Table A4-5 Memory type encoding

ARCACHE[3:0]	AWCACHE[3:0]	Memory type
0b0000	0b0000	Device Non-bufferable
0b0001	0b0001	Device Bufferable
0b0010	0b0010	Normal Non-cacheable Non-bufferable
0b0011	0b0011	Normal Non-cacheable Bufferable
0b1010	0b0110	Write-Through No-Allocate
0b1110 (0b0110)	0b0110	Write-Through Read-Allocate
0b1010	0b1110 (0b1010)	Write-Through Write-Allocate
0b1110	0b1110	Write-Through Read and Write-Allocate
0b1011	0b0111	Write-Back No-Allocate
0b1111 (0b0111)	0b0111	Write-Back Read-Allocate
0b1011	0b1111 (0b1011)	Write-Back Write-Allocate
0b1111	0b1111	Write-Back Read and Write-Allocate

Table A4-4 ARCACHE bit allocations

Signal	AXI4 definition	Description
ARCACHE[3]	Other Allocate	When asserted, the transaction must be looked up in a cache because it could have been allocated in the cache by another transaction, either a write transaction or a transaction from another Manager. The transaction must also be looked up in a cache if **ARCACHE[2]** is asserted. When deasserted, if **ARCACHE[2]** is also deasserted, then the transaction does not need to be looked up in a cache.
ARCACHE[2]	Allocate	When asserted, the transaction must be looked up in a cache because it could have been allocated. The transaction must also be looked up in a cache if **ARCACHE[3]** is asserted. When deasserted, if **ARCACHE[3]** is also deasserted, then the transaction does not need to be looked up in a cache. When asserted, this specifacation recommends that this transaction is allocated in the cache for performance reasons.
ARCACHE[1]	Modifiable	When asserted, the characteristics of the transaction can be modified and a larger quantity of read data can be fetched than is required. When deasserted the characteristics of the transaction must not be modified.
ARCACHE[0]	Bufferable	This bit has no effect when **ARCACHE[3:1]** are deasserted. When **ARCACHE[3:2]** are deasserted and **ARCACHE[1]** is asserted: • If this bit is deasserted, the read data must be obtained from the final destination. • If this bit is asserted, the read data can be obtained from the final destination or from a write that is progressing to the final destination. When either **ARCACHE[3]** is asserted, or **ARCACHE[2]** is asserted, this bit can be used to distinguish between Write-Through and Write-Back memory types.

Table A4-3 AWCACHE bit allocations

Signal	AXI4 definition	Description
AWCACHE[3]	Allocate	When asserted, the transaction must be looked up in a cache because it could have been previously allocated. The transaction must also be looked up in a cache if **AWCACHE[2]** is asserted. When deasserted, if **AWCACHE[2]** is also deasserted, then the transaction does not need to be looked up in a cache and the transaction must propagate to the final destination. When asserted, this specification recommends that this transaction is allocated in the cache for performance reasons.
AWCACHE[2]	Other Allocate	When asserted, the transaction must be looked up in a cache because it could have been previously allocated in the cache by another transaction, either a read transaction or a transaction from another Manager. The transaction must also be looked up in a cache if **AWCACHE[3]** is asserted. When deasserted, if **AWCACHE[3]** is also deasserted, then the transaction does not need to be looked up in a cache and the transaction must propagate to the final destination.
AWCACHE[1]	Modifiable	When asserted, the characteristics of the transaction can be modified and writes can be merged. When deasserted, the characteristics of the transaction must not be modified.
AWCACHE[0]	Bufferable	When deasserted, if both of **AWCACHE[3:2]** are deasserted, the write response must be given from the final destination. When asserted, if both of **AWCACHE[3:2]** are deasserted, the write response can be given from an intermediate point, but the write transaction is required to be made visible at the final destination *in a timely manner*. When deasserted, if either of **AWCACHE[3:2]** is asserted, the write response can be given from an intermediate point, but the write transaction is required to be made visible at the final destination *in a timely manner*. When asserted, if either of **AWCACHE[3:2]** is asserted, the write response can be given from an intermediate point. The write transaction is not required to be made visible at the final destination.

