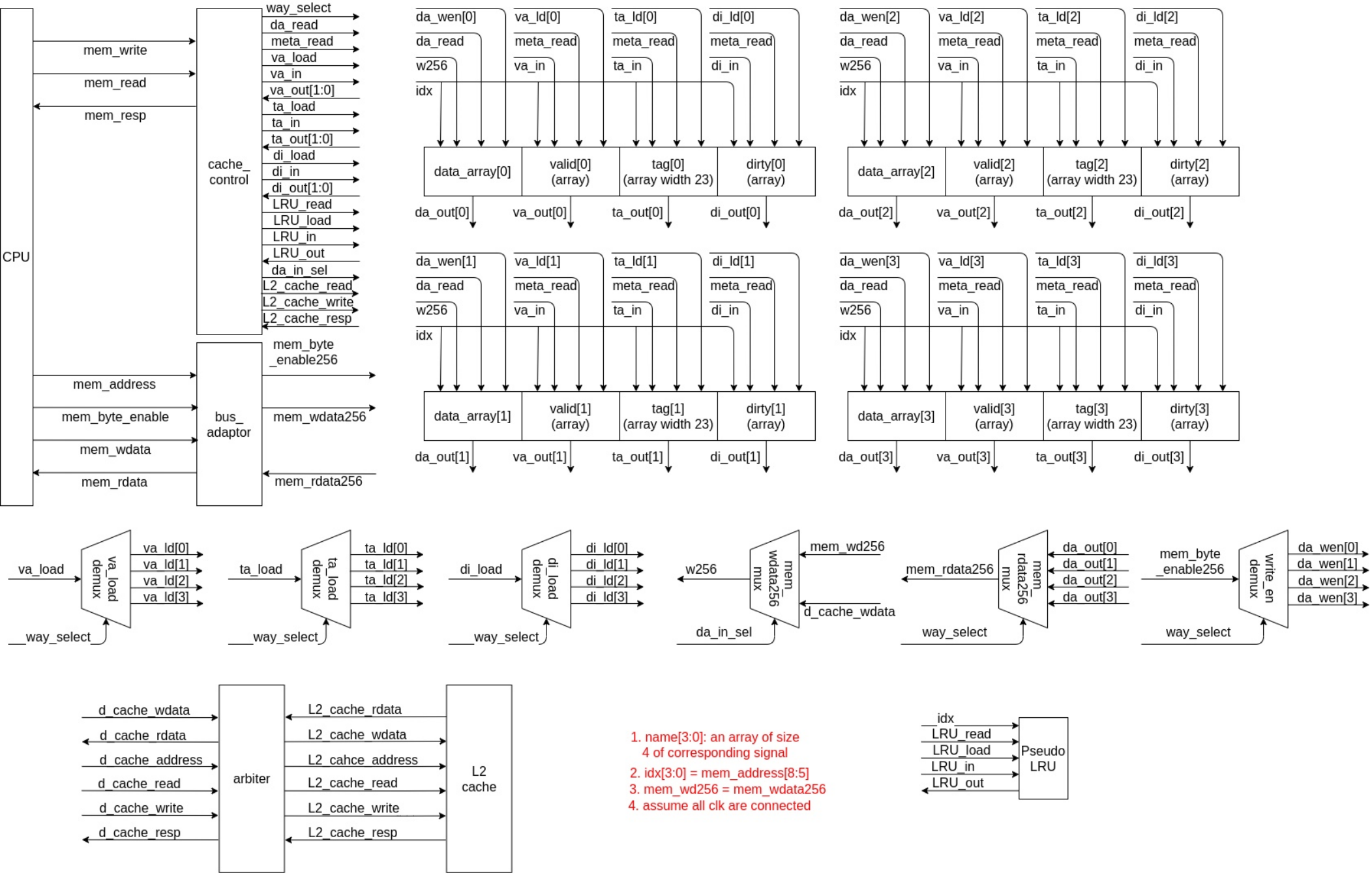
