

INSTRUCTION READ or WRITE to MEMORY

↓ cache64::b_transport()

this method can be called by multiple threads, e.g. when there is one cache per n cores or low-level caches.



current SYSTEMC thread id (tid)

pushed to Requests data structure

only progress if tid top of Requests

AND Cache is not busy with some other Request (busyFlag)

cache geometry (struct cache_geom):

- bytes - capacity in bytes
- ways - associativity
- linesize - size of cache line in bytes

metrics that are derived for the caches:

- $\text{dmapping} = \text{bytes} / (\text{Ways} * \text{linesize})$

Number of lines in one way.

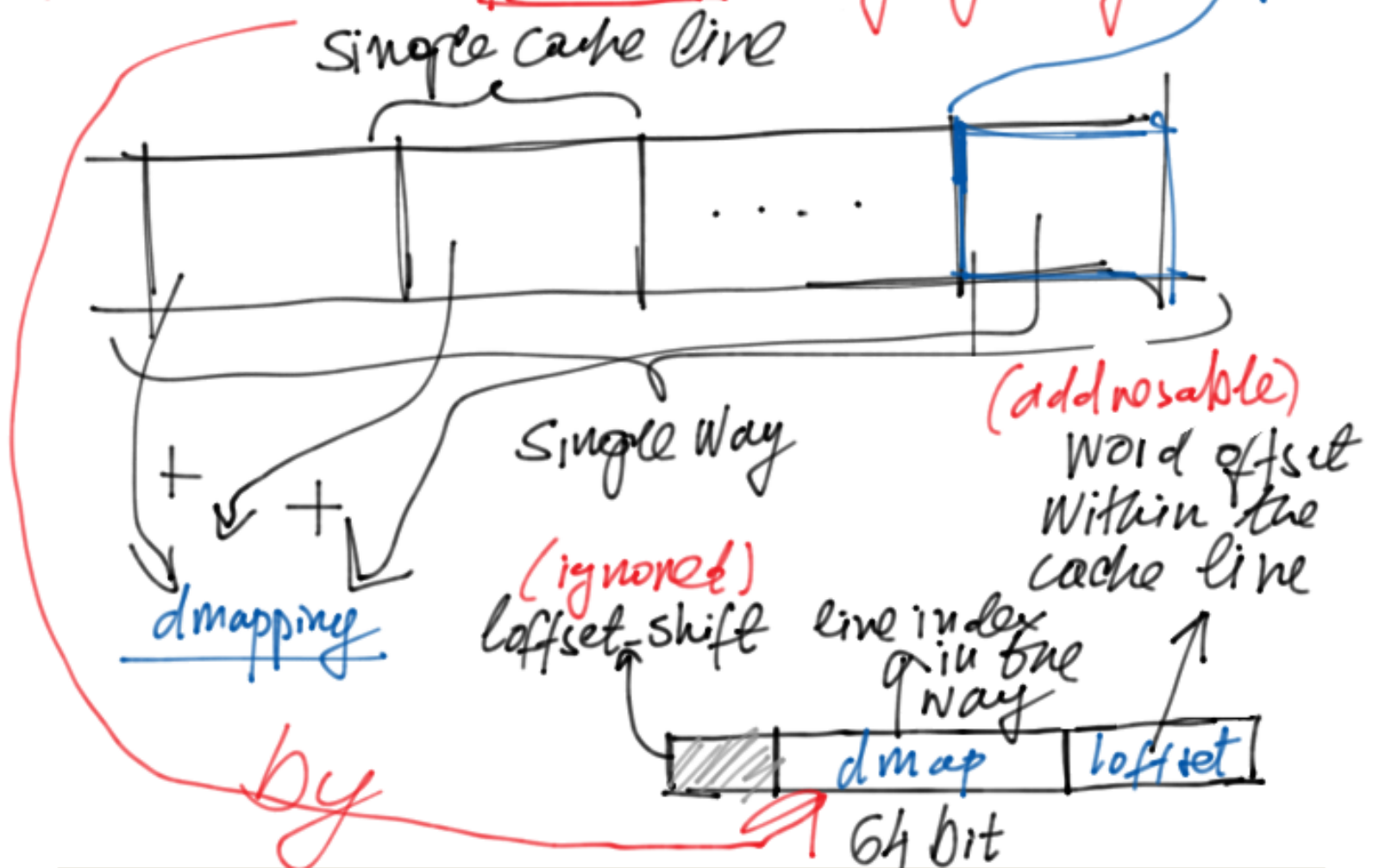
- $\text{word64s-per-line} = \text{linesize} / 8$

Words per cache line (1 word = 8 bytes)


- $\text{dmap-shift} = 3 + \log_2(\text{word64s-per-line})$

- $\text{loffset-shift} = 3$.

data is stored in indexed array by way. line-addr



if it is **LOAD-LINKED STORE-CONDITIONAL** access to memory than we:

① **invalidate** that line in the write buffer and the cache  **is this step necessary??**

② Go directly to the secondary storage.

