CS4223 Tutorial 3: Cache Coherence

Question 1:
The cache coherence mechanism for a multiprocessor system used a MESI protocol. Consider a system with two processors, P1 and P2, with the initial cache state shown in the following table. For this problem, assume each cache holds only 4 lines and uses direct-mapped organization. Two memory lines with the same set (e.g., L1 and L5) map to the same cache line.

P1		Set	P2	
Line	State		Line	State
L1	M	0	L5	I
L2	E	1	L6	M
L3	S	2	L3	S
L4	I	3	L8	E

(A) What is the state of each cache after the following sequence of memory references is completed? Fill in the table below.

P2 reads line L1
P1 writes line L2
P2 writes line L3
P1 reads line L8

P1		Set	P2	
Line	State		Line	State
L1	S	0	L1	S
L2	M	1	L6	M
L3	I	2	L3	M
L8	S	3	L8	S

(B) Assume that the caches are returned again to the original state above. Describe a simple action by P2 that would leave an exclusive copy of line L3 in P1's cache even though the cache state would be S.

L3 gets evicted from P2's cache.

Question 2:
Suppose we are designing new **invalidation-based** cache coherence protocol for bus based shared memory multiprocessors with write-back caches. This protocol uses 5 states.
Modified (M): This cache has the only copy, and main memory is not up-to-date.
Shared-Modified (Sm): Other caches have a (shared) copy of this cache line, but the copy in main memory is not up-to-date and this cache is the owner (responsible for supplying data).
Shared-Clean (Sc): This cache has a copy of the data, but other caches presumably do, too. Main memory may or may not (e.g., if someone else has it as Sm) have an up-to-date value.
Exclusive (E): This is the only cache with a copy of this data, but main memory also has the correct, up-to-date data.
Invalid (I): This cache does not have a copy of the data.

The *Shared-modified* state makes sense if caches have a way to send data directly to one another, without involving (slower) main memory. For example, if I have address *a* in my cache as *Modified*, and I see (via snooping) some other cache trying to read address *a*, I can send the data directly to that cache (which goes to the *Shared-Clean* state), and then I transition to *Shared-modified* state.

(A) In the following table, given a pair of cache states corresponding to the same memory block, fill up the permitted possibilities with YES and the remaining ones with NO. It is possible for two caches to be in the invalid (I) state together for a memory block. So we have YES for the cell (I-I). Fill up the remaining cells.

	I	E	Sc	Sm	M
I	YES	YES	YES	YES	YES
E	YES	NO	NO	NO	NO
Sc	YES	NO	YES	YES	NO
Sm	YES	NO	YES	NO	NO
M	YES	NO	NO	NO	NO

(B) Draw the state transition diagram for this new cache coherence protocol. Remember that you are designing an **invalidation-based** cache coherence protocol, i.e., **on a write**, other cache copies should be invalidated rather than updated.

For this, you can map S state to Sc state. The only additional state is Sm. Here are the extra transitions:

M → Sm on BusRd/supply data to other cache but not flush to main memory

Sm → Sm on PrRd
Sm → M on PrWr/BusRdEx to invalidate other copies

Sm → I on BusRdEx (supply data to the other cache in case it does not have a copy because BusRdEx can be generated from both I and S state in the other cache)
Sm → Sm on BusRd/supply data to other cache but not flush to main memory

(C) Explain the actions that need to be taken when a cache line is evicted corresponding to the four different states (E, Sc, Sm, M).

ANS:

E state, Sc state: do nothing.

Sm state, M state: write to main memory.

Question 3:

The write-once cache coherence protocol is a variant of the MESI cache coherence protocol. **Write-once is an invalidation-based cache coherence protocol for write-back, write allocate caches.** In this protocol, each cache line can be in any of the following four states.

