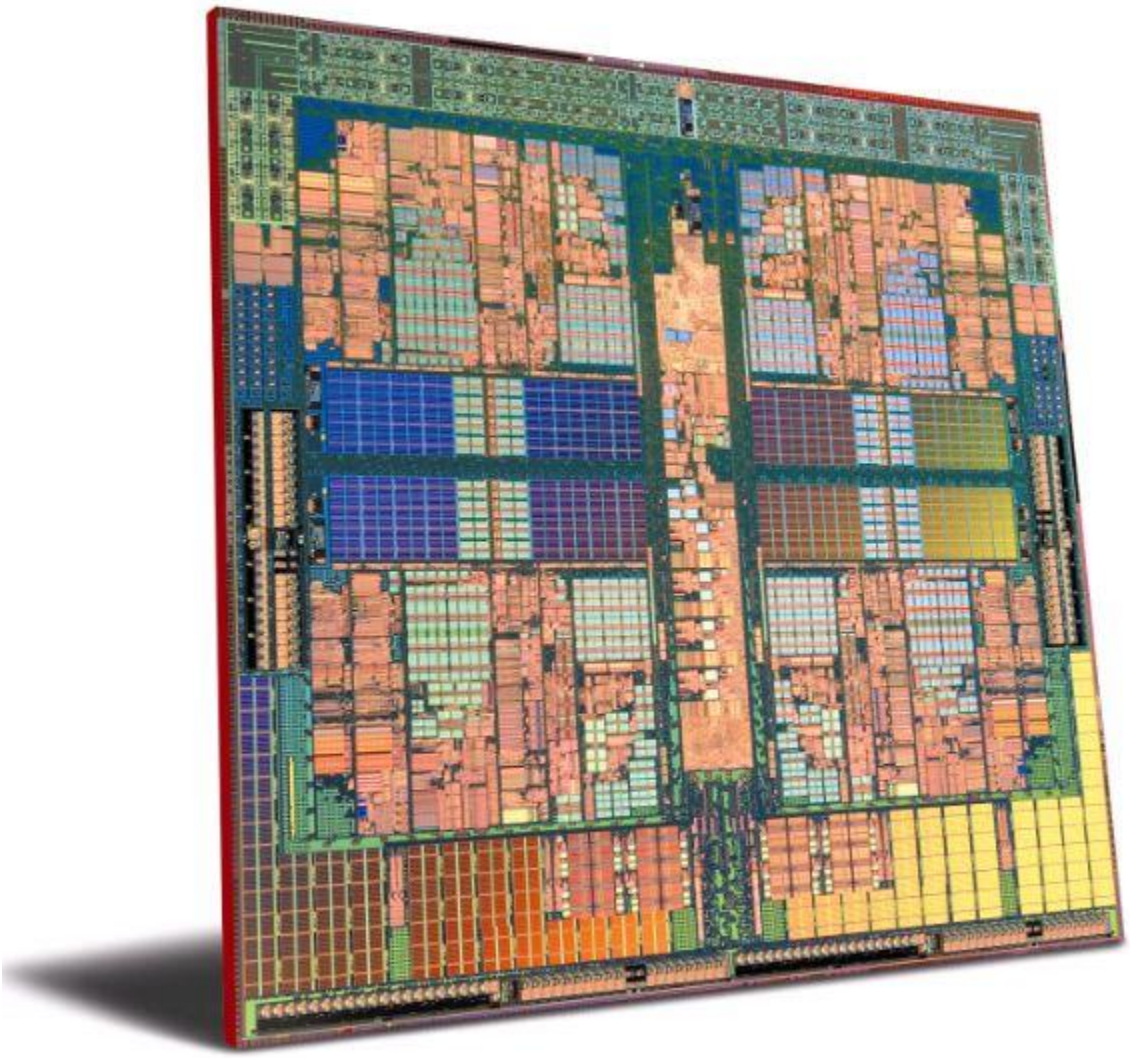
Final Report for Final Project-MIPS Pipelined Processor with Two Level Cache and Branch Prediction
Version 1.0.0
Designed by Shih-Chieh Lin, Chi-Wen Cheng, Yen-Hsiang, Tseng



[MIPS Pipelined Processor Report]

This document gives a brief explanation about how our team design and implement the MIPS Pipelined Processor . If you have any question about it, please feel free to contact us.

Computer-aided VLSI System Design Final Project

Version 1.0.0

Final Submission

January 9th 2014

This hardware/software support is available via phone or email.

Please contact your local office, or directly:

Designer:	Yen-Hsiang, Tseng
	Chi-Wen Cheng
	Shih-Chieh Lin
Phone:	+886-930-671-038
	+886-917-066-756
	+886-919-938-754
Email:	b99901041@ntu.edu.tw
	b99901042@ntu.edu.tw
	b99901100@ntu.edu.tw