Two types of the access to memory

TLM_READ_COMMAND
check if cache line is present in the write buffer
(yes) check if required word is present in the buffer. if yes serve from the buffer, no clear the write buffer and (no) call **lookup()** method to find it in the cache.

TLM_WRITE_COMMAND
Two different write policies are supported:
① **WRITE_THROUGH** calls the method **write-through()**
② **WRITE_BACK** calls the method **write-back()**.

Implementation of write-buffer class

Fields: m-addr, d-map, wbl, evict-addr, parent,
linesize, secondary-width-bytes

sc_mutex (lock)
points to the cache 64

Data is stored in smallram and information whether data is dirty or not in boolean Dirty array (set to size of MAX.LINE)

should we consider setting this dynamic?

for the cold write init() Method is called on the Buffer
data is not present in the caches

- ① take a lock wbl
- ② set m-addr and dmap and all entries to dirty
- ③ assertion is that m-addr == -1 (i.e. empty)

Hit() - check if passed address is equal to m-addr.

Word-present() - check if bytes in the passed offset are present

clean() →

- ① take a lock on wbl
- ② Store m-addr to evict-addr and release wbl lock
- ③ take parent's ml lock and check if m-addr is present in secondary-regs of parent, if not add it to invalid.reg and release ml lock
- ④ Send data to secondary and when finished Reset evict

Read access to the cache (MESI protocol)

If data is not present in the write buffer then we call the method lookup()

- ① iterate through all the cache ways
- ② call lookup() on the cacheway structure

- check Status[dmapping] is not in invalid state
- check that Tags[dmapping] is the address of the requested cache line

Side note

If lookup() was called from ^{non-write} (service == true) then if nothing was found in the cache we return false

If cacheway lookup() was successful we return cacheway

TRUE
do-a-snoop() only if not nested i.e. snoop called from one of the members
FALSE
for each member of the consistent group call Method grp-snoop()

- a) check in the write buffer under lock (wbl), if the line is present then it is first evicted from the buffer
- b) call lookup() on the cache from the group.

if a cache line has been found in 'grouped' cache we need to change the state of that line

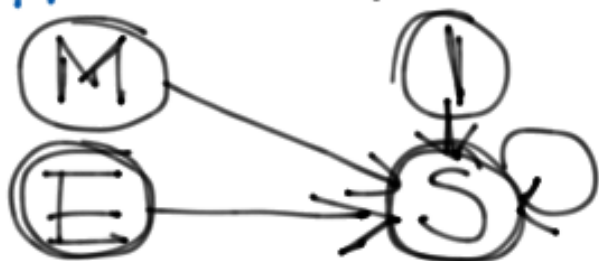
M modified E Exclusive S shared I invalid



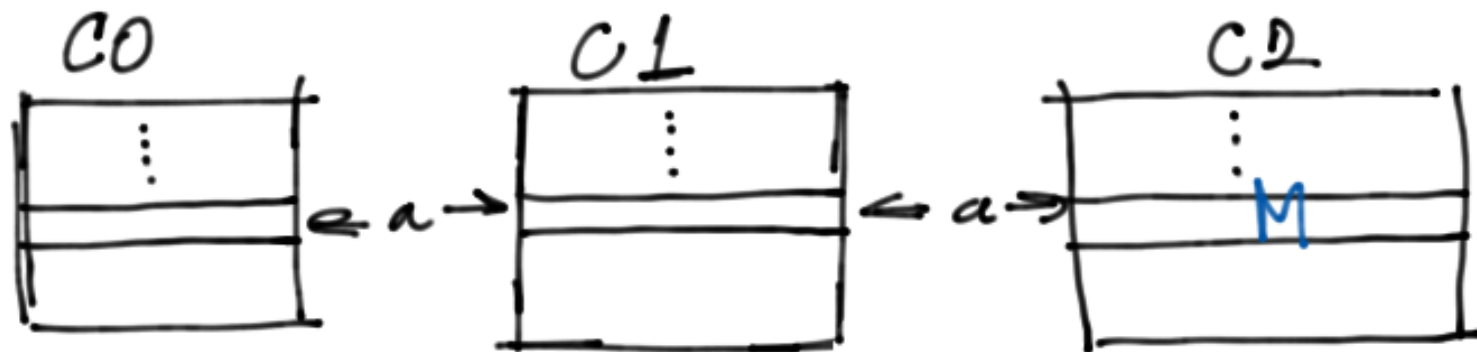
All this transactions are generated by snooping HW on this cache in response to commands issued by other cache.

