Study and performance analysis of cache coherence protocols in shared-memory multiprocessors

Anthony Gégo

Appendix: detailed gem5 protocols documentation

Notice: These protocols have been designed by the gem5 authors

This document has been automatically generated !

List of Tables

1.1 mesi_three_level - Directory - State . . . 3
1.2 mesi_three_level - Directory - action . . . 4
1.3 mesi_three_level - Directory - Event . . . 4
1.4 mesi_three_level - Directory - table . . . 5
1.5 mesi_three_level - DMA - State . . . 6
1.6 mesi_three_level - DMA - action . . . 6
1.7 mesi_three_level - DMA - Event . . . 6
1.8 mesi_three_level - DMA - table . . . 7
1.9 mesi_three_level - L0Cache - State . . . 8
1.10 mesi_three_level - L0Cache - action . . . 8
1.11 mesi_three_level - L0Cache - Event . . . 8
1.12 mesi_three_level - L0Cache - table . . . 9
1.13 mesi_three_level - L1Cache - State . . . 10
1.14 mesi_three_level - L1Cache - action . . . 10
1.15 mesi_three_level - L1Cache - Event . . . 11
1.16 mesi_three_level - L1Cache - table . . . 12
1.17 mesi_three_level - L2Cache - State . . . 13
1.18 mesi_three_level - L2Cache - action . . . 13
1.19 mesi_three_level - L2Cache - Event . . . 14
1.20 mesi_three_level - L2Cache - table . . . 15

2.1 mesi_two_level - Directory - State . . . 16
2.2 mesi_two_level - Directory - action . . . 17
2.3 mesi_two_level - Directory - Event . . . 17
2.4 mesi_two_level - Directory - table . . . 18
2.5 mesi_two_level - DMA - State . . . 19
2.6 mesi_two_level - DMA - action . . . 19
2.7 mesi_two_level - DMA - Event . . . 19
2.8 mesi_two_level - DMA - table . . . 20
2.9 mesi_two_level - L1Cache - State . . . 21
2.10 mesi_two_level - L1Cache - action . . . 21
2.11 mesi_two_level - L1Cache - Event . . . 22
2.12 mesi_two_level - L1Cache - table . . . 23
2.13 mesi_two_level - L2Cache - State . . . 24
2.14 mesi_two_level - L2Cache - action . . . 24
2.15 mesi_two_level - L2Cache - Event . . . 25
2.16 mesi_two_level - L2Cache - table . . . 26

3.1 mi_example - Directory - State . . . 27
3.2 mi_example - Directory - action . . . 28
3.3 mi_example - Directory - Event . . . 28
3.4 mi_example - Directory - table . . . 29
3.5 mi_example - DMA - State . . . 30
3.6 mi_example - DMA - action . . . 30

3.7 mi_example - DMA - Event . . . 30
3.8 mi_example - DMA - table . . . 31
3.9 mi_example - L1Cache - State . . . 32
3.10 mi_example - L1Cache - action . . . 32
3.11 mi_example - L1Cache - Event . . . 32
3.12 mi_example - L1Cache - table . . . 33

4.1 moesi_cmp_directory - Directory - State . . . 35
4.2 moesi_cmp_directory - Directory - action . . . 35
4.3 moesi_cmp_directory - Directory - Event . . . 36
4.4 moesi_cmp_directory - Directory - table . . . 37
4.5 moesi_cmp_directory - DMA - State . . . 38
4.6 moesi_cmp_directory - DMA - action . . . 38
4.7 moesi_cmp_directory - DMA - Event . . . 38
4.8 moesi_cmp_directory - DMA - table . . . 39
4.9 moesi_cmp_directory - L1Cache - State . . . 40
4.10 moesi_cmp_directory - L1Cache - action . . . 40
4.11 moesi_cmp_directory - L1Cache - Event . . . 41
4.12 moesi_cmp_directory - L1Cache - table . . . 42
4.13 moesi_cmp_directory - L2Cache - State . . . 43
4.14 moesi_cmp_directory - L2Cache - action . . . 44
4.15 moesi_cmp_directory - L2Cache - Event . . . 45
4.16 moesi_cmp_directory - L2Cache - table- part 1/4 . . . 46
4.17 moesi_cmp_directory - L2Cache - table- part 2/4 . . . 47
4.18 moesi_cmp_directory - L2Cache - table- part 3/4 . . . 48
4.19 moesi_cmp_directory - L2Cache - table- part 4/4 . . . 49

5.1 moesi_hammer - Directory - State . . . 51
5.2 moesi_hammer - Directory - action . . . 52
5.3 moesi_hammer - Directory - Event . . . 53
5.4 moesi_hammer - Directory - table- part 1/2 . . . 54
5.5 moesi_hammer - Directory - table- part 2/2 . . . 55
5.6 moesi_hammer - DMA - State . . . 56
5.7 moesi_hammer - DMA - action . . . 56
5.8 moesi_hammer - DMA - Event . . . 56
5.9 moesi_hammer - DMA - table . . . 57
5.10 moesi_hammer - L1Cache - State . . . 58
5.11 moesi_hammer - L1Cache - action . . . 59
5.12 moesi_hammer - L1Cache - Event . . . 60
5.13 moesi_hammer - L1Cache - table- part 1/2 . . . 61
5.14 moesi_hammer - L1Cache - table- part 2/2 . . . 62

Chapter 1

mesi_three_level

State	Description
I	dir is the owner and memory is up
ID	Intermediate state for DMA_READ when in I
ID W	Intermediate state for DMA_WRITE when in I
IM	Intermediate State I¿M
M	memory copy may be stale, i.e. other modified copies may exist
MI	Intermediate State M¿I
M DRD	Intermediate State when there is a dma read
M DRDI	Intermediate State when there is a dma read
M DWR	Intermediate State when there is a dma write
M DWRI	Intermediate State when there is a dma write

Table 1.1: mesi_three_level - Directory - State

action	Function name	Description
aa	aa_sendAck	Send ack to L2
a	a_sendAck	Send ack to L2
da	da_sendDMAAck	Send Ack to DMA controller
drp	drp_sendDMAData	Send Data to DMA controller from incoming PUTX
dr	dr_sendDMAData	Send Data to DMA controller from directory
d	d_sendData	Send data to requestor
inv	inv_sendCacheInvalidate	Invalidate a cache block
j	j_popIncomingRequestQueue	Pop incoming request queue
kd	kd_wakeUpDependents	wake
k	k_popIncomingResponseQueue	Pop incoming request queue
q	l_popMemQueue	Pop off
p	p_popIncomingDMARequestQueue	Pop incoming DMA queue
qf	qf_queueMemoryFetchRequest	Queue off
qfd	qf_queueMemoryFetchRequestDMA	Queue off
qw	qw_queueMemoryWBRequest	Queue off
qwp	qw_queueMemoryWBRequest_partial	Queue off
qwt	qw_queueMemoryWBRequest_partialTBE	Queue off
v	v_allocateTBE	Allocate TBE
w	w_deallocateTBE	Deallocate TBE
zz	zz_recycleDMAQueue	recycle DMA queue
z	z_stallAndWaitRequest	recycle request queue

Table 1.2: mesi_three_level - Directory - action

Event	Description
CleanReplacement	Clean Replacement in L2 cache
Data	writeback data arrives
DMA READ	A DMA Read memory request
DMA WRITE	A DMA Write memory request
Fetch	A memory fetch arrives
Memory Ack	Writeback Ack from memory arrives
Memory Data	Fetched data from memory arrives

Table 1.3: mesi_three_level - Directory - Event

	Fetch	Data	Memory Data	Memory Ack	DMA READ	DMA WRITE	CleanReplacement
I	qf j / IM				qfd j / ID	qwp j / ID W	
ID	z	z	dr q kd / I		zz	zz	
ID W	z	z		da q kd / I	zz	zz	
M	inv z	qw k / MI			inv j / M DRD	v inv j / M DWR	a k kd / I
IM	z	z	d q kd / M		zz	zz	
MI	z	z		aa q kd / I	zz	zz	
M DRD		drp qw k / M DRDI			zz	zz	
M DRDI	z	z		aa q kd / I	zz	zz	
M DWR		qwt k / M DWRI			zz	zz	
M DWRI	z	z		aa da w q kd / I	zz	zz	

Table 1.4: mesi_three_level - Directory - table

State	Description
BUSY RD	Busy
BUSY WR	Busy
READY	Ready to accept a new request

Table 1.5: mesi_three_level - DMA - State

action	Function name	Description
a	a_ackCallback	Notify dma controller that write request completed
d	d_dataCallback	Write data to dma sequencer
p	p_popRequestQueue	Pop request queue
p	p_popResponseQueue	Pop request queue
s	s_sendReadRequest	Send a DMA read request to memory
s	s_sendWriteRequest	Send a DMA write request to memory

Table 1.6: mesi_three_level - DMA - action

Event	Description
Ack	DMA write to memory completed
Data	Data from a DMA memory read
ReadRequest	A new read request
WriteRequest	A new write request

Table 1.7: mesi_three_level - DMA - Event

	ReadRequest	WriteRequest	Data	Ack
READY	s p / BUSY RD	s p / BUSY WR		
BUSY RD			d p / READY	
BUSY WR				a p / READY

Table 1.8: mesi_three_level - DMA - table

State	Description
E	
I	
IM	
Inst IS	
IS	
M	
S	
SM	

Table 1.9: mesi_three_level - L0Cache - State

action	Function name	Description
a	a_issueGETS	Issue GETS
b	b_issueGETX	Issue GETX
c	c_issueUPGRADE	Issue GETX
f	ff_deallocateCacheBlock	Deallocate L1 cache block.
fi	fi_sendInvAck	send data to the L2 cache
cc	forward_eviction_to_cpu	sends eviction information to the processor
f	f_sendDataToL1	send data to the L2 cache
g	g_issuePUTX	send data to the L2 cache
hx	hhx_store_hit	If not prefetch, notify sequencer that store completed.
h	hh_store_hit	If not prefetch, notify sequencer that store completed.
hxi	hx_ifetch_hit	notify sequencer the ifetch completed.
hxd	hx_load_hit	notify sequencer the load completed.
hi	h_ifetch_hit	If not prefetch, notify sequencer the ifetch completed.
hd	h_load_hit	If not prefetch, notify sequencer the load completed.
i	i_allocateTBE	Allocate TBE (number of invalidates=0)
kd	kd_wakeUpDependents	wake
k	k_popMandatoryQueue	Pop mandatory queue.
l	l_popRequestQueue	Pop incoming request queue and profile the delay within this virtual network
o	oo_allocateDCacheBlock	Set L1 D
o	o_popIncomingResponseQueue	Pop Incoming Response queue and profile the delay within this virtual network
p	pp_allocateICacheBlock	Set L1 I
s	s_deallocateTBE	Deallocate TBE
udh	uu_profileDataHit	Profile the demand miss
ud	uu_profileDataMiss	Profile the demand miss
uih	uu_profileInstHit	Profile the demand miss
ui	uu_profileInstMiss	Profile the demand miss
u	u_writeDataToCache	Write data to cache
ui	u_writeInstToCache	Write data to cache
z	z_stallAndWaitMandatoryQueue	recycle cpu request queue

Table 1.10: mesi_three_level - L0Cache - action

Event	Description
Ack	Ack for processor
Ack all	Last ack for processor
Data	Data for processor
Data Exclusive	Data for processor
Fwd GETS	GETS from other processor
Fwd GETX	GETX from other processor
Fwd GET INSTR	GET_INSTR from other processor
Ifetch	I
Inv	Invalidate request from L2 bank
L0 Replacement	L0 Replacement
Load	Load request from the home processor
Store	Store request from the home processor
WB Ack	Ack for replacement

Table 1.11: mesi_three_level - L0Cache - Event

	Load	Ifetch	Store	Inv	L0 Replacement	Fwd GETX	Fwd GETS	Fwd GET INSTR	Data	Data Exclusive	Ack	Ack all	WB Ack
I	o i a ud k / I S	p i a ui k / I nst IS	o i b ud k / I M	fi l									
S	hd udh k	hi uih k	i c ud k / SM	cc fi f l / I	cc f / I								
E	hd udh k	hi uih k	h udh k / M	cc fi f l / I	cc g f / I	cc fi f l / I	f l / S	f l / S					
M	hd udh k	hi uih k	h udh k	cc f f l / I	cc g f / I	cc f f l / I	f l / S	f l / S					
Inst IS	z	z	z	fi l	z				ui hxi s o kd / S	ui hxi s o kd / E			
IS	z	z	z	fi l	z				u hxd s o kd / S	u hxd s o kd / E			
IM	z	z	z	fi l	z					u hx s o kd / M			
SM	z	z	z	fi l / IM	z					u hx s o kd / M			

Table 1.12: mesi_three_level - L0Cache - table

State	Description
E	a L1 cache entry Exclusive
EE	a L1 cache entry Exclusive
E IL0	
I	a L1 cache entry Idle
IM	L1 idle, issued GETX, have not seen response yet
IS	L1 idle, issued GETS, have not seen response yet
M	a L1 cache entry Modified
MM	a L1 cache entry Modified
MM IL0	
M I	L1 replacing, waiting for ACK
M IL0	
S	a L1 cache entry Shared
SINK WB ACK	This is to sink WB_Acks from L2
SM	L1 idle, issued GETX, have not seen response yet
SM IL0	
SS	a L1 cache entry Shared
S IL0	

