mp_cache

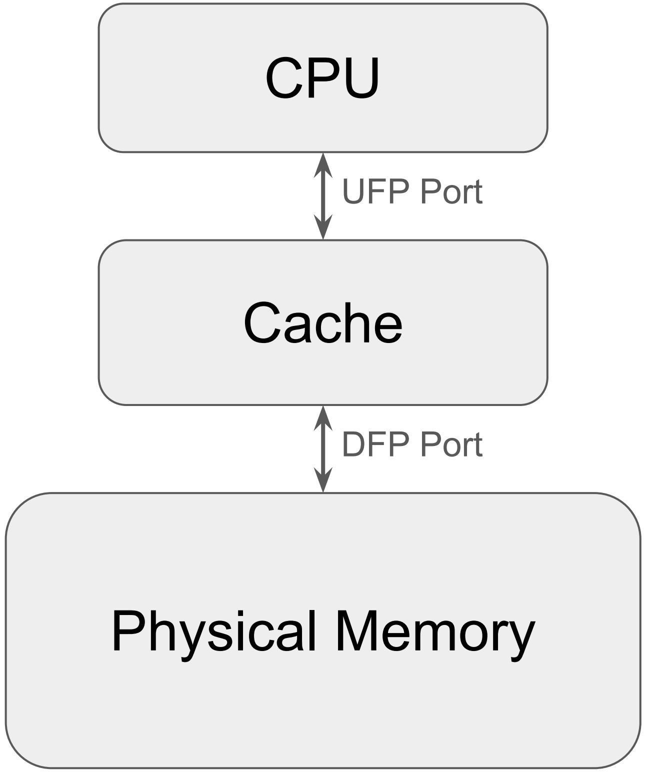
Part 2

Announcements and Reminders

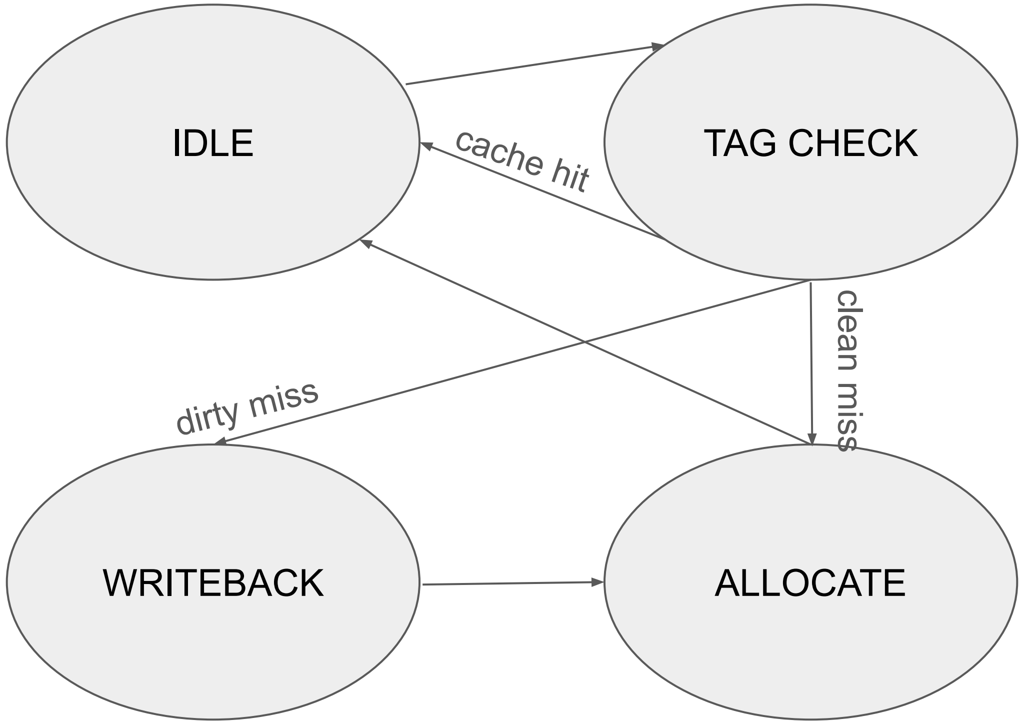
- Midterm 1
  - Regular (not conflict) time - **March 5th** (Tues) from **7:00pm - 10:00pm** in **ECEB 1013, 1015, 1017**
  - Review Session - March 3rd (Sun) from 12:30pm - 3:00pm in ECEB 1002
  - Good luck!
- mp_ooo group form
  - Fill out by March 6th (Wed)
  - Try not to use RISC or CACHE puns pls (for Stanley's sanity)
- mp_cache
  - Due on March 8th. First pre-deadline AG run on March 6th (Wed)