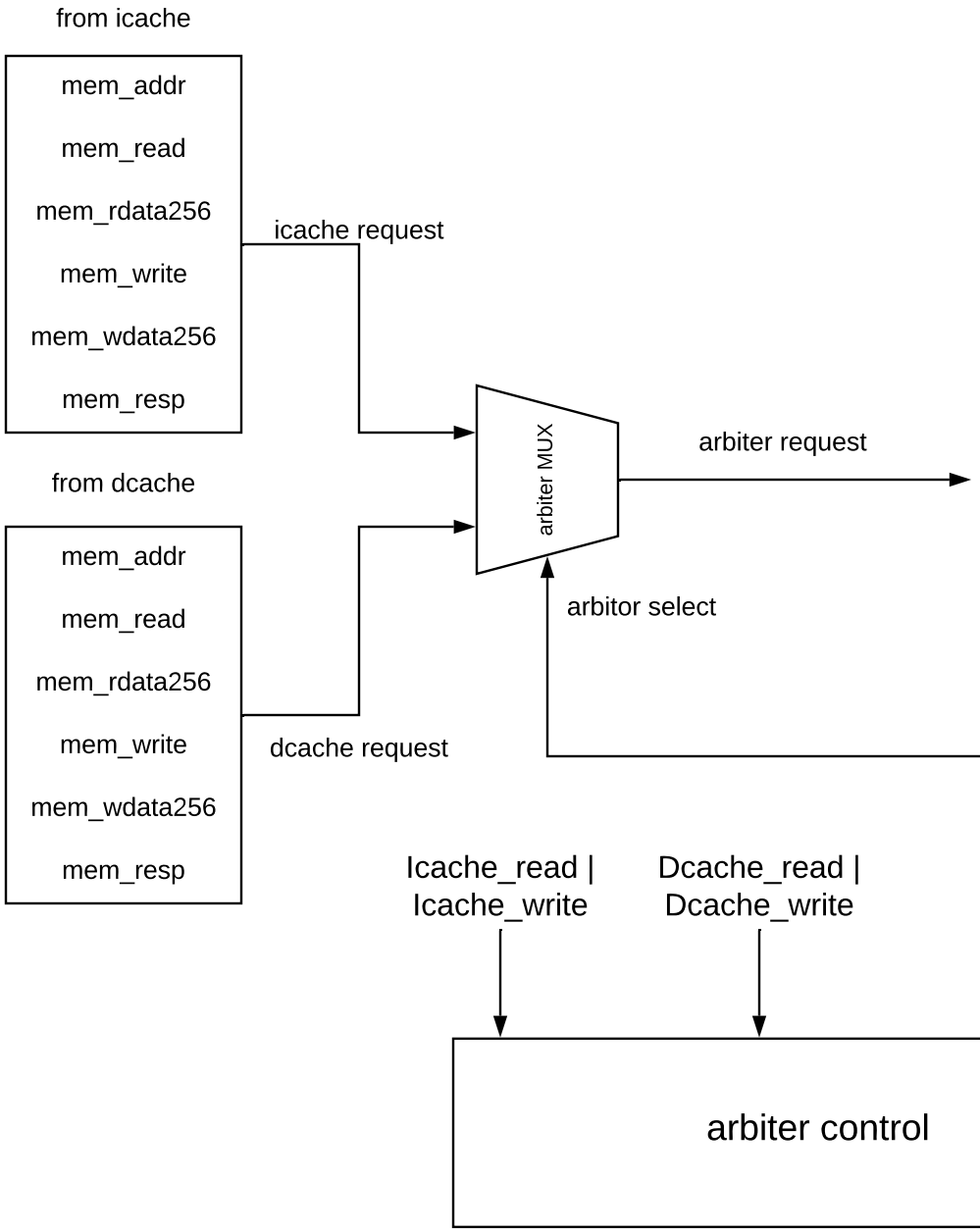
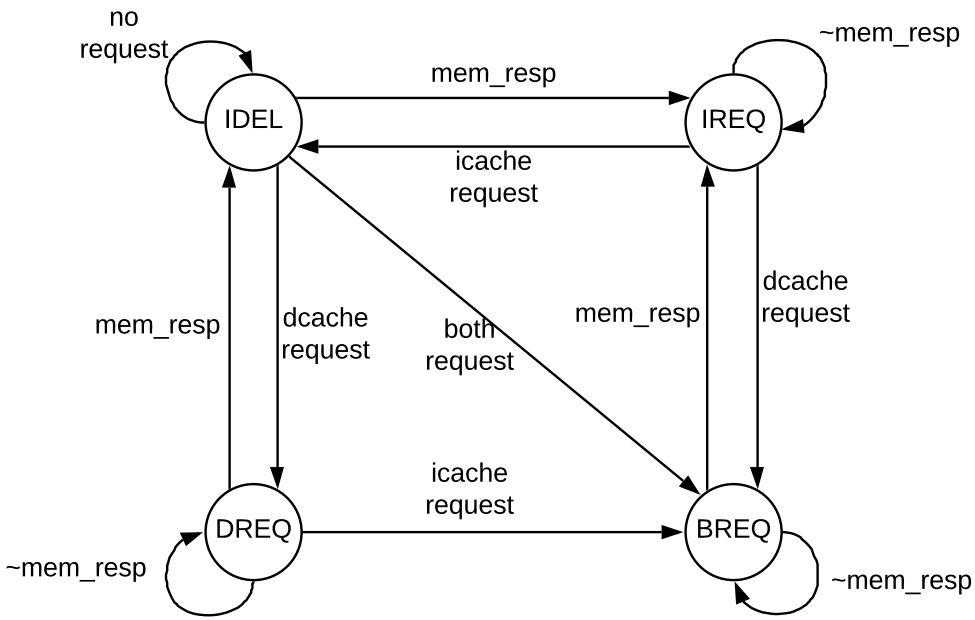
