**Project 2: Direct Mapped Cache**

**Project Report**

Team 3:  
Sam Lee, Miguel Tirado, Jesus Nunez

Muhammad Mujeeb, Angel Smith-Evans

Dr. Jing Pang

CPE 186: Computer Hardware System Design  
Fall 2019

**Table of Contents**

[**1. Introduction**](#_k2ykc8mwns4q) **2**

[**2. Direct Mapped Cache**](#_tkd340yzed5) **2**

[**A. Design Purpose**](#_yibqh9tbtfgn) **2**

[**B. Directed Mapped Cache Schematic**](#_m8uru0awz2ra) **2**

[**C. Directed Mapped Cache Controller State Diagram**](#_lj2qj8lduqj) **3**

[**D. Verilog Design and Testbench**](#_6l9sbanqev7f) **4**

[**Verilog code:**](#_4b0xsc8aws7j) **5**

[**Testbench verilog code:**](#_8eeek5m1ezj3) **9**

[**E. Simulation Waveform**](#_1tt23pfg7drt) **9**

[**F. Results Discussion**](#_qq4itdywpgx2) **11**

[**3. Conclusion**](#_gwj0tn39wos0) **11**

# 

# **1. Introduction**

A cache is a hardware or software that is used to store something, usually data, temporarily in a computing environment. A small amount of faster, more expensive memory is used to improve the performance of recently accessed or frequently accessed data that is stored temporarily in a rapidly accessible storage media that's local to the cache client and separate from bulk storage. The cache is frequently used by cache clients, such as the CPU, applications, web browsers or operating systems. The cache is used because main storage can't keep up with the demands of the cache clients. Cache shortens data access times, reduces latency and improves input/output. Because almost all application workloads depend on I/O operations, caching improves application performance. In this project, our team attempted to implement a virtual cache using verilog hardware coding language.

# **2. Direct Mapped Cache**

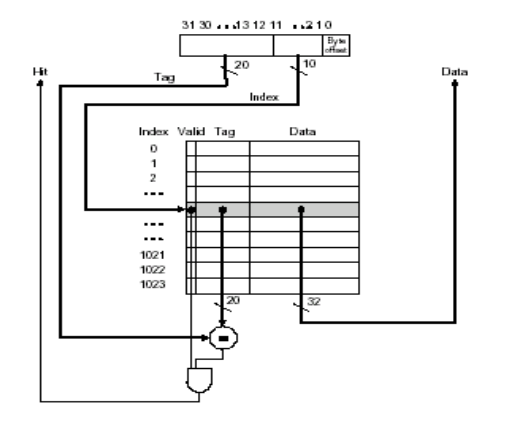
## **A. Design Purpose**

In our cache design we have several inputs and outputs in order for the cache to function correctly with the CPU and main memory. Those inputs and outputs were: caddress, cstrobe, crw, cready, cdata, maddress, mstrobe, mrw, mdata, and mready. The implementation of this design was inspired by the lectures given by Dr. Pang in the CPE186 class.

## 

## **B. Directed Mapped Cache Schematic**

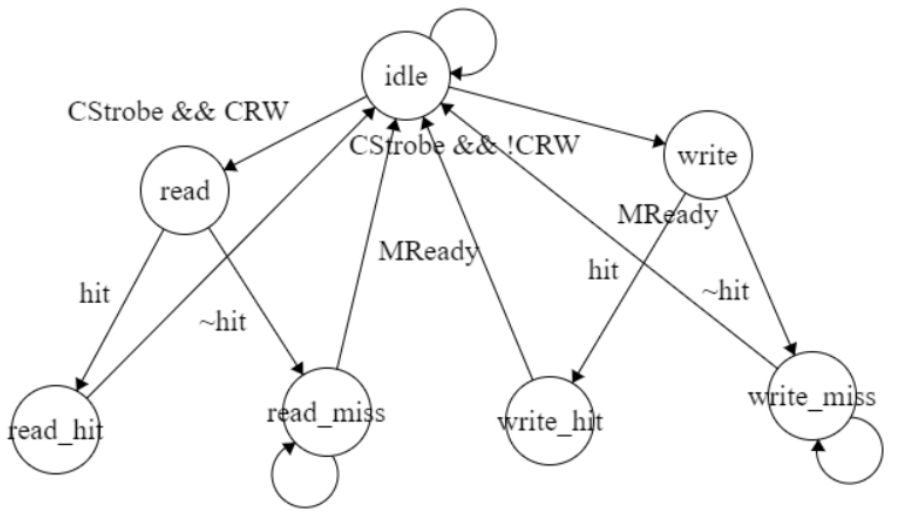
Introduced below is the Directly Mapped Cache schematic that was lectured on in our class CPE 186 by Dr. Pang. This schematic was pulled from her lecture slides. It shows how a data address is used to map values to the cache and how the cache has wires which are wired to the memory to validate “hits” and map “data” in other memory.



