4 states: MESI

4-state MESI write-back, Invalidate Protocol

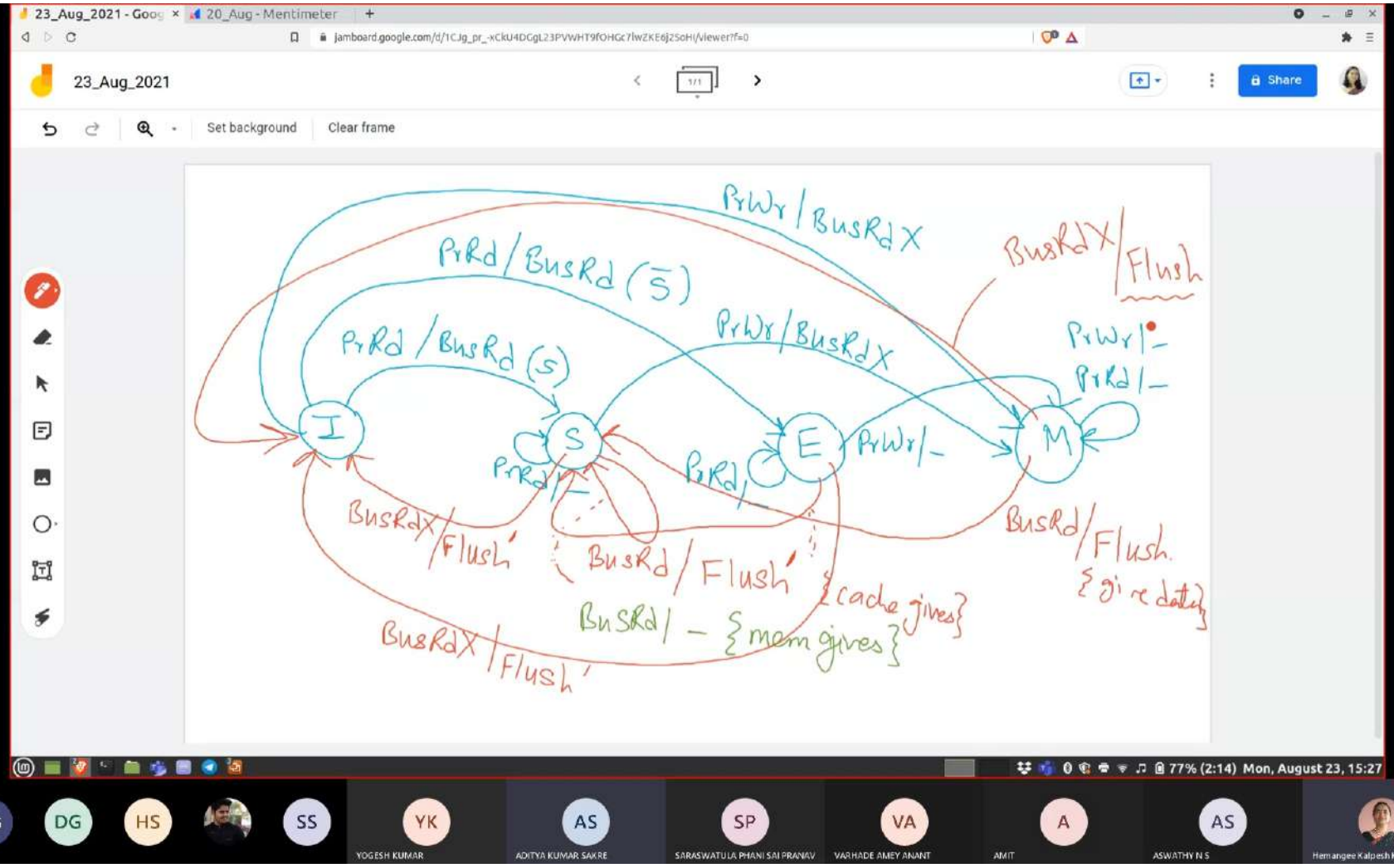
- Concern in MSI protocol:
  - If a sequential application is running on multiprocessor, i.e. no sharing of data
  - We still need 2 bus transactions in MSI for Rd / Wr
    - 1st a BusRd to load block from memory in 'S' state
    - 2nd a BusRdX (or BusUpgr) that converts 'S' -> 'M'
- Therefore add another state 'E'
  - Exclusive clean = Exclusive un-owned
  - Memory also has updated copy
- Exclusive = writable = No sharers, so write without informing others
- Modified = owned = written

