Digital System Design
HW05 : AntVengers

105060016 謝承儒 電資 20

目的 :

設計出軟體(C 語言)和硬體(Verilog)來跑減法，求出兩數之間的最大公因數。

MMAP :

一. Input、Output、Others (FSM)

1. Input :
   A. clk : 頻率 100MHz 的週期。
   B. rst_n : 當它為 0 則系統初始化。
   C. [31:0] A : 輸入的數字。
   D. [31:0] B : 另一個輸入的數字。
   E. start : 當它為 1 時，代表可以開始運算。
2. Output :
   A. [31:0] Y : 最大公因數的值。
3. Others :
   A. [1:0] state, next_state : 代表現在和下個 state。
      a. 00 : START
      b. 01 : CALCULATE
      c. 10 : STOP
   B. [31:0] next_A、next_B : 代表 A、B 下一個 cycle 的值。

二. 設計過程

1. A、B 的值何時改變

   A、B 的值只有在 CALCULATE 這個 state 時要改變，這時 next_A、next_B 會存入 A、B。其餘 state 時，A、B 保持原樣。

2. 各個 state 間的轉換

   A. START :
      一開始的狀態，next_A、next_B 分別存入 A、B。若 start=0 代表還不能開始算，next_state 仍為 START;反

之，next_state 為 CALCULATE 開始運算。

B. CALCULATE ：

依照網路上提供的方法，將 A、B 做比大小，把大的減小的得到的新值，存入大的值裏頭(next_A 或是 next_B)。

若兩數相等，則 next_A、next_B 為 A、B 來保持原值，next_state 為 STOP。

C. STOP ：

因為已經計算完，next_A、next_B 一樣存入 A、B 不變，next_state 為 STOP。

三. 遇到的問題

1. 無法初始化

因為這次並無法使用 rst_n 來初始化值，不管是 state 或是 Y 在一開始都會是 unknown。

後來利用 defalut 的想法，在 FSM 的 always 區域開頭，就先把 next_state 設為 START，如此經過 1 個 cycle 後，state 就能被設成 START。

至於 Y，就一直把 A 丟進去，因為算到最後 A、B 都會是答案。

2. 在還沒讀完值時，就開始運算

本來是設計當 state 進入到 START，next_state 就設為 CALCULATE，但這樣會在讀完值前就開始做運算，後來就利用 start 來判斷是否該進入 CALCULATE。

PCPI ：

一. Input、Output、Others (FSM)

1. Input ：
   A. [31:0] pcpi_rs1, pcpi_rs2 ： 輸入的 2 個數字。
   B. pcpi_insn_valid ： 代表是否可以開始計算。
2. Output ：
   A. [31:0] pcpi_rd ： 最大公因數。
   B. pcpi_wait ： 若為 1，代表還沒算完。
   C. pcpi_wr ： 若為 1，就將現在的 pcpi_rd 傳出去。
   D. pcpi_ready ： 若為 1，代表已經算完。
3. Others ：
   A. [31:0] A, B, next_A, next_B
   B. [1:0] state, next_state

二. 設計過程

FSM 基本上和 MMAP 時一樣，只是多了 pcpi_wait、pcpi_ready、pcpi_wr 三個 Output 需要改變。

1. START ：
   在這 state 時，因為都還沒開始，所以三個都為 0。
2. CALCULATE ：
   在這 state 時，先將 wait 設為 1(因為正在計算)，其餘 2 個為 0。
   若 A==B，則就將 wait 設為並把 next_state 設為 STOP。
3. STOP ：
   在這 state 時，把 wr、ready 設為 1，因為已經算完並把值送出去，然後 next_state 為 START，回到初始狀態。

三. 遇到的問題

1. 當兩數互質時答案會是 2，相同時答案會是 3

   本來是在 A==B 時，只把 ready 設成 1 代表計算完畢。到 STOP 時，再只把 wr 設成 1 將值送出，但這麼寫就會出現這個 Bug。後來改成在 STOP 時，同時把 ready、wr 設為 1，就沒問題了。

2. 一直印出 DONE

   因為本來設計到 STOP 後，就會一直停留在 STOP，但這樣 ready

就會一直是 1，因此印出無數的 DONE。
後來將 next_state 改成 START 後就解決了。

四． 討論

1. What are the advantages of running applications on a hardware accelerator?
   在硬體的運算速度可以大幅上升，縮短跑出結果所需的時間。
2. Which version scales better if the input values to the GCD are getting larger?

```
MMAP :
GCD: A=10000 B=10001
Soft ans:1
Elapsed: 430323
Hard ans:1
Elapsed: 10189

Cycle counter ......... 572458
Instruction counter ... 122351
CPI: 4.67
Status:DONE

PCPI :
GCD: A=10000 B=10001
Soft ans:1
Elapsed: 520276
Hard ans:1
Elapsed: 10089
Cycle counter ......... 662248
Instruction counter ... 119948
CPI: 5.52
Status:DONE
```

```
MMAP :
GCD: A=100 B=101
Soft ans:1
Elapsed: 4623
Hard ans:1
Elapsed: 295

Cycle counter ......... 128788
Instruction counter .... 29065
CPI: 4.43
Status:DONE

PCPI :
GCD: A=100 B=101
Soft ans:1
Elapsed: 5476
Hard ans:1
Elapsed: 189
Cycle counter ......... 129404
Instruction counter .... 28978
CPI: 4.46
Status:DONE
```

從這 2 張圖，可以看出不管數字多少，硬體的速度都遠勝於軟體，而且當數字越大，縮短的時間倍率就越高。

3. Given a specific application, under what circumstances does its software version outperforms its hardware version?
   只有在當 2 數相同時，軟體的速度可以勉強跟硬體一樣快。
4. Roughly describe the hardware architecture of PicoRV32 (picorv32.v).
   現在看不懂，花幾天來研究。

五． 資料來源

1. 老師的 PPT