*Fig 1. The Direct Mapping Cache Schematic from Lecture Slides*

## **C. Directed Mapped Cache Controller State Diagram**

For our state machine we have six states: idle, read hit, read miss, write hit, write miss, and write memory. In the idle state, no memory access is underway. In the read hit state, read access is satisfied from the cache during the cycle and control returns to the idle state at the next transition. In the read miss state, wait state counter is loaded to time the wait for completion of the main memory access. In the write hit state, cache has been hit on the write operation. In the write miss state, cache has been missed on a write operation. In the write memory state, main memory write is in progress.



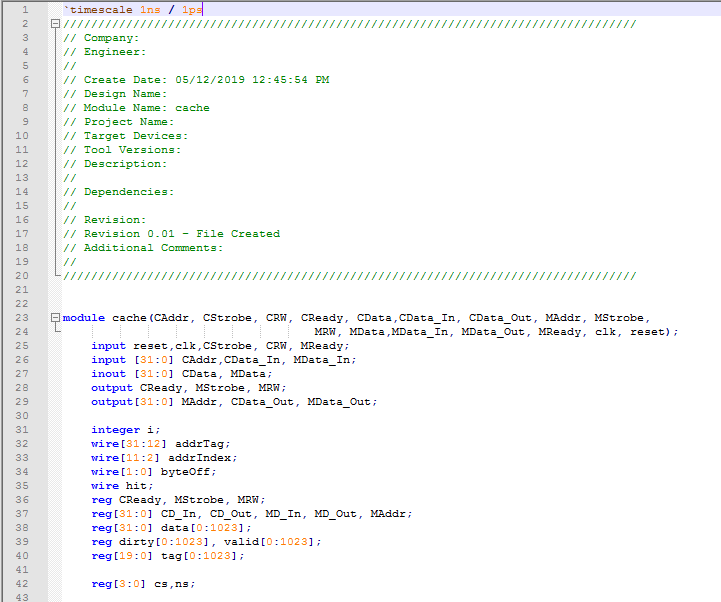
In this finite state machine above, if there is a hit then there is a read and write, and if not then there is a miss in rad and write. After going through those states in retunrs to idle state.

## 

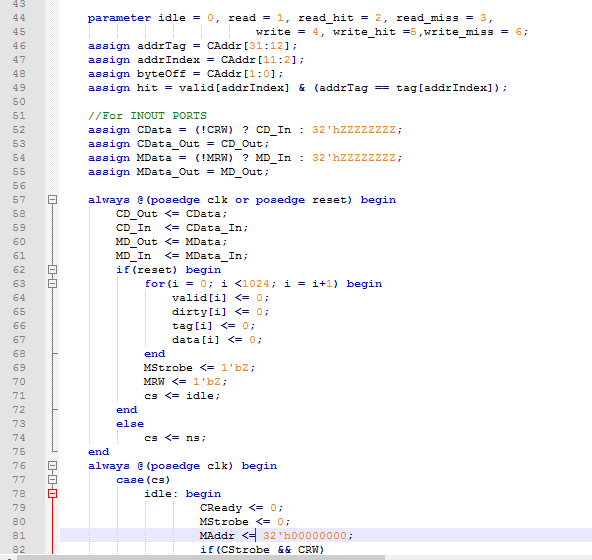
## **D. Verilog Design and Testbench**

We will present our code below as screenshots from Notepad++ to display an readable version with automated color typing.