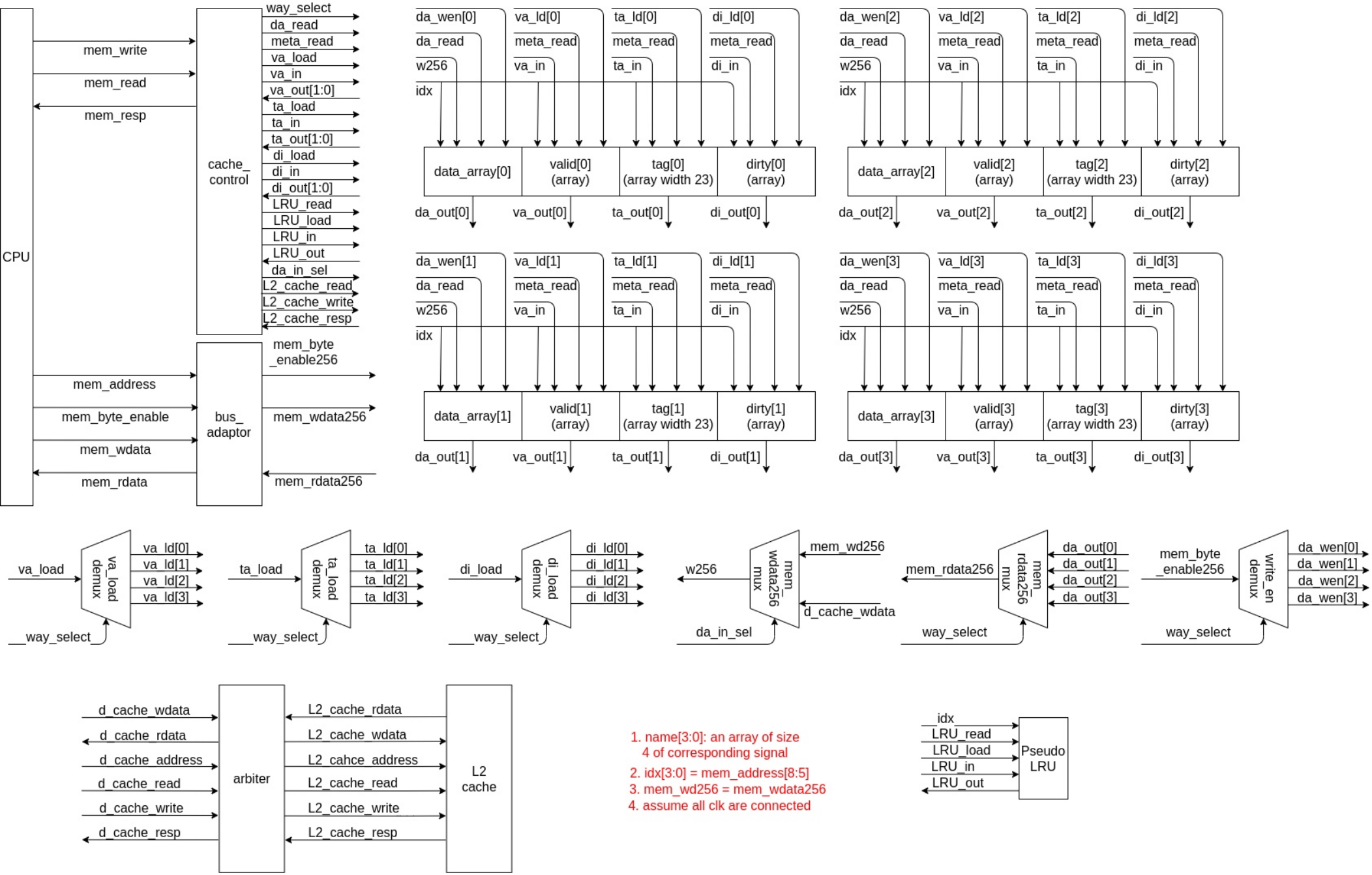

