CpE 313: Microprocessor Systems Design
Fall 2004

Cache Coherence
Handout 20

December 2, 2004

Shoukat Ali

shoukat@umr.edu

UMR UNIVERSITY OF MISSOURI-ROLLA
The Name. The Degree. The Difference.

1

Dell PowerEdge 6600 Server – A Shared Memory System

list price for a 4-node system = $4,199

2

IBM eServer xSeries 440 – Another Shared Memory System

3

Sun Fire 12K Server – Yet Another Shared Memory System

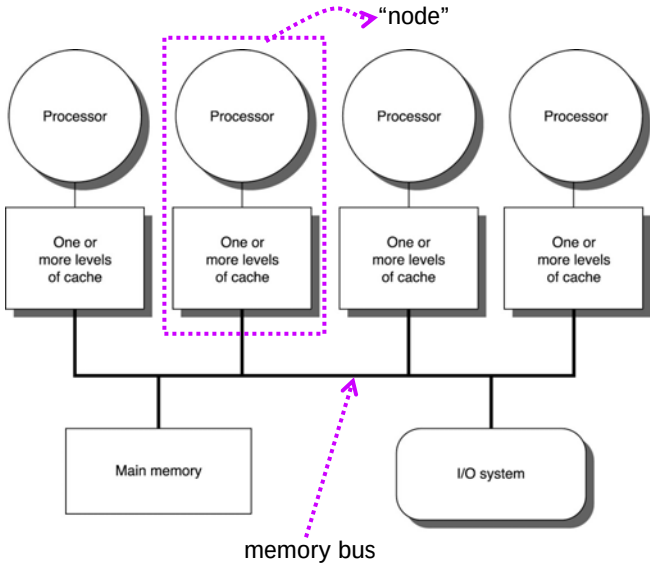
list price for:

a 4-node system
= $354,330

a 36-node system
= $1,382,530

4

What is a Shared Memory System?



- each processor has its own $
- all processors share
  - ONE memory
  - ONE memory bus

5

Cache Coherence Problem in SM Multiprocessors

- only issue that we can investigate in this course
- will set one problem (up to 5%) on it in the final exam

6

Cache Coherence Problem in SM Multiprocessors

- example of the problem for a 2-node SM multiprocessor
- processor 1 reads location A
- processor 2 reads location A
- processor 1 writes location A
- now, processor 2 has stale data in its cache

- solution: if proc 1 wants to write location A, it must
  - either update copy of A in the $ of proc 2
  - or invalidate copy of A in the $ of proc 2

- “or” part of the solution is very similar to the way Alpha 21264 handles the synonym problem

7

Cache Coherence Protocols

- informal definition of cache coherence
  - a method of making memory accesses appear to be consistent despite the presence of caches

- cache coherence (CC) protocol: a protocol for maintaining cache coherence

- two types of CC protocols depending on how writes are handled
  - invalidate copies in other caches
    - invalidation protocols
  - update copies in other caches
    - update protocols

8

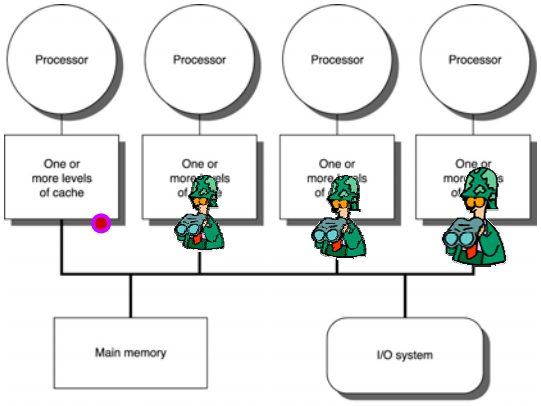
Invalidation-Based Cache Coherence

- basic idea
  - a block of data can be shared among multiple processors as long as no processor wants to write to it
  - if a processor wants to write a cache block, it must ensure that no other processor has a copy of this block
    - if yes, that copy must first be invalidated
- how does a processor know a data item is being shared?
  - have its cache snoop on other processors to see what blocks they are requesting from the memory

9

Snooping Caches

- a snoopy cache is like a snoopy next door neighbor who is always watching to see what you are doing
- a snoopy cache is always *monitoring the shared memory bus* to see what other processors are requesting from the memory



10

Illinois (or MESI) Protocol (Pentium, PowerPC, MIPS)

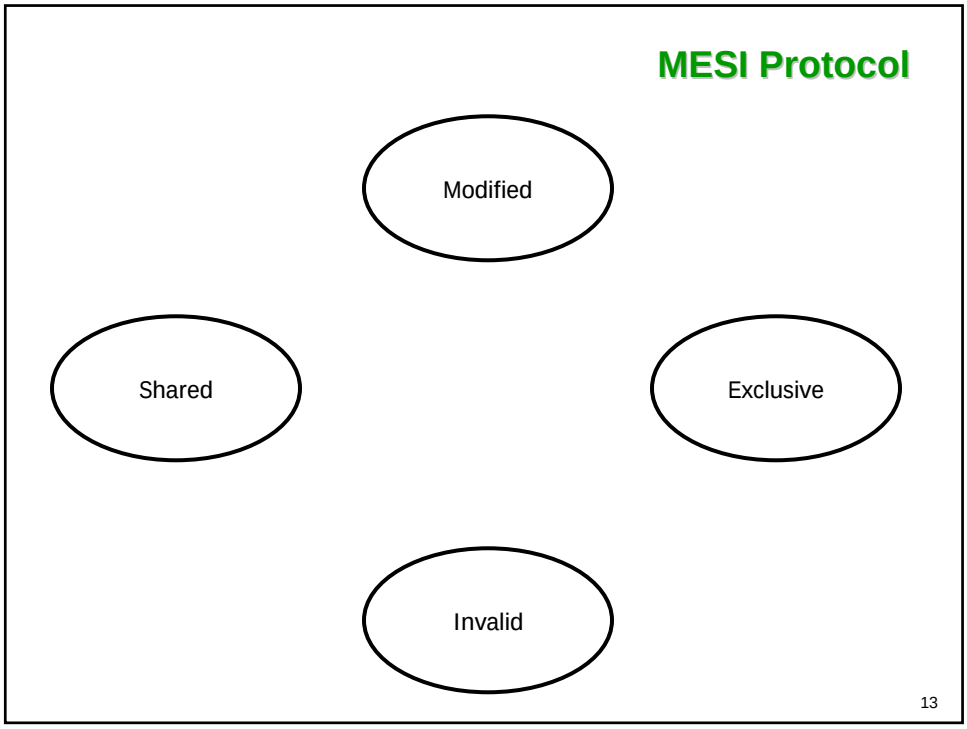
- in a processor's cache, each block can be in one of four *states*
  - modified (this processor has the only copy, the copy is "dirty," i.e., it has been modified since fetched, memory is "stale")
  - exclusive (this processor has the only copy, the copy is "clean," i.e., it has not been written since fetched, memory is up-to-date)
  - shared (this processor and some other processors have identical copies of this block, memory is up-to-date)
  - invalid (this processor has no copy of the block)
- a cache records the states of its blocks
  - similar to how uniprocessor caches track valid/invalid

