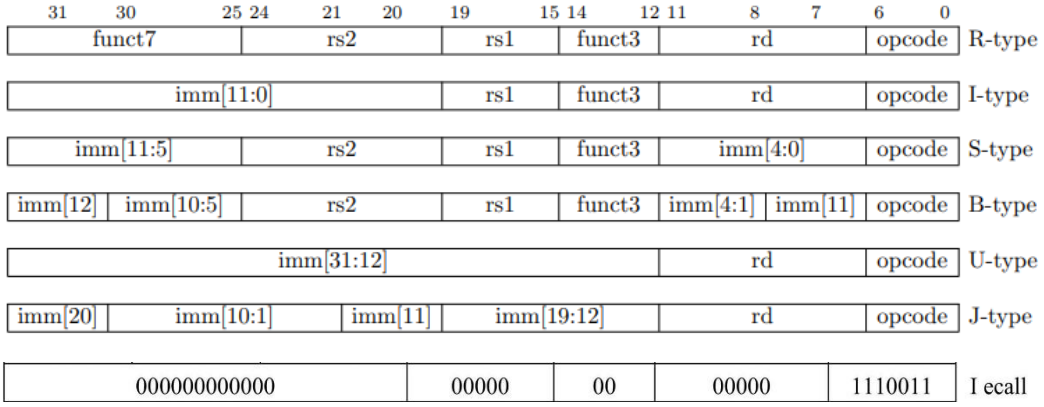
VLSI 系統設計
期末專題報告

組別:Wakuwaku		
姓名	學號	授課教師
王樟翔	E64094065	李昆忠 教授
游承嘉	E24116241	
鄭詰嚴	E14096724	
呂順瑋	E24094740	

目錄

一、 系統簡介

1. 指令集格式



圖一、 RV32I指令集格式

2. 指令集格式欄位的名稱、長度、說明

表一、 圖一指令集個欄位的名稱、長度與說明

名稱	長度	說明
opcode	7 bits	opcode 用來分辨指令的操作。
rd	5 bits	Register Destination 是指令的結果存放的目的地地址。
rs1	5 bits	Register Source 1。這是執行指令所需的第一個操作數地址。
rs2	5 bits	Register Source 2。這是執行指令所需的第一個操作數地址。
funct3	3 bits	額外功能字段。在某些指令中，用於進一步分辨不同的用途。
funct7	7 bits	額外功能字段。與 funct3 功能類似
imm	12 or 20bits	用於計算的立即數

3. Branch 指令與 Jump 指令的地址

Branch 指令：

Branch 指令的地址計算基於相對地址和 pc 的值。計算的方式是將相對地址加上當前指令的地址，生成分支目標地址。如果分支條件滿足，則 pc 會被更新為該目標地址，進行跳轉。

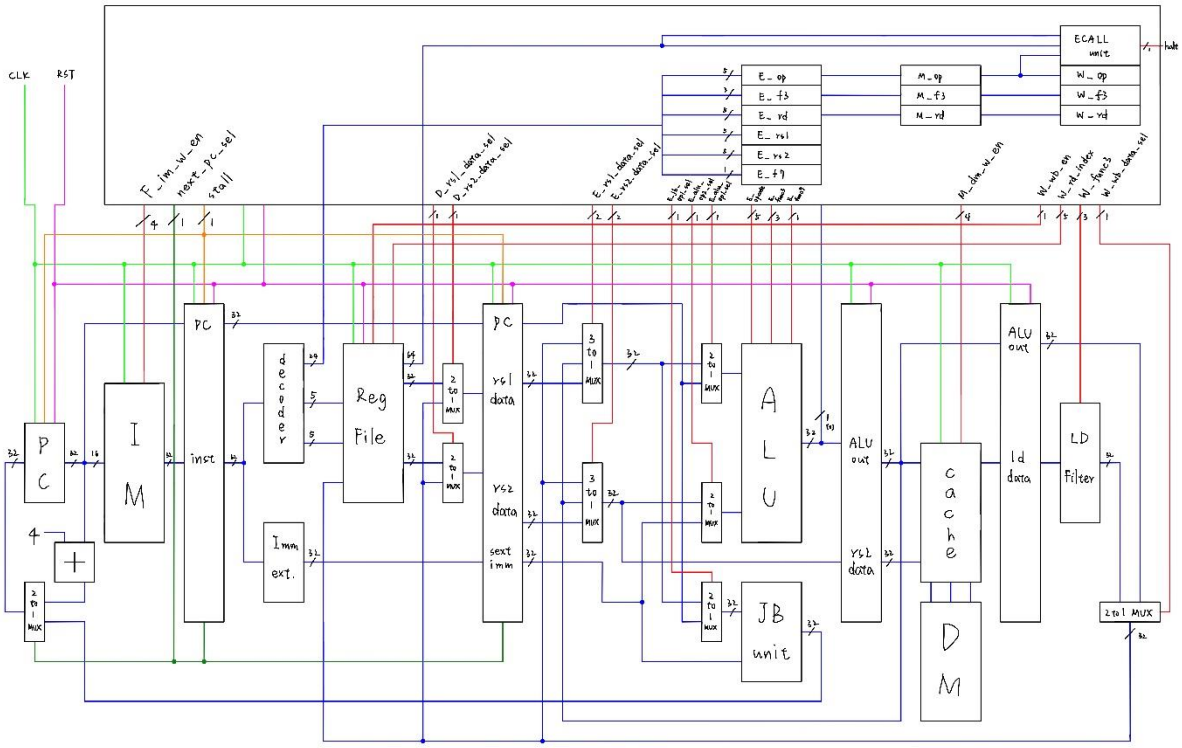
Jump 指令：

Jump 指令的地址計算基於相對地址和 pc 的值。計算的方式是將相對地址加上當前指令的地址，生成跳轉目標地址。同時，Jump 指令

會將下一條指令的地址保存到鏈接暫存器中，通常是 x1，以實現跳轉後的返回地址的記錄。

4. 架構說明

i. 架構圖及說明(需自行繪製)



圖二、 CPU 架構圖(加上 cache)

ii. 個別元件說明

Adder:

負責下一個 pc 的運算。

ALU:

ALU 實現了多種算術和邏輯運算，包括加法、減法、位移、比較等。根據不同的指令碼和功能碼，選擇不同的操作，並提供廣泛的運算功能。

Controller:

這個 Controller module 執行對 CPU 各階段的控制邏輯，根據當前指令的 opcode、func3、func7 等訊號，輸出下一步的控制信號，包括是否停頓、下一個 PC 的選擇、各階段對暫存器的操作等。同時它也處理了一些跳轉指令的相對應邏輯，確保指令能夠正確順序地在不同階段執行。

