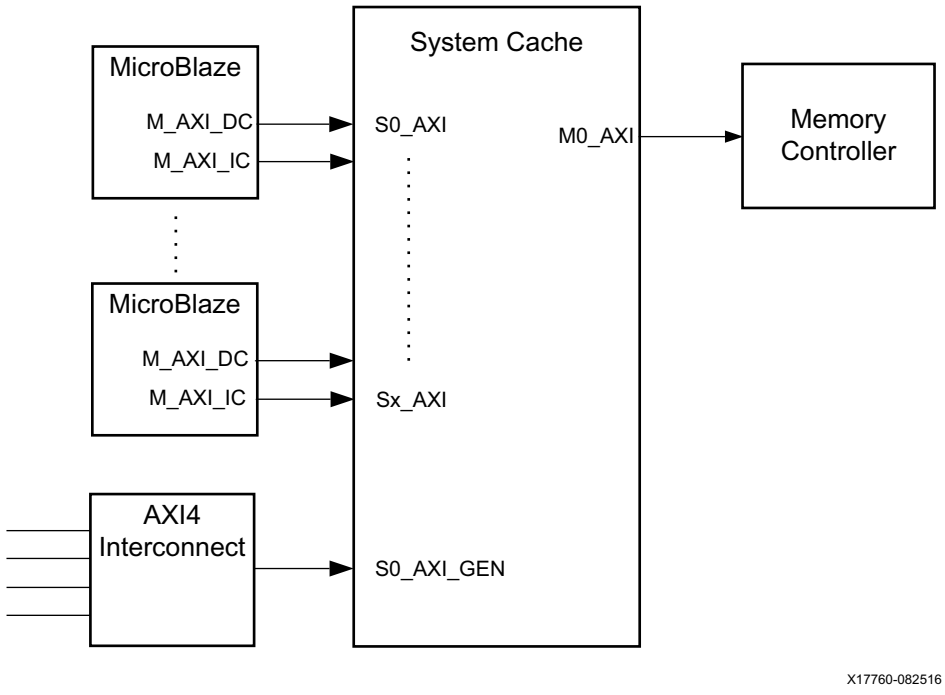

Figure 2-4: **Typical System With Multiple MicroBlaze Processors**

The 16 cache interfaces optimized for MicroBlaze provide support for up to eight cache coherent MicroBlaze processors, as shown in Figure 2-5.

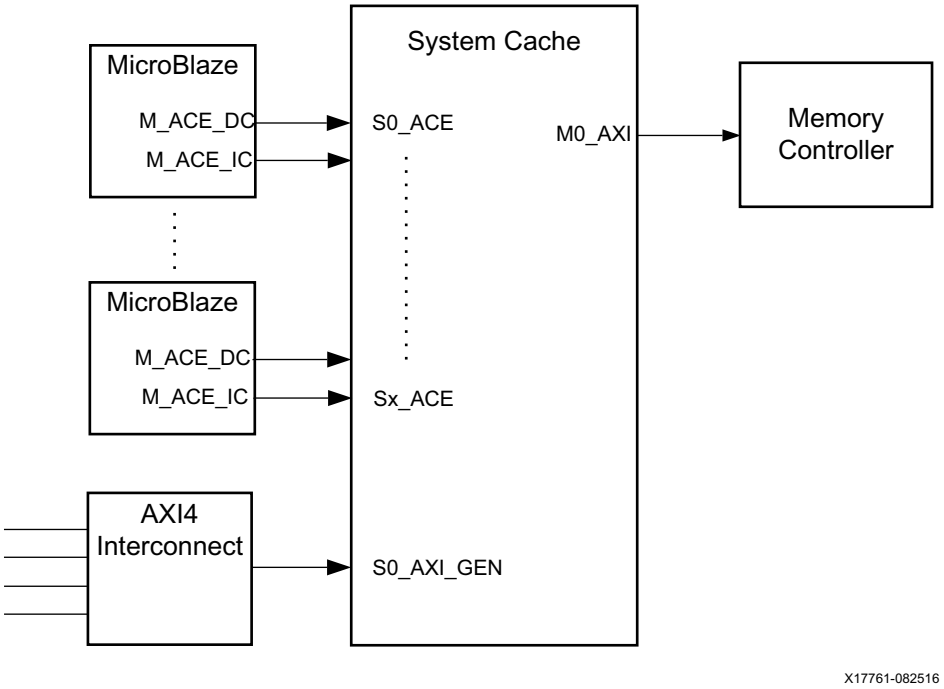

Figure 2-5: **Typical System With Multiple Coherent MicroBlaze Processors**

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 Figure 2-6.

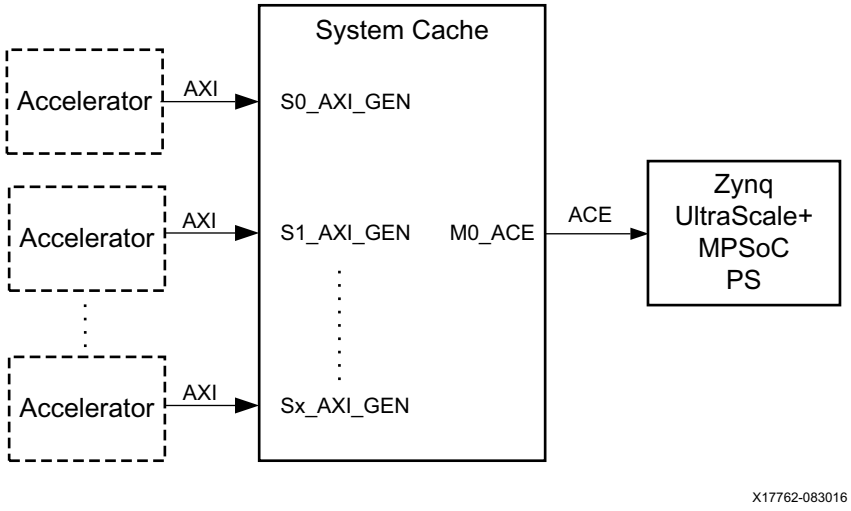

Figure 2-6: **Typical System With Multiple Accelerators**

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* [Ref 1]). 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.

Master Port Cache Coherency

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).

Exclusive Monitor

The optional exclusive monitor provide 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 Table 4-5. The selected size is a trade-off between performance and resource usage, in particular the number of block RAMs.

The associativity can be configured with the parameter C_NUM_WAYS according to Table 4-5. Increased associativity generally provides better hit rate, which gives better performance but requires more area resources.

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

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.

Statistics and Control

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
- Arbitration statistics
- Functional unit statistics
  - Stall cycles
  - Internal queue usage
- Port statistics for the master interface
  - Read and write latency

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

For details on the registers used to read statistics and control how statistics is gathered, see Register Space.

Standards

The System Cache core adheres to the AMBA® AXI4 and ACE interface standard (see *ARM® AMBA AXI Protocol Specification, Version 2.0 ARM IHI 0022E* [Ref 1]).