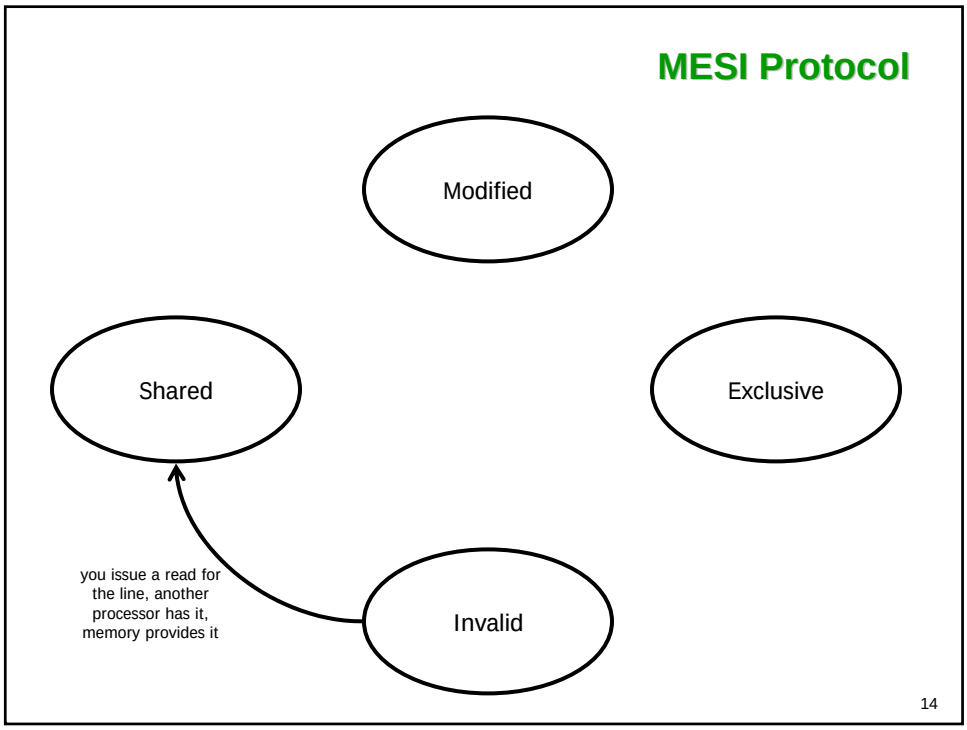
11

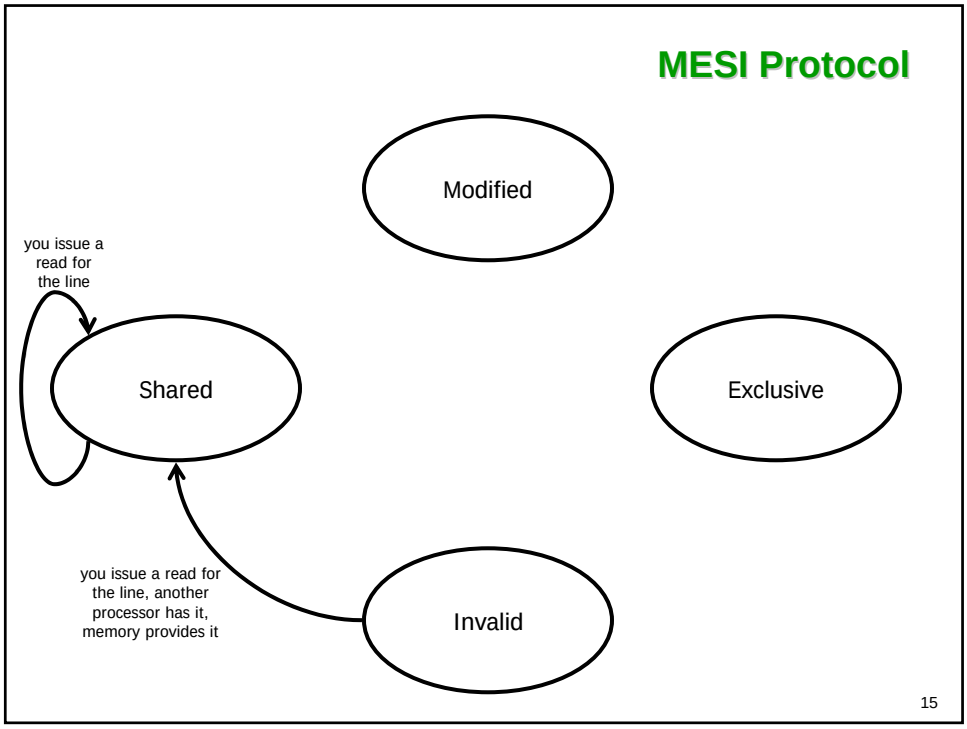
Interpreting the Illinois FSM

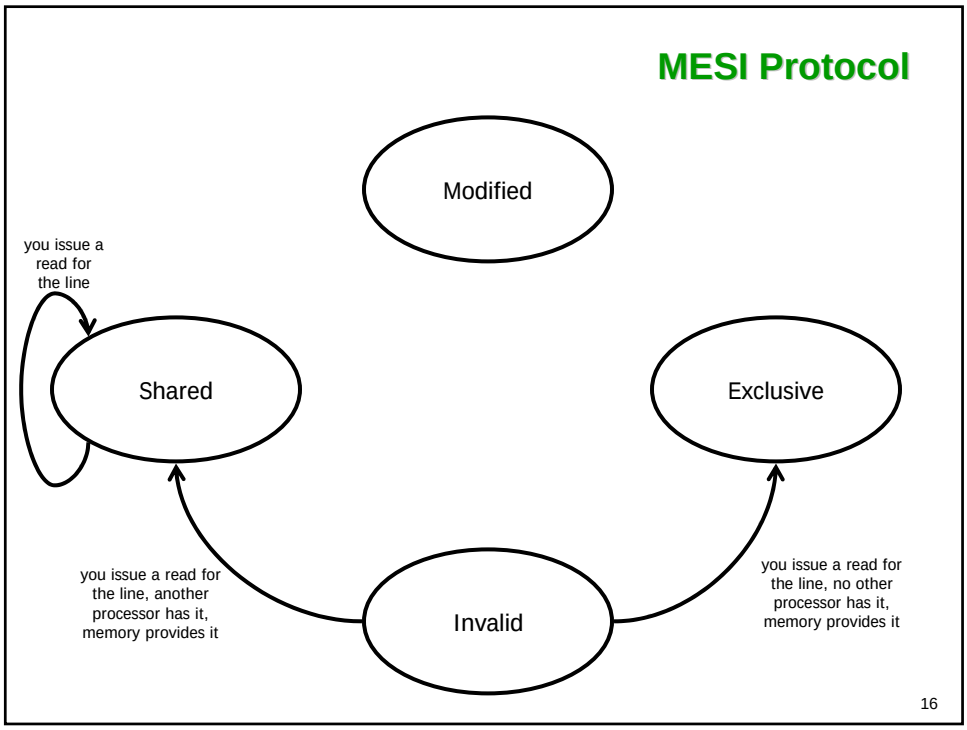
- FSM is drawn from the perspective of a given processor called "you"
- there are other processors as well
- "your" cache (as well as other caches) are monitoring the SM bus
  - your cache will either see a memory request from you or from another processor
  - your cache will compare the requested address with its tags
    - if address is found, it will take an action

