Final report

Specifications

Specification: 4-way write back allocate with 128 sets and 1 blk = 64 bytes, LRU replacement policy

The cache is designed to take in 8 inputs:

- clk
- rst_b : asynchronous reset
- op : operation to be performed, 0 = read, 1 = write
- adr: 32 bit address, [31:13] tag bits, [12:6] index bits, [5:0] block offset bits, in this implementation a word has been taken to mean 1 bye
- start: gives the signal to start the operation
- mblk: 512 bit block, to be taken from main memory during a miss
- got: signal from main memory that the cache can read mblk
- i_word: word to be written at the address pointed to by adr in case of a write operation

The cache has the following 4 outputs:

- o_word: the word asked by the processor
- done: signals to the processor that o_word is valid and can be taken
- madr: a block address to main memory for requesting a blk in case of a miss, it represents the adr without the block offset bits, so adr[31:6]
- get: a signal to main memory to start bring in the block from madr

LRU:

- Implemented using custom registers that reset to a certain value
- Has 3 inputs: clk, rst_b, and updated[1:0] which tells the LRU which register to reset in age, everything younger than updated gets incremented.

Implementation process

1. Create a rough draft on paper having the course as a guide.
2. Try to implement the rough draft to check how feasible it is.
3. Realize that the given architecture presented for DM and the one presented for 4-way associativity is very simplified. Panic.

4. Go back and repeat steps 1 and 2 for a while, revising the design for the faced challenges.
5. Start from a small scale, start with the LRU module. Take 4 hours cause you wrote a variable name wrong and ModelSim does not know what intellisense even means.
6. Begin implementing the main cache memory, use lots of 128-i mux-es.
7. Barely be able to simulate it due to its huge size
8. Fail at successfully implementing it, try as good as you can and hope it's worth something even if non-functional.

Challenges

Implementing the cache, cannot really make a controller without the part you want to control. But the sheer logistics of it grinded progress to a halt, and so I focused on simply writing code that compiles with no warnings/errors, even if it isn't really functional.

Solution: NULL

Performance

Unable to be measured due to failing to implement the cache. Only performance I can review is that of the software used, and I am left unimpressed. Waiting minutes for something to simulate while you are on the event horizon of the deadline is not ideal. Not having intellisense isn't ideal either.

Finale

1% change, 99% hope. Did not finish. I will keep this experience in mind and maybe work on stuff the moment I get the stuff to work on.

Also kinda realized this is more about a cache controller, not the cache itself. So, yhea… kinda late for that realization huh?

Anyhow. Happy summer!