Report

B07901068 黃敬騰

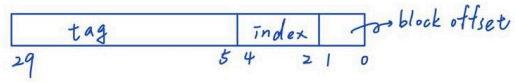
(a) The cycle time used in **cache_syn.sdc** : 10ns

The gate-level pass cycle time for **cache_dm_syn.v** : 10ns

The gate-level pass cycle time for **cache_2way_syn.v** : 10ns

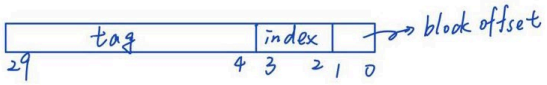
(b) Placement of cache

Direct-mapped



index	valid	tag	data (4 words)
0			
1			
⋮			
7			

2way-associated

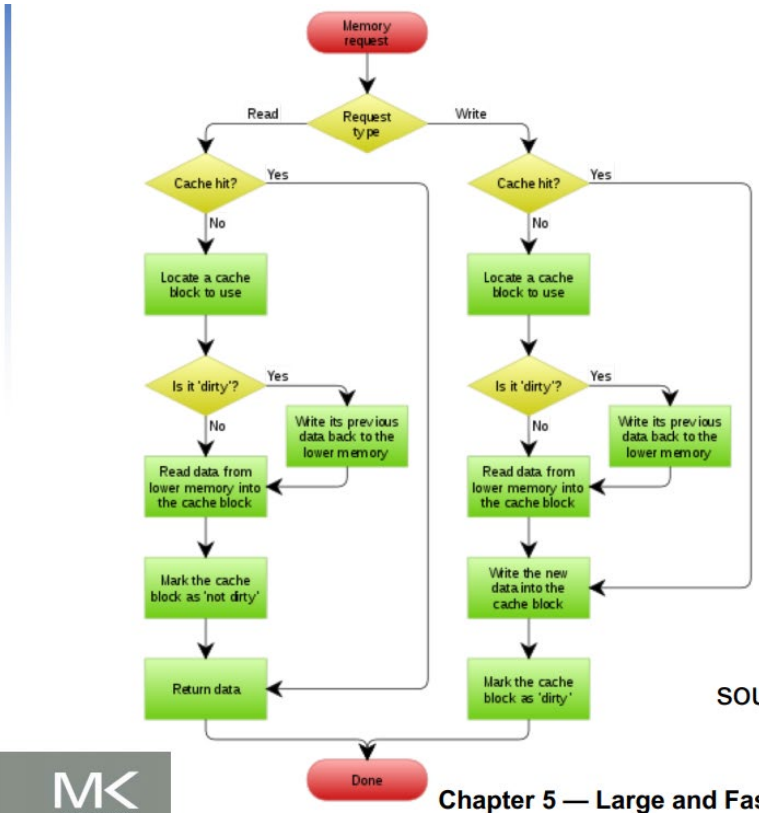


index	valid	tag	data	valid	tag	data
0						
1						
⋮						
3						

For 2way-associated cache, if a miss occur(memory allocation is needed), I will choose the replaced block as "less recently accessed". If a block is just replaced by the block from lower memory, it will be marked as "recently used" and the other block in the same index will be marked as "less recently used".

