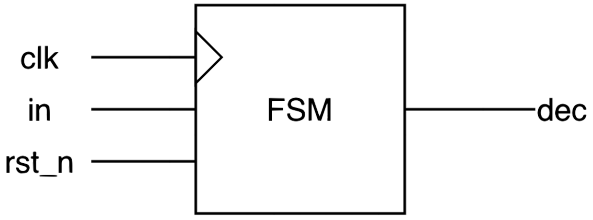
Lab5 :
Keyboard and Audio Modules

組別 : 25
組員 : 林書辰、楊景遇

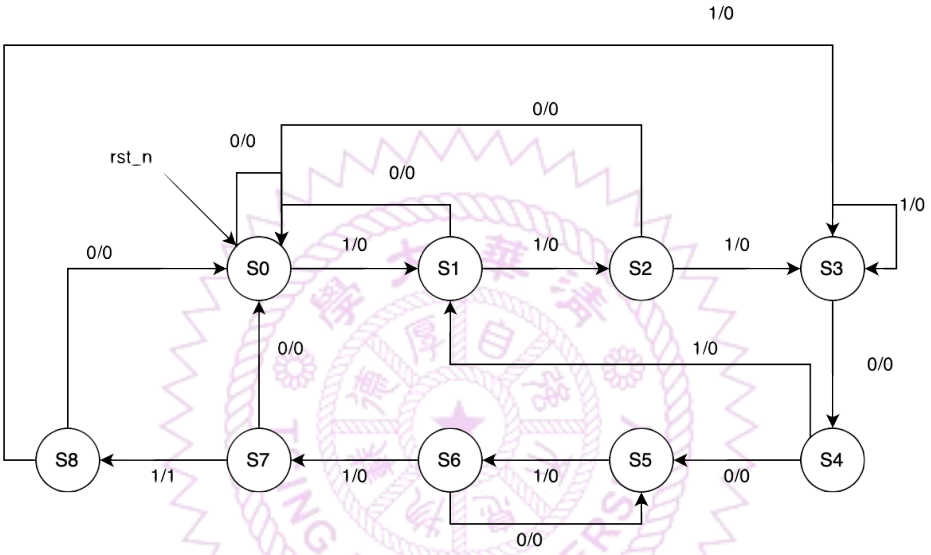
分工 :
林書辰 : Q1, Q2, FPGA2
楊景遇 : Q3, Q4, FPGA1

Q1 : Sliding Window

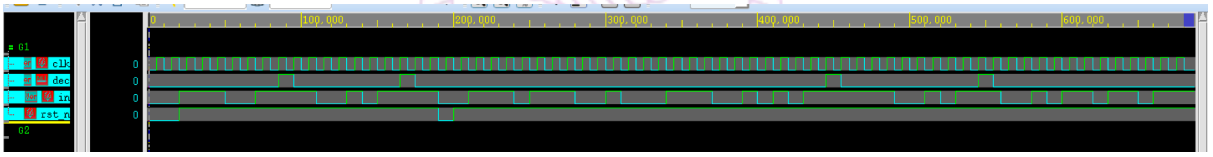
1. Block Diagram



2. State Diagram



3. Wave Diagram



3. Design

一開始我先設計了 rst_n 的功能。

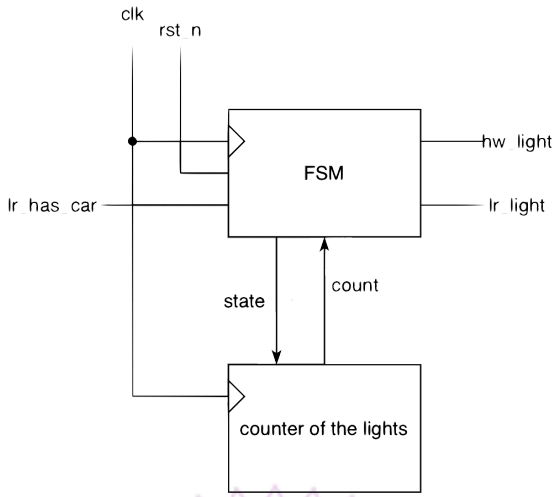
我設計了九個 state，第八個 state代表了如果出現正確答案會到達的 state，中間的其他 state 我多設計了如果沒有符合正確答案的 state 的話會到的 state，舉例像是 S4 的下一個應該要是 0，但如果出現 1 的話並不是回到 S0 而是 S1 因為我可以是那個一是下一個可能答案的第一個 1。

