NYCU-EE DCS-2025

HW05

Simple CPU

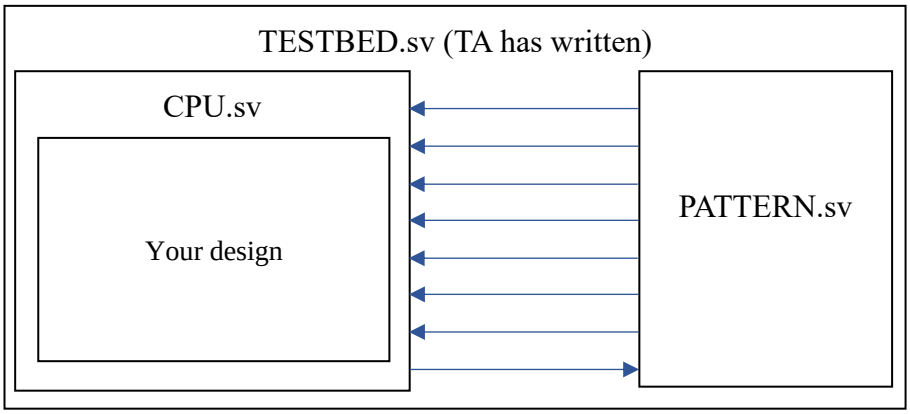
資料準備

1. 從 TA 目錄資料夾解壓縮:

   % tar -xvf ~dcsTA01/HW05.tar

2. 解壓縮資料夾 hw05 包含以下:
   a. 00_TESTBED/
   b. 01_RTL/
   c. 02_SYN/
   d. 03_GATE/

Block Diagram



設計描述

此次作業的Clock period可自行調整。

這次作業需要設計一個簡單的CPU，並根據輸入的指令(32-bit)，判斷需要做哪些運算，更新相對應暫存器的值，並輸出結果。

類型	位																													
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	opcode (6)						rs (5)					rt (5)					rd (5)					shamt (5)					funct (6)					
I	opcode (6)						rs (5)					rt (5)					immediate (16)														

- **R-type 運算指令**

Opcode(6)	Funct(6)	operation	desciption
000000	100000	Add	rd = rs + rt
000000	011000	Mult	rd = (rs * rt) >> 15
000000	000000	Left shift	rd = rt << shamt bits
000000	000010	Right shift	rd = rt >> shamt bits
000000	110001	ReLU	$rd = \begin{cases} 0, & if \ \ rt<0 \\ rt, & if \ \ rt\geq 0 \end{cases}$
000000	110010	Leaky ReLU	$rd = \begin{cases} (rs*rt)>>15, & if \ \ rt<0 \\ rt, & if \ \ rt\geq 0 \end{cases}$

- **I-type 運算指令**

Opcode(6)	operation	desciption
001000	Addi	rt = rs + immediate

- CPU 的暫存器以及 immediate 的 data type 是 signed fixed-point，除了作為 sign bit 的第一個 bit 外，其餘 15 bits 為 fraction bits，因此所有乘法的結果除了 sign bit 之外，只會取最大的 15 bits (相當於算完乘法後 >>15，上方表格皆有標示)，除此之外和 signed int 運算無差別。
- 若opcode不在列表中，則為**instruction fail**，instruction_fail output 設為 1
- Shamt, immediate即為value，可直接做計算。
- R、I type有寫回功能，分別用Rd/Rt表示寫回地址，須將值存起來進行下次計算。
- Instruction的rs(5), rt(5), rd(5)為address, 上述運算須取register的value出來運算，共有6個register。

Address(5 bits)	Value(初始為0)
10001	0
10010	0
01000	0
10111	0
11111	0
10000	0

- 範例

雖然暫存器 data type 為 fixed-point 小數，為了直觀方便起見，這些數值在以下範例用十進位整數形式表達。

以下為一些指令例子：

1. Instruction : 001000-10001-10010-0000000000000100

   Opcode-rs-rt-imm

   Addi instruction, reg(10001) = 0 ， imm = 4, reg(10010) = 0 + 4 = 4

2. Instruction : 001100-10001-10010-0000000000000100

   NO opcode 001100 -> instruction fail !

