ECE552 PRA0102 - Station 8
Ting-Hao (Tim) Cheng (997289090)
Isaiah Tan (999004937)

Section 5 Questions

1. How does the FSM know it has received the last Inv Ack reply for a TBE entry?
The Directory keeps track of all cores sharing the block, so the cache controller can get the total number of sharers and know how many Inv Acks to expect.

2. How is it possible to receive a PUTM request from a NonOwner core? Please provide a set of events that will lead to this scenario.

1. Core2 is the owner of M block.
2. Core1.GETM -> Directory updates owner to core1
3. Core2.Replace -> Core2 replaces the M block and issue PUTM (Happens before core2.Fwd_GetM)
4. Core2 receives Fwd_GetM from Core1 and forward data

3. Why do we need to differentiate between a PUTS and a PUTS-Last request?
So the Directory can know when to invalidate the block. It needs to keep the block until it receives PUTS-Last (until there are no more sharers).

4. How is it possible to receive a PUTS Last request for a block in modified state in the directory? Please provide a set of events that will make this possible.
1. The block is in the Shared state to begin with
2. Core1.GETM -> Directory updates state S->M right away, and Inv is sent to all sharers minus the requestor
3. One of the sharers replaces the S block before receiving Inv

5. Why is it not possible to get an Invalidation request for a cache block in a Modified state? Please explain.
Because a cache block in Modified state has only one owner and no sharer. If the directory receives a GETM or GETS request,. the request is forwarded to the owner; inv is not sent.

6. Why is it not possible for the cache controller to get a Fwd-GetS request for a block in SI_A state? Please explain.
Because only the owner will receive Fwd-GetS. When you are in SI_A state, you are a sharer and thus not responsible for sending data.

7. Was your verification testing exhaustive? How can you ensure that the random tester has exercised all possible transitions (i.e., all {State, Event} pairs)?

Our verification testing was not that exhaustive. The simulator would complain if we did not cover a required transition; however, it would not complain if a transition is not entered. What we did was we tested progressively (until a bug is hit) with increasing number of cores, increasing set of instructions, and decreasing cache sizes. This would increase the number of replacement calls concurrent to the normal load/stores. This increases the chances of entering different states and transitions, but it still does not guarantee that all cases would be covered.

Section 6 Questions

1. Why are transient states necessary?
Because in reality state changes don't happen immediately or atomically, so we need to handle situations where the block is no longer in a valid state but has yet finished entering the next state. The reason transitions are not immediate is the need to wait for data (from other caches or memory), or wait for acknowledge (from directory/mem/caches). Each transient state is created to wait for one specific event.

2. Why does a coherence protocol use stalls?
Stalls are necessary to make sure operations run in the proper states (no intermediate states) and order. This is important for enforcing coherence rules and invariants.

3. What is deadlock and how can we avoid it?
Deadlock happens when a system cannot make forward progress because of two inter-dependent events. Deadlock can be avoided by using virtual networks, which separate messages and the resources used to manage these messages by types.

4. What is the functionality of Put-Ack (i.e., WB-Ack) messages? They are used as a response to which message types?
Put-Ack or WB-Ack is used by the Directory to tell cache controllers that a data writeback was successful, so that the cache controllers can safely discard the data. It is used as a response to the PUT message types (PutS and PutM)

5. What determines who is the sender of a Data reply? Which are the possible options?
The state of the block determines the sender of a data reply. The possible senders are a local cache, a remote cache, or a directory.

6. What is the difference between a Put-Ack and an Inv-Ack message?
A Put-Ack means that a data writeback happened and cache blocks are invalidated, whereas a Inv-Ack only means that cache blocks are invalidated.

ECE552 PRA0102 - Station 8
Ting-Hao (Tim) Cheng (997289090)
Isaiah Tan (999004937)

<u>Protocol Modifications</u>

Table	State,Event	Modification
8.1	I, Inv	Send Inv_Ack instead of do nothing
8.1	I, Put_Ack	Pop the message from queue instead of do nothing
8.2	S^D, all events	Split to MS_D (waiting for modified data block from cache) and MS_A (waiting for WB acknowledge from mem). Old owner is removed at MS_A.
8.2	I, GETM	Added transition state IM_D. (I->M, waiting for data to be read from mem).
8.2	I, GETS	Added transition state IS_D. (I->S, waiting for data to be read from mem).
8.2	M, PutM+data from Owner	Added transition state MI_A (M->I, waiting for WB acknowledge from mem). Transition to MI_A after sending WB request.
8.2	{MS_D, MS_A}, PutM+data from Owner	Old owner wants to replace. Remove as sharer, send put-ack.
8.2	{MS_D, MS_A}, PUTS_Last	Used stalls instead. In the case PUTS_Last is sent, the last sharer wants to invalidate. Should wait for stable state (S), not force MS into MI

<u>Work Delegation</u>

<u>Isaiah:</u>
State definitions
Directory transitions
lab report

<u>Tim:</u>
Cache transitions
lab report