Decoder:

將一條完整的指令提取其中所有可能的資訊，包括指令識別，暫存器位置等等，並且輸入 Controller 做進一步訊號控制。

Imme_Ext:

根據不同指令提取其中立即數的部分。

JB_Unit:

計算跳轉後的執行位置。

LD_Filter:

把記憶體中下載的資料，根據 func3 擷取所需長度。

Mux2:

二選一多工器，根據 controller 的控制訊號選擇所需資料，用於大部分的資料選擇。

Mux3:

三選一多工器，根據 controller 的控制訊號選擇所需資料，此設計只用於 EX 階段中，資料提前使用的部分。

RegFile:

於 CPU 內暫存資料與讀取的地方。

Reg_PC:

用於暫存下一個 pc，，並且輸入 stall 控制訊號來預防 hazard 的產生。

Reg_D:

用於 IF 與 ID 階段之間的傳輸資料暫存，暫存的資料有，pc 和指令 inst，並且輸入 stall 和 jb 控制訊號分別來進行 hazard 的預防與 branch prediction。

Reg_E:

用於 ID 與 EX 階段之間的傳輸資料暫存，暫存的資料有，pc、rs1_data、rs2_data 和立即數，並且輸入 stall 和 jb 控制訊號分別來進行 hazard 的預防與 branch prediction。

Reg_M:

用於 EX 與 MEM 階段之間的傳輸資料暫存，暫存的資料有，ALU 運算結果和 rs2_data。

Reg_W

用於 MEM 與 WB 階段之間的傳輸資料暫存，暫存的資料有，ALU 運算結果和記憶體下載的資料。

二、 系統目前可執行之指令

R type：

ADD, SUB, SLL, SLT, SLTU, XOR, SRL, SRA, OR, AND

功能： 對兩個暫存器執行整數運算。

分類： 算術型指令。

I type：

JALR, LB, LH, LW, LBU, LHU, ADDI, SLTI, SLTIU, XORI, ORI, ANDI, SLLI, SRLI, SRAI

功能： 處理立即數，進行載入、加法、位元運算等操作。

分類： 載入型、算術型、位移型指令。

S type：

SB, SH, SW

功能： 將數據存儲到記憶體。

分類： 存儲型指令。

B type：

BEQ, BNE, BLT, BGE, BLTU, BGEU

功能： 根據條件執行分支。

分類： 分支型指令。

U type：

LUI, AUIPC

功能： 將一個立即數的上半部分載入目的暫存器。

分類： 載入型指令。

J type：

JAL

功能： 無條件跳轉，同時將返回地址載入目的暫存器。

分類： 分支型指令。

三、 系統驗證方法與結果分析

1. 驗證方法

驗證方法是自動執行所有系統指令的測試檔案，在每條指令分別驗證前將 rst（重置）信號置為 1，然後載入每個指令的測試檔案，將指令及相應的數據存儲到指令存儲器和數據存儲器中開始測試。接著等待 halt 信號置為 1，表示處理器執行完成。

在針對單一指令測試完成後使用 check task 檢查處理器的結果。檢查 GP（通用目的暫存器 3）是否為 1，A2（暫存器 12）是否為 0，來判斷測試是否通過。檢查通過的話，將 flag 設置為 TRUE，否則為 FALSE。並且使用$display 在控制台上顯示每個測試的結果，經由 flag 判斷通過顯示"Passed"，否則顯示"Failed"。

執行完一條系統指令之後會將處理器的 rst 信號重新置為 1，然後進行下一個測試，直到完成所有的測試用例。每個測試用例的指令和數據存儲器初始化後都會執行相應的指令，並等待 halt 信號。

2. 結果分析(需附上 nWave 截圖)

i. “單一”指令正確性

```
Chronologic VCS simulator copyright 1991-2020
Contains Synopsys proprietary information.
Compiler version R-2020.12-SP2_Full64; Runtime version R-2020.12-SP2_Full64;  Jan  7 13:19 2024
lui   ...... Passed
auipc ...... Passed
jal   ...... Passed
jalr  ...... Passed
beq   ...... Passed
bne   ...... Passed
blt   ...... Passed
bge   ...... Passed
bltu  ...... Passed
bgeu  ...... Passed
lb    ...... Passed
lh    ...... Passed
lw    ...... Passed
lbu   ...... Passed
lhu   ...... Passed
sb    ...... Passed
sh    ...... Passed
sw    ...... Passed
addi  ...... Passed
slti  ...... Passed
sltiu ...... Passed
xori  ...... Passed
ori   ...... Passed
andi  ...... Passed
slli  ...... Passed
srli  ...... Passed
srai  ...... Passed
add   ...... Passed
sub   ...... Passed
sll   ...... Passed
slt   ...... Passed
sltu  ...... Passed
xor   ...... Passed
srl   ...... Passed
sra   ...... Passed
or    ...... Passed
and   ...... Passed
sudoku...... Passed
fibo  ...... Passed
sort  ...... Passed
```

圖三、 testbench 測試結果

ii. 程式正確性

A. 排序(需自行撰寫組合語言)

```
case_1:
    .word 809, -451, 723, -917, 148, 632, -295, -877, 388, -522, -321, 999, -773, -27, 447, 104, -651, 762, -816, 273
case_2:
    .word 177, -888, -469, -412, 691, -282, 365, -122, 480, -707, -846, 324, -589, 833, 266, -943, 194, -614, -38, 573
case_3:
    .word -786, 209, -369, 567, -703, -118, -863, 949, -521, -697, 172, 816, -32, -676, 447, 89, 933, 655, -185, -293
case_4:
    .word -987, 236, -74, -868, 328, -560, -316, 117, -435, -849, 643, 578, 985, 707, 285, -327, 741, -656, 408, 0
case_5:
    .word 840, 222, 624, -848, -497, -920, -661, 565, -550, -801, -625, 761, 399, 993, 1000, -453, -355, -145, -954, 820
case_6:
    .word 316, -783, 510, 651, -386, 281, 867, 939, 477, -999, -175, -689, -824, -707, -28, 458, -569, 605, -944, -370
case_7:
    .word -771, 758, 430, -298, 761, -685, -596, -88, 304, 524, -124, -480, -356, -351, 27, -226, 134, -912, 730, -330
case_8:
    .word -674, -583, 175, 881, 621, -544, 914, -90, -777, 993, -870, 232, 490, -667, -192, 861, 566, -997, 447, 314
case_9:
    .word -438, -689, 905, 172, 466, 632, -665, 31, 745, -220, -579, 388, 899, 772, -481, 122, -327, -51, -611, -773
case_10:
    .word 1000, 8, 7, 6, 5, 4, 3, 2, 1, 0, -1, -2, -3, -4, -5, -6, -7, -8, -9, -1000
```

```
 -917   -877   -816   -773   -651   -522   -451   -321   -295    -27    104    148    273    388    447    632    723    762    809    999
 -943   -888   -846   -707   -614   -589   -469   -412   -282   -122    -38    177    194    266    324    365    480    573    691    833
 -863   -786   -703   -697   -676   -521   -369   -293   -185   -118    -32     89    172    209    447    567    655    816    933    949
 -987   -868   -849   -656   -560   -435   -327   -316    -74      0    117    236    285    328    408    578    643    707    741    985
 -954   -920   -848   -881   -661   -625   -550   -497   -453   -355   -145    222    399    565    624    761    820    840    993   1000
 -999   -944   -824   -783   -707   -689   -569   -386   -370   -175    -28    281    316    458    477    510    605    651    867    939
 -912   -771   -685   -596   -480   -356   -351   -330   -298   -226   -124    -88     27    134    304    430    524    730    758    761
 -997   -870   -777   -674   -667   -583   -544   -192    -90    175    232    314    447    490    566    621    861    881    914    993
 -773   -689   -665   -611   -579   -481   -438   -327   -220    -51     31    122    172    388    466    632    745    772    899    905
-1000     -9     -8     -7     -6     -5     -4     -3     -2     -1      0      1      2      3      4      5      6      7      8   1000
```