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.

Maximum Frequencies

For details about performance, visit Performance and Resource Utilization.

Cache Latency

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.

The latency depends on many factors such as traffic from other ports and conflict with earlier transactions. The numbers in Table 2-1 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 2-1: **System Cache Core Latencies for Optimized Port**

Type	Optimized Port Latency
Read Hit	5
Read Miss	6 + latency added by memory subsystem
Read Miss Dirty	Maximum of: 6 + latency added by memory subsystem 6 + latency added for evicting dirty data (cache line length * 32 / M_AXI Data Width)

Table 2-1: **System Cache Core Latencies for Optimized Port *(Cont'd)***

Type	Optimized Port Latency
Write Hit	2 + burst length
Write Miss	Non-bufferable transaction: 6 + 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 Table 2-2 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 2-2: **System Cache Core Latencies for Cache Coherent Optimized Port**

Type	Coherent Optimized Port Latency
DVM Message	8 + latency added by snooped masters
DVM Sync	11 + latency added by snooped masters
Read Hit	8 + latency added by snooped masters
Read Miss	9 + latency added by snooped masters + latency added by memory subsystem
Read Miss Dirty	Maximum of: 9 + latency added by snooped masters + latency added by memory subsystem 9 + latency added by snooped masters + latency added for evicting dirty data (cache line length * 32 / M_AXI Data Width)
Write Hit	Maximum of: 2 + burst length 5 + latency added by snooped masters
Write Miss	Non-bufferable transaction: 9 + 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 Table 2-3 for latency values.

Table 2-3: **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 application vary depending on access patterns, hit/miss ratio and other factors. Example values from a system (see Figure 2-1) running the iperf network testing tool with the LWIP TCP/IP stack in raw mode are shown in Tables 2-4 to 2-7. Table 2-4 contains the hit rate for transactions from all ports. Table 2-5, Table 2-6 and Table 2-7 show per port hit rate and latencies for the three active ports.

Table 2-4: **Application Total Hit Rates**

Type	Hit Rate
Read	99.82%
Write	92.93%

Table 2-5: **System Cache Hit Rate and Latencies for MicroBlaze D-Side Port**

Type	Hit Rate	Min	Max	Average	Standard Deviation
Read	99.68%	5	289	7	3
Write	96.63%	3	30	3	1

Table 2-6: **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 2-7: **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 Utilization

The System Cache core uses instantiated BRAM; how many varies with configuration. Distributed RAM is inferred by the RTL code and is also dependent on the configuration. For details about resource utilization, visit Performance and Resource Utilization.

Port Descriptions

The block diagram for the System Cache core is shown in Figure 2-7. 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.

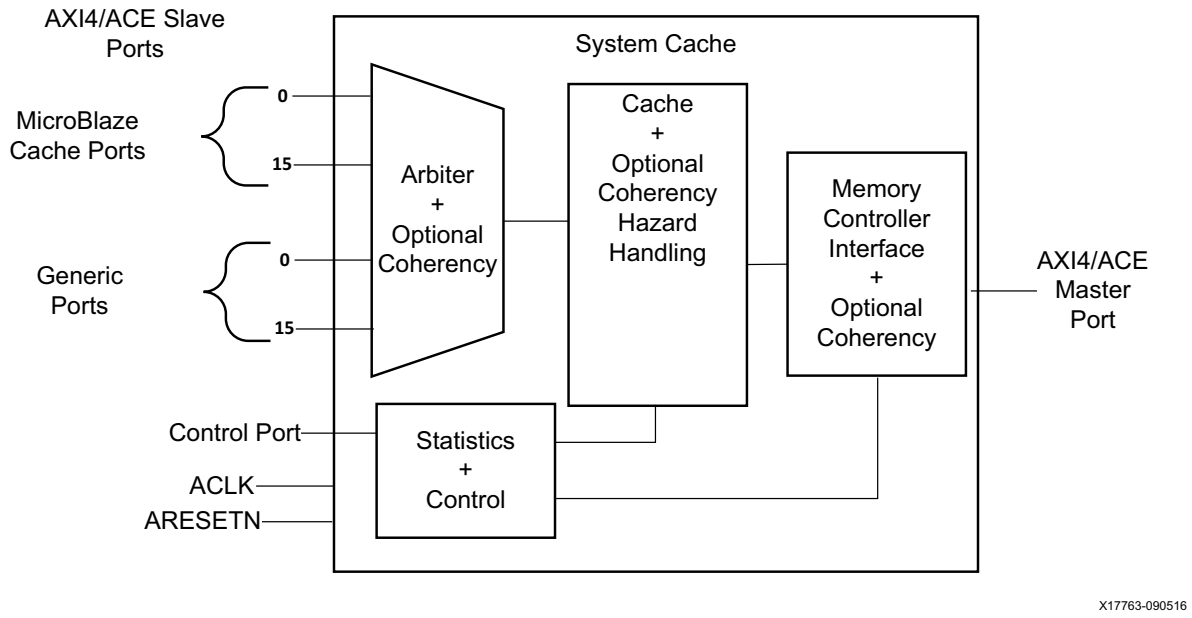

Figure 2-7: **Core Block Diagram**

Table 2-8: **I/O Interfaces**

Interface Name	Type	Description
ACLK	Input	Core clock
ARESETN	Input	Synchronous reset of core
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	AX4-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 module are 64 bits wide. The address map structure is shown in Table 2-9. 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 is always started at once.

Table 2-9: **Address Structure**

Bits	Description
16:14	Category
13:10	Port Numbers
9:5	Functionality
4:3	Register
2	High/Low
1:0	Always 00

Category

Table 2-10: **Category Field**

Address (binary)	Description
0_00yy_yyxx_xxxx_xx00	Statistics Functionality Field for Ports (Optimized ports)
0_01yy_yyxx_xxxx_xx00	Statistics Functionality Field for Ports (Generic ports)
0_1000_00xx_xxxx_xx00	Statistics Field for Arbiter
0_1100_00xx_xxxx_xx00	Statistics Field for Access
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
1_1100_00xx_xxxx_xx00	Statistics Field for Control

Port Numbers

Table 2-11: **Port Numbers**

Address (binary)[(1)]	Port Number[(2)]
0_0z00_00xx_xxxx_xx00	0
0_0z00_01xx_xxxx_xx00	1

Table 2-11: **Port Numbers *(Cont'd)***

Address (binary)[1]	Port Number[2]
0_0z00_10xx_xxxx_xx00	2
0_0z00_11xx_xxxx_xx00	3
0_0z01_00xx_xxxx_xx00	4
0_0z01_01xx_xxxx_xx00	5
0_0z01_10xx_xxxx_xx00	6
0_0z01_11xx_xxxx_xx00	7
0_0z10_00xx_xxxx_xx00	8
0_0z10_01xx_xxxx_xx00	9
0_0z10_10xx_xxxx_xx00	10
0_0z10_11xx_xxxx_xx00	11
0_0z11_00xx_xxxx_xx00	12
0_0z11_01xx_xxxx_xx00	13
0_0z11_10xx_xxxx_xx00	14
0_0z11_11xx_xxxx_xx00	15

Notes:

1. z is 0 for Optimized ports and 1 for Generic ports.
2. Statistics for the ports are defined in Table 2-12 when used, 0 otherwise.

