ECE/CS 552: INTRODUCTION TO COMPUTER ARCHITECTURE

Project Description – Phase 2

Due on Dec. 4th.

You are allowed to use the published phase 1 reference design in its entirety or portion of it to help you design phase 2 project. However, in your submission, the portions of your phase II design that are copied from the phase 1 reference design should be declared in both the phase 2 project report as well as the Verilog code of corresponding modules in the form of comments.

Phase 2 project submissions include a required part and an optional bonus part. Everyone is required to submit the required part. The required part submission should NOT include any components of the optional bonus part so that their performance may be compared fairly. Violations may be penalized with a 30% penalty of the required part grade.

Both required part and optional part submissions should include comprehensively commented, executable Verilog code. The bonus part should also include a corresponding project report, which is not required for the required part. In the project report, you should provide an overview of your design, especially how the cache controller works and how you modify the required part.

The submission is same as stage 1, except that you need to submit a separate bonus version for the bonus part grading. The required part should not include anything of the bonus part.

1) Review of the First Stage Project

Before you start the second stage of the project, you should have a working design pipelined design that implements the following instructions.

Table 1: Table of opcodes

Function	Opcode
ADD	0000
PADDSB	0001
SUB	0010
NAND	0011
XOR	0100
SLL	0101
SRL	0110
SRA	0111
LW	1000
SW	1001
LHB	1010
LLB	1011
B	1100
CALL	1101
RET	1110
HLT	1111

To see the detailed information of the ISA, please refer to the first stage project description. You cannot proceed if you do not have a working 5-stage pipe design for these 16 instructions.

2) Memory System

The memory system consists of a main memory, and separated I-cache and D-cache. The cache access time is one cycle for a cache hit. The main memory read or write delay will be 4 cycles.

Both caches are direct-mapped, and have a block (line) size of 4 words (8-bytes), and a data array size of 64 blocks (512 bytes). The main memory is word-addressable meaning each of the 16-bit memory address will results in reading or writing two bytes. Thus, the entire memory space is 2×2^{16} = 128KB. Hence, when presented to the cache, the 16-bit memory address will be partitioned into the following fields: a) address[1:0] is the block offset, address[7:2] is the index field and address[15:8] is the tag field. Each cache block also has two status bit: Valid, and Dirty. The Valid bit is set when a data block is read from the main memory due to a cold miss. It is reset during cache initiation. The Dirty bit is set when the valid content is modified due to a memory write that results in a cache hit. It is reset when the corresponding block of data is written back to the main memory. The Dirty bit is used only during a memory write access. Hence it is ignored for the I-cache since no memory write operation needs to be performed for I-cache.

Verilog models for the shared main memory (unified_mem.v) and a basic caches (cache.v) will be provided. Note that both the I-cache and the D-cache are different instantiation of the same module cache.v.

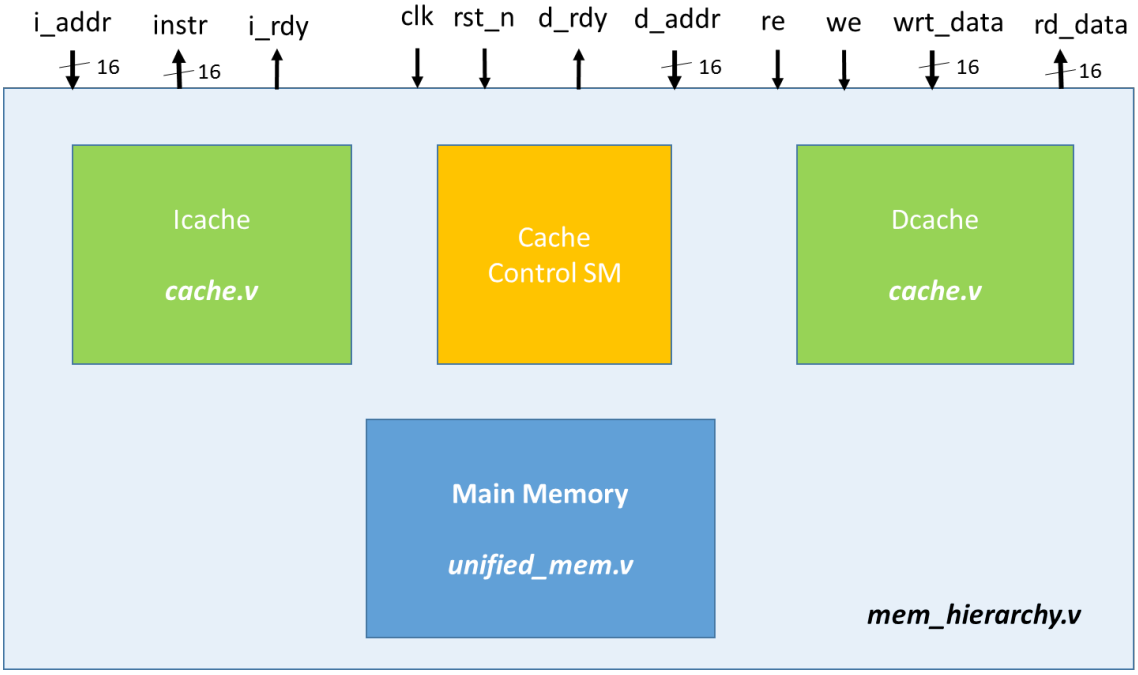

Figure 1: System Memory

(Miss a stall signal)