圖四、 排序測試結果

B. 費氏數列

0 -> 21

34 -> 377

610 -> 2584

4181 -> 6765

四、 SuperLint 與 ICC 檢查結果

1. SuperLint 檢查結果(需附上 SuperLint 截圖)

```
============================================================
SUMMARY
============================================================
          Properties Considered              : 2
               assertions                    : 2
                - proven                     : 0 (0%)
                - bounded_proven (user)      : 0 (0%)
                - bounded_proven (auto)      : 0 (0%)
                - marked_proven              : 0 (0%)
                - cex                        : 0 (0%)
                - ar_cex                     : 0 (0%)
                - undetermined               : 0 (0%)
                - unknown                    : 2 (100%)
                - error                      : 0 (0%)
               covers                        : 0
                - unreachable                : 0
                - bounded_unreachable (user): 0
                - covered                    : 0
                - ar_covered                 : 0
                - undetermined               : 0
                - unknown                    : 0
                - error                      : 0
```

```
INFO (ISL018): Violation Count: Errors = 0 Warnings = 125 Info = 4
159
[<embedded>] %
[<embedded>] % # Prove
[<embedded>] % set_superlint_prove_parallel_tasks_on
[<embedded>] % set_prove_no_traces_true
[<embedded>] % check_superlint -prove -time_limit 10m -bg
INFO (ISL008): Running multiple proof threads in parallel, one per task.
    Expect high resource allocation.
WARNING (WG002): "superlint_prove_parallel_tasks" is included as an initial release to gather feedback from early adopters
WARNING (WSL054): "superlint_prove_parallel_tasks" is enabled, overriding the following settings: proofgrid_per_engine_max
    For message help, type "help -message WSL054".
INFO (IPF127): Verbosity level: 6
INFO (IPF127): Using Proof Grid in local mode, with 1 licenses
INFO (IPF127): Trace generation: Suppressing
INFO (IPF127): Engine C/G memory limit: 2048MB
INFO (IPF127): Using rule superlint_chunking_rule.
INFO (IPF127): (traces_plan) Using action: prove_action.
INFO (IPF036): Starting proof on task: "<SL_AUTO_FORMAL_FSM>", 14 properties to prove with 0 already proven/unreachable
INFO (IPF031): Settings used for proof thread 0:
    orchestration                = off
    time_limit                   = 599s
    per_property_time_limit      = 10s * 10 ^ scan
    engine_mode                  = Ht J N I L
    proofgrid_per_engine_max_jobs = 1
    proofgrid_max_jobs           = 35
    max engine jobs              = Ht J N I L, total 5 (max 35)
    proofgrid_mode               = local
    proofgrid_restarts           = 10
    trace_generation             = off
```

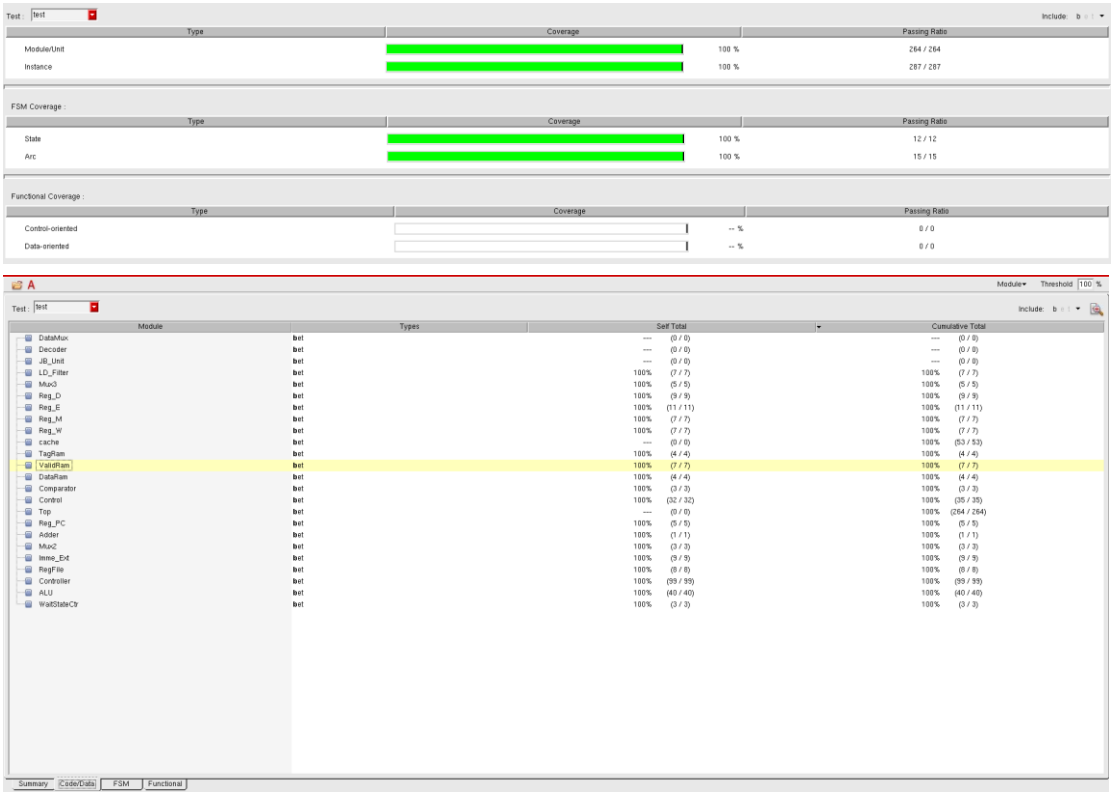
圖五、 superlint 結果(加上 cache)

superlint 0 Errors, 125 Warnings

圖六、 cache testbench 結果(加上 cache)

