TileLink Coherence State-Transition Tables

Paul Loewenstein

December 3, 2019

Copyright 2019 Western Digital Corporation.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

`http://www.apache.org/licenses/LICENSE-2.0`

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

1 Terminology

It is important to refer to TileLink Specification 1.8.0 or later. Earlier versions have incorrect cache state descriptions.

1.1 Cache Line States

This document uses the state names from the TileLink Specification 1.8.0, except that state "TT" has been split into "TT" with read-write access and no branches, and "TB" with read-only access and branches.

```
TT => TIP (no branches)
TB => TIP (with branches)
T  => TRUNK
N  => INVALID
```

1.2 Message Names

Here we invent some names for the useful subset of the similarly named TileLink messages.

- ProbeBlockT, ProbeBlockB and ProbeBlockN are probe requests capping the probee's state to TT, B and N respectively.
- AcquireBlockT and AcquireBlockB are requests from a node in state N for access and data in states TT and B respectively.
- AcquireBlockU is an access request from a node in state B to upgrade to state TT.
- AcquirePermT is a request for write access to a cache line without requesting the current data. This is normally used when the entire cache line is to be written atomically.
- GrantDataT and GrantDataB are data supplied for installation in states TT and B respectively.

- GrantT is for upgrade from B to TT in response to AcquireBlockU. The supplying node can only send GrantT instead of GrantDataT if the responding node can determine that the AcquireU requester is in state B, usually by the responding node maintaining a sufficiently precise directory. GrantT is also used in response to AquirePermT, which never requires a data-carrying response.
- Get is a request for data without any associated access rights.
- PutFullData is for writing a full cache line without installing the cache line in a local cache.
- PutPartialData is for writing a partial cache line without installing the cache line in a local cache.
- AccessAckData supplies data for and acknowledges the completion of Get.
- AccessAck acknowledges the completion of PutFullData or PutPartialData.

1.3 Omissions

1.3.1 No AcquirePermT

There is no table as yet for AcquirePermT, for supporting an allocating whole-cache-line write. It should resemble that for AcquireBlockU, with GrantT replacing GrantDataT.

1.3.2 TL-UL and TL-UH messages on Channels B and C

This document omits the TL-UL and TL-UH messages on Channels B and C. These messages are for implementing an update protocol, which does not support the RVWMO memory model. The RVWMO memory model requires multi-copy write atomicity, which is not maintained by a TileLink update protocol[1].

1.3.3 Hints

This document omits the Intent and HintAck messages. The action to be taken on Intent messages is very system specific and may vary dynamically even on the same system. Load and store prefetching can be handled as for load and store misses, without using Intent and HintAck, at the cost of being forced to prefetch to the cache from which the loads and stores are serviced (usually at the leaf nodes).

1.4 Table Symbols

Tables 3–10 use action symbols described in Table 1.

Symbol	Description	
↑	Input away from root	本层接收
⇑	Output away from root	下层发出
↓	Input towards root	下层接收
⇓	Output towards root	本层发出

Table 1: Input/output up/down action symbols

[1] In fairness to the TileLink V1 designers, the RVWMO memory model had not been finalized when the TileLink protocol was designed.

© 2019 Western Digital Corporation.

2 Servicing a Load Miss

Servicing a load miss involves the requests AcquireBlockB and ProbeBlockB. Table 3 allows a node in state TT to provide exclusive access so that read-modify-write of local data need not result in a store miss. If loads are performed to innermost caches, then state T is not possible and the miss is always serviced via an AcquireBlockB.

2.1 Obtaining Read Access

Table 2 shows the actions and state transitions to obtain read access. The protocol may provide read-write access; this optimizes the case of a read-modify-write to a non-shared location, which would otherwise require two coherence transactions. Depending on the initial cache state, read access is obtained either by issuing an AcquireBlockB towards the root, or by issuing a ProbeBlockB up the trunk. The latter cannot occur in the typical case of a load miss at a leaf node.

