Digital System Design

HW02 : Register Transfer

105060016 謝承儒 電資 20

Part A : Fibonacci Series

一. 目的

設計出一個由 4 個 Register(R3、R2、R1、R0)儲存數字的系統，R0 輸出為斐波那契數列，R2、R3 分別為前兩項的數字。有兩種不同的設計 A1、A2。

二. Input、Output、Others(A1)

1. Input :
   A. clk : 頻率 100MHz 的週期。
   B. rst_n : 當它為 0 則系統初始化。(R3~0 裡的值皆為 0)
   C. [7:0] data : 從外部設定 Register 的值。
   D. [2:0] sel : 決定甚麼值會存入 Register。
   E. [3:0] load : 決定哪些 Register 可以存入值。
2. Output :
   A. [7:0] R3、R2、R1、R0：R2、R3 分別為前兩項(a_{n-1}、a_{n-2})的數字，R0 為最新的一項(a_n)。
3. Others :
   A. [7:0] temp : 經由 sel 控制的 MUX 後，得到準備送入 Register 的值。

三. 設計過程(A1)

首先，我們需要先決定在甚麼時候 Register 可以存值。接著，決定是甚麼值要把存入進去。

1. Register 何時可以存值
   這次系統使用 4-bit 的 load 來控制，load[0]控制 R0、load[1]控制 R1…依此類推。當 load[0]為 1 時，temp 就可以進入 R0；為 0，temp 就無法進入 R0。
2. 甚麼值會被送到 Register 前
   由 sel 來控制的 MUX 來選擇。
   當 sel 為 3'd0、1、2 時，temp 分別為 R0、R1、R2 的值。
   為 3'd3 時，temp 為 R2+R3(最新的項數)。

當 3’d4 時，temp 為 data(重新賦值)。

3. 得到最新項後，要同步放入 R3 或 R2，記得得交換放，如此 R3 和 R2 才會是前兩項。

四． Input、Output、Others(A2)

基本上和 A1 沒有差太多，將本來只由 1 個 MUX 來控制 temp，改成由 4 個 MUX 分別來控制 temp_r3~0。

只列出有更動的部分。

1. Input ：
   A. [7:0] data_r3~0 ： 從外部分別設定 R3~0 的值。
   B. [2:] sel_r3~0 ： 分別決定甚麼值可以存入 R3~0。
2. Others ：
   A. [7:0] temp_r3~0： 經由 sel_r3~0 後，得到分別送入 R3~0 的值。

五. 設計過程(A2)

基本上和 A1 上來改變就可以，只需要將 1 個 MUX 分成 4 個 MUX 就行了。

六. Bug 與解決方法

1. 4 個 Register 的存值順序
   本來我把 4 個判斷 Register 的 code 都寫在同個 always，用 if-else if 來連結，不過發現這麼寫，就會只存到優先順序較高的 Register。因此，就把 4 個 Register 都分別用 always 來寫，共有 4 個 always。

七. 討論

1. A1、A2 兩者中會選擇何種?
   我會選擇 A2。和 A1 相比，雖然 A2 操作起來比較複雜，必須設定很多東西，像是 data、sel 就比 A1 多 3 個。但因為 4 個 Register 是分開執行，所以可以在 1 個 clk 的周期內就設定好全部 Register 的值。相比之下，A1 必須 1 個個設定會慢下許多。
2. 有更好的改良方式嗎?

八. 資料來源

1. 老師的 PPT

Part B : Inner Product

一. 目的

從 Part A 的設計中，改造出一個可以計算 A = (a_0,a_1,a_2,a_3)、B = (b_0,b_1,b_2,b_3)兩向量內積的系統。此次我使用 A1 的設計。

二. Input、Output、Others

1. Input :
   A. clk : 頻率 100MHz 的週期。
   B. rst_n : 當它為 0 則系統初始化。(R3~0 裡的值皆為 0)
   C. [7:0] data : 從外部設定 Register 的值。
   D. [2:0] sel : 決定甚麼值會存入 Register。
   E. [3:0] load : 決定哪些 Register 可以存入值。
2. Output :
   A. [7:0] R3、R2、R1、R0 :
      R1 是 A 的向量、R2 是 B 的向量、R3 是內積的值。
3. Others :
   A. [7:0] temp : 經由 sel 控制的 MUX 後，得到準備送入 Register 的值。

三. 設計過程

首先，先讓 a_0 放入 R1，再讓 b_0 放入 R2，接著將 R1、R2 相乘再和 R3 的值相加後，便再次放入 R3。然後，重複前面的步驟將 $a_{1\sim3}$、$b_{1\sim3}$ 也這麼做，就能再 R3 得到內積的值。

基本上和 Part A 的設計沒有差太多，只是多了一個乘法器的部分，所以將 MUX 做下列的更動。

1. 甚麼值會被送到 Register 前
   由 sel 來控制的 MUX 來選擇。
   當 sel 為 3'd0、1、2 時，temp 分別為 R0、R1、R2 的值。
   為 3'd3 時，temp 為 R1*R2 + R3(內積)。
   當 3'd4 時，temp 為 data(重新賦值)。

四. Bug 與解決方法

1. 在跑完 syn 後，用 syn 的去跑波形圖，會有 200 多個 Warnings。
   沒找到解決方法。
2. 一開始 R3 的值為 X
   在第一次跑時，因為 R3 初始為 X，所以無法相加。後來先 rst 後，

把 R3 重設為 0 就可以跑出結果了。

五. 討論

1. 如果向量裡的數量超過 4 個(ex.A=[a_0,a_1,a_2,a_3,a_4])的話，做法一樣嗎?有何不同?
   我覺得做法是一樣的，只是多做幾次而已，但要注意 overflow 的問題。

六. 資料來源

1. 老師的 PPT