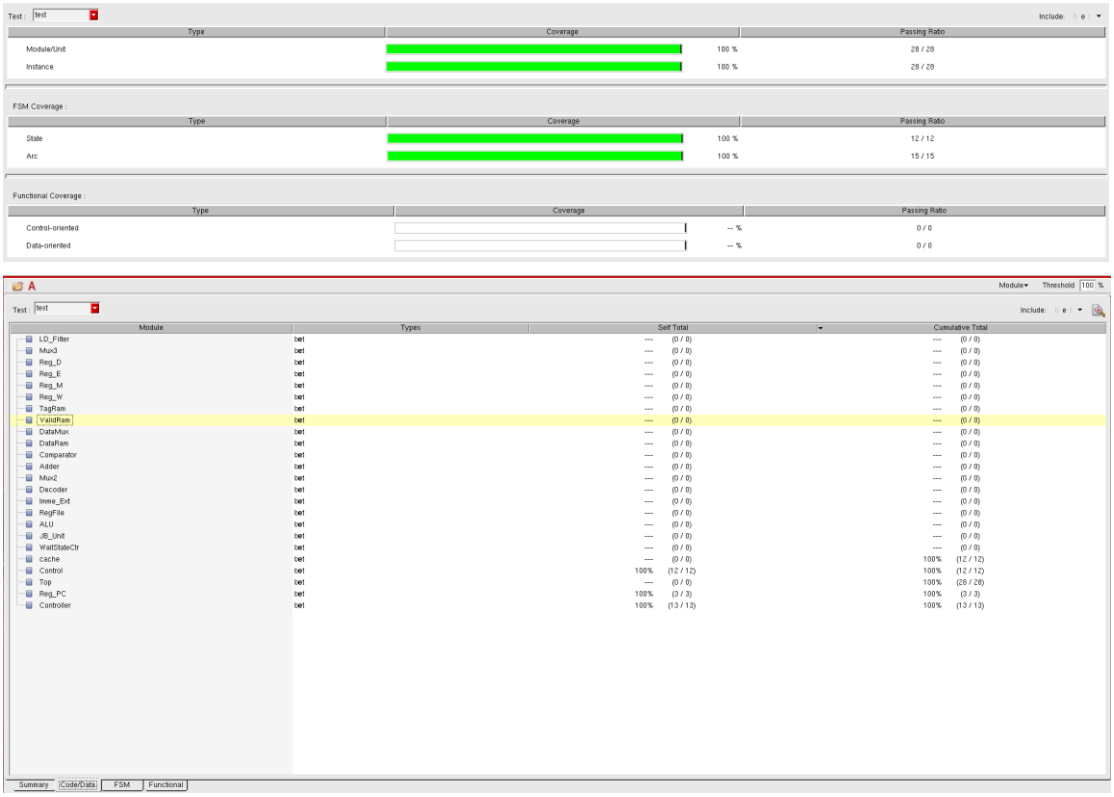
2. ICC 檢查結果(需附上 ICC 截圖)

Block Coverage



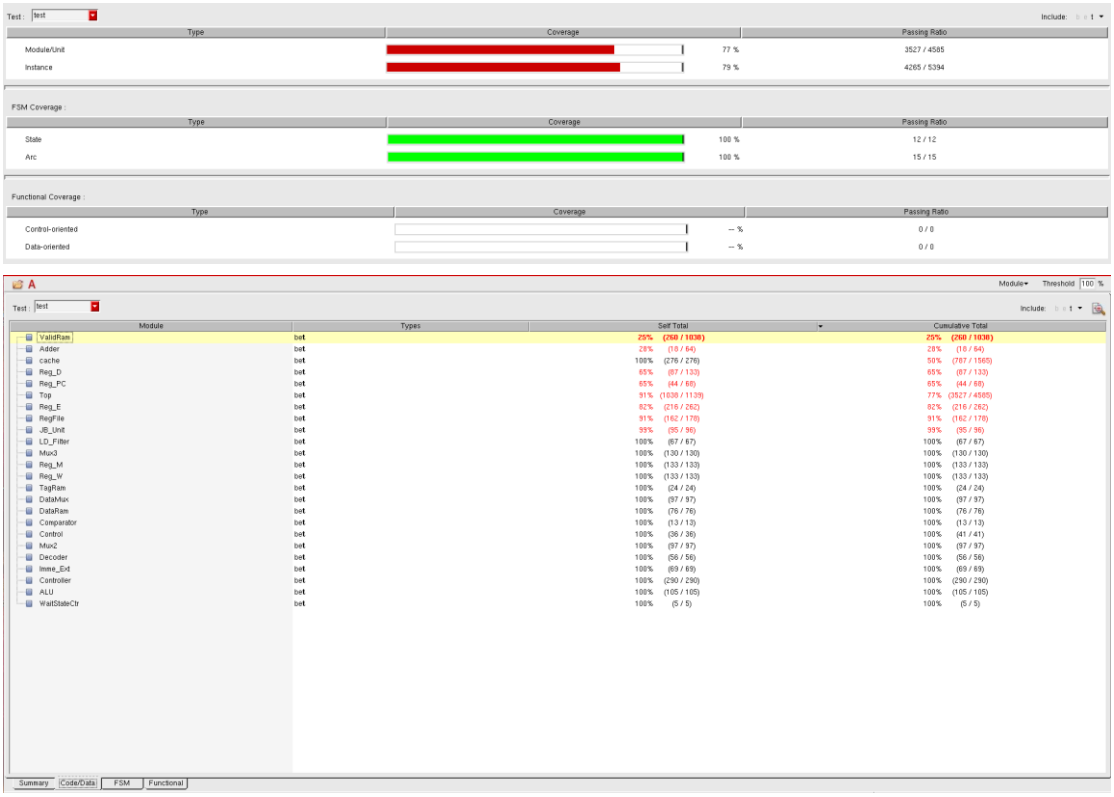
圖七、 Block Coverage 結果(加上 cache)

Expression Coverage



圖八、 Expression Coverage 結果(加上 cache)

Toggle Coverage



圖九、 Toggle Coverage 結果(加上 cache)

五、 合成結果

未加上 cache 合成成功
加上 cache 合成失敗

1. 速度(需附上截圖，Setup time 和 Hold time slack 都>0)

