CA2020 Project 2 Report

B07902001 徐維謙
B07902075 林楷恩
B07902123 蔡奇夆

Module Explanation

Dcache Controller

Interface to CPU

Inputs

- **cpu_data_i**: 32 bits, the word the CPU wants to write.
- **cpu_addr_i**: 32 bits, the address of the word that the CPU wants to read/write.
- **cpu_MemRead_i**: 1 bit, set to 1 if the CPU wants to read.
- **cpu_MemWrite_i**: 1 bit, set to 1 if the CPU wants to write.

Outputs

- **cpu_data_o**: 32 bits, the word returned to the CPU.
- **cpu_stall_o**: 1 bits, indicates if the CPU needs to stall to wait for the memory operations. Thus, its value is set to 1 when a data access from CPU causes cache miss.

Interface to SRAM

the inputs and outputs here are actually internal signals because **dcache_sram** is an internal unit of **dcache_controller**.

Inputs

- **sram_cache_tag**: 25 bits, containing the valid bit, the dirty bit, and the 23-bit tag. The controller uses the dirty bit to decide if **write back** should be performed. If the dirty bit is set, then it uses the tag part combined with the 4-bit index to know the memory address to be written.
- **sram_cache_data**: 256 bits, the referenced block if that block is in sram (cache hit).
- **hit**: 1 bit, set to 1 if the accessed data is in sram.

Outputs

- **cache_sram_index**: 4 bits, the cache index of the block, which is the 5th to 8th bits of the address.
- **cache_sram_tag**: 25 bits, containing the valid bit, the dirty bit, and the 23-bit tag. The valid bit is always 1 because the block to be written is either from memory or CPU. The dirty bit is set to 1 if **write hit** happens.
- **cache_sram_data**: 256 bits, the block to be written to dcache_sram. When **write hit** happens, this block is set to the block read from dcache_sram with the target word modified. When the memory read is done, this block is set to the memory's data output. Since there are only 2 cases that this block is effective, we just distinguish them by the **hit** signal returned by the dcache_sram. If **hit** is 1, then we regards it as the first case.
- **cache_sram_enable**: 1 bit, set to 1 if the controller wants to read/write to dcache_sram.
- **cache_sram_write**: 1 bit, set to 1 if the controller wants to write to dcache_sram.

Interface to Memory

Inputs

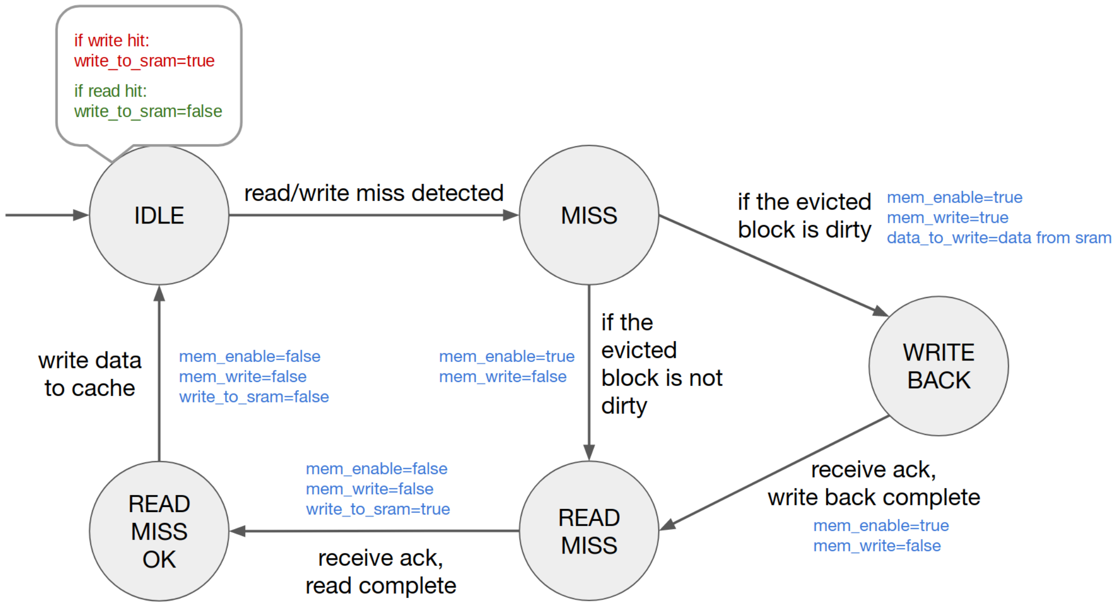
- **mem_data_i**: 256 bits, the block returned by the memory
- **mem_ack_i**: 1 bits, set to 1 if the last read/write is completed (only last 1 cycle)

Outputs

- **mem_data_o**: 256 bits, the block to be written to the memory
- **mem_addr_o**: 32 bits, address of the block to be read/write
- **mem_enable_o**: 1 bit, set to 1 if we want memory to read/write data
- **mem_write_o**: 1 bit, set to 1 if we want memory to write data

Finite State Machine

In this graph, the black text on the arrows means event or condition and the blue text on the arrows means the action to do.



