Report of Pipelined CPU with L1 Data Cache

Team GJMI: B04202008, B04901036, B04901165

January 1, 2018

1 Members and Team Works

1.1 B04202008

- Data cache implementation

1.2 B04901036

- Data cache debug

1.3 B04901165

- Pipelined CPU modification for data cache

2 Implementation

The implementation of the pipelined CPU with L1 data cache was straightforward. We started from the sample code provided by TA and completed the data cache implementation. Then, the memory stage in pipelined CPU from project 1 was modified to insert the L1 data cache. After testing the data cache functioned correctly, the CPU implementation was complete.

3 Cache Controller in detail

3.1 control units & states

There are 4 states(STATE_IDLE, STATE_MISS, STATE_READMISS, STATE_READMISSOK, STATE_WRITEBACK) and 4 controls units (mem_enable, mem_write, cache_we, write_back) (Fig. 1).

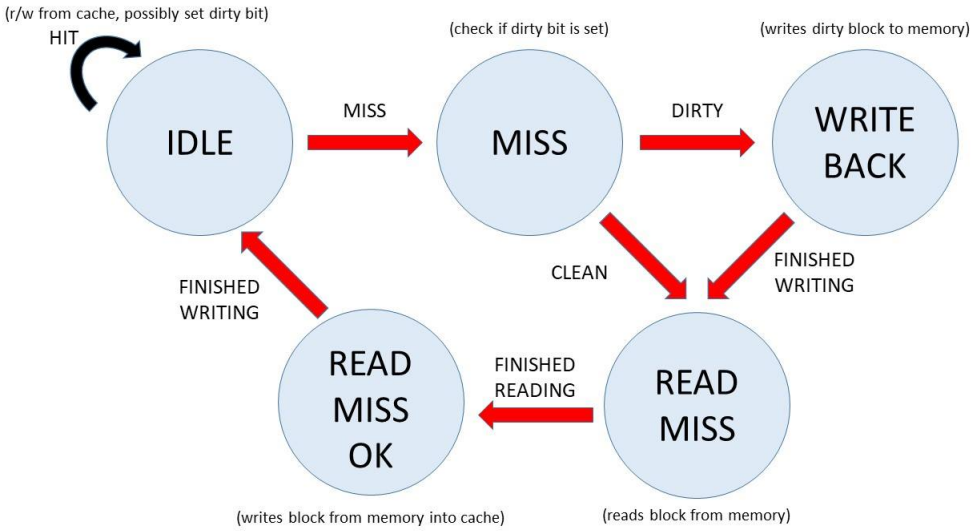

Figure 1: State of L1 data cache controller.

STATE_IDLE, STATE_MISS
Initially, it is in the STATE_IDLE. After being called, the cache goes to STATE_MISS. The job of STATE_MISS is to decide whether it is necessary to write back by checking the dirty bit. If the dirty bit is true, then entering STATE_WRITEBACK. Otherwise, skipping writeback process and directly entering into STATE_READMISS.

STATE_WRITEBACK
Before going into STATE_WRITEBACK, write_back, mem_write, and mem_enable control units have to be turned on so that we can write into memory. After entering into STATE_WRITEBACK, the job is waiting for data memory acknowledge. Once getting acknowledge, write_back and mem_write unit need to be turned off and ready to go to STATE_READMISS; Otherwise, keeping staying in STATE_WRITEBACK.

STATE_READMISS
The job of STATE_READMISS is to load data from memory, so mem_enable must be turned on beforehand. Then, similar to STATE_WRITEBACK, cache keeps waiting unilt acknowledge is gotten. The only job left is to write in the sram and nothing to do with memory, so we have to turn off mem_enable and turn on cache_we. The job is almost done until now, and cache goes into STATE_READMISSOK.

STATE_READMISSOK
STATE_READMISSOK does nothing but turn off cache_we after writing into sram. Afterwards, cache goes into STATE_IDLE again.

4 Problem and Solution

4.1 CPU stalled forever

Problems: CPU stalled forever after read-miss occured in L1 data cache.

Solutions: The stage register in the data memory should be initialized. Without initialization, the stage of the data memory was undefined, so it could never ack the CPU request, hence the CPU stalled and waited the ack from the memory forever.

4.2 Read-miss output data different to the reference

Problem: TestBench outputted the data in cache at the time when read-miss occured, which is before the data was updated. However, the output in the reference file is the data after it was updated.

Solution: We modified TestBench to delay the output timing when read-miss occured. Specifically, TestBench outputs data at the cycle when the new data block is ready in cache, which is consistent with the reference file.