4-state: MESI

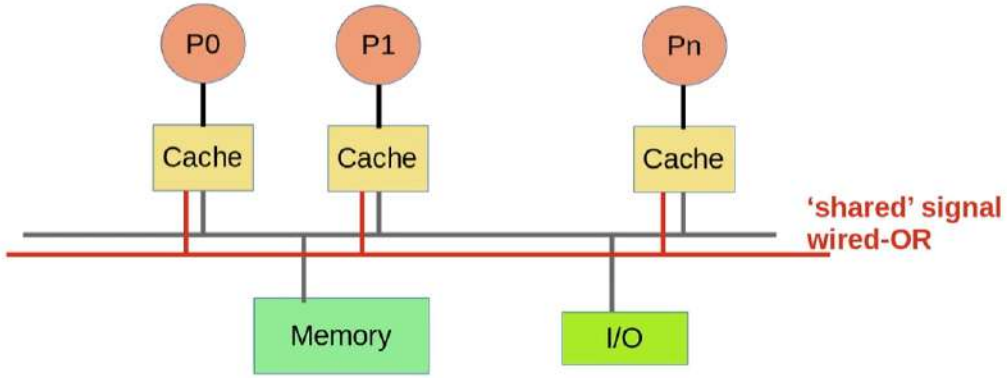
- States
  - Invalid
  - Exclusive: Only this cache has a copy and not modified
  - Shared: Two or more caches may have copy
  - Modified: Dirty
- FSM shown on another slide
- Variants of this used in Intel Pentium, PowerPC, MIPS 4400 of Silicon Graphics Challenge multiprocessor
- It was first published by researchers are UIUC and hence it is often called the Illiniois protocol

I -> E ??

- On PrRd we transition from ‘I’ to ‘E’ if no cache has a copy of the block
- How can you tell? => Need hardware support
- All cache controllers snoop on BusRd
- Assert additional signal: ‘shared’ signal if cache has the block. This is the variable ‘S’ and ‘S-bar’ in the FSM
- Wired-OR signal, therefore any cache making ‘shared’=1 we know that we should perform I->S
- And if ‘shared’=0 then we should perform I->E

