

write-
buff-0

write-
buff-1

read-
buff-0

get-buffer

regs to save

read/write/data in addr

(hit & valid)

write-
buff-1

write-
buff-3

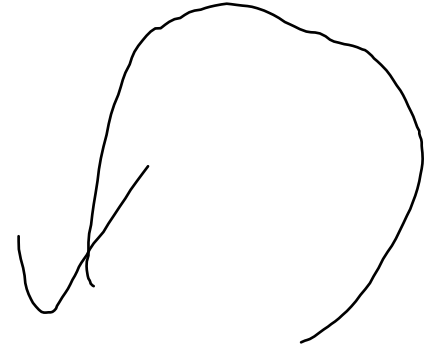
read-
buff-1

1 reg to save
read data from mem

(~wr & mod)

4 regs to save
4 writers from cache

2 reg to save
read data from
mem

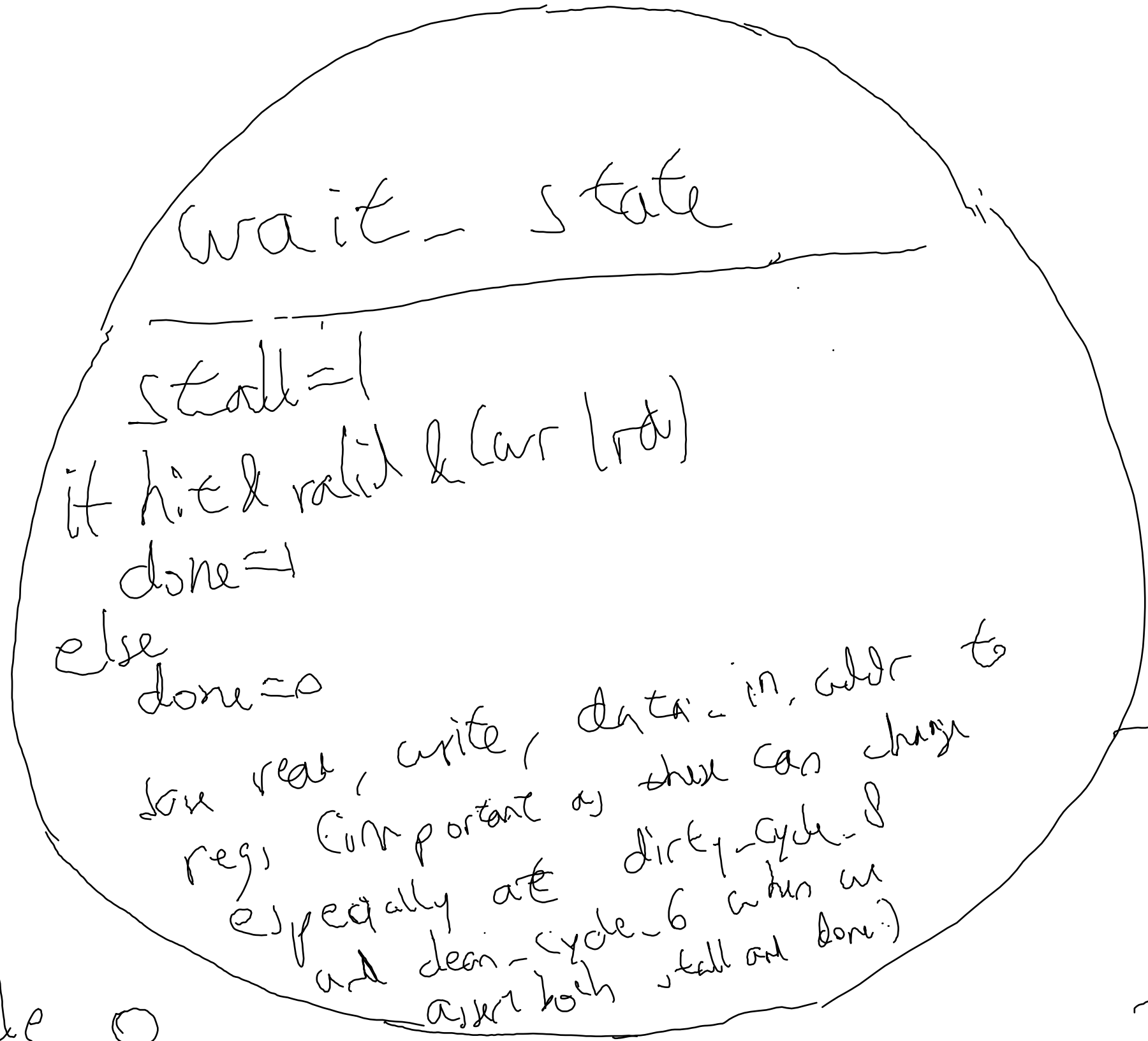


tag-m

reg to save the
old effect from cache

(cur load) & (~hit & ~valid)
& dirty

to dirty-cycle-0



(cur load) & (~hit & ~valid)
& ~dirty

to clean-cycle-0

from
wait_state

clean_cycle_0

stall=1
assert mem_rd = 1
the address to memory module
is same as addr_in but
change offset (addr[2:1]) to 0

clean_cycle_1

same as clean_cycle_0
but use offset 1 instead

clean_cycle_2

stall=1
still assert mem_rd = 1,
and still in addr_in for
mem_addr, and change offset to 2
the first read is now ready, save it
to cache with comp = 0, cache_addr
is addr_in but the offset is 000
if addr_in[2:1] == 0:
if rd:
then save mem output to write buffer
if wr:
then save data in to cache instead of
memory output

clean_cycle_3
same as clean_cycle_2
but with cache offset 110
done=1

if rd:
if addr_in[2:1] == 3
use mem out as data out
else
use write buffer as data out
if wr:
if addr_in[2:1] == 3
use data in as cache input instead of mem out

back
to wait
state

clean_cycle_4

stall=1, data from which
read ready, again save it to
cache with offset 100
again if addr_in[2:1] = 2,
save to write buffer or data in to cache

clean_cycle_5

same as clean_cycle_4,
but use offset 3 for mem,
data from second memory is ready,
again save that to cache with
offset 010
again if addr_in[2:1] = 1, save data in to cache
