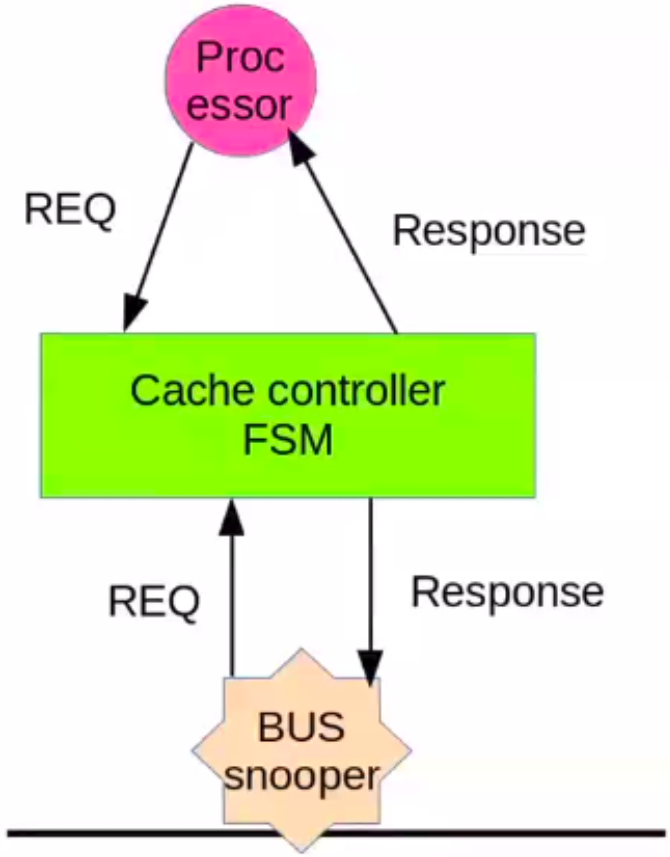
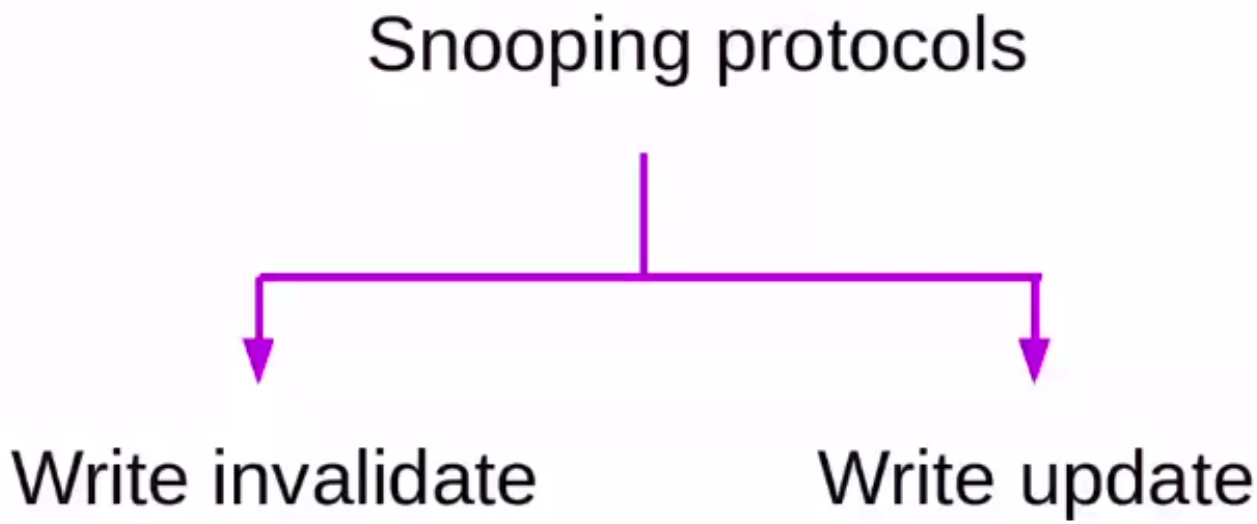
Protocol Implementation/Modelling

- In multiprocessor, each block has a state in each cache and the states change according to the state transition diagram
- If there are ‘p’ caches, then the block state is a vector of size=p
- Cache state is manipulated by a set of ‘p’ distributed finite state machines, implemented by the cache controllers
- Controller FSMs will receive inputs from processor or bus snooper, and they act accordingly