Table of Contents

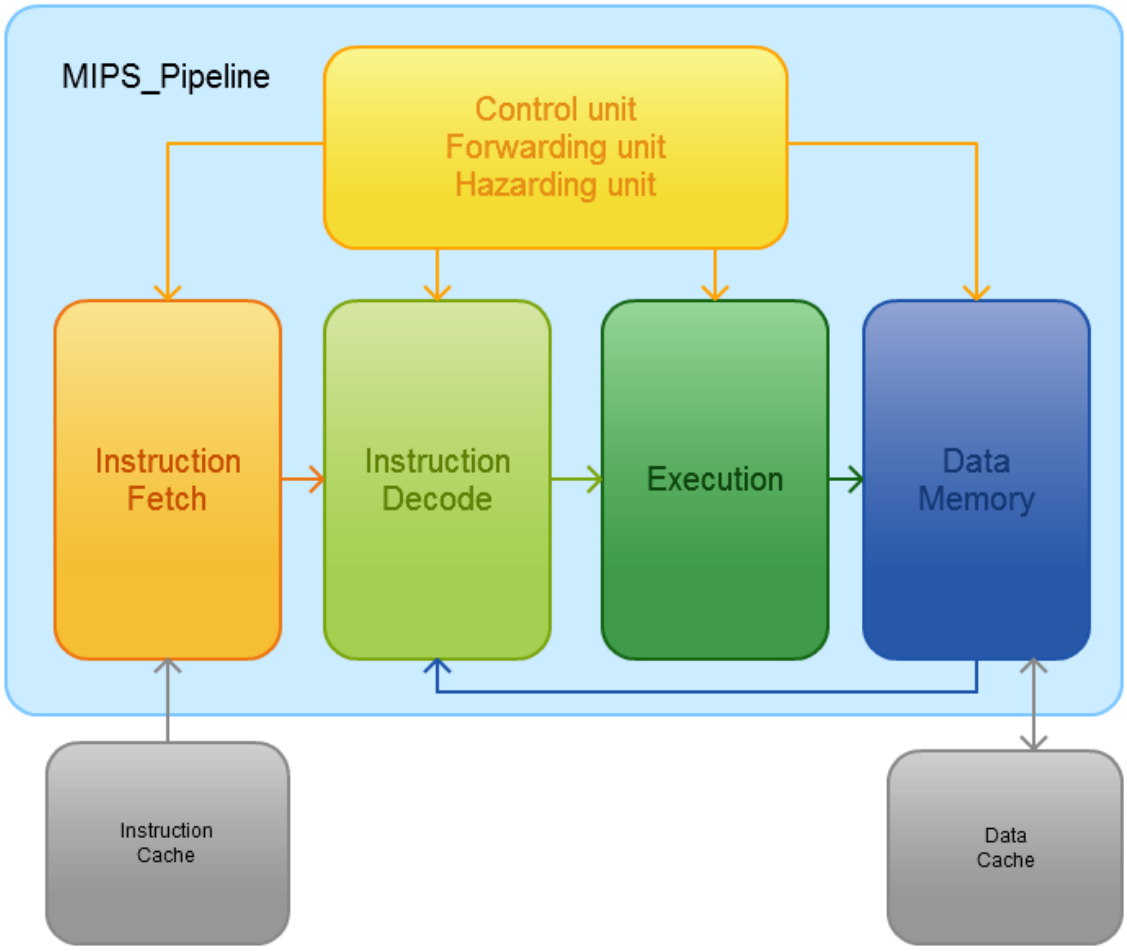
Contents

I. 架構設計

Processor

MIPS_Pipeline

這是整個 design 的 top module，因為希望整個架構清楚，所以這邊負責的是四個主要的 datapath submodule 之間的連線，還有整個電路的 control signal，像是分類 instruction,forwarding, hazard control 等。



Control unit 的部分與 single-cycle MIPS 類似，同樣是根據 instruction 來給出控制訊號，只是多了幾個要支援的指令，但給出相對應的 control signal 應該不難。

Forwarding 的部分我們在這次的 design 做了兩個，一個是大部分人處理 data hazard 會做的 forwarding 到 Execution 的 forwarding unit，另一個是我們後來為了提早 branch 判斷而拉到 Instruction Decode 的 branch forwarding unit。後者一開始純粹是為了解決 early branch 發生的 data hazard 而直覺想出的做法，後來證實這是一個很糟糕的設計，大大增加了 critical path 的長度，但我們想也是趁機學到一課。

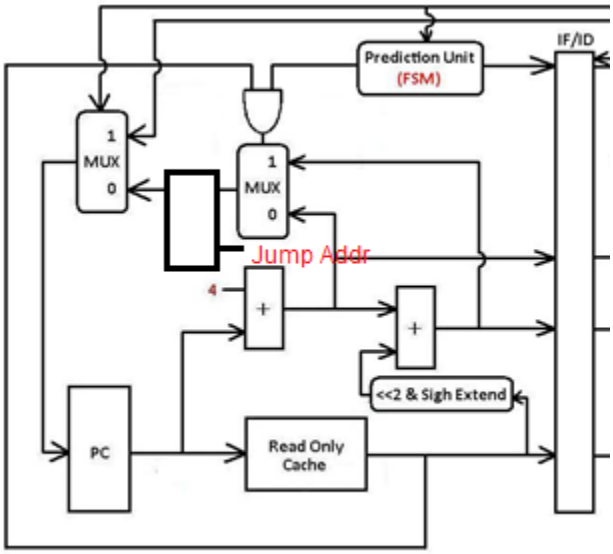
Hazard detection unit 在我們的設計裡面只有 load word 的時候可能發生錯誤，因為這時候不可能靠 forwarding 的方式來解決，此時必須停住 processer。