Table 1.13: mesi_three_level - L1Cache - State

action	Function name	Description
a	a_issueGETS	Issue GETS
b	b_issueGETX	Issue GETX
c	c_issueUPGRADE	Issue GETX
d2t	d2t_sendDataToL2_fromTBE	send data to the L2 cache
d2	d2_sendDataToL2	send data to the L2 cache because of M downgrade
dt	dt_sendDataToRequestor_fromTBE	send data to requestor
d	d_sendDataToRequestor	send data to requestor
e	e_sendAckToRequestor	send invalidate ack to requestor (could be L2 or L1)
f	ff_deallocateCacheBlock	Deallocate L1 cache block.
fi	fi_sendInvAck	send data to the L2 cache
cc	forward_eviction_to_L0	sends eviction information to the processor
ft	ft_sendDataToL2_fromTBE	send data to the L2 cache
f	f_sendDataToL2	send data to the L2 cache
g	g_issuePUTX	send data to the L2 cache
h	hh_xdata_to_l0	If not prefetch, notify sequencer that store completed.
h	h_data_to_l0	If not prefetch, send data to the L0 cache.
i	i_allocateTBE	Allocate TBE (number of invalidates=0)
j	jj_sendExclusiveUnblock	send unblock to the L2 cache
j	j_sendUnblock	send unblock to the L2 cache
kd	kd_wakeUpDependents	wake
k	k_popL0RequestQueue	Pop mandatory queue.
l	l_popL2RequestQueue	Pop incoming request queue and profile the delay within this virtual network
o	oo_allocateCacheBlock	Set cache tag equal to tag of block B.
o	o_popL2ResponseQueue	Pop Incoming Response queue and profile the delay within this virtual network
q	q_updateAckCount	Update ack count
s	s_deallocateTBE	Deallocate TBE
uh	uu_profileHit	Profile the demand hit
um	uu_profileMiss	Profile the demand miss
ureql0	u_writeDataFromL0Request	Write data to cache
uresl0	u_writeDataFromL0Response	Write data to cache
uresl2	u_writeDataFromL2Response	Write data to cache
z0	z0_stallAndWaitL0Queue	recycle L0 request queue
z2	z2_stallAndWaitL2Queue	recycle L2 request queue

Table 1.14: mesi_three_level - L1Cache - action

Event	Description
Ack	Ack for processor
Ack all	Last ack for processor
Data	Data for processor
DataS fromL1	data for GETS request, need to unblock directory
Data all Acks	Data for processor, all acks
Data Exclusive	Data for processor
Fwd GETS	GETS from other processor
Fwd GETX	GETX from other processor
Inv	Invalidate request from L2 bank
L0 Ack	Ack for processor
L0 DataAck	
L0 Invalidate Else	
L0 Invalidate Own	
L1 Replacement	
Load	Load request
Store	Store request
WB Ack	Ack for replacement
WriteBack	Writeback request

Table 1.15: mesi_three_level - L1Cache - Event

	Load	Store	WriteBack	L0 DataAck	Inv	L0 Invalidate Own	L0 Invalidate Else	L1 Replacement	Fwd GETX	Fwd GETS	Data	Data Exclusive	DataS from L1	Data all Acks	L0 Ack	Ack	Ack all	WB Ack
I	o i a um k / IS	o i b um k / IM			fi l													
S	h uh k	i c um k / SM				cc / S IL0	cc / S IL0											
SS	h uh k / S	i c um k / SM			fi f l / I			f / I										
E			ureql0 k / MM			cc / E IL0	cc / E IL0											
EE	h uh k / E	h uh k / M			fi f l / I			i g f / M I	d f l / I	d d2 l / S S								
M			ureql0 k / MM			cc / M IL0	cc / M IL0											
MM	h uh k / M	h uh k / M			f f l / I			i g f / M I	d f l / I	d d2 l / S S								
IS	z0	z0			z2			z0	z2	z2		uresl2 h j s o kd / E	uresl2 j h s o kd / S	uresl2 h s o kd / S				
IM	z0	z0			fi l			z0			uresl2 q o / SM			uresl2 h j s o kd / M		q o		
SM	z0	z0			fi l / IM	z0	cc / SM IL0	z0								q o	j h s o kd / M	
M I	z0	z0			ft l / SINK WB ACK			z0	dt l / SINK WB ACK	dt d2t l / SINK WB ACK								s o f kd / I
SINK WB ACK	z0	z0			fi l			z0										s o f kd / I
S IL0	z0	z0			z2	z0	z2	z0	z2	z2					k kd / SS			
E IL0	z0	z0	ureql0 k k d / MM IL0	uresl0 k k d / MM	z2	z0	z2	z0	z2	z2					k kd / EE			
M IL0	z0	z0	ureql0 k k d / MM IL0	uresl0 k k d / MM	z2	z0	z2	z0	z2	z2					k kd / MM			
MM IL0	z0	z0			z2			z0	z2	z2					k kd / MM			
SM IL0						z0	z2								k kd / IM			

Table 1.16: mesi_three_level - L1Cache - table

State	Description
IM	L2 idle, got L1_GETX, issued memory fetch, have not seen response(s) yet
IS	L2 idle, got L1_GET_INSTR or multiple L1_GETS, issued memory fetch, have not seen response yet
ISS	L2 idle, got single L1_GETS, issued memory fetch, have not seen response yet
I I	L2 replacing clean data, need to inv sharers and then drop data
M	L2 cache entry Modified, not present in any L1s
MCT I	L2 cache replacing, clean in L2, getting data or ack from exclusive
MT	L2 cache entry Modified in a local L1, assume L2 copy stale
MT I	L2 cache replacing, getting data from exclusive
MT IB	Blocked for L1_GETS from MT, got unblock, waiting for data
MT IIB	Blocked for L1_GETS from MT, waiting for unblock and data
MT MB	Blocked for L1_GETX from MT
MT SB	Blocked for L1_GETS from MT, got data, waiting for unblock
M I	L2 cache replacing, have all acks, sent dirty data to memory, waiting for ACK from memory
NP	Not present in either cache
SS	L2 cache entry Shared, also present in one or more L1s
SS MB	Blocked for L1_GETX from SS
S I	L2 replacing dirty data, collecting acks from L1s

Table 1.17: mesi_three_level - L2Cache - State

action	Function name	Description
a	a_issueFetchToMemory	fetch data from memory
b	b_forwardRequestToExclusive	Forward request to the exclusive L1
ct	ct_exclusiveReplacementFromTBE	Send data to memory
cc	c_exclusiveCleanReplacement	Send ack to memory for clean replacement
c	c_exclusiveReplacement	Send data to memory
dd	dd_sendExclusiveDataToRequestor	Send data from cache to reqeustor
ds	ds_sendSharedDataToRequestor	Send data from cache to reqeustor
d	d_sendDataToRequestor	Send data from cache to reqeustor
ee	ee_sendDataToGetXRequestor	Send data from cache to GetX ID
ex	ex_sendExclusiveDataToGetSRequestors	Send data from cache to all GetS IDs
e	e_sendDataToGetSRequestors	Send data from cache to all GetS IDs
fwm	fwm_sendFwdInvToSharersMinusRequestor	invalidate sharers for request, requestor is sharer
fw	fw_sendFwdInvToSharers	invalidate sharers for request
f	f_sendInvToSharers	invalidate sharers for L2 replacement
i	i_allocateTBE	Allocate TBE for request
j	jj_popL1RequestQueue	Pop incoming L1 request queue
kd	kd_wakeUpDependents	wake
k	kk_removeRequestSharer	Remove L1 Request sharer from list
k	k_popUnblockQueue	Pop incoming unblock queue
l	ll_clearSharers	Remove all L1 sharers from list
mu	mmu_markExclusiveFromUnblock	set the exclusive owner
m	mm_markExclusive	set the exclusive owner
mr	mr_writeDataToCacheFromRequest	Write data from response queue to cache
m	m_writeDataToCache	Write data from response queue to cache
nu	nnu_addSharerFromUnblock	Add L1 sharer to list
n	nn_addSharer	Add L1 sharer to list
o	o_popIncomingResponseQueue	Pop Incoming Response queue
q	qq_allocateL2CacheBlock	Set L2 cache tag equal to tag of block B.
qq	qq_writeDataToTBE	Write data from response queue to TBE
q	q_updateAck	update pending ack count
r	rr_deallocateL2CacheBlock	Deallocate L2 cache block. Sets the cache to not present, allowing a replacement in parallel with a fetch.
set	set_setMRU	set the MRU entry
s	ss_recordGetSL1ID	Record L1 GetS for load response
s	s_deallocateTBE	Deallocate external TBE
ts	ts_sendInvAckToUpgrader	Send ACK to upgrader
t	t_sendWBAck	Send writeback ACK
uh	uu_profileHit	Profile the demand hit
um	uu_profileMiss	Profile the demand miss
x	xx_recordGetXL1ID	Record L1 GetX for store response
zn	zn_recycleResponseNetwork	recycle memory request
zz	zz_stallAndWaitL1RequestQueue	recycle L1 request queue

Table 1.18: mesi_three_level - L2Cache - action

Event	Description
Ack	writeback ack
Ack all	writeback ack
Exclusive Unblock	Unblock from L1 requestor
L1 GETS	a L1D GETS request for a block maped to us
L1 GETX	a L1D GETX request for a block maped to us
L1 GET INSTR	a L1I GET INSTR request for a block maped to us
L1 PUTX	L1 replacing data
L1 PUTX old	L1 replacing data, but no longer sharer
L1 UPGRADE	a L1D GETX request for a block maped to us
L2 Replacement	L2 Replacement
L2 Replacement clean	L2 Replacement, but data is clean
Mem Ack	ack from memory
Mem Data	data from memory
MEM Inv	Invalidation from directory
Unblock	Unblock from L1 requestor
WB Data	data from L1
WB Data clean	clean data from L1

Table 1.19: mesi_three_level - L2Cache - Event

	L1 GET INSTR	L1 GETS	L1 GETX	L1 UPGRADE	L1 PUTX	L1 PUTX old	L2 Replacement	L2 Replacement clean	Mem Data	Mem Ack	WB Data	WB Data clean	Ack	Ack all	Unblock	Exclusive Unblock	MEM Inv
NP	q l n i s a um j / IS	q l n i s a um j / ISS	q l i x a u m j / IM		t j	t j											o
SS	ds n set uh j	ds n set uh j	d fwm set uh j / SS MB	fwm ts set uh j / SS MB	t j	t j	i f r / S I	i f r / I I									i f r / S I
M	d n set uh j / SS	dd set uh j / MT MB	d set uh j / MT MB		t j	t j	i c r / M I	i cc r / M I									i c r / M I
MT	b um set j / MT IIB	b um set j / MT IIB	b um set j / MT MB		l mr t j / M	t j	i f r / MT I	i f r / MCT I									i f r / MT I
M I	zz	zz	zz	zz	t j	t j				s o kd / NP							o
MT I	zz	zz	zz	zz	zz	zz					qq ct o / M I	ct o / M I		ct o / M I			o
MCT I	zz	zz	zz	zz	zz	zz					qq ct o / M I	cc o / M I		cc o / M I			o
I I	zz	zz	zz	zz	t j	t j							q o	cc o / M I			o
S I	zz	zz	zz	zz	t j	t j							q o	ct o / M I			o
ISS	n s um j / IS	n s um j / IS	zz		t j	t j	zz	zz	m ex s o / MT MB								zn
IS	n s um j	n s um j	zz		t j	t j	zz	zz	m e s o kd / SS								zn
IM	zz	zz	zz		t j	t j	zz	zz	m ee s o / MT MB								zn
SS MB	zz	zz	zz	zz	zz	zz	zz	zz								mu k kd / MT	zn
MT MB	zz	zz	zz	zz	zz	zz	zz	zz								mu k kd / MT	zn
MT IIB	zz	zz	zz	zz	zz	zz	zz	zz			m o / MT SB	m o / MT SB			nu k / MT IB		zn
MT IB	zz	zz	zz	zz	t j	t j	zz	zz			m o kd / SS	m o kd / SS					zn
MT SB	zz	zz	zz	zz	t j	t j	zz	zz							nu k kd / SS		zn

Table 1.20: mesi_three_level - L2Cache - table

Chapter 2

mesi_two_level

State	Description
I	dir is the owner and memory is up
ID	Intermediate state for DMA_READ when in I
ID W	Intermediate state for DMA_WRITE when in I
IM	Intermediate State I¿M
M	memory copy may be stale, i.e. other modified copies may exist
MI	Intermediate State M¿I
M DRD	Intermediate State when there is a dma read
M DRDI	Intermediate State when there is a dma read
M DWR	Intermediate State when there is a dma write
M DWRI	Intermediate State when there is a dma write

Table 2.1: mesi_two_level - Directory - State

action	Function name	Description
aa	aa_sendAck	Send ack to L2
a	a_sendAck	Send ack to L2
da	da_sendDMAAck	Send Ack to DMA controller
drp	drp_sendDMAData	Send Data to DMA controller from incoming PUTX
dr	dr_sendDMAData	Send Data to DMA controller from directory
d	d_sendData	Send data to requestor
inv	inv_sendCacheInvalidate	Invalidate a cache block
j	j_popIncomingRequestQueue	Pop incoming request queue
kd	kd_wakeUpDependents	wake
k	k_popIncomingResponseQueue	Pop incoming request queue
q	l_popMemQueue	Pop off
p	p_popIncomingDMARequestQueue	Pop incoming DMA queue
qf	qf_queueMemoryFetchRequest	Queue off
qfd	qf_queueMemoryFetchRequestDMA	Queue off
qw	qw_queueMemoryWBRequest	Queue off
qwp	qw_queueMemoryWBRequest_partial	Queue off
qwt	qw_queueMemoryWBRequest_partialTBE	Queue off
v	v_allocateTBE	Allocate TBE
w	w_deallocateTBE	Deallocate TBE
zz	zz_recycleDMAQueue	recycle DMA queue
z	z_stallAndWaitRequest	recycle request queue