Period: 40ns
Frequency:2.5*10^7Hz

```
****************************************
Report : timing
        -path full
        -delay min
        -max_paths 1
        -sort_by group
Design : Top
Version: Q-2019.12
Date   : Sat Dec 30 11:57:32 2023
****************************************

 # A fanout number of 1000 was used for high fanout net computations.

Operating Conditions: fast   Library: fast
Wire Load Model Mode: top

  Startpoint: controller/M_op_reg_0_
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: controller/W_op_reg_0_
            (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: min

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  Top                tsmc18_wl10           slow

  Point                                                   Incr       Path
  ------------------------------------------------------------------------
  clock clk (rise edge)                                  20.00      20.00
  clock network delay (ideal)                             2.00      22.00
  controller/M_op_reg_0_/CK (DFFRHQX1)                    0.00 #    22.00 r
  controller/M_op_reg_0_/Q (DFFRHQX1)                     0.39      22.39 f
  controller/W_op_reg_0_/D (DFFRHQX1)                     0.00      22.39 f
  data arrival time                                                 22.39

  clock clk (rise edge)                                  20.00      20.00
  clock network delay (ideal)                             2.00      22.00
  controller/W_op_reg_0_/CK (DFFRHQX1)                    0.00      22.00 r
  library hold time                                      -0.07      21.93
  data required time                                                21.93
  ------------------------------------------------------------------------
  data required time                                                21.93
  data arrival time                                                -22.39
  ------------------------------------------------------------------------
  slack (MET)                                                       0.46
```

圖十、Max Timing report 結果(未加上 cache)

Hold Time slack (min) is 0.46 ns

```
clock clk (rise edge)                                  60.00      60.00
clock network delay (ideal)                             2.00      62.00
clock uncertainty                                      -0.10      61.90
regpc/current_pc_reg_9_/CK (DFFRHQX1)                   0.00      61.90 r
library setup time                                     -0.23      61.67
data required time                                                61.67
-------------------------------------------------------------------------
data required time                                                61.67
data arrival time                                                -59.70
-------------------------------------------------------------------------
slack (MET)                                                       1.98
```

圖十一、Min Timing report 結果(未加上 cache)

Setup Time slack(Max) is 1.98 ns

2. 面積(需附上截圖)

```
****************************************
Report : area
Design : Top
Version: Q-2019.12
Date   : Sat Dec 30 11:57:32 2023
****************************************

Library(s) Used:

    slow (File: /home/ncku_class/vsd2023/vsd202300/Desktop/vsd2023/synopsys/slow.db)

Number of ports:                         3077
Number of nets:                         10043
Number of cells:                         7093
Number of combinational cells:           5623
Number of sequential cells:              1430
Number of macros/black boxes:               0
Number of buf/inv:                       1115
Number of references:                      23

Combinational area:              96661.857459
Buf/Inv area:                     7890.220899
Noncombinational area:           98687.636108
Macro/Black Box area:                0.000000
Net Interconnect area:         1002965.052826

Total cell area:                195349.493567
Total area:                    1198314.546393
1
```

圖十二、Area report 結果(未加上 cache)

Total cell area is 195349.493567 um2, Gate counts is 19535 units

3. 功耗(需附上截圖)

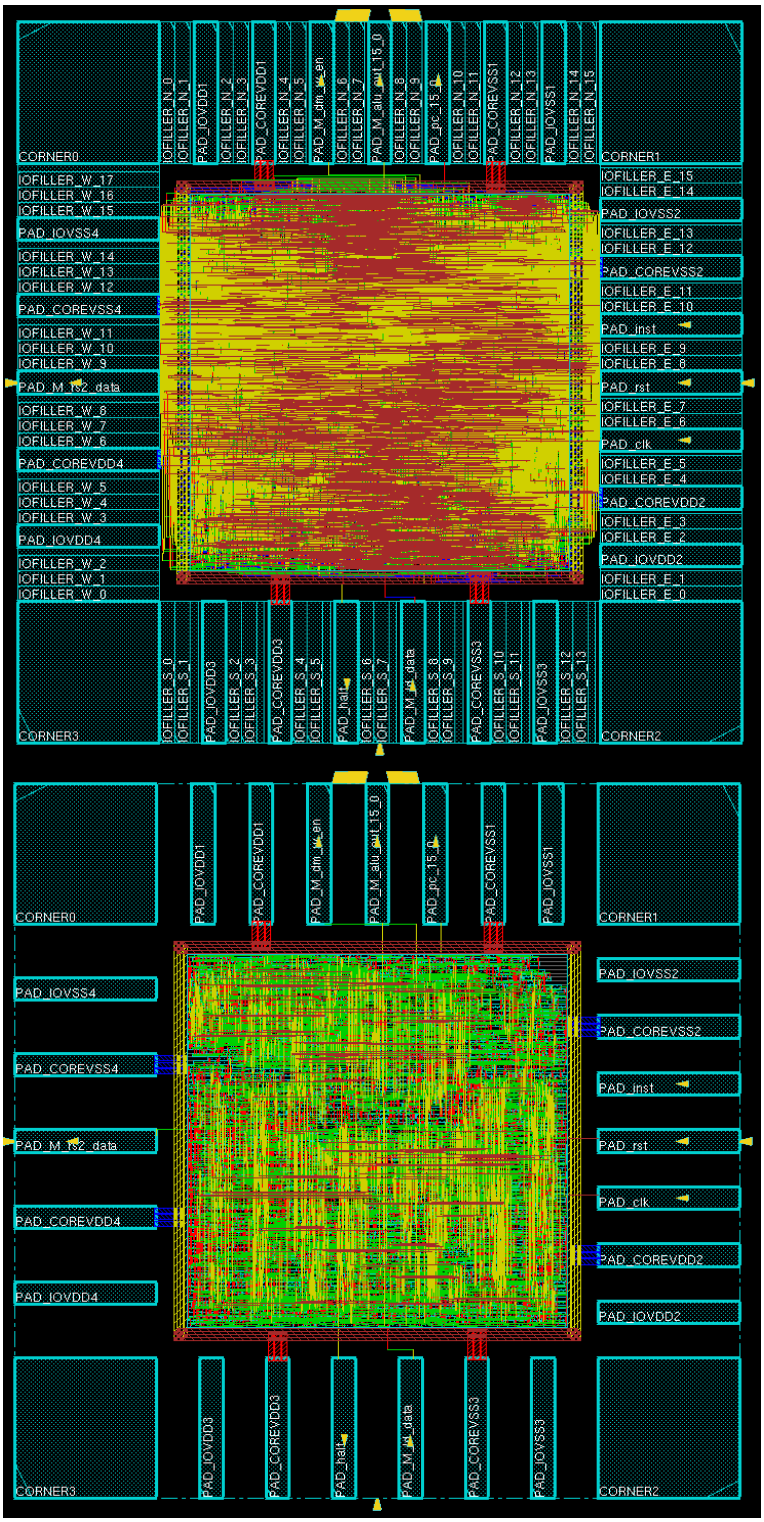
```
****************************************
Report : power
        -analysis_effort low
Design : Top
Version: Q-2019.12
Date   : Sat Dec 30 11:57:34 2023
****************************************


Library(s) Used:

    slow (File: /home/ncku_class/vsd2023/vsd202300/Desktop/vsd2023/synopsys/slow.db)


Operating Conditions: slow   Library: slow
Wire Load Model Mode: top

Design        Wire Load Model            Library
------------------------------------------------
Top                    tsmc18_w110       slow


Global Operating Voltage = 1.62
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


  Cell Internal Power  =   2.1311 mW   (87%)
  Net Switching Power  = 327.6532 uW   (13%)
                         ---------
Total Dynamic Power    =   2.4587 mW  (100%)

Cell Leakage Power     =   6.0786 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    ) Attrs
------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000             0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000             0.0000  (   0.00%)
register           2.0579        2.6233e-02        3.0953e+06             2.0872  (  84.68%)
sequential      3.8417e-03        2.6533e-03        2.7721e+04          6.5227e-03 (   0.26%)
combinational   6.9357e-02           0.2988         2.9556e+06             0.3711  (  15.06%)
------------------------------------------------------------------------------------------------
Total              2.1311 mW        0.3277 mW     6.0786e+06 pW          2.4648 mW
1
```