Functionality

Statistics Functionality Field for Ports

Table 2-12: **Statistics Field for Optimized and Generic Ports**

Address (binary)[1]	Functionality	Access	Statistics Format	Description
0_0zyy_yy00_000x_xx00	Read Segments	R	COUNT[2]	Number of segments per read transaction.
0_0zyy_yy00_001x_xx00	Write Segments	R		Number of segments per write transaction.
0_0zyy_yy00_010x_xx00	RIP	R	QUEUE[4]	Read Information Port queue statistics.
0_0zyy_yy00_011x_xx00	R	R		Read data queue statistics.
0_0zyy_yy00_100x_xx00	BIP	R		BRESP Information Port queue statistics.
0_0zyy_yy00_101x_xx00	BP	R		BRESP Port queue statistics.
0_0zyy_yy00_110x_xx00	WIP	R		Write Information Port queue statistics.
0_0zyy_yy00_111x_xx00	W	R		Write data queue statistics.

Table 2-12: **Statistics Field for Optimized and Generic Ports *(Cont'd)***

Address (binary)[1]	Functionality	Access	Statistics Format	Description
0_0zyy_yy01_000x_xx00	Read Blocked	R	COUNT[2]	Number of cycles a read was prohibited from taking part in arbitration.
0_0zyy_yy01_001x_xx00	Write Hit	R	SCOUNT[3]	Number of write hits.
0_0zyy_yy01_010x_xx00	Write Miss	R		Number of write misses.
0_0zyy_yy01_011x_xx00	Write Miss Dirty	R		Number of dirty write misses.
0_0zyy_yy01_100x_xx00	Read Hit	R		Number of read hits.
0_0zyy_yy01_101x_xx00	Read Miss	R		Number of read misses.
0_0zyy_yy01_110x_xx00	Read Miss Dirty	R		Number of dirty read misses.
0_0zyy_yy01_111x_xx00	Locked Write Hit	R		Number of locked write hits.
0_0zyy_yy10_000x_xx00	Locked Read Hit	R		Number of locked read hits.
0_0zyy_yy10_001x_xx00	First Write Hit	R		Number of first write hits.
0_0zyy_yy10_010x_xx00	Read Latency	R	COUNT[2]	Read latency statistics.
0_0zyy_yy10_011x_xx00	Write Latency	R		Write latency statistics.
0_0zyy_yy10_100x_xx00	Read Latency Configuration	R/W	LONGINT[5]	Configuration for read latency statistics collection. Default value 0. Available modes are defined in Table 2-31.
0_0zyy_yy10_101x_xx00	Write Latency Configuration	R/W		Configuration for read latency statistics collection. Default value 4. Available modes are defined in Table 2-32.

Notes:

1. z is 0 for Optimized ports and 1 for Generic ports.
2. See Table 2-24 for the COUNT register fields
3. See Table 2-25 for the SCOUNT register fields
4. See Table 2-26 for the QUEUE register fields.
5. See Table 2-27 for the LONGINT register fields.

Statistics Field for Arbiter

Table 2-13: **Statistics Field for Arbiter**

Address (binary)	Functionality	Access	Description[1]
0_10yy_yy00_000x_xx00	Valid	R	The number of clock cycles a transaction takes after being arbitrated.
0_10yy_yy00_001x_xx00	Concurrent	R	Number of transactions available to select from when arbitrating.

Notes:

1. Statistics format is COUNT. See Table 2-24 for the COUNT register fields.

Statistics Field for Access

Table 2-14: **Statistics Field for Access**

Address (binary)	Functionality	Access	Description[1]
0_11yy_yy00_000x_xx00	Valid	R	The number of clock cycles a transaction takes after passing the access stage.
0_11yy_yy00_001x_xx00	Snoop Fetch Stall		Time snoop fetch stalls because of conflicts.
0_11yy_yy00_010x_xx00	Snoop Request Stall		Time snoop request stalls because of conflicts.
0_11yy_yy00_011x_xx00	Snoop Action Stall		Time snoop action stalls because of conflicts.

Notes:
1. Statistics format is COUNT. See Table 2-24 for the COUNT register fields.

Statistics Field for Lookup

Table 2-15: **Statistics Field for Lookup**

Address (binary)	Functionality	Access	Description[1]
1_00yy_yy00_000x_xx00	Reserved		
1_00yy_yy00_001x_xx00	Fetch Stall	R	Time fetch stalls because of conflict.
1_00yy_yy00_010x_xx00	Mem Stall		Time memory stalls because of conflict.
1_00yy_yy00_011x_xx00	Data Stall		Time stalled due to memory access.
1_00yy_yy00_100x_xx00	Data Hit Stall		Time stalled due to conflict.
1_00yy_yy00_101x_xx00	Data Miss Stall		Time stalled due to full buffers.

Notes:
1. Statistics format is COUNT. See Table 2-24 for the COUNT register fields.

Statistics Field for Update

Table 2-16: **Statistics Field for Update**

Address (binary)	Functionality	Access	Statistics Format	Description
1_01yy_yy00_000x_xx00	Stall	R	COUNT[1]	Cycles transactions are stalled.
1_01yy_yy00_001x_xx00	Tag Free	R		Cycles tag interface is free.
1_01yy_yy00_010x_xx00	Data free	R		Cycles data interface is free.

Table 2-16: **Statistics Field for Update *(Cont'd)***

Address (binary)	Functionality	Access	Statistics Format	Description
1_01yy_yy00_011x_xx00	Read Information	R	QUEUE[2]	Queue statistics for read transactions.
1_01yy_yy00_100x_xx00	Read Data	R		Queue statistics for read data.
1_01yy_yy00_101x_xx00	Evict	R		Queue statistics for evict information.
1_01yy_yy00_110x_xx00	BRESP Source	R		Queue statistics for BRESP source information.
1_01yy_yy00_111x_xx00	Write Miss	R		Queue statistics for write miss information.
1_01yy_yy01_000x_xx00	Write Miss Allocate	R		Queue statistics for allocated write miss data.

Notes:
1. See Table 2-24 for the COUNT register fields.
2. See Table 2-26 for the QUEUE register fields.

