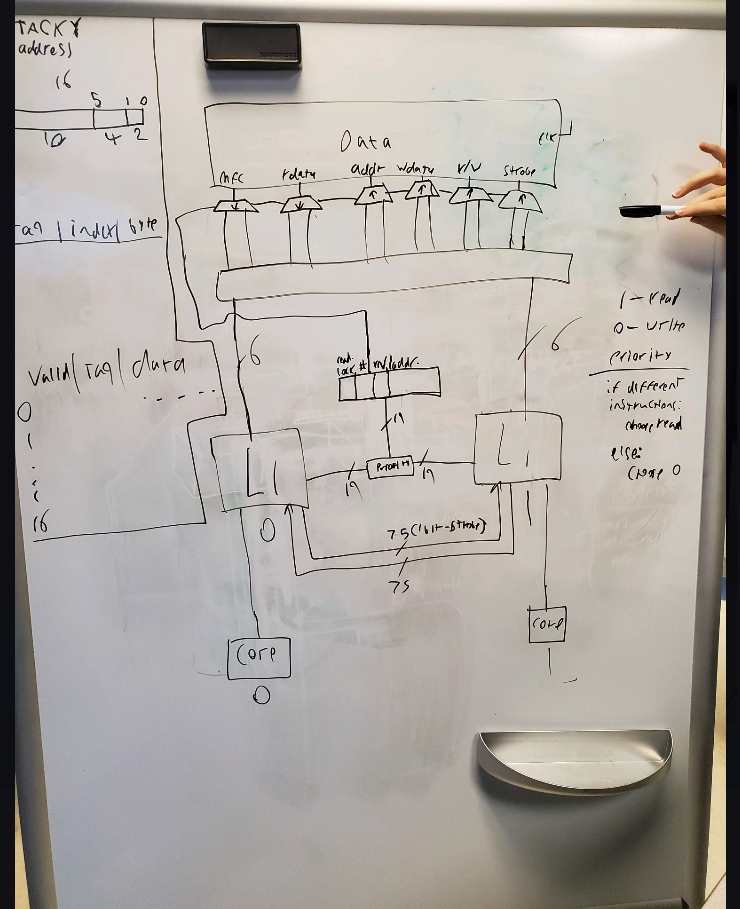
CPE 480 Assignment 4: Concurrent caches

Overall structure:



Cache flag logic:

* 2-bit flag: 19 bits: 16 bit addr, read lock bit, cache select bit, and read/write bit
* Read Lock bit: 1=locked, 0=unlocked
* Cache Select bit: specifies which L1 cache is in control
* r/w bit: specifies whether the cache is writing or reading
* addr: the address of the write/read

in addition to the flag logic would be a priority block:

* prioritize writes over reads
* prioritize cache 0 over cache 1 (mostly arbitrary)
* PRIO list:
  + W0
  + W1
  + R0
  + R1
* When the two diff caches want to write to the same address
  + L0 just puts its value in first.
* Pseudo code for the priority selection

If(diff instruction){

Choose read

}

Else {

Choose 0

}

Data logic:

* Each bus coming out of the data block has separate busses going to each of the L1 caches, controlled by the cache select bit from the “cache flag logic” section.
* I/O
  + mcf: Out
  + rdata: Out
  + addr: In
  + wdata: In
  + rnotw: In
  + strobe: In
  + clk: In

write while reading:

* check if the address being read is the address that you’re trying to read:
  + if not, go ahead and write
  + if so, wait for the read to complete, then write

note: cache 0 has priority when both caches are trying to write at the same time.

Update vs invalidate:

* Update: L0 changes and then shares the changes with L1
* invalidate: L0 changes and then tells L1 to go fetch it from slowmem
* update is more efficient

structure of cache line:

- valid bit: whether that line was instantiated with actual data

- tag: address

- blocks: data

update block:

* changed cache sends the tag and the data of the changed block to the other cache. The second cache looks and sees if it has that tag, then change the data if it does.
* 75 bits between the caches: 64 bits of data, 10 bits of tag and 1 bit for the strobe