圖十三、Min Timing report 結果(未加上 cache)

Total Power is 2.4648 mW

六、 Layout 結果

1. (需附上截圖)



圖十四、Layout 結果(未加上 cache)

```
============================
Floorplan/Placement Information
============================
Total area of Standard cells: 244164.413 um^2
Total area of Standard cells(Subtracting Physical Cells): 231983.136 um^2
Total area of Macros: 0.000 um^2
Total area of Blockages: 0.000 um^2
Total area of Pad cells: 561839.450 um^2
Total area of Core: 244186.639 um^2
Total area of Chip: 891593.858 um^2
Effective Utilization: 1.0000e+00
Number of Cell Rows: 98
% Pure Gate Density #1 (Subtracting BLOCKAGES): 99.991%
% Pure Gate Density #2 (Subtracting BLOCKAGES and Physical Cells): 95.002%
% Pure Gate Density #3 (Subtracting MACROS): 99.991%
% Pure Gate Density #4 (Subtracting MACROS and Physical Cells): 95.002%
% Pure Gate Density #5 (Subtracting MACROS and BLOCKAGES): 99.991%
% Pure Gate Density #6 (Subtracting MACROS and BLOCKAGES and Physical Cells): 95.002%
% Core Density (Counting Std Cells and MACROs): 99.991%
% Core Density #2(Subtracting Physical Cells): 95.002%
% Chip Density (Counting Std Cells and MACROs and IOs): 90.400%
% Chip Density #2(Subtracting Physical Cells): 89.034%
# Macros within 5 sites of IO pad: No
Macro halo defined?: No
```

圖十五、Layout Area 結果(未加上 cache)

```
                ################################################
                ##                                            ##
                ##        C A L I B R E   S Y S T E M         ##
                ##                                            ##
                ##            L V S   R E P O R T             ##
                ##                                            ##
                ################################################



REPORT FILE NAME:         lvs.rep
LAYOUT NAME:              CHIP.spi ('chip')
SOURCE NAME:              CHIP.spi ('chip')
RULE FILE:                Calibre-lvs-cur
HCELL FILE:               (-automatch)
CREATION TIME:            Sat Dec 30 16:33:34 2023
CURRENT DIRECTORY:        /project/dr562/pj12/pj1217/successful/Calibre/lvs
USER NAME:                pj1217
CALIBRE VERSION:          v2020.2_14.12    Thu Apr 2 15:39:27 PDT 2020


                          OVERALL COMPARISON RESULTS


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


  Warning:  Ambiguity points were found and resolved arbitrarily.
  Warning:  LVS property resolution maximum exceeded.
  Warning:  Source and layout refer to the same data.
```

圖十六、LVS report 結果(未加上 cache)

七、 管線化

1. 設計說明

管線化設計是一種將處理器執行指令的不同階段分開執行的架構。這種設計允許多個指令在同一時間進行不同的階段，從而提高處理器的效能。在我們的設計中總共有五個階段，分別是抓取指令（IF）、解碼（ID）、執行（EX）、記憶體存取（MEM）和寫回（WB）等階段。

2. 階段的劃分

IF 階段： 從指令記憶體中讀取指令。

ID 階段： 解析指令，確定操作數，並從暫存器中取得需要的數據。

EX 階段： 執行指令，進行運算或位元操作。

MEM 階段： 存取記憶體，例如讀取或寫入數據。

WB 階段： 把執行結果寫回暫存器。

3. 管線化的危障(需說明解決方法)

- Data Hazard：就是當要做運算時沒有辦法拿到最新的資料，直接拿到的資料不是最新的資料，解決方式是將最新的資料從 MEM Stage 或 WB Stage forward 到 ID Stage 或 EX Stage。

- Control Hazard：當遇到 jump 或當 branch 在 ALU 計算出來須要執行跳轉時，會有錯誤的指令已經開始執行，解決方式是需要實作一個 flush 的訊號，可以清除掉錯誤的指令，並拿取正確新的指令。

- Structural Hazard： 硬體資源不夠，在同時 fetch 指令和 load 資料時會同時使用到 memory，但不可能同時讀取兩個資料出來。解決方式是將 SRAM 分成 im 和 dm，就可以同時 fetch 指令和 load 資料。

八、 特殊設計(例如 cache、支援浮點數運算等額外設計都可提出)

1. Branch Prediction:

利用當前指令的操作碼（E_op）進行檢查，生成 next_pc_sel 信號。如果當前指令是分支或跳躍指令如 JAL、JALR 或 BRANCH，則根據特定條件判斷是否需要進行跳轉。

