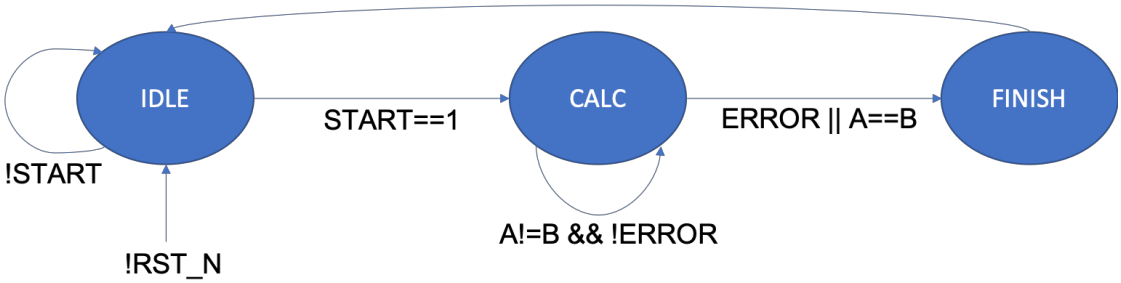
CS5120 Homework Assignment 01

Student ID: 107062612 Name: 熊祖玲

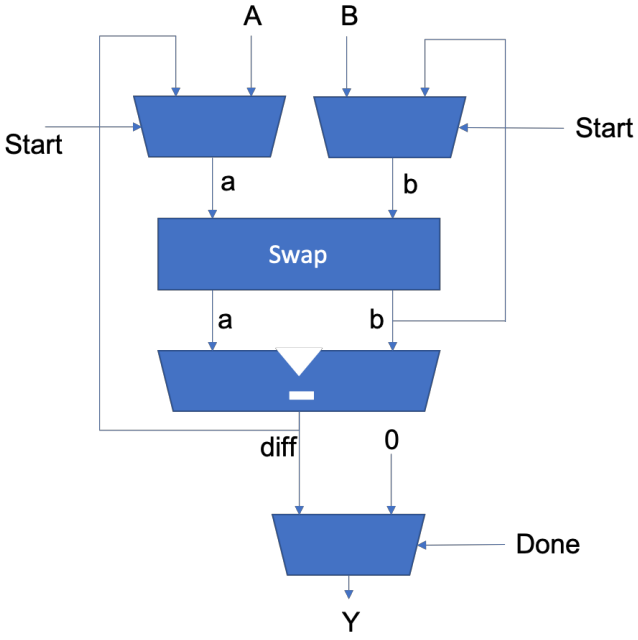
1. Design Concept

如下圖一的 finite state machine 所示，當 RST_N 訊號由 1 變 0 時，並且將所有訊號做初始化，並且進入 IDLE 狀態；當 START 訊號變成 1 時，將輸入訊號放到暫存器中，若輸入訊號 A、B 其中一個為 0 時，將 next_error 設為 1，並且進入下一個狀態 (CALC)；若 START 為 0 時，則停留在 IDLE 階段。在 CALC 階段時，若 ERROR (當 clock 由 0 變 1 時，會將 next_error 的值送到 ERROR) 為 1 或是輸入的兩個數字相等，則進入 FINISH 階段；反之，則重複地進行 GCD 運算。在 FINISH 階段會設定 output 訊號，並且將 next_done 設為 1，最後回到 IDLE 狀態等待下一次的計算。



