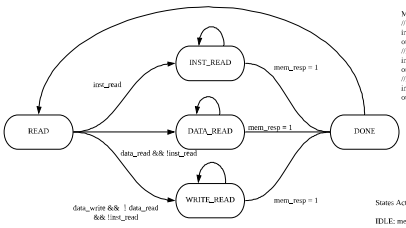


L1 Cache Arbiter

Module Ports:
// from Icache
input inst_read, inst_addr, inst_wdata
output inst_rdata, inst_resp
//from Dcache
input data_read, data_write, data_addr, data_wdata
output data_rdata, data_resp
//from pmem
input mem_rdata, mem_resp
output mem_read, mem_write, mem_addr, mem_wdata

States Action:

IDLE: mem_read = mem_write = 0

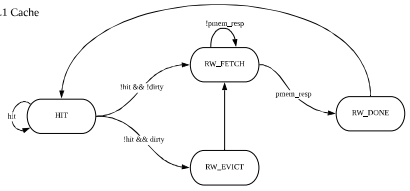
INST_READ: mem_read = 1, mem_addr = inst_addr
done_mux_sel = IR

DATA_READ: mem_read = 1, mem_addr= data_addr
done_mux_sel = DR

DATA_WRITE: mem_read = 1, mem_addr= data_addrmem_wdata = data_wdata
done_mux_sel = DW

DONE: case(done_mux_sel):
IR: inst_resp = 1, inst_rdata = mem_rdata
DR: data_resp = 1, data_rdata = mem_rdata
DW: data_resp = 1

L1 Cache



Module Description:
We are palnning to implement 2-way, 8-words set-associative caches the same as MP2.The SM and module description are the ones for data cache, while instruction cache will not need any components or design that's used for WRITE operation, such as RW_EVICT state and dirty metaarray. (Eviction is never needed if WRITE is never performed) Also note that in the following paragraph, all data and metaarray are assumed to be read/written from the proper sets (decided by addr[7:5]) and proper way (decided by the output of LRU register). For simplicity, we will not reiterate on the value of set index and way index.

State Transition Description:
hit = (addr[31:8] == tag) && valid
mem_read and mem_write are inputs.
dirty is simply the ouput of the dirty metaarray.

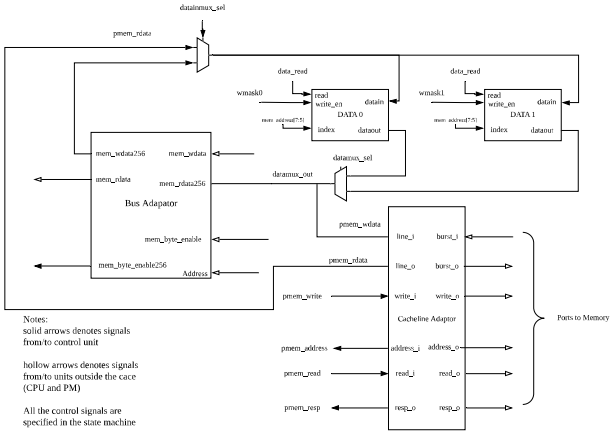
Simplified Ports List:
//Ports to CPU
input mem_read, mem_write, mem_wdata, mem_addr
output mem_resp, mem_rdata
//Ports to physical memory
input pmem_resp, pmem_rdata
output pmem_read, pmem_write, pmem_wdata, pmem_addr

State Action:
HIT: if (hit)
if (mem_read)
mem_rdata = data_arr_out,
update lru, mem_resp = 1
else if (mem_write)
data_arr_in = mem_wdata
update LRU, raise response to cpu, set dirty array

RW_EVICT: pmem_write = 1, pmem_wdata = data_array[LRU_out]

RW_FETCH: pmem_read = 1, data_arr_in = pmem_rdata,
updatetag and valid array

READ_SEND: mem_resp = 1, update LRU
if (mem_read)
mem_rdata = data_arr_out, clear dirty
else if(mem_write)
data_arr_in = mem_wdata, set dirty



Notes:
solid arrows denotes signals from/to control unit

hollow arrows denotes signals from/to units outside the cace (CPU and PM)

All the control signals are specified in the state machine