Instruction Cache Design

1043335 賴詩雨

## Source Code

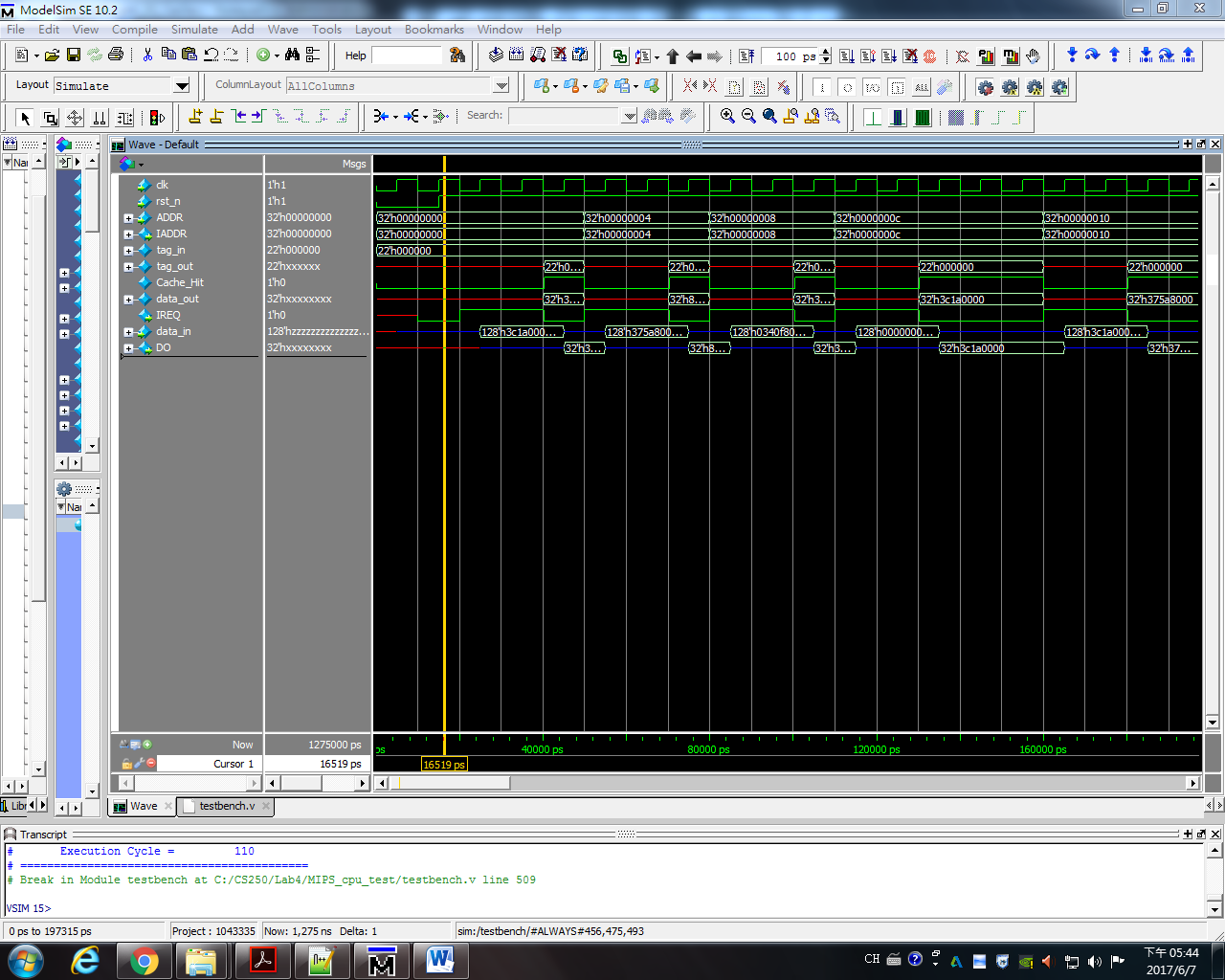
* 實作一 ( 紅框內為填上的答案 )

|  |
| --- |
|  |

* 實作二

|  |
| --- |
|  |

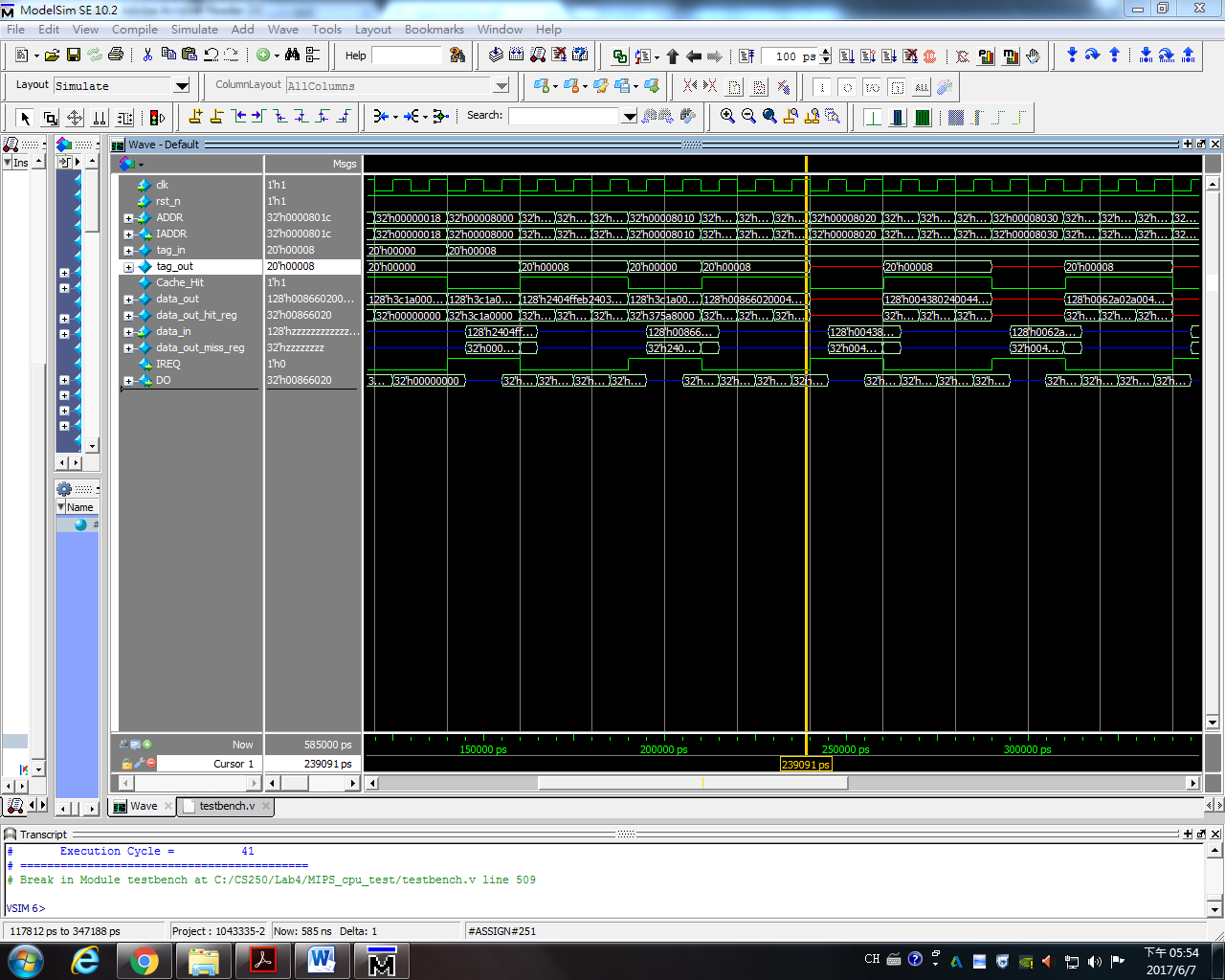
## 實驗結果

* 實作一波形圖
* 實驗一分析

1. 我們會在cache miss時將ADDR傳到IADDR ( memory )，使用IREQ去控制
2. 因為cache是256MB，所以 index為 8 bits，為ADDR[2]~ADDR[9]
3. ADDR[10]~ARRD[31]為tag\_in的位置
4. 利用index取得data\_out、tag\_out 和 valid\_out在cache memory的值
5. 檢查 valid\_out 是否為一，如果是，再去比對 tag\_in和tag\_out，並將結果放進 Cache\_Hit
6. 如果CacheEnable 且cache miss的話，就去access memory，並把結果放到IREQ
7. 再去判斷output\_enable，決定DO是DO\_reg還是32’bz

