Computer Architecture Lab2 Report

B09901171 電機二 黃柏睿

January 14, 2022

1 Modules

1.1 PC

Program counter. Count the instruction index when PCWrite is on and MEMStall is off. Done by TA.

1.2 Data Memory

Store the datas. Takes 10 cycles for access, and output ack signal when data is ready. Done by TA.

1.3 Pipeline Registers

Transfer data from previous stage to next stage at clock edges, and hold the data when stall signal from cache or hazard detection unit is on. Reset all data to zero when reset signal on.

1.4 DCache Controller

Handle memory access request from CPU. Determine read/write hit/miss from SRAM and control the sequential action as the state graph below. In each state, there are four signal to be controlled: mem_enable, mem_write, cache_write and write_back, which control the action of memory and SRAM.

First of all, the controller is usually in "Idle" state, when CPU send memory access request and unfortunately miss on SRAM, it will enter "Miss" state then check if the cache block is dirty. If it is dirty, controller will enter "Write Back" state and loop in the state until the memory is ready.

After writing back the data or the cache block isn't dirty at first, controller will enter "Read Miss" state which reads data from memory. Same as "Write Back" state, it will loop in the state until the memory is ready.

After fetching data from memory, controller will enter "Read Miss OK" state, and write the new data on the cache block. Finally, return to "Idle" state.

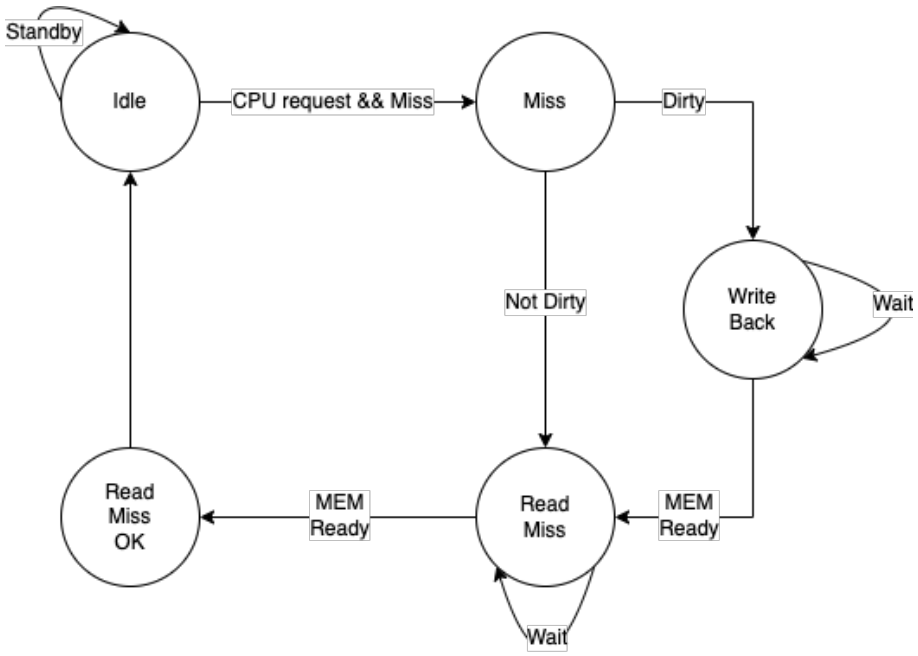

Figure 1: State Graph of the Controller

State	mem_enable	mem_write	cache_write	write_back
Idle	0	X	0	X
Miss	0	X	0	X
Write Back	1	1	X	1
Read Miss	1	0	X	0
Read Miss OK	0	X	1	X

1.5 DCache SRAM

A 2-way associative 1KB cache. Stores tags and data of the cache. Reset all data to 0 when receive reset signal. Output whether the request is a hit by comparing tags in both 2 set, and output the data and tags. There is a 16*bit* register, LRU, which stores the least recent used set, in order to achieve LRU replacement policy. This table is updated at every access. When there is a miss, it can easily find which set should be replaced according to the LRU table, and the tag and data of the block will be output.

note: For my debug, the content of SRAM will be output to *cache_data.txt*.

1.6 Testbench

Time domain simulation of CPU. Done by TA mostly. Output three files after simulation, the first one record value of registers and memory every cycle (*output.txt*), the second one record cache status every access (*cache.txt*), the last one record cache data every cycle (*cache_data.txt*).

2 Difficulties Encountered and Solutions in This Lab

2.1 Tag and Data Output from SRAM when Miss

At first, I think that the outputs from SRAM at miss are don't care. However, when I try some testcases, I found that there are error write back on clean blocks. After reviewing the waveform, I found that the controller needs to know the status of the cache block on misses, so that it can determine if write back is needed. Therefore, I make the SRAM output the LRU block on misses, the problem is solved.

2.2 Final Flushing

I found that the the second cycle from the end of the output is correct but there are strange 0s at the last cycle. After dumping out all the tags and data in the cache, I found that there are invalid cache block flushed to the memory, which causes the error. However, checking the cache block if it is valid before writing back solved the problem.

3 Development Environment

Hardware & OS: MacBook Pro (13-inch, 2016, Two Thunderbolt 3 ports) with macOS Big Sur 11.5.2
IDE: VScode with TeroSHDL extension. Using iverilog for compiling, gtkwave for .vcd waveform visualizing.