Digital System Design - Homework1

戴宏穎

A. Using A Single Multiplexer-Based Bus

1. File in hw1_A/hw1_A.v。
2. Testbench in hw1_A/hw1_A_testbench.v。
3. 這部分的設計因為可以參考講義上的圖作業說明也有詳細的規定，所以沒有花很多時間。不過在寫 testbench 的部分花比較多時間，主要是不太熟悉 testbench 的寫法，每次下指令的時候就會一直看到錯誤出現，感覺很是挫折。
4. Makefile 的部分，是參考講義裡的範例來修改，下了 -f 這個參數以後 ncverilog 都會顯示錯誤，於是直接用 `include 的方式引入。我的 Makefile 的使用方式請參考 Readme.txt。
5. 在 testcase 的部分，有些指令（例如：交換）在 Part A 的設計下沒辦法在一個 Cycle 裡面完成，所以寫 testbench 的時候就拆開成多個 Cycles 而且用當時沒有使用的 reg 來暫存，以完成交換的目的。
    1. R1 ← 0x1234
    2. R2 ← 0x8888
    3. R1 ← R2, R2 ← R1 => 多個 Cycles，多使用一個暫存器 R0
    4. R3 ← R1, R0 ← R1 => 多個 Cycles，因為沒辦法一次達成
6. 執行完 testbench 的輸出格式如下：
    1. ==========
    2. 狀態說明（可能會沒有顯示）
    3. Time 0 status （從開始到這個指令的時間）
    4. rst_n = 0 data = xxxx control = xxxxxxx （控制訊號與外部資料的值）
    5. R0 = xxxx R1 = xxxx R2 = xxxx R3 = xxxx （當前 reg 的值）
    6. ==========

7. 執行結果：
    1. ==========
    2. Time 160 status
    3. rst_n = 1 data = 8888 control = 0010001
    4. R0 = 8888 R1 = 8888 R2 = 1234 R3 = 8888
    5. ==========

B. Using Dedicated MUX-Based Transfer

1. File in hw1_B/hw1_B.v。
2. Testbench of Testcase B in hw1_B/hw1_B_testbench.v。
3. Testbench of Testcase A in hw1_B/hw1_B_testbench_testcaseA.v。
4. 這部分的設計特別是 control 訊號的部分，格式如下：
    1. 16'b MUX0_MUX1_MUX2_MUX3_LoadR3LoadR2LoadR1LoadR0
    2. MUX0, MUX1, MUX2, MUX3 => 3-bit
    3. LoadR3, LoadR2, LoadR1, LoadR0 => 各 1-bit
5. Q: Write a testbench to perform the same tasks as previous design. Any different consideration?
    1. R1 ← 0x1234
    2. R2 ← 0x8888
    3. R1 ← R2, R2 ← R1 => 一個 Cycles
    4. R3 ← R1, R0 ← R1 => 一個 Cycles
    5. 原本需要多個 Cycles 的動作因為有多個 MUX 的關係，所以可以在一個 Cycles 就把任務完成，例如做多重賦值或是 reg shift 的部分。
    6. 執行 testcase A 的結果：
        1. ==========
        2. Time 120 status
        3. rst_n = 1 data = 8888 control = 0001111110011001
        4. R0 = 8888 R1 = 8888 R2 = 1234 R3 = 8888
        5. ==========
        6. 可得知原本需要 160 的執行時間現在只要 120 而已，有不少需要拆開成多個 Cycles 執行或是需要而外暫存器的動作可以被簡化到單個 Cycle。

6. 在 testcase B 的部分：
    1. R0 ← 0x0101
    2. R0 ← 0x0202, R1 ← R0
    3. R0 ← 0x0303, R1 ← R0, R2 ← R1
    4. R0 ← 0x0404, R1 ← R0, R2 ← R1, R3 ← R2
    5. R0 ← R3, R1 ← R0, R2 ← R1, R3 ← R2
    6. R0 ← R3, R1 ← R0, R2 ← R1, R3 ← R2
    7. 執行結果：
        1. ==========
        2. Time 160 status
        3. rst_n = 1 data = 0404 control = 0100000010101111
        4. R0 = 0202 R1 = 0101 R2 = 0404 R3 = 0303
        5. ==========
        6. 透過可以直接互傳的功能，交換或是 shift 可以一個 Cycle 內就完成。
7. Q: Can you do that using the single MUX-based bus approach in (A)? If yes, how? If no, why?

    Ans: No, 因為就算能把一個 Cycle 內要完成的事情拆開成多個 Cycles 來執行，但是在交換的部分沒有而外的暫存器可以使用，所以無法達成。