Statistics Field for Backend

Table 2-17: **Statistics Field for Backend**

Address (binary)	Functionality	Access	Statistics Format	Description
1_10yy_yy00_000x_xx00	Write Address	R	QUEUE[1]	Queue statistics for write address channel information.
1_10yy_yy00_001x_xx00	Write Data	R		Queue statistics for write channel data.
1_10yy_yy00_010x_xx00	Read Address	R		Queue statistics for read address channel information.
1_10yy_yy00_011x_xx00	Search Depth	R	COUNT[2]	Transaction search depth for read access before released.
1_10yy_yy00_100x_xx00	Read Stall	R		Cycles stall due to search.
1_10yy_yy00_101x_xx00	Read Protected Stall	R		Cycles stall due to conflict.
1_10yy_yy00_110x_xx00	Read Latency	R		Read latency statistics for external transactions to memory.
1_10yy_yy00_111x_xx00	Write Latency	R		Write latency statistics for external transactions to memory.

Table 2-17: **Statistics Field for Backend *(Cont'd)***

Address (binary)	Functionality	Access	Statistics Format	Description
1_10yy_yy01_000x_xx00	Read Latency Configuration	R/W	LONGINT[3]	Configuration for read latency statistics collection. Default value 0. Available modes are defined in Table 2-31.
1_10yy_yy01_001x_xx00	Write Latency Configuration	R/W		Configuration for read latency statistics collection. Default value 4. Available modes are defined in Table 2-32.

Notes:
1. See Table 2-26 for the QUEUE register fields.
2. See Table 2-24 for the COUNT register fields.
3. See Table 2-27 for the LONGINT register fields.

Statistics Field for Control

Table 2-18: **Control**

Address (binary)	Functionality	Access	Description[1]
1_11yy_yy00_0000_0x00	Statistics Reset	W	Writing to this register resets all statistics data.
1_11yy_yy00_0000_1x00	Statistics Enable	R/W	Configuration for enabling statistics collection. Default value 1. 0: Statistics collection disabled 1: Statistics collection enabled
1_11yy_yy00_0001_0x00	Cache Clean	W	Invalidate address written to register Non-Secure if C_ENABLE_NONSECURE is set.
1_11yy_yy00_0001_1x00	Cache Flush	W	Flush any Dirty data for address written to register. Non-Secure if C_ENABLE_NONSECURE is set.
1_11yy_yy00_0010_0x00	Version Register 0 Basic	R	Basic configuration and version register, bits defined in Table 2-19.
1_11yy_yy00_0010_1x00	Version Register 1 Extended	R	Extended configuration and version register, bits defined in Table 2-20.
1_11yy_yy00_0011_0x00	DVM First	W	First or only part of a DVM transaction. Non-Secure if C_ENABLE_NONSECURE is set.
1_11yy_yy00_0011_1x00	DVM Second	W	Second part if needed of a DVM transaction. Non-Secure if C_ENABLE_NONSECURE is set.
1_11yy_yy00_0100_0x00	Memory Barrier	W	Insert memory barrier Non-Secure if C_ENABLE_NONSECURE is set.
1_11yy_yy00_0100_1x00	Synchronization Barrier	W	Insert synchronization barrier Non-Secure if C_ENABLE_NONSECURE is set.
1_11yy_yy00_0101_0x00	Secure Cache Clean	W	Invalidate Secure address written to register. Only available if C_ENABLE_NONSECURE is set
1_11yy_yy00_0101_1x00	Secure Cache Flush	W	Flush any Dirty data for Secure address written to register. Only available if C_ENABLE_NONSECURE is set.

Table 2-18: **Control *(Cont'd)***

Address (binary)	Functionality	Access	Description[1]
1_11yy_yy00_0110_0x00	Secure DVM First	W	First or only part of a Secure DVM transaction. Only available if C_ENABLE_NONSECURE is set.
1_11yy_yy00_0110_1x00	Secure DVM Second	W	Second part if needed of a Secure DVM transaction. Only available if C_ENABLE_NONSECURE is set.
1_11yy_yy00_0111_0x00	Secure Memory Barrier	W	Insert Secure memory barrier. Only available if C_ENABLE_NONSECURE is set.
1_11yy_yy00_0111_1x00	Secure Synchronization Barrier	W	Insert Secure synchronization barrier. Only available if C_ENABLE_NONSECURE is set.

Notes:
1. Statistics Format is LONGINT. See Table 2-27 for the LONGINT register fields.

Version Register 0

Table 2-19: **Version Register 0 Bit Field Definition**

Bit	Default Value	Access	Description
63:32	N/A	N/A	Reserved
31:30	C_ENABLE_VERSION_REGISTER	R	Version registers available: 0 - Basic register set, only this register 1 - Full register set, this and register1, see Table 2-20 2 and 3 are reserved
29:25	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	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	C_ENABLE_EXCLUSIVE	R	Internal Exclusive monitor implementation: 0 - Disabled 1 - Enabled 2 to 3 are reserved
17:16	C_ENABLE_COHERENCY	R	Cache coherency implementation: 0 - Disabled 1 - Optimized port cache coherency 2 - Master port cache coherency 3 - reserved

Table 2-19: **Version Register 0 Bit Field Definition *(Cont'd)***

Bit	Default Value	Access	Description
15:8	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 - Reserved for future use.
7:0	4	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 to 255 are reserved

Version Register 1

Table 2-20: **Version Register 1 Bit Field Definition**

Bit	Default Value	Access	Description
63:2	N/A	N/A	Reserved
21:19	Log_2(C_Lx_CACHE_LINE_LENGTH/4)	R	Cache line length of connected masters on the optimized port, see Table 2-23.
18:15	Log_2(C_Lx_CACHE_SIZE/64)	R	Cache size of connected masters on the optimized port, see Table 2-22.
14:12	Log_2(C_CACHE_LINE_LENGTH/4)	R	Internal cache line length, see Table 2-23.
11:8	Log_2(C_CACHE_SIZE/64)	R	Internal cache size, see Table 2-22.
7:5	Log_2(C_CACHE_DATA_WIDTH/2)	R	Internal cache data width, see Table 2-21.
4:2	Log_2(C_M0_AXI_DATA_WIDTH/8)	R	Width of Master AXI4 interfaces used to access memory sub system, see Table 2-21.
1:0	Log_2(C_NUM_WAYS/2)	R	Number of associative sets: 0 = 2-way set associative 1 = 4-way set associative 2 to 3 = Reserved

Interface and Cache Data Width

Table 2-21: **Interface and Cache Data Width**

Value	Description
0	8-bit data interface and path
1	16-bit data interface and path

Table 2-21: **Interface and Cache Data Width *(Cont'd)***

Value	Description
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

Table 2-22: **Cache Size**

Value	Description
0	64 byte cache size
1	128 byte cache size
2	256 byte cache size
3	512 byte cache size
4	1K byte cache size
5	2K byte cache size
6	4K byte cache size
7	8K byte cache size
8	16K byte cache size
9	32K byte cache size
10	64K byte cache size
11	128K byte cache size
12	256K byte cache size
13	512K byte cache size

Table 2-23: **Cache Line Length Definitions**

Value	Description
0	4 word cache line
1	8 word cache line
2	16 word cache line
3-7	Reserved

Register

Statistics Record for COUNT

Table 2-24: **Register Field for COUNT**

Address (binary)	Register	Access	Format	Description
x_xxxx_xxxx_xxx0_0x00	Events	R	LONGINT(1)	Number of times the event has been triggered.
x_xxxx_xxxx_xxx0_1x00	Min Max Status	R	MINMAX(2)	Min, max and status information defined according to Table 2-29.
x_xxxx_xxxx_xxx1_0x00	Sum	R	LONGINT(1)	Sum of measured data(x).
x_xxxx_xxxx_xxx1_1x00	Sum^2	R	LONGINT(1)	Sum of measured data squared(x^2).