圖一、Finite State Machine 圖

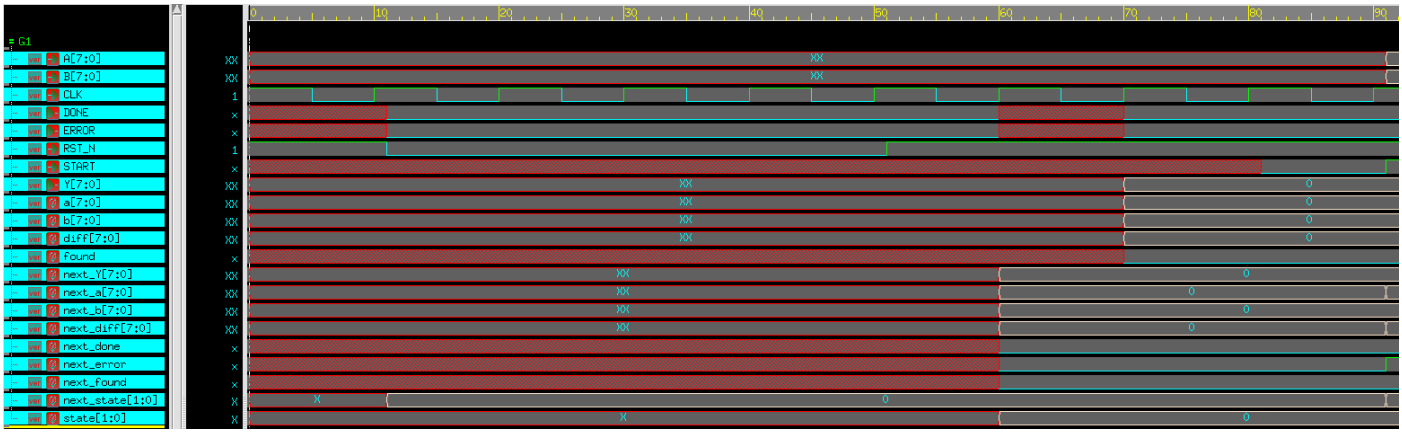
以下介紹 GCD 的計算方式（如圖二所示）：首先，當 START 訊號變成 1 時，將輸入訊號 A、B 放到暫存器 a、b 中，接著比較 a、b 的大小，較大者於下一個 clock cycle 設為 a，較小者則設為 b，之後計算出兩數的差值 diff (a-b)，並於下一個 clock cycle 將 a 設為 diff，再重複執行比大小、計算差值，當 a、b 相等時，表示計算已完成，則 gcd(A, B)=a=b。



圖二、block diagram

2. Simulation and Discussion

當 RST_N 訊號為 0 時，將訊號初始化。



圖三、初始化之波形圖

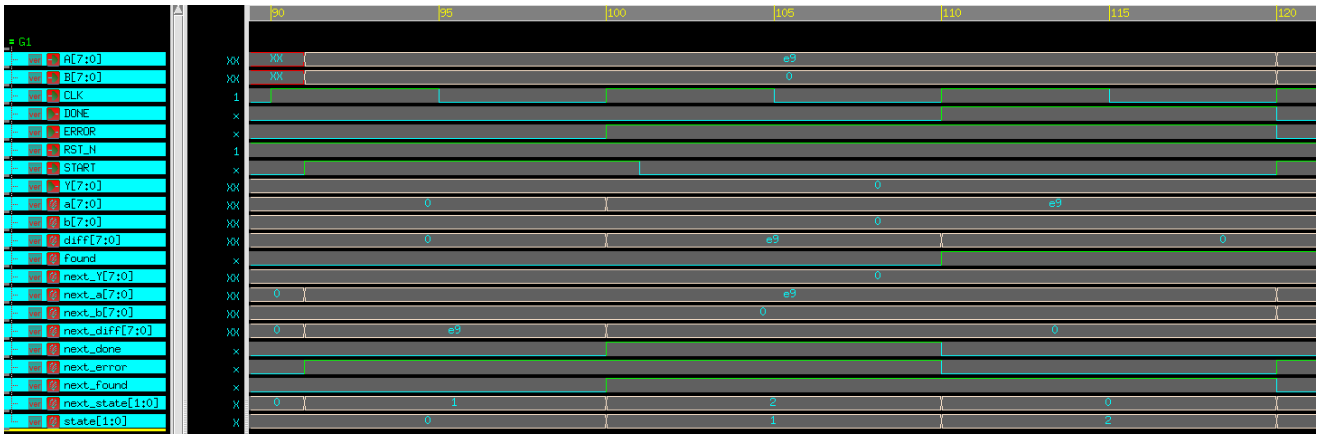
[Testcase1] A=233, B=0 (波形圖如圖四所示)

[Testcase2] A=0, B=233 (波形圖如圖五所示)