Table 2.2: mesi_two_level - Directory - action

Event	Description
CleanReplacement	Clean Replacement in L2 cache
Data	writeback data arrives
DMA READ	A DMA Read memory request
DMA WRITE	A DMA Write memory request
Fetch	A memory fetch arrives
Memory Ack	Writeback Ack from memory arrives
Memory Data	Fetched data from memory arrives

Table 2.3: mesi_two_level - Directory - Event

	Fetch	Data	Memory Data	Memory Ack	DMA READ	DMA WRITE	CleanReplacement
I	qf j / IM				qfd j / ID	qwp j / ID W	
ID	z	z	dr q kd / I		zz	zz	
ID W	z	z		da q kd / I	zz	zz	
M	inv z	qw k / MI			inv j / M DRD	v inv j / M DWR	a k kd / I
IM	z	z	d q kd / M		zz	zz	
MI	z	z		aa q kd / I	zz	zz	
M DRD		drp qw k / M DRDI			zz	zz	
M DRDI	z	z		aa q kd / I	zz	zz	
M DWR		qwt k / M DWRI			zz	zz	
M DWRI	z	z		aa da w q kd / I	zz	zz	

Table 2.4: mesi_two_level - Directory - table

State	Description
BUSY RD	Busy
BUSY WR	Busy
READY	Ready to accept a new request

Table 2.5: mesi_two_level - DMA - State

action	Function name	Description
a	a_ackCallback	Notify dma controller that write request completed
d	d_dataCallback	Write data to dma sequencer
p	p_popRequestQueue	Pop request queue
p	p_popResponseQueue	Pop request queue
s	s_sendReadRequest	Send a DMA read request to memory
s	s_sendWriteRequest	Send a DMA write request to memory

Table 2.6: mesi_two_level - DMA - action

Event	Description
Ack	DMA write to memory completed
Data	Data from a DMA memory read
ReadRequest	A new read request
WriteRequest	A new write request

Table 2.7: mesi_two_level - DMA - Event

	ReadRequest	WriteRequest	Data	Ack
READY	s p / BUSY RD	s p / BUSY WR		
BUSY RD			d p / READY	
BUSY WR				a p / READY

Table 2.8: mesi_two_level - DMA - table

State	Description
E	a L1 cache entry Exclusive
I	a L1 cache entry Idle
IM	L1 idle, issued GETX, have not seen response yet
IS	L1 idle, issued GETS, have not seen response yet
IS I	L1 idle, issued GETS, saw Inv before data because directory doesn't block on GETS hit
M	a L1 cache entry Modified
M I	L1 replacing, waiting for ACK
NP	Not present in either cache
PF IM	Issued GETX, have not seen response yet
PF IS	Issued GETS, have not seen response yet
PF IS I	Issued GETs, saw inv before data
PF SM	Issued GETX, received data, waiting for acks
S	a L1 cache entry Shared
SINK WB ACK	This is to sink WB_Acks from L2
SM	L1 idle, issued GETX, have not seen response yet

Table 2.9: mesi_two_level - L1Cache - State

action	Function name	Description
ai	ai_issueGETINSTR	Issue GETINSTR
a	a_issueGETS	Issue GETS
b	b_issueGETX	Issue GETX
c	c_issueUPGRADE	Issue GETX
d2t	d2t_sendDataToL2_fromTBE	send data to the L2 cache
d2	d2_sendDataToL2	send data to the L2 cache because of M downgrade
dg	dg_invalidate_sc	Invalidate store conditional as the cache lost permissions
dt	dt_sendDataToRequestor_fromTBE	send data to requestor
d	d_sendDataToRequestor	send data to requestor
e	e_sendAckToRequestor	send invalidate ack to requestor (could be L2 or L1)
f	ff_deallocateL1CacheBlock	Deallocate L1 cache block. Sets the cache to not present, allowing a replacement in parallel with a fetch.
fi	fi_sendInvAck	send data to the L2 cache
cc	forward_eviction_to_cpu	sends eviction information to the processor
ft	ft_sendDataToL2_fromTBE	send data to the L2 cache
f	f_sendDataToL2	send data to the L2 cache
g	g_issuePUTX	send data to the L2 cache
hx	hhx_store_hit	Notify sequencer that store completed.
h	hh_store_hit	Notify sequencer that store completed.
hx	hx_load_hit	Notify sequencer the load completed.
hi	h_ifetch_hit	Notify sequencer the instruction fetch completed.
hd	h_load_hit	Notify sequencer the load completed.
i	i_allocateTBE	Allocate TBE (isPrefetch=0, number of invalidates=0)
j	jj_sendExclusiveUnblock	send unblock to the L2 cache
j	j_sendUnblock	send unblock to the L2 cache
kd	kd_wakeUpDependents	wake
k	k_popMandatoryQueue	Pop mandatory queue.
l	l_popRequestQueue	Pop incoming request queue and profile the delay within this virtual network
mp	mp_markPrefetched	Write data from response queue to cache
o	oo_allocateL1DCacheBlock	Set L1 D
o	o_popIncomingResponseQueue	Pop Incoming Response queue and profile the delay within this virtual network
pai	pai_issuePfGETINSTR	Issue GETINSTR for prefetch request
pa	pa_issuePfGETS	Issue prefetch GETS
pb	pb_issuePfGETX	Issue prefetch GETX
po	po_observeMiss	Inform the prefetcher about the miss
ppm	ppm_observePfMiss	Inform the prefetcher about the partial miss
p	pp_allocateL1ICacheBlock	Set L1 I
pq	pq_popPrefetchQueue	Pop the prefetch request queue
q	q_updateAckCount	Update ack count
s	s_deallocateTBE	Deallocate TBE
udh	uu_profileDataHit	Profile the demand hit
udm	uu_profileDataMiss	Profile the demand miss
uih	uu_profileInstHit	Profile the demand hit
uim	uu_profileInstMiss	Profile the demand miss
u	u_writeDataToL1Cache	Write data to cache
z	z_stallAndWaitMandatoryQueue	recycle L1 request queue

Table 2.10: mesi_two_level - L1Cache - action

Event	Description
Ack	Ack for processor
Ack all	Last ack for processor
Data	Data for processor
DataS fromL1	data for GETS request, need to unblock directory
Data all Acks	Data for processor, all acks
Data Exclusive	Data for processor
Fwd GETS	GETS from other processor
Fwd GETX	GETX from other processor
Fwd GET INSTR	GET_INSTR from other processor
Ifetch	I
Inv	Invalidate request from L2 bank
L1 Replacement	L1 Replacement
Load	Load request from the home processor
PF Ifetch	instruction fetch request from prefetcher
PF Load	load request from prefetcher
PF Store	exclusive load request from prefetcher
Store	Store request from the home processor
WB Ack	Ack for replacement

Table 2.11: mesi_two_level - L1Cache - Event

	Load	Ifetch	Store	Inv	L1 Replacement	Fwd GETX	Fwd GETS	Fwd GET INSTR	Data	Data Exclusive	DataS from L1	Data all Acks	Ack	Ack all	WB Ack	PF Load	PF Ifetch	PF Store
NP	o i a udm po k / IS	p i ai uim po k / IS	o i b udm po k / IM	fi l	f											o i pa pq / PF IS	p i pai pq / PF IS	o i pb pq / PF IM
I	o i a udm po k / IS	p i ai uim po k / IS	o i b udm po k / IM	fi l	f											o i pa pq / PF IS	p i pai pq / PF IS	o i pb pq / PF IM
S	hd udh k	hi uih k	i c udm k / SM	cc fi l / I	cc f / I											pq	pq	pq
E	hd udh k	hi uih k	h udh k / M	cc fi l / I	cc i g f / M I	cc d l / I	d d2 l / S	d d2 l / S								pq	pq	pq
M	hd udh k	hi uih k	h udh k	cc f l / I	cc i g f / M I	cc d l / I	d d2 l / S	d d2 l / S								pq	pq	pq
IS	z	z	z	fi l / IS I	z					u hx j s o kd / E	u j hx s o kd / S	u hx s o k d / S				pq	pq	pq
IM	z	z	z	fi l	z				u q o / SM			u hx j s o kd / M	q o			pq	pq	pq
SM	z	z	z	cc fi dg l / IM	z								q o	j hx s o k d / M		pq	pq	pq
IS I	z	z	z	fi l	z					u hx j s o kd / E	u j hx s o kd / I	u hx s o k d / I				pq	pq	pq
M I	z	z	z	ft l / SINK WB ACK	z	dt l / SINK WB ACK	dt d2t l / SINK WB ACK	dt d2t l / SINK WB ACK							s o kd / I	pq	pq	pq
SINK WB ACK	z	z	z	fi l	z										s o kd / I	pq	pq	pq
PF IS	udm ppm k / IS	udm ppm k / IS	z	fi l / PF IS I	z					u j s mp o kd / E	u j s o kd / S	u s mp o k d / S				pq	pq	pq
PF IM	z	z	udm ppm k / IM	fi l	z				u q o / PF SM			u j s mp o kd / M	q o			pq	pq	pq
PF SM	z	z	udm ppm k / SM	fi l / PF IM	z								q o	j s mp o k d / M				
PF IS I	udm ppm k / IS I		z	fi l	z					u j s o kd / E	j s o kd / I	s o kd / I						

Table 2.12: mesi_two_level - L1Cache - table

State	Description
IM	L2 idle, got L1_GETX, issued memory fetch, have not seen response(s) yet
IS	L2 idle, got L1_GET_INSTR or multiple L1_GETS, issued memory fetch, have not seen response yet
ISS	L2 idle, got single L1_GETS, issued memory fetch, have not seen response yet
I I	L2 replacing clean data, need to inv sharers and then drop data
M	L2 cache entry Modified, not present in any L1s
MCT I	L2 cache replacing, clean in L2, getting data or ack from exclusive
MT	L2 cache entry Modified in a local L1, assume L2 copy stale
MT I	L2 cache replacing, getting data from exclusive
MT IB	Blocked for L1_GETS from MT, got unblock, waiting for data
MT IIB	Blocked for L1_GETS from MT, waiting for unblock and data
MT MB	Blocked for L1_GETX from MT
MT SB	Blocked for L1_GETS from MT, got data, waiting for unblock
M I	L2 cache replacing, have all acks, sent dirty data to memory, waiting for ACK from memory
NP	Not present in either cache
SS	L2 cache entry Shared, also present in one or more L1s
SS MB	Blocked for L1_GETX from SS
S I	L2 replacing dirty data, collecting acks from L1s

Table 2.13: mesi_two_level - L2Cache - State

action	Function name	Description
a	a_issueFetchToMemory	fetch data from memory
b	b_forwardRequestToExclusive	Forward request to the exclusive L1
ct	ct_exclusiveReplacementFromTBE	Send data to memory
cc	c_exclusiveCleanReplacement	Send ack to memory for clean replacement
c	c_exclusiveReplacement	Send data to memory
dd	dd_sendExclusiveDataToRequestor	Send data from cache to reqeustor
ds	ds_sendSharedDataToRequestor	Send data from cache to reqeustor
d	d_sendDataToRequestor	Send data from cache to reqeustor
ee	ee_sendDataToGetXRequestor	Send data from cache to GetX ID
ex	ex_sendExclusiveDataToGetSRequestors	Send data from cache to all GetS IDs
e	e_sendDataToGetSRequestors	Send data from cache to all GetS IDs
fwm	fwm_sendFwdInvToSharersMinusRequestor	invalidate sharers for request, requestor is sharer
fw	fw_sendFwdInvToSharers	invalidate sharers for request
f	f_sendInvToSharers	invalidate sharers for L2 replacement
i	i_allocateTBE	Allocate TBE for request
j	jj_popL1RequestQueue	Pop incoming L1 request queue
kd	kd_wakeUpDependents	wake
k	kk_removeRequestSharer	Remove L1 Request sharer from list
k	k_popUnblockQueue	Pop incoming unblock queue
l	ll_clearSharers	Remove all L1 sharers from list
mu	mmu_markExclusiveFromUnblock	set the exclusive owner
m	mm_markExclusive	set the exclusive owner
mr	mr_writeDataToCacheFromRequest	Write data from response queue to cache
m	m_writeDataToCache	Write data from response queue to cache
nu	nnu_addSharerFromUnblock	Add L1 sharer to list
n	nn_addSharer	Add L1 sharer to list
o	o_popIncomingResponseQueue	Pop Incoming Response queue
q	qq_allocateL2CacheBlock	Set L2 cache tag equal to tag of block B.
qq	qq_writeDataToTBE	Write data from response queue to TBE
q	q_updateAck	update pending ack count
r	rr_deallocateL2CacheBlock	Deallocate L2 cache block. Sets the cache to not present, allowing a replacement in parallel with a fetch.
set	set_setMRU	set the MRU entry
s	ss_recordGetSL1ID	Record L1 GetS for load response
s	s_deallocateTBE	Deallocate external TBE
ts	ts_sendInvAckToUpgrader	Send ACK to upgrader
t	t_sendWBAck	Send writeback ACK
uh	uu_profileHit	Profile the demand hit
um	uu_profileMiss	Profile the demand miss
x	xx_recordGetXL1ID	Record L1 GetX for store response
zn	zn_recycleResponseNetwork	recycle memory request
zz	zz_stallAndWaitL1RequestQueue	recycle L1 request queue

