Computer Organization 2016

HOMEWORK IV

Due date: 2016/7/4 23:59

The goal of this homework is to let the students be familiar with the MIPS instruction set and Verilog, a hardware description language (HDL) used to model electronic systems. In this homework, you need to implement a Cache of MIPS CPU. Follow the instruction table in this homework and satisfy all the homework requirements. Please use the benchmark provided by TA to verify your module correctly. The verification tool is Modelsim.

General rules for deliverables

- This homework needs to be completed by INDIVIDUAL student. If your code is copied, you will not get any scores.
- Compress all files into a single zip file, and upload the compressed file to Moodle.
  - The file hierarchy

    F740XXXXX(your id)(folder)

    SRC(folder)   * Store your source code

    F740XXXXX.docx(your Project Report)

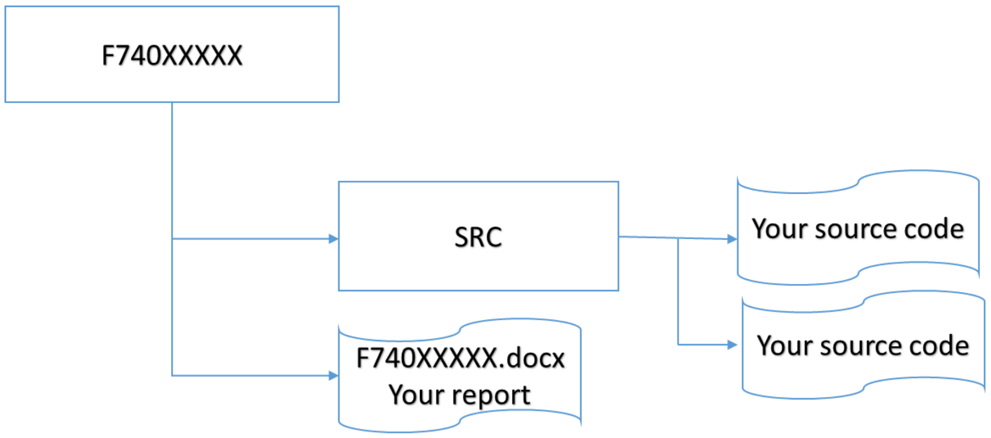


Fig.1 File hierarchy for homework submission

- **Important!** AVOID submitting your homework in the last minute. Late submission is not accepted.
- You should finish all the requirements (shown below) in this HW and Project report.
- Please use Project Report template on Archive to finish your Project report. HW4 Archive on Course website:

  Computer Organization 2016 course website

Exercise

You need to implement a Direct-map Cache module to speed up the pipeline CPU. In addition, you need to verify your module by using Modelsim. Note that, TAs will provide a simple architecture for the Cache and you just need to implement some incomplete/missing modules, e.g. Cache.v, Cache_Control.v… ,etc.

Please finish all the modules, use the benchmark provided by TAs to verify the Cache, take a snapshot, and finally explain the snapshot, including the wires, signals …… in your report.

