Project 2 Cache Controller

1 Purpose

This project is intended to help you to understand the cache architecture and its mechanism.

2 Problem

In this project, you will design a first-level data cache controller with Verilog HDL step by step. You may need to review the knowledge about the language to make sure you can finish the project smoothly.

3 Cache characteristics and mechanism

The TMS320C621x/C671x DSP has a two-level memory architecture for program and data, the first-level program cache (L1P), the first-level data cache (L1D), and the second-level memory (L2). In this project, you only need to concern about the L1D cache. Table 1 lists some specific parameters that may be used in this project. You can get more information from the reference guide pdf, *TMS320C621x(C671x) Two Level Internal Memory Reference Guide*.

Table 1

A Simplified Model of TMS320C621x/C671x DSP	
Internal memory structure	Two Level
L1D size	4Kbytes
L1D organization	2-way set associative
L1D line size	32 bytes
L1D replacement strategy	Least Recently Used (LRU)
L1D read miss action	1 line allocates in L1D
L1D read hit action	Data read from L1D
L1D write miss action	No allocation in L1D, data sent to L2
L1D write hit action	Data updated in L1D, line marked dirty
L2 line size	128 bytes
L2 → L1D read path width	128-bit
L1D → L2 write path width	32-bit

Note1: When a dirty block is replaced, the whole L1D cache line must be written back, but the L1D to L2 write path width is only 32-bit, i.e. you need to repeat 8 times to complete. Similarly, one load from L2 cache returns 128 bytes of data and you need to repeat 8 times to complete.
Note2: For simplicity, you can assume an infinite write buffer, and ignore TLB. Besides, assume only one ld/st is permitted to be processed at a time.

4 State machine design

In the first step of designing a cache controller, you need to design a state machine. You need to consider how many states you need, what functions each state realized, and how the states transfer to each other.

(1) Draw a state diagram, assign each state a unique binary code to indicate, and mark the

transition condition signal.
(2) According to your state diagram, list a state transition table, which gives the next state of each current state based on all the possible condition signals. You must make sure all of the state codes that may show up in real circuit are listed in the table (including the ones that don't show up in your state diagram).
(3) Draw a flow diagram of your cache controller. This is very helpful for your design later.
Hints:

The functionality of L1D cache is to accomplish the processing of two kinds of instructions: Ld and St. When you need to write back or load data from L2 cache, you will probably need more than one cycles (also more than one states).

You can reference your supplementary materials of your lecture, Digital System Design I, to review the knowledge about state machine.

5 Circuit design

Based on your flow diagram, design your cache controller and draw the circuit. You should indicate the important signals aside their wires, and make a list beside the diagram showing the meaning and function of each signal (with their source and destination if necessary).
Hints:

You can realize the cache controller based on a counter or a few mux. If you chose to design based on counter, then the counter's different states map to different states in your design. You only need to add some logic to make it transform properly and give right output signals. If you chose to design based on mux, you should use a few mux (the number of which is equal to the width of your state code). The current state code is used as the selective signal to all the mux, and you need to give each mux proper inputs so that it outputs the right signal. And the combination of the outputs of all the mux gives the next state code.

6 Verilog HDL realization

Since you have already completed the circuit design, you can easily describe the circuit in Verilog HDL. Please add enough notes to make your program easy to understand, because this might affect your mark. You should focus on description of behavior, and add some description of structure and data stream if necessary.

Remember that what you should focus is the output control signals and the states change of cache controller; you can ignore the specific data.

The module name and some of the signal names are given below, you must not change these names in your program. And the meanings of the signals are given for reference. If the definition is not clear, you can define the meaning of the signal according to your own needs and understanding. But be sure to describe the meanings of the signals in your report clearly. Of course, you can also add some signals according to your own needs.

```
module    cache_controller(...);
input     ld, st;
input     addr[31:0];
input     tag_loaded[20:0], valid, dirty;
input    l2_ack;
output   hit, miss;
output   load_ready, write_l1;
output   read_l2, write_l2;
```

signal meaning:
ld/st – load or store request
addr[31:0] – cache access address
tag_loaded[20:0] – the tag of the indexed cache data address
valid – valid indication of the block in cache
dirty – indicate whether the cache line is dirty
l2_ack – indicate that the data loaded from L2 cache is arrived
hit/miss – indicate the access is a hit or miss (these signals are used to make your design understandable)
load_ready – indicate the data is successfully loaded and the data is ready for processor
write_l1 – enable for L1 cache write
read_l2 – load request to L2 cache
write_l2 – enable for write back to write buffer

Note:

All of your work must be presented in electronic version (no paper version), including the diagram drawing. You can make use of certain software to do that.

Describe your design process in as much detail as you can in the report. The report should include the simulation results to ensure the correctness of your code.

You can reference Character 5.9 (Using a Finite-State Machine to Control a Simple Cache) of the textbook, *Computer Organization and Design: The Hardware Software Interface [RISC-V Edition]*.

Requirement: A report (word or pdf) on your step-by-step implementation of the above processes, and your Verilog HDL code.
Deadline: 2022.1.4 23:59

Compress your files into a rar (or zip) format named by your student ID and name, for example 319010xxxx_yourname_project2, and upload the compressed file to Learning in ZJU (http://courses.zju.edu.cn/) before the deadline. If you want to modify your program, you can upload it again before the deadline to cover the old version.

No Copy! The same reports or codes will both get zero scores.