Table 2.14: mesi_two_level - L2Cache - action

Event	Description
Ack	writeback ack
Ack all	writeback ack
Exclusive Unblock	Unblock from L1 requestor
L1 GETS	a L1D GETS request for a block maped to us
L1 GETX	a L1D GETX request for a block maped to us
L1 GET INSTR	a L1I GET INSTR request for a block maped to us
L1 PUTX	L1 replacing data
L1 PUTX old	L1 replacing data, but no longer sharer
L1 UPGRADE	a L1D GETX request for a block maped to us
L2 Replacement	L2 Replacement
L2 Replacement clean	L2 Replacement, but data is clean
Mem Ack	ack from memory
Mem Data	data from memory
MEM Inv	Invalidation from directory
Unblock	Unblock from L1 requestor
WB Data	data from L1
WB Data clean	clean data from L1

Table 2.15: mesi_two_level - L2Cache - Event

	L1 GET INSTR	L1 GETS	L1 GETX	L1 UPGRADE	L1 PUTX	L1 PUTX old	L2 Replacement	L2 Replacement clean	Mem Data	Mem Ack	WB Data	WB Data clean	Ack	Ack all	Unblock	Exclusive Unblock	MEM Inv
NP	q l n i s a um j / IS	q l n i s a um j / ISS	q l i x a u m j / IM		t j	t j											o
SS	ds n set uh j	ds n set uh j	d fwm set uh j / SS MB	fwm ts set uh j / SS MB	t j	t j	i f r / S I	i f r / I I									i f r / S I
M	d n set uh j / SS	dd set uh j / MT MB	d set uh j / MT MB		t j	t j	i c r / M I	i cc r / M I									i c r / M I
MT	b um set j / MT IIB	b um set j / MT IIB	b um set j / MT MB		l mr t j / M	t j	i f r / MT I	i f r / MCT I									i f r / MT I
M I	zz	zz	zz	zz	t j	t j				s o kd / NP							o
MT I	zz	zz	zz	zz	zz	zz					qq ct o / M I	ct o / M I		ct o / M I			o
MCT I	zz	zz	zz	zz	zz	zz					qq ct o / M I	cc o / M I		cc o / M I			o
I I	zz	zz	zz	zz	t j	t j							q o	cc o / M I			o
S I	zz	zz	zz	zz	t j	t j							q o	ct o / M I			o
ISS	n s um j / IS	n s um j / IS	zz		t j	t j	zz	zz	m ex s o / MT MB								zn
IS	n s um j	n s um j	zz		t j	t j	zz	zz	m e s o kd / SS								zn
IM	zz	zz	zz		t j	t j	zz	zz	m ee s o / MT MB								zn
SS MB	zz	zz	zz	zz	zz	zz	zz	zz								mu k kd / MT	zn
MT MB	zz	zz	zz	zz	zz	zz	zz	zz								mu k kd / MT	zn
MT IIB	zz	zz	zz	zz	zz	zz	zz	zz			m o / MT SB	m o / MT SB			nu k / MT IB		zn
MT IB	zz	zz	zz	zz	t j	t j	zz	zz			m o kd / SS	m o kd / SS					zn
MT SB	zz	zz	zz	zz	t j	t j	zz	zz							nu k kd / SS		zn

Table 2.16: mesi_two_level - L2Cache - table

Chapter 3

mi_example

State	Description
I	Invalid
ID	Intermediate state for DMA_READ when in I
ID W	Intermediate state for DMA_WRITE when in I
IM	Intermediate state I
M	Modified
MI	Intermediate state M
M DRD	Blocked on an invalidation for a DMA read
M DRDI	Intermediate state M_DRD
M DWR	Blocked on an invalidation for a DMA write
M DWRI	Intermediate state M_DWR

Table 3.1: mi_example - Directory - State

action	Function name	Description
a	a_sendWriteBackAck	Send writeback ack to requestor
b	b_sendWriteBackNack	Send writeback nack to requestor
c	c_clearOwner	Clear the owner field
da	da_sendDMAAck	Send Ack to DMA controller
drp	drp_sendDMAData	Send Data to DMA controller from incoming PUTX
dr	dr_sendDMAData	Send Data to DMA controller from directory
d	d_sendData	Send data to requestor
e	e_ownerIsRequestor	The owner is now the requestor
f	f_forwardRequest	Forward request to owner
inv	inv_sendCacheInvalidate	Invalidate a cache block
i	i_popIncomingRequestQueue	Pop incoming request queue
q	l_popMemQueue	Pop off
lq	l_queueMemoryWBRequest	Write PUTX data to memory
la	l_sendWriteBackAck	Send writeback ack to requestor
p	p_popIncomingDMARequestQueue	Pop incoming DMA queue
qf	qf_queueMemoryFetchRequest	Queue off
qfd	qf_queueMemoryFetchRequestDMA	Queue off
qwp	qw_queueMemoryWBRequest_partial	Queue off
qwt	qw_queueMemoryWBRequest_partialTBE	Queue off
r	r_allocateTbeForDmaRead	Allocate TBE for DMA Read
v	v_allocateTBE	Allocate TBE
v	v_allocateTBEFromRequestNet	Allocate TBE
w	w_deallocateTBE	Deallocate TBE
y	y_recycleDMARequestQueue	recycle dma request queue
z	z_recycleRequestQueue	recycle request queue

Table 3.2: mi_example - Directory - action

Event	Description
DMA READ	A DMA Read memory request
DMA WRITE	A DMA Write memory request
GETS	A GETS arrives
GETX	A GETX arrives
Memory Ack	Writeback Ack from memory arrives
Memory Data	Fetched data from memory arrives
PUTX	A PUTX arrives
PUTX NotOwner	A PUTX arrives

Table 3.3: mi_example - Directory - Event

	GETX	GETS	PUTX	PUTX NotOwner	DMA READ	DMA WRITE	Memory Data	Memory Ack
I	qf e i / IM			b i	r qfd p / ID	v qwp p / ID W		
M	f e i		c v lq i / MI	b i	v inv p / M DRD	v inv p / M DWR		
M DRD	z		drp c lq i / M DRDI					
M DWR	z		qwt c i / M DWRI					
M DWRI	z							la da w q / I
M DRDI	z							la w q / I
IM	z	z	z	z	y	y	d q / M	
MI	z	z	z	z	y	y		la w q / I
ID	z	z	z	z	y	y	dr w q / I	
ID W	z	z	z	z	y	y		da w q / I

Table 3.4: mi_example - Directory - table

State	Description
BUSY RD	Busy
BUSY WR	Busy
READY	Ready to accept a new request

Table 3.5: mi_example - DMA - State

action	Function name	Description
a	a_ackCallback	Notify dma controller that write request completed
d	d_dataCallback	Write data to dma sequencer
p	p_popRequestQueue	Pop request queue
p	p_popResponseQueue	Pop request queue
s	s_sendReadRequest	Send a DMA read request to memory
s	s_sendWriteRequest	Send a DMA write request to memory

Table 3.6: mi_example - DMA - action

Event	Description
Ack	DMA write to memory completed
Data	Data from a DMA memory read
ReadRequest	A new read request
WriteRequest	A new write request

Table 3.7: mi_example - DMA - Event

	ReadRequest	WriteRequest	Data	Ack
READY	s p / BUSY RD	s p / BUSY WR		
BUSY RD			d p / READY	
BUSY WR				a p / READY

Table 3.8: mi_example - DMA - table

State	Description
I	Not Present/Invalid
II	Not Present/Invalid, issued PUT
IM	Issued request for STORE/ATOMIC
IS	Issued request for LOAD/IFETCH
M	Modified
MI	Modified, issued PUT
MII	Modified, issued PUTX, received nack

Table 3.9: mi_example - L1Cache - State

action	Function name	Description
a	a_issueRequest	Issue a request
b	b_issuePUT	Issue a PUT request
e	ee_sendDataFromTBE	Send data from TBE to requestor
e	e_sendData	Send data from cache to requestor
cc	forward_eviction_to_cpu	sends eviction information to the processor
h	h_deallocateL1CacheBlock	deallocate a cache block
i	i_allocateL1CacheBlock	Allocate a cache block
m	m_popMandatoryQueue	Pop the mandatory request queue
n	n_popResponseQueue	Pop the response queue
o	o_popForwardedRequestQueue	Pop the forwarded request queue
ph	p_profileHit	Profile cache miss
pi	p_profileMiss	Profile cache miss
rx	rx_load_hit	External load completed.
r	r_load_hit	Notify sequencer the load completed.
sx	sx_store_hit	External store completed.
s	s_store_hit	Notify sequencer that store completed.
u	u_writeDataToCache	Write data to the cache
v	v_allocateTBE	Allocate TBE
w	w_deallocateTBE	Deallocate TBE
x	x_copyDataFromCacheToTBE	Copy data from cache to TBE
z	z_stall	stall

Table 3.10: mi_example - L1Cache - action

Event	Description
Data	Data from network
Fwd GETX	Forward from network
Ifetch	Ifetch request from processor
Inv	Invalidate request from dir
Load	Load request from processor
Replacement	Replace a block
Store	Store request from processor
Writeback Ack	Ack from the directory for a writeback
Writeback Nack	Nack from the directory for a writeback

Table 3.11: mi_example - L1Cache - Event

	Load	Ifetch	Store	Data	Fwd GETX	Inv	Replacement	Writeback Ack	Writeback Nack
I	v i a pi m / IS	v i a pi m / IS	v i a pi m / IM			o	h		
II	z	z	z				z		w o / I
M	r ph m	r ph m	s ph m		e cc o / I	v b x cc h / MI	v b x cc h / MI		
MI	z	z	z		e o / II	o	z	w o / I	o / MII
MII	z	z	z		e w o / I		z		
IS	z	z	z	u rx w n / M	z	z	z		
IM	z	z	z	u sx w n / M	z	z	z		

Table 3.12: mi_example - L1Cache - table

Chapter 4

moesi_cmp_directory

State	Description
I	Invalid
IS	Blocked, was in idle
M	Modified
MD	In M, waiting for dma ack from L2
MI	Blocked on a writeback
MIS	Blocked on a writeback, but don't remove from sharers when received
MM	Blocked, going to modified
MM DMA	Blocked, going to I
MO	Blocked, going to owner or maybe modified
O	Owner
OD	In O, waiting for dma ack from L2
OI D	In O, going to I, waiting for data
OO	Blocked, was in owned
OS	Blocked on a writeback
OSS	Blocked on a writeback, but don't remove from sharers when received
S	Shared
SS	Blocked, was in shared
XI M	In a stable state, going to I, waiting for the memory controller
XI U	In a stable state, going to I, waiting for an unblock

Table 4.1: moesi_cmp_directory - Directory - State

action	Function name	Description
a	a_sendDMAAck	Send DMA Ack that write completed, along with Inv Ack count
aa	a_sendDMAAck2	Send DMA Ack that write completed, along with Inv Ack count
a	a_sendWriteBackAck	Send writeback ack to requestor
b	b_sendWriteBackNack	Send writeback nack to requestor
c	cc_clearSharers	Clear the sharers field
c	c_clearOwner	Clear the owner field
cc	c_moveOwnerToSharer	Move owner to sharers
d	d_sendDataMsg	Send data to requestor
e	e_ownerIsUnblocker	The owner is now the unblocker
f	f_forwardRequest	Forward request to owner
f	f_forwardRequestDirIsRequestor	Forward request to owner
g	g_sendInvalidations	Send invalidations to sharers, not including the requester
i	i_popIncomingRequestQueue	Pop incoming request queue
j	j_popIncomingUnblockQueue	Pop incoming unblock queue
m	m_addUnlockerToSharers	Add the unlocker to the sharer list
n	n_incrementOutstanding	Increment outstanding requests
o	o_decrementOutstanding	Decrement outstanding requests
d	p_fwdDataToDMA	Send data to requestor
qf	qf_queueMemoryFetchRequest	Queue off
qw	qw_queueMemoryWBRequest	Queue off
/qw	qw_queueMemoryWBRequest2	Queue off
qwmt	qw_queueMemoryWBRequestFromMessageAndTBE	Queue off
q	q_popMemQueue	Pop off
v	v_allocateTBE	Allocate TBE entry
w	w_deallocateTBE	Deallocate TBE entry
z	zz_recycleRequest	Recycle the request queue

Table 4.2: moesi_cmp_directory - Directory - action

Event	Description
Clean Writeback	The final message as part of a PutX/PutS, no data
Data	Data to directory
Dirty Writeback	The final message as part of a PutX/PutS, contains data
DMA ACK	DMA Ack
DMA READ	DMA Read
DMA WRITE	DMA Write
Exclusive Unblock	The processor become the exclusive owner (E or M) of the line
GETS	A GETS arrives
GETX	A GETX arrives
Last Unblock	An unblock message arrives, we're not waiting for any additional unblocks
Memory Ack	Writeback Ack from memory arrives
Memory Data	Fetched data from memory arrives
PUTO	A PUTO arrives
PUTO SHARERS	A PUTO arrives, but don't remove from sharers list
PUTX	A PUTX arrives
Unblock	An unblock message arrives