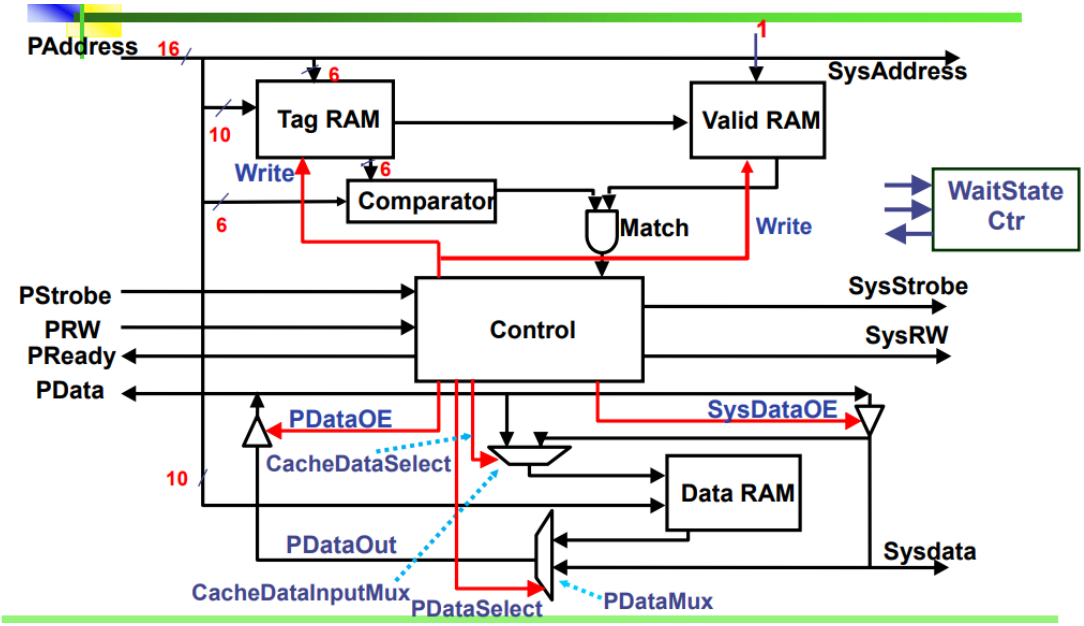
2. Cache

加入 cache 實現了基本的直接映射快取，用於提高對存儲器的訪問效率。Cache 的目的是在存儲器和 CPU 之間提供一個快速的緩存，

以減少存儲器訪問的延遲。根據當前訪問的地址，TagRam 和 Comparator 確定是否命中快取，如果命中，則根據 Control 模組的控制，選擇是讀取快取數據還是將數據寫入快取。

我們的 Cache 是採用 direct map 和 write through 的寫法，有參考老師的講義，但有做修正。

架構是參考老師的講義



圖十七、講義中 cache 架構圖(加上 cache)

其中有把 buffer 改成 MUX，控制訊號為 0 時，輸出為 0

PData 拆成兩條線，分別為 PData 為輸入，另一條是 OutData 為輸出

Sysdata 也拆成兩條線，一條為 Sysdata 為輸出，CData 為輸入

TagRam： 負責存儲每個快取組的標記（tag），以便比較當前訪問的地址是否匹配快取中的數據。

ValidRam： 存儲每個快取組的有效位，用於指示對應的快取組是否包含有效的數據。

DataMux： 用於選擇要寫入快取的數據，可以是來自外部的數據（CData）或先前從快取讀取的數據。

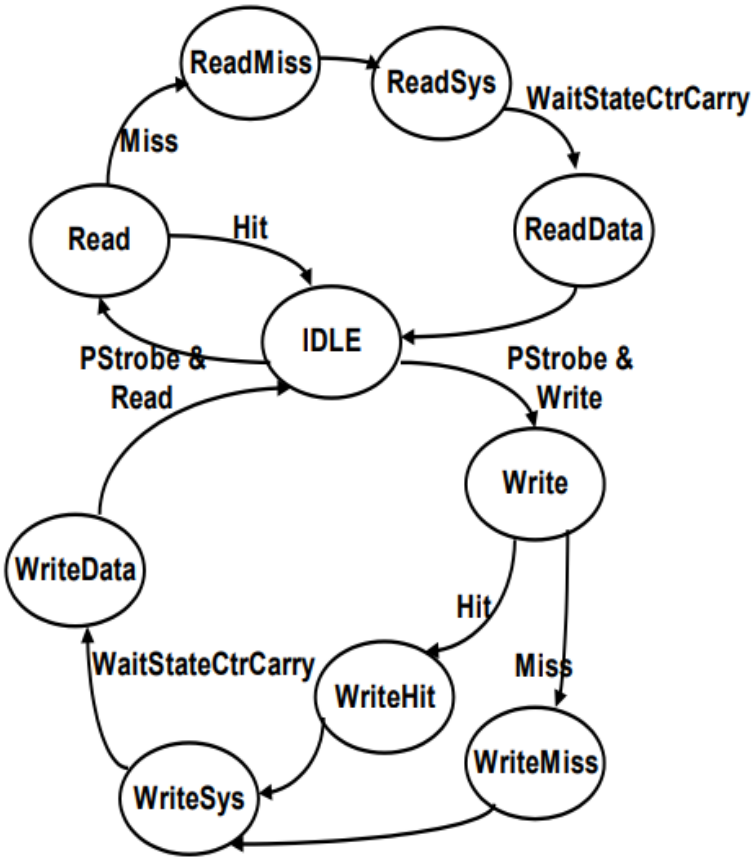
DataRam： 存儲實際的數據內容，每個快取組有一個對應的 DataRam。

Comparator： 用於比較快取中的標記（tag）與當前訪問的地址的標記是否匹配。

Control： 控制整個 cache 模組的行為，根據比較結果、有效位等信號，確定是否從快取讀取數據、將數據寫入快取，以及是否將數據寫入系統存儲器。

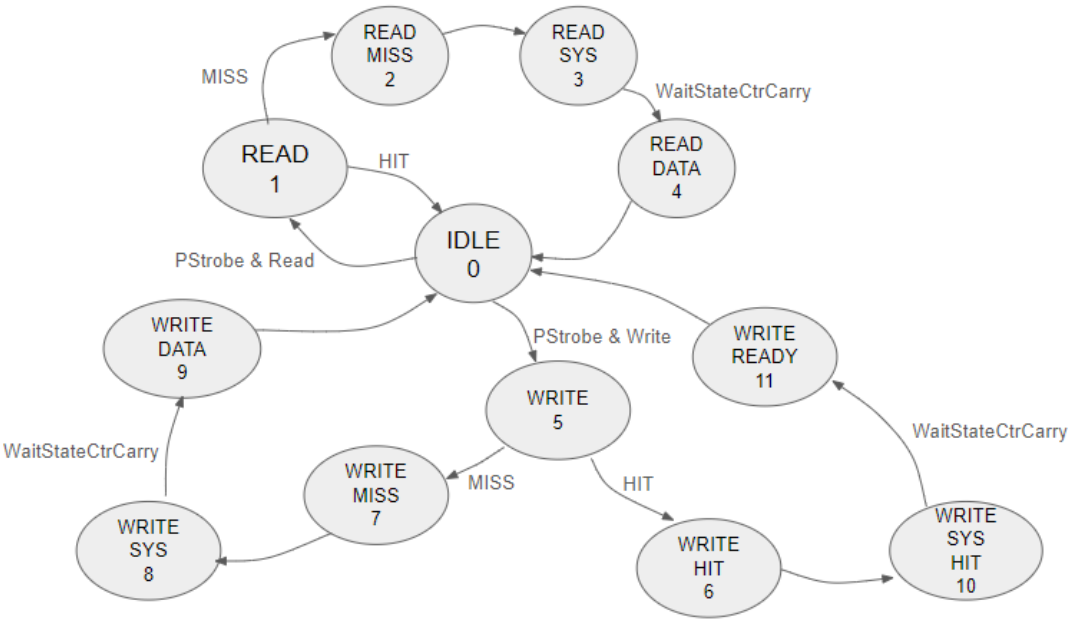
state diagram:

