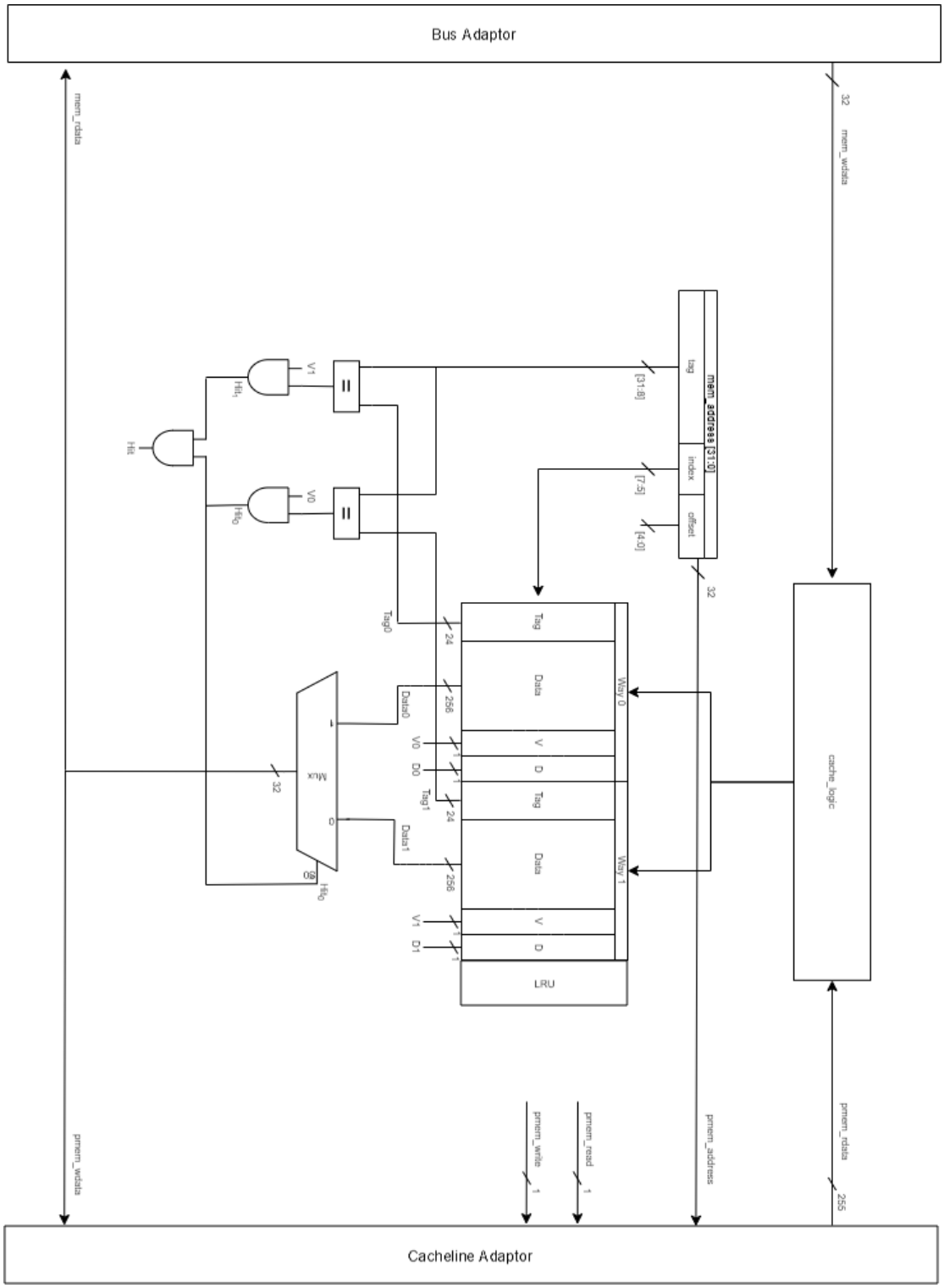
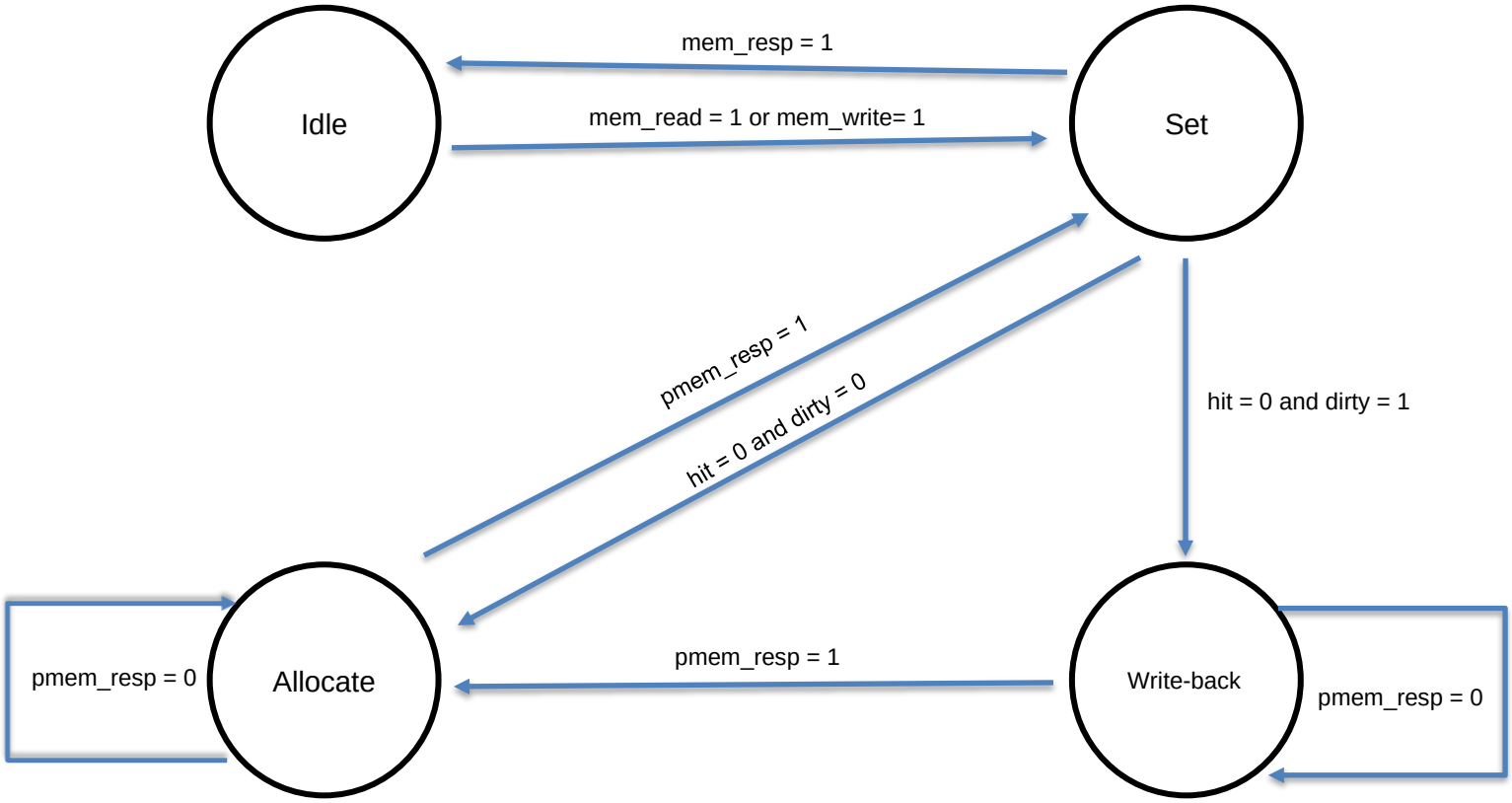
Datapath