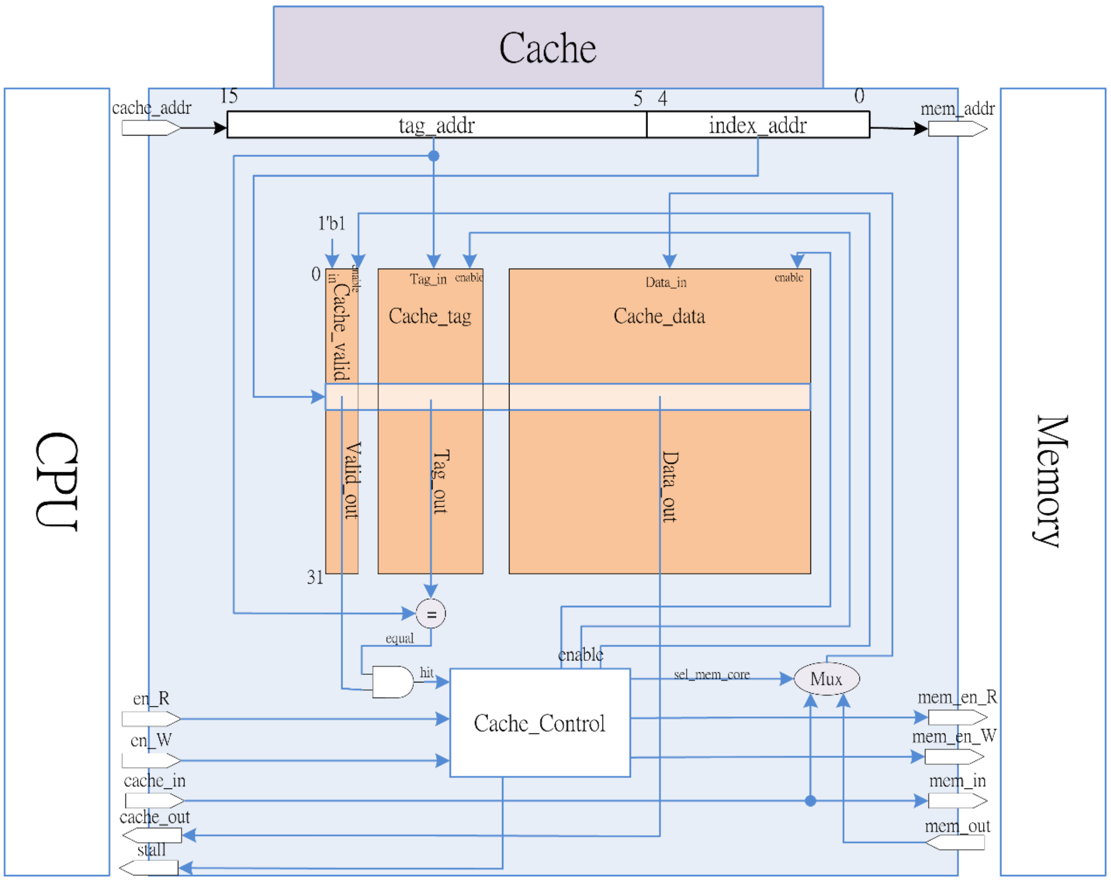


Fig.2 Cache DataPath reference

Description

1. The Direct map Cache you need to implement has 32 cachelines. Each cacheline has one valid bit, 11 tag bits, and 32 data bits.

2. When the Cache receives the request address from the CPU, there are 4 cases:
   - I. Cache hit –
     - i. Read hit – directly read the corresponded data in the cache
     - ii. Write hit – write data in both cache and memory(write through)

II. Cache miss –

  i. Read miss – need to inform memory to get the missing data, and then place the data into the cache.

  ii. Write miss – skip the cache, directly write memory(write through)

3. There are 4 modules, Cache_Control.v, Cache_valid.v, Cache_tag.v, and Cache_data.v, that will be used in the Cache.v:

   I. Cache_Control – the module coping with the cache hit or miss, enabling the 3 parts of the cache (valid, tag, data) and sending signal to memory to fetch the missing data

   II. Cache_valid – the module storing the **valid part** of the cache

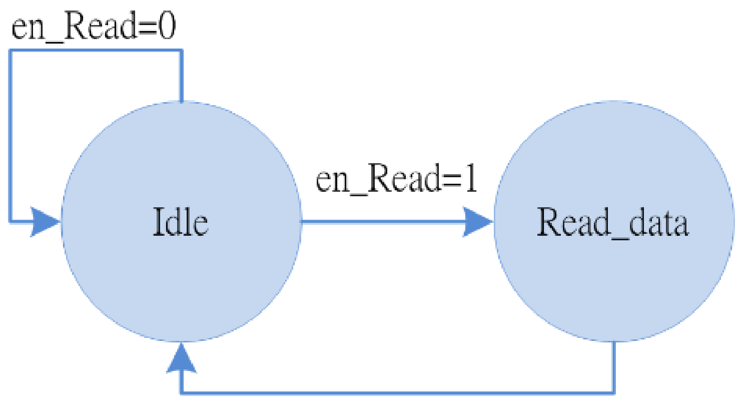
   III. Cache_tag – the module storing the **tag part** of the cache

   IV. Cache_data – the module storing the **data part** of the cache

4. Note that the memory modules(IM.v, DM.v) are not ideal memories anymore, that means there exists latency during each access to the memory. But in order to simplify the module, only the READ access to the memory has the latency, and it will follow the FSM(finie state machine) below:

   I. Idle state: wait for en_Read signal

   II. Read data state: output the data and return to Idle state

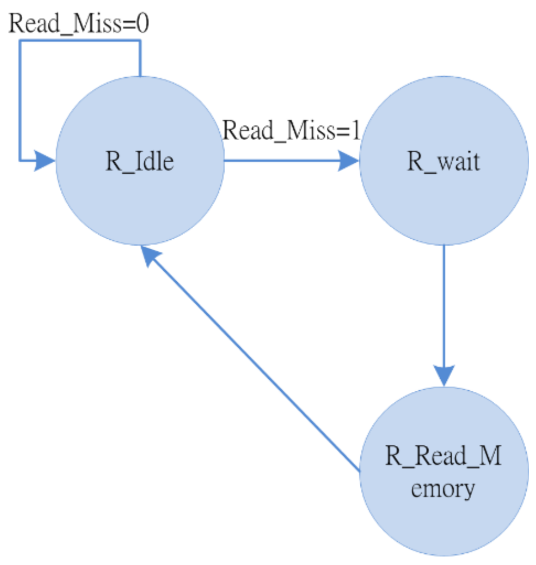


5. The Cache_Control also includes an FSM to handle the read miss latency, The state diagram is shown below:

   I. R_Idle: wait Read_Miss signal become 1,then raise Read_mem to inform memory

   II. R_wait: wait for the response of memory

   III. R_Read_Memory: get data from memory and **replace the corresponded cacheline,** then return to Idle state



6. Port description:

Cache.v		
port	I/O	Description
clk	input	Clock signal
rst	input	Reset signal
stall	output	When Read miss occur, the signal used to stall the CPU
cache_addr	input	The request address from CPU
en_R	input	Represent the current access is read access
en_W	input	Represent the current access is write access
cache_in	input	The data input from CPU
cache_out	output	The data output to CPU
mem_addr	input	Memory access address
mem_en_R	input	Read memory enable signal
mem_en_W	input	Write memory enable signal
mem_in	input	The data received by memory
mem_out	output	The data that will be written into memory

Cache_Control.v		
port	I/O	Description
clk	input	Clock signal
rst	input	Reset signal
en_R	input	Represent the current access is read access
en_W	input	Represent the current access is write access
hit	input	Represent the current access is hit in the cache
Read_mem	output	Read memory enable signal

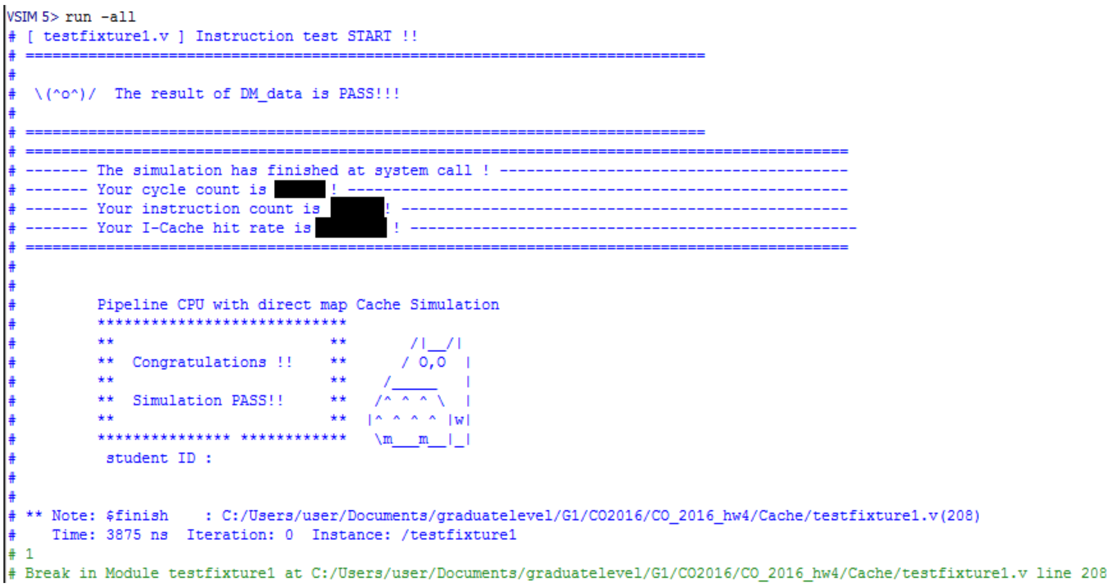
Write_mem	output	Write memory enable signal
Valid_enable	output	Enable writing valid part of cacheline
Tag_enable	output	Enable writing tag part of cacheline
Data_enable	output	Enable writing data part of cacheline
sel_mem_core	output	Select the Data_in is from memory or from the core(CPU)
stall	output	When Read miss occur, the signal used to stall the CPU