Action	Trans. State		Cache State		Clean/Dirty		Notes
	Curr.	Next	Curr.	Next	Curr.	Next	
Load Miss	Idle	ldm1	N		-		
		ldm2	T		C,D		18
⇓AcquireBlockB	ldm1	ldm3	N		-		19
↑GrantDataT	ldm3	ldm4	N	TT	-	C	21
↑GrantDataB	ldm3	ldm4	N	B	-	C	21
⇓GrantAck	ldm4	Idle	TT		C,D		
			B		C		
⇑ProbeBlockB	ldm2	ldm5	T		C,D		7,18
↓ProbeAck	ldm5	Idle	T	TB	C,D		
			TT				9
↓ProbeAckData	ldm5	Idle	T	TB	D		20

Table 2: State transitions for servicing load miss or prefetch

2.2 Receive AcquireBlockB

Table 3 shows the actions and state transitions after a node receives an AcquireBlockB request. Depending on its state it can service the request directly, issue another AcquireBlockB towards the root and/or issue a ProbeBlockB up the trunk.

2.3 Receive ProbeBlockB

Table 4 shows the actions and state transitions after a node receives a ProbeBlockN request. Depending on the node's state the node can either respond directly or issue other ProbeBlockN requests towards the leaves.

© 2019 Western Digital Corporation.

Action	Trans. State		Cache State		Clean/Dirty		Notes
	Curr.	Next	Curr.	Next	Curr.	Next	
↓AcquireBlockB	Idle	aqb1	TT,TB		C,D		
			B		C		
		aqb2	T		C,D		18
		aqb3	N		-		
⇑GrantDataT	aqb1	aqb4	TT	T	C,D		
	aqb8	aqb5					
⇑GrantDataB	aqb1	aqb4	TT,TB	TB	C,D		
			B		C		
	aqb8	aqb5					
⇑ProbeBlockB	aqb2	aqb6	T		C,D		7,18
↓ProbeAck	aqb6	aqb1	T	TB	C,D		
			TT				9
↓ProbeAckData	aqb6	aqb1	T	TB	C,D	D	20
⇓AcquireBlockB	aqb3	aqb7	N		-		19
↑GrantDataT	aqb7	aqb8	N	TT	-	C	21
↑GrantDataB	aqb7	aqb8	N	B	-	C	21
↓GrantAck	aqb4	Idle	TB,T		C,D		
			B		C		
	aqb5	aqb9	T		C,D		
			B		C		
⇓GrantAck	aqb8	aqb1	TT		C,D		
			B		C		
	aqb9	Idle	T		C,D		
			B		C		

Table 3: State transitions for node receiving AcquireBlockB

Action	Trans. State		Cache State		Clean/Dirty		Notes
	Curr.	Next	Curr.	Next	Curr.	Next	
↑ProbeBlockB	Idle	pbb1	TT,TB		C,D		
			N		-		
		pbb2	T		C,D		
⇓ProbeAck	pbb1	Idle	TT,TB	B	C		
			N		-		8
⇓ProbeAckData	pbb1	Idle	TT,TB	B	D	C	
⇑ProbeBlockB	pbb2	pbb3	T		C,D		7
↓ProbeAck	pbb3	pbb1	T	TB	C,D		
↓ProbeAckData	pbb3	pbb1	T	TB	C,D	D	20

Table 4: State transitions for node receiving ProbeBlockB

3 Servicing a Store Miss

Servicing a store miss involves obtaining read/write access using the requests AcquireBlockT and ProbeBlockN. It is assumed that the store is to less than a cache line, so the cache line contents need to be obtained.

© 2019 Western Digital Corporation.

3.1 Obtaining Read-Write Access

Table 5 shows the actions and state transitions for obtaining read-write access at any node in the tree. Typical systems only perform this operation at leaf nodes, which can only be in state B with no branches, or state N; in this case the only reachable states are stm5, stm13 and stm12.

Action	Trans. State		Cache State		Clean/Dirty		Notes
	Curr.	Next	Curr.	Next	Curr.	Next	
Store Miss	Idle	stm1	TB		C,D		18
		stm2	B		C		3,18
		stm3					4
		stm4	T		C,D		18
		stm3	N		-		
⇑ProbeBlockN	stm1	stm5	TB		C,D		5,18
	stm13	stm6			C		
	stm2	stm7	B				
	stm12	stm8					
	stm4	stm9	T		C,D		7,18
↓ProbeAck	stm5		TT,TB		C,D		9,10,18
	stm6						
	stm7		B		C		10,18
	stm8						
	stm5	Idle	TT,TB	TT	C,D		9,11
	stm6	stm10					
	stm7	stm3	B		C		11
	stm8	stm11					11,19
	stm9	Idle	TT,T	TT	C,D		9
↓ProbeAckData	stm9	Idle	T	TT	C,D	D	20
⇓AcquireBlockU	stm2	stm12	B		C		18
	stm7	stm8					
	stm3	stm11					19
⇓AcquireBlockT	stm3	stm11	N		-		19
↑GrantT	stm12	stm13	B	TB	C		18,22
	stm8	stm6					
	stm11	stm10		TT			22
↑GrantDataT	stm11	stm10	N	TT	-	C	21
⇓GrantAck	stm13	stm1	TB		C		18
	stm6	stm5					
	stm10	Idle	TT		C,D		