Quick Cache High level

- UFP Port
  - 32 bit data bus
  - 4 bit mask to control byte enable access
- DFP Port
  - 256 bit data bus
  - 1 bit read and write signal
- Non-pipelined (transaction stays till resp)
- PLRU Replacement Policy
- 4-ways, 16-sets, 256-bit cachelines



State Machine



Examples : Init

Suppose this is on reset for set = 1

UFP
- rmask = ‘0
- wmask = ‘0
- addr = ‘x
- wdata = ‘x

- rdata = ‘x
- resp = 0

DFP
- read = 0
- write = 0
- addr =’x
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	0	{‘x,x}	‘x
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 0th way
NEXT_PLRU = 0th way
STATE: idle
NEXT_STATE: idle

Examples : Transaction 1 Part 1

UFP

rmask = f
wmask = ‘0
addr = {A, 4’d1, 5’b00010}
wdata = ‘x

rdata = ‘x
resp = 0

DFP

read = 0
write = 0
addr =’x
wdata = ‘x

rdata = ‘x
resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	0	{‘x,x}	‘x
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 0th way
NEXT_PLRU = 0th way
STATE: idle
NEXT_STATE:

Examples : Transaction 1 Part 1

UFP
- rmask = f
- wmask = ‘0
- addr = {A, 4’d1, 5’b00100}
- wdata = ‘x

- rdata = ‘x
- resp = 0

DFP
- read = 0
- write = 0
- addr =’x
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	0	{‘x,x}	‘x
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 0th way
NEXT_PLRU = 0th way
STATE: idle
NEXT_STATE: tag_check

Examples : Transaction 1 Part 2

```
UFP
        rmask = f
        wmask = ‘0
        addr = {A, 4’d1, 5’b00100}
        wdata = ‘x

        rdata = ‘x
        resp = 0
DFP
        read = 0
        write = 0
        addr =’x
        wdata = ‘x

        rdata = ‘x
        resp = 0
```

WAY	VALID	{TAG,DIRTY}	DATA
0	0	{‘x,x}	‘x
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 0th way
NEXT_PLRU = 0th way
STATE: tag_check
NEXT_STATE:

Examples : Transaction 1 Part 2

UFP

- rmask = f
- wmask = ‘0
- addr = {A, 4’d1, 5’b00100}
- wdata = ‘x

- rdata = ‘x
- resp = 0

DFP

- read = 0
- write = 0
- addr =’x
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	0	{‘x,x}	‘x
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 0th way
NEXT_PLRU = 0th way
STATE: tag_check
NEXT_STATE: allocate

Examples : Transaction 1 Part 3

UFP

rmask = f
wmask = ‘0
addr = {A, 4’d1, 5’b00100}
wdata = ‘x

rdata = ‘x
resp = 0

DFP

read = 1
write = 0
addr ={A, 4’d1, 5’d0}
wdata = ‘x

rdata = ‘x
resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	0	{‘x,x}	‘x
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 0th way
NEXT_PLRU = 0th way
STATE: allocate
NEXT_STATE:

Examples : Transaction 1 Part 3

UFP

- rmask = f
- wmask = ‘0
- addr = {A, 4’d1, 5’b00100}
- wdata = ‘x

- rdata = ‘x
- resp = 0

DFP

- read = 1
- write = 0
- addr = {A, 4’d1, 5’d0}
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	0	{‘x,x}	‘x
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 0th way
NEXT_PLRU = 0th way
STATE: allocate
NEXT_STATE: allocate

10 cycles later (or more or less idk)

Examples : Transaction 1 Part 4

UFP

    rmask = f
    wmask = ‘0
    addr = {A, 4’d1, 5’b00100}
    wdata = ‘x

    rdata = ‘x
    resp = 0

DFP

    read = 1
    write = 0
    addr ={A, 4’d1, 5’d0}
    wdata = ‘x

    rdata = abcdef01abcdef01abcdef01abcdef01
    resp = 1

WAY	VALID	{TAG,DIRTY}	DATA
0			
1			
2			
3			

PLRU = 0th way
NEXT_PLRU = 0th way
STATE: allocate
NEXT_STATE:

Examples : Transaction 1 Part 4

UFP

- rmask = f
- wmask = ‘0
- addr = {A, 4’d1, 5’b00100}
- wdata = ‘x

- rdata = ‘x
- resp = 0

DFP

- read = 1
- write = 0
- addr ={A, 4’d1, 5’d0}
- wdata = ‘x

- rdata = abcdef01abcdef01abcdef01abcdef01
- resp = 1

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,0}	abcdef01abcdef01abcdef01 abcdef01
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 0th way
NEXT_PLRU = 0th way
STATE: allocate
NEXT_STATE:

Examples : Transaction 1 Part 4

UFP

- rmask = f
- wmask = ‘0
- addr = {A, 4’d1, 5’b00100}
- wdata = ‘x

- rdata = ‘x
- resp = 0

DFP

- read = 1
- write = 0
- addr ={A, 4’d1, 5’d0}
- wdata = ‘x

- rdata = M[A]
- resp = 1

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{‘x,x} {A,0}	‘x(abcdef01abcdef01abcdef01abcdef01)
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 0th way
NEXT_PLRU = 0th way
STATE: allocate
NEXT_STATE:

Data in SRAM array isn’t there yet due to non-transparent read

Examples : Transaction 1 Part 4

UFP

rmask = f
wmask = ‘0
addr = {A, 4’d1, 5’b00100}
wdata = ‘x

rdata = ‘x
resp = 0

DFP

read = 1
write = 0
addr ={A, 4’d1, 5’d0}
wdata = ‘x

rdata = M[A]
resp = 1

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{‘x,x} {A,0}	‘x(abcdef01abcdef01abcdef01abcdef01)
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 0th way
NEXT_PLRU = 0th way
STATE: allocate
NEXT_STATE: idle

Data in SRAM array isn’t there yet due to non-transparent read

Examples : Transaction 1 Part 5

UFP

- rmask = f
- wmask = ‘0
- addr = {A, 4’d1, 5’b00100}
- wdata = ‘x

- rdata = ‘x
- resp = 0

DFP

- read = 0
- write = 0
- addr =’x
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,0}	abcdef01abcdef01abcdef01 abcdef01
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 0th way
NEXT_PLRU = 0th way
STATE: idle
NEXT_STATE:

Examples : Transaction 1 Part 5

UFP

- rmask = f
- wmask = ‘0
- addr = {A, 4’d1, 5’b00100}
- wdata = ‘x

- rdata = ‘x
- resp = 0

DFP

- read = 0
- write = 0
- addr =’x
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,0}	abcdef01abcdef01abcdef01 abcdef01
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 0th way
NEXT_PLRU = 0th way
STATE: idle
NEXT_STATE: tag_check

Examples : Transaction 1 Part 6

UFP

- rmask = f
- wmask = ‘0
- addr = {A, 4’d1, 5’b00100}
- wdata = ‘x

- rdata = ‘x
- resp = 0

DFP

- read = 0
- write = 0
- addr =’x
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,0}	abcdef01abcdef01abcdef01 abcdef01
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 0th way
NEXT_PLRU = 0th way
STATE: tag_check
NEXT_STATE:

Examples : Transaction 1 Part 6

UFP
- rmask = f
- wmask = ‘0
- addr = {A, 4’d1, 5’b00100}
- wdata = ‘x

- rdata = abcdef01
- resp = 1

DFP
- read = 0
- write = 0
- addr =’x
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,0}	abcdef01abcdef01abcdef01 abcdef01
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 0th way
NEXT_PLRU =
STATE: tag_check
NEXT_STATE:

Examples : Transaction 1 Part 6

UFP

- rmask = f
- wmask = ‘0
- addr = {A, 4’d1, 5’b00100}
- wdata = ‘x

- rdata = abcdef01
- resp = 1

DFP

- read = 0
- write = 0
- addr =’x
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,0}	abcdef01abcdef01abcdef01 abcdef01
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 0th way
NEXT_PLRU = 2nd way
STATE: tag_check
NEXT_STATE:

Examples : Transaction 1 Part 6

UFP

- rmask = f
- wmask = ‘0
- addr = {A, 4’d1, 5’b00100}
- wdata = ‘x

- rdata = abcdef01
- resp = 1

DFP

- read = 0
- write = 0
- addr =’x
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,0}	abcdef01abcdef01abcdef01 abcdef01
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 0th way
NEXT_PLRU = 2nd way
STATE: tag_check
NEXT_STATE: = idle

Examples : Transaction 1 Part 7

UFP
- rmask = 0
- wmask = ‘0
- addr = ‘x
- wdata = ‘x

- rdata = ‘x
- resp = 0

DFP
- read = 0
- write = 0
- addr =’x
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,0}	abcdef01abcdef01abcdef01 abcdef01
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 2nd way
NEXT_PLRU = 2nd way
STATE: idle
NEXT_STATE: idle

Examples : Transaction 2 Part 1

UFP

- rmask = ‘0
- wmask = 4’b1100
- addr = {A, 4’d1, 5’b01100}
- wdata = 2’h2345xxxx

- rdata = ‘x
- resp = 0

DFP

- read = 0
- write = 0
- addr =’x
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,0}	abcdef01abcdef01abcdef01 abcdef01
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 2nd way
NEXT_PLRU = 2nd way
STATE: idle
NEXT_STATE:

Examples : Transaction 2 Part 2

UFP

- rmask = ‘0
- wmask = 4’b1100
- addr = {A, 4’d1, 5’b01100}
- wdata = 2’h2345xxxx

- rdata = ‘x
- resp = 0

DFP

- read = 0
- write = 0
- addr =’x
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,0}	abcdef01abcdef01abcdef01 abcdef01
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 2nd way
NEXT_PLRU = 2nd way
STATE: idle
NEXT_STATE: tag_check

Examples : Transaction 2 Part 3

UFP

- rmask = ‘0
- wmask = 4’b1100
- addr = {A, 4’d1, 5’b01100}
- wdata = 2’h2345xxxx

- rdata = ‘x
- resp = 1

DFP

- read = 0
- write = 0
- addr =’x
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,0(1)}	abcdef01abcd(2345)ef01ab cdef01abcdef01
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 2nd way
NEXT_PLRU = 2nd way
STATE: tag_check
NEXT_STATE: idle

Examples : Transaction 2 Part 4

UFP

- rmask = ‘0
- wmask = ‘0
- addr = ‘x
- wdata = ‘x

- rdata = ‘x
- resp = 0

DFP

- read = 0
- write = 0
- addr =’x
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,1}	abcdef012345ef01abcdef01 abcdef01
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 2nd way
NEXT_PLRU = 2nd way
STATE: idle
NEXT_STATE: idle

Examples : Transaction 3

Write to {B, 4’d1, 5’b00000}

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,1}	abcdef01abcdef01abcd2301 abcdef01
1	0	{‘x,x}	‘x
2	0	{‘x,x}	‘x
3	0	{‘x,x}	‘x

PLRU = 2nd way

Examples : Transaction 3

Write to {B, 4’d1, 5’b00000}

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,1}	abcdef01abcdef01abcd2301 abcdef01
1	0	{‘x,x}	‘x
2	1	{B,1}	M[B]
3	0	{‘x,x}	‘x

PLRU = 1st way

Examples : Transaction 4

Read from {C, 4’d1, 5’b00000}

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,1}	abcdef01abcdef01abcd2301 abcdef01
1	0	{‘x,x}	‘x
2	1	{B,1}	M[B]
3	0	{‘x,x}	‘x

PLRU = 1st way

Examples : Transaction 4

Read from {C, 4’d1, 5’b00000}

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,1}	abcdef01abcdef01abcd2301 abcdef01
1	1	{C,0}	M[C]
2	1	{B,1}	M[B]
3	0	{‘x,x}	‘x

PLRU = 3rd way

Examples : Transaction 5

Read from {D, 4’d1, 5’b00000}

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,1}	abcdef01abcdef01abcd2301 abcdef01
1	1	{C,0}	M[C]
2	1	{B,1}	M[B]
3	1	{D,0}	M[D]

PLRU = 0th way

Examples : Transaction 6

Read from {C, 4’d1, 5’b01000}

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,1}	abcdef01abcdef01abcd2301 abcdef01
1	1	{C,0}	M[C]
2	1	{B,1}	M[B]
3	1	{D,0}	M[D]

PLRU = 2nd way

Note: PLRU isn’t the LRU

Examples : Transaction 7 Part 1

UFP
- rmask = f
- wmask = ‘0
- addr = {E, 4’d1, 5b00000}
- wdata = ‘x

- rdata = ‘x
- resp = 0

DFP
- read = 0
- write = 0
- addr =’x
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,1}	abcdef01abcdef01abcd2301 abcdef01
1	1	{C,0}	M[C]
2	1	{B,1}	M[B]
3	1	{D,0}	M[D]

PLRU = 2nd way
NEXT_PLRU = 2nd way
STATE: idle
NEXT_STATE: tag_check

Examples : Transaction 7 Part 2

UFP

- rmask = f
- wmask = ‘0
- addr = {E, 4’d1, 5b00000}
- wdata = ‘x

- rdata = ‘x
- resp = 0

DFP

- read = 0
- write = 0
- addr =’x
- wdata = ‘x

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,1}	abcdef01abcdef01abcd2301 abcdef01
1	1	{C,0}	M[C]
2	1	{B,1}	M[B]
3	1	{D,0}	M[D]

PLRU = 2nd way
NEXT_PLRU = 2nd way
STATE: tag_check
NEXT_STATE: writeback

Examples : Transaction 7 Part 2

UFP
- rmask = f
- wmask = ‘0
- addr = {E, 4’d1, 5b00000}
- wdata = ‘x

- rdata = ‘x
- resp = 0

DFP
- read = 0
- write = 1
- addr = {B, 4’d1, 5’d0}
- wdata = M[B]

- rdata = ‘x
- resp = 0

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,1}	abcdef01abcdef01abcd2301 abcdef01
1	1	{C,0}	M[C]
2	1	{B,1}	M[B]
3	1	{D,0}	M[D]

PLRU = 2nd way
NEXT_PLRU = 2nd way
STATE: writeback
NEXT_STATE: writeback

10 cycles later (or more or less idk)

Examples : Transaction 7 Part 3

UFP

- rmask = f
- wmask = ‘0
- addr = {E, 4’d1, 5b00000}
- wdata = ‘x
- rdata = ‘x
- resp = 0

DFP

- read = 0
- write = 1
- addr = {B, 4’d1, 5’d0}
- wdata = M[B]
- rdata = ‘x
- resp = 1

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,1}	abcdef01abcdef01abcd2301 abcdef01
1	1	{C,0}	M[C]
2	1	{B,1}	M[B]
3	1	{D,0}	M[D]

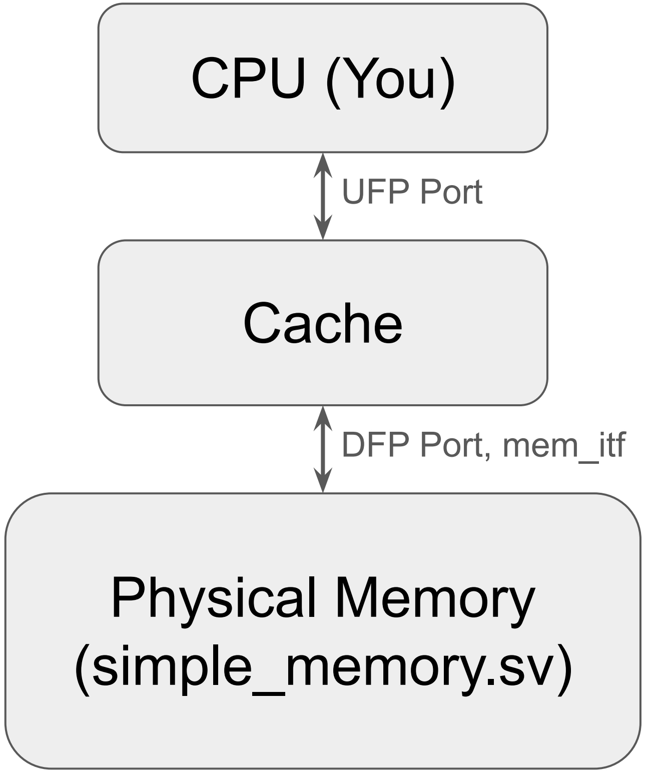
PLRU = 2nd way
NEXT_PLRU = 2nd way
STATE: writeback
NEXT_STATE: allocate

Examples : Transaction 7

WAY	VALID	{TAG,DIRTY}	DATA
0	1	{A,1}	abcdef01abcdef01abcd2301 abcdef01
1	1	{C,0}	M[C]
2	1	{E,0}	M[E]
3	1	{D,0}	M[D]

PLRU = 0th way

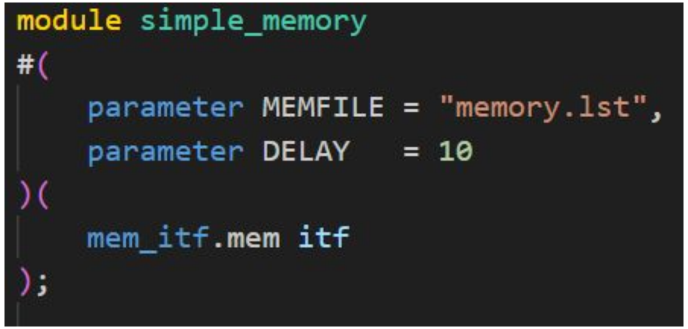
VERIFICATION :)


CPU (You)
UFP Port
Cache
DFP Port, mem_itf
Physical Memory (simple_memory.sv)

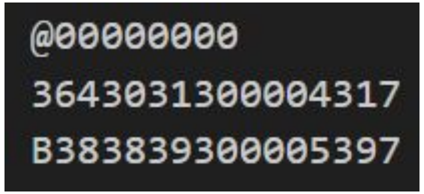
What is simple_memory.sv? not so simple :(

- Simulates physical memory
- mem_itf – Connection between DFP and physical memory
    - look at how top_tb.sv in mp_pipeline instantiates memory
- DELAY – delay for a physical memory response for a cache miss

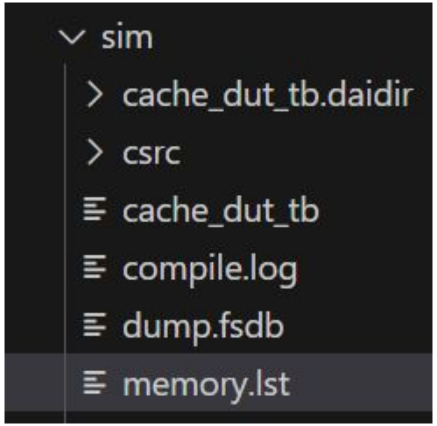
- MEMFILE – memory file that initializes the **internal_memory_array**
    - IEEE has a spec on how instantiate that memory ($readmemh())



```
module simple_memory
#(
    parameter MEMFILE = "memory.lst",
    parameter DELAY   = 10
)(
    mem_itf.mem itf
);
```



```
@00000000
3643031300004317
B383839300005397
```

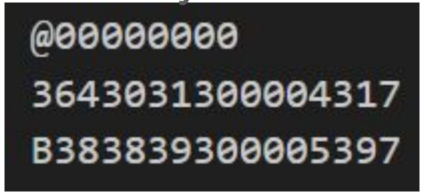


```
∨ sim
  > cache_dut_tb.daidir
  > csrc
    cache_dut_tb
    compile.log
    dump.fsdb
    memory.lst
```

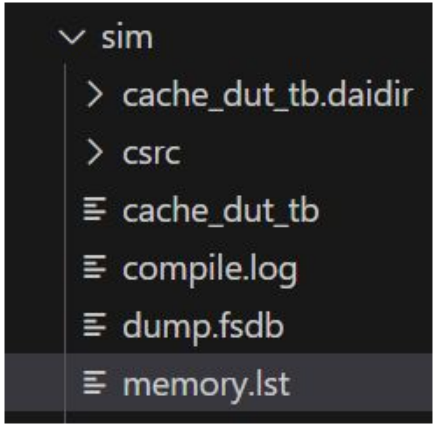
simple_memory.sv – MEMFILE

- @XXXXXXXX is the address
  - Address is cacheline addressable not byte addressable
- The text below it is the 256-bit cacheline in hex
- IEEE spec on $readmemh is good to look through

memory.lst

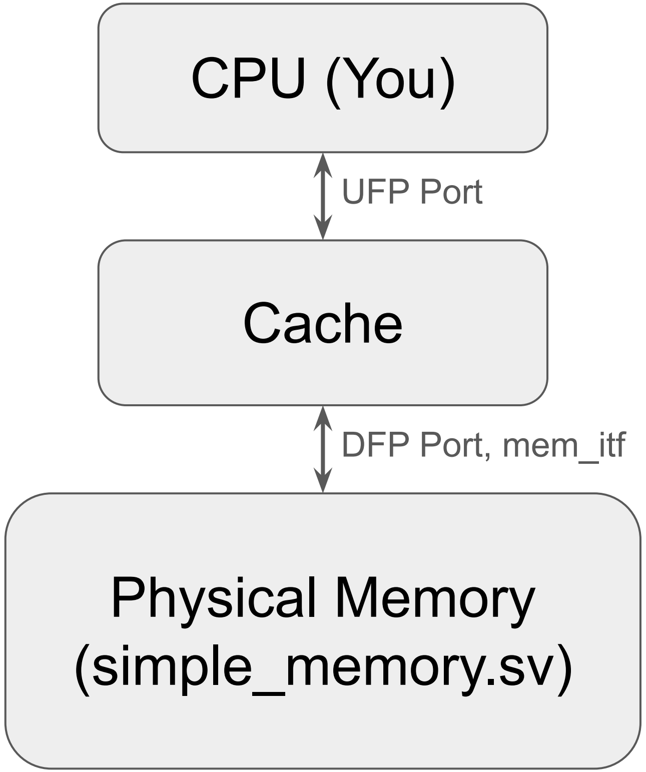


```
@00000000
3643031300004317
B383839300005397
```



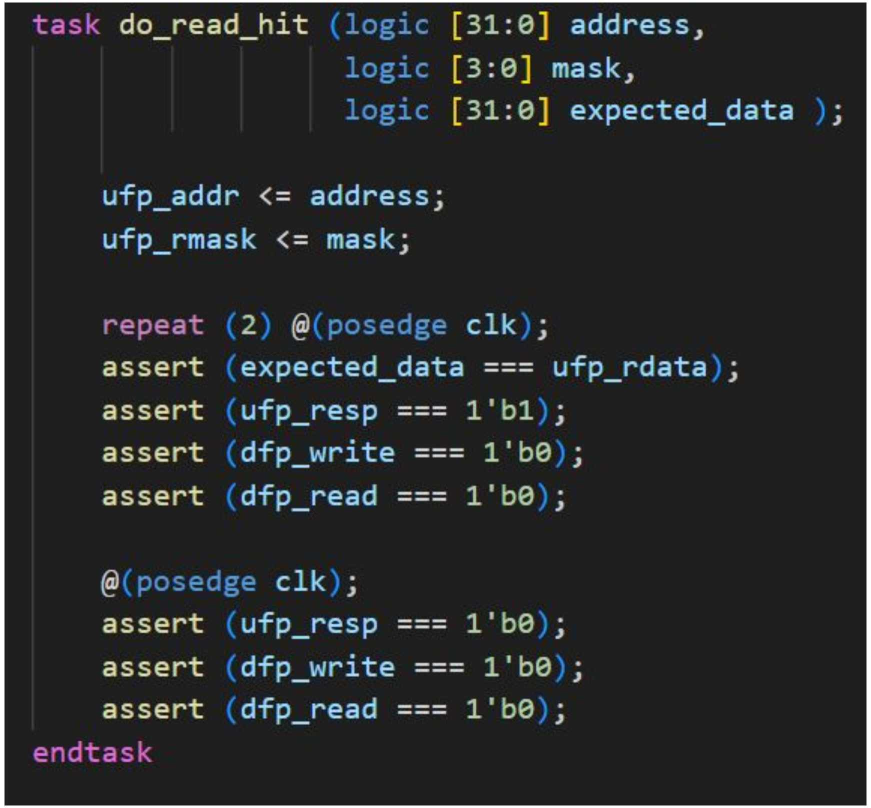
```
v sim
  > cache_dut_tb.daidir
  > csrc
    cache_dut_tb
    compile.log
    dump.fsdb
    memory.lst
```

- Keep the memory.lst file in mp_cache/sim/sim/
  - This will get removed in make clean
  - You might have to ‘make run_cache_dut_tb’ once to get that directory to show up


CPU (You)
UFP Port
Cache
DFP Port, mem_itf
Physical Memory
(simple_memory.sv)

Transaction tasks

- Have tasks for each type of transaction
  - read_hit, read_clean_miss, write_clean_miss, etc…
  - Check out mp_verif for guide on how to model these
- Know what the expected behavior for your transaction is before you call these tasks
  - This means expected data output, timing, etc
  - Be rigorous in what you check, make sure you don’t get a hit response on a miss, check the cycle after
- How do you keep track of expected data?
  - HVL Model Cache

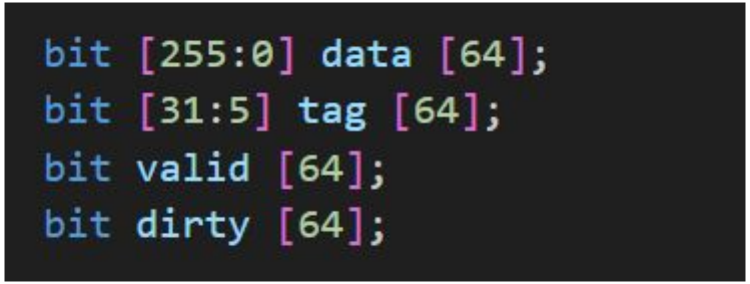


```
task do_read_hit (logic [31:0] address,
                  logic [3:0] mask,
                  logic [31:0] expected_data );

    ufp_addr <= address;
    ufp_rmask <= mask;

    repeat (2) @(posedge clk);
    assert (expected_data === ufp_rdata);
    assert (ufp_resp === 1'b1);
    assert (dfp_write === 1'b0);
    assert (dfp_read === 1'b0);

    @(posedge clk);
    assert (ufp_resp === 1'b0);
    assert (dfp_write === 1'b0);
    assert (dfp_read === 1'b0);
endtask
```

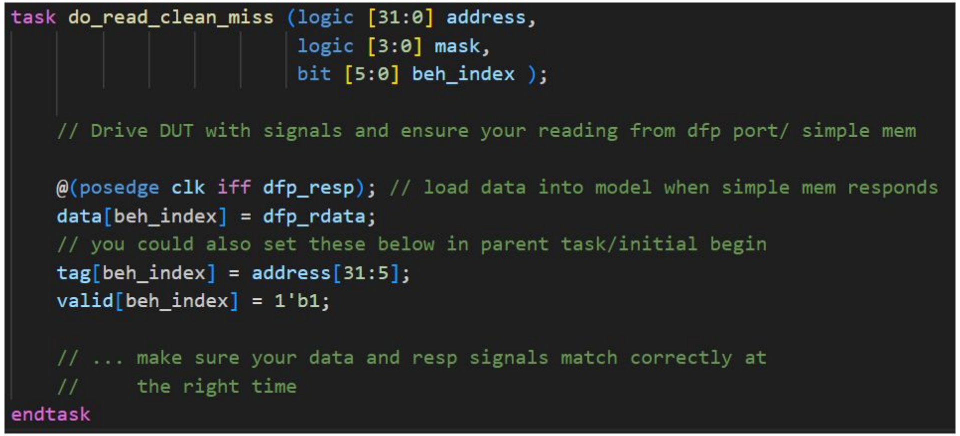
Behavioral HVL model

- Does not have to model 4-way 16-set associative
  - Unless you want it to :)
- Used to keep track of data in cache
- Remember to initialize data since bit type

- Many different ways to use in order to test different functionalities
  - Start with small cases then slowly expand



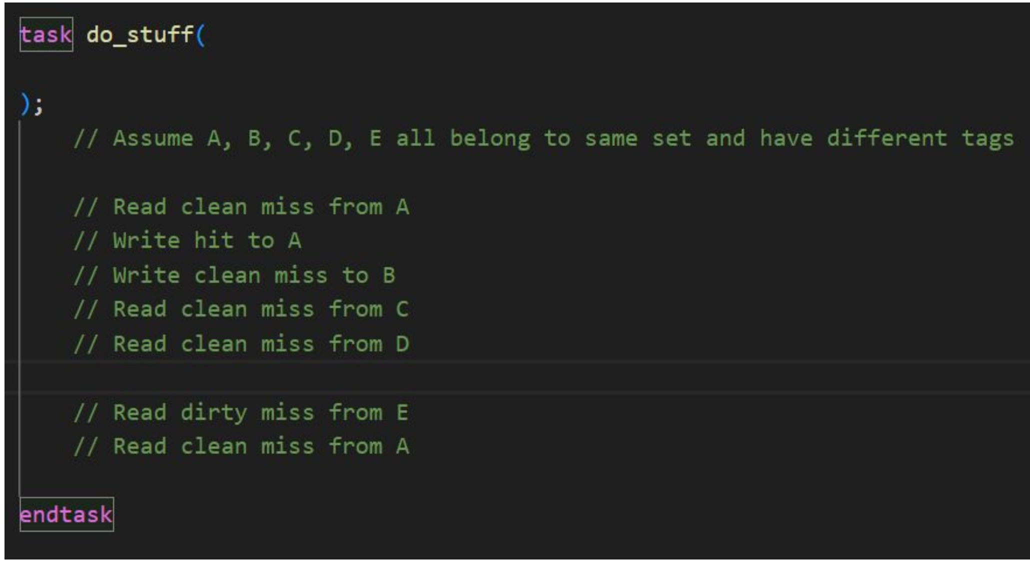
```
bit [255:0] data [64];
bit [31:5] tag [64];
bit valid [64];
bit dirty [64];
```



```
task do_read_clean_miss (logic [31:0] address,
                          logic [3:0] mask,
                          bit [5:0] beh_index );

    // Drive DUT with signals and ensure your reading from dfp port/ simple mem

    @(posedge clk iff dfp_resp); // load data into model when simple mem responds
    data[beh_index] = dfp_rdata;
    // you could also set these below in parent task/initial begin
    tag[beh_index] = address[31:5];
    valid[beh_index] = 1'b1;

    // ... make sure your data and resp signals match correctly at
    //     the right time
endtask
```

Targeted test cases

- Use the transaction tasks for building blocks to test different properties
- Add covergroups to ensure certain targeted testcases hit certain covergroups
  - All sets/all ways ?
- If a certain property is difficult to verify, use targeted testcases to simulate + verdi to check
  - PLRU ?



```
task do_stuff(

);
    // Assume A, B, C, D, E all belong to same set and have different tags

    // Read clean miss from A
    // Write hit to A
    // Write clean miss to B
    // Read clean miss from C
    // Read clean miss from D

    // Read dirty miss from E
    // Read clean miss from A

endtask
```