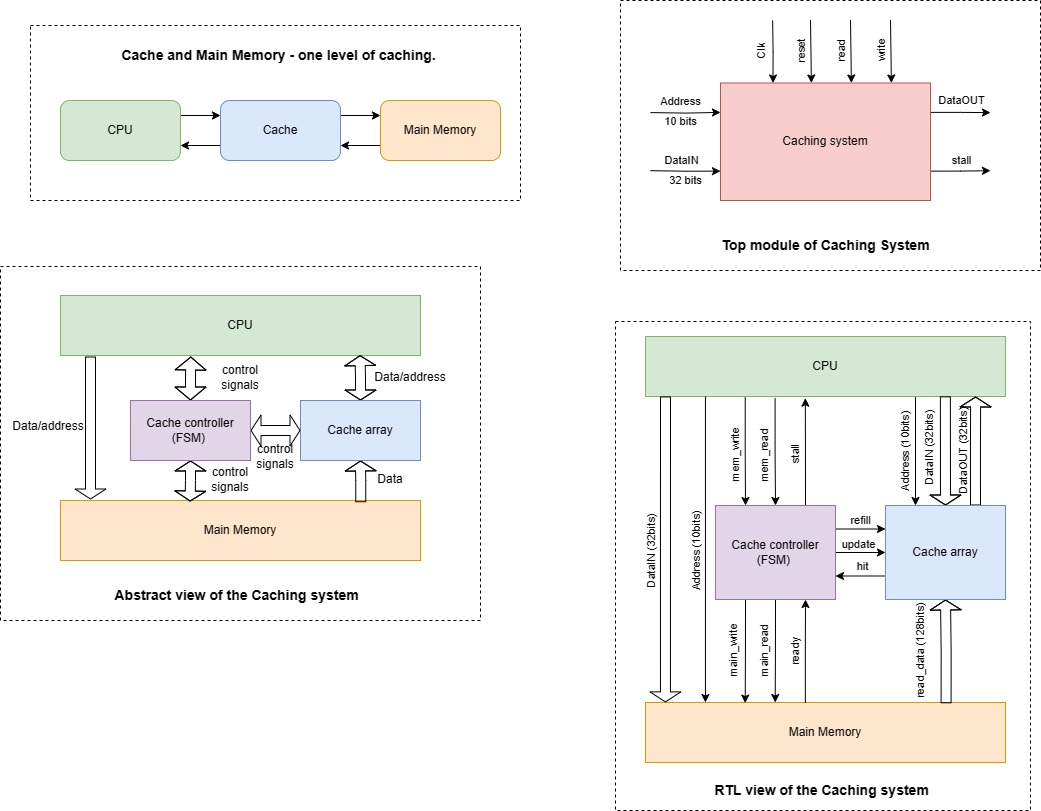
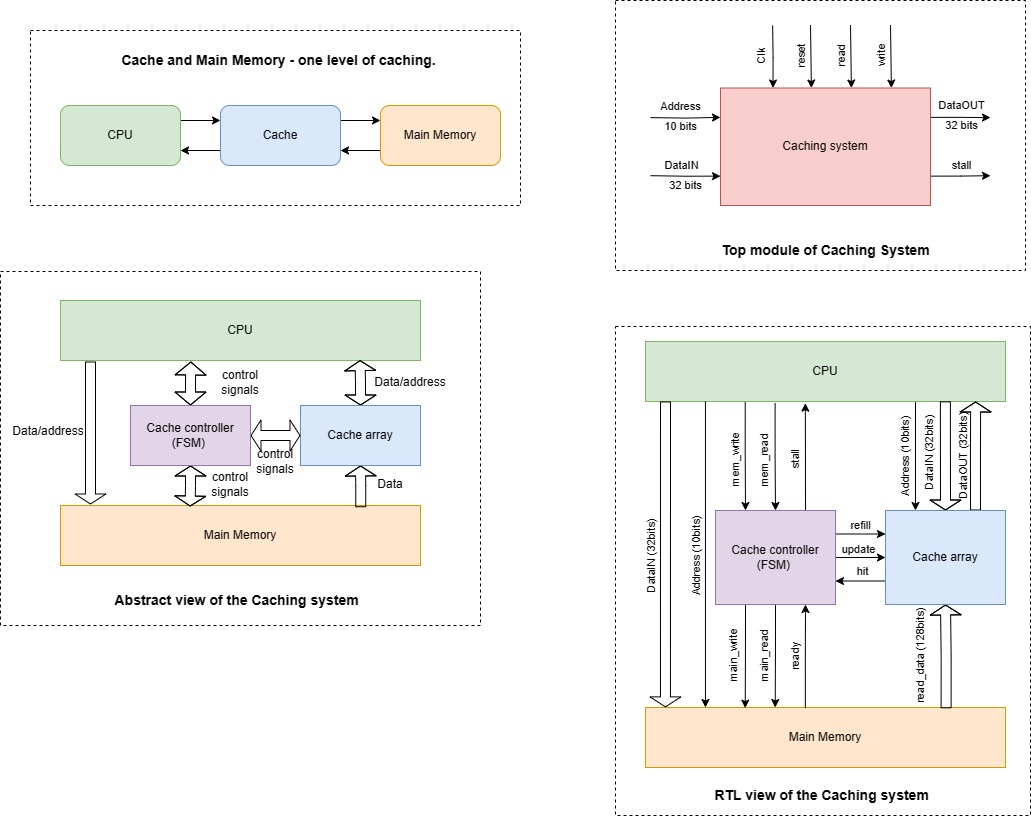
Cache Controller Implementation with Write-Through Policy