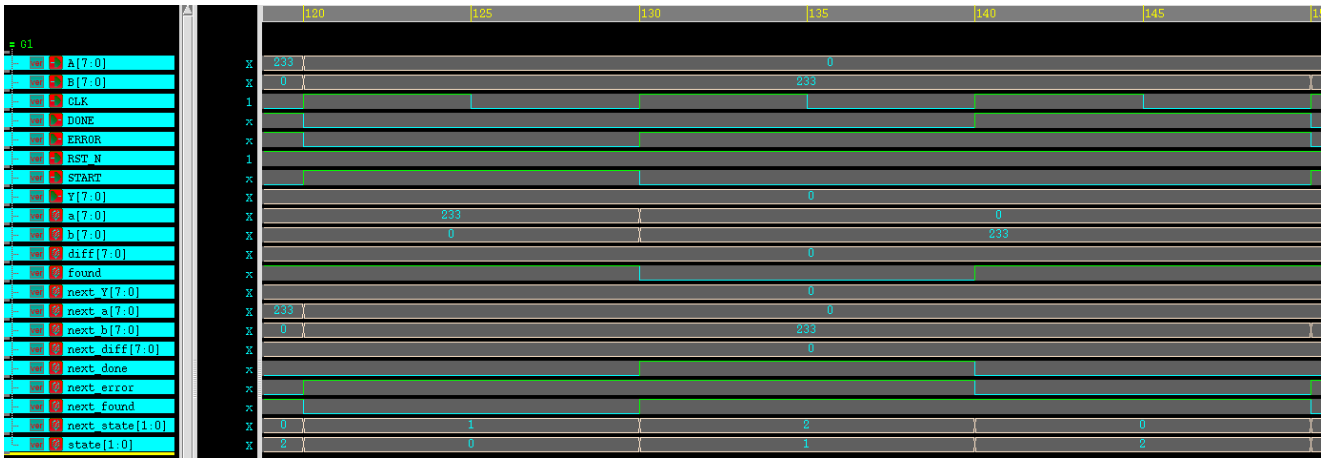
[Testcase3] A=0, B=0 (波形圖如圖六所示)

目的：驗證當其中一個訊號為 0 時是否正確，

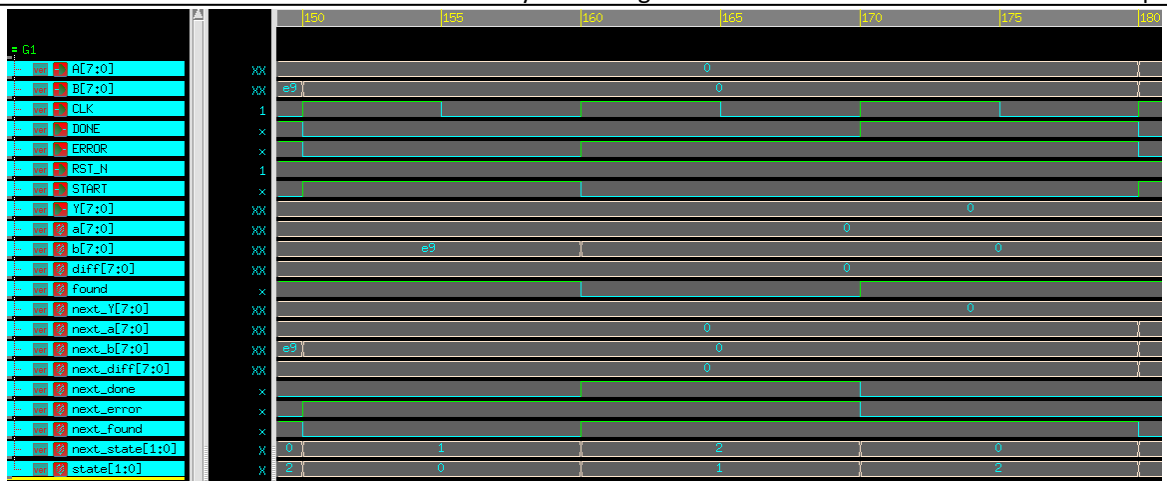
當 START 為 1 時，a、b 有正確儲存 A、B 訊號，並且進入 CALC 狀態 (state=1)，且因為 A、B 其中一個為 0，ERROR 也有正確的變為 1，偵測到 ERROR 為 1 後，亦正確進入 FINISH 狀態 (state=2)，最後輸出答案為 0。