Table 5: State transitions for servicing a store miss or prefetch

3.2 Receive AcquireBlockT

Table 6 shows the actions and state transitions after a node receives an AcquireBlockT request. Depending on its state it can service the request directly, issue an AcquireBlockT or AcquireBlockU towards the root and/or issue a ProbeBlockN towards the leaves before supplying GrantDataT to the requester.

© 2019 Western Digital Corporation.

Action	Trans. State		Cache State		Clean/Dirty		Notes
	Curr.	Next	Curr.	Next	Curr.	Next	
↓AcquireBlockT	Idle	aqt1	TT,TB		C,D		1
		aqt2	TB				2,18
		aqt3	B		C		
		aqt4					1
		aqt5	T		C,D		18
		aqt4	N		-		
⇑ProbeBlockN	aqt2	aqt6	TB		C,D		6,18
	aqt14	aqt7			C		
	aqt3	aqt8	B				
	aqt13	aqt9					
	aqt5	aqt10	T		C,D		7,18
↓ProbeAck	aqt6		TT,TB		C,D		9,10,18
	aqt7						
	aqt8		B		C		10,18
	aqt9						
	aqt6	aqt1	TB	TT,TB	C,D		11,17
			TT				9,11
	aqt7	aqt11					
	aqt8	aqt4	B		C		11
	aqt9	aqt12					11,19
	aqt10	aqt1	T	TT	C,D		
↓ProbeAckData	aqt10	aqt1	T	TT	C,D	D	20
⇓AcquireBlockU	aqt3	aqt13	B		C		18
	aqt8	aqt9					
	aqt4	aqt12					19
⇓AcquireBlockT	aqt4	aqt12	N		-		19
↑GrantT	aqt13	aqt14	B	TB	C		18,22
	aqt9	aqt7					
	aqt12	aqt11					22
↑GrantDataT	aqt12	aqt11	N	TT	-	C	21
⇑GrantDataT	aqt1	aqt15	TT,TB	T	C,D		
	aqt11	aqt16	TT				
			TB		C		
↓GrantAck	aqt15	Idle	T		C,D		
	aqt16	aqt17					
⇓GrantAck	aqt14	aqt2	TB		C		18
	aqt7	aqt6					
	aqt11	aqt1	TT		C,D		
	aqt16	aqt15	T		C		
	aqt17	Idle			C,D		

Table 6: State transitions for node receiving AcquireBlockT

3.3 Receive AcquireBlockU

Table 7 shows the actions and state transitions after a node receives an AcquireBlockU request. Depending on its state it can service the request directly, issue another an AcquireBlockT or AcquireBlockU towards the root and/or issue a ProbeBlockN towards the leaves before supplying GrantDataT or GrantT to the requester.

© 2019 Western Digital Corporation.