Notes:
1. See Table 2-27 for the LONGINT register fields.
2. See Table 2-29 for the MINMAX register fields.

Statistics Record for SCOUNT

Table 2-25: **Register Field for SCOUNT**

Address (binary)	Register	Access	Format	Description
x_xxxx_xxxx_xxx0_0x00	Events	R	LONGINT(1)	Number of times the event has been triggered.
x_xxxx_xxxx_xxx0_1x00	Reserved			
x_xxxx_xxxx_xxx1_0x00	Sum	R	LONGINT(1)	Sum of measured data(x).
x_xxxx_xxxx_xxx1_1x00	Sum^2	R	LONGINT(1)	Sum of measured data squared(x^2).

Notes:
1. See Table 2-27 for the LONGINT register fields.

Statistics Record for QUEUE

Table 2-26: **Register Field for QUEUE**

Address (binary)	Register	Access	Format	Description
x_xxxx_xxxx_xxx0_0x00	Empty Cycles	R	LONGINT(1)	Clock cycles the queue has been idle.
x_xxxx_xxxx_xxx0_1x00	Index Updates	R	LONGINT(1)	Number of times updated with push or pop.
x_xxxx_xxxx_xxx1_0x00	Index Max	R	MINMAX(2)	Maximum depth for queue (only maximum field used).
x_xxxx_xxxx_xxx1_1x00	Index Sum	R	LONGINT(1)	Sum of queue depth when updated.

Notes:
1. See Table 2-27 for the LONGINT register fields.
2. See Table 2-29 for the MINMAX register fields.

High/Low

Address Decoding for LONGINT

Table 2-27: **High-Low Field for LONG INT**

Address (binary)	High Low	Description
x_xxxx_xxxx_xxxx_x000	LOW	LONGINT Bits 31-0, least significant half
x_xxxx_xxxx_xxxx_x100	HIGH	LONGINT Bits 63-32, most significant half

Table 2-28: **LONG INT Register Bit Allocation**

Bits	Default Value	Access Type	Description
63:0	N/A	N/A	Long Integer

Address Decoding for MIN MAX

Table 2-29: **High-Low Field for MIN MAX**

Address (binary)	High Low	Description
x_xxxx_xxxx_xxxx_x000	LOW	MIN MAX Bits 31-0
x_xxxx_xxxx_xxxx_x100	HIGH	MIN MAX Bits 63-32

Table 2-30: **MIN MAX Register Bit Definition**

Bits	Default Value	Access Type	Description
63:48	0xFFFF	R	Minimum unsigned measurement encountered.
47:32	0x0000	R	Maximum unsigned measurement encountered, saturates when 0xFFFF is reached.
31:2	N/A	N/A	Reserved
1	0	R	Flag if number of concurrent events of the measured type has been reached, indicating that the resulting statistics are inaccurate.
0	0	R	Flag if measurements have been saturated; this means the statistics results are less accurate. Both average and standard deviation measurements will be lower than the actual values.

READ Latency

Table 2-31: **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 2-32: **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 2-33: **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 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 Figure 3-1.

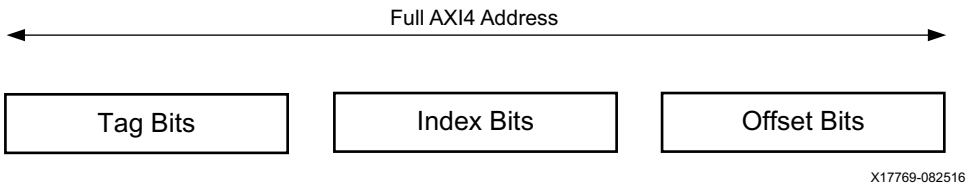

Figure 3-1: **Address Bit Usage**

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 Figure 3-1. 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 replace 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 or dirty. See Figure 3-2 for tag contents.

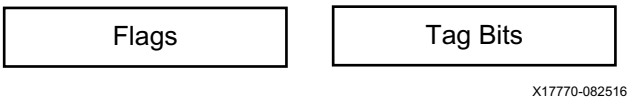

Figure 3-2: **Cache Line Tag Contents**

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 transactions according to ARCACHE and AWCACHE, as well as the current cache line state.

AXI Properties

Table 3-1 and Table 3-2 show the ARCACHE and AWCACHE bus contents for Read and Write channels respectively, according to the AMBA® AXI and ACE Protocol Specification [Ref 1].

Table 3-1: **ARCACHE Bit Field**

Other Allocate	Read Allocate	Modifiable	Bufferable
3	2	1	0

Table 3-2: **AWCACHE Bit Field**

Write Allocate	Other Allocate	Modifiable	Bufferable
3	2	1	0

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.

Table 3-3 and Table 3-4 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.

Table 3-3: **Parameter to ARCACHE Bit Mapping**

Other Allocate	Read Allocate	Modifiable	Bufferable
C_Sx_AXI_FORCE_WRITE_ALLOCATE	C_Sx_AXI_FORCE_READ_ALLOCATE	N/A	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 3-4: **Parameter to AWCACHE Bit Mapping**

Write Allocate	Other Allocate	Modifiable	Bufferable
C_Sx_AXI_FORCE_WRITE_ALLOCATE	C_Sx_AXI_FORCE_READ_ALLOCATE	N/A	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.

Property Translation

All transactions in the Master Coherent configuration always have ARDOMAIN and AWDOMAIN set to InnerShareable.

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.

Table 3-5 show the transaction settings for DVM type operations.

Table 3-5: **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

Table 3-6 and Table 3-7 show the transactions settings for Barrier type operations on read and write channel respectively.

Table 3-6: **ARCACHE and ARSNOOP Settings for Barrier Transactions**

Transaction	M0_AXI_ARCACHE	M0_AXI_ARSNOOP
Read Barrier	0010	0000

Table 3-7: **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. Table 3-8 and Table 3-9 show all types of events. Some of the write related events actually appear on the master read channel.

Table 3-8: **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 3-9: **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 Transaction Flow

Table 3-10 and Table 3-11 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 3-10: **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 3-11: **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

Table 3-11: **Complete Slave AWCACHE to Cache Event and Potential Master Transaction Mapping**

Slave AWCACHE	Cache Event	Cache Action	Master ACE Transaction	
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

General Design Guidelines

The are no golden settings to achieve maximum performance for all cases, as performance is application and system dependent. This chapter 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

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 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.

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.

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 Figure 3-3.