If lookup() was successful in one of the neighbouring caches then we will call insert() on cache64 and consecutively on cacheway

if there is a dash such that there is line in the array that is in M state it will be evicted to the secondary storage.



The line on the cache that initiated read will move to state S



t_1 C0 requests address a that is in state M in C2
C2 sends a to secondary storage

t_2 C1 requests address a

should we had line for address a in state S at time t_2 for both C0 and C2 i.e. do not wait for reply from the secondary storage.

Does this mean that we need to use non-blocking transaction

~~READ MISS~~

if nothing was found we will do the secondary lookup:

- check if **secondary-reqs** do not have this address if it has it then request for that line has already been sent
- do a **secondary-lookup()** until **invalid-reqsc()** for that line is empty

While doing these checks lock **ml** on instance of **cache 64** is being held



the state of the line becomes **exclusive**

Write access to the cache (MESI protocol)

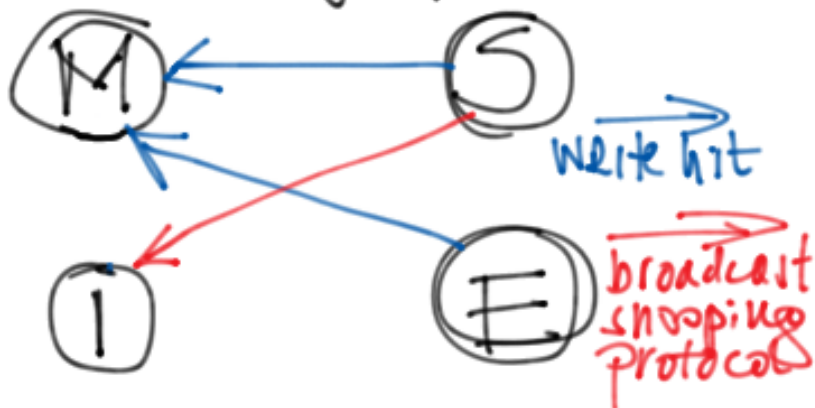
WRITE-THROUGH policy is not implemented at the moment.

① We first check whether line is present in the write buffer
(Warm buffered write)
If it is then we mark lanes in the buffer as dirty
both of these checks are done under
Write buffer lock **WBL**

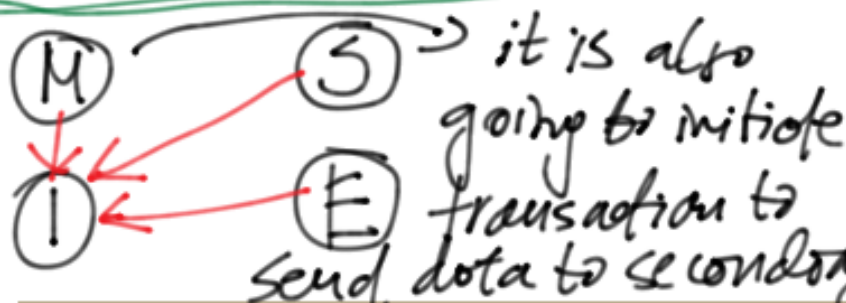
→ **TRUE** = warm buffered write

FALSE ② call **lookup()** method on instance of **cache64**
and pass **service = false**
YES line found

→ new state of the line is **M**
→ need to invalidate all lines
in consistent group that are in **S**



NO
① clean the write buffer
② check if the caches in
the consistent group have
the cache line
a) clean write buffer if
it has the line (**WBL**)
b) check if line is
present in any of the
caches in the group
(**Waylock**)



③ write new data
to buffer (**init()**)