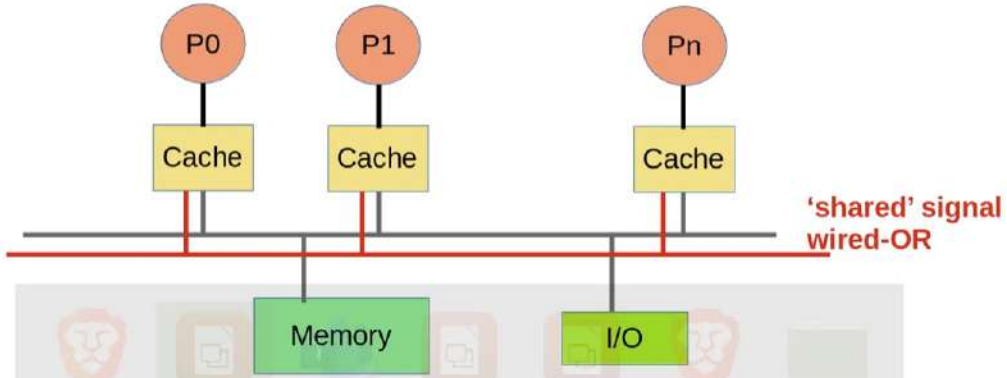

23_Aug_2021 - Goog
20_Aug - Mentimeter
jamboard.google.com/d/1CJg_pr_-xCkU4DGgL23PVWHT9fOHGc7lwZKE6j2SoH/viewer?f=0
23_Aug_2021
Share
Set background
Clear frame
PrWr / BusRdX
PrRd / BusRd (S̄)
PrRd / BusRd (S)
PrWr / BusRdX
BusRdX / Flush
PrWr / -
PrRd / -
I
S
E
M
PrRd / -
PrWr / -
PrRd / -
BusRdX / Flush'
BusRd / Flush'
{cache gives}
BusRd / - {mem gives}
BusRdX / Flush'
BusRd / Flush.
{give data}
77% (2:14) Mon, August 23, 15:27
+25
DG
HS
SS
YK
YOGESH KUMAR
AS
ADITYA KUMAR SAKRE
SP
SARASWATULA PHANI SAI PRANAV
VA
VARHADE AMEY ANANT
A
AMIT
AS
ASWATHY N S
Hemangee Kalpesh Kapoor

I -> E ??