Table 4.3: moesi_cmp_directory - Directory - Event

	GETX	GETS	PUTX	PUTO	PUTO SHARERS	Unblock	Last Unblock	Exclusive Unblock	Clean Write back	Dirty Write back	Memory Data	Memory Ack	DMA READ	DMA WRITE	DMA ACK	Data
I	qf i / MM	qf i / IS	b i	b i							d q	q	qf i / XI M	/qw a i / XI U		
S	qf g i / MM	qf n i / SS	b i	b i							d q	q	d i	/qw a g i / XI U		
O	f g i / MM	f n i / OO	b i	a i / OS	a i / OSS						d q	q	f i / OD	f g v i / O I D		
M	f i / MM	f i / MO	a i / MI	a i / MI	a i / MIS						d q	q	f i / MD	f g v i / O I D		
IS	z	z	z	z	z	m j / S		c e j / M			d q	q	z	z		
SS	z	qf n i	z	z	z	m o j	m o j / S				d q	q	z	z		
OO	z	f n i	z	z	z	m o j	m o j / O				d q	q	z	z		
MO	z	z	z	z	z	m j / O		c e j / M			d q	q	z	z		
MM	z	z	z	z	z			c e j / M			d q	q	z	z		
MM DMA																
MI	z	z	z	z	z	j / M			c c j / I	c c qw j / I	d q	q	z	z		
MIS	z	z	z	z	z	j / M			cc j / S	cc qw j / S	d q	q	z	z		
OS	z	z	z	z	z	j / O			c j / S	c qw j / S	d q	q	z	z		
OSS	z	z	z	z	z	j / O			cc j / S	cc qw j / S	d q	q	z	z		
XI M	z	z	z	z	z						d q / I	q	z	z		
XI U	z	z	z	z	z			c c j / I				q	z	z		
OI D	z	z	z	z	z								z	z		qwmt aa w j / XI U
OD	z	z	z	z	z								z	z	j / O	
MD	z	z	z	z	z								z	z	j / M	

Table 4.4: moesi_cmp_directory - Directory - table

State	Description
BUSY RD	Busy
BUSY WR	Busy
READY	Ready to accept a new request

Table 4.5: moesi_cmp_directory - DMA - State

action	Function name	Description
a	a_ackCallback	Notify dma controller that write request completed
/d	d_dataCallbackFromTBE	data callback with data from TBE
o	o_checkForCompletion	Check if we have received all the messages required for completion
p	p_popRequestQueue	Pop request queue
p	p_popResponseQueue	Pop request queue
pp	p_popTriggerQueue	Pop trigger queue
s	s_sendReadRequest	Send a DMA read request to memory
s	s_sendWriteRequest	Send a DMA write request to memory
t	t_updateTBEData	Update TBE Data
u	u_sendExclusiveUnblockToDir	send exclusive unblock to directory
u	u_updateAckCount	Update ack count
v	v_allocateTBE	Allocate TBE entry
w	w_deallocateTBE	Deallocate TBE entry

Table 4.6: moesi_cmp_directory - DMA - action

Event	Description
All Acks	All acks received
Data	Data from a DMA memory read
DMA Ack	DMA write to memory completed
Inv Ack	Invalidation Ack from a sharer
ReadRequest	A new read request
WriteRequest	A new write request

Table 4.7: moesi_cmp_directory - DMA - Event

	ReadRequest	WriteRequest	Data	DMA Ack	Inv Ack	All Acks
READY	s v p / BUSY RD	s v p / BUSY WR				
BUSY RD			t /d w p / READY		u o p	/d w pp / READY
BUSY WR				u o p	u o p	a u w pp / READY

Table 4.8: moesi_cmp_directory - DMA - table

State	Description
I	Idle
II	Issued PutX/O, saw Fwd_GETS or Fwd_GETX, waiting for ack
IM	Issued GetX
IS	Issued GetS
M	Modified (dirty)
MI	Issued PutX, waiting for ack
MM	Modified (dirty and locally modified)
MM W	Modified (dirty and locally modified)
M W	Modified (dirty)
O	Owned
OI	Issued PutO, waiting for ack
SM	Issued GetX, received data
S	Shared
OI	Issued PutS, waiting for ack
SM	Issued GetX, we still have an old copy of the line

Table 4.9: moesi_cmp_directory - L1Cache - State

action	Function name	Description
a	a_issueGETS	Issue GETS
b	b_issueGETX	Issue GETX
d	dd_issuePUTO	Issue PUTO
ds	dd_issuePUTS	Issue PUTS
d	d_issuePUTX	Issue PUTX
e	ee_sendDataExclusive	Send data from cache to requestor, don't keep a shared copy
e	e_sendData	Send data from cache to requestor
ee	e_sendDataToL2	Send data from cache to requestor
cc	forward_eviction_to_cpu	sends eviction information to the processor
f	f_sendAck	Send ack from cache to requestor
g	gg_sendUnblockExclusive	Send unblock exclusive to memory
g	g_sendUnblock	Send unblock to memory
h	hh_store_hit	Notify sequencer that store completed.
hx	hx_load_hit	Notify sequencer the load completed.
hi	h_ifetch_hit	Notify the sequencer about ifetch completion.
hd	h_load_hit	Notify sequencer the load completed.
i	ii_allocateL1DCacheBlock	Set L1 D
i	i_allocateTBE	Allocate TBE
j	jj_allocateL1ICacheBlock	Set L1 I
jj	jj_unsetUseTimer	Unset use timer.
j	j_popTriggerQueue	Pop trigger queue.
k	kk_deallocateL1CacheBlock	Deallocate cache block. Sets the cache to invalid, allowing a replacement in parallel with a fetch.
k	k_popMandatoryQueue	Pop mandatory queue.
l	l_popForwardQueue	Pop forwareded request queue.
m	mm_decrementNumberOfMessages	Decrement the number of messages for which we're waiting
m	m_decrementNumberOfMessages	Decrement the number of messages for which we're waiting
n	n_popResponseQueue	Pop response queue
o	o_checkForCompletion	Check if we have received all the messages required for completion
oo	o_scheduleUseTimeout	Schedule a use timeout.
q	qq_sendWBDataFromTBEToL2	Send data from TBE to L2
q	q_sendDataFromTBEToCache	Send data from TBE to cache
qq	q_sendExclusiveDataFromTBEToCache	Send data from TBE to cache
s	s_deallocateTBE	Deallocate TBE
ub	ub_dmaUnblockL2Cache	Send dma ack to l2 cache
udh	uu_profileDataHit	Profile the demand hit
udm	uu_profileDataMiss	Profile the demand miss
uih	uu_profileInstHit	Profile the demand hit
uim	uu_profileInstMiss	Profile the demand miss
u	u_writeDataToCache	Write data to cache
v	v_writeDataToCacheVerify	Write data to cache, assert it was same as before
xx	xx_store_hit	Notify sequencer that store completed.
z	zz_recycleMandatoryQueue	Send the head of the mandatory queue to the back of the queue.
z	z_recycleRequestQueue	Send the head of the mandatory queue to the back of the queue.

Table 4.10: moesi_cmp_directory - L1Cache - action

Event	Description
Ack	Received an ack message
All acks	Received all required data and message acks
Data	Received a data message, responder has a shared copy
Exclusive Data	Received a data message
Fwd DMA	A GetS from another processor
Fwd GETS	A GetS from another processor
Fwd GETX	A GetX from another processor
Ifetch	I
Inv	Invalidations from the directory
L1 Replacement	Replacement
Load	Load request from the processor
Own GETX	We observe our own GetX forwarded back to us
Store	Store request from the processor
Use Timeout	lockout period ended
Writeback Ack	Writeback O.K. from directory
Writeback Ack Data	Writeback O.K. from directory
Writeback Nack	Writeback not O.K. from directory

Table 4.11: moesi_cmp_directory - L1Cache - Event

	Load	Ifetch	Store	L1 Replacement	Own GETX	Fwd GETX	Fwd GETS	Fwd DMA	Inv	Ack	Data	Exclusive Data	Writeback Ack	Writeback Ack Data	Writeback Nack	All acks	Use Timeout
I	i i a udm k / IS	j i a uim k / IS	i i b udm k / IM	k					f l								
S	hd udh k	hi uih k	i b udm k / SM	i ds cc k / OI			e l	e ub l	f cc l / I								
O	hd udh k	hi uih k	i b udm k / SM	i d cc k / OI		e cc l / I	e l	e ub l									
M	hd udh k	hi uih k	h udh k / M M	i d cc k / MI		e cc l / I	e l / O	e ub l									
M W	hd udh k	hi uih k	h udh k / M M W	z	z	z	z	z	z								jj / M
MM	hd udh k	hi uih k	h udh k	i d cc k / MI		e cc l / I	e cc l / I	e ub l									
MM W	hd udh k	hi uih k	h udh k	z	z	z	z	z	z								jj / MM
IM	z	z	z	z					f l	m o n	u m o n / S M	u m o n / S M					
SM	hd udh k	hi uih k	z	z			e l	e ub l	f cc l / IM	m o n	m o n / SM	m o n / SM					
SM	hd udh k	hi uih k	z	z	m o l	e l / IM	e l	e ub l		m o n						xx g s oo j / MM W	
IS	z	z	z	z					f l		u m hx g s n / S	u m hx g oo s n / M W					
OI	z	z	z	z			q l	q ub l	f l / II				g s l / I	q s l / I	ds l		
OI	z	z	z	z		qq l / II	q l	q ub l					g s l / I	q s l / I	d l		
MI	z	z	z	z		qq l / II	q l / OI	q ub l					g s l / I	q s l / I	d l / OI		
II	z	z	z	z					f l				g s l / I	g s l / I	s l / I		

Table 4.12: moesi_cmp_directory - L1Cache - table

State	Description
I	Invalid
IFGS	Blocked, forwarded global GETS to local owner
IFGX	Blocked, forwarded global GETX to local owner/exclusive. No other on
IFGXX	Blocked, forwarded global GETX to local owner but may need acks from other sharers
IFLO	Blocked, forwarded local GETS to local owner
IFLOSX	Blocked, forwarded local GETS to local owner w/ other sharers, chip is exclusive
IFLOX	Blocked, forwarded local GETS to local owner but chip is exclusive
IFLOXX	Blocked, forwarded local GETX to local owner/exclusive, chip is exclusive
IFLS	Blocked, forwarded local GETS to _some_ local sharer
IFLXO	Blocked, forwarded local GETX to local owner with other sharers, chip is exclusive
IGM	Blocked, issued local GETX to directory. Need global acks and data
IGMIO	Blocked, issued local GETX, local owner with possible local sharer, may need to INV
IGMIOF	Blocked, issued local GETX, local owner, waiting for global acks, got Fwd_GETX
IGMIOFS	Blocked, issued local GETX, local owner, waiting for global acks, got Fwd_GETS
IGMLS	Blocked, issued local GETX to directory but may need to INV local sharers
IGMO	Blocked, have data for local GETX but need all acks
IGS	Semi
II	Blocked, handling invalidations
ILO	Idle/NP, but local owner exists
ILOD	Blocked, waiting for DMA ack
ILOS	Idle/NP, but local owner exists and local sharers as well
ILOSD	Blocked, waiting for DMA ack
ILOSW	local WB request, was ILOS
ILOSX	Idle/NP, but local owner exists, local sharers exist, chip is exclusive
ILOSXD	Blocked, waiting for DMA ack
ILOSXW	local WB request, was ILOSX
ILOW	local WB request, was ILO
ILOX	Idle/NP, but local owner exists and chip is exclusive
ILOXD	Blocked, waiting for DMA ack
ILOXW	local WB request, was ILOX
ILS	Idle/NP, but local sharers exist
ILSI	Blocked, doing writeback, was OLS got Fwd_GETX
ILSW	local WB request, was ILS
ILX	Idle/NP, but local exclusive exists
ILXD	Blocked, waiting for DMA ack
ILXW	local WB request, was ILX
ISFGS	Blocked, forwarded global GETS to local owner, local sharers exist
IW	local WB request from only sharer, was ILS
M	Modified
MI	Blocked, doing writeback, was M
MII	Blocked, doing writeback, was M, got Fwd_GETX
MM	Blocked, was M satisfying local GETX
NP	Not Present
O	Owned, no local sharers
OFGX	Blocked, forwarded global GETX to owner and got data but may need acks
OGMIO	Blocked, issued local GETX, was owner, may need to INV
OGMIOF	Blocked, issued local GETX, was owner, waiting for global acks, got Fwd_GETX
OI	Blocked, doing writeback, was O
OLS	Owned with local sharers
OLSF	Blocked, got Fwd_GETX with local sharers, waiting for local inv acks
OLSI	Blocked, doing writeback, was OLS
OLSS	Blocked, satisfying local GETS
OLSW	local WB request, was OLS
OLSX	Owned with local sharers, chip is exclusive
OLSXS	Blocked, satisfying local GETS
OLSXW	local WB request from sharer, was OLSX