Action	Trans. State		Cache State		Clean/Dirty		Notes
	Curr.	Next	Curr.	Next	Curr.	Next	
↓AcquireBlockU	Idle	aqu1	TT,TB		C,D		1
		aqu2	TB				2,18
		aqu3	B		C		
		aqu4					1
		aqu5	T		C,D		18
		aqu4	N		-		
⇑ProbeBlockN	aqu2	aqu6	TB		C,D		6,18
	aqu14	aqu7			C		
	aqu3	aqu8	B				
	aqu13	aqu9					
	aqu5	aqu10	T		C,D		7,18
↓ProbeAck	aqu6		TT,TB		C,D		9,10,18
	aqu7						
	aqu8		B		C		10,18
	aqu9						
	aqu6	aqu1	TB	TT,TB	C,D		11,17
			TT				9,11
	aqu7	aqu11					
	aqu8	aqu4	B		C		11
	aqu9	aqu12					11,19
	aqu10	aqu1	T	TT	C,D		
↓ProbeAckData	aqu10	aqu1	T	TT	C,D	D	20
⇓AcquireBlockU	aqu3	aqu13	B		C		18
	aqu8	aqu9					
	aqu4	aqu12					19
⇓AcquireBlockT	aqu4	aqu12	N		-		19
↑GrantT	aqu13	aqu14	B	TB	C		18,22
	aqu9	aqu7					
	aqu12	aqu11					22
↑GrantDataT	aqu12	aqu11	N	TT	-	C	21
⇑GrantT	aqu1	aqu15	TB	T	C,D		23
	aqu11	aqu16			C		
⇑GrantDataT	aqu1	aqu15	TT	T	C,D		
	aqu11	aqu16					
↓GrantAck	aqu15	Idle	T		C,D		
	aqu16	aqu17					
⇓GrantAck	aqu14	aqu2	TB		C		18
	aqu7	aqu6					
	aqu11	aqu1	TT				
	aqu16	aqu15	T				
	aqu17	Idle			C,D		

Table 7: State transitions for node receiving AcquireBlockU

3.4 Receive ProbeBlockN

Table 8 shows the actions and state transitions after a node receives a ProbeBlockN request. Depending on the node's state the node can either respond directly or issue other ProbeBlockN requests towards the leaves.

© 2019 Western Digital Corporation.

Action	Trans. State		Cache State		Clean/Dirty		Notes
	Curr.	Next	Curr.	Next	Curr.	Next	
↑ProbeBlockN	Idle	pbn1	TT		C,D		
		pbn2	TB				
		pbn3	T				
		pbn2	B		C		
		pbn1	N		-		
⇓ProbeAck	pbn1	Idle	TT,B	N	C	-	
			N		-		8
⇓ProbeAckData	pbn1	Idle	TT	N	D	-	
⇑ProbeBlockN	pbn2	pbn4	TB		C,D		5
			B		C		
	pbn3	pbn5	T		C,D		7
↓ProbeAck	pbn4		TB		C,D		10
			B		C		
	pbn4	pbn1	TB	TT	C,D		11
			B		C		
	pbn5		T	TT	C,D		
↓ProbeAckData	pbn5	pbn1	T	TT	C,D	D	20

Table 8: State transitions for node receiving ProbeBlockN

4 Victimization

The TileLink specification is not specific on victimization policy, especially on the possibility of silent victimization. The rules below define as liberal a policy as seems reasonable.

- A node may only victimize its entry if the entry is in state TT or B.
- A node may only victimize its entry if there are no branches from that entry.
- A node may victimize from state B silently, (without sending Release).
- To victimize from state TT the node must issue a Release (for clean data) or ReleaseData (for dirty data) towards root.

4.1 Evicting a Chosen Victim

Table 9 shows the actions and state transitions to evict a cache line from a node. Victimization can be initiated from any node except the root node.

4.2 Receiving Release or ReleaseData

Table 10 shows the actions and state transitions when a node receives an Release or ReleaseData request. Because TileLink systems have inclusive caches, the request can be serviced directly without involving any other nodes.

© 2019 Western Digital Corporation.

Action	Trans. State		Cache State		Clean/Dirty		Notes
	Curr.	Next	Curr.	Next	Curr.	Next	
Victim chosen	Idle	vct1	TT		C,D		
		vct2	TB				18
		vct3	T				
		vct1	B		C		4
		vct4					3,18
	Idle		B	N	C	-	4
⇓Release	vct1	vct5	TT,B	N	C	-	
⇓ReleaseData	vct1	vct5	TT	N	D	-	
⇑ProbeBlockN	vct2	vct6	TB		C,D		5,18
	vct4	vct7	B		C		
	vct3	vct8	T		C,D		7,18
↓ProbeAck	vct6		TT,TB		C,D		9,10,18
	vct7		B		C		10,18
	vct6	vct1	TT,TB	TT	C,D		9,11
	vct7		B		C		11
		Idle	B	N	C	-	
	vct8	vct1	TT,T	TT	C,D		9
↓ProbeAckData	vct8	vct1	T	TT	C,D	D	20
↑ReleaseAck	vct5	Idle	N		-		