最後是 IF_Flush 訊號，在 prediction miss 或是 jr 指令的時候必須清掉 IF 準備進到 ID 的指令，因為這個指令會是錯的。

Instruction Fetch

Instruction Fetch 的部分做的事情並不複雜，是負責根據 program counter 與 instruction cache 來進行指令的讀取。

比較麻煩的是在 branch 跟 jump 的處理，當發生這些指令的時候，指令應該如何預測、如何在發現 miss 之後進行修正，最後會發現這邊也是整個 critical path 上面的一部分，因此在處理這些 address 時也不能輕忽其重要性。



Instruction Decode

Instruction Decode 原本最重要的工作是要負責解出 instruction 並給出控制訊號，但這個部份我把它移到 top module 去做了，因此剩下的工作最重要的是裡面的 32*32 的 register file，這裡我們也有提出一些方法試圖改進，雖然最後證實也是不好的設計，但在設計特色裡面會詳加說明。

把控制訊號以及 register 的 address buffer 住傳給後面的 module 也是重要的工作之一。其他還有像是 sign extension、JAL buffer(見設計特色)、判斷 branch 是否正確等工作也是在這個 stage 完成。

Execution

Execution 是負責運算處理的 stage，根據 control signal 的 ALUOp[2:0]以及 Instruction[5:0]來決定 ALUctrl 訊號。ALU 根據 ALUctrl 訊號來決定應該要對 ALUData1 以及 ALUData2 做什麼樣的運算。ALUOp 以及 ALUctrl 的訊號要給多少並不重要，只要不要重複或不會不夠用就可以了。

這個 Stage 另一個重要的 unit 是根據 forwarding unit 的訊號來決定 ALUData 應該要是誰。

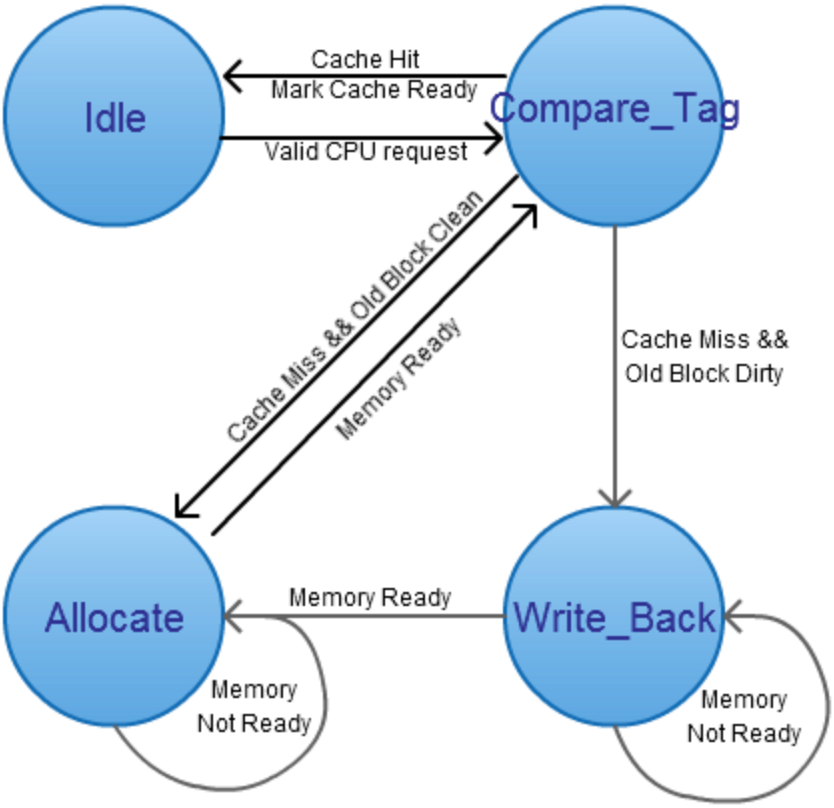
Data Memory

這個 Stage 做的事情再簡單不過，主要是負責跟 data memory 或是 cache 進行溝通，根據 MemRead 或是 MemWrite 來對 memory 進行讀寫。

Cache

Direct Map Cache

Finite State Machine 如下：



我們後來決定用 Meely machine 的方式操控，因為 Moore machine 似乎會在 L2 讀寫時發生錯誤，因此採用比較穩定的 Meely machine 的寫法。

2-way Associated Cache

基本上跟 Direct map 很接近，但在很多判斷時必須要考慮 2 個 way，相對來說複雜許多。

替換的方式是用 Least Recent Used，對只有 2 個 way 來說還不算太複雜，在 write back 時都是根據這個 Least Recent Used 的 bit 來決定，只要這一次一個 way 被改到，另一個 way 的 LRU 就會被設成 1。

II. 設計特色

Baseline

Register

我們 register 的部分不一樣的地方是我們的 register 是看 clock 的 negative edge，一開始是因為當 Data memory 寫回 register 的資料與讀取的是同一個的時候會來不及，為了避免資料來不及送到 Execution 而想出來的寫法。

後來發現這個半個 cycle 在 critical path 上非常傷，合成時 clock 能從原本 8ns 降到 4.7ns，但我覺得還是很值得參考，因此沒有還是保留了原本的設計。

