# State machine for the caches

pmem_resp == 0

**STORE**

if (!(mem_read && hit) || (mem_write && hit)) && (dirty)

(hit&&mem_read) || (hit && mem_write)

**HIT**

pmem_resp == 1

pmem_resp == 1

**LOAD**

if ( !(mem_read && hit) || (mem_write && hit)) && (!dirty)

pmem_resp == 0

state machine for the caches

# State machine for the arbiter

mem_read_i==0 &&mem_read_d==0 &&mem_write_d==0

**IDLE**

mem_read_i

mem_resp=1&& (!(mem_read_d ||mem_write_d))

mem_resp&& !mem_read_d&& !mem_write_d=1

mem_read_d||mem_write_d)&&!mem_read_i

mem_resp=0

**I-CACHE**

**D-CACHE**

mem_resp=1&& (mem_read||mem_write_d)

mem_resp&& !mem_read_d&& !mem_write_d=0

state machine for the arbitor

Physical Memory

pmem_resp | pmem_rdata | pmem_wdata | pmem_write | pmem_read | pmem_addr

mem_read

mem_write

L2 cache

clk

mem_write_l2 (from arbiter control)

mem_wdata

addr

rdata

mem_addr

MUX for rdata

MUX_sel(from Control)

MUX(For address) 00 01

MUX(for wdata)

Arbitor

inst_rdata

mem_addr_i

mem_addr_d

mem_read_l2 (from arbiter control)

data_rdata

wdata_d(pmem_wdata)

mem_write_i mem_read_i (To Control)

wdata_I(pmem_wdata)

I-cache

read_d (pmem_read)

D-cache

mem_read_d (To Control) mem_write_d (To Control)

mem_resp_i (from Control)

write_d(pmem_write)

mem_resp_d (from Control)