or mem to write buffer
depending on write

from wait-state

dirty-cycle-0

still=1
save tag-out from cache
comp=0
mem_offset=cache_offset=0



dirty-cycle-1

still=1
comp=0
mem_offset=cache_offset=1



dirty-cycle-2

still=1
comp=0
mem_offset=cache_offset=2

data from first memory read
ready, save to read buffer-0



dirty-cycle-3

still=1 comp=0
mem_offset=cache_offset=3

data from second memory read
ready, save to read buffer-1

mem rd=1
cache rd(enable=1, ar=1)=1
use addr_in for mem
and cache, but change the offset
to their respective offset

to write buffer
0 to 3 in
that order

mem-ar=1
(can write buffer 0 to 3
in that order)

Cache wr(enable=1, ar=1)=1

use addr_in for cache
(change offset as per value)

use the tag from
the cache (that we sent
on dirty-cycle-0),
index from addr_in and
offset per word for
memory

dirty-cycle-4

third mem read ready, save to cache with
cache_offset=2, save to outbuffer if rd=1
and addr_in[2:1]=2. if ar=1 & addr_in[2:1]=2,
use data_in instead for cache input

mem_offset=0
still=1

dirty-cycle-7

save read-buffer-1 with cache_offset=1
output to device
to if rd=1 and
addr_in[2:1]=1, data in outbuffer for device
if wr=1 and addr_in[2:1]=1
use data_in instead for cache input
mem_offset=3
still=1, done=1

back
to wait
state

dirty-cycle-6

save read-buffer-0 with cache_offset=0,
save to outbuffer to if rd=1 and
addr_in[2:1]=0, if wr=1 and addr_in[2:1]=0,
use data_in instead for cache input
mem_offset=2
still=1

dirty-cycle-5

fourth mem read ready, save to cache with
cache_offset=3, save to outbuffer if rd=1
and addr_in[2:1]=3. if ar=1 & addr_in[2:1]=3,
use data_in instead for cache input
mem_offset=1
still=1