action	Function name	Description
a	a_issueGETS	issue local request globally
a	a_issueGETX	issue local request globally
b	b_issuePUTO	Issue PUTO
bb	b_issuePUTO_ls	Issue PUTO
b	b_issuePUTX	Issue PUTX
cd	cd_sendDataFromTBEToFwdDma	Send data from TBE to external GETX
ccc	c_sendDataFromTBEToFwdGETS	Send data from TBE to external GETX
cc	c_sendDataFromTBEToFwdGETX	Send data from TBE to external GETX
c	c_sendDataFromTBEToL1GETS	Send data from TBE to L1 requestors in TBE
c	c_sendDataFromTBEToL1GETX	Send data from TBE to L1 requestors in TBE
ccc	c_sendExclusiveDataFromTBEToFwdGETS	Send data from TBE to external GETX
cc	c_sendExclusiveDataFromTBEToL1GETS	Send data from TBE to L1 requestors in TBE
da	da_sendDmaAckUnblock	Send dma ack to global directory
dd	dd_sendDataToFwdGETS	send data
dd	dd_sendDataToFwdGETX	send data
dd	dd_sendExclusiveDataToFwdGETS	send data
d	d_sendDataToL1GETS	Send data directly to L1 requestor
d	d_sendDataToL1GETX	Send data and a token from TBE to L1 requestor
eee	ee_addLocalIntAck	add a local ack to wait for
eeee	ee_issueLocalInvExceptL1Requestor	Send local invalidates to sharers if they exist
eeeeee	ee_issueLocalInvExceptL1RequestorInTBE	Send local invalidates to sharers if they exist
ee	ee_sendLocalInv	Send local invalidates
eee	ee_sendLocalInvSharersOnly	Send local invalidates to sharers if they exist
e	e_sendAck	Send ack with the tokens we've collected thus far.
e	e_sendAckToL1Requestor	Send ack with the tokens we've collected thus far.
eee	e_sendAckToL1RequestorFromTBE	Send ack with the tokens we've collected thus far.
f	f_sendExclusiveUnblock	Send unblock to global directory
f	f_sendUnblock	Send unblock to global directory
gg	gg_clearLocalSharers	Clear local sharers
gg	gg_clearOwnerFromL1Response	Clear sharer from L1 response queue
gg	gg_clearSharerFromL1Response	Clear sharer from L1 response queue
g	g_recordLocalExclusive	Record new local exclusive sharer from unblock message
g	g_recordLocalSharer	Record new local sharer from unblock message
hh	hh_countLocalSharersExceptL1GETXRequestorInTBE	counts number of acks needed for L1 GETX
h	h_clearIntAcks	clear IntAcks
h	h_countLocalSharersExceptRequestor	counts number of acks needed for L1 GETX
i	i_allocateTBE	Allocate TBE for internal/external request(isPrefetch=0, number of invalidates=0)
i	i_copyDataToTBE	Copy data from response queue to TBE
jd	jd_forwardDmaRequestToLocalOwner	Forward dma request to local owner
j	j_forwardGlobalRequestToLocalOwner	Forward external request to local owner
kk	kk_forwardLocalGETSToLocalOwner	Forward local request to local owner
kk	kk_forwardLocalGETXToLocalExclusive	Forward local request to local owner
k	k_forwardLocalGETSToLocalSharer	Forward local request to local sharer/owner
k	k_forwardLocalGETXToLocalOwner	Forward local request to local owner
ll	ll_writebackNack	Send writeback nack to L1
l	l_writebackAckDropData	Send writeback ack to L1 indicating to drop data
l	l_writebackAckNeedData	Send writeback ack to L1 requesting data
mm	mm_decrementNumberOfMessagesExt	Decrement the number of messages for which we're waiting
mmm	m_decrementNumberOfMessagesExt	Decrement the number of messages for which we're waiting
m	m_decrementNumberOfMessagesInt	Decrement the number of messages for which we're waiting
m	m_popRequestQueue	Pop request queue.
n	n_popResponseQueue	Pop response queue
n	n_popTriggerQueue	Pop trigger queue.
oo	o_checkForExtCompletion	Check if we have received all the messages required for completion
o	o_checkForIntCompletion	Check if we have received all the messages required for completion
o	o_popL1RequestQueue	Pop L1 request queue.
qq	qq_sendDataFromTBEToMemory	Send data from TBE to directory
r	rr_deallocateL2CacheBlock	Deallocate L2 cache block. Sets the cache to not present, allowing a replacement in parallel with a fetch.
rrr	r_setMRU	manually set the MRU bit for cache line
s	s_deallocateTBE	Deallocate external TBE
ss	s_recordGetSL1ID	record local GETS requestor
ss	s_recordGetXL1ID	record local GETX requestor
t	t_recordFwdSID	record global GETS requestor
t	t_recordFwdXID	record global GETX requestor
uh	uu_profileHit	Profile the demand hit
um	uu_profileMiss	Profile the demand miss
u	u_writeDataToCache	Write data to cache
v	vv_allocateL2CacheBlock	Set L2 cache tag equal to tag of block B.
w	w_assertIncomingDataAndCacheDataMatch	Assert that the incoming data and the data in the cache match
y	y_copyCacheStateToDir	Copy cache state to directory state
/y	y_copyDirToCacheAndRemove	Copy dir state to cache and remove
zz	zz_recycleL1RequestQueue	Send the head of the mandatory queue to the back of the queue.
zz	zz_recycleRequestQueue	Send the head of the mandatory queue to the back of the queue.
zz	zz_recycleResponseQueue	Send the head of the mandatory queue to the back of the queue.

Table 4.14: moesi_cmp_directory - L2Cache - action

Event	Description
All Acks	Received all ack messages
Data	Received a data message, responder has a shared copy
Data Exclusive	Received a data message
DmaAck	DMA ack from local L1
Exclusive Unblock	Local L1 is telling L2 dir to unblock
ExtAck	Received an ack message
Fwd DMA	A request from DMA
Fwd GETS	A GetS from another processor
Fwd GETX	A GetX from another processor
IntAck	Received an ack message
Inv	Invalidations from the directory
L1 GETS	local L1 GETS request
L1 GETX	local L1 GETX request
L1 PUTO	local owner wants to writeback
L1 PUTS	local sharer wants to writeback
L1 PUTS only	only local sharer wants to writeback
L1 PUTX	local exclusive wants to writeback
L1 WBCLEANDATA	Writeback from L1, with data
L1 WBDIRTYDATA	Writeback from L1, with data
L2 Replacement	L2 Replacement
Own GETX	A GetX from this node
Unblock	Local L1 is telling L2 dir to unblock
Writeback Ack	Writeback O.K. from directory
Writeback Nack	Writeback not O.K. from directory

Table 4.15: moesi_cmp_directory - L2Cache - Event

	L1 GETS	L1 GETX	L1 PUTO	L1 PUTX	L1 PUTS only	L1 PUTS	Fwd GETX	Fwd GETS	Fwd DMA	Own GETX	Inv	IntAck	ExtAck	All Acks	Data	Data Exclusive	L1 WBCLEANDATA	L1 WBDIRTYDATA	Writeback Ack	Writeback Nack	Unblock	Exclusive Unblock	DmaAck	L2 Replacement
NP	i ss a u m o / IG S	i ss a u m o / IG M	ll o	ll o		ll o					i t e s m													
I	i ss a u m o / IG S	i ss a u m o / IG M	ll o	ll o		ll o					i t e s m													r
ILS	k um o / IFLS	i ss a h um o / IGMLS	ll o	ll o	l o / IW	l o / IL SW					i t ee g g m / II													y r
ILX	kk um o / IFLOXX	kk e um o / IFLO XX	ll o	l o / IL XW	ll o	ll o	i t j m / IFGX	i t j m / IFGS	i t jd m / ILXD		i t ee g g m / II				i ccc s n									y r
ILO	kk um o / IFLO	i ss a h um o / IGMIO	l o / IL OW	l o / IL OW		ll o	i t j m / IFGX	i t j m / IFGS	i t jd m / ILOD		i t ee g g m / II				i ccc s n									y r
ILOX	kk um o / IFLOX	kk e um o / IFLO XX	l o / IL OXW	ll o		ll o	i t j m / IFGX	i t j m / IFGS	i t jd m / ILOXD						i ccc s n									
ILOS	kk um o / IFLO	i ss a u m o / IG MIO	l o / IL OSW	l o / IL OSW	l o / IL OW	l o / IL OSW	i t j ee e eee m / IFGXX	i t j m / ISFGS	i jd m / ILOSD						i ccc s n									y r
ILOSX	kk um o / IFLOSX	i ss h e eee k ee e um o / IFLXO	l o / IL OSXW	l o / IL OSXW	l o / IL OXW	l o / IL OSXW	i t j ee e eee m / IFGXX	i t j m / ISFGS	i t jd m / ILOSX D						i ccc s n									
S	d rrr uh o / SS	i ss a y rrr r u m o / IG M		ll o		ll o					i t e s r m / I													r / I
O	d rrr uh o / OO	i ss a y r um o / IGMO		ll o			dd y r m / I	dd m	dd da m															b i r / OI
OLS	d rrr uh o / OLS S	i ss a h y r um o / OGMI O		ll o	l o / OW	l o / OL SW	i t ee m / OLSF	dd m	dd da m															y bb i r / OLSI
OLSX	d rrr uh o / OLS XS	i ss h e eee d y rrr r uh o / IFL OX	ll o	ll o	l o / OX W	l o / OL SXW	i t ee m / OLSF	dd m / O LS	dd da m															y bb i r / OLSI
SLS	d rrr uh o / SLS S	i ss a h y r um o / IGML S		ll o	l o / SW	l o / SL SW					i t ee r m / II													y r / IL S
M	i h d rr r s uh o / OO	i h d y r s uh o / MM	ll o	ll o		ll o	dd r m / I	dd r m / I	dd da m															b i r / MI
IFGX	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz				i cc gg s n / I	i cc gg s n / I								zz
IFGS	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz				i ccc s n / ILO	i ccc gg s n / I								zz
ISFGS	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz				i ccc s n / ILOS									zz
IFGXX	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz	m o n		cc gg s n / I		i m o n								zz
OFGX	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz													zz
OLSF	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz	m o n		cc gg s r n / I										zz

Table 4.16: moesi_cmp_directory - L2Cache - table- part 1/4

	L1 GETS	L1 GETX	L1 PUTO	L1 PUTX	L1 PUTS only	L1 PUTS	Fwd GETX	Fwd GETS	Fwd DMA	Own GETX	Inv	IntAck	ExtAck	All Acks	Data	Data Exclusive	L1 WBCLEANDATA	L1 WBDIRTYDATA	Writeback Ack	Writeback Nack	Unblock	Exclusive Unblock	DmaAck	L2 Replacement
ILOW	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz						v /y gg u n / O	v /y gg u n / O			gg n / ILO			zz
ILOXW	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz						v /y gg u n / M	v /y gg u n / M			gg n / ILOX			zz
ILOSW	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz						v /y gg u n / OLS	v /y gg u n / OLS			gg n / ILOS			zz
ILOSXW	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz						v /y gg u n / OLSX	v /y gg u n / OLSX			gg n / ILOSX			zz
SLSW	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz										gg n / SLS			zz
OLSW	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz										gg n / OLS			zz
ILSW	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz						v /y u g g n / SLS				gg n / ILS			zz
IW	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz						v /y u g g n / S							zz
OW	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz										gg n / O			zz
SW	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz										gg n / S			zz
OXW	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz										gg n / M			zz
OLSXW	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz										gg n / OLSX			zz
ILXW	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz				i ccc sn		gg v /y u n / M	gg v /y u n / M			n / ILX			zz
IFLS	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz										g n / ILS			zz
IFLO	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz										g n / ILOS			zz
IFLOX	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz										g n / ILOSX	g s n / ILX		zz
IFLOXX	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz										g n / ILOSX	g n / ILX		zz
IFLOSX	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz										g n / ILOSX	g n / ILX		zz
IFLXO	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz											g s n / ILX		zz
IGS	zz	zz	zz	zz	zz	zz	zz	zz	zz	zz	t e m				i mmm cn	i mmm ccn					g f s n / ILS	g f s n / ILX		zz

Table 4.17: moesi_cmp_directory - L2Cache - table- part 2/4

	L1 GETS	L1 GETX	L1 PUTO	L1 PUTX	L1 PUTS only	L1 PUTS	Fwd GETX	Fwd GETS	Fwd DMA	Own GETX	Inv	IntAck	ExtAck	All Acks	Data	Data Exclusive	L1 WBCLEANDATA	L1 WBDIRTYDATA	Writeback Ack	Writeback Nack	Unblock	Exclusive Unblock	DmaAck	L2 Replacement
IGM	zz	zz	zz	zz	zz	zz	zz	zz	zz	zz	t e m		mmm oo n		i mmm oo n / IGMO	i mmm oo n / IGMO								zz
IGMLS	zz	zz	zz	zz	zz	zz					t ee m	m o n	mmm oo n	gg h e n / IGM	eeeeee i mmm oo n / IGMO	eeeeee i mmm oo n / IGMO								zz
IGMO	zz	zz	zz	zz	zz	zz	t cc m / IGM	t ccc m	zz	mm oo m			mmm oo n	c n								g f s n / ILX		zz
IGMIO	zz	zz	zz	zz	zz	zz	t j eee eee m / IGMIOF	t j m / IGMIOFS	zz	mm oo m			mmm oo n	hh eeeee e k eee n / IGMO										
OGMIO	zz	zz	zz	zz	zz	zz	t eee m / OGMIOF	t ccc m	zz	mm oo m			mmm oo n	eeeeee c n / IGMO										
IGMIOF	zz	zz	zz	zz	zz	zz						m o n		gg cc n / IGM		i m o n								
IGMIOFS	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz				i ccc n / IGMIO									zz
OGMIOF	zz	zz	zz	zz	zz	zz						m o n		gg hh cc n / IGM										
II	zz	zz	zz	zz	zz	zz						m o n		e s n / I										
MM	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz											g n / IL X		zz
SS	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz										g n / SL S			zz
OO	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz										g n / OL S	g y r n / ILX		zz
OLSS	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz										g n / OL S			zz
OLSXS	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz										g n / OL SX			zz
SLSS	zz	zz	zz	zz	zz	zz	zz	zz	zz		zz										g n / SL S			zz
OI	zz	zz	zz	zz	zz	zz	t cc m / MII	t ccc m	cd da m										qq s m / I	b m				zz
MI	zz	zz	zz	zz	zz	zz	t cc m / MII	t ccc m / OI	cd da m / OI										qq s m / I					zz
MII	zz	zz	zz	zz	zz	zz													f s m / I	s m / I				zz
OLSI	zz	zz	zz	zz	zz	zz	t ee m / ILSI	t ccc m	cd da m										qq s m / ILS					zz
ILSI	zz	zz	zz	zz	zz	zz						m o n		gg cc n / MII					f s m / ILS					zz