Table 9: State transitions for victimizing a cache entry

Action	Trans. State		Cache State		Clean/Dirty		Notes
	Curr.	Next	Curr.	Next	Curr.	Next	
↓Release	Idle	rel1	TB		C,D		
			TB	TT			
			T				
			B		C		
↓ReleaseData	Idle	rel1	T	TT	C,D	D	
⇑ReleaseAck	rel1	Idle	TT,TB		C,D		
			B		C		

Table 10: State transitions for node receiving Release or ReleaseData

5 Non-Allocating Requests

Non-allocating requests are normally used for direct memory access (DMA) reads and writes by IO devices. Reads and writes use TL-UL messages. TL-UH adds non-allocating read-modify-write operations. How these operations interact with the caches is somewhat system specific. This document assumes that:

- A non-allocating read
  - Is serviced from a node in state TT, TB, B or T.
  - Causes no cache state change anywhere in the system.
- A non-allocating write is serviced from a node in state TT. This node may have been upgraded by the non-allocating write from state TB or T, thereby rendering all other nodes to be in state T or N.

If the node initiating a non-allocating request has a cacheless path to the root, then the request is serviced from the root node. Logically, all nodes on this path have cache state N.

© 2019 Western Digital Corporation.

As well as IO devices, RISC-V cores can also use non-allocating requests to avoid cache pollution or to obviate the need for caches.

5.1 Receiving a Get Request

The servicing of a Get request is described in Table 11. A Get request is serviced directly if the node is in state TT, TB or B. If the node is in state T it is serviced by issuing a ProbeBlockT up the trunk. If the node is in state N it is serviced by issuing a Get towards the root.

Action	Trans. State		Cache State		Clean/Dirty		Notes
	Curr.	Next	Curr.	Next	Curr.	Next	
↓Get	Idle	get1	TT,TB		C,D		
			B		C		
		get2	T		C,D		18
		get3	N		-		
⇑AccessAckData	get1	Idle	TT,TB,T		C,D		14
			B		C		
	get5		T		C,D		15
			N		-		16
⇑ProbeBlockT	get2	get4	T		C,D		7,18
↓ProbeAckData	get4	get5	T		C,D		
↓ProbeAck	get4	get1	TT		C,D		9
			T				
⇓Get	get3	get6	N		-		19
↑AccessAckData	get6	get5	N		-		

Table 11: State transitions for node receiving Get

5.2 Receiving a ProbeBlockT Request

The servicing of a ProbeBlockT request is shown in Table 12. A ProbeBlockT can only be received by a node in state TT, TB, T or N. If received in state N, it is necessarily racing with a release. If in state T, the request is serviced by issuing another ProbeBlockT up the trunk, otherwise it is serviced directly.

Action	Trans. State		Cache State		Clean/Dirty		Notes
	Curr.	Next	Curr.	Next	Curr.	Next	
↑ProbeBlockT	Idle	pbt1	TT,TB		C,D		
			N		-		
		pbt2	T		C,D		
⇓ProbeAck	pbt1	Idle	TT,TB,T		C		
			N		-		8
⇓ProbeAckData	pbt1	Idle	TT,TB,T		D		12
	pbt4		T		C,D		13
⇑ProbeBlockT	pbt2	pbt3	T		C,D		7
↓ProbeAckData	pbt3	pbt4	T		C,D		
↓ProbeAck	pbt3	pbt1	TT,T		C,D		9

Table 12: State transitions for node receiving ProbeBlockT

© 2019 Western Digital Corporation.

5.3 Receiving a PutPartialData Request

The servicing of a PutPartialData request is shown in Table 13. The policy enshrined in this table is optimized for simplicity over efficiency for the case of meeting a cache in state B. A cache in state B is treated identically to one in state N—the PutPartialData is forwarded towards the root. Once the AccessAck is received, the cache will have been invalidated by a ProbeBlockN.