12

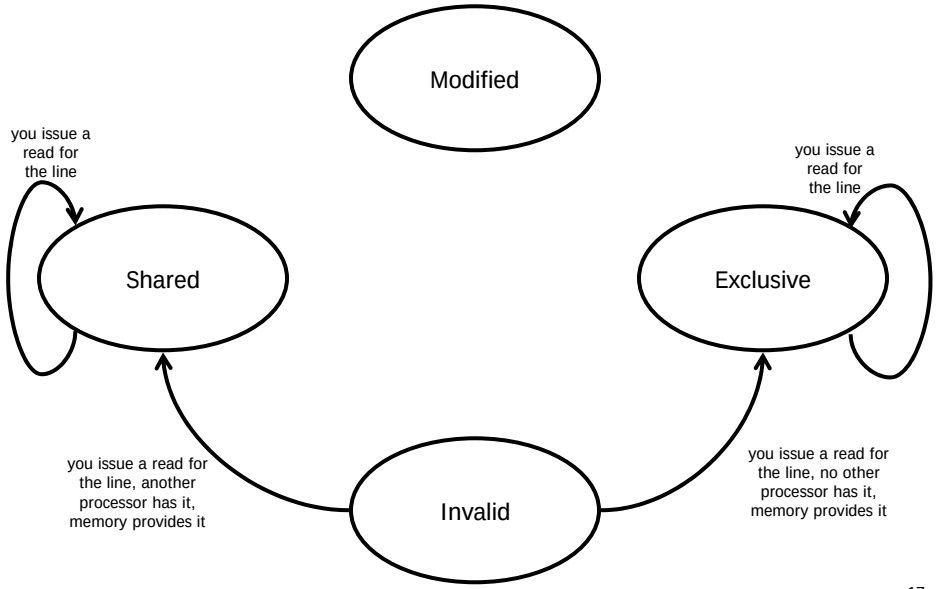

MESI Protocol
Modified
Shared
Exclusive
Invalid
13


MESI Protocol
Modified
Shared
Exclusive
you issue a read for the line, another processor has it, memory provides it
Invalid
14


MESI Protocol
Modified
you issue a read for the line
Shared
Exclusive
you issue a read for the line, another processor has it, memory provides it
Invalid
15


MESI Protocol
Modified
you issue a read for the line
Shared
Exclusive
you issue a read for the line, another processor has it, memory provides it
Invalid
you issue a read for the line, no other processor has it, memory provides it
16

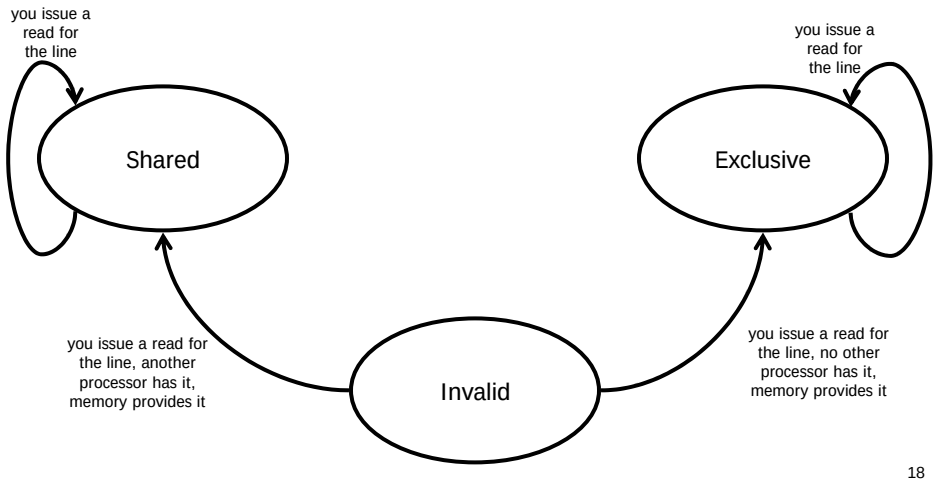
MESI Protocol



17

MESI Protocol

Note: How do you know if another processor has the block you need?
Answer: The bus is physically augmented with another wire called “shared signal.” For address phase of EVERY bus transaction, ALL caches check their tags to see if they have the requested data. If yes, “shared” line is raised.



18

Some Notation

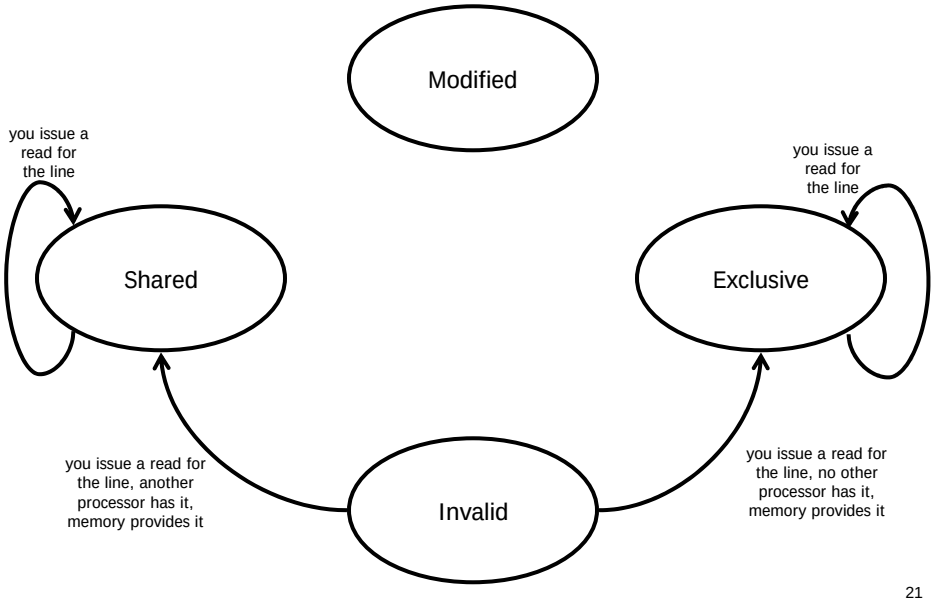
- A/B notation
  - if the cache controller observes the event A from the processor or the bus, then, in addition to state change, it generates the bus transaction or action B
  - A/-- means no transaction is generated on bus in response to event A