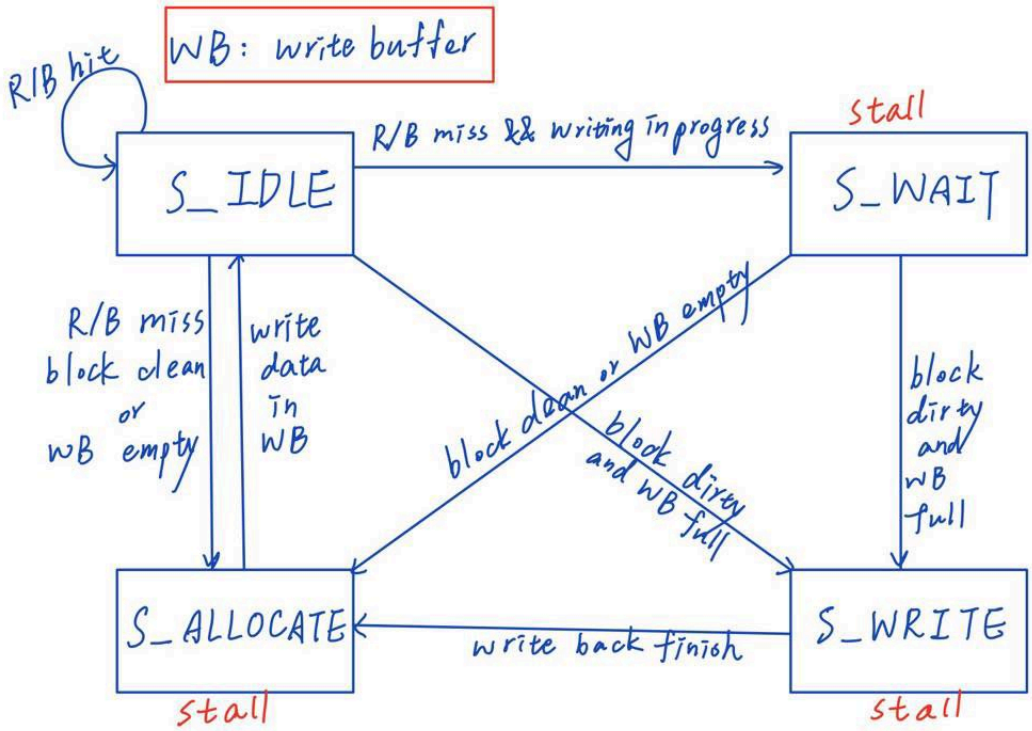
(c) Read/Write policy

I choose **write-back** as memory allocation, picture below is the block diagram of write-back method I use.



source: wiki

(d) FSM(with write buffer)



(e) Performance Evaluation

Direct-mapped(without write buffer)

Read miss rate : 1280 / 2048 = 62.5%

Write miss rate : 256 / 1024 = 25%

Write back (dirty) rate : 256 / 3072 = 8.33%

Execution cycles : 12031

Stalled cycles : 8960

2way-associated(without write buffer)

Read miss rate : 512 / 2048 = 25%

Write miss rate : 256 / 1024 = 25%

Write back (dirty) rate : 256 / 3072 = 8.33%

Execution cycles : 8191

Stalled cycles : 5120

(f) Compare

From (e), we can see that the most significant difference between two designs is “Read miss rate”, which is much higher in direct-mapped cache design. The reason is in the **last part of processor read**, whose input address pattern is 0 -> 32 -> 1 -> 33 -> 2 -> 34 …… 991 -> 1023.

For direct-mapped cache, memory of address 32 will erase memory of address 0 since they possess the same index and 1 must erase 32 for the same reason because each entry **has the capacity of only one block**, so it has to allocate memory on **every** read operation.

On the other hand, every entry of 2way-associated cache has cacpacity of 2 blocks, so memory allocation only occurs at 0 , 32, 4, 36 ……., which is **four times smaller** than that of direct-mapped.

Additional features(Bonus)

1. **Write buffer**

   In order to decrease the processor stall on writing the dirty memory back to lower level memory, I make a 1-entry write buffer in both direct-mapped and 2way-associated cache. When a dirty block must be written back to memory, It will first check whether the write buffer is empty. If so, It will put the dirty block and its address in the buffer and proceed directly to fetch new block from memory. After that, the data in write buffer can write the data to memory without stalling due to memory write back. In short, the design only have to stall on writing if the write buffer is not empty.

   Below is the improvement in total time(RTL simulation)

   Without write buffer

   Direct-mapped

```
Processor: Read initial data from memory.
    Done correctly so far! ^_^

Processor: Write new data to memory.
    Finish writing!

Processor: Read new data from memory.
    Done correctly so far! ^_^

==== CONGRATULATIONS! Pass cache read-write-read test. ====

Finished all operations at:              120405 ns
Exit testbench simulation at:            120505 ns
```

   2-way associated

```
Processor: Read initial data from memory.
    Done correctly so far! ^_^

Processor: Write new data to memory.
    Finish writing!

Processor: Read new data from memory.
    Done correctly so far! ^_^

==== CONGRATULATIONS! Pass cache read-write-read test. ====

Finished all operations at:               82005 ns
Exit testbench simulation at:             82105 ns
```

   With write buffer

   Direct-mapped

```
Processor: Read initial data from memory.
    Done correctly so far! ^_^

Processor: Write new data to memory.
    Finish writing!

Processor: Read new data from memory.
    Done correctly so far! ^_^

==== CONGRATULATIONS! Pass cache read-write-read test. ====

Finished all operations at:              110415 ns
Exit testbench simulation at:            110515 ns
```

   2-way associated

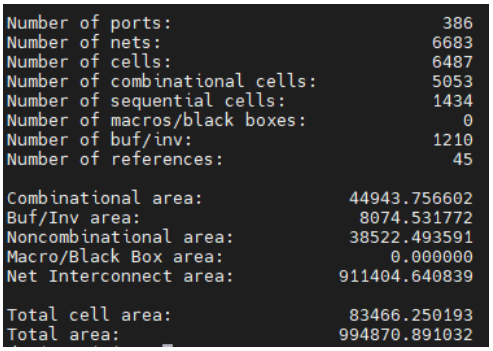
```
Processor: Read initial data from memory.
    Done correctly so far! ^_^

Processor: Write new data to memory.
    Finish writing!

Processor: Read new data from memory.
    Done correctly so far! ^_^

==== CONGRATULATIONS! Pass cache read-write-read test. ====

Finished all operations at:               71975 ns
Exit testbench simulation at:             72075 ns
```