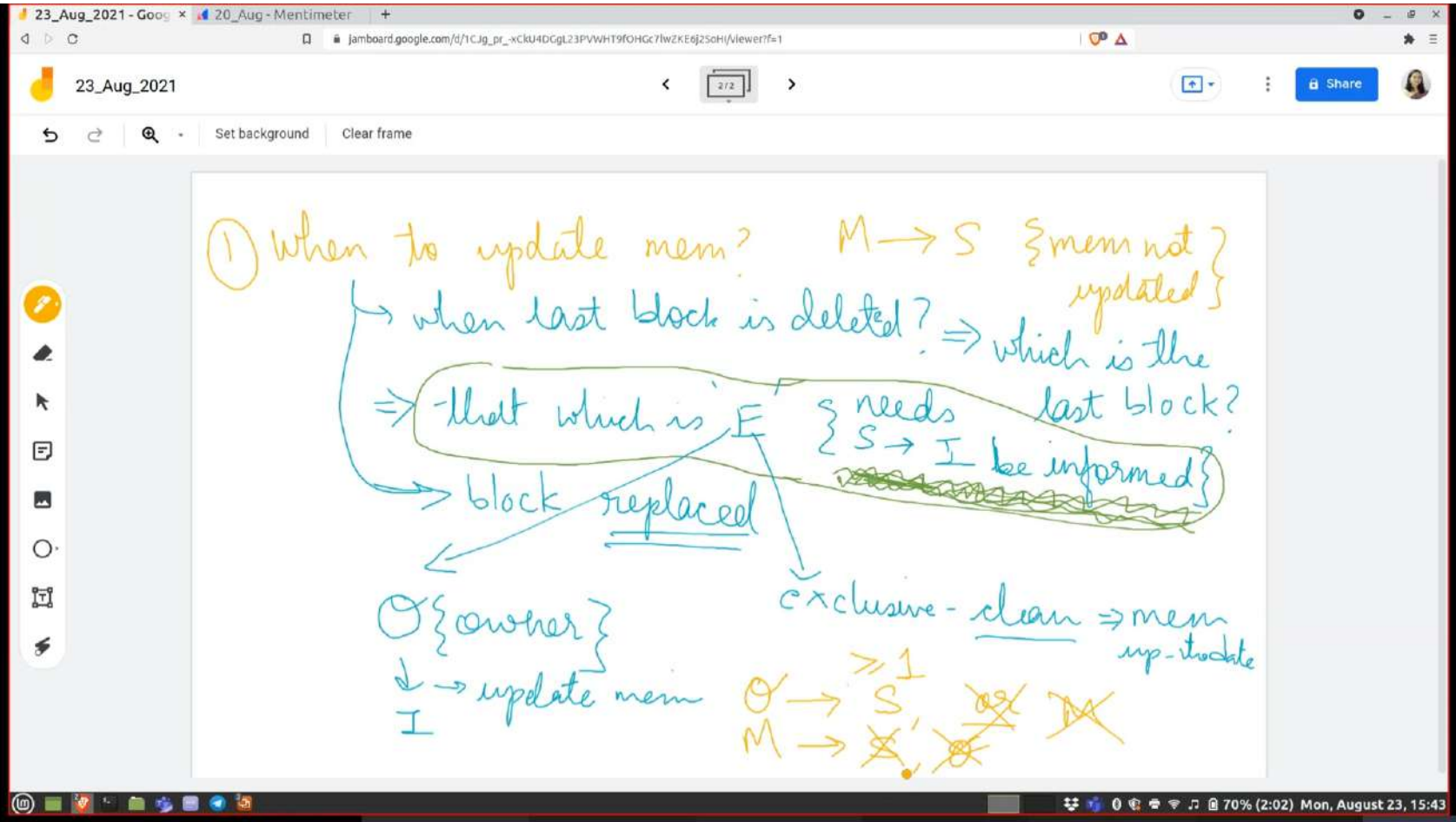


- BusRd(**S**) means shared line is asserted on BusRd transaction
- BusRd(**S-bar**) means 'S' is un-asserted
- **Flush'** : if cache-to-cache transfers: only one cache flushes data, giving it to requestor
- Satisfying Coherence: Arguments same as MSI
- **MOESI** protocol: Owned state = Exclusive but memory not valid
  - M->S : But one cache has updated copy
  - Memory not updated. Ownership transferred. Cache in state 'O' provides the data
- It is possible to have a block in state 'S' even when there are not other sharers
  - Why? Because, others removed their copies and deletion does not trigger transaction. This cache may not know it is the only one left with a copy

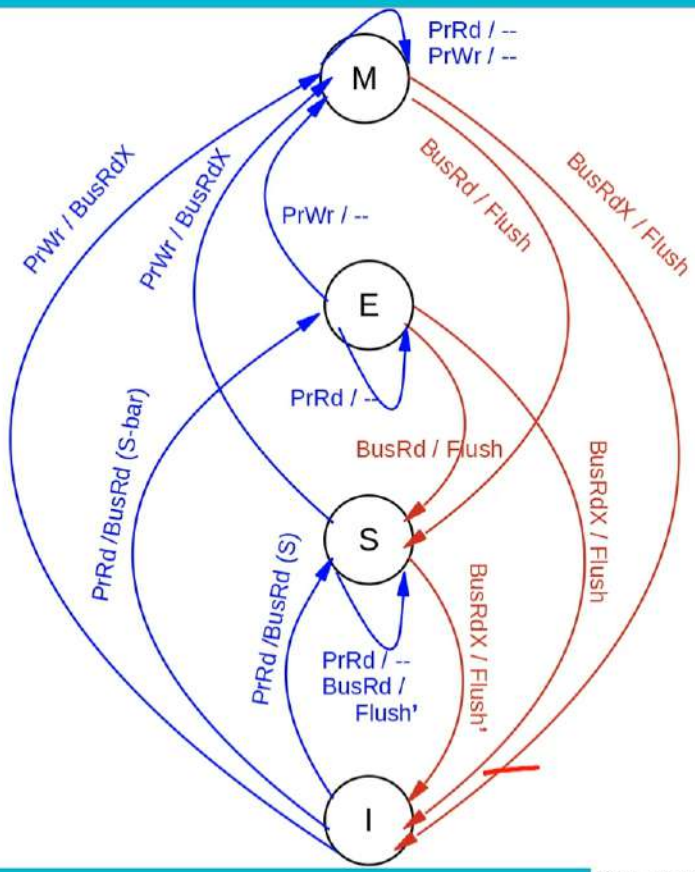
I -> E ??