19

20

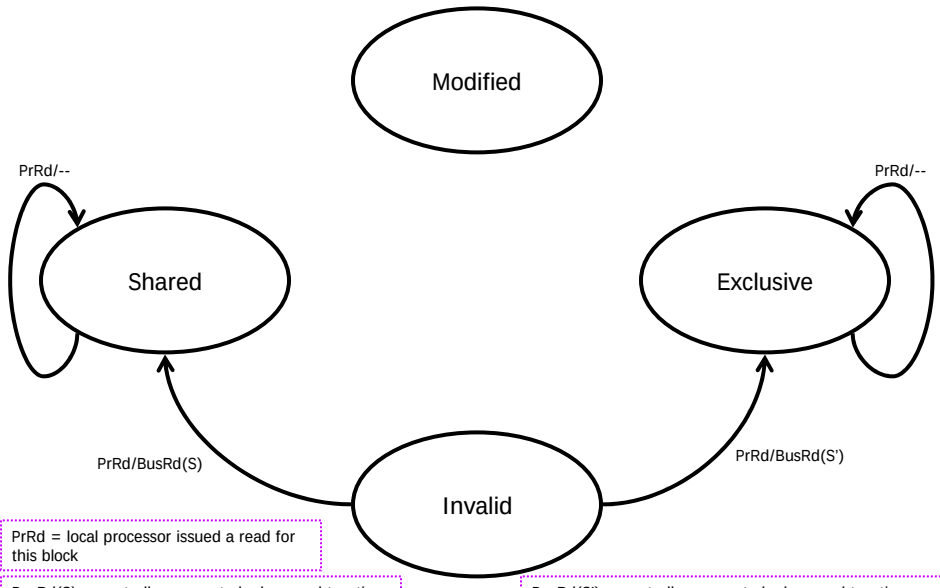
Repeated Slide

MESI Protocol



21

MESI Protocol

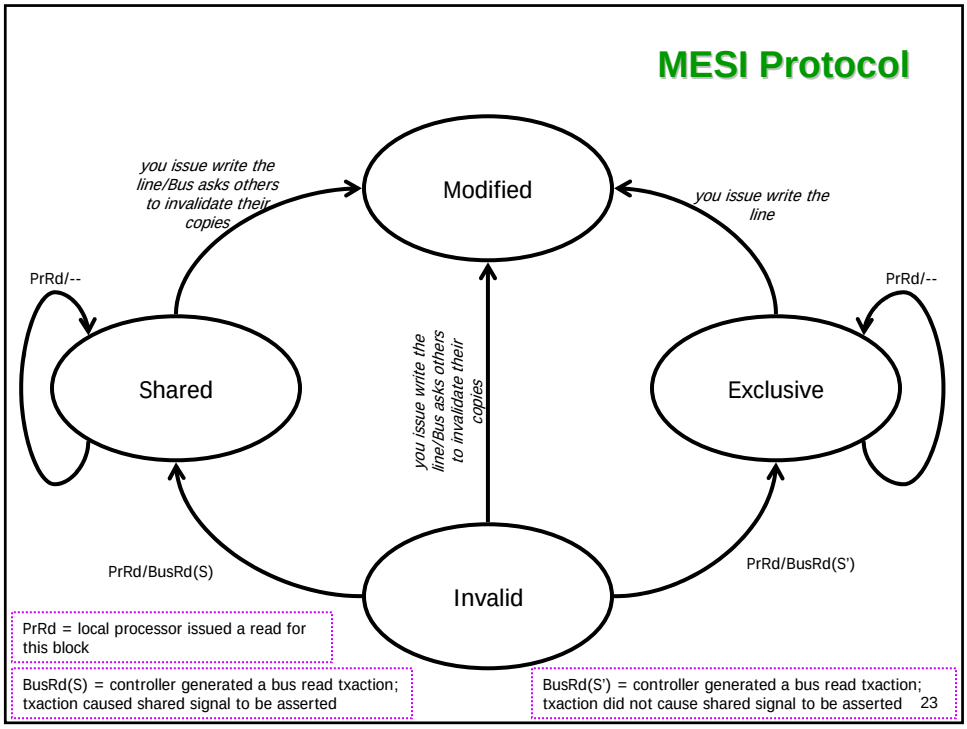


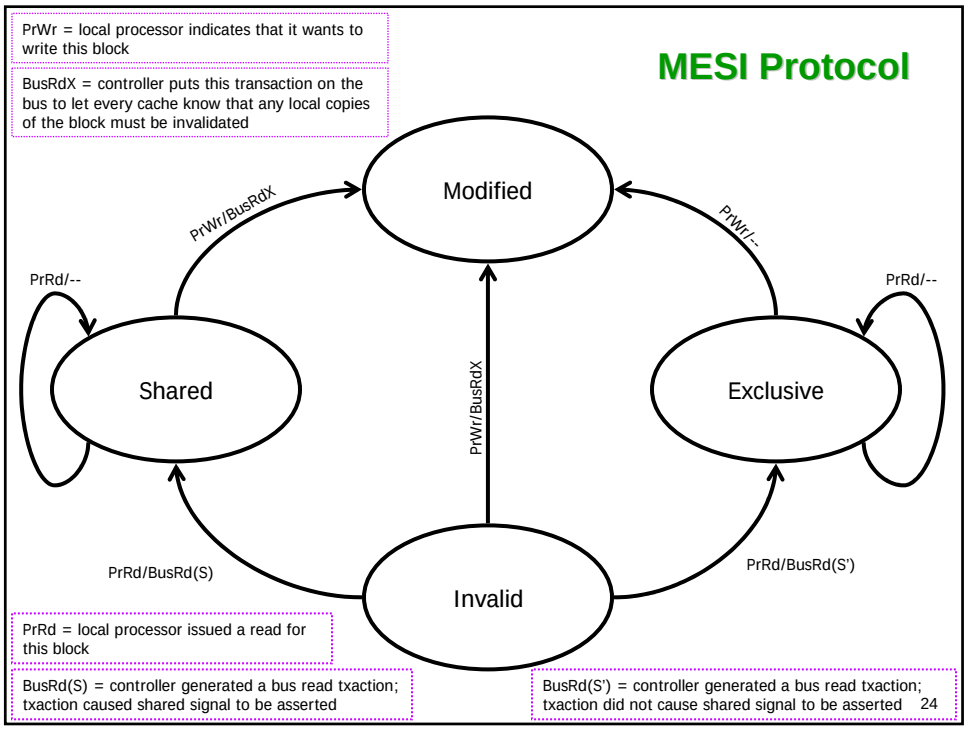
PrRd = local processor issued a read for this block