Table A4-6 Protection encoding

AxPROT	Value	Function
[0]	0	Unprivileged access
	1	Privileged access
[1]	0	Secure access
	1	Non-secure access
[2]	0	Data access
	1	Instruction access

Table A7-2 AXI4 atomic access encoding

AxLOCK	Access type
0b0	Normal access
0b1	Exclusive access

Table A3-2 Burst size encoding

AxSIZE[2:0]	Bytes in transfer
0b000	1
0b001	2
0b010	4
0b011	8
0b100	16
0b101	32
0b110	64
0b111	128

Table A3-3 Burst type encoding

AxBURST[1:0]	Burst type
0b00	FIXED
0b01	INCR
0b10	WRAP
0b11	Reserved

Table D3-2 Shareability domain encoding

AxDOMAIN[1:0]	Domain
0b00	Non-shareable
0b01	Inner Shareable
0b10	Outer Shareable
0b11	System

Table D3-5 Barrier transaction signal encoding

AxBAR[1:0]	Barrier type
0b00	Normal access, respecting barriers
0b01	Memory barrier
0b10	Normal access, ignoring barriers
0b11	Synchronization barrier

Table D3-7 Permitted read address control signal combinations

Transaction group	ARBAR[0]	ARDOMAIN	ARSNOOP	Transaction type
Non-snooping	0b0	0b00 0b11	0b0000	ReadNoSnoop
Coherent	0b0	0b01 0b10	0b0000	ReadOnce
			0b0001	ReadShared
			0b0010	ReadClean
			0b0011	ReadNotSharedDirty
			0b0111	ReadUnique
			0b1011	CleanUnique
			0b1100	MakeUnique
Cache maintenance	0b0	0b00 0b01 0b10	0b1000	CleanShared
			0b1001	CleanInvalid
			0b1101	MakeInvalid
Barrier	0b1	0b00 0b01 0b10 0b11	0b0000	Barrier
DVM	0b0	0b01 0b10	0b1110	DVM Complete
			0b1111	DVM Message

Table D3-8 Permitted write address control signal combinations

Transaction group	AWBAR[0]	AWDOMAIN	AWSNOOP	Transaction type
Non-snooping	0b0	0b00 0b11	0b000	WriteNoSnoop
Coherent	0b0	0b01 0b10	0b000	WriteUnique
			0b001	WriteLineUnique
Memory update	0b0	0b00 0b01 0b10	0b010	WriteClean
			0b011	WriteBack
		0b01 0b10	0b100	Evict
		0b00 0b01 0b10	0b101	WriteEvict[a]
Barrier	0b1	0b00 0b01 0b10 0b11	0b000	Barrier

a. A component that supports the WriteEvict transaction must provide the **AWUNIQUE** signal.

Table D3-19 ACSNOOP encodings

ACSNOOP[3:0]	Transaction
0b0000	ReadOnce
0b0001	ReadShared
0b0010	ReadClean
0b0011	ReadNotSharedDirty
0b0111	ReadUnique
0b1000	CleanShared
0b1001	CleanInvalid
0b1101	MakeInvalid
0b1110	DVM Complete
0b1111	DVM Message

Table D3-21 Snoop response bit allocations

Signal	Name	Meaning	
CRRESP[0]	DataTransfer	HIGH	Indicates that a full cache line of data will be provided on the snoop data channel for this transaction.
		LOW	Indicates that no data will be provided on the snoop data channel for this transaction.
CRRESP[1]	Error	HIGH	When HIGH, the Error bit indicates that the snooped cache line is in error. Typically, this is caused by a corrupt cache line that has been detected through the use of an *Error Correction Code* (ECC) system.
		LOW	Indicates that no Error condition has been detected.
CRRESP[2]	PassDirty	HIGH	When HIGH, it indicates that before the snoop process, the cache line was held in a Dirty state and the responsibility for writing the cache line back to main memory is being passed to the initiating Manager or the interconnect. For all transactions, except MakeInvalid, if the cache line was held in a Dirty state before the snoop process and a copy is not being retained by the cache, then the PassDirty bit must be set HIGH.
		LOW	Indicates the responsibility for writing the cache line back to main memory is not being passed.
CRRESP[3]	IsShared	HIGH	Indicates that the snooped cache retains a copy of the cache line after the snoop process has completed.
		LOW	Copy of cache line was not retained.
CRRESP[4]	WasUnique	HIGH	Indicates that the cache line was held in a Unique state before the snoop process. The WasUnique bit must only be HIGH if it is known that no other cache can have a copy of the cache line.
		LOW	No information is provided on whether or not the cache line was held in a Unique state before the snoop process.