CPU
mem_write
mem_read
mem_resp
cache_
control
way_select
da_read
meta_read
va_load
va_in
va_out[1:0]
ta_load
ta_in
ta_out[1:0]
di_load
di_in
di_out[1:0]
LRU_read
LRU_load
LRU_in
LRU_out
da_in_sel
L2_cache_read
L2_cache_write
L2_cache_resp
mem_address
mem_byte_enable
mem_wdata
mem_rdata
bus_
adaptor
mem_byte
_enable256
mem_wdata256
mem_rdata256
da_wen[0]
va_ld[0]
ta_ld[0]
di_ld[0]
w256
idx
data_array[0]
valid[0]
(array)
tag[0]
(array width 23)
dirty[0]
(array)
da_out[0]
va_out[0]
ta_out[0]
di_out[0]
da_wen[1]
va_ld[1]
ta_ld[1]
di_ld[1]
data_array[1]
valid[1]
(array)
tag[1]
(array width 23)
dirty[1]
(array)
da_out[1]
va_out[1]
ta_out[1]
di_out[1]
da_wen[2]
va_ld[2]
ta_ld[2]
di_ld[2]
data_array[2]
valid[2]
(array)
tag[2]
(array width 23)
dirty[2]
(array)
da_out[2]
va_out[2]
ta_out[2]
di_out[2]
da_wen[3]
va_ld[3]
ta_ld[3]
di_ld[3]
data_array[3]
valid[3]
(array)
tag[3]
(array width 23)
dirty[3]
(array)
da_out[3]
va_out[3]
ta_out[3]
di_out[3]
va_load
va_load
demux
va_ld[0]
va_ld[1]
va_ld[2]
va_ld[3]
way_select
ta_load
ta_load
demux
ta_ld[0]
ta_ld[1]
ta_ld[2]
ta_ld[3]
way_select
di_load
di_load
demux
di_ld[0]
di_ld[1]
di_ld[2]
di_ld[3]
way_select
w256
mem_
wdata256
mux
mem_wd256
d_cache_wdata
da_in_sel
mem_rdata256
mem_
rdata256
mux
da_out[0]
da_out[1]
da_out[2]
da_out[3]
way_select
mem_byte
_enable256
write_en
demux
da_wen[0]
da_wen[1]
da_wen[2]
da_wen[3]
way_select
d_cache_wdata
d_cache_rdata
d_cache_address
d_cache_read
d_cache_write
d_cache_resp
arbiter
L2_cache_rdata
L2_cache_wdata
L2_cahce_address
L2_cache_read
L2_cache_write
L2_cache_resp
L2
cache
1. name[3:0]: an array of size
4 of corresponding signal
2. idx[3:0] = mem_address[8:5]
3. mem_wd256 = mem_wdata256
4. assume all clk are connected
idx
LRU_read
LRU_load
LRU_in
LRU_out
Pseudo
LRU