3. Instruction : 000000-10001-10010-10111-00001-000000

   Opcode-rs-rt-rd-shamt-funct

   Sll instruction, reg(10010) = 4, shamt = 1, reg(10111) = 8

4. Instruction : 000000-10001-10010-10111-00001-110010

   Opcode-rs-rt-rd-shamt-funct

Leaky ReLU instruction, reg(10001) = 4096 reg(10010) = -2025,

reg(10111) = (4096 * -2025) << 15 = -254

Input.txt

第一行為測資總數量，第二行開始為instruction。

```
5000
00000010111101111111100101000000
00100010000100000011001111000000
00100010001100100111110101111100
11000101111011101101111100010000
01111100100001101010111001111111
00000011111101110100000000100000
00000010001100001111100001000000
00100010001101110000100111000011
```

00_TESTBED 中附上與之對應的 input_readable.txt 供參考

Output.txt

由左到右分別為

instruction fail, out_0, out_1, out_2, out_3, out_4, out_5

(若為instruction fail, 則register file輸出當前的值)

```
0       0       0       0       0       0       0
0       0       0       0       0       0   13248
0       0   32124       0       0       0   13248
1       0   32124       0       0       0   13248
1       0   32124       0       0       0   13248
0       0   32124       0       0       0   13248
0       0   32124       0       0   26496   13248
0       0   32124       0    2499   26496   13248
1       0   32124       0    2499   26496   13248
1       0   32124       0    2499   26496   13248
1       0   32124       0    2499   26496   13248
0    -944   32124       0    2499   26496   13248
1    -944   32124       0    2499   26496   13248
0    -944   32124       0    2499       0   13248
```

- 助教的 demo 測資會排除任意連續四個 instruction 出現 read after write data dependency 的情形:

  instruction 1 -> r1 = r2 + r3

  instruction 2 -> r2 = r2 << 3

  instruction 3 -> r3 = r3 + r4

  instruction 4 -> r4 = r1 + r4

Inputs

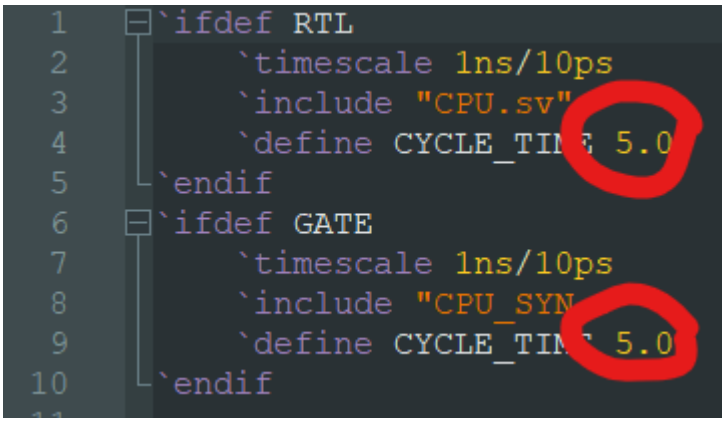
Signal name	Number of bit	Description
clk	1 bit	clock
rst_n	1 bit	Asynchronous active-low reset
in_valid	1 bit	為 1 時給值(連續 pull high)
instruction	32 bits	in_valid = 1 時給值, 一個 cycle 給一個 instruction

Outputs

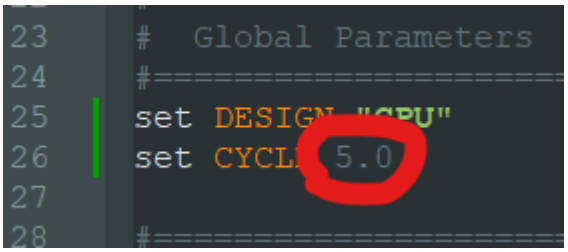
Signal name	Number of bit	Description
out_valid	1 bit	為 1 時輸出值(連續 pull high)
Instruction_fail	1 bit	opcode 不在上述 R/I-type 時為 1
out_0	16 bits	暫存器 10001 的值
out_1	16 bits	暫存器 10010 的值
out_2	16 bits	暫存器 01000 的值
out_3	16 bits	暫存器 10111 的值
out_4	16 bits	暫存器 11111 的值
out_5	16 bits	暫存器 10000 的值