- BusRd(**S**) means shared line is asserted on BusRd transaction
- BusRd(**S-bar**) means 'S' is un-asserted
- **Flush** : if cache-to-cache transfers: only one cache flushes data, giving it to requestor
- Satisfying Coherence: Arguments same as MSI
- **MOESI** protocol: Owned state = Exclusive but memory not valid
  - M->S : But one cache has updated copy
  - Memory not updated. Ownership transferred. Cache in state 'O' provides the data
- It is possible to have a block in state 'S' even when there are not other sharers
  - Why? Because, others removed their copies and deletion does not trigger transaction. This cache may not know it is the only one left with a copy


23_Aug_2021 - Goog
20_Aug - Mentimeter
jamboard.google.com/d/1CJg_pr_-xCkU4DGgL23PVWHT9fOHGc7lwZKE6j2SoH/viewer?f=1
23_Aug_2021
2/2
Share
Set background
Clear frame
1) When to update mem?
M → S {mem not updated}
→ when last block is deleted? ⇒ which is the last block?
⇒ that which is in E {needs S → I be informed}
→ block replaced
O {owner}
→ update mem
I
exclusive - clean ⇒ mem up-to-date
≥1
O → S
M →
70% (2:02) Mon, August 23, 15:43
+25
SP
HS
VA
SS
YK
YOGESH KUMAR
AS
ADITYA KUMAR SAKRE
TANVISH
MOHIT JAIN
A
AMIT
AS
ASWATHY N S
Hemangee Kalpesh Kapoor

4-state MESI write-back, Invalidate Protocol



Lower level protocol choices

- Who supplied data on miss when not in 'M' state?

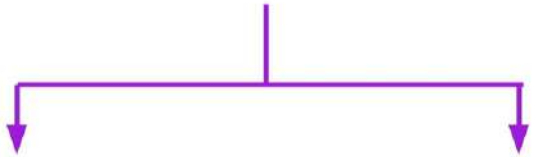
  => Memory? Or Cache?

- Original Illinois MESI: cache supplies data. Since cache is assumed faster than memory
- Not true in modern systems: as intervening in another cache is more expensive than getting from memory
- Cache-to-cache sharing (i.e. data transfer) adds complexity
  - How does memory know it should supply data and it must wait for all caches?
  - Need selection algorithm if multiple caches has valid data
  - Valuable for cache-coherent machines with distributed memory
  - May be cheaper to obtain from nearby cache than from distant memory. Especially when constructed out of network of SMP nodes (e.g. Stanford DASH)