IDEL: waiting for icache/dcache request
todo: arbitor select dcache request as default
IREQ: arbiter receives icache request
todo: arbitor select icache request
DREQ: arbiter receives dcache request
todo: arbitor select dcache request
BREQ: arbiter receives both request
todo: arbitor select dcache request


CPU
mem_write
mem_read
mem_resp
cache_
control
way_select
da_read
meta_read
va_load
va_in
va_out[1:0]
ta_load
ta_in
ta_out[1:0]
di_load
di_in
di_out[1:0]
LRU_read
LRU_load
LRU_in
LRU_out
da_in_sel
L2_cache_read
L2_cache_write
L2_cache_resp
mem_address
mem_byte_enable
mem_wdata
mem_rdata
bus_
adaptor
mem_byte
_enable256
mem_wdata256
mem_rdata256
da_wen[0]
va_ld[0]
ta_ld[0]
di_ld[0]
w256
idx
data_array[0]
valid[0]
(array)
tag[0]
(array width 23)
dirty[0]
(array)
da_out[0]
va_out[0]
ta_out[0]
di_out[0]
da_wen[1]
va_ld[1]
ta_ld[1]
di_ld[1]
data_array[1]
valid[1]
(array)
tag[1]
(array width 23)
dirty[1]
(array)
da_out[1]
va_out[1]
ta_out[1]
di_out[1]
da_wen[2]
va_ld[2]
ta_ld[2]
di_ld[2]
data_array[2]
valid[2]
(array)
tag[2]
(array width 23)
dirty[2]
(array)
da_out[2]
va_out[2]
ta_out[2]
di_out[2]
da_wen[3]
va_ld[3]
ta_ld[3]
di_ld[3]
data_array[3]
valid[3]
(array)
tag[3]
(array width 23)
dirty[3]
(array)
da_out[3]
va_out[3]
ta_out[3]
di_out[3]
va_load
va_load
demux
va_ld[0]
va_ld[1]
va_ld[2]
va_ld[3]
way_select
ta_load
ta_load
demux
ta_ld[0]
ta_ld[1]
ta_ld[2]
ta_ld[3]
way_select
di_load
di_load
demux
di_ld[0]
di_ld[1]
di_ld[2]
di_ld[3]
way_select
w256
mem_
wdata256
mux
mem_wd256
d_cache_wdata
da_in_sel
mem_rdata256
mem_
rdata256
mux
da_out[0]
da_out[1]
da_out[2]
da_out[3]
way_select
mem_byte
_enable256
write_en
demux
da_wen[0]
da_wen[1]
da_wen[2]
da_wen[3]
way_select
d_cache_wdata
d_cache_rdata
d_cache_address
d_cache_read
d_cache_write
d_cache_resp
arbiter
L2_cache_rdata
L2_cache_wdata
L2_cahce_address
L2_cache_read
L2_cache_write
L2_cache_resp
L2
cache
1. name[3:0]: an array of size
4 of corresponding signal
2. idx[3:0] = mem_address[8:5]
3. mem_wd256 = mem_wdata256
4. assume all clk are connected
idx
LRU_read
LRU_load
LRU_in
LRU_out
Pseudo
LRU