Specifications

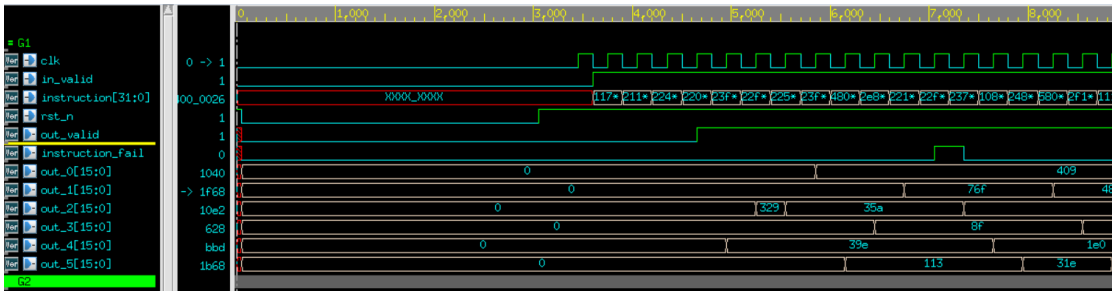
1. Top module name: **CPU**(File name : **CPU.sv**)
2. 在非同步負準位 reset 後，所有的 output 訊號必須全部歸零。
3. Output 要在 Input 後的 10 cycles 內輸出。
4. Output 要連續輸出，意即 out_valid 須連續 pull high。
5. 所有 Output 訊號要在輸出結束後全部歸零。
6. 02_SYN result 不行有 error 且不能有任何 latch。
7. Input delay = 0.5 * clock period; Output delay = 0.5 * clock period
8. Timing report 的 slack 必須為 non-negative 且 result 為 Met。
9. 03_Gate 不能有 timing violation。Latency 與 01_RTL 相同。
10. 請勿使用 *error*, *latch*, *congratulation* or *fail* 當作 logic / wire / reg / submodule / parameter 的名稱，否則 demo 結果會是 fail
    Note: *代表在該 word 前後的任何符號，比如: error_test 即為禁止的。
11. 更改 clock period:
    (1) HW05/00_TESTBED/PATTERN.sv 圈起部分

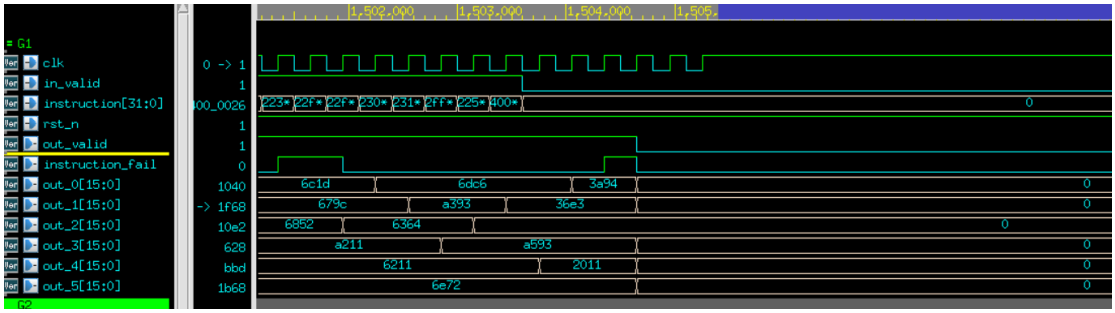


(2) HW05/02_SYN/syn.tcl



Example waveform





上傳檔案

1. 請將HW05/01_RTL裡的CPU.sv依以下命名規則重新命名後上傳至E3。
   命名規則：CPU_{clock cycle time}_dcsxxx.sv，xxx為工作站帳號號碼, clock cycle time請取到小數第一位。 例如：某同學的工作站帳號為dcs230，clock cycle time為5.0ns，他的檔名應為 CPU_5.0_dcs230.sv。命名錯誤扣5分
2. report_dcsxx.pdf, xx is your server account. 上傳至new E3。
3. 1de: 5/8 23:59:59
   2de & report: 5/15 23:59:59

Grading policy

1. Pass the RTL & Synthesis simulation. 60%
2. Performance = Area × Total cycles × Clock period^2 30%
3. Report 10%

Note

Template folders and reference commands:

1. 01_RTL/ (RTL simulation) → **./01_run**
2. 02_SYN/ (synthesis) → **./01_run_dc**
3. 03_GATE/ (gate-level simulation) → **./01_run**

報告請包括以下內容

1. 描述你的設計方法，例如如何加速(減少critical path)或降低面積或優化方法。
2. 基於以上，畫出你的架構圖(Block diagram)
3. 心得報告，不限於此次作業，對於作業或上課內容都可以寫下。
4. 遇到的困難與如何解決。(optional)

Hint:

本次 Performance 和 Clock period 平方正比，Pipeline 使用與否將大幅影響 Performance score。

本次助教 demo 不會有 hidden case。