A block diagram of the memory system is depicted in Figure 1. The input output signals of the memory system is described in the table below:

Signal:	Dir:	Description:

clk	in	clock
rst_n	in	Active low reset
addr[13:0]	in	Address. Lowest 2-bits not used (4-words/line)
wr_data[63:0]	in	Data to be written to cache line
wdirty	in	Hold high while writing to mark a line as dirty
we	in	Write enable to write a line to the cache
re	in	Read enable
rd_data[63:0]	out	Cache line read out
tag_out[7:0]	out	Tag bits for the cache line. Need this when evicting line from cache
hit	out	Hit means the tag bits matched and the line was valid
dirty	out	Cache line has been modified. Need to write back.

As shown in figure 1, both the I-cache and the D-cache will share the same main memory. Hence, it is possible that both the I-ache and the D-cache encounter cache miss in the same cycle. When this happens, the D-cache miss should be serviced first.

Upon a cache hit, both caches will be able to handle the transaction autonomously without the oversight of the cache controller. Only upon a cache miss, the cache controller state machine will get involved to handle the cache miss. Since the main memory has a single read port, only one cache miss can be serviced at a time. Thus the same cache controller needs to service both the I-cache read miss or the D-cache read or write miss. The read and write miss handling policy will be elaborated later.

Initially, assume both the program (machine code) as well as data are both stored in the main memory. It is necessary to reserve a block of high memory space for storing the program. The address space that the data are stored or read must not overlap with the address space of the program. For example, in a 16-bit address space, you may want to designate the data address space from 0x0000 to 0x7FFF and the program address space from 0x8000 to 0xFFFF.

Verilog modules are provided (cache.v for both the I-cache and the D-cache, and unified_mem.v for the main memory). The interface of these memories is described below:

Note that upon a read hit, the cache output the entire cache line. Thus, a MUX is needed to select the desired word using the word offset field, namely, address[1:0].

Cache Timing

The caches are single cycle access. Address and ***re/we*** (not both) should be presented and valid in the first half of the clock. The cache reads data from the main memory upon a triggering clock edge. Meanwhile, the tag field is also updated. Thus, the hit status signal will change to hit (due to matching tag values) and data will be read or written into the cache.

Unified Memory Interface

Signal:	Dir:	Description:
clk	in	Clock
rst_n	in	Active low reset
addr[13:0]	in	Address. Lowest 2-bits not used (4-word access)
re	in	Read enable
we	in	Write enable
wdata[63:0]	in	Data to write to 4-word line
rd_data[63:0]	out	64-bit line read

rdy	out	Indicates operation (read/ write) is complete this cycle

Unified Memory Timing

The main memory has a timing of 4-cycles for a read or write. The read and write both occur on the clock high phase of the 4th clock cycle. ***Rdy*** is also asserted on the clock high phase of the 4th cycle. Both ***re*** & ***we*** and ***addr*** should be asserted prior to the rising edge of the first clock cycle of access. This timing relations are depicted in the timing diagram in Figure 2.

A write timing is similar to read timing. In particular, the write occurs during the clock high of the 4th clock cycle.