Bus Adaptor
mem_rdata
32
mem_wdata
tag
mem_address [31:0]
index
offset
[31:8]
[7:5]
[4:0]
32
V1
V0
Hit1
Hit0
Hit
=
=
Tag0
24
Tag
Data
Way 0
Data0
256
V0
D0
Tag1
Data1
V1
D1
LRU
Way 1
V
D
MUX
32
cache_logic
pmem_wdata
pmem_write
pmem_read
pmem_address
pmem_rdata
255
Cacheline Adaptor

Controller State Machine



Controller Description

State Name	Description	Transition Conditions	Outputs
Idle	Wait state, waiting for a read or write request from the CPU	Set: cache ready signal set	None
Set	Sets valid and tag if tag is valid and there has been a hit	Idle: mem_read or mem_write = 1 Allocate: memory is ready to be interacted with	Valid set Hit set
Allocate	Read a new block from memory	Write-back: memory is ready to be interacted with Set: cache miss and clean block Allocate: pmem_resp = 0	Data read to cacheline Valid set Tag set Dirty Set
Write-back	Write block to memory	Set: cache miss and dirty block Write-back: pmem_resp = 0	pmem_write = 1

Testing Questions

1. Analyze your cache design to identify two edge cases you will deliberately test.

1) I will test to make sure that we are not able to read to the cache when the memory ready signal as not been set to high
2) I will test reading/evicting a value from the cache when it is not present in the cache.

2. Provide a brief description of how you will test one of your identified edge cases. This may be an English description or code, and may be risc-v assembly or cache input stimuli.

To test the first edge case, I will purposely set the pmem_resp to low and then try to read from the cache testing whether values are being outputted from mem_rdata from the bus adaptor.

3. Briefly describe how you will unit test your cache as the DUT itself, rather than as part of your processor.

I would feed inputs directly to the cache through the bus adaptor and sample signals through the outputs of the cache as well after running reads and writes. This way I can make sure the cache is working without relying on a functional CPU.