Cache_valid.v / Cache_tag.v / Cache_data.v		
port	I/O	Description
clk	input	Clock signal
rst	input	Reset signal
*_Address	input	The address (index) of the cacheline
*_enable	input	The write enable signal
*_in	input	The input data that is written into address of *_Address when the *_enable is 1.
*_out	output	The output data that is read from address of *_Address

※ * = Valid, Tag, Data

Homework Requirements

1. Please implement these modules:
   I. **Cache.v**(wire connection)
   II. **Cache_controll.v**(signal assignment)
2. Verify your Cache with the benchmark and take a snapshot (e.g. Fig.3), and **please write down why the hit rate could reach such a high level? (you can explain by observing the IM_data.dat & the waveform)**



```
VSIM 5> run -all
# [ testfixture1.v ] Instruction test START !!
# ============================================================================
#
#  \(^o^)/  The result of DM_data is PASS!!!
#
# ============================================================================
# ============================================================================================
# ------- The simulation has finished at system call ! --------------------------------------
# ------- Your cycle count is [redacted] ! --------------------------------------------------
# ------- Your instruction count is [redacted]! --------------------------------------------
# ------- Your I-Cache hit rate is [redacted] ! --------------------------------------------
# ============================================================================================
#
#
#        Pipeline CPU with direct map Cache Simulation
#        ****************************
#        **                        **       /|__/|
#        **  Congratulations !!    **      / 0,0  |
#        **                        **     /_____  |
#        **  Simulation PASS!!     **    /^ ^ ^ \  |
#        **                        **   |^ ^ ^ ^ |w|
#        *************** ************    \m___m__|_|
#         student ID :
#
#
# ** Note: $finish    : C:/Users/user/Documents/graduatelevel/G1/CO2016/CO_2016_hw4/Cache/testfixture1.v(208)
#    Time: 3875 ns  Iteration: 0  Instance: /testfixture1
# 1
# Break in Module testfixture1 at C:/Users/user/Documents/graduatelevel/G1/CO2016/CO_2016_hw4/Cache/testfixture1.v line 208
```

Fig.3 Simulation Successful snapshot

3. Take snapshot
   I. Using waveform to verify the execute results.
   II. Please annotate the waveform (as shown in Fig. 4)

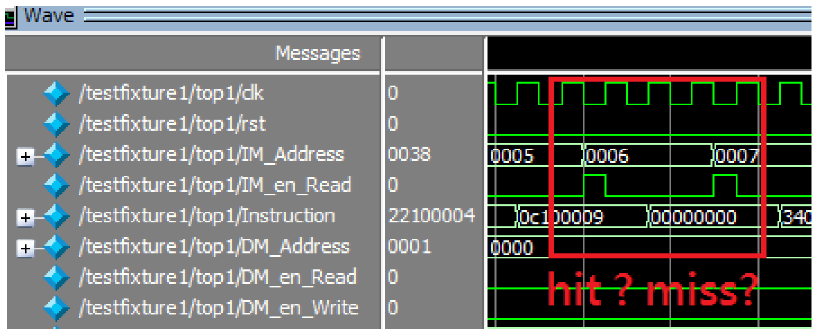

Wave
Messages
/testfixture1/top1/clk 0
/testfixture1/top1/rst 0
/testfixture1/top1/IM_Address 0038
/testfixture1/top1/IM_en_Read 0
/testfixture1/top1/Instruction 22100004
/testfixture1/top1/DM_Address 0001
/testfixture1/top1/DM_en_Read 0
/testfixture1/top1/DM_en_Write 0
0005
0006
0007
0c100009
00000000
340
0000
hit ? miss?

Fig.4 waveform snapshot

   III. At least list the following situations in the waveform snapshot. Then explain them as detailed as possible.
      i. I-Cache Miss –take snapshot to one "cache miss" access, explain what cache does and how much cycles there are between this instruction access and next instruction access
      ii. I-Cache Hit –take snapshot to one "cache miss" access, explain what cache does and how much cycles there are between this instruction access and next instruction access
   IV. Optional bonus : Find other situations and take waveform snapshot, and explain how it works (e.g. D-cache write miss, the write through mechanism)

4. Finish the Project Report.

TIPS

- **Please refer to the lab2 tutorial and build your project.**

Important

When you upload your file, please check you have satisfied all the homework requirements, including the **File hierarchy**, **Requirement file** and **Report format**.

If you have any questions, please contact us.