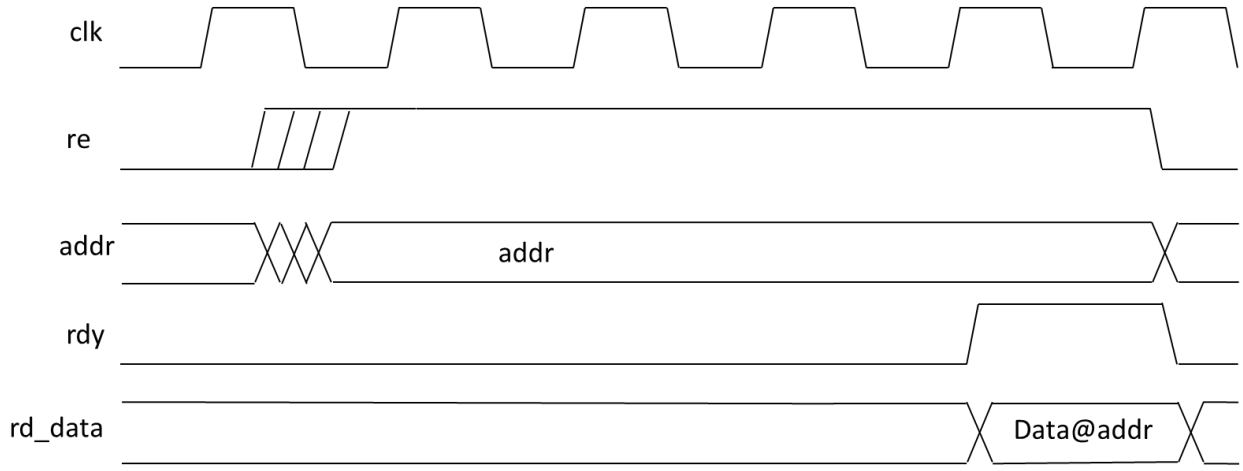

Figure 2: Timing Diagram of Main Memory

Cache control

Upon a read miss (I-cache or D-cache) or a write miss (D-cache only), the cache unit will output hit = 0. The cache control will pick up this signal and act on it. Note that when both caches encounter misses in the same cycle, the D-cache will be serviced first.

I-Cache Read Miss: When the I-cache encounter a read miss, the cache controller (CC) will send signal to the pipeline control unit to stall the pipeline at the PC stage by deactivate the PCWrite control. It will also send the missing *block* address to the main memory and issue a memory read request. The CC will check for the Rdy status from the main memory. When Rdy is set in the present cycle, meaning the fetched cache block is at rd_data field, the CC will send control signal to the I-cache to write the data into the designated cache block and update the corresponding tag field. Thus in the next cycle, the tags will match and execution resumes.

D-Cache Read Miss: The CC will send control signal to the pipeline control unit to stall the pipeline. The CC will first check the valid bit of the block at the cache address corresponding to the index field. If it is invalid (cold miss), or if it is valid but the dirty bit is deactivated, then continue to read block from main memory. Else, if it is valid, and dirty then present the block address of the current block (current tag values || index field) to the main memory, place the dirty block on the cache output, and issue a memory write request. The CC will wait for the Rdy signal from the main memory to activate. Then, the CC will present the

desired block address to the main memory and issue a memory read which will deactivate the Rdy signal. When Rdy signal is activated (again), write the fetched block into the cache and update the tag field. Then in the following cycle, cache will respond with a hit and the read will proceed in the same cycle.

D-Cache Write Miss: A write-back policy will be implemented for the D-Cache. The data will always be written into the cache, and the dirty bit of that block will be set accordingly. Upon a write miss, the CC will check for Valid and Dirty bit of the corresponding block. If both are active (= 1), the current block will need to be written back to the main memory first, and the desired block will be fetched from the main memory. If it is not dirty, then the missing block will be fetched from the main memory. The cache status then will turn into Hit in the following cycle and the write will take place at that cycle.

3) Bonus part

Extra bonus credits (up to additional 25%) will be awarded for the projects that implement 2-way set associative data cache and/or synthesis of your code.
The bonus part will be graded only after you pass all the tests given in stage 1 and 2. If there are error found in the required part, the bonus submission will be omitted.

a) 2-way set associative data cache (up to 15%)

In this part, you have two options. First option is to change the direct-mapped data cache to 2-way set associative one based on the cache module we supply. A second option is to change the cache modules using what are provided at
http://pages.cs.wisc.edu/~david/courses/cs552/S12/includes/cache-mod.html
The first option will give you up to 10% bonus, and the second option gives you up to 15% bonus if you do it correctly. You must select only one of those two option. Whichever option you choose, your cache should meet the following detail requirement.

The data cache has the same size as the module we supply, which has a block size of 4 words (8-bytes) and a data array size of 256 words (organized as 32 2-way lines with each set 4-word). The address[1:0] field is the word offset, address[6:2] is the index field and address[15:7] are tag bits.

When the cache performs a read, the 2 tags of a same cache line should be compared at the same time. If there is a hit, the corresponding data of the hit set should be given. Otherwise there is a miss.
When the cache performs a write, a least recently used (LRU) replacement policy should be implemented. In order to do that, the LRU set of each cache line should be recorded. When there is write, if it is write hit, just directly write the data to the correct position and update the LRU set number to the other one in the same cache line. If it is a write miss, always write the LRU set back to unified memory. And after write the new data-to-data cache, the LRU set number should be updated.

As for the interface, only one input, named toggle, is needed to add to the supplied cache module. The toggle is used to update the recorded LRU set number of each cache line. When there is a read or write, the toggle signal should be asserted to the 2-way set associative data cache. Once receiving the toggle signal, the data cache should update the least recently used set number of corresponding cache line.

b) Synthesis (up to 10%)

The synthesis of your code is not required in the required part of your project. The TA will only check the synthesis of your code by eyes. If there is no obvious non-synthesizable code, you will not lose credit for synthesis in the required part of your project. However, if you can pass the synthesis using the script

given by TA, which means there is no latch, and can meet the timing constrain, 15% bonus will be given as bonus.

4) Project Grading

This phase of the project will be worth 70% of the project score (bonus part is not included).

$$TotalScore = QuantScore\_stage1 * 30\% + QuantScore\_stage2 * 70\% + BonusScore * 25\%$$