4. Test

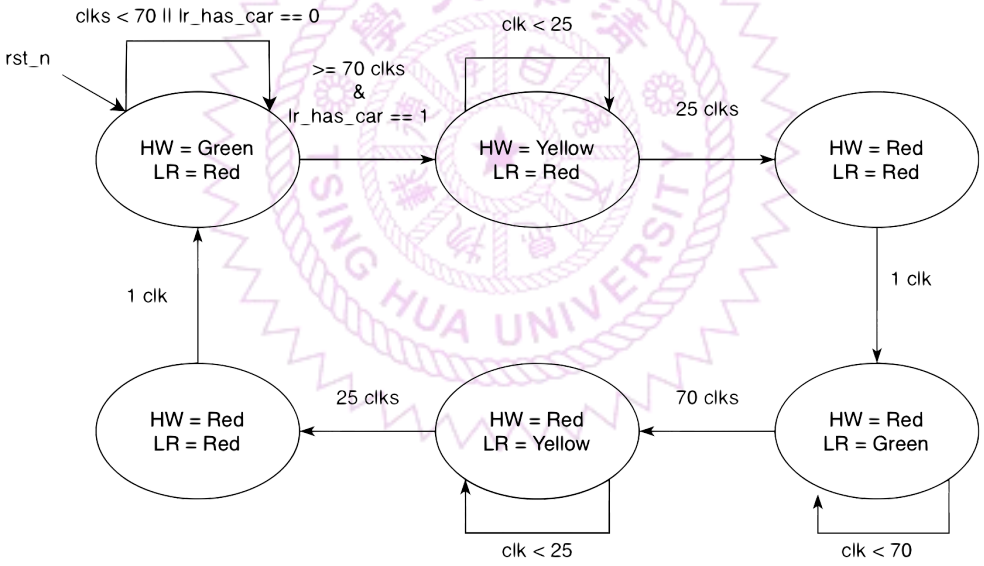
我測試了許多的答案，從一個 (01) 到三個 (01) 都有，還有其他錯誤的答案來看答案有沒有對，中間我也有在某一個有可能對的情況下 rst_n = 0，來看他有沒有偵測到 rst_n 。

Q2 : Traffic Light

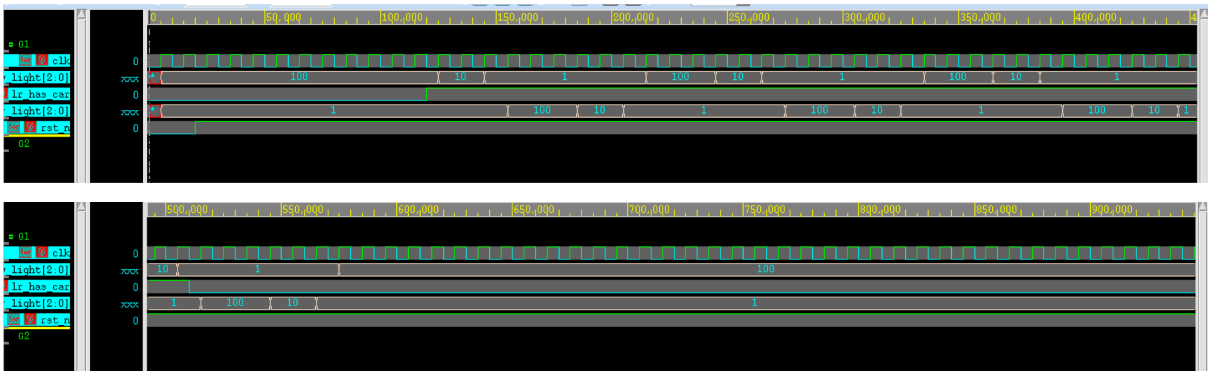
1. Block Diagram



2. State Diagram



3. Waveform



3. Design

首先我寫了 rst_n 的功能，也就是 rst_n == 0，state 回到 hw == green & lr = red

再來我分別設計了 hw, lr 的綠燈還有黃燈的 counter。

因為在換到下一個state之後，counter會在第二個 clk 才變成 1，所以我寫說如果黃燈的 counter == 24 還有綠燈的 counter >= 69 的時候就可以換到下一個 state，但如果是 highway 的綠燈，就要再判斷一個 lr_has_car 是不是一。

在數 highway 的 counter 的時候還有 overflow 的問題要注意，所以我在數 counter 的時候紀錄如果 > 70，counter 就維持在 70
（如圖，hg = highway green）

```
hg_next = (hg + 1 > 70) ? 70 : (hg + 1);
```

4. Test

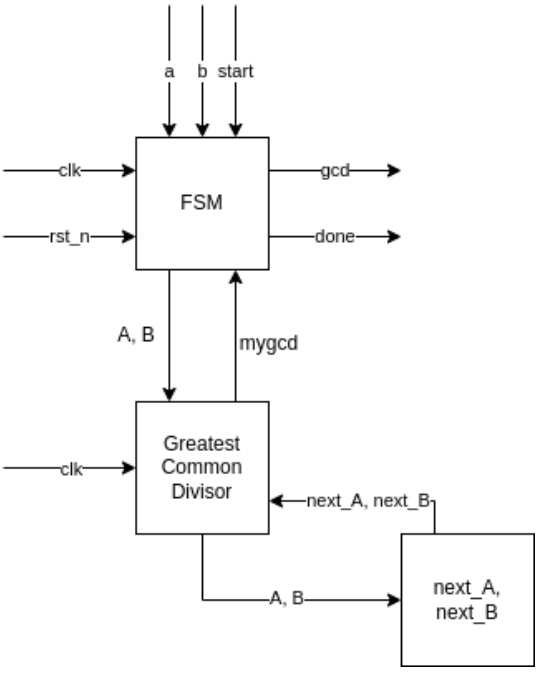
在測試 tb 的時候，我讓綠燈最少三個 clk，黃燈兩個 clk，為了方便測試。