PC Buffer

這個 Buffer 是為了 jal 指令在將 PC 存到$31 register 時而設計的。若照原本的設計，指令通過 ID 後，應該會再經過 EXE、MEM 再回到 register，PC 在這時候才會更新。在我們的 design 裡面，jal 指令要存入的 PC 會直接存入 register，但若此時 register 正在更新從 WriteBack 傳回的值，則先將 PC 存在 Buffer 中，等到 register idle 時再將其寫入。

Early Jump and Branch

這是我們後來覺得唯一有顯著 improvement 的 baseline 設計特色。原本的 Jump 以及 Branch 是在 Instruction 讀出來在 Instruction Decode 中做的，然後發生 Jump 或是 Branch 時直接更新 IF 的 PC，並清掉 IF 準備進到 ID 的 instruction，並沒有做任何預測，而是看到結果直接判斷 PC 應該要是多少。

後來我們發現直接把 Jump 以及 Branch 拉到 IF 做會更好。Jump(j)跟 Jump and Link 沒有問題，直接跳即可，而 Jump register(jr)仍必須等到 Instruction Decode 才能知道 PC，而

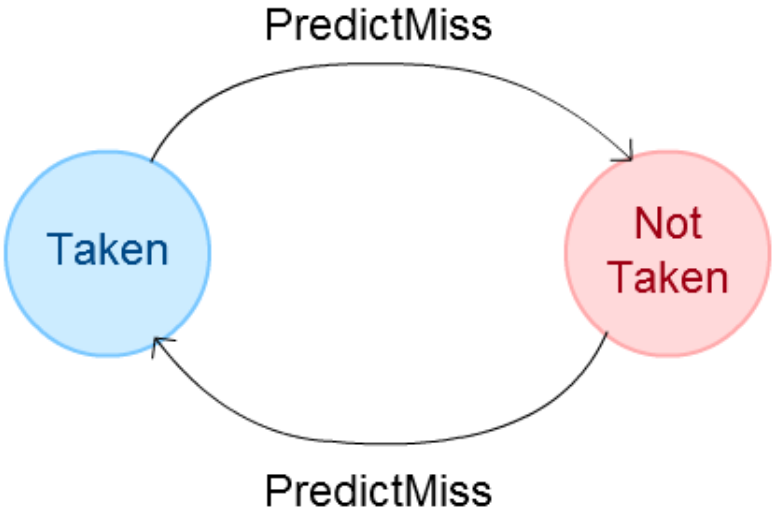
Branch 則必須進行預測，若猜對則繼續執行，猜錯了則在 Instruction Decode 更新正確的 PC 傳回 IF。 而 Branch Prediction Unit 的設計我們在下面會解說。

Extension

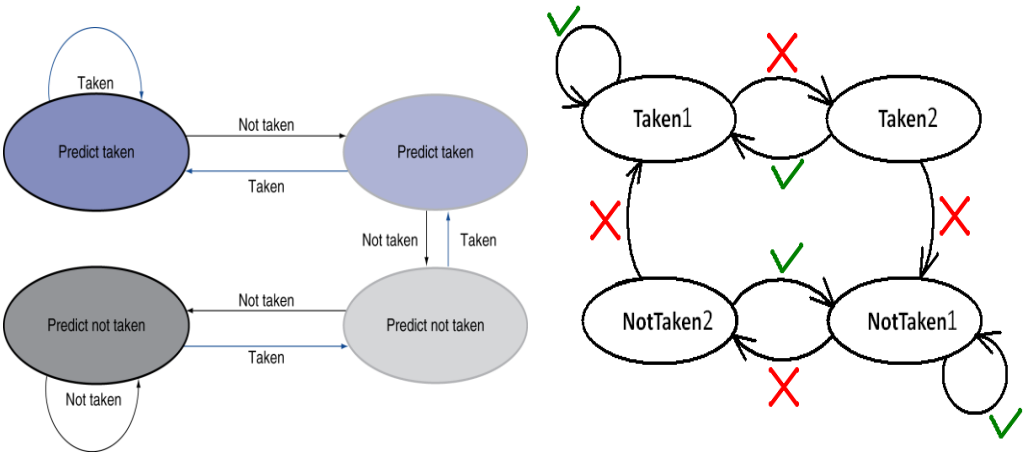
Branch Prediction

正確的 Prediction 在我們的 design 中能偷到 1 個 cycle，雖然不多，但若到現今許多市面上的處理器動輒 20 幾個 pipeline stage，instruction fetch 也拆成好幾個 stage，這時候正確的 prediction 就能賺到非常多，因此探討 Branch prediction 仍然是一項非常重要的議題。

我們實做了五種簡單的 prediction，其中 not take 其實就是 baseline 的設計。1-bit 的 prediction 的想法很簡單，其 FSM 如下圖。



2-bit predict 則有兩種，scheme 1(左)與 scheme 2(右)。



根據 FSM 的不同，預測的模型也會有所不同。以下是根據兩份我們比較這五種不同的預測方法的結果。

1. Testbench1(with 328 branch instruction)

	Not take	**Take***	**1-bit***	**2-bit SC1**	**2-bit SC2**
Exe. Cycle	3612.5	3783	3764	3628	3612.5
Miss rate	8.23%	83.1%	53.9%	11.9%	9.14%
Improve	-	-	1.8%	1.9%	2.3%

(*者數據可能因為 bug 不太準)

2. Testbench2(with 1000 branch instruction)

	Not take	**Take**	**1-bit**	**2-bit SC1**	**2-bit SC2**
Exe. Cycle	435.5	335.5	335.5	335.5	335.5
Miss rate	99%	1%	1%	1%	1%
Improve	-	-	13.0%	13.0%	13.0%