Shared Memory Multiprocessors

Coherence protocols
Ex: Dragon

Snooping protocols



Write invalidate

Write update

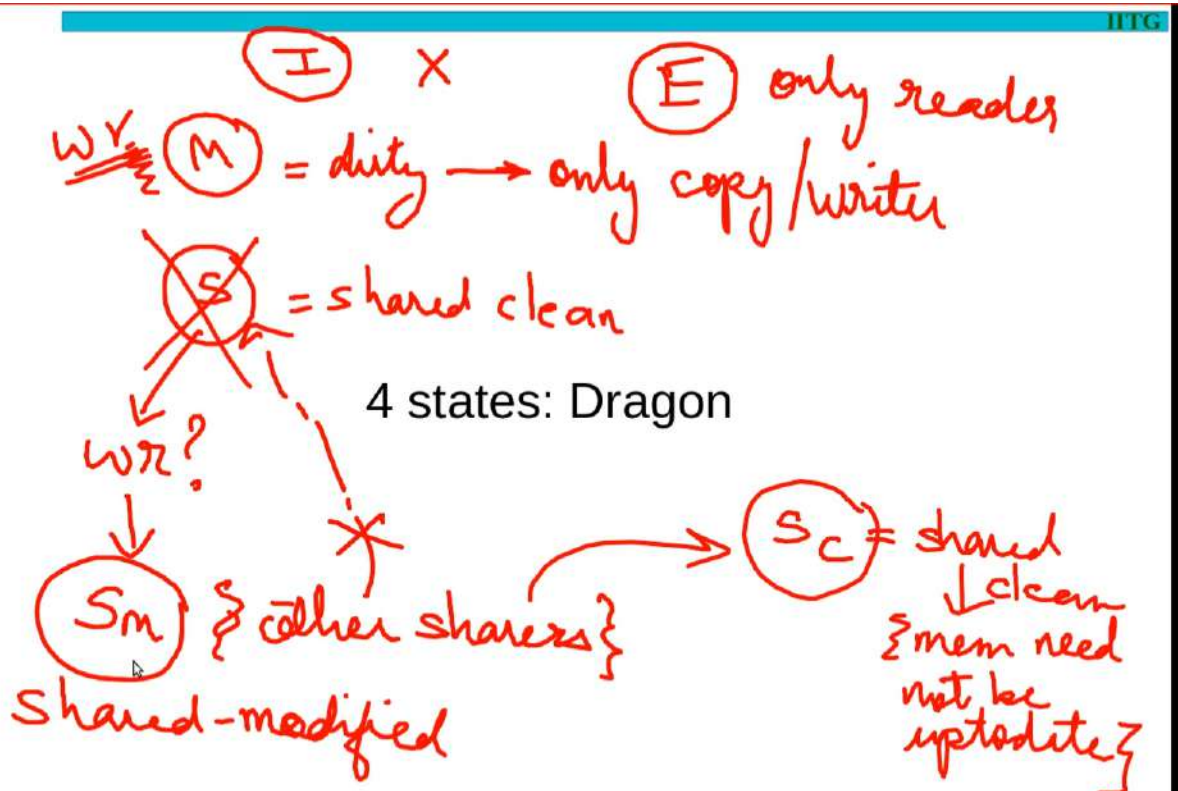
Write-update protocol

- When shared item is written, update all cached copies
- It must broadcast the write to all
- This uses considerable bandwidth, therefore most recent systems use write-invalidate protocols
- +ve avoids misses on later references
- -ve multiple useless updates

4 states: Dragon

4-state (Dragon) write-back, Update protocol

- E=Exclusive
  - Only one cache (this one) has copy and it is not modified, i.e. Memory is up-to-date
- M=Modified
  - Exclusive ownership; Modified dirty; Present in this cache only; Memory is stale
- Sc=Shared-Clean
  - Potentially two or more caches (including this one) have the block
  - Memory may/may-not be up-to-date
  - When in Sc, there could be block in other cache in Sc or Sm state. Therefore Memory may not be up-to-date
  - If no other Sm then Memory is up-to-date
- Sm=Shared-Modified



4-state (Dragon) write-back, Update protocol

- E=Exclusive
  - Only one cache (this one) has copy and it is not modified, i.e. Memory is up-to-date
- M=Modified
  - Exclusive ownership; Modified dirty; Present in this cache only; Memory is stale
- Sc=Shared-Clean
  - Potentially two or more caches (including this one) have the block
  - Memory may/may-not be up-to-date
  - When in Sc, there could be block in other cache in Sc or Sm state. Therefore Memory may not be up-to-date
  - If no other Sm then Memory is up-to-date
- Sm=Shared-Modified

4-state (Dragon) write-back, Update protocol

- E=Exclusive
- M=Modified
- Sc=Shared-Clean
- Sm=Shared-Modified
  - Potentially two or more caches (including this one) have the block
  - Memory is NOT up-to-date
  - Memory updated by this cache on replacement
  - Block in Sm state in only-one cache at a time
  - Possible that one Sm and some Sc states
  - Possible that no Sm and some Sc states

4-state (Dragon) write-back, Update protocol

- E=Exclusive
- M=Modified
- Sc=Shared-Clean
- Sm=Shared-Modified
- I = ?
  - There is no explicit 'I' Invalid state
  - Since it is update protocol, cache can keep copy until it gets replaced
  - If tag matches, copy gets updated
  - However, if block not in cache, then it is special invalid state
  - Initially load-cache (Read) then normal protocol