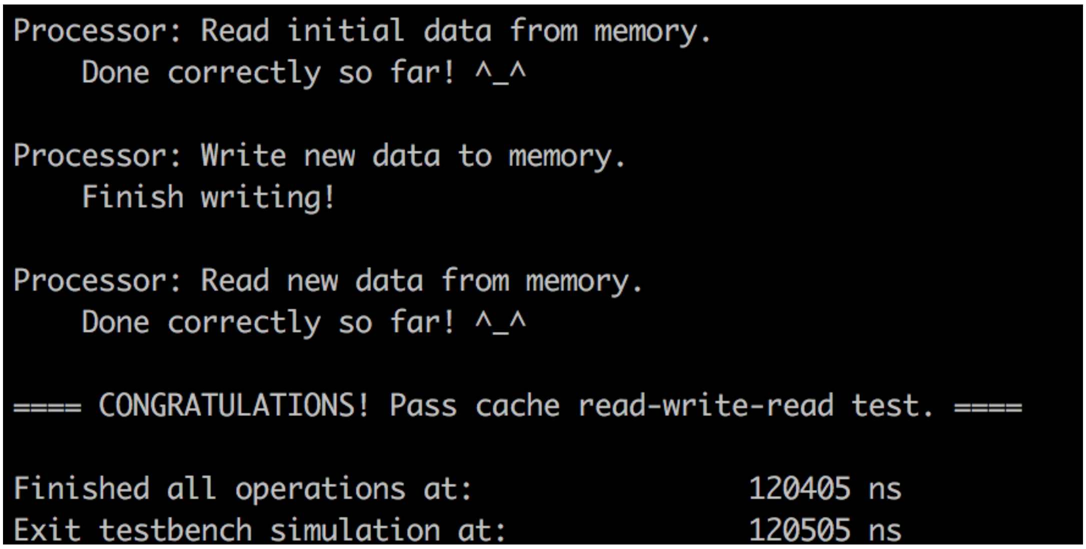
1. Direct mapping

simulation:



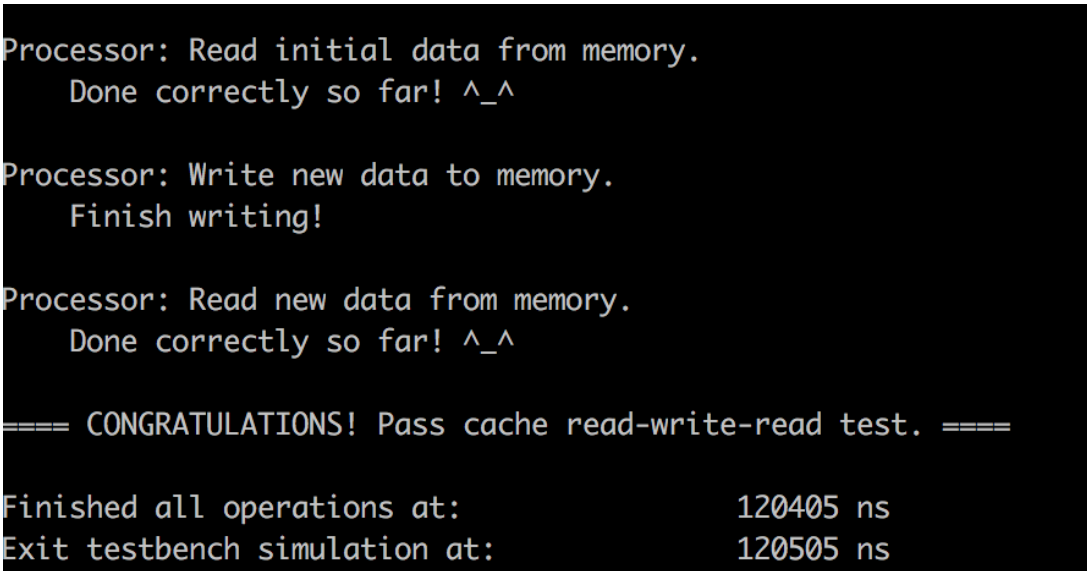
```
Processor: Read initial data from memory.
    Done correctly so far! ^_^

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
```

Structure:

Number of blocks: 8

Number of words per block: 4

2. 2-way set associative



```
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
```

Structure:

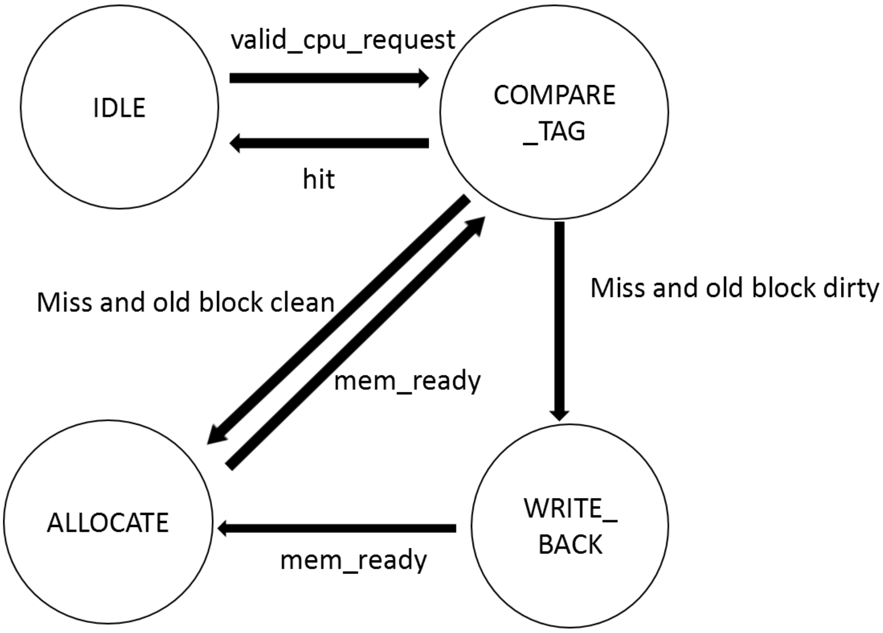
Number of blocks (per way): 4

Number of words per block: 4

Placement policy : LRU (least recently used)

Direct mapping 和 2-way 在 FSM 的設計上相似，首先 compare tag，若 hit 則無需

做任何動作，若 not hit 則需要向 memory 拿取 data 並 allocate，此時又分成兩種情況，若該 block 為 dirty，必須先將那格 data write back，才能 allocate。
這兩者的不同在於 2-way 使用了 LRU (least recently used)，為選擇 LFU 的原因是因為 LFU 需要額外使用較多的 bits 紀錄使用次數，而 2-way 中每一個 block 只需要一個 bit 就能記錄何者為較舊的 data。
FSM 參考講義上的圖：



心得：這次作業實作起來沒有很複雜，FSM 寫出來後基本上不會有很大的問題，兩小題也只差在 LRU 的使用，稍微修改即可。在測試合成的的時候有一個小發現，即使只是 assignment，順序不同也會影響他合成的邏輯，area 會不一樣，反而是 if else 等條件式不同的寫法 compiler 都有辦法理解並合成一樣的 design。然後在 gate level simulation 忘記改 SDF，浪費不少時間。