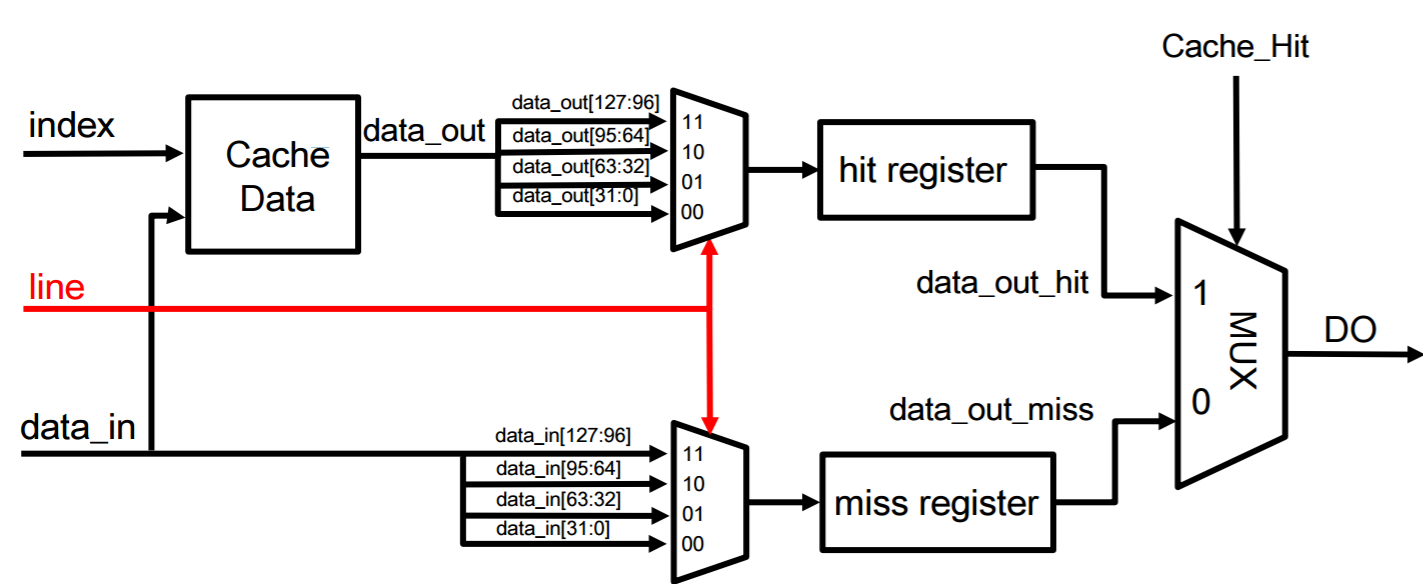
* 當處理器發出記憶體訪問請求時，會先查看快取內是否有請求資料。如果存在（命中），則不經訪問記憶體直接返回該資料；如果不存在（失效），則要先把記憶體中的相應資料載入快取，再將其返回處理器。
* 實驗一結果

由實驗一波形圖可以看到讀到一個值的時候就會產生miss，是因為每個entry只有1 word。從圖上的黃色框框中可以看到當ADDR改變時，就會產生 Cache hit ( IREQ = 0 )，之後改變了為 Cache miss ( IREQ = 1 )。

* 實作二波形圖
* 實驗二分析

1. 我們會在cache miss時將ADDR傳到IADDR ( memory )，使用IREQ去控制
2. 因為line有四個，所以是2 bits，為ADDR[2]~ADDR[3]
3. 因為cache是256MB，所以 index為 8 bits，為ADDR[4]~ADDR[11]
4. ADDR[12]~ARRD[31]為tag\_in的位置
5. Line 有四種可能，分別是00, 01, 10, 11，所以要判斷不同可能要分別給data\_out\_hit\_reg和data\_out\_miss\_reg什麼值

* 當處理器發出記憶體訪問請求時，會先查看快取內是否有請求資料。如果存在（命中），則不經訪問記憶體直接返回該資料；如果不存在（失效），則要先把記憶體中的相應資料載入快取，再將其返回處理器。
* 透過miss register可以先將memory的資料output到CPU，以節省 CPU等待的時間



* 實驗二結果

由實驗二波形圖可以看到讀到四個值的時候就會產生miss，是因為每個entry有4 words。從圖上的黃色框框中可以看到當ADDR改變時，就會產生 Cache hit ( IREQ = 0 )，之後改變了為 Cache miss ( IREQ = 1 )。

## 心得

* 實驗心得:

在這次實驗以前，我就對於Direct Mapped I-Cache 有一些租淺的了解，不過在老師講完課以後，整個觀念就都通達了。很感謝助教們用心幫我們先寫好code讓我們能少寫很多，也有更多時間去研究整個流程是如何運作的。

非常感謝助教們在實驗課的時候幫我找問題，雖然最後發現是我犯了很蠢的錯誤，不過還是謝謝助教們一直幫我看是哪裡出錯。

寫報告真的是能夠更進一步好好分析完整個流程，因為要好好想過以後才能完成這個報告，所以現在完成這份報告的我，比剛填完code的我又更熟悉整個運作方式了。

* 對本次Lab的改進建議:

這次的lab因為有助教幫我們寫好幾乎90%的code，所以很快可以結束，不過我在寫的過程中還是出了很多錯，但是正是因為這樣，也學到很多而且在跌跌撞撞中更能釐清自己原本所模糊的地方。