The Meaning of Each State

- IDLE: no memory operation in this state
- MISS: determine if we need to **write back**
- READMISS: wait for the memory to return the block we want to read
- WRITEBACK: wait for the memory to write the dirty block
- READMISSOK: write the block read from the memory to dcache_sram, which takes one cycle

What happens in dcache_controller in each scenario

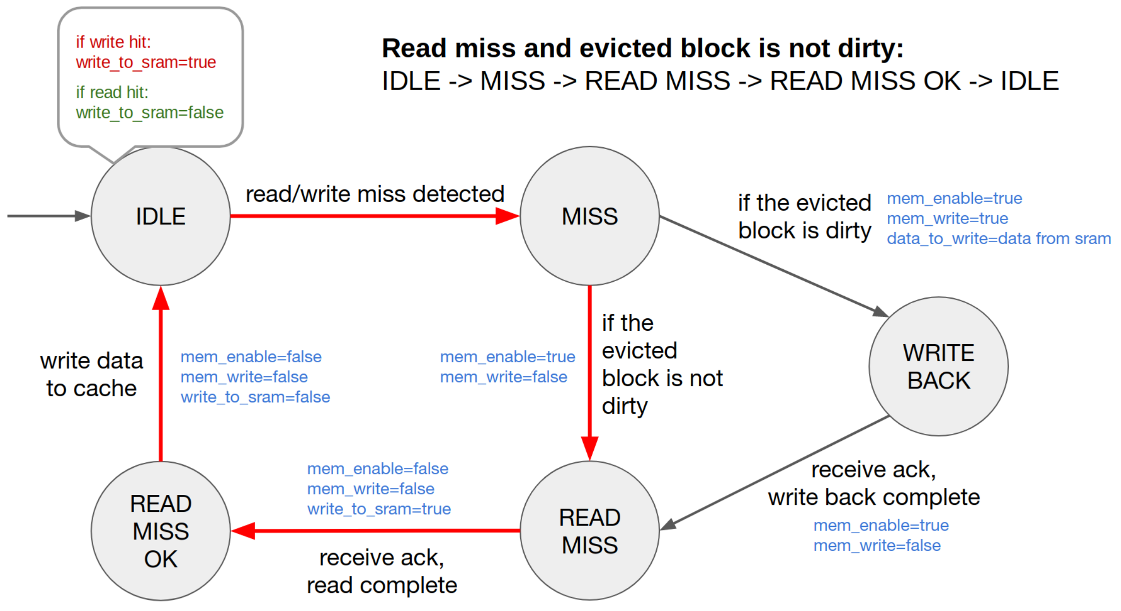
Read Hit

There is no state transition in this scenario. The controller directly forward the data fetched from dcache_sram to the CPU.