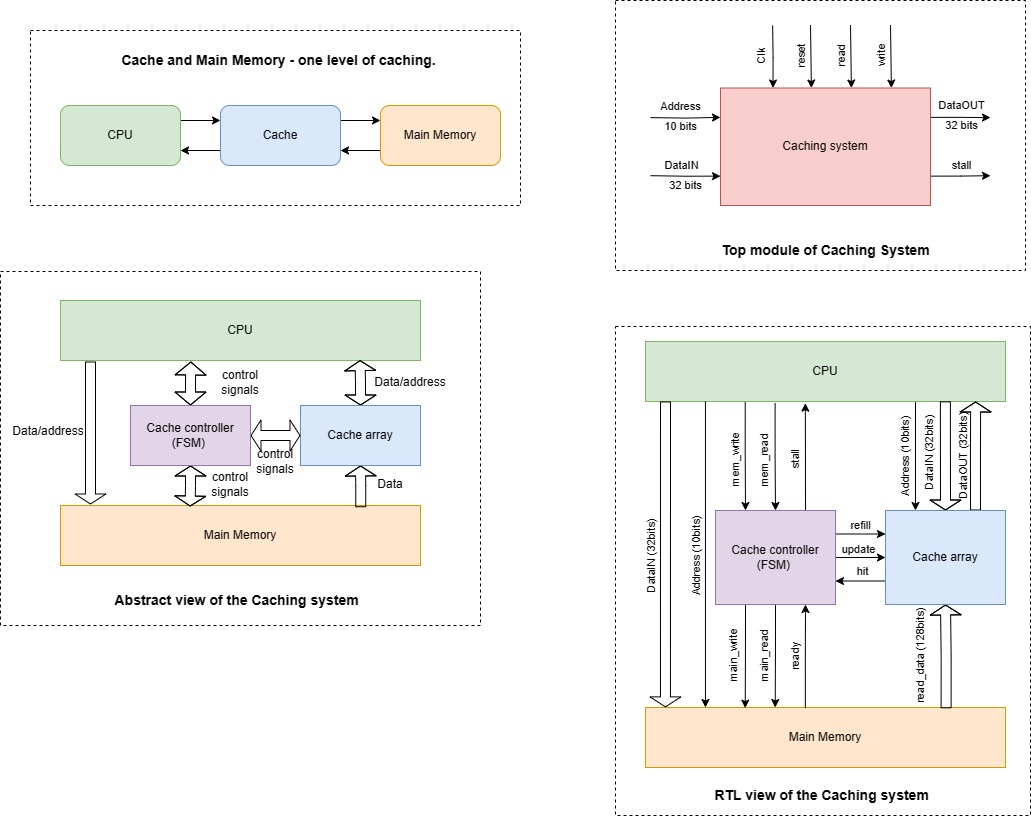
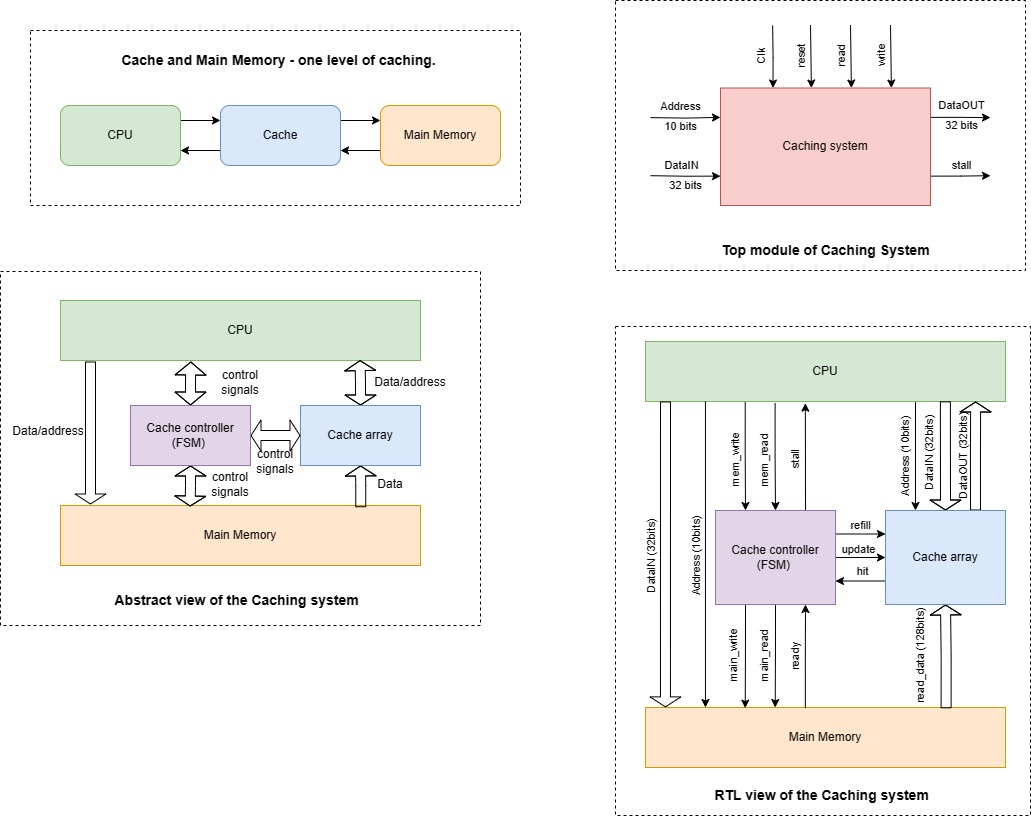
* *A RTL of a Direct Mapped Cache with main memory*