老師的：



圖十八、講義中 cache 的 state diagram(加上 cache)

改良後：



圖十九、改良過後的 cache 的 state diagram(加上 cache)

一般狀態下，Cache 會在 IDLE 這個狀態，然後 Pstrobe 決定了 Cache 要不要啟動，而 PRW 則決定是否要讀還是寫(PRW 為 1 是讀 0 是寫)，假設要讀的話，會進到 READ 這個狀態，這裡會判斷是否 HIT，HIT 回到 IDLE，MISS 則走另一條迴路。

再來是寫了，這邊會看到我的有跟老師有明顯不同，是因為老師在 WRITE 這個狀態下如果有 HIT 會把 PReady 設成 1，但是這會有問題，考慮到 CPU 是靠 PReady 來判斷 Cache 是否完成此動作，但 HIT 話 PReady 太早為 1，而把資料寫入 Memory 是在 WRITESYS 這個狀態，假設有兩個寫的指令，那 Cache 會還沒回到 IDLE 就進到下一個指令，故我有修改 State Diagram(如下圖)，就是把 WRITE 拆成兩條迴路。

(註:State 下面數字是我程式所設的代表值)

```
1 0000_0001
2 0000_0002
3 0000_0003
4 0000_0004
5 0000_0005
6 0000_0006
7 0000_0007
8 0000_0008
9 0000_0009
10 0000_000A
11
```

圖二十、cache 的 testbench(未加上 cache)

九、 問題與討論

如何將進行中的程式停止?
我們加上了 ecall 來停止程式的執行

如何解決合成進行太久?
需要把 memory 從 CPU 中拔出，合成 CPU 時就不會執行太長的時間。

cache 如何判斷暫停時間?
當程式是 LOAD 或是 STORE 時，需要暫停來等到 cache 取得資料，再用 PReady 來繼續執行 CPU。

如何處理 stall、cache 的 stop、jump 和 branch 的優先級?
stall 同時又有 stop 時必須是最優先的，要用 PReaady 先讓 stop 結束讀取或是儲存到資料才能繼續接下來的步驟，再來是只有 stop 時，因為要先等到 LOAD 或是 STORE 讀取或是儲存到資料才能正確執行，再來是 stall 進行在程式中加入 bobble，最後才是 jump 和 branch 來進行跳轉的動作。

十、 心得

鄭喆嚴：
製作 Pipeline CPU 一開始在設計階段，需要深入瞭解指令 Pipeline CPU 的原理和硬體細節，並需要仔細考慮各個階段的設計和連接，在實際實作過程中遇到了不少挑戰，這讓我更加熟悉處理器架構的細節，也提高了我的硬體設計能力。到了進行 layout 的步驟，發生錯誤後需要不斷調整和優化，最後才 layout 成功。整個製作過程讓我體會到硬體設計的複雜性和精細性。同時在不斷發生問題的過程也培養了我解決問題的能力，但看到一個自己製作的 Pipeline CPU 正確運行，感到了莫大的成就感，這個期末 project 不僅讓我學到了很多硬體設計的知識，也讓我體會到了解決問題能力的重要性。
游承嘉：
這次 project 讓我對 verilog 與 rv32i 組合語言更加熟悉，在設計測資時需要了解各個 instruction 的功能，並給出測試各種極限功能的狀況，寫 sort 時因為是使用遞迴，需要謹慎確認 stack pointer 等暫存器值的分配，與 caller、callee save 的使用，確保記憶體的正常使用。最後加入 cache 時，遭遇許多 pipeline 的 bypassing 問題，因為沒有完整的架構可以參考，只能一步步嘗試可能造成錯誤的指令組合。過程中雖然辛苦，但是在一次次檢查並思考解決方案的過程中，越來越清楚整個架構，在過程中學到了很多。
王樟翔：
我覺得這次報告讓我對 verilog 更加熟練，老實說，在修這堂課之前我是沒有碰過 verilog 的，所以有時寫出來的東西可能會讓會寫的人在想我在寫什麼，不過經過這次的 project，我可以明顯察覺到我 verilog 有在進步，另外我計組是自學的，而我原本學的是 mips，而這次 project 讓我學到了 risc v 的架構是什麼，也順便加深我對 cpu 的概念。不過我覺得最讓我頭疼的是 debug，當結構越複雜的時候，你越難找出錯誤的位置，而且有時候只是大寫打成小寫，vcs 也不會告訴你有打錯，然後某一條線就直接變 floating，這我真的會找到哭出來，另外還有一些是邏輯上的錯誤，有些邏輯甚至課本都沒提到，也只有當你做下去才知道。

呂順瑋：
經由這次 pipeline CPU 的設計，不只對 RTL 的設計更加了解，熟悉 verilog 的語法，以及對於 CPU 和 cache 的構造和運作原理更加清楚之外，也深刻體會到良好團隊合作的進行方式，包括團隊的共識、問題討論、適當的分工和所有人的參與。雖然這次專題我主要負責的部分較少與程式執行有關，但是在團隊進行的每個階段中，都需要一起了解並且把結果記錄下來，在整理的過程也同時讓我學習到整個設計的細節。

十一、 分工

表二、分工及貢獻度

姓名	學號	負責項目	貢獻度
鄭喆嚴	E14096724	Pipeline CPU RTL、ICC	25%
游承嘉	E24116241	testbench(單一、fibonacci、sort)、Layout	25%
王樟翔	E64094065	Layout、Synthesis、Cache	25%
呂順瑋	E24094740	書面報告、簡報、Superlint	25%

十二、 參考資料

Risc-V Instruction set Manual, https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf