Lab4 Report

Team25

組長：蔡登瑞、組員：蔡政諺

Contents

● Designs

1. Mealy Sequence Detector

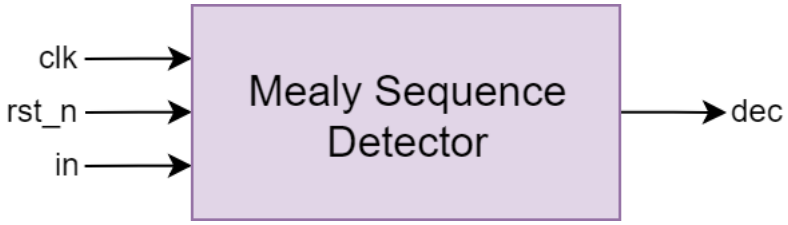
這題我們設計了 7 個 state。每四個 clock cycle 為一次循環，並在每個循環檢驗 in 是否符合我們想要的 sequence(1001)。

S0 為初始的 state，每次循環的第一個 state 都會是 S0。

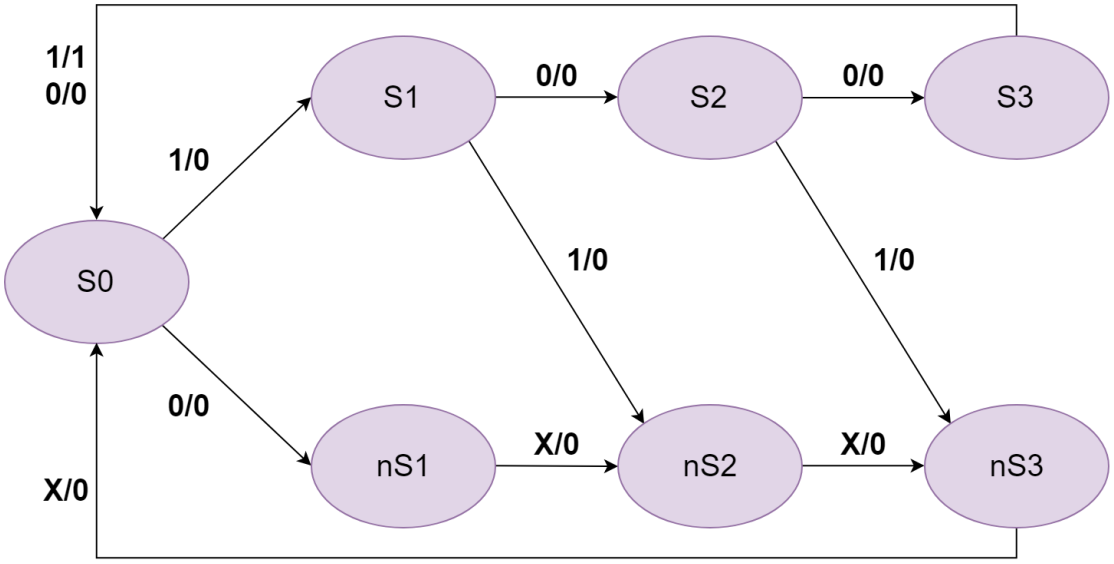
S1~S3 可以說是 detect 順利進行的 state，若每個 clock cycle 的 in 都符合，state transition 便是 S0→S1→S2→S3，而且在 S3 時，如果 in 是 1，dec 就會輸出 1；in 是 0，則 dec 輸出 0。

反過來說，**nS1~nS3** 則象徵這次的 detect 已經失敗了。如果說這次的 detect 在 S[i](0≦i≦2)時不符合需求，下一個 state 就會進入 nS[i+1]。nS 不管 in 為何都只會往下一個 nS 走，並在循環結束後回到 S0。而且在 nS 中，無論 in 為何，dec 都是 0。

Block diagram



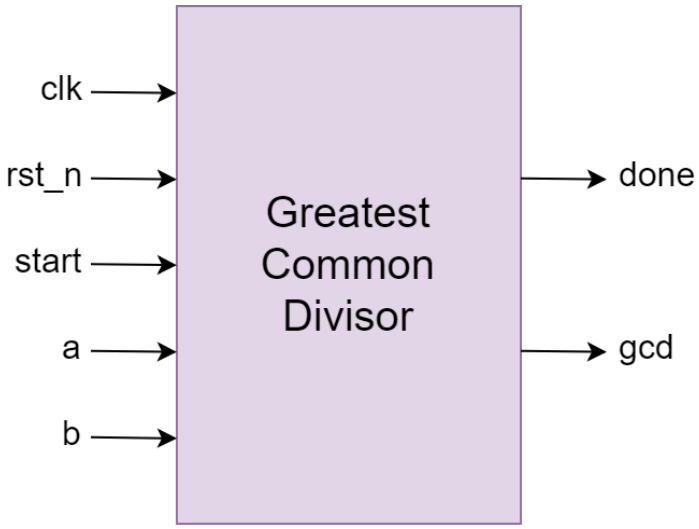
State Transition Diagram



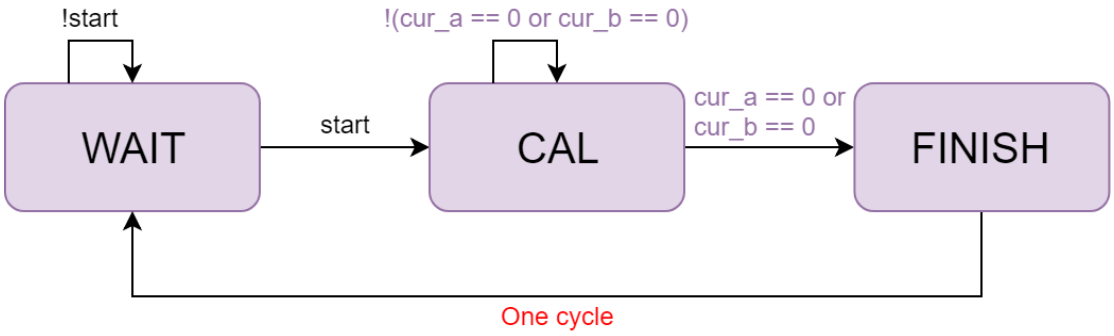
2. Greatest Common Divisor

這題的 State Transition Diagram，我們參照 PDF 上面給的，用了 3 個 state。一開始先在 WAIT 這個 state 等待 start 的啟動，啟動後進入到 CAL 的 state 進行輾轉消去法，當 a 或 b 其中一個為 0 時就進入到 FINISH 的 state，否則就一直在 CAL 這個 state 重複運行輾轉消去法。若一開始 a 或 b 或是兩個都已經為 0，就下個 state 直接進入到 FINISH，只有在 FINISH 的 state 時，gcd 跟 done 才會有值輸出，不然其餘時候兩個都為 0。WAIT 到 CAL 到 FINISH 這樣跑完一次後，回到 WAIT 等待下一個 start 再啟動。

Block diagram



State Transition diagram



3. Stopwatch_fpga

這一次實作 FPGA 的題目，我們的想法和上次 Lab3 的 FPGA 那題差不多。主要是這次多了 Finate State Machine 這個東西。在這次的實作裡，一樣用到了 debounce 跟 one pulse 去做 rst_n 跟 start 兩個訊號的處理。

再來，我們設計了一個 module—AN_replace，這是用來處理 FPGA 板上面四個數字的輪流顯示。

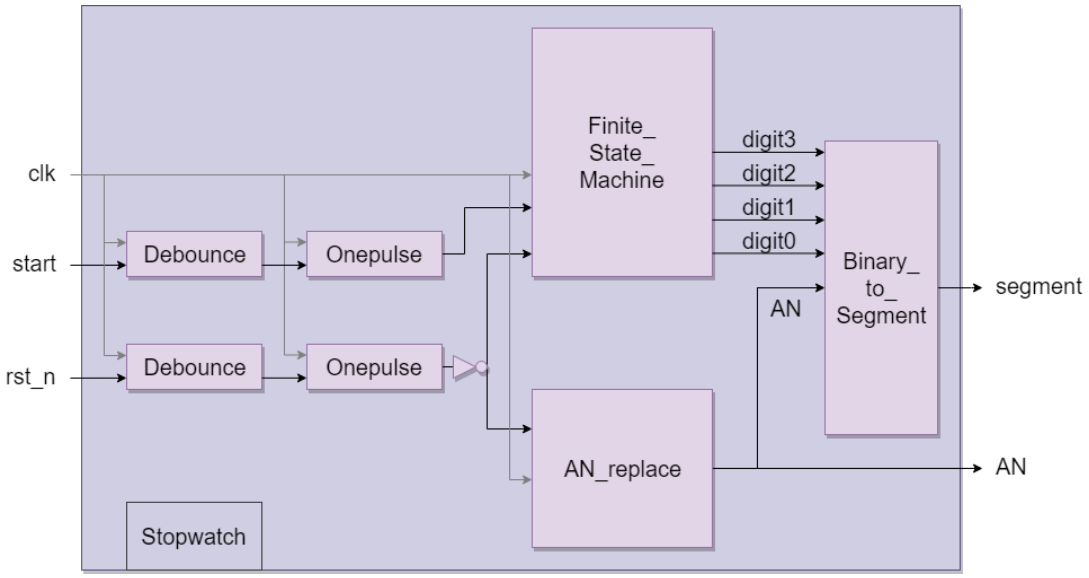
接著，如何以 Finate State Machine 製作出 Stopwatch。首先，我們參照 pdf 上所寫的分成 3 個 state，分別是 RESET、WAIT、COUNT，而 state 跟 state 之間的轉換就如下圖的 State Transition Diagram。

最後，我們設計了一個 module—Binary_to_Segment，看名字就知道是用來將四個 digit 的 binary 轉換成七段顯示器 segment 顯示在 FPGA 板上。

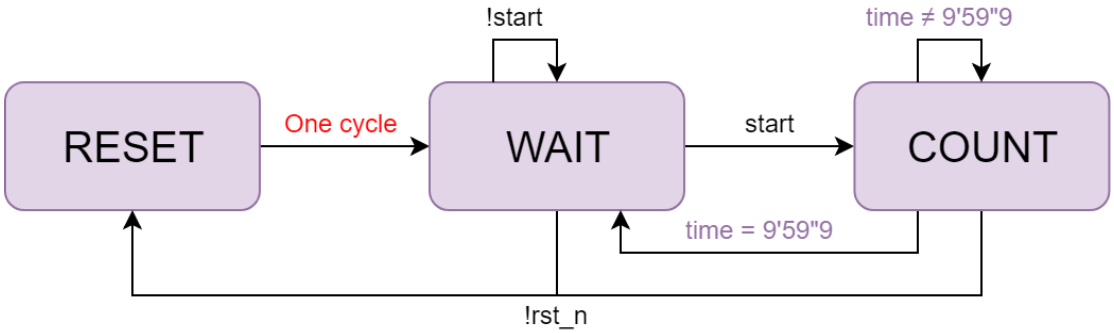
然而，我們覺得最重要的是 clock 訊號的處理，debounce、onepulse 跟 AN_replace 跟 Finate State Machine，分別需要用三個不同的 clock。我們這次的處理 clock 的方法和上次 lab3 使用的方法不一樣。這次我們沒有使用到 Clock Divider 去做除頻，而是在 module 裡面用 counter 去計算。而是使用類似於第八章裡面提到的 Low Frequency Sigmals，當 counter 等於某個固定的值時，next 的值才會給予 current，否則就 current 的值不會更動。這樣一來，所有用 posedge clk trigger 的的 always block 使用的 clk 就都會是同一個 clock。

處理好最重要的 clock 的訊號後，接著處理的是 reset 的訊號。原本我們 clock 的寫法跟上次 lab3 的寫法是一樣的，後來我們發現了一個問題，那就是除了 input 的 clk、rst_n、start 之外，其他寫在各個 module 裡面的 reg，大多都需要依靠 rst_n 來賦予起始值，也就是說，每一次的 reset 就是把全部的程式重新啟動一次，再加上我們為了避免使用到 Asynchronous 的 DFF，這次的寫法就變成將 reset 用最原始的 clk 去做 debounce 跟 one pulse 後，只要 reset==0 且 posedge clk 來的時候，不管目前的 state 或是 digit 是跑到什麼程度，全部都重新 reset 一次，整個程式重新啟動一次。