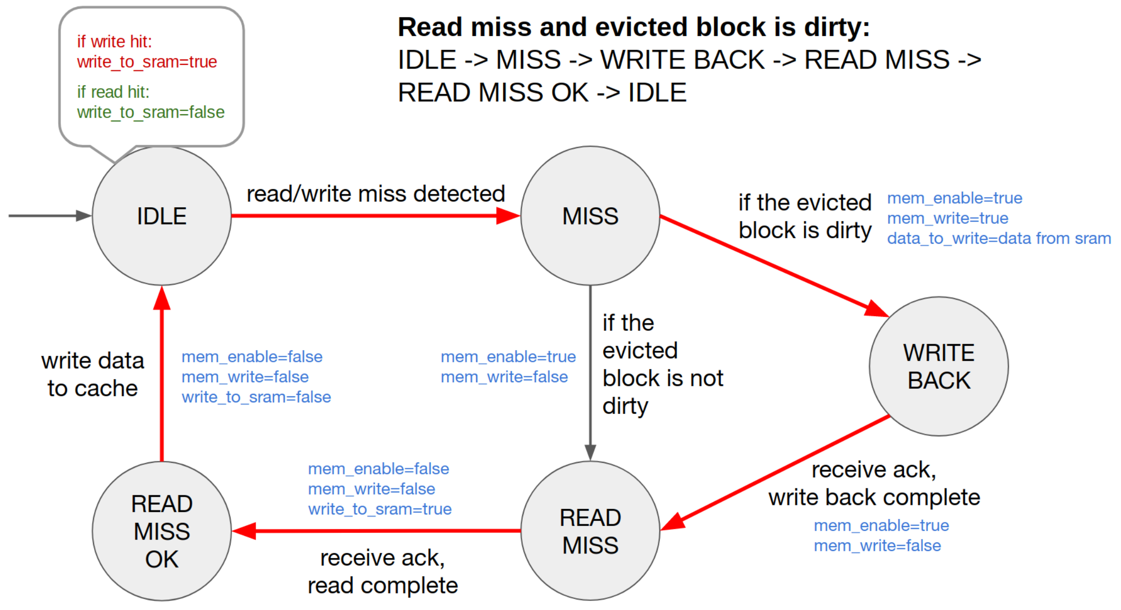
Write Hit

There is no state transition in this scenario. The controller changed the word at the specified offset in the block fetched from dcache_sram to the word given by the CPU and tells dcache_sram write this block in the next cycle.