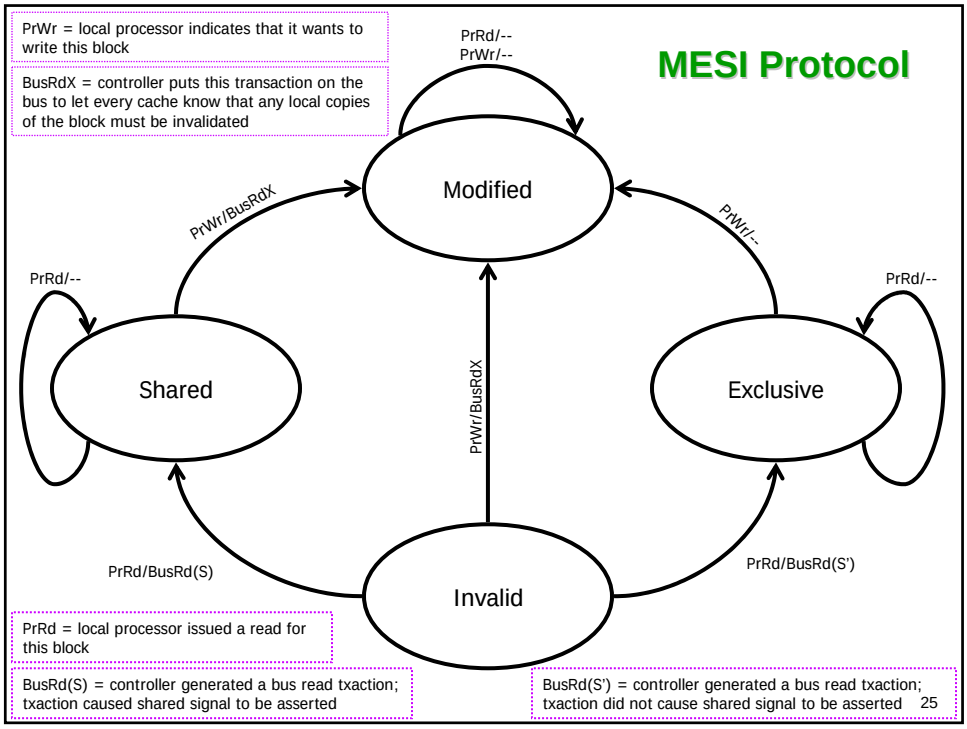
BusRd(S) = controller generated a bus read txaction; txaction caused shared signal to be asserted

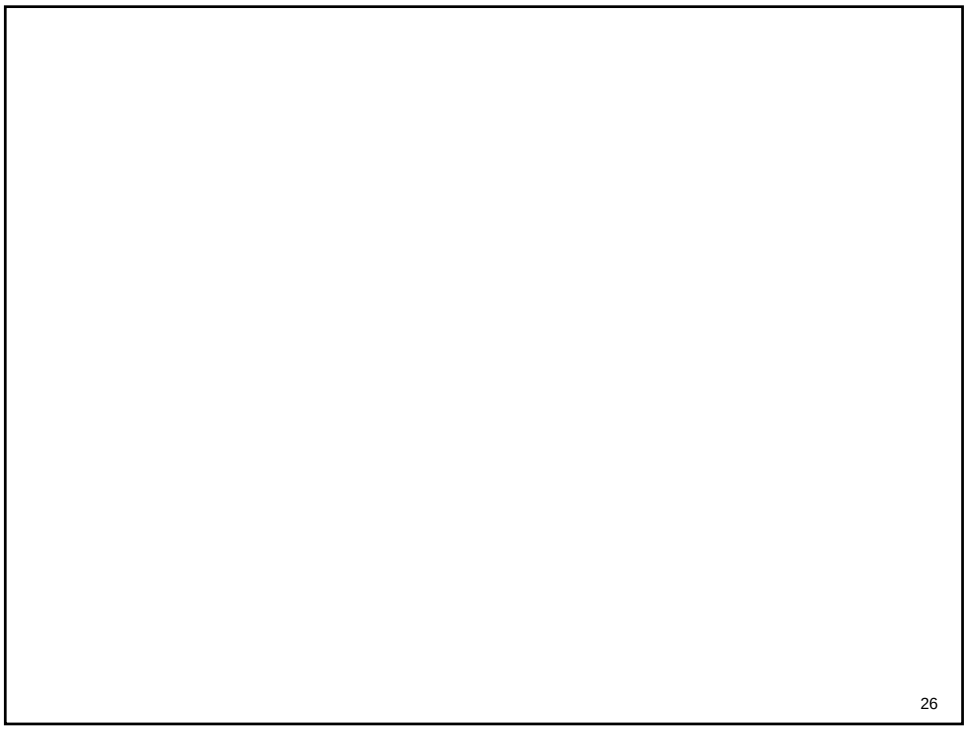
BusRd(S') = controller generated a bus read txaction; txaction did not cause shared signal to be asserted