根據這兩份 testbench 我們可以發現，撇開亂猜的方式，動態預測中最佳者為 2-bit scheme 2，因此我們在合成模擬也只有提供這一種架構。我們可以發現這一塊 prediction unit 能夠提將 miss rate 大幅降低(假設不知道的情況下猜 not take 與 take 各半)，而面積也不算大。

2-bit branch prediction unit area: 2873.7(um^2)

L2 Cache

在現今許多計算機架構中，處理器的效能往往被卡在 memory 的讀寫上，因此插入 level 2 甚至 level 3 的 cache 用來加速運算。在這次的 design 中，我們實做了 level 2 的 cache，觀察 L2 Cache 為整體效能帶來的提升。

我們的 L2 Cache 架構模仿 L1，使用的是 direct map 的 placement policy，而其他規格有統一規定便不多談。我們直接看看這兩份 testbench 因 L2 的加入而帶來的改變。

1. Testbench1(basic instruction)

	L1 Cache	L2cache
Exe. Cycle	3612.5	2023.5
Improve	-	44.0%

2. Testbench2(advanced instruction)

	L1 Cache	L2cache
Exe. Cycle	4164.5	2423.5
Improve	-	41.8%

我們可以發現加入了 L2 Cache 之後，效能提升了至少 40%，是因為這兩份 testbench 中有許多 lw sw 的指令，因此能夠大幅提升效能。

L1 D-cache miss rate = 49/305=16.06%

L2 D-cache miss rate = 6/49=18.37%

Avg. memory access time

= Hit1+MR1*(Hit2+MR2*MP2)

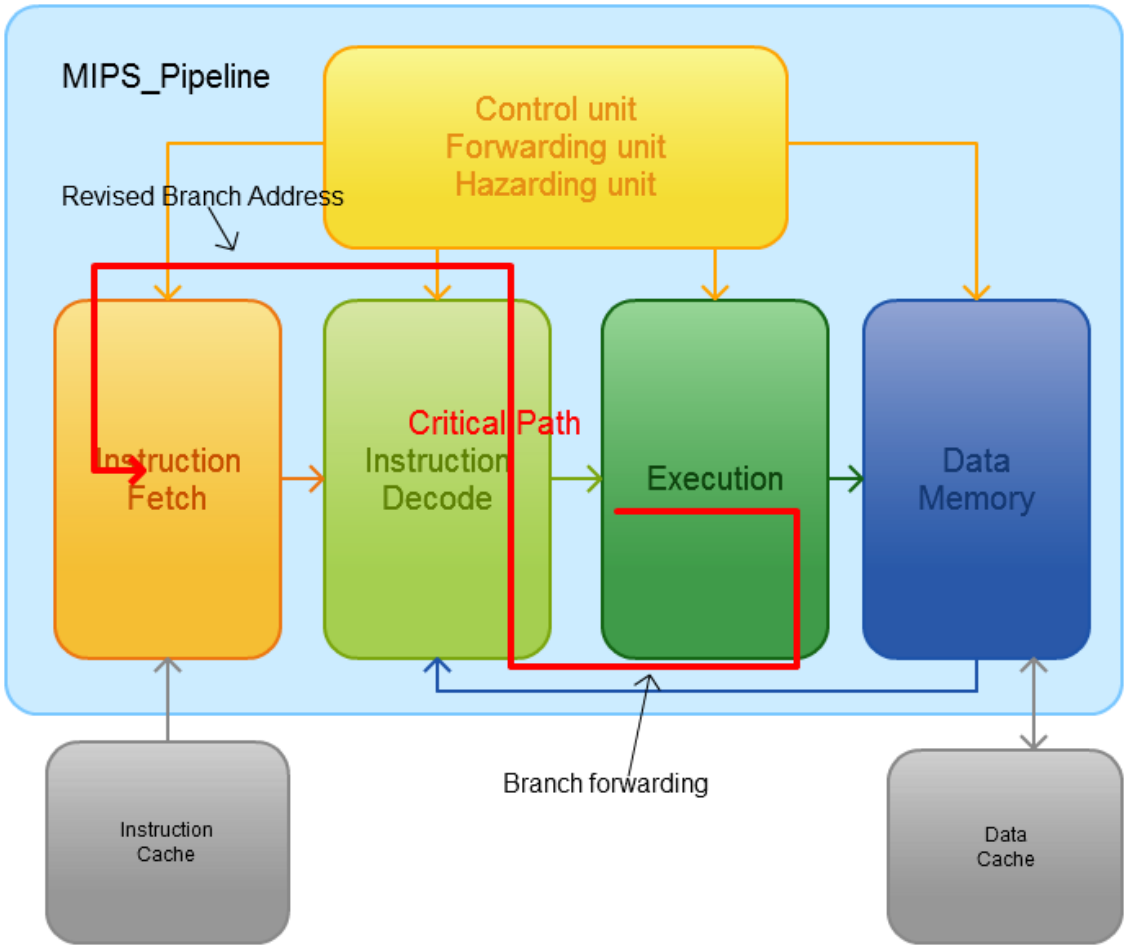
= 5.75+16.06%*(5.75+18.37%*15)

= 7.116 ns

	Testbench2
Avg. memory access time(ns)	7.116
Total execution time(ns)	14103.8

Critical Path

我們的 critical path 非常長



圖中紅色的路線即是 critical path。從 ID 的 register 出發，經過 Execution 的 ALU，被 branch forwarding 回 ID 判斷 branch 的 equal，如果 miss 則將 address 拉回 IF 的 PC 做更新。

critical path 會這麼長主要是因為我的兩個錯誤的 design methodology 造成的，第一個是 register 的反向，第二個是為了判斷 branch 是否 equal 的 forwarding。第一個我當初設計的時候沒想到，這個設計毫無條件的直接造成了半個 cycle 的 critical path，我後來曾經將 register 改回 positive edge trigger，critical path 降到了 4.7ns。第二個設計是 branch forwarding 從 ALU 的 output 拉回來，當初想說為了比較早判斷出 equal 才在 ID 判斷 equal，但沒想到造成了反效果，只有減少 cycle，卻

增長了 critical path。第二個設計解決的方法有兩個，一個是不要從 ALU 的 output 拉，而是從 Execution 的 output register 拉，就能忽略 Execution stage 中的時間；另一個是直接放棄在 Instruction Decode Stage 做 equal 的判斷，直接把檢查是否 miss 的 unit 拉到 execution，利用原本 data hazard 的 forwarding 做 equal 的判斷，這樣雖然在 branch miss 的時候會多造成一個 cycle，但整體的 critical path 卻能大幅度的降低，反而會比較快，也能省去不需要的 branch forwarding 的面積。

後來曾經花了一天把 design 改成上述後者，critical path 更降到了 3.6ns，然後因為跑不過 post-simulation，想到其他的要全部重合，一氣之下就把當天改的全砍了變回 presentation 時的樣子。後來想想原本錯誤的設計還是值得大家參考。

Ⅲ . 合成結果

Synthesis Results

1. Baseline

Area(um^2)

Combinational	68075.92
Noncombinational	44774.02
Total cell	112849.94

2. Extension-Branch Prediction

Area(um^2)

Combinational	70837.59
Noncombinational	44886.04
Total cell	115723.63

3. Extension-L2 Cache

Area(um^2)

Combinational	70766.30
Noncombinational	45003.16
Total cell	115769.46

DFT Insertion

One chain is inserted for total 1391 sequential cells.

Timing (ns)

data required time	10.17
data arrival time	-10.17
slack (MET)	0.00

Timing overhead: 0%

Area(um^2)

Combinational	76014.66
Noncombinational	60159.25
Total cell	1208252.99

Total cell area: 136173.91 (um^2)

Total area: 1344426.91 (um^2)

Area overhead: 18.4%

Function mode simulation：

```
Reading instruction memory......
Simulation complete via $finish(1) at time 20236259 PS + 0
./tb_L2Cache.v:282             wait(DCACHE_wen && DCACHE_wdata==32'hd5d && DCACHE_addr_o == 30'h100) $
finish;
ncsim> exit
[b99042@cad21 DFT_ATPG2]$
```

ATPG Result

```
TEST-T> report_summaries
      Uncollapsed Stuck Fault Summary Report
  -----------------------------------------------
  fault class                     code   #faults
  ------------------------------  ----  ---------
  Detected                         DT      69303
  Possibly detected                PT          0
  Undetectable                     UD       1181
  ATPG untestable                  AU          2
  Not detected                     ND        192
  -----------------------------------------------
  total faults                              70678
  test coverage                             99.72%
  fault coverage                            98.05%
  -----------------------------------------------
              Pattern Summary Report
  -----------------------------------------------
  #internal patterns                          272
      #basic_scan patterns                    272
  -----------------------------------------------
```

Warnings：

```
rule  severity  #fails  description *}
----  --------  ------  ---------------------------------- *}
N23   warning        2  inconsistent UDP *}
B7    warning        9  undriven module output pin *}
B8    warning       84  unconnected module input pin *}
B9    warning       21  undriven module internal net *}
B10   warning       51  unconnected module internal net *}
C6    warning      992  TE port captured data affected by new capture  (nomask) *}
C22   warning        2  clock as data for unstable cells  (nomask) *}
C25   warning        2  unstable cell clock input connected from multiple sources  (nomask) *}
```

做完 DFT 時有些 DRC violations（主要是 C6），推測是因為我們同時使用了 rising/falling edge trigger 的 Flip-Flop。這些 violations 可能造成 ATPG 的 fault coverage 下降，但是做完 ATPG 發現 fault coverage 仍有 98%，因此我們沒有將此 violation 修掉。若要修正此 violation，就將 Flip-Flop 都改為 rising edge trigger 即可。

Place & Route

Timing information：

1. pre-cts：

```
------------------------------------------------------------
     optDesign Final Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):|  0.975  |  0.975  |  2.040  | 14.173  |   N/A   |   N/A   |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |  0.000  |   N/A   |   N/A   |
|    Violating Paths:|    0    |    0    |    0    |    0    |   N/A   |   N/A   |
|          All Paths:|  5661   |  2779   |  3180   |   99    |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |    133 (266)     |
|   max_fanout   |      0 (0)       |     0      |      5 (5)       |
+----------------+------------------+------------+------------------+
```