Dcache signals (Icache signals will be the same except all write are disabled)

signal	length	connection	description
mem_resp	1	Cpu and cache control	Response signal note cpu data is ready
mem_read	1	Cpu and cache control	Control signal note cache to read some data
mem_addr	32	Cpu and cache	Address associated with read/write request from CPU
way_select	4	Cache control and data arrays as well as metadata arrays	Distinguish the data/metadata way that we are going to deal with
da_read	1	Cache control and data arrays	Read data from the data array (should always be high in our design)
meta_read	1	Cache control and metadata arrays	Read data from the metadata array (should always be high in our design)
va_load	1	Cache control and valid arrays	Load valid arrays
va_in	1	Cache control and valid arrays	Input to valid arrays (should always be high since valid array will only be filled)
va_out	1*4	Cache control and valid arrays	Output from valid arrays to indicate the validity of the 4 ways corresponding to the 4 bits
ta_load	1	Cache control and tag arrays	Load tag arrays
ta_in	23	CPU and tag array	Input to tag arrays

			(23-bit data comes from mem_addr [31:9]
ta_out	23*4	Tag_array and 4 comparators	Input stored tags to comparators to check for cache hit
di_load	1	Cache control and dirty arrays	Load dirty arrays
di_in	1	Cache control and dirty arrays	Input to dirty arrays (high when doing write hit, low when doing write back)
di_out	1*4	Cache control and dirty arrays	Output from dirty arrays to indicate the dirty of the 4 ways corresponding to the 4 bits
LRU_read	1	LRU and cache control	Read lru array and get corresponding signal associated with corresponding idx
LRU_load	1	LRU and cache control	Load LRU to update it
LRU_in	3	LRU and cache control	Input to LRU to update the least recently used way
LRU_out	3	LRU and cache control	Corresponding LRU array value associated with lru read request made by cache control
idx	4	LRU and cache control	Idx of which lru control unit made its read request
mem_byte_enable	4	CPU and bus adaptor	Indicate which bytes among the 4 will be written

mem_wdata	32	CPU and bus adaptor	Write data from CPU
mem_rdata	32	CPU and bus adaptor	Read data from cache and pass back to CPU
mem_byte_ enable256	32	Bus adaptor to data arrays	Indicate the bytes to be written in the data array
mem_wdata256	256	Bus adaptor to data arrays	Input to data arrays
mem_rdata256	256	Bus adaptor and data arrays	Output from data arrays to the bus adaptor

Arbiter Signals

signal	length	connection	connection
Arbiter select	1	Arbiter control and aibiter mux	Select who's request to take (Icache or Dcache)
Icache_reqeust	struct	Icache and arbiter	Signal a read or write request is initiated by Icache
Dcache_request	struct	Dcache and arbiter	Signal a read or write request is initiated by Dcache
l2mem_resp	1	Arbiter and L2 cache	Control signal note previous request is done and memory is ready
arbiter_request	struct	Arbiter and L2 cache	Request made by arbiter to L2 cache

Request struct

signal	length	connection
l2mem_addr	32	Requested address to read or

		write.
l2mem_read	1	Read signal.
l2mem_rdata256	256	Cacheline read from L2 cache.
l2mem_write	1	Write signal.
l2mem_wdata256	256	Cacheline write to L2 cache.
l2mem_resp	1	Response from L2 cache.