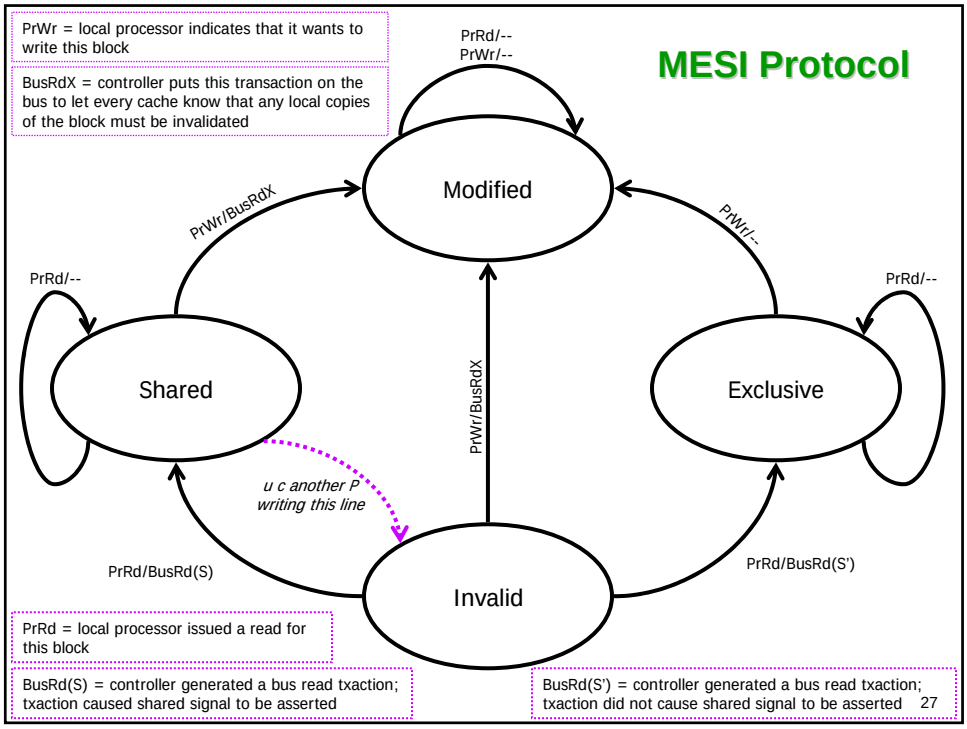
22

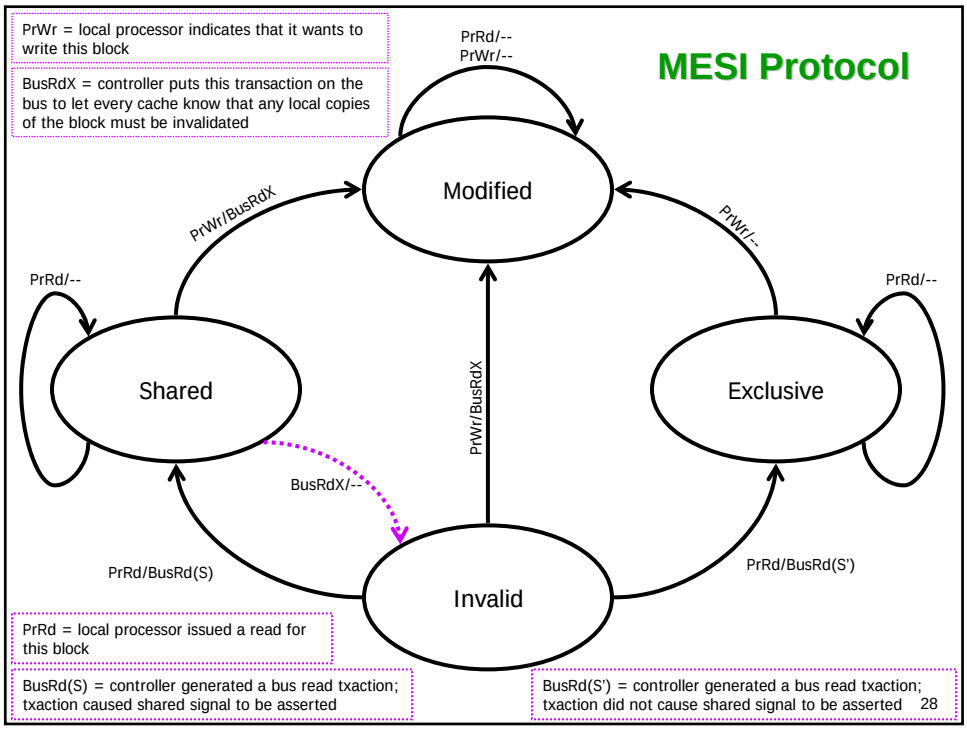

MESI Protocol
Modified
you issue write the line/Bus asks others to invalidate their copies
you issue write the line
PrRd/--
PrRd/--
Shared
you issue write the line/Bus asks others to invalidate their copies
Exclusive
PrRd/BusRd(S)
PrRd/BusRd(S')
Invalid
PrRd = local processor issued a read for this block
BusRd(S) = controller generated a bus read txaction; txaction caused shared signal to be asserted
BusRd(S') = controller generated a bus read txaction; txaction did not cause shared signal to be asserted
23

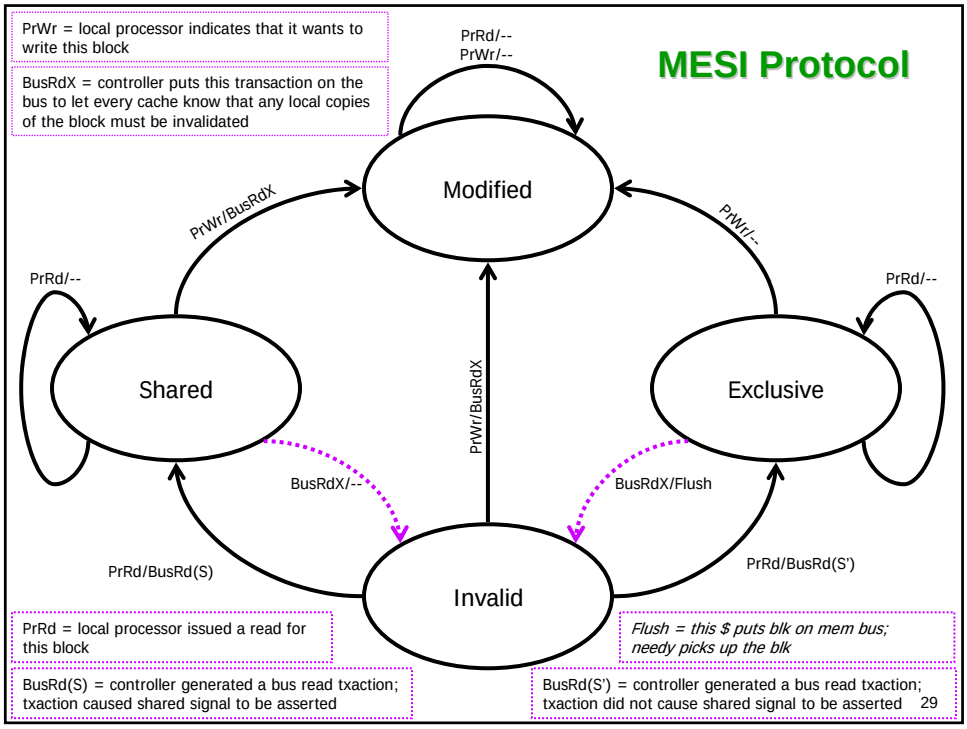

PrWr = local processor indicates that it wants to write this block
BusRdX = controller puts this transaction on the bus to let every cache know that any local copies of the block must be invalidated
MESI Protocol
Modified
PrWr/BusRdX
PrWr/--
PrRd/--
PrRd/--
Shared
PrWr/BusRdX
Exclusive
PrRd/BusRd(S)
PrRd/BusRd(S')
Invalid
PrRd = local processor issued a read for this block
BusRd(S) = controller generated a bus read txaction; txaction caused shared signal to be asserted
BusRd(S') = controller generated a bus read txaction; txaction did not cause shared signal to be asserted
24

