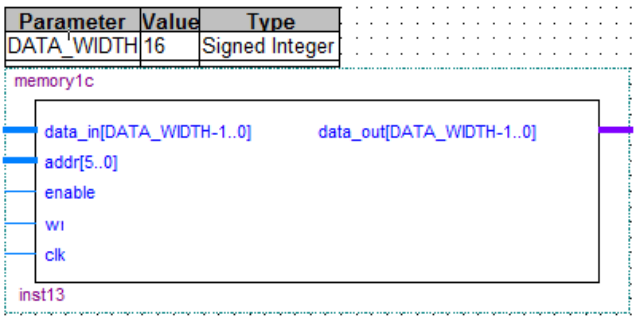
ECE/CS 552 Fall 2010 Project

Memory Module Description

TA: Guangyu Shi

In stage 2 of the project, you are asked to build up the memory system for WISC-F10 based on two 1-cycle L1 caches and a shared 3-cycle main memory. Please download the memory modules from the course website.

1. 1-cycle memory module for cache



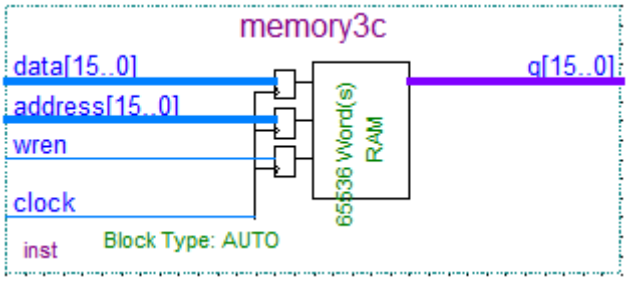
The source file for constructing caches is memory1c.v. It is read-combinational, write-sequential memory module. Each module has 64 entries. Since the cache line size is 8, you should instantiate 8 copies of this module as your data array, and 1 copy as your tag array.

Each instance of this module in data array should have a width of 1 word, that is, 16 bits. All the address buses, enable, clock and input data bus should be connected together. Only one write enable signal out of the 8 write enable signals can be active in one cycle. You can design the control logic with behavioral verilog.

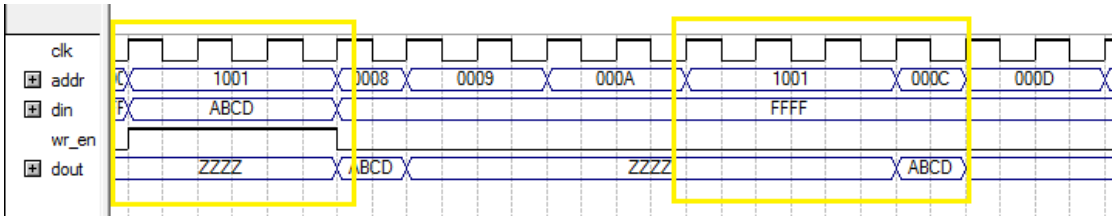
The width of the tag array is 8 bits: 7 bit tag and 1 valid bit. When updating the cache under a cache miss, tag array should be updated. Comparators can be written in behavioral verilog.
You CAN modify memory1c.v if you want, as long as it remains a read-combinational write-sequential design.

You may initialize your cache using the method in stage 1. However, I do not think it is necessary, as long as you define 0 as “invalid”.

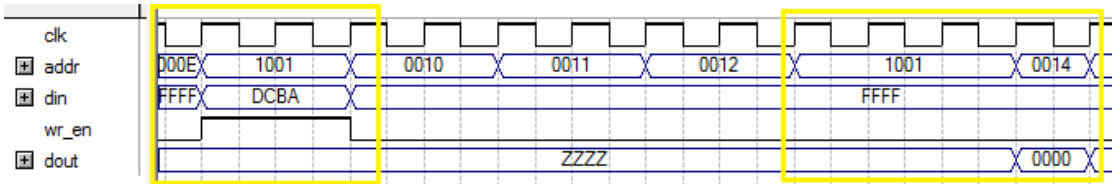
2. 3-cycle main memory



The main memory module memory3c.v has three cycles read/write delay. For each read, you must hold the address for 3 cycles, and the data will be read at the beginning of the fourth cycle. Otherwise, high impedance will appear on the read port q. For each write, you must hold the address, data and write enable signals for 3 cycles. Otherwise, 16'h0000 will be written in the address. Here are sample simulation waveforms of this memory module.