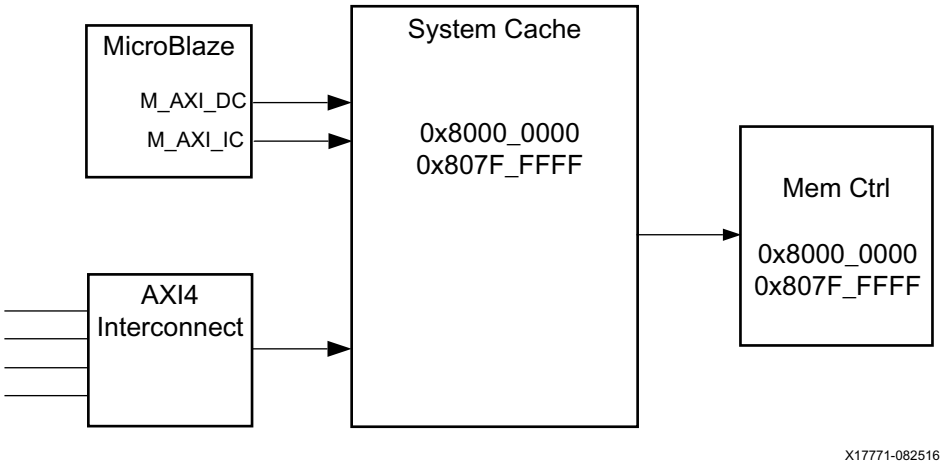

Figure 3-3: **Single Memory Controller Configuration**

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 Figure 3-4.

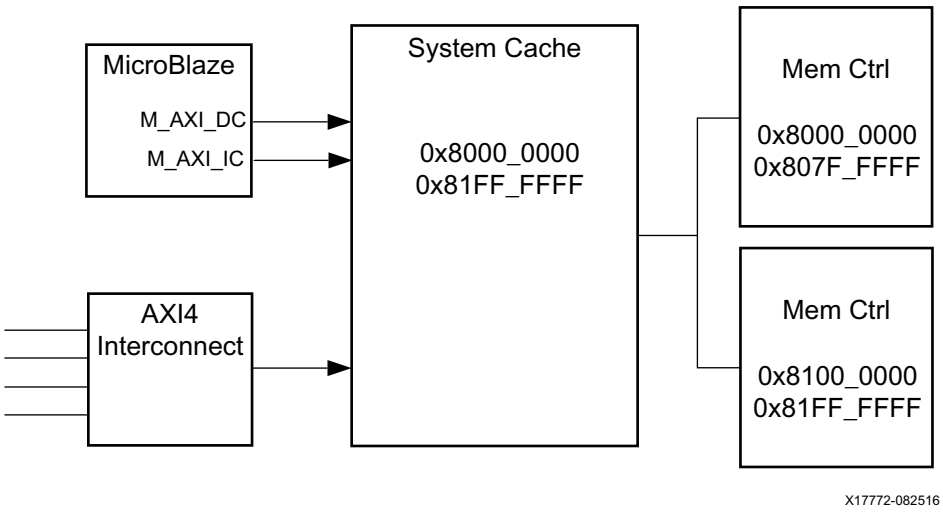

Figure 3-4: **Dual Memory Controller Configuration with Hole**

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.

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.

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, Figure 3-5 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.

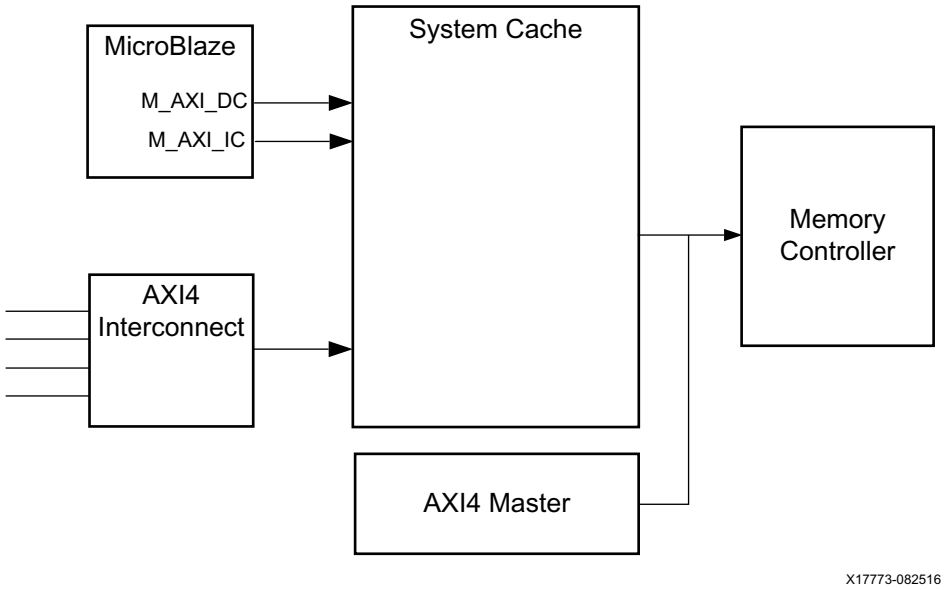

Figure 3-5: **Back-Door DMA Connection with One Additional Master**

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 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.

Protocol Description

All interfaces to the System Cache core adhere to the AXI4, AXI4-Lite and ACE 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 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
  - 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 AXI4 burst types and sizes
  - FIXED bursts are handled as INCR type burst operations (no QUEUE burst capability)
  - Up to 16 beats for WRAP bursts
  - Up to 16 beats for FIXED bursts (treated as INCR burst type)
  - 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 not supported
- All transactions executed in order regardless of thread ID value. No read reordering or write reordering is implemented.

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

Design Flow Steps

This chapter 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) [Ref 2]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [[Ref 3]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 4]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 5]

Customizing and Generating the Core

This section includes information on 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) [Ref 2] 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 you can 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) [Ref 3] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 4].

The System Cache core parameters are divided into two categories: core and system. See Table 4-5 for allowed values. The core parameter tab is shown in Figure 4-1.

Note: Figures in this chapter are illustrations of the Vivado IDE. This layout might vary from the current version.