Action	Trans. State		Cache State		Clean/Dirty		Notes
	Curr.	Next	Curr.	Next	Curr.	Next	
↓PutPartialData	Idle	ptp1	TT		C,D		
		ptp2	TB				18
		ptp3	B		C		
		ptp4	T		C,D		18
		ptp3	N		-		
⇑AccessAck	ptp1	Idle	TT		C,D	D	24
			N		-		
⇑ProbeBlockN	ptp2	ptp5	TB		C,D		5,18
	ptp4	ptp6	T				7,18
↓ProbeAck	ptp5		TB		C,D		10,18
	ptp5	ptp1	TT,TB	TT			9,11
	ptp6		TT,T				9
↓ProbeAckData	ptp6	ptp1	T	TT	C,D	D	20
⇓PutPartialData	ptp3	ptp7	B		C		19
			N		-		
↑AccessAck	ptp7	ptp1	N		-		

Table 13: State transitions for node receiving PutPartialData

5.4 Receiving a PutFullData Request

The servicing of a PutFullData request is shown in Table 14. The policy enshrined in this table is optimized for simplicity over efficiency for the case of meeting a cache in state B. A cache in state B is treated identically to one in state N—the PutFullData is forwarded towards the root. Once the AccessAck is received, the cache will have been invalidated by a ProbePermN.

5.5 Receiving a ProbePermN Request

Table 15 shows the actions performed upon receipt of a ProbePermN request, which may be issued by a node processing a PutFullData request.

© 2019 Western Digital Corporation.

Action	Trans. State		Cache State		Clean/Dirty		Notes
	Curr.	Next	Curr.	Next	Curr.	Next	
↓PutFullData	Idle	ptf1	TT		C,D		
		ptf2	TB				18
		ptf3	B		C		
		ptf4	T		C,D		18
		ptf3	N		-		
⇑AccessAck	ptf1	Idle	TT		C,D	D	25
			N		-		
⇑ProbePermN	ptf2	ptf5	TB		C,D		5,18
	ptf4	ptf6	T				7,18
↓ProbeAck	ptf5		TB		C,D		10,18
	ptf5	ptf1	TT,TB	TT			9,11
	ptf6		TT,T				9
⇓PutFullData	ptf3	ptf7	B		C		19
			N		-		
↑AccessAck	ptf7	ptf1	N		-		

Table 14: State transitions for node receiving PutFullData

Action	Trans. State		Cache State		Clean/Dirty		Notes
	Curr.	Next	Curr.	Next	Curr.	Next	
↑ProbePermN	Idle	ppn1	TT		C,D		
		ppn2	TB				
		ppn3	T				
		ppn2	B		C		
		ppn1	N		-		
⇓ProbeAck	ppn1	Idle	TT	N	C,D	-	
			B		C		
			N		-		8
⇑ProbePermN	ppn2	ppn4	TB		C,D		5
			B		C		
	ppn3	ppn5	T		C,D		7
↓ProbeAck	ppn4		TB		C,D		10
			B		C		
	ppn4	ppn1	TB	TT	C,D		11
			B		C		
	ppn5		T	TT	C,D		

Table 15: State transitions for node receiving ProbePermN

6 Notes to Tables

1. There are either no branches, or AcquireBlock was received from the only branch.
2. There are one or more branches from which the AcquireBlock was not received.
3. There are branches from this node.
4. There are no branches from this node.
5. ProbeBlockN is sent up all branches.

© 2019 Western Digital Corporation.

6. ProbeBlockN is sent up all branches except the requesting branch.
7. Probe is sent up the trunk.
8. ProbeAck may not be issued while awaiting ReleaseAck for same cache line address.
9. Current cache state TT indicates that either Release or ReleaseData was received before ProbeAck.
10. ProbeAck is not the last expected ProbeAck.
11. ProbeAck is the last expected ProbeAck.
12. ProbeAckData data is supplied from cache.
13. ProbeAckData data is supplied from received ProbeAckData.
14. AccessAckData data is supplied from cache.
15. AccessAckData data is supplied from received ProbeAckData.
16. AccessAckData data is supplied from received AccessAckData.
17. With a precise directory, TB indicates that the AcquireBlockT request was received from a branch with a valid copy of the data.
18. Incoming probes to this transaction's address are stalled until this transaction's probes have been acknowledged.
19. In the next transaction state, incoming probes to this transaction's address must be serviced, possibly causing a cache state downgrade from B to N.
20. Update cache with data supplied by ProbeAckData.
21. Install GrantData data in cache.
22. GrantDataT may be received instead of GrantT; in which case the received data may be either discarded or installed.
23. If this node does not have a sufficiently precise directory to determine that the requesting branch has the data, then GrantDataT may be sent instead of GrantT.
24. Cache data is updated using the data and mask in PutPartialData request.
25. Cache data is overwritten with the data in PutPartialData request.

© 2019 Western Digital Corporation.