

A2 Report

Yezheng Shao

1 Baseline protocol - MSI

The baseline protocol is based on the MSI protocol introduced in the Primer on Memory Consistency and Cache Coherence. There are three additional directory controller states.(MX_R, SM_A, MM_A). Also, an additional FwdAck acknowledgement for the directory controller. No changes to the cache controller. MX_R means the directory is transforming to S or I but is waiting for a response, which can be either FwdAck or Data, to complete its transformation. See Figure 1 and Figure 2 in the Appendix for the detailed protocol table.

1.1 Challenge

1. The original protocol doesn't comply with the out-of-order channel. Racing would occur while forwarding requests from the directory.
2. Processor request from S to M is a 4-hop operation since the requester needs an extra step to send an acknowledgement to the directory for it to complete the transition from SM_A to M.

1.2 Solution

1. To avoid the racing scenario, the directory will change to a transient state (MM_A or MX_R) after it sends a forward request (Fwd-GetM, Fwd-GetS) to the current owner. Then the directory will stall in the transient state until it receives the FwdAck or Data from the owner. An Fwd-Ack is sent from the owner to the directory once the owner forwards its data to the requesting node. Once the directory controller receives the FwdAck or data, it completes the previous request and updates its state.
2. To simplify the GetM operation for the processor in the S state, the invalidated nodes need to send the InvAck to the directory as well. Therefore, the requesting node won't need an additional hop to send an acknowledgement to the director. However, the directory would also need an extra field to keep track of how many InvAck it needs to receive.

2 Optimized Protocol - MESI

By adding an extra E state to the baseline protocol, the processor that has exclusive read access won't need to send the GetM request to the directory to gain the write permission. It can silently change itself to M and write to the cache directly. See Figures 3 and 4 in the Appendix for the protocol table.

3 Verification

To verify the protocol, invariants and rules are implemented to simulate the model. The rules include how the processors would react when they receive requests in a stable state, and how the network is set up. An InBox is implemented to simulate the message queue in the channel. Also, the start state is initialized as I state. Invariants are implemented to verify the protocol meets the cache coherence and SWMR policy. A value field is included in the directory controllers and cache controllers to keep track of the last written value.

4 Appendix

Figure 1: MSI directory protocol – directory controller

	GetS	GetM	PutS-Not Last	PutS-Last	PutM + data from Owner	PutM + data from Non-Owner	Data	FwdAck	InvAck
I	Send data to Req, add Req to shares/ S	Send data to Req, set Owner to Req/M	Send PutAck to Req	Send PutAck to Req		Send PutAck to Req			
S	Send data to Req, add Req to Sharers	Send data to Req, send Inv to Shares, clear Shares, set Owner to Req/ SM_A	Remove Req from Sharers, send PutAck to Req	Remove Req from Req from Sharers, send PutAck to Req / I		Remove Req from Sharers, send PutAck to Req			
M	Send FwdGetS to Owner, add Req and Owner to Sharers, clear Owner/ MX_D	Send FwdGetM to Owner, set Owner to Req/ MM_A	Send Put-Ack to Req	Send Put-Ack to Req	Copy data to memory, clear Owner, send PutAck to Req/I	Send Put-Ack to Req			
MX_R	Stall	Stall	Stall	Stall		Stall	- account =0 / I - account > 1 / S	- account =0 / I - account > 1 / S	
SM_A	Stall	Stall	Stall	Stall	Stall	Stall	Stall	Stall	account--
MM_A	Stall	Stall	Stall	Stall	Stall	Stall	Stall	-/M	