# Specifications

*  One level of caching between the CPU and main memory.
* The main memory module:
  + It has a capacity of 4 Kbytes (word addressable using 10 bits).
  + It is accessed (for read) takes 4 clock cycles and (for write) takes 1 cycle.
* The cache array module:
  + The total cache capacity is 512 bytes.
  + That each cache block is 16 bytes.
  + The cache has 32 blocks in total.
  + The cache uses direct mapping.
* The cache uses write-through for write-hit and write-around policies for write-miss.

## RTL Schematic

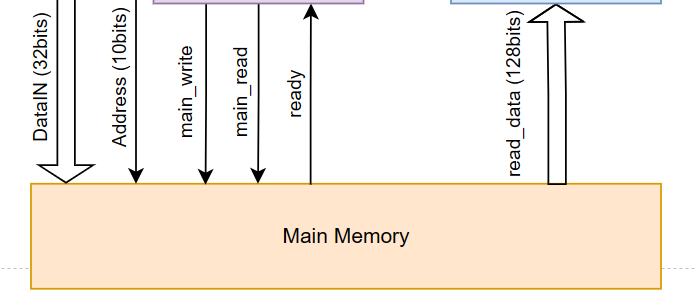
1. **The top module**



* **The ports of the top module**

|  |  |  |  |
| --- | --- | --- | --- |
| Signal | Direction | Bus width | Description |
| *Clk* | in | 1 bit | Clock input is the same as the Processor. |
| *reset* | in | 1 bit | Active-low Asynchronous Reset to reset all the states, signals, cache array, and main memory. |
| *read* | in | 1 bit | An active-high signal from the processor to read from the Cache. |
| *write* | in | 1 bit | An active-high Signal from the processor to write to the Cache and memory or only the memory based on the policy for writing. |
| *Address* | in | 10 bits | The requested address of the word addressable main memory. |
| *DataIN* | in | 4 bytes | The data of the requested address should be written to the Cache and memory or only the memory based on the policy for writing. |
| *DataOut* | Out | 4 bytes | The data of the requested address should be read from the cache. |
| *Stall* | Out | 1 bit | An active-high signal to stall the  processor on read/write misses  and write hits. |

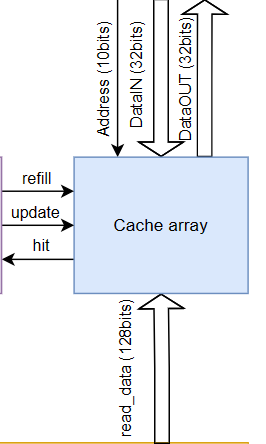
## 2) the main memory module



* **The ports of the main memory module**

|  |  |  |  |
| --- | --- | --- | --- |
| Signal | Direction | Bus width | Description |
| *Clk* | In | 1 bit | Clock input is the same as the Processor. |
| *reset* | In | 1 bit | Active-low Asynchronous Reset to reset the main memory. |
| *Main\_read* | In | 1 bit | An active-high signal from the FSM to read from the main memory and store a block in the cache. |
| *Main\_write* | In | 1 bit | An active-high Signal from the FSM to write to the main memory. |
| *Address* | In | 10 bits | The requested address of the word addressable main memory comes from the processor. |
| *DataIN* | in | 4 bytes | The data of the requested address should be written to the Cache and memory or only the memory based on the policy for writing. |
| *Ready* | out | 1 bit | An active-high signal from the Main Memory to acknowledge that the data is ready and the writing or reading is done. |
| *Read\_data* | Out | 16 bytes (a block) | A reading block from the main memory to be updated in the cache array. |

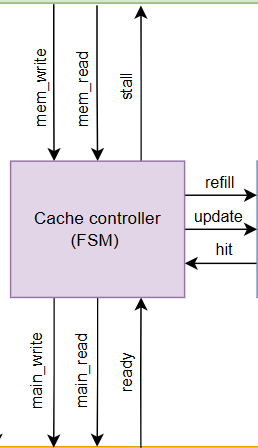
## 2) The cache array module



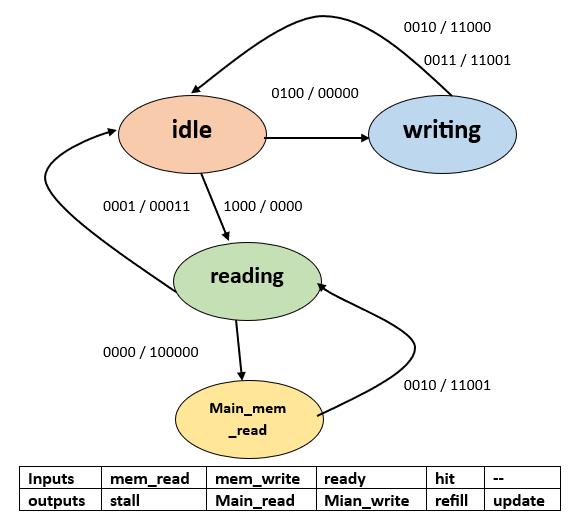
* **The ports of the cache array module**