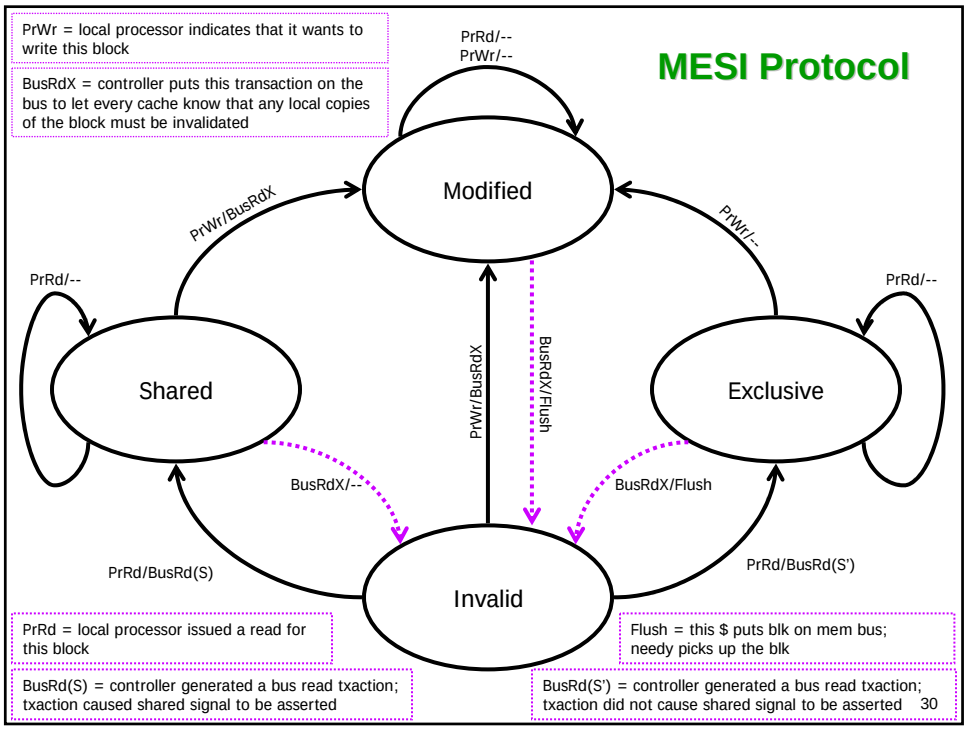

PrWr = local processor indicates that it wants to write this block
BusRdX = controller puts this transaction on the bus to let every cache know that any local copies of the block must be invalidated
MESI Protocol
PrRd/--
PrWr/--
Modified
PrWr/BusRdX
PrWr/--
PrRd/--
PrRd/--
Shared
PrWr/BusRdX
Exclusive
PrRd/BusRd(S)
Invalid
PrRd/BusRd(S')
PrRd = local processor issued a read for this block
BusRd(S) = controller generated a bus read txaction; txaction caused shared signal to be asserted
BusRd(S') = controller generated a bus read txaction; txaction did not cause shared signal to be asserted
25

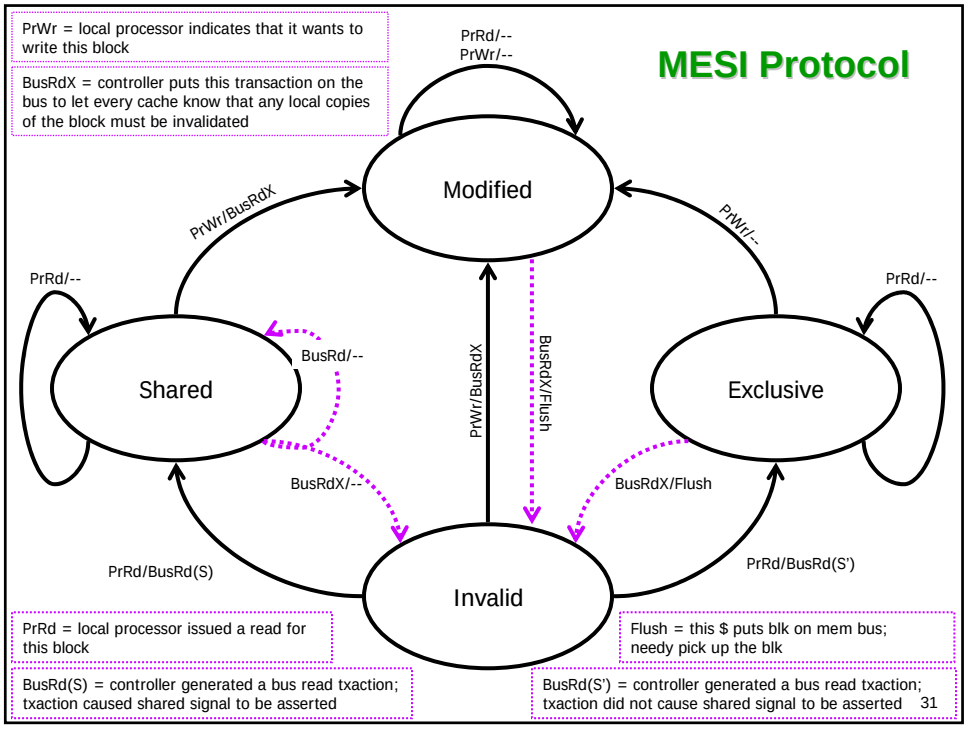

26

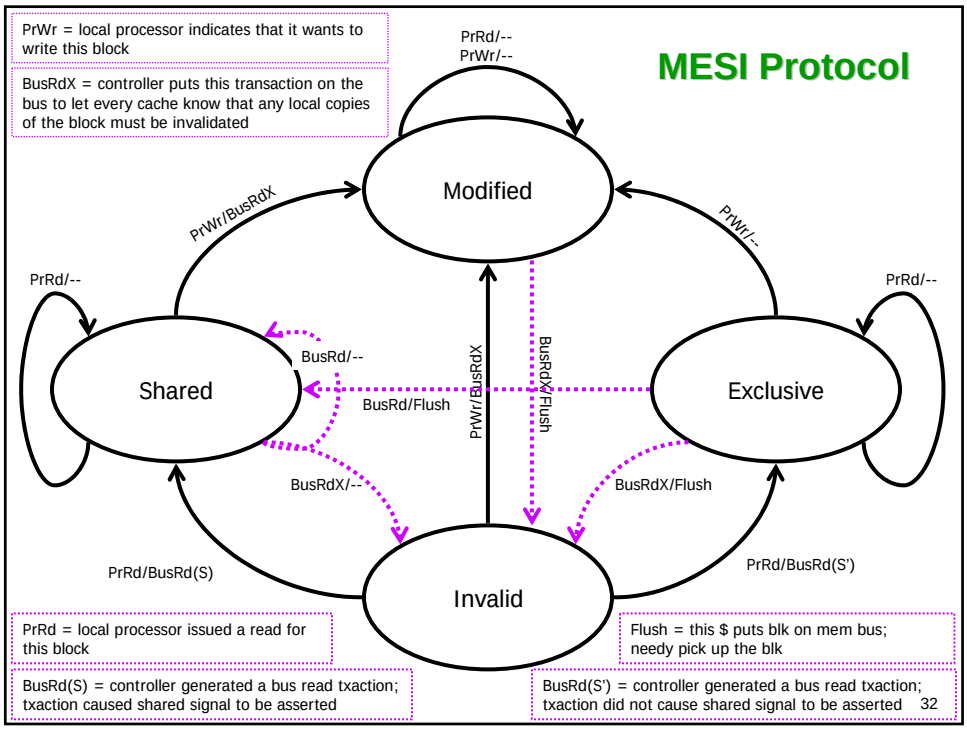

PrWr = local processor indicates that it wants to write this block
BusRdX = controller puts this transaction on the bus to let every cache know that any local copies of the block must be invalidated
PrRd/--
PrWr/--
MESI Protocol
Modified
PrWr/BusRdX
PrWr/--
PrRd/--
PrRd/--
Shared
Exclusive
PrWr/BusRdX
u c another P writing this line
PrRd/BusRd(S)
PrRd/BusRd(S')
Invalid
PrRd = local processor issued a read for this block
BusRd(S) = controller generated a bus read txaction; txaction caused shared signal to be asserted
BusRd(S') = controller generated a bus read txaction; txaction did not cause shared signal to be asserted
27