Table 4.18: moesi_cmp_directory - L2Cache - table- part 3/4

	L1 GETS	L1 GETX	L1 PUTO	L1 PUTX	L1 PUTS only	L1 PUTS	Fwd GETX	Fwd GETS	Fwd DMA	Own GETX	Inv	IntAck	ExtAck	All Acks	Data	Data Exclusive	L1 WBCLEANDATA	L1 WBDIRTYDATA	Writeback Ack	Writeback Nack	Unblock	Exclusive Unblock	DmaAck	L2 Replacement
ILOSD	zz	zz	zz	zz	zz	zz	zz	zz	zz	zz	zz												s da n / ILOS	zz
ILOSXD	zz	zz	zz	zz	zz	zz	zz	zz	zz	zz	zz												s da n / ILOSX	zz
ILOD	zz	zz	zz	zz	zz	zz	zz	zz	zz	zz	zz												s da n / ILO	zz
ILXD	zz	zz	zz	zz	zz	zz	zz	zz	zz	zz	zz												s da n / ILX	zz
ILOXD	zz	zz	zz	zz	zz	zz	zz	zz	zz	zz	zz												s da n / ILOX	zz

Table 4.19: moesi_cmp_directory - L2Cache - table- part 4/4

Chapter 5

moesi_hammer

State	Description
E	Exclusive Owner, no probe filter entry
NO	Not Owner, probe filter entry exists, block in E/M at Owner
NOB	Not Owner, Blocked
NOB	Not Owner, Blocked, next queued request GETS
NOB	Not Owner, Blocked, forwarded merged GETS, waiting for responses
NO B W	Not Owner, Blocked, waiting for Dram
NOB	Not Owner, Blocked, next queued request GETX
NO DR B	Not Owner, Dma Read waiting for cache responses
NO DR B D	Not Owner, Dma Read waiting for cache responses including dirty data
NO DR B W	Not Owner, Dma Read waiting for Dram and cache responses
NO DW B W	Not Owner, Dma Write waiting for Dram and cache responses
NO DW W	Not Owner, Dma Write waiting for Dram
NO F	Blocked on a flush
NO F W	Not Owner, Blocked, waiting for Dram
NO R	Was Not Owner or Sharer, replacing probe filter entry
NO W	Not Owner, waiting for Dram
NX	Not Owner, probe filter entry exists, block in O at Owner
O	Data clean, probe filter entry exists
OB	Owner, Blocked
O B W	Owner, Blocked, waiting for Dram
O DR B	Owner, Dma Read waiting for cache responses
O DR B W	Owner, Dma Read waiting for Dram and cache responses
O R	Was data Owner, replacing probe filter entry
O W	Owner, waiting for Dram
S	Data clean, probe filter entry exists pointing to the current owner
S R	Was Not Owner or Sharer, replacing probe filter entry
WB	Blocked on a writeback
WB E W	Blocked on memory write, will go to E
WB O W	Blocked on memory write, will go to O

Table 5.1: moesi_hammer - Directory - State

action	Function name	Description
ano	ano_assertNotOwner	Assert that request is not current owner
ans	ans_assertNotSharer	Assert that request is not a current sharer
auno	auno_assertUnblockerNotOwner	assert unblocker not owner
ac	a_assertCacheData	Assert that a cache provided the data
a	a_sendWriteBackAck	Send writeback ack to requestor
b	b_sendWriteBackNack	Send writeback nack to requestor
cs	cs_clearSharers	clear current sharers
da	da_sendDmaAck	Send Ack to DMA controller
dr	dr_sendDmaData	Send Data to DMA controller from memory
dt	dt_sendDmaDataFromTbe	Send Data to DMA controller from tbe
d	d_sendData	Send data to requestor
fb	fb_forwardRequestBcast	Forward requests to all nodes
fc	fc_forwardRequestConditionalOwner	Forward request to one or more nodes
fn	fn_forwardRequestIfNecessary	Forward requests if necessary
frr	fr_forwardMergeReadRequestsToOwner	Forward coalesced read request to owner
fr	f_forwardReadFromDma	Forward requests
fw	f_forwardWriteFromDma	Forward requests
g	g_popTriggerQueue	Pop trigger queue
ia	ia_invalidateAllRequest	invalidate all copies
io	io_invalidateOwnerRequest	invalidate all copies
i	i_popIncomingRequestQueue	Pop incoming request queue
j	j_popIncomingUnblockQueue	Pop incoming unblock queue
k	k_wakeUpDependents	wake
ld	ld_queueMemoryDmaWrite	Write DMA data to memory
l	ll_checkIncomingWriteback	Check PUTX/PUTO response message
ly	ly_queueMemoryWriteFromTBE	Write data to memory from TBE
q	l_popMemQueue	Pop off
lq	l_queueMemoryWBRequest	Write PUTX data to memory
mu	mu_decrementNumberOfUnblocks	Decrement the number of messages for which we're waiting
m	m_decrementNumberOfMessages	Decrement the number of messages for which we're waiting
nofc	nofc_forwardRequestConditionalOwner	Forward request to one or more nodes if the requestor is not the owner
n	n_popResponseQueue	Pop response queue
oc	oc_sendBlockAck	Send block ack to the owner
os	os_checkForMergedGetSCompletion	Check for merged GETS completion
o	o_checkForCompletion	Check if we have received all the messages required for completion
pa	pa_setPendingMsgsToAll	set pending msgs to all
pfa	pfa_probeFilterAllocate	Allocate ProbeFilterEntry
pfd	pfd_probeFilterDeallocate	Deallocate ProbeFilterEntry
po	po_setPendingMsgsToOne	set pending msgs to one
ppfd	ppfd_possibleProbeFilterDeallocate	Deallocate ProbeFilterEntry
pd	p_popDmaRequestQueue	pop dma request queue
qd	qd_queueMemoryRequestFromDmaRead	Queue off
qf	qf_queueMemoryFetchRequest	Queue off
rs	rs_recordGetSRequestor	Record GETS requestor in TBE
s	rs_removeSharer	remove current sharer
rx	rx_recordExclusiveInTBE	Record Exclusive in TBE
rc	r_recordCacheData	record data from cache response to TBE
rt	r_recordDataInTBE	Record Data in TBE
rd	r_recordMemoryData	record data from memory to TBE
rr	r_setMRU	manually set the MRU bit for pf entry
r	r_setSharerBit	We saw other sharers
saa	saa_setAcksToAllIfPF	Non
sa	sa_setAcksToOne	Forwarded request, set the ack amount to one
sc	sc_signalCompletionIfPF	indicate that we should skip waiting for cpu acks
so	so_setOwnerBit	We saw other sharers
spa	spa_setPendingAcksToZeroIfPF	if probe filter, no need to wait for acks
sp	sp_setPendingMsgsToMergedSharers	Set pending messages to waiting sharers
uo	uo_updateOwnerIfPf	update owner
us	us_updateSharerIfFBD	update sharer if full
vd	vd_allocateDmaRequestInTBE	Record Data in TBE
v	v_allocateTBE	Allocate TBE
w	w_deallocateTBE	Deallocate TBE
zd	zd_stallAndWaitDMARequest	Stall and wait the dma request queue
z	z_stallAndWaitRequest	Recycle the request queue

Table 5.2: moesi_hammer - Directory - action

Event	Description
Ack	Received an ack message
All acks and data no sharers	Received all acks and no other processor has a shared copy
All acks and owner data	Received shared data and message acks
All acks and shared data	Received shared data and message acks
All Unblocks	Received all unblocks for a merged gets request
Data	Received a data message, responder had a owner or exclusive copy, they gave it to us
DMA READ	A DMA Read memory request
DMA WRITE	A DMA Write memory request
Exclusive Data	Received a data message, responder had an exclusive copy, they gave it to us
GETF	A GETF arrives
GETS	A GETS arrives
GETX	A GETX arrives
Memory Ack	Writeback Ack from memory arrives
Memory Data	Fetched data from memory arrives
Pf Replacement	probe filter replacement
PUT	A PUT arrives
PUTF	A PUTF arrives
Shared Ack	Received an ack message, responder has a shared copy
Shared Data	Received a data message, responder has a shared copy
Unblock	An unblock message arrives
UnblockM	An unblock message arrives
UnblockS	An unblock message arrives
Writeback Clean	The final part of a PutX (no data)
Writeback Dirty	The final part of a PutX (data)
Writeback Exclusive Clean	The final part of a PutX (no data, exclusive)
Writeback Exclusive Dirty	The final part of a PutX (data, exclusive)

Table 5.3: moesi_hammer - Directory - Event

	GETX	GETS	PUT	Unblock	Unblock S	Unblock M	Writeback Clean	Writeback Dirty	Writeback Exclusive Clean	Writeback Exclusive Dirty	Pf Replacement	DMA READ	DMA WRITE	Memory Data	Memory Ack	Ack	Shared Ack	Shared Data	Data	Exclusive Data	All acks and shared data	All acks and owner data	All acks and data no sharers	All Unblocks	GETF	PUTF
NX	rr fb cs i / NOB	rr anofc i / NOB	a i / WB								v pa iapfd / NO R	vd fr pd / NODR B D	vd fw pd / NODW B W												rr fb cs i / NO F	
NO	rr anofc cs ii / NOB	rr anoans fci / NOB	a i / WB								v po iopfd / NO R	vd fr pd / NODR B D	vd fw pd / NODW B W												rr nofccs oci / NO F	
S	rr fb cs i / NOB	rr anofb i / NOB	a i / WB								v pa iapfd / S R	vd fr pd / NODR B D	vd fw pd / NODW B W												rr fb cs i / NO F	
O	rr v rtsa qffb cs i / NO B W	rr v rtsaa qffn i / O B W	b i								v pa iapfd / O R	vd spaqd fr pd / O DR B W	vd fw pd / NODW B W												rr v rtsa qffb cs i / NO F W	
E	pfa v rx saa qf fn i / NO B W	pfa v rx saa qf fn i / NO B W	b i									vd qd spa fr pd / NODR B W	vd spasc fw pd / NODW B W												pfa v rx saa qf fn i / NO F W	
O R	z	z	z								z	zd	zd			m o n							w k g / E		z	
S R	z	z	z								z	zd	zd			m o n			m o n				w k g / E		z	
NO R	z	z	z								z	zd	zd			m o n			rc m on	rc m on			ly w kg / WB E W		z	
NOB	z / NOB	v rs i / NOB	z		us k j / NX	uo us kj / NO					z	zd	zd												z / NOB	
NOB	z	z	z		us k j / NX	uo us kj / NO					z	zd	zd												z	
NOB	z	rs i	z		us frsp j / NOB	uo frsp j / NOB					z	zd	zd												z	
NOB	z	z	z		us mu os j						z	zd	zd											w k g / NX	z	
OB	z	z	z		us k j / O	us uo kj / NO					z	zd	zd												z	
NO B W	z	z	z		us j / NO W	uo j / NO W					z	zd	zd	d w q / NOB											z	
O B W	z	z	z		us j / O W						z	zd	zd	d w q / OB											z	
NO W	z	z	z								z	zd	zd	w k q / NO											z	
O W	z	z	z								z	zd	zd	w k q / O											z	
NO DW B W	z	z	z								z	zd	zd			m o n			rc m on	rc m on			ld g / NO DW W		z	
NO DR B W	z	z	z								z	zd	zd	rd o q / NO DR B		m n	m r n	rc m soo n	rc m n	rc m n					z	
NO DR B D	z	z	z								z	zd	zd			m o n	m r o n	rc m soo n	rc m on	rc m on	dt w kg / S	dt ly wk g / WB O W	ac dt ly w ppfd k g / WB E W		z	

Table 5.4: moesi_hammer - Directory - table- part 1/2

	GETX	GETS	PUT	Unblock	Unblock S	Unblock M	Writeback Clean	Writeback Dirty	Writeback Exclusive Clean	Writeback Exclusive Dirty	Pf Replacement	DMA READ	DMA WRITE	Memory Data	Memory Ack	Ack	Shared Ack	Shared Data	Data	Exclusive Data	All acks and shared data	All acks and owner data	All acks and data no sharers	All Unblocks	GETF	PUTF
NO DR B	z	z	z								z	zd	zd			m o n	m r o n	rc m so o n	rc m o n	rc m o n	dt w k g / S	dt ly w k g / WB O W	dt ly w ppfd k g / WB E W		z	
NO DW W	z	z	z								z	zd	zd		da w ppfd k q / E										z	
O DR B W	z	z	z								z	zd	zd	rd dr o q / O DR B		m n	m r n								z	
O DR B	z	z	z								z	zd	zd			m o n	m r o n					ly w k g / WB O W	ly w pf d k g / WB E W		z	
WB	z	z	z	auno k j / NX			l s k j / O	s lq j / WB O W	l s pfd k j / E	s lq pf d j / W B E W	z	zd	zd												z	
WB O W	z	z	z								z	zd	zd		k q / O										z	
WB E W	z	z	z								z	zd	zd		k q / E										z	
NO F	z	z	z			us uo j					z														z	a i / W B
NO F W	z	z	z								z	zd	zd	d w q / NO F											z	

Table 5.5: moesi_hammer - Directory - table- part 2/2

State	Description
BUSY RD	Busy
BUSY WR	Busy
READY	Ready to accept a new request

Table 5.6: moesi_hammer - DMA - State

action	Function name	Description
a	a_ackCallback	Notify dma controller that write request completed
d	d_dataCallback	Write data to dma sequencer
p	p_popRequestQueue	Pop request queue
p	p_popResponseQueue	Pop request queue
s	s_sendReadRequest	Send a DMA read request to memory
s	s_sendWriteRequest	Send a DMA write request to memory