Read Miss and the Evicted Block is Not Dirty or No Block is Evicted



Read Miss and the Evicted Block is Dirty

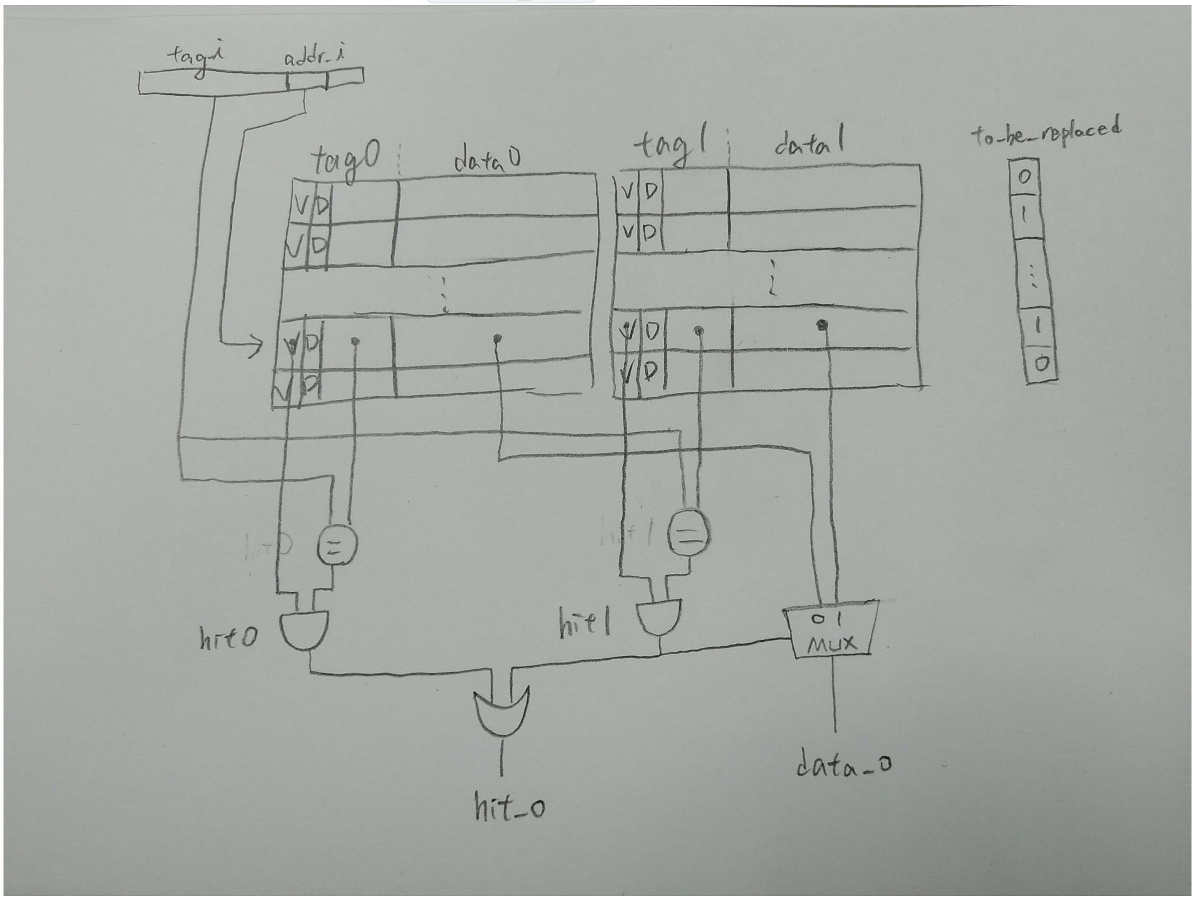


Write Miss

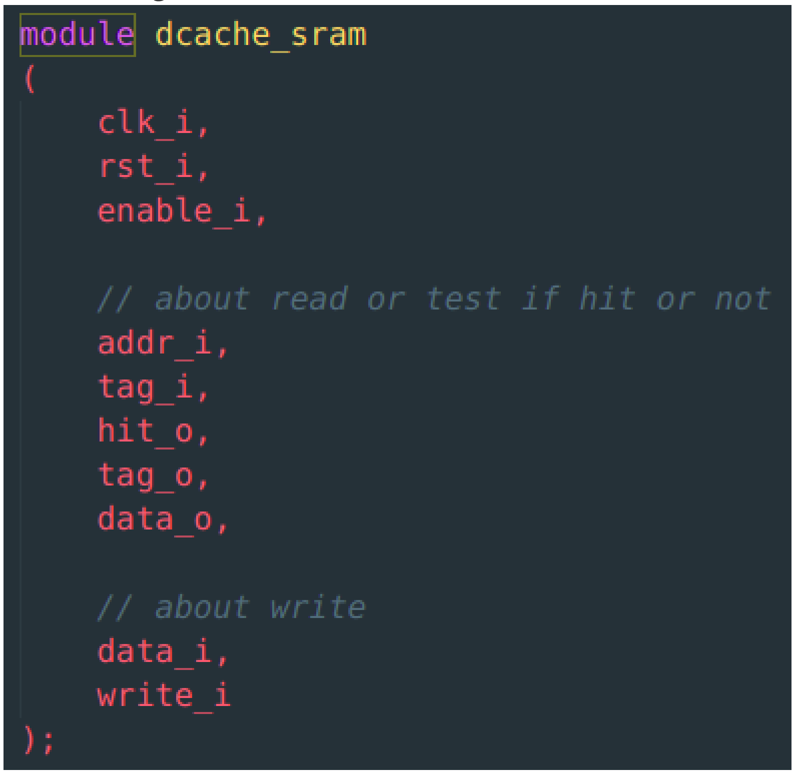
Write miss is not handled as a separated case. Instead, it is regarded as a combination of read miss and write hit. The controller first handles it as a read miss, then the controller will detect write hit event after the block fetched from the memory is written into the dcache_sram.

Dcache SRAM

The main internal structure of `dcache_sram` is as the below photo:



We categorize the I/O interface into "read-related", "write-related", and "other" classes.



```
module dcache_sram
(
    clk_i,
    rst_i,
    enable_i,

    // about read or test if hit or not
    addr_i,
    tag_i,
    hit_o,
    tag_o,
    data_o,

    // about write
    data_i,
    write_i
);
```

If `enable`, `dcache_sram` determines `data_o` based on input data, calculate `hit_o`, and output `data_o` according to the hit data.

If `write_i`, we forcefully update the cache content. The valid bit is always set to 1. The dirty bit is determine
On read hit or write data, we update the corresponding `to_be_replaced` to keep track of the least recently used entry of the cache.

`select` is an internal wire to determine which cache data should be output. If `hit_o`, it should output the hit data; otherwise, it should output the data to be replaced.

It is noteworthy that the logic of reading cache and writing cache has no conflict, since reading can be done within the same cycle and writing takes effect on the next cycle.

testbench

There is no midification in `testbench.v`. We initialize all the registers upon the `rst_i` signal.

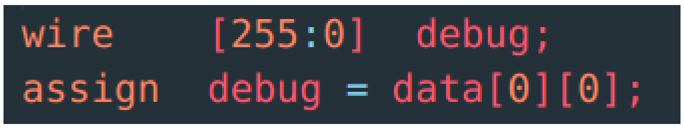
CPU

We replace `Data_Memory` with `dcache_controller` and add the input/output wire for communication with external `Data_Memory`.

Members & Teamwork

Name	Student ID	Work
徐維謙	B07902001	discussion
林楷恩	B07902075	`dcache_controler`, discussion, debug, report
蔡奇夆	B07902123	`dcache_sram`, discussion, debug, report

Difficulties Encountered and Solutions in This Project

- Array of registers is not recorded, so `gtkwave` will not show the value of the content in the cache. To know the value for easier debugging, we intentionally construct a wire connected to the value we are interested in. Then, we can examine the exact value of cache content.


```
wire    [255:0]  debug;
assign  debug = data[0][0];
```

Development Environment

- OS: Ubuntu 20.04
- Compiler: `iverilog 10.3 (stable)`