PrWr = local processor indicates that it wants to write this block
BusRdX = controller puts this transaction on the bus to let every cache know that any local copies of the block must be invalidated
PrRd/--
PrWr/--
MESI Protocol
Modified
PrWr/BusRdX
PrWr/--
PrRd/--
PrRd/--
Shared
Exclusive
PrWr/BusRdX
BusRdX/--
PrRd/BusRd(S)
PrRd/BusRd(S')
Invalid
PrRd = local processor issued a read for this block
BusRd(S) = controller generated a bus read txaction; txaction caused shared signal to be asserted
BusRd(S') = controller generated a bus read txaction; txaction did not cause shared signal to be asserted
28


PrWr = local processor indicates that it wants to write this block
BusRdX = controller puts this transaction on the bus to let every cache know that any local copies of the block must be invalidated
MESI Protocol
PrRd/--
PrWr/--
Modified
PrWr/BusRdX
PrWr/--
PrRd/--
PrRd/--
Shared
Exclusive
PrWr/BusRdX
BusRdX/--
BusRdX/Flush
PrRd/BusRd(S)
PrRd/BusRd(S')
Invalid
PrRd = local processor issued a read for this block
Flush = this $ puts blk on mem bus; needy picks up the blk
BusRd(S) = controller generated a bus read txaction; txaction caused shared signal to be asserted
BusRd(S') = controller generated a bus read txaction; txaction did not cause shared signal to be asserted
29


PrWr = local processor indicates that it wants to write this block
BusRdX = controller puts this transaction on the bus to let every cache know that any local copies of the block must be invalidated
MESI Protocol
PrRd/--
PrWr/--
Modified
PrWr/BusRdX
PrWr/--
PrRd/--
PrRd/--
Shared
Exclusive
PrWr/BusRdX
BusRdX/Flush
BusRdX/--
BusRdX/Flush
PrRd/BusRd(S)
PrRd/BusRd(S')
Invalid
PrRd = local processor issued a read for this block
Flush = this $ puts blk on mem bus; needy picks up the blk
BusRd(S) = controller generated a bus read txaction; txaction caused shared signal to be asserted
BusRd(S') = controller generated a bus read txaction; txaction did not cause shared signal to be asserted
30


PrWr = local processor indicates that it wants to write this block
BusRdX = controller puts this transaction on the bus to let every cache know that any local copies of the block must be invalidated
MESI Protocol
PrRd/--
PrWr/--
Modified
PrWr/BusRdX
PrWr/--
PrRd/--
PrRd/--
Shared
BusRd/--
PrWr/BusRdX
BusRdX/Flush
Exclusive
BusRdX/--
BusRdX/Flush
PrRd/BusRd(S)
PrRd/BusRd(S')
Invalid
PrRd = local processor issued a read for this block
Flush = this $ puts blk on mem bus; needy pick up the blk
BusRd(S) = controller generated a bus read txaction; txaction caused shared signal to be asserted
BusRd(S') = controller generated a bus read txaction; txaction did not cause shared signal to be asserted
31

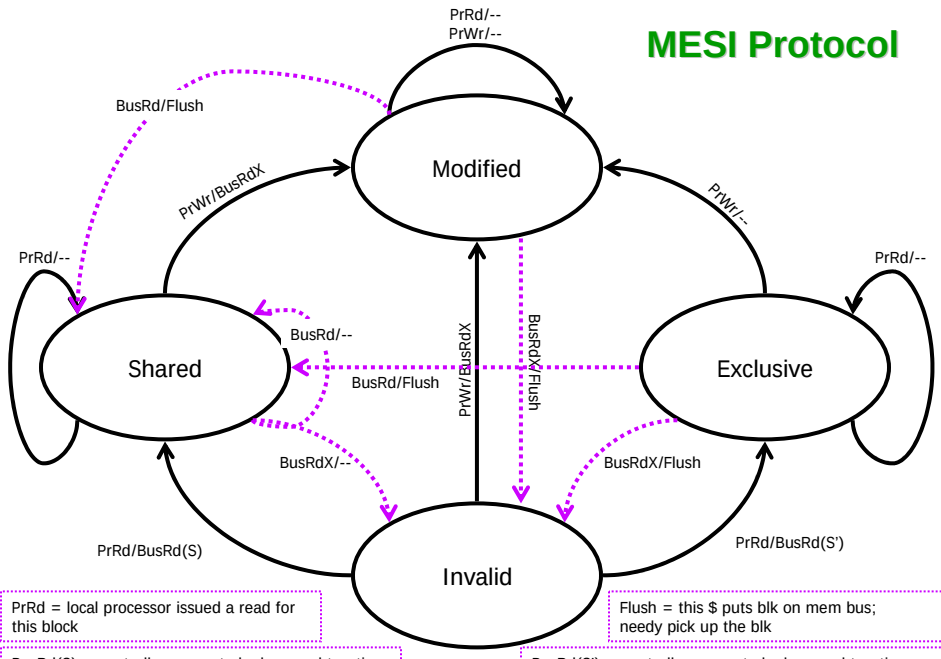

PrWr = local processor indicates that it wants to write this block
BusRdX = controller puts this transaction on the bus to let every cache know that any local copies of the block must be invalidated
MESI Protocol
PrRd/--
PrWr/--
Modified
PrWr/BusRdX
PrWr/--
PrRd/--
PrRd/--
Shared
BusRd/--
BusRd/Flush
PrWr/BusRdX
BusRdX/Flush
Exclusive
BusRdX/--
BusRdX/Flush
PrRd/BusRd(S)
PrRd/BusRd(S')
Invalid
PrRd = local processor issued a read for this block
Flush = this $ puts blk on mem bus; needy pick up the blk
BusRd(S) = controller generated a bus read txaction; txaction caused shared signal to be asserted
BusRd(S') = controller generated a bus read txaction; txaction did not cause shared signal to be asserted
32

Alas!

- there is no line from shared to exclusive

33

MESI Protocol



34

Summary of MESI Protocol

- each cache watches all of the transactions on the memory bus
- if another processor requests a copy of a line in your cache, then you handle the request by sending the line over the bus and adjusting the state in your cache appropriately
- owner provides data
  - owner = cache with blk in M or E state
  - otherwise, owner = memory
- if your copy is S, E, or M, it becomes invalid if some other processor wants to write the data
- if your copy is S, E, or M, it becomes shared if some other processor wants to read the data

35