


TIMECACHE

→ Tackles flush+reload attacks

→  shared software → flush+reload
No shared software → prime + probe

Attack ⇒ Flush + Reload

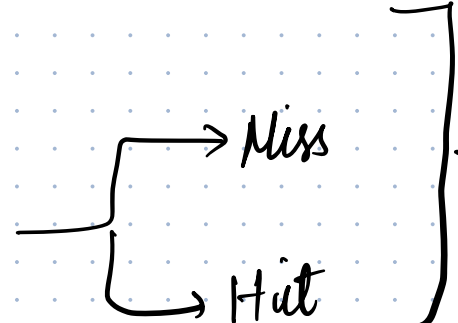
1) Flush

2) Wait

3) Reload

Miss

Hit

 But here there must be a Trojan & spy sight?

Side channel?

→ First access miss:

→ If a core is accessing a block it has never accessed before (and is put in that place by another core), then that access is considered as miss

→ Content security bit ⇒ S-bit

→ On hit, this S-bit is checked,

→ If it matches, it's considered as hit

→ If it doesn't, it is considered as a miss and fetched from main and the S-bit is reset to that corresponding core.

But when the miss happens, how does the replacement policy work?

↓
First access miss
only sends request
down but doesn't
replace the data

↓
Instead just set S-bit.

↓
S-bits are cleared
before giving the block
to a core

Any external
lib/OS handles
the saving of
a mapping of
S-bit using
timestamps

→ The resetting of the S-bit is handled based on timestamps.

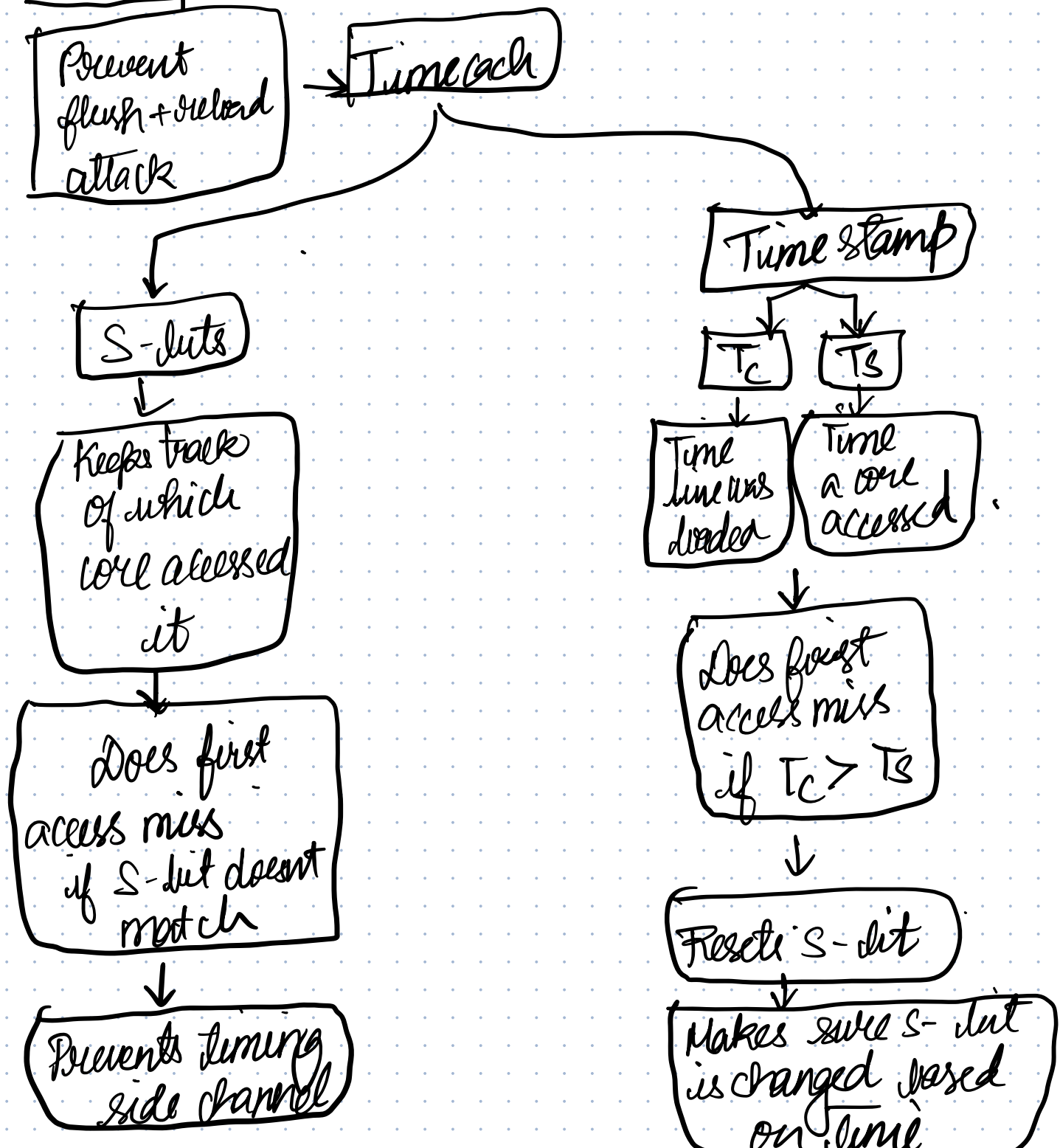
→ Each process would get its own T_s . This is updated whenever a process uses the block.

→ T_c stores when the cache line was brought in.

$T_c > T_s \rightarrow$ First access miss

$T_c < T_s \rightarrow$ Check S-dits

Mindmap:



First access
miss

Even if the block exists, a miss
is initiated

lower is thrown
and the LLC data
is used

why?

LLC has the most
updated data of that
block