圖四、testcase 1 之波形圖



圖五、testcase 2 之波形圖

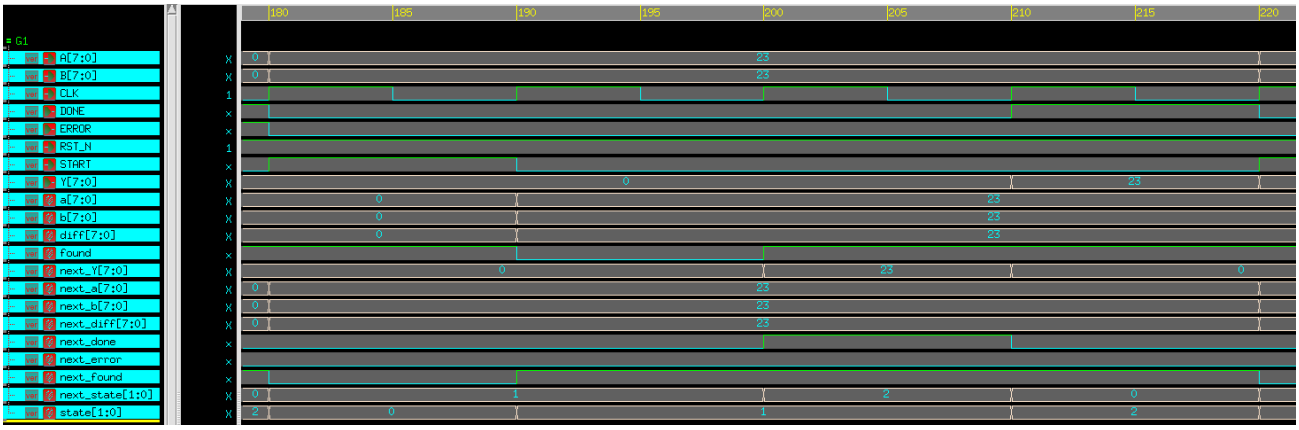


圖六、testcase 3 之波形圖

[Testcase4] A=23, B=23 (波形圖如圖七所示)

目的：驗證當兩個訊號相同時是否正確

當 START 為 1 時，a、b 有正確儲存 A、B 訊號，並進入 CALC 狀態（state=1），偵測到 a、b 相等後，正確進入 FINISH 狀態（state=2），且將 DONE 設為 1，最後輸出答案為 23。

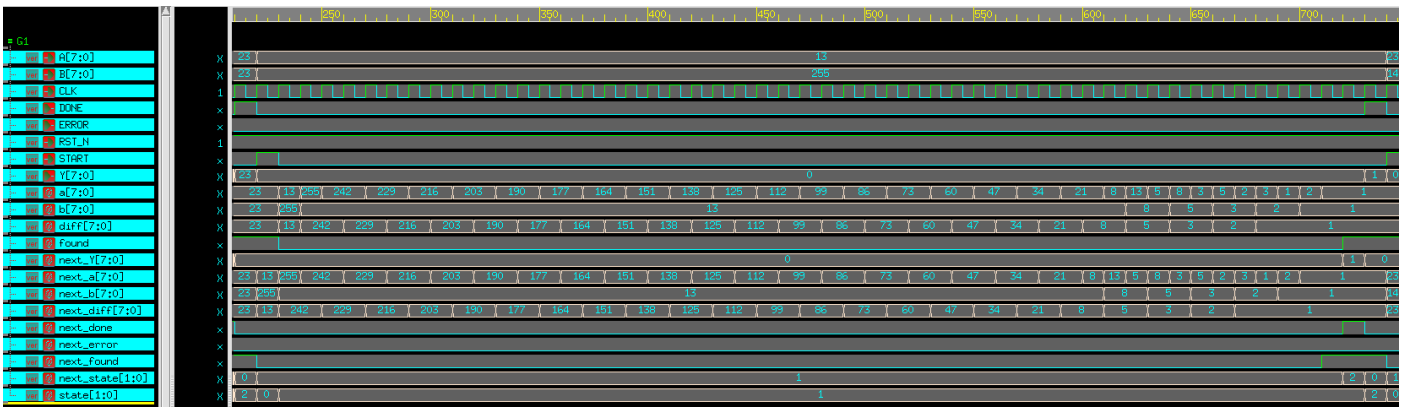


圖七、testcase 4 之波形圖

[Testcase5] A=233, B=13 (波形圖如圖八所示)