Block diagram



State Transition diagram

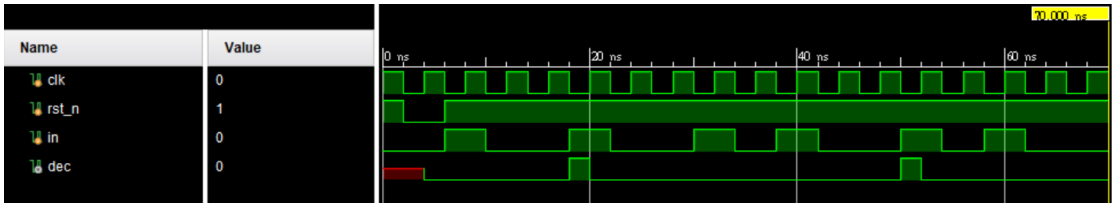


- **How we test our designs?**

1. Mealy Sequence Detector

這題我是參照作業 pdf 所提供的波形圖去寫 testbench。若 simulation 結果與 pdf 中的圖形相同，就表示 design 無誤。

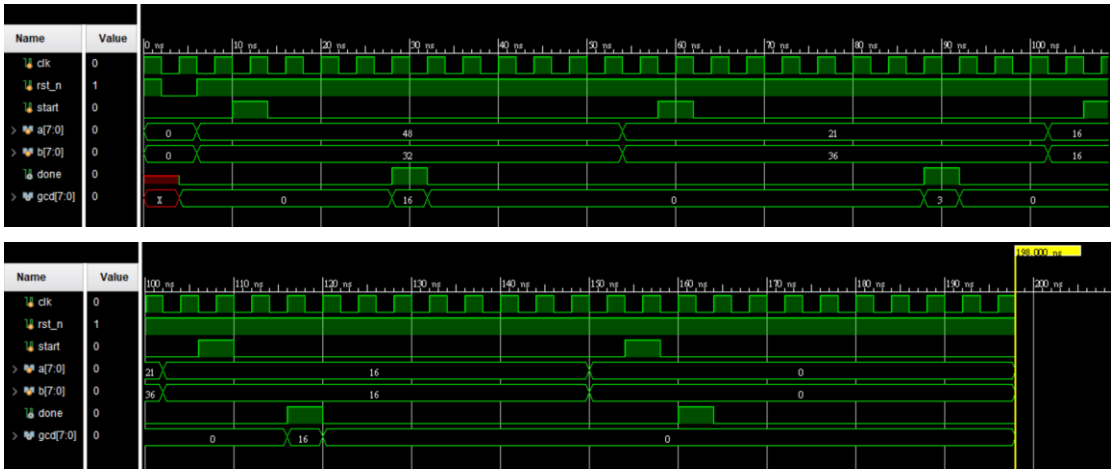
Waveform



2. Greatest Common Divisor

這題我挑了四組數字去做輾轉消去法，數字間的關係分別為 a>b、a<b、a=b。最後則測試 a、b 皆為 0 的測資。

Waveform



- **What we have learned from Lab4?**

這次的 Lab 我們首度練習 Finite State Machine 的寫法。有了 Finite State Machine 的概念，我們可以讓一份 code 有不只一種狀態，並能彼此轉移，這使得我們的程式可以包含多種不同的功能。

這次的 FPGA 實作與 Lab3 有些相仿，大觀念都是以 counter 計數，並將 clock 除頻與將 input 變成 one pulse。最大的出入就是這次是以 Finite State Machine 製作計數器，當數字即將 overflow 時，就會轉換成 WAIT 的 state。

Contribution List

- 蔡登瑞
  - 實作 FPGA
- 蔡政諺
  - 寫 testbench
  - 畫 state transition diagram
- 共同完成
  - 寫 code
  - 畫 block diagram
  - 寫 report