

Project #6: Cache Implementation

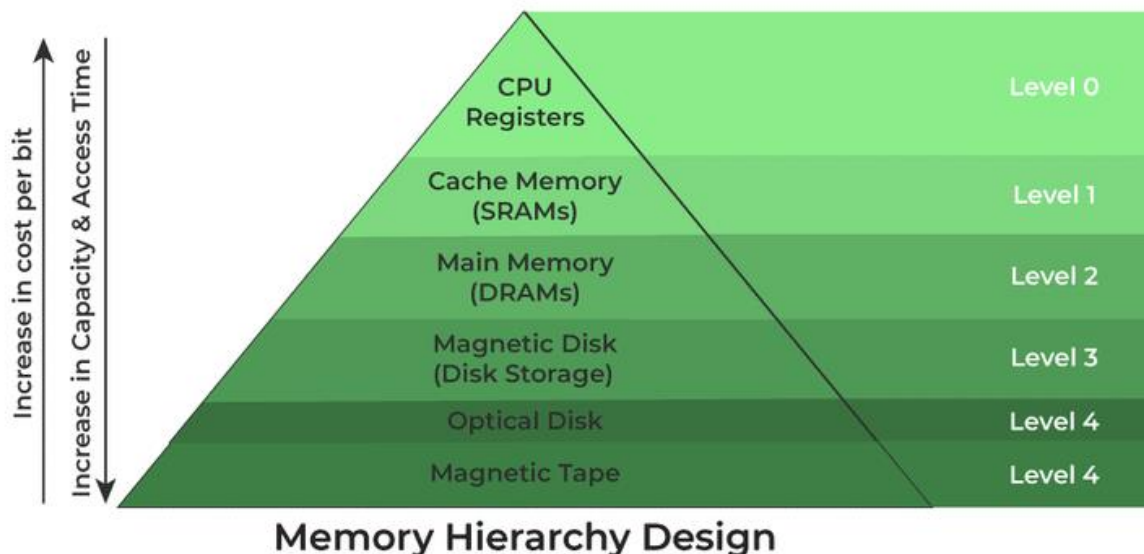
ECE 552: Fall 2025

Project Introduction

In this project phase, you will create a finite state machine for a cache system, which will be integrated into your processor that you completed in previous phases. This is the **final required project phase for ECE 552**.

Implementations must be created using **Verilog**; **SystemVerilog will not be allowed and will result in a grade of 0 for Problem 2 if used**. This assignment is designed to be completed **in a group of 2-4**. Collaboration between groups is not permitted; we will perform code comparisons between groups.

You are **permitted** to use generative artificial intelligence (e.g. ChatGPT); if you choose to do so, please indicate that in your submission. Please ensure your usage of LLMs adheres to the course policies.



Problem 1: Cache FSM Schematic

You will be designing a two-way set-associative cache with a **Least Recently Used** replacement policy. That is, when determining which cache line to evict, you will evict the line from the set that was the least recently used. This means that on cache reads and writes, you must update a variable to keep track of which line was not accessed last.

Please read the comments in the `cache.v` interface. The parameters of the cache are as follows:

- **16-byte** cache lines (4 words * 32 bits per word)
- **32 sets**
- **2-way set associative** with **least recently used** replacement
- **write-through**: writes immediately appear in the backing memory
- **write-allocate**: write misses populate the respective cache line

Please draw the (Mealy) FSM for your cache system. Draw each of the states of the FSM, add the transition signals, and denote the output of each stage. There is an example of a cache FSM in the textbook.

Problem 2: Cache FSM Implementation

1. Using the provided `cache.v` interface, finish the cache system in Verilog. **Please read the comments in the `cache.v` interface.**
2. Verify your cache on Gradescope. If the cache test does not pass, you will not be able to pass any of the other processor integration tests.

Problem 3: Cache Integration

You will now integrate the cache system into your processor. Update your pipeline to be able to handle arbitrary stalls from memory. **Please read the comments in the cache.v interface.**

1. When the “o_busy” signal from your instruction cache is driven high, your pipeline should not issue any more instructions until one is available. The remaining instructions in the pipeline should continue to be executed.
2. When the “o_busy” signal from your data cache is driven high, your pipeline should stall; instructions should continue to execute in-order but should wait until data is present at data memory before moving to the next pipeline stage.

You will notice a substantial increase in CPI after implementing the cache into your processor; this is expected. You are still expected to fall within a certain tolerance of the CPI of the reference implementation.

Once you have finished this step, you are done with the requirements for the project for ECE 552. Additional extra credit will be available in Project 7, which is optional.

Submission

Submit the following files to the appropriate **GRADESCOPE** assignment:

Deliverable	Points	Notes
schematic.pdf	20 (10%)	See problem 1.
cache.v	50 (25%)	See problem 2.
All Verilog files required for your processor to run	130 (65%)	See problem 3.
project6.txt	0	Submission Template <i>[Group Member 1]: Name</i> <i>[Group Member 2]: Name</i> ... ai_statement
FPGA Extra Credit	20 (10%)	Contact TAs for details; NOT REQUIRED

For information about how to submit as a group on Gradescope, see [this](#). Only one member of your group needs to submit this assignment.

If you **have used LLMs**, include the following in project6.txt:

We certify that our usage of LLMs is consistent with UW-Madison's academic integrity policies.

Otherwise, include the following:

We certify that we have not used LLMs to complete this assignment in any shape or form.

```
module ai_statement (input wire used_llms, output reg [31:0] ai_statement);
  always @(*) begin
    if (used_llms)
      ai_statement = "I certify that my usage of LLMs is consistent with UW-Madison's academic integrity policies.";
    else
      ai_statement = "I certify that I have not used LLMs to complete this assignment in any shape or form.";
    end
endmodule
```

The results of your submission will be made available after you submit via Gradescope and Canvas. You may submit as many times as you would like before the deadline.