Table 5.7: moesi_hammer - DMA - action

Event	Description
Ack	DMA write to memory completed
Data	Data from a DMA memory read
ReadRequest	A new read request
WriteRequest	A new write request

Table 5.8: moesi_hammer - DMA - Event

	ReadRequest	WriteRequest	Data	Ack
READY	s p / BUSY RD	s p / BUSY WR		
BUSY RD			d p / READY	
BUSY WR				a p / READY

Table 5.9: moesi_hammer - DMA - table

State	Description
I	Idle
II	Issued PutX/O, saw Other_GETS or Other_GETX, waiting for ack
IM	Issued GetX
IM F	Issued GetX due to a Flush
IR	Idle
IS	Issued GetS
ISM	Issued GetX, received valid data, waiting for all acks
ISM F	Issued GetX, received data, waiting for all acks
M	Modified (dirty)
MI	Issued PutX, waiting for ack
MI F	Issued PutX due to a Flush, waiting for ack
MM	Modified (dirty and locally modified)
MMR	Modified (dirty and locally modified)
MMT	MM block transferring to L0
MM F	Issued GETF due to a Flush, waiting for ack
MMW	Issued GetX, received exclusive data
MM WF	Issued GetX, received exclusive data
MR	Modified (dirty)
MT	M block transferring to L1
MW	Issued GetS, received exclusive data
O	Owned
OI	Issued PutO, waiting for ack
OM	Issued GetX, received data
OM F	Issued GetX, received data
OR	Owned
OT	O block transferring to L1
S	Shared
SM	Issued GetX, we still have a valid copy of the line
SM F	Issued GetX, we still have an old copy of the line
SR	Shared
SS	Issued GetS, received data, waiting for all acks
ST	S block transferring to L1

Table 5.10: moesi_hammer - L1Cache - State

action	Function name	Description
a	a_issueGETS	Issue GETS
bf	bf_issueGETF	Issue GETF
b	b_issueGETX	Issue GETX
bo	b_issueGETXIfMoreThanOne	Issue GETX
ct	ct_sendExclusiveDataFromTBE	Send exclusive data from tbe to requestor
c	c_sendExclusiveData	Send exclusive data from cache to requestor
df	df_issuePUTF	Issue PUTF
d	d_issuePUT	Issue PUT
e	ee_sendDataShared	Send data from cache to requestor, remaining the owner
emt	emt_sendDataSharedMultipleFromTBE	Send data from tbe to all requestors
em	em_sendDataSharedMultiple	Send data from cache to all requestors, still the owner
et	et_sendDataSharedFromTBE	Send data from TBE to requestor, keep a shared copy
e	e_sendData	Send data from cache to requestor
f	ff_sendAckShared	Send shared ack from cache to requestor
cc	forward_eviction_to_cpu	sends eviction information to the processor
f	f_sendAck	Send ack from cache to requestor
g	gg_deallocateL1CacheBlock	Deallocate cache block. Sets the cache to invalid, allowing a replacement in parallel with a fetch.
gm	gm_sendUnblockM	Send unblock to memory and indicate M/O/E state
gr	gr_deallocateCacheBlock	Deallocate an L1 or L2 cache block.
gs	gs_sendUnblockS	Send unblock to memory and indicate S state
g	g_sendUnblock	Send unblock to memory
hf	hh_flush_hit	Notify sequencer that flush completed.
h	hh_store_hit	Notify sequencer that store completed.
li	hp_copyFromTBEToL2	Copy data from TBE to L2 cache entry.
hx	hx_external_load_hit	load required external msgs
hi	h_ifetch_hit	Notify sequencer the ifetch completed.
hd	h_load_hit	Notify sequencer the load completed.
i	ii_allocateL1DCacheBlock	Set L1 D
it	it_allocateTBE	Allocate TBE
i	i_allocateTBE	Allocate TBE
j	jj_allocateL1ICacheBlock	Set L1 I
j	j_popTriggerQueue	Pop trigger queue.
ka	ka_wakeUpAllDependents	wake
kd	kd_wakeUpDependents	wake
k	k_popMandatoryQueue	Pop mandatory queue.
ll	ll_L2toL1Transfer	
l	l_popForwardQueue	Pop forwareded request queue.
m	m_decrementNumberOfMessages	Decrement the number of messages for which we're waiting
fu	nb_copyFromTBEToL1	Copy data from TBE to L1 cache entry.
n	n_popResponseQueue	Pop response queue
o	o_checkForCompletion	Check if we have received all the messages required for completion
p	pp_incrementNumberOfMessagesByOne	Increment the number of messages for which we're waiting by one
p	p_decrementNumberOfMessagesByOne	Decrement the number of messages for which we're waiting by one
qm	qm_sendDataFromTBEToCache	Send data from TBE to cache, multiple sharers, still the owner
q	qq_sendDataFromTBEToMemory	Send data from TBE to memory
q	q_sendDataFromTBEToCache	Send data from TBE to cache
r	rr_deallocateL2CacheBlock	Deallocate L2 cache block. Sets the cache to not present, allowing a replacement in parallel with a fetch.
r	r_setSharerBit	We saw other sharers
sq	sq_sendSharedDataFromTBEToCache	Send shared data from TBE to cache, still the owner
sxt	sxt_trig_ext_store_hit	store required external msgs.
sx	sx_external_store_hit	store required external msgs.
s	s_deallocateTBE	Deallocate TBE
t	t_sendExclusiveDataFromTBEToMemory	Send exclusive data from TBE to memory
uf	uf_writeDataToCacheTBE	Write data to TBE
uo	uo_updateCurrentOwner	When moving SS state, update current owner.
udh	uu_profileL1DataHit	Profile the demand hits
udm	uu_profileL1DataMiss	Profile the demand miss
uih	uu_profileL1InstHit	Profile the demand hits
uim	uu_profileL1InstMiss	Profile the demand miss
uh	uu_profileL2Hit	Profile the demand hits
um	uu_profileL2Miss	Profile the demand miss
u	u_writeDataToCache	Write data to cache
vt	vt_writeDataToTBEVerify	Write data to TBE, assert it was same as before
v	vv_allocateL2CacheBlock	Set L2 cache tag equal to tag of block B.
v	v_writeDataToCacheVerify	Write data to cache, assert it was same as before
z	zz_stallAndWaitMandatoryQueue	Send the head of the mandatory queue to the back of the queue.
z	z_stall	stall

Table 5.11: moesi_hammer - L1Cache - action

Event	Description
Ack	Received an ack message
All acks	Received all required data and message acks
All acks no sharers	Received all acks and no other processor has a shared copy
Block Ack	the directory is blocked and ready for the flush
Complete L2 to L1	L2 to L1 transfer completed
Data	Received a data message
Exclusive Data	Received a data message, responder had an exclusive copy, they gave it to us
Flush line	flush the cache line from all caches
Ifetch	I
Invalidate	Invalidate block
L1 to L2	L1 to L2 transfer
L2 Replacement	L2 Replacement
Load	Load request from the processor
Merged GETS	A Merged GetS from another processor
NC DMA GETS	special GetS when only DMA exists
Other GETS	A GetS from another processor
Other GETS No Mig	A GetS from another processor
Other GETX	A GetX from another processor
Shared Ack	Received an ack message, responder has a shared copy
Shared Data	Received a data message, responder has a shared copy
Store	Store request from the processor
Trigger L2 to L1D	Trigger L2 to L1
Trigger L2 to L1I	Trigger L2 to L1
Writeback Ack	Writeback O.K. from directory
Writeback Nack	Writeback not O.K. from directory

Table 5.12: moesi_hammer - L1Cache - Event

	Load	Ifetch	Store	L2 Replacement	L1 to L2	Trigger L2 to L1D	Trigger L2 to L1I	Complete L2 to L1	Other GETX	Other GETS	Merged GETS	Other GETS No Mig	NC DMA GETS	Invalidate	Ack	Shared Ack	Data	Shared Data	Exclusive Data	Writeback Ack	Writeback Nack	All acks	All acks no sharers	Flush line	Block Ack
I	i i a u dm um k / IS	j i a u im um k / IS	i i b u dm um k / IM						f l	f l		f l	f l	f l										it bf k / IM F	
S	hd udh k	hi uih k	i b udm um k / SM	cc r ka / I	i g v l i s	i r i f u s z l l / ST	i r j f u s z l l / ST		f cc gr l / I	f l		f l	f l	f cc gr l / I										i bf cc g k / SM F	
O	hd udh k	hi uih k	i b p u dm um k / OM	i d cc r ka / OI	i g v l i s	i r i f u s z l l / OT	i r j f u s z l l / OT		e cc gr l / I	e l	em l	e l	e l	e cc gr l / I										i bf p cc g k / OM F	
M	hd udh k	hi uih k	h udh k / MM	i d cc r ka / MI	i g v l i s	i r i f u s z l l / MT	i r j f u s z l l / MT		c cc gr l / I	e l / O	em l / O	e l / O	e l / O	c cc gr l / I										i bf p cc g k / MM F	
MM	hd udh k	hi uih k	h udh k	i d cc r ka / MI	i g v l i s	i r i f u s z l l / MMT	i r j f u s z l l / MMT		c cc gr l / I	c cc gr l / I	em l / O	e l / O	e l / O	c cc gr l / I										i bf p cc g k / MM F	
IR	i i a u dm um k / IS	j i a u im um k / IS	i i b u dm um k / IM		z				z	z	z	z	z	z										it bf k / IM F	
SR	hd udm uh k ka / S	hi uim uh k ka / S	i b udm um k / SM		z				z	z	z	z	z	z										i bf cc g k / SM F	
OR	hd udm uh k ka / O	hi uim uh k ka / O	i b p u dm um k / OM		z				z	z	z	z	z	z										i bf p cc g k / OM F	
MR	hd udm uh k ka / M	hi uim uh k ka / M	h udm u h k ka / MM		z				z	z	z	z	z	z										i bf p cc g k / MM F	
MMR	hd udm uh k ka / MM	hi uim uh k ka / MM	h udm u h k ka / MM		z				z	z	z	z	z	z										i bf p cc g k / MM F	
IM	z	z	z	z	z				f l	f l		f l	f l	f l	m o n		u m o n / ISM		u m o s x n kd / MMW					z	
SM	hd udh k	hi uih k	z	z	z				f cc l / IM	f l		f l	f l	f cc l / IM	m o n		v m o n / ISM		v m o n / ISM					z	
OM	hd udh k	hi uih k	z	z	z				e p cc l / IM	e l	em l	e l	e l	e p cc l / IM	m o n							sxt gm s j kd / MM	sxt gm s j kd / MM	z	
ISM	hd udh k	hi uih k	z	z	z										m o n								sxt gm s j kd / MM	z	
MW	hd udh k	hi uih k	h udh k / MMW	z	z										m o n								gm s j kd / M	z	
MMW	hd udh k	hi uih k	h udh k	z	z										m o n								gm s j kd / MM	z	
IS	z	z	z	z	z				f l	f l		f l	f l	f l	m o n	m r o n	u m o h x uo n kd / SS	u r m o hx uo n kd / SS	u m o h x n kd / MW					z	
SS	hd udh k	hi uih k	z	z	z										m o n	m r o n						gs s j kd / S	gs s j kd / S	z	
OI	z	z	z	z	z				q l / I I	sq l	qm l	sq l	sq l	q l / I I						q s l k d / I				z	
MI	z	z	z	z	z				q l / I I	sq l / OI	qm l / OI	sq l / OI	sq l / OI	q l / I I						t s l k d / I				z	

Table 5.13: moesi_hammer - L1Cache - table- part 1/2

	Load	Ifetch	Store	L2 Repl acement	L1 to L 2	Trigger L2 to L1D	Trigger L2 to L1I	Complet e L2 to L1	Other G ETX	Other G ETS	Merged GETS	Other G ETS No Mig	NC DMA GETS	Invalid ate	Ack	Shared Ack	Data	Shared Data	Exclusi ve Data	Writeba ck Ack	Writeba ck Nack	All ack s	All ack s no sh arers	Flush l ine	Block A ck
II	z	z	z	z	z				f l	f l		f l	f l	f l						g s l k d / I	s l kd / I			z	
ST	z	z	z	z	z			j kd / SR	z	z	z	z	z	z										z	
OT	z	z	z	z	z			j kd / OR	z	z	z	z	z	z										z	
MT	z	z	z	z	z			j kd / MR	z	z	z	z	z	z										z	
MMT	z	z	z	z	z			j kd / MMR	z	z	z	z	z	z										z	
MI F	z	z	z		z															hf t s l kd / I				z	
MM F	z	z	z		z				ct p l / IM F	ct p l / IM F	emt l / OM F	et l / OM F	sq l / OM F	ct p l / IM F	m o n							df j kd / MI F	df j kd / MI F	z	df l kd / MI F
IM F	z	z	z	z	z				f l	f l		f l	f l	f l	m o n		uf m o n / ISM F		uf m o n / MM WF					z	
ISM F	z	z	z	z	z										m o n								df j kd / MI F	z	
SM F	z	z	z	z	z				f cc l / IM F	f l		f l	f l	f cc l / IM F	m o n		vt m o n / ISM F		vt m o n / ISM F					z	
OM F	z	z	z	z	z				q p cc l / IM F	et l	emt l	et l	et l	q p cc l / IM F	m o n							df j kd / MI F	df j kd / MI F	z	
MM WF	z	z	z	z	z										m o n								df j kd / MI F	z	

Table 5.14: moesi_hammer - L1Cache - table- part 2/2