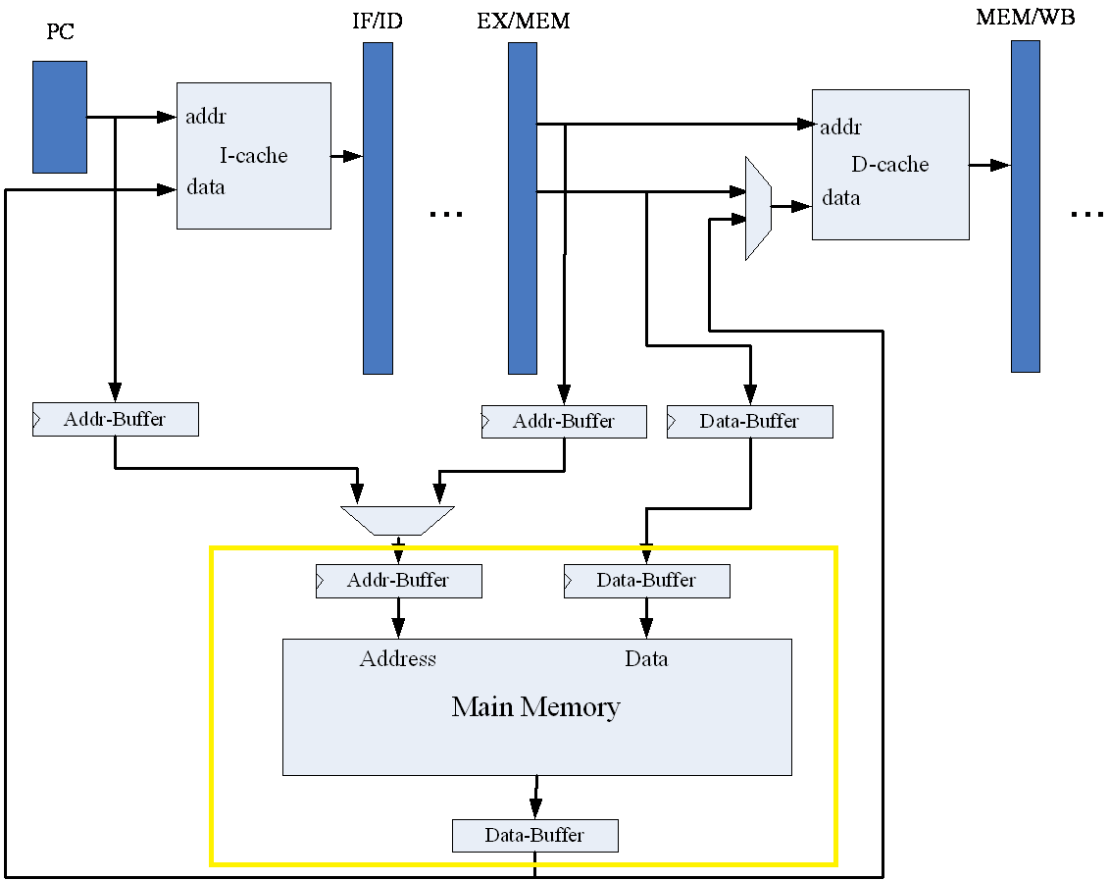


In this picture, write holds for 3 cycles. Next time when you read, after holding the address for 3 cycles the data will appear. It is identical to the value written in.



In this picture, write only holds for 2 cycles. Next time when you read from the same location, you read "0000". Not only the new data did not written in, but the old data is also destroyed.

One thing worth notice is that this memory is fully synchronized. Although we do not see latches on the output, the output q is also a registered value. Recall figure 1 in the project description:



You may regard memory3c as a RAM plus the address and the directly connected data buffers (in the yellow box).

You SHOULD NOT modify memory3c.v. Besides, do not reconfigure memory3c using Quartus Megafunction Wizard, because it will overwrite the source code.

To initialize this memory, use the assembler for asyncram (asmbl-asyncram.pl) to generate “meminit.mif”. The input assembly code format is the same. The output format is slightly different than the assembler we used in the first stage.