目的：驗證當兩數互質相同時是否正確

當 START 為 1 時，a、b 有正確儲存 A、B 訊號，並進入 CALC 狀態（state=1），偵測到 a<b 時，有正確地將兩數交換，於每一個 clock cycle 進行相減，得到正確的差值，並放到 a、b 暫存器中，當 a=b 時，正確進入 FINISH 狀態（state=2），且將 DONE 設為 1，最後輸出答案為 1。

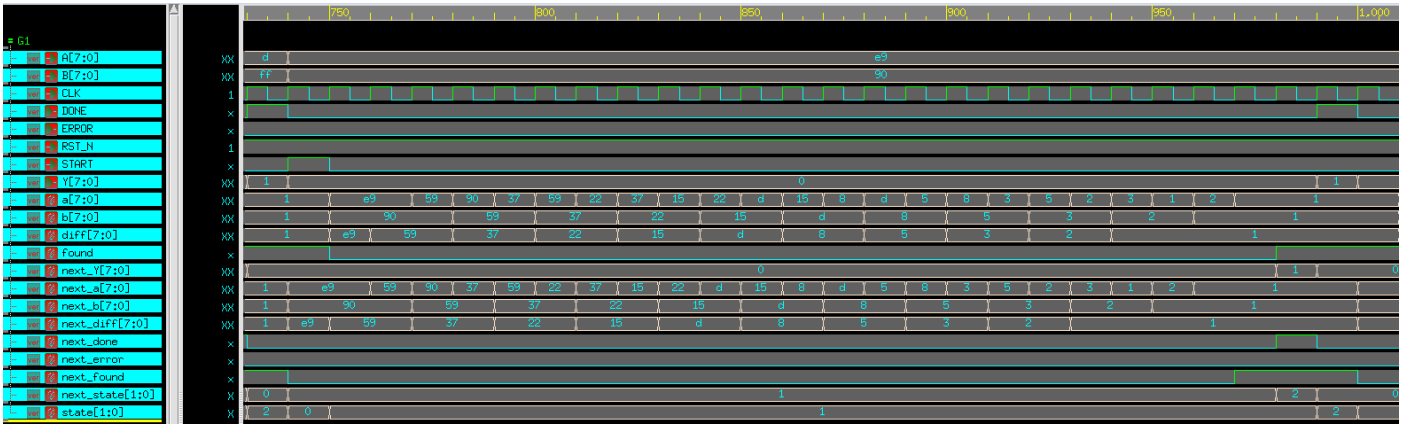


圖八、testcase 5 之波形圖

[Testcase5] A=233, B=144 (波形圖如圖九所示)

目的：驗證當輸入為兩個 fibonacci 數是否正確

當 START 為 1 時，a、b 有正確儲存 A、B 訊號，並進入 CALC 狀態 (state=1)，偵測到 b<a 時，正確地將兩數維持不動，於每一個 clock cycle 進行相減，得到正確的差值，並放到 a、b 暫存器中並且適時地交換兩個數值，當 a=b 時，正確進入 FINISH 狀態 (state=2)，且將 DONE 設為 1，最後輸出答案為 1。由於是最後一筆測資，模擬也正確的結束。



圖九、testcase 6 之波形圖

3. Summary

這次作業對我來說有點困難，因為之前從來沒有寫過 verilog，所以都是用 C 的想法在寫這次作業，導致一直都找不到問題在哪裡，後來去請教實驗室學長，才知道每個訊號都要有一個 next 的訊號，在 CLK 由 0 變 1 時再將訊號設為 next 的值，這樣才不會有同時設值的問題，而且要用 finite state machine 的架構去寫 verilog 的程式，根據每個狀態決定要做什麼運算。作業的難度不會很高，但是對於沒有碰過 verilog 的人來說需要花一些時間才能換個邏輯寫程式。