Table D5-2 Snoop transaction options on the address snoop channel

Snoop transaction	Required cache line state change	Snoop transaction option
ReadOnce	None	ReadOnce ReadClean, ReadNotSharedDirty, ReadShared ReadUnique, CleanInvalid CleanShared
ReadClean	Shared or Invalid	ReadClean, ReadNotSharedDirty, ReadShared ReadUnique, CleanInvalid
ReadNotSharedDirty	Shared or Invalid	ReadClean, ReadNotSharedDirty, ReadShared ReadUnique, CleanInvalid
ReadShared	Shared or Invalid	ReadClean, ReadNotSharedDirty, ReadShared ReadUnique, CleanInvalid
ReadUnique	Invalid	ReadUnique, CleanInvalid
MakeInvalid	Invalid	ReadUnique, CleanInvalid, MakeInvalid
CleanInvalid	Invalid	ReadUnique, CleanInvalid
CleanShared	Clean or Invalid	ReadUnique, CleanInvalid CleanShared

Table A3-5 RRESP and BRESP encoding

RRESP[1:0] BRESP[1:0]	Response
0b00	OKAY
0b01	EXOKAY
0b10	SLVERR
0b11	DECERR

Table D3-14 Additional RRESP read response bits

Signal	Source	Name	Meaning	
RRESP[2]	Interconnect	PassDirty	HIGH	The cache line is Dirty with respect to main memory and the initiating Manager must ensure that the cache line is written back to main memory, at some time. The initiating Manager must either perform the write, or pass the responsibility to perform the write to another Manager.
			LOW	It is not the responsibility of the initiating Manager to ensure that the cache line is written back to main memory.
RRESP[3]	Interconnect	IsShared	HIGH	Another copy of the associated data might be held in another cache and the cache line must be held in a Shared state.
			LOW	It is the only cached copy of the associated data and the cache line can be held in a Unique state.

Memory barriers

A Manager component issues a memory barrier to guarantee that if another Manager component in the appropriate domain can observe any transaction after the barrier it must be able to observe every transaction prior to the barrier.

Synchronization barriers

A Manager component issues a synchronization barrier to determine when every Manager component in the appropriate domain can observe all transactions that preceded the barrier transaction. For System domain synchronization barriers, all transactions that are issued before the barrier transaction must have reached the destination Subordinate components before the barrier transaction completes.

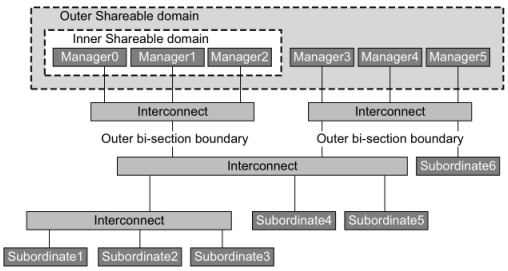

Figure D1-4 Example system using shareability domains

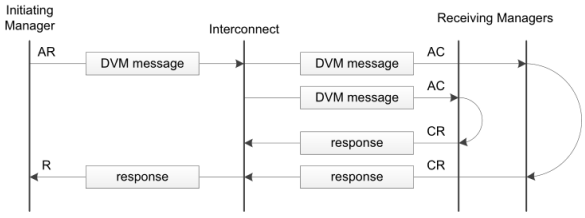

Figure D13-1 One-part DVM message flow

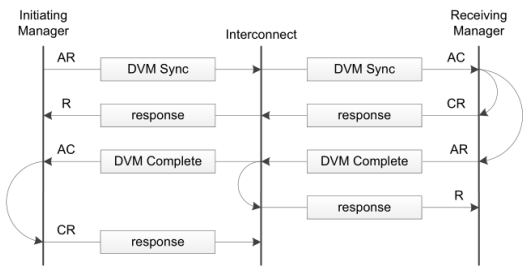

Figure D13-3 Synchronization flow between initiating Manager and receiving Manager