   The time save is about **10000ns**, **1000 clock cycle**. Since the testbench in this homework does not produce many “dirty” blocks (only 256 operations induce write-back), there is few needs of writing back. Therefore the timing improvement is not so clear. But I think if the memory access is more random, the improvement will be significantly greater than this amount !!

The additional **area cost** to implement write buffer

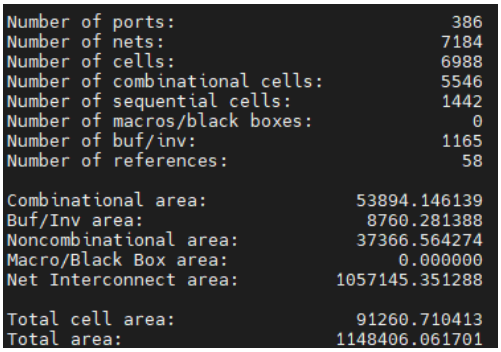
Without write buffer

Direct-mapped



```
Number of ports:                          386
Number of nets:                          6683
Number of cells:                         6487
Number of combinational cells:           5053
Number of sequential cells:              1434
Number of macros/black boxes:               0
Number of buf/inv:                       1210
Number of references:                      45

Combinational area:              44943.756602
Buf/Inv area:                     8074.531772
Noncombinational area:           38522.493591
Macro/Black Box area:                0.000000
Net Interconnect area:          911404.640839

Total cell area:                 83466.250193
Total area:                     994870.891032
```

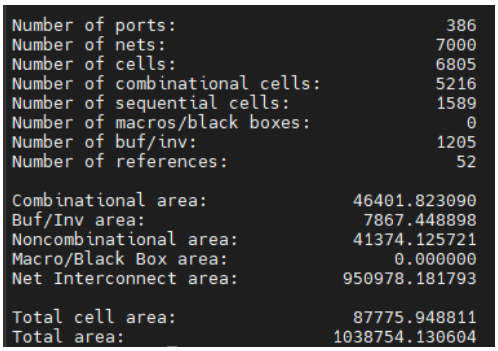
2way-associated



```
Number of ports:                          386
Number of nets:                          7184
Number of cells:                         6988
Number of combinational cells:           5546
Number of sequential cells:              1442
Number of macros/black boxes:               0
Number of buf/inv:                       1165
Number of references:                      58

Combinational area:              53894.146139
Buf/Inv area:                     8760.281388
Noncombinational area:           37366.564274
Macro/Black Box area:                0.000000
Net Interconnect area:         1057145.351288

Total cell area:                 91260.710413
Total area:                    1148406.061701
```

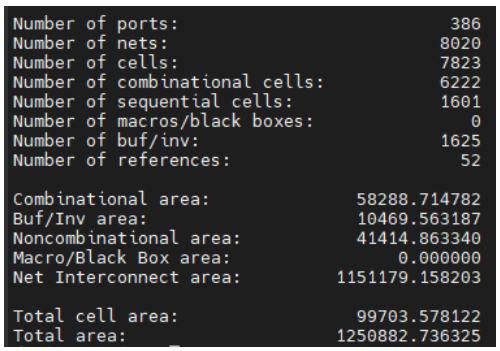
With write buffer

Direct-mapped



```
Number of ports:                          386
Number of nets:                          7000
Number of cells:                         6805
Number of combinational cells:           5216
Number of sequential cells:              1589
Number of macros/black boxes:               0
Number of buf/inv:                       1205
Number of references:                      52

Combinational area:              46401.823090
Buf/Inv area:                     7867.448898
Noncombinational area:           41374.125721
Macro/Black Box area:                0.000000
Net Interconnect area:          950978.181793

Total cell area:                 87775.948811
Total area:                     1038754.130604
```

2way-associated



```
Number of ports:                          386
Number of nets:                          8020
Number of cells:                         7823
Number of combinational cells:           6222
Number of sequential cells:              1601
Number of macros/black boxes:               0
Number of buf/inv:                       1625
Number of references:                      52

Combinational area:              58288.714782
Buf/Inv area:                    10469.563187
Noncombinational area:           41414.863340
Macro/Black Box area:                0.000000
Net Interconnect area:         1151179.158203

Total cell area:                 99703.578122
Total area:                    1250882.736325
```