2. post-cts：

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):|  0.823  |  0.823  |  1.275  | 14.928  |   N/A   |   N/A   |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |  0.000  |   N/A   |   N/A   |
|    Violating Paths:|    0    |    0    |    0    |    0    |   N/A   |   N/A   |
|          All Paths:|  5661   |  2779   |  3180   |   99    |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+

+----------------+-------------------------------+-------------------+
|                |              Real             |       Total       |
|    DRVs        +------------------+------------+-------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)   |
+----------------+------------------+------------+-------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)        |
|   max_tran     |      0 (0)       |   0.000    |     133 (266)     |
|   max_fanout   |      0 (0)       |     0      |     32 (32)       |
+----------------+------------------+------------+-------------------+
```

3. post-route(setup)：

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):|  0.985  |  0.985  |  1.212  | 15.061  |   N/A   |   N/A   |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |  0.000  |   N/A   |   N/A   |
|    Violating Paths:|    0    |    0    |    0    |    0    |   N/A   |   N/A   |
|          All Paths:|  5661   |  2779   |  3180   |   99    |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+

+----------------+-------------------------------+-------------------+
|                |              Real             |       Total       |
|    DRVs        +------------------+------------+-------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)   |
+----------------+------------------+------------+-------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)        |
|   max_tran     |      0 (0)       |   0.000    |     133 (266)     |
|   max_fanout   |      0 (0)       |     0      |     32 (32)       |
+----------------+------------------+------------+-------------------+
```

4. post-route(hold)：

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):|  0.140  |  0.140  |  2.914  |  1.024  |   N/A   |   N/A   |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |  0.000  |   N/A   |   N/A   |
|    Violating Paths:|    0    |    0    |    0    |    0    |   N/A   |   N/A   |
|          All Paths:|  5661   |  2779   |  3180   |   99    |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+
```

Power Planning：

power rings：

top/bottom layer 為 metal7，left/right layer 為 metal6，width 為 2 um，spacing 為 0.24 um。use wire group, interleaving，number of bits 為 15。

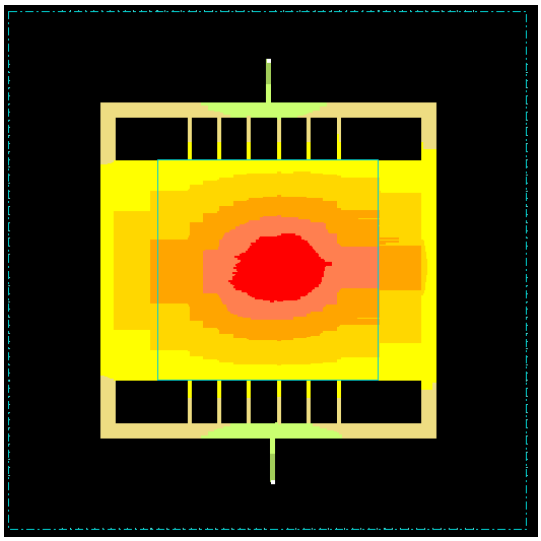
power strips：

使用 metal6，width 為 1 um，spacing 為 0.24 um。use wire group, interleaving，number of bits 為 5。總共加了六條 strips。

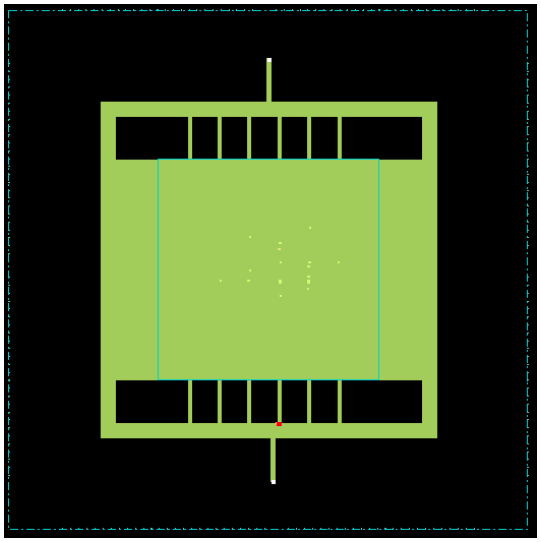
Report：(unit：mW)