第一張 Waveform 是在測試如果有車的情況他會跑到的所有情況，在一開始我讓 highway 的綠燈先跑了超過三個 clk，然後判斷 road 一旦有車就轉成黃燈，剩下的就是如果 lr_light 變成綠燈再變回 hw_light 綠燈的情況。

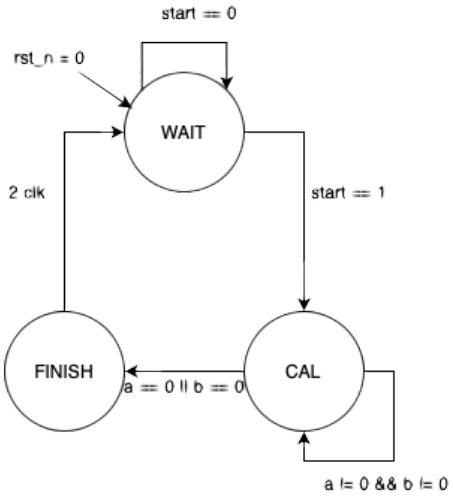
第二張 Waveform 是在測試如果沒有車的話 highway 會一直維持綠燈。

Q3 :

1. Block Diagram

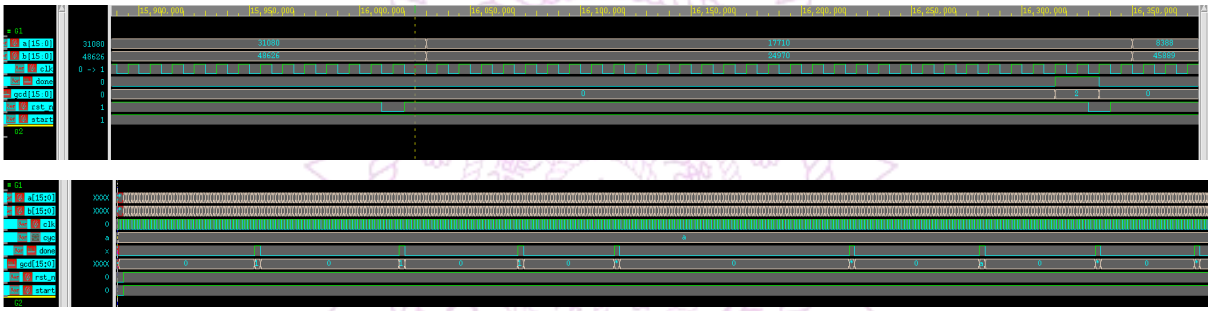


2. State Diagram



3. Waveform

（圖一）rst_n 功能測試



3. Design

我將每個state要做的事情用combinational circuit寫，而sequential circuit的部份只做reset跟更新每個reg或output，不做任何計算，維持良好的coding style。

combinational circuit的部份，我先判斷當前的state是什麼再執行相對應的運算，若state是WAIT且start = 1，讓下一個A, B分別變成input a, b以利接下來的計算，並將下一個state變成CAL。
在CAL階段，先判斷A是0或B是0，在這兩個狀況下下一個state是FINISH，不然的話下一個state還是CAL，並且下一個A或下一個B會是A-B或B-A，取決於A > B還是A <= B。
如果下一個state是FINISH的話，把答案存在mygcd這個reg裏面，在state = FINISH時再輸出出來。

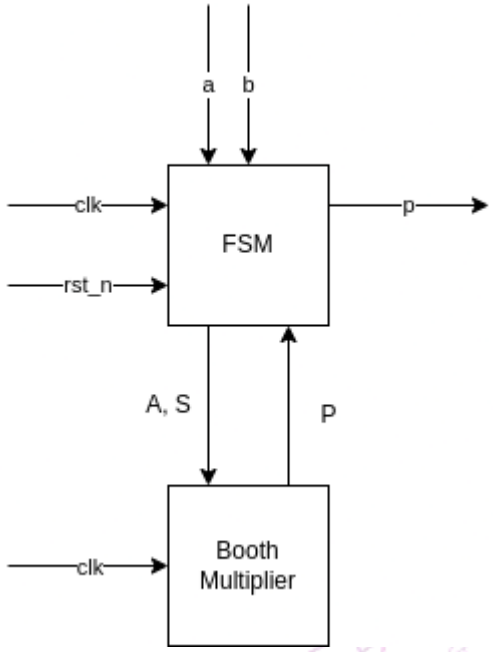
而在FINISH state，用一個counter紀錄維持這個state多久了，如果已經顯示兩個clock cycle就回到WAIT state。
只有在這個state讓gcd = mygcd，done = 1，在其他state時gcd =0, done = 0。

4. Test