Snooping protocol=distributed FSM

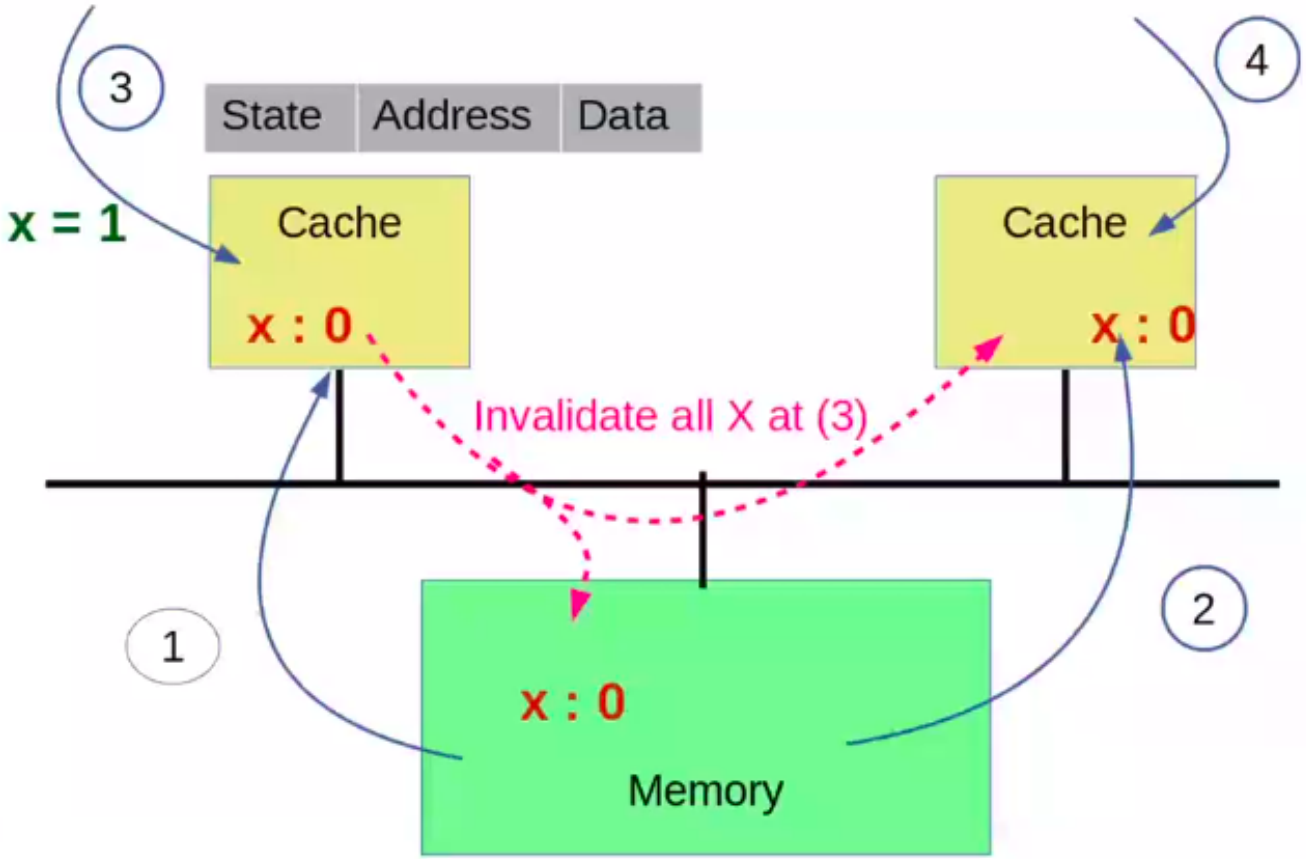
- Thus the snooping protocol is a distributed algorithm represented by a collection of cooperating FSMs
- All these FSMs are cooridnated by bus transactions
- It is specified by the following components
  - Set of **states** of memory blocks in local caches
  - State **transition diagram**, which takes input from processor or bus and produces the next state
  - The **actions** associated with each state transition that tell what actions to do on the bus, the cache and the processor


Snooping protocols
Write invalidate
Write update

Write-invalidate protocol

- Processor has exclusive access to data item before it writes to it
- i.e. a write by one processor, invalidates all other copies of that data
- -ve: readers will incur a miss
- Example= <next slide>
- +ve multiple writes (to same location) can be done by the processor without generating additional traffic
- +ve also, clear out copies that will never be used again

Example: Wr-inv protocol



(1) A reads X=0

(2) B read X=0

(3) A writes X=1;
Inv all copies,
memory updated

(4) B reads X
=> results in
cache miss

Cache containts
<state, addr, data>

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

Latest Copy search?

- The latest copy depends on type of cache: Write-through vs write-back
- Write-back
  - Search is hard, as latest copy may be in some other cache
  - -ve: this is a slow retrieval of data compared to memory read
  - +ve: less memory bandwidth needed in normal operation. Therefore can support large number of multiprocessors => more commonly used

Latest Copy search?

- The latest copy depends on type of cache: Write-through vs write-back
- Write-through
  - Simple protocol
  - Each write is observable
  - Only 1 write goes on the bus

    => only 1 write can take place at any time in any processor
  - Some complications in presence of write-buffers
  - Uses lot of bandwidth

Ex: write-thru cache

- 200 Mhz dual issue, CPI=1, 15% stores of 8 bytes

  How many processors will 1 GB/s bus support before saturation?

- ANS=

  Single processor will generate how many writes? Stores?

  =0.15 stores/instr * 1 instr/cycle * 200 * 10^6 cycle/sec

  = 30 * 10^6 stores/sec

  Each store = 8 bytes

  Per processor 8 * 30 * 10^6 bytes/sec = 240 MB/sec

  1 GB/s / 240 MB = 4 processors will results in saturation of bus

  Max support up to 4 processors

Design space for Snooping Protocols

- In a write back cache, the processor holds the block while modifying it

  => called modified or dirty-block

  - Such a block is also said to be in **exclusive** mode
  - The cache is called **owner** of the block, as it must **supply the data** upon request for that block

- A cache has exclusive copy of the block if it is the only cache with a valid copy of the block
  - The memory may/may-not have valid copy

  => in this state the cache can modify data without informing others

- If block is not exclusive, then need to inform others by generating bus transaction. This transaction is called a **write-miss**
- On a write-miss, in invalidation protocol, a read-exclusive transaction is generated to **tell other** caches of the impending write and to acquire the block in exclusive ownership