```
Total Power
-----------------------------------------------
Total Internal Power:       150.8      97.93%
Total Switching Power:      3.039      1.974%
Total Leakage Power:      0.1468    0.09534%
Total Power:         154
-----------------------------------------------
```

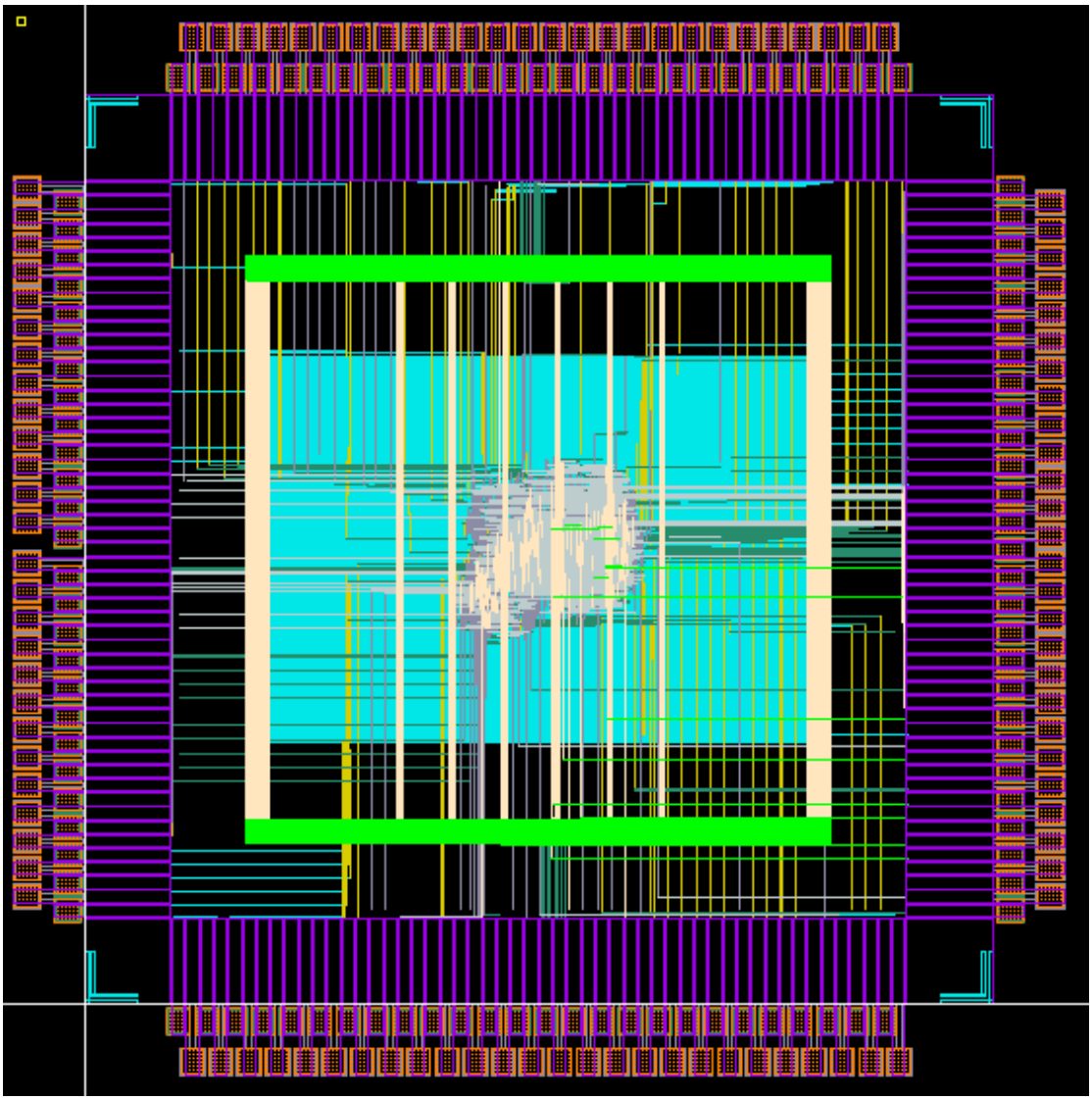
IR-drop：



Electron migration：



Final Layout Graph：



（有填 dummy metal，但必須拉近才會顯示）

Chip Area：2.6 × 2.6 (mm2)

Total I/O Pins：211(200 for function, 3 for test, 8 for power)

Post-layout simulation：

```
Simulation complete via $finish(1) at time 83230 NS + 4
./tb_L2Cache.v:282              wait(DCACHE_wen && DCACHE_wdata==32'hd5d && DCACHE_addr_o == 30'h100) $
finish;
ncsim> exit
[b99042@cad21 post_sim]$
```

DRC Result

```
RULECHECK OD.DN.1 ................ TOTAL Result Count = 1
RULECHECK OD.DN.3 ................ TOTAL Result Count = 1
RULECHECK PO.DN.1 ................ TOTAL Result Count = 1
RULECHECK M1.DN.1L ............... TOTAL Result Count = 1
RULECHECK M2.DN.1L ............... TOTAL Result Count = 1
RULECHECK M3.DN.1L ............... TOTAL Result Count = 1
RULECHECK M4.DN.1L ............... TOTAL Result Count = 1
RULECHECK M5.DN.1L ............... TOTAL Result Count = 1
RULECHECK M6.DN.1L ............... TOTAL Result Count = 1
RULECHECK M7.DN.1L ............... TOTAL Result Count = 1
RULECHECK M8.DN.1L ............... TOTAL Result Count = 1
RULECHECK DOD.R.1 ................ TOTAL Result Count = 1
RULECHECK DPO.R.1 ................ TOTAL Result Count = 1
[b99042@cad21 DRC]$
```

LVS Result

```
REPORT FILE NAME:         lvs.rep
LAYOUT NAME:              layout.spi ('CHIP')
SOURCE NAME:              source.spi ('CHIP')
RULE FILE:                CL013G_1P8M.lvs
HCELL FILE:               (-automatch)
CREATION TIME:            Fri Dec 27 16:56:30 2013
CURRENT DIRECTORY:        /home/raid5_3/userb99/b99042/CVSD/final/Layout/LVS
USER NAME:                b99042
CALIBRE VERSION:          v2012.2_26.20    Thu Jun 14 11:53:18 PDT 2012



                          OVERALL COMPARISON RESULTS



                     #       ##################          _   _
                    #        #                #         *   *
                  #   #      #     CORRECT    #           |
                   # #       #                #         \___/
                    #        ##################
```