|  |  |  |  |
| --- | --- | --- | --- |
| Signal | Direction | Bus width | Description |
| *Clk* | in | 1 bit | Clock input is the same as the Processor. |
| *reset* | in | 1 bit | Active-low Asynchronous Reset to reset all the states, signals, cache array, and main memory. |
| *Address* | in | 10 bits | The requested address of the word addressable main memory. |
| *Read\_data* | in | 16 bytes (a block) | A reading block from the main memory to be updated in the cache array. |
| *DataIN* | in | 4 bytes | The data of the requested address should be written to the Cache based on the write-through policy for write-hit. |
| *Update* | In | 1 bit | Active-high signal to Cache to update an entire Cache block using the data block read from Main Memory. |
| *Refill* | in | 1 bit | Active-high signal to Cache to refill the corresponding word on the Cache block, on a write hit. |
| *hit* | out | 1bit | Active-high signal to FSM to indicate the block of a specific address of main memory is stored in the cache. |
| *DataOut* | Out | 4 bytes | The data of the requested address should be read from the cache. |

## 3) the FSM module



* **The state diagram of FSM module**



* **The ports of the FSM module**

|  |  |  |  |
| --- | --- | --- | --- |
| Signal | Direction | Bus width | Description |
| *Clk* | in | 1 bit | Clock input is the same as the Processor. |
| *reset* | in | 1 bit | Active-low Asynchronous Reset to reset all the states, signals, cache array, and main memory. |
| *hit* | in | 1bit | An active-high signal from the cache array to indicate the block of a specific address of the main memory is stored in the cache. |
| *Ready* | in | 1 bit | An active-high signal from the Main Memory to acknowledge that the data is ready and the writing or reading is done. |
| *Mem\_read* | in | 1 bit | An active-high signal from the processor to read from the Cache. |
| *Mem\_write* | in | 1 bit | An active-high Signal from the processor to write to the Cache and memory or only the memory based on the policy for writing. |
| *Update* | Out | 1 bit | Active-high signal to Cache to update an entire Cache block using the data block read from Main Memory. |
| *Refill* | Out | 1 bit | Active-high signal to Cache to refill the corresponding word on the Cache block, on a write hit. |
| *Stall* | Out | 1 bit | An active-high signal to stall the  processor on read/write misses  and write hits. |
| *Main\_read* | Out | 1 bit | An active-high signal from the FSM to read from the main memory and store a block in the cache. |
| *Main\_write* | Out | 1 bit | An active-high Signal from the FSM to write to the main memory. |

## Test cases

**Waveforms**

**Code of Blocks**

## Top  level Architecture of Risc\_V with Cache memory

## Hit and Miss Latencies

* **Handling Read Hits and Misses**

All Read Hits are straight forward. No stalling on the processor for Read Hits. However, Read Misses stall the processor so as to fetch the requested data block from the Main Memory in the meantime. Fetching of the data block is followed by updating the Cache Line and then finally followed by a Read Hit to complete the read cycle. Read Miss has the most expensive penalty in the design.

* **Handling Write Hits and Misses**

All Write Hits/Misses stall the processor for same number of cycles. Hence, they are equivalent in the amount of penalty. This is because of the Write-Through Policy in the design that mandates writing to Main Memory on all Write Hits for Data Coherence/Consistency. Write-Around Policy is followed on Write Misses. The misses write the data directly to the Main Memory, without loading the data block to the Cache.

|  |  |  |
| --- | --- | --- |
| **Action** | **Latency (clock cycles)** | **Stall (clock cycles)** |
| Read Hit | 2 | - |
| Read Miss | 6 | 4 |
| Write Hit | 4 | 2 |
| Write Miss | 4 | 2 |

*Table. Hit time/ Miss penalties*

**Code of Blocks**

**Copyright Notice**

### Cache Controller. v. 1. 0 © 2019

All rights reserved to the Developer:

*Mitu Raj, iammituraj@gmail.com*