Figure 2: MSI directory protocol – cache controller

	load	Store	Evict	FwdGet S	FwdGet M	Inv	Put Ack	Data from Dir (ack = 0)	Data from Dir (ack > 0)	Data from Owner	InvAck Last
I	Send GetS to Dir/IS_ D	Send GetM to Dir/IM_ AD									
IS_D	Stall	Stall	Stall			Stall		-/S		-/S	
IM_A	Stall	Stall	Stall	Stall	Stall						-/M
IM_AD	Stall	Stall	Stall	Stall	Stall			-/M	-/IM_A	-/M	
S	Hit	Send GetM to Dir/ SM_AD	Send PutS to Dir / SI_A			Send InvAck to Req and Dir/ I					
SM_A	Hit	Stall	Stall	Stall	Stall						-/M
SM_AD	Hit	Stall	Stall	Stall	Stall	Send InvAck to Req and Dir/IM_AD		-/M	-/SM_A		
M	Hit	Hit	Send putM + Data to Dir/ MI_A	Send Data to Req and Dir/ S	Send Data to Req and FwdAck to Dir / I						
SI_A	Stall	Stall	Stall			Send InvAck to Req and Dir/ I	-/I				
MI_A	Stall	Stall	Stall	Send data to Req and Dir/SI_ A	Send data to Req/II_ A		-/I				
II_A	Stall	Stall	Stall				-/I				

Figure 3: MESI directory protocol – directory controller


	GetS	GetM	PutS- 	PutS-Last	PutE + data from owner	PutE + data from Non-Owner	PutM + data from Owner	PutM + data from Non-Owner	Data	FwdAck	InvAck
I	Send data to Req, add Req to shares/S	Send data to Req, set Owner to Req/M	Send PutAck to Req	Send PutAck to Req		Send PutAck to Req		Send PutAck to Req			
S	Send data to Req, add Req to Sharers	Send data to Req, send Inv to Sharers, clear Shares, set Owner to Req/SM_A	Remove Req from Sharers, send PutAck to Req	Remove Req from Req from Sharers, send PutAck to Req / I		Remove Req from Sharers, send PutAck to Req		Remove Req from Sharers, send PutAck to Req			
M	Send FwdGet S to Owner, add Req and Owner to Sharers, clear Owner/MX_R	Send FwdGet M to Owner, set Owner to Req	Send Put-Ack to Req	Send Put-Ack to Req		Send Put-Ack to Req	Copy data to memory, clear Owner, send PutAck to Req/I	Send Put-Ack to Req			
E	Same as M	Same as M	Send Put-Ack to Req	Send Put-Ack to Req	Clear Owner, send PutAck to Req/I	Send Put-Ack to req	Same as M	Send Put-Ack to Req			
MX_R	Stall	Stall	Remove Req from Sharers, send Put-Ack to Req	Remove Req from Sharers, send Put-Ack to Req				Stall	-account = 0 / I -account > 1 / S	account = 0 / I account > 1 / S	
SM_A	Stall	Stall	Stall	Stall		Stall	Stall	Stall	Stall	-/M	account-
MM_A	Stall	Stall	Send PutAck to Req	Send PutAck to Req		Stall	Stall	Stall	Stall	-/M	

Figure 4: MESI directory protocol – cache controller

	load	Store	Evict	FwdGetS	FwdGetM	Inv	PutAck	E-Data from Dir	Data from Dir (ack = 0)	Data from Dir (ack > 0)	Data from Owner	InvAck Last
M	Hit	Hit	Send putM + Data to Dir/ MI_A	Send Data to Req and Dir/ S	Send Data to Req and FwdAck to Dir/ I							
S	Hit	Send GetM to Dir/ SM_AD	Send PutS to Dir/ SI_A			Send InvAck to Req and Dir/ I						
I	Send GetS to Dir/IS_D	Send GetM to Dir/IM_AD										
E	Hit	-/M	Send putE to Dir/ MI_A	Send Data to Req and Dir/S	Send Data to Req and FwdAck to Dir/ I							
IS_D	Stall	Stall	Stall	Stall	Stall	Stall		-/E	-/S		-/S	
IM_A	Stall	Stall	Stall	Stall	Stall							-/M
IM_AD	Stall	Stall	Stall	Stall	Stall				-/M	-/IM_A	-/M	
II_A	Stall	Stall	Stall				-/I					
SM_A	Hit	Stall	Stall	Stall	Stall							Send FwdAck to Dir/ /M
SM_AD	Hit	Stall	Stall	Stall	Stall	Send InvAck to Req and Dir/IM_AD			-/M	-/SM_A		
SI_A	Stall	Stall	Stall		Send InvAck to Req/ II_A	Send InvAck to Req and Dir/I						
MI_A	Stall	Stall	Stall	Send data to Req and Dir/SI_A	Send data to Req/II_A		-/I					