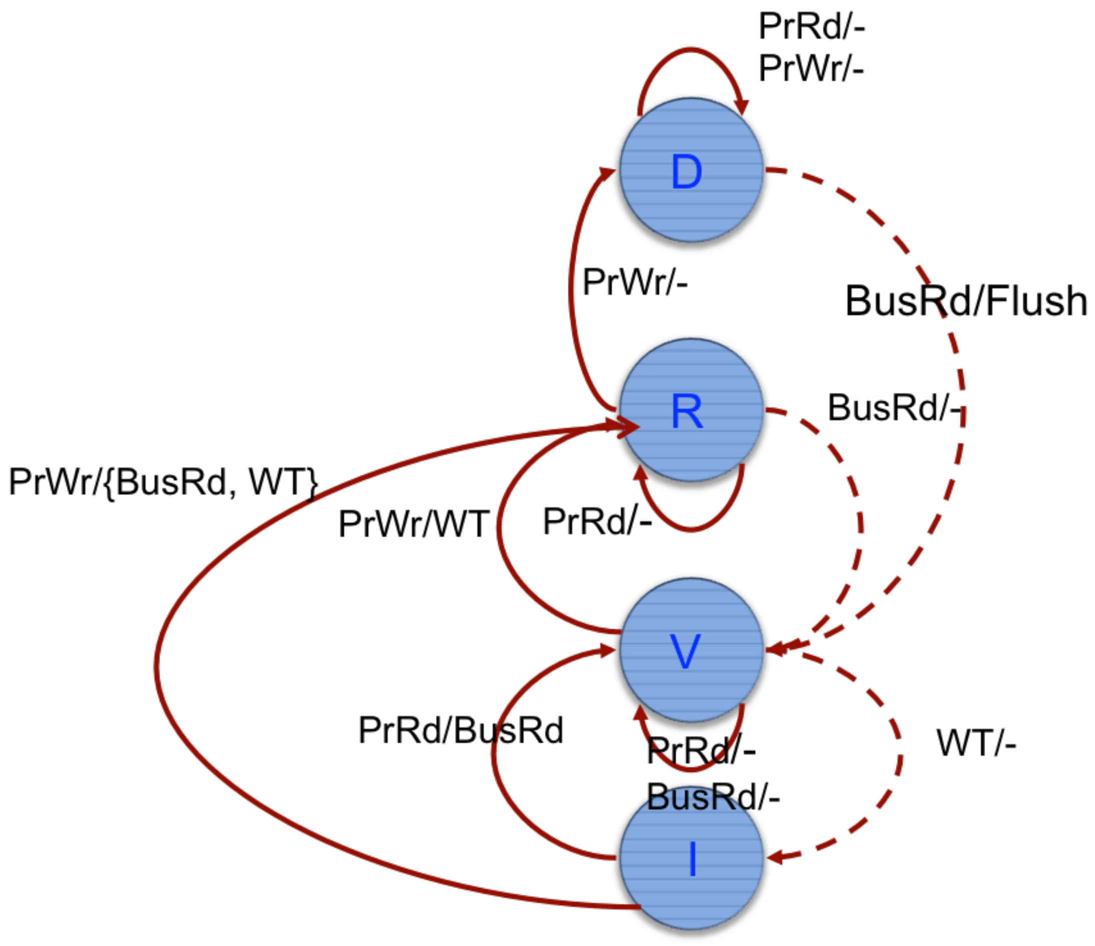
Invalid (I): This cache does not have a copy of the memory block. This is same as the invalid state in MESI protocol.
Valid (V): This cache has a coherent copy of the memory block. The memory blocks may possibly be shared with other caches, but the content is not modified. This is same as the shared state in MESI protocol.
Reserved (R): This cache has the only copy of the memory block, but it is still coherent with the main memory. This is similar to the exclusive state in MESI protocol.
Dirty (D): This cache has the only copy of the memory block, but it is not coherent with the main memory. This is same as the modified state in MESI protocol.

However, the main difference between MESI and write-once protocol is that the **write-once protocol does not have BusRdEx (bus read exclusive) transaction or the shared (S) signal** that MESI requires. Instead, the **first write to a cache line (either in Invalid or Valid state) generates a write-through** to main memory, which implicitly invalidates other cache lines containing the same memory block. The protocol performs write-back on all subsequent writes, reducing overall bus traffic in consecutive writes. Thus the transitions to some of the cache states happen under different scenarios compared to the MESI protocol.

A) Indicate the legal and illegal pair of cache states for any memory block in the following table.

	I	V	R	D
I				
V			X	X
R		X	X	X
D		X	X	X

B) Draw the state transition diagram corresponding to the write-once cache coherence protocol. Please state your assumptions, if any, clearly. [Hint: Think how the transition in and out of the Reserved state should happen.]



WT stands for write through or BusWrite as mentioned during tutorial.

C) Explain the advantage and disadvantages of write-once cache coherence protocol compared to the MESI cache coherence protocol.

Write-once protocol does not need additional shared signal and BusRdEx transaction. Specially the absence of the shared signal simplifies the design.

However, private read followed by multiple writes will still generate bus read followed by write-through traffic on first write, which does not happen in MESI protocol. Private read followed by multiple writes generate only one bus read in MESI.

D) Which of the cache states will generate a write-back transaction when a memory block is evicted from the cache?

Only the D state needs to generate write-back transaction when a memory block is evicted from the cache.