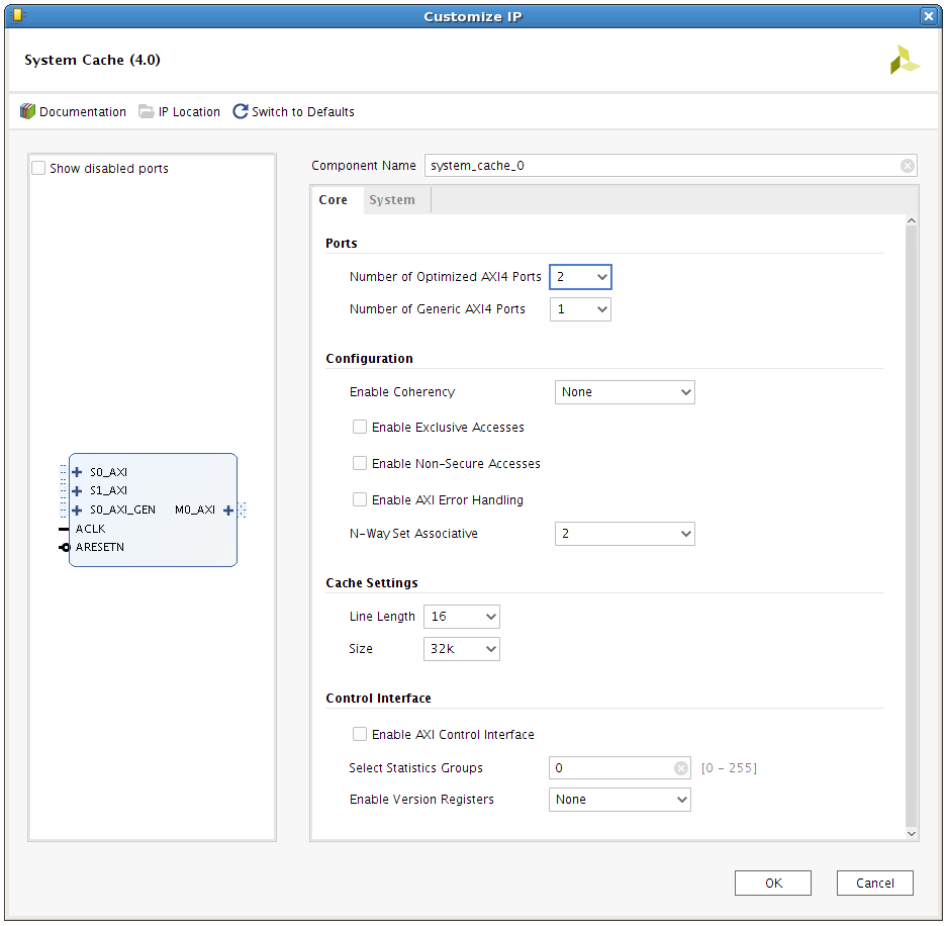

Figure 4-1: **Core Parameter Tab**

- **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 Coherency**: Set cache coherency mode to None, Optimized ports, or Master port.
- **Enable Exclusive Monitor**: Enables Exclusive handling for non-coherent implementation.
- **Enable Non-Secure Accesses**: 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.
- **N-Way Set Associative**: Specifies 2- or 4-way set associative cache.
- **Line Length**: Cache line length is fixed to 16.
- **Size**: Sets the size of the system cache in bytes.
- **Enable AXI Control Interface**: Set if statistics interface is available.
- **Select Statistics Groups**: Bit mask that determines which statistics groups are included.
- **Enable Version Register**: Set level of Version registers that should be included.

The system parameter tab is shown in Figure 4-2 with the data width parameters visible for the slave interfaces.

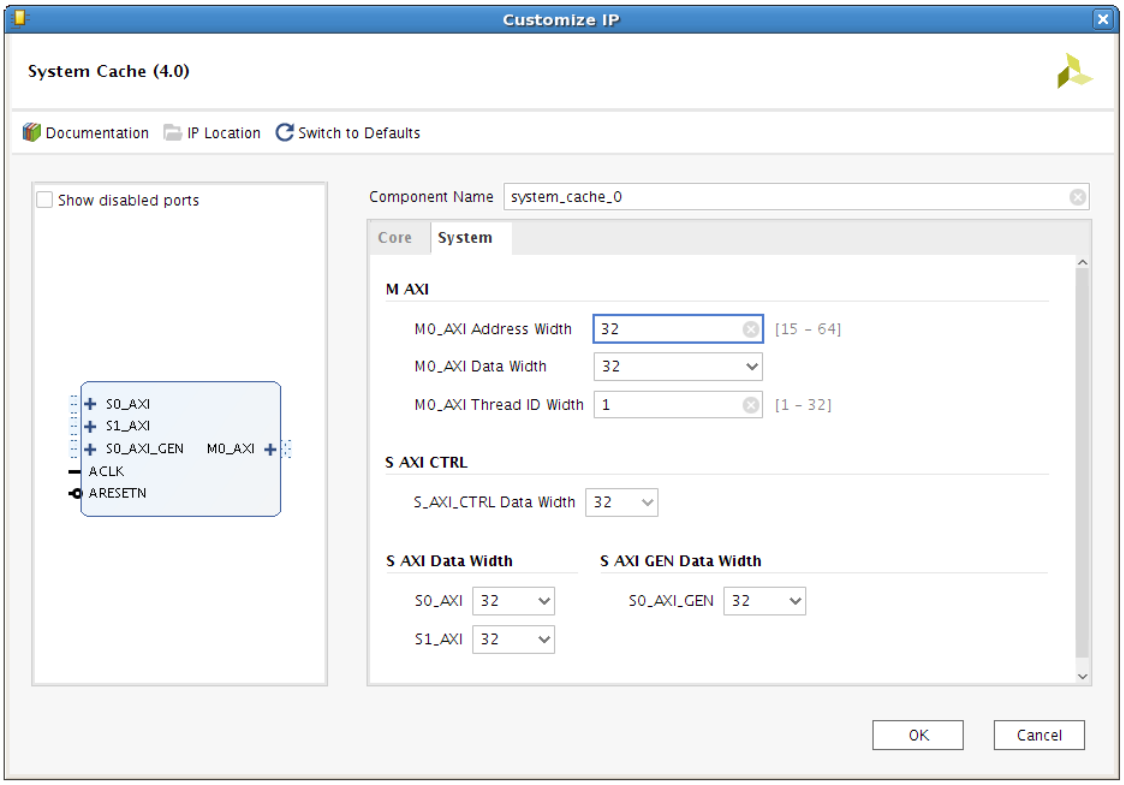

Figure 4-2: **System Parameter Tab**

- **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 Data Width**: Sets the data width of the Optimized ports individually.
- **Sx_AXI_GEN Data Width**: Sets the data width of the generic ports individually.

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. Tables 4-1 to 4-5 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)

Table 4-1: **System Cache I/O Interfaces**

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_FAMILY	FPGA Architecture	Supported architectures	virtex7	string
C_INSTANCE	Instance Name	Any instance name	system_cache	string
C_FREQ	System Cache clock frequency	Any valid frequency for the device	0	natural
C_BASEADDR	Cacheable area base address		0xFFFF FFFF FFFF FFFF	std_logic_vector

Table 4-1: **System Cache I/O Interfaces *(Cont'd)***

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_HIGHADDR	Cacheable area high address. Minimum size is 32KB		0x0000 0000 0000 0000	std_logic_vector
C_ENABLE_COHERENCY	Enable implementation of cache coherent optimized ports	0, 1, 2	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
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 - Reserved for future use	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_NUM_OPTIMIZED_PORTS	Number of ports optimized for MicroBlaze processor cache connection	0–16	1	natural
C_NUM_GENERIC_PORTS	Number of ports supporting full AXI4	0–16	0	natural
C_NUM_MASTER_PORTS	Number of master ports	1	1	natural
C_NUM_WAYS	Cache associativity	2, 4	2	natural
C_CACHE_DATA_WIDTH	Cache data width used internally. Automatically calculated to match AXI4 master interface	32, 64, 128, 256, 512	32	natural
C_CACHE_LINE_LENGTH	Cache line length. Constant value.	16	16	natural

Table 4-1: **System Cache I/O Interfaces *(Cont'd)***

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_CACHE_SIZE	Cache size in bytes	32768, 65536, 131072, 262144, 524288	32768	natural
C_Lx_CACHE_LINE_LENGTH	Cache line length on masters connected to optimized ports. Automatically assigned with manual override	4, 8, 16	4	natural
C_Lx_CACHE_SIZE	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

Table 4-2: **MicroBlaze Processor Cache Optimized AXI4 Slave Interface Parameters**

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_Sx_AXI_ADDR_WIDTH(1)	Address width	15–64	32	natural
C_Sx_AXI_DATA_WIDTH(1)	Data width	32, 128, 256, 512	32	natural
C_Sx_AXI_RRESP_WIDTH	Width of RRESP. Automatically assigned.	2, 4	2	natural
C_Sx_AXI_ID_WIDTH(1)	ID width, automatically assigned	1–32	1	natural
C_Sx_AXI_SUPPORT_UNIQUE	Reserved	0	0	natural
C_Sx_AXI_SUPPORT_DIRTY	Reserved	0	0	natural
C_Sx_AXI_FORCE_READ_ALLOCATE	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	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	Force write transactions to use write allocate, also function as override value for other allocate on read channel	0, 1	0	natural

Table 4-2: **MicroBlaze Processor Cache Optimized AXI4 Slave Interface Parameters *(Cont'd)***

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_Sx_AXI_PROHIBIT_WRITE_ALLOCATE	Prohibit write transactions from using write allocate, also function as override value for other allocate on read channel	0, 1	1	natural
C_Sx_AXI_FORCE_READ_BUFFER	Force read transactions to use buffer bit	0, 1	0	natural
C_Sx_AXI_PROHIBIT_READ_BUFFER	Prohibit read transactions from using buffer bit	0, 1	0	natural
C_Sx_AXI_FORCE_WRITE_BUFFER	Force write transactions to use buffer bit	0, 1	0	natural
C_Sx_AXI_PROHIBIT_WRITE_BUFFER	Prohibit write transactions from using buffer bit	0, 1	0	natural

Notes:
1. x = 0 - 15

Table 4-3: **Generic AXI4 Slave Interface Parameters**

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_Sx_AXI_GEN_ADDR_WIDTH	Address Width	15–64	32	natural
C_Sx_AXI_GEN_DATA_WIDTH	Data Width	32, 64, 128, 256, 512	32	natural
C_Sx_AXI_GEN_ID_WIDTH	ID width, automatically assigned	1–32	1	natural
C_Sx_AXI_GEN_SUPPORT_UNIQUE	Reserved	0	0	natural
C_Sx_AXI_GEN_SUPPORT_DIRTY	Reserved	0	0	natural
C_Sx_AXI_GEN_FORCE_READ_ALLOCATE	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	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	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	Prohibit write transactions from using write allocate, also function as override value for other allocate on read channel	0, 1	1	natural

Table 4-3: **Generic AXI4 Slave Interface Parameters *(Cont'd)***

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_Sx_AXI_GEN_FORCE_READ_BUFFER	Force read transactions to use buffer bit	0, 1	0	natural
C_Sx_AXI_GEN_PROHIBIT_READ_BUFFER	Prohibit read transactions from using buffer bit	0, 1	0	natural
C_Sx_AXI_GEN_FORCE_WRITE_BUFFER	Force write transactions to use buffer bit	0, 1	0	natural
C_Sx_AXI_GEN_PROHIBIT_WRITE_BUFFER	Prohibit write transactions from using buffer bit	0, 1	0	natural

Table 4-4: **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

Table 4-5: **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

User Parameters

Table 4-6 shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl console).

Table 4-6: **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 Coherency	C_ENABLE_COHERENCY	0

Table 4-6: **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

Vivado IDE Parameter/Value[1]	User Parameter/Value[1]	Default Value
Enable Exclusive Accesses	C_ENABLE_EXCLUSIVE	0
Number of Associative Sets	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	
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	
M0_AXI Address Width	C_M0_AXI_ADDR_WIDTH	32
M0_AXI Data Width	C_M0_AXI_DATA_WIDTH	32
M0_AXI Thread ID Width	C_M0_AXI_THREAD_ID_WIDTH	1
Sx_AXI	C_Sx_AXI_DATA_WIDTH	32
Sx_AXI_GEN	C_Sx_AXI_GEN_DATA_WIDTH	32

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.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3].

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

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) [Ref 5].

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) [Ref 3].

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 (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) [Ref 6].

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 describe changes between the ISE version of the core and the current version.

The supported cache sizes has been increased to support 256KB and 512KB.

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

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

Debugging

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

Finding Help on Xilinx.com

To help in the design and debug process when using the System Cache 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.

Documentation

This product guide is the main document associated with the System Cachecore. 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 proper 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

AR: 54452

Technical Support

Xilinx provides technical support at the Xilinx Support web page 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 contact Xilinx Technical Support, navigate to the Xilinx Support web page.

Debug Tools

There are many tools available to address System Cache 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 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) [Ref 7].

Reference Boards

All Xilinx development boards for 7 series, UltraScale™ and UltraScale+™ devices support System Cache. 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* [Ref 1] 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.

AXI4 Checks

Either use bus analyzer or connect the relevant AXI4 signals to a logic analyzer in the Vivado debug feature. Make sure the data is captured with ACLK.

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.
- Make sure 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.
- Make sure 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-Lite interface.

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

References

These documents provide supplemental material useful with this product guide:

1. AMBA® AXI and ACE Protocol Specification (ARM IHI 0022E)
2. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
3. *Vivado Design Suite User Guide: Designing with IP* (UG896)
4. *Vivado Design Suite User Guide: Getting Started* (UG910)
5. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
6. *ISE to Vivado Design Suite Migration Guide* (UG911)
7. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/05/2017	4.0	• Added section for AXI transaction properties and translation
10/05/2016	4.0	• 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	3.1	• Added support for UltraScale+ families
06/24/2015	3.1	• Moved performance and resource utilization data to the web
04/01/2015	3.1	• 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
10/02/2013	3.0	• 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	1.0	Initial release as a Product Guide; 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 2013–2017 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. AMBA, AMBA Designer, ARM, ARM1176JZ-S, CoreSight, Cortex, PrimeCell, and MPCore are trademarks of ARM in the EU and other countries. All other trademarks are the property of their respective owners.