### Verilog code:

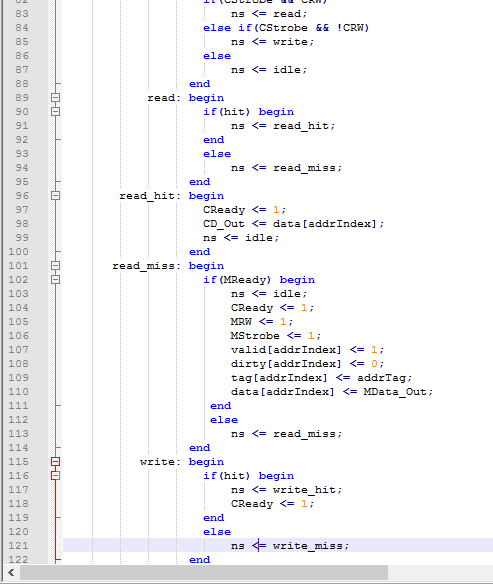


*Fig 2. Verilog Code part 1*

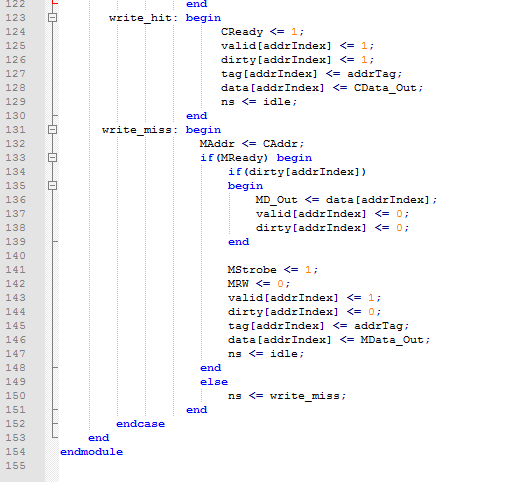


*Fig 3. Verilog code*

*part 2*



*Fig 4. Verilog code part 3*



*Fig 5. Verilog code part 4*

### Testbench verilog code:

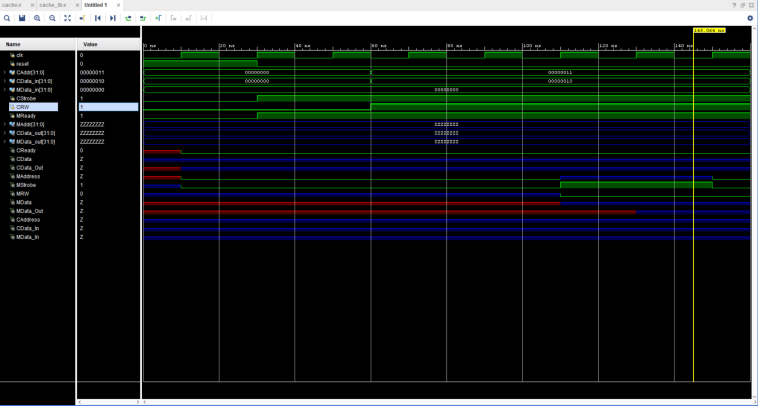
### 

*Fig 6. Verilog testbench code*

## **E. Simulation Waveform**

## 

*Fig 7. Simulation Waveform from 0ns to 60ns*



*Fig 8. Simulation Waveform from 60ns to 160ns*

## **F. Results Discussion**

As seen in the waveform above in *Fig 7* and *Fig 8,* the clock oscillates between 0 and 1 every 10ns, and reset go to 0 at 30ns. The signal CStrobe and Mready go to 1 at 30ns as reset toggles to 0. The other signals are set to null values of “ZZZZZZZZ”. The signal CRW toggles to 1 at 60ns. Our behavioural models the intention of the design, but without time to properly implement the design, our behavioural does not fully show a Directly Mapped Cache design.

# **3. Conclusion**

The final implementation of the cache design that our team came up with was close to the idea of the cache design. However, due to time constraints, our team was not able to implement the design fully and correctly in time, although our members tried really hard to put this together. Because it was the end of the semester, and we put a lot of effort into the PCI Arbiter’s design (Project 1), and we also were busy with other classes in our curriculum, we couldn’t dedicate enough time to a full implementation of the project. With this being the second and final project of the semester, it shows how much we have come to learn throughout the course and our growth as juniors going into senior design for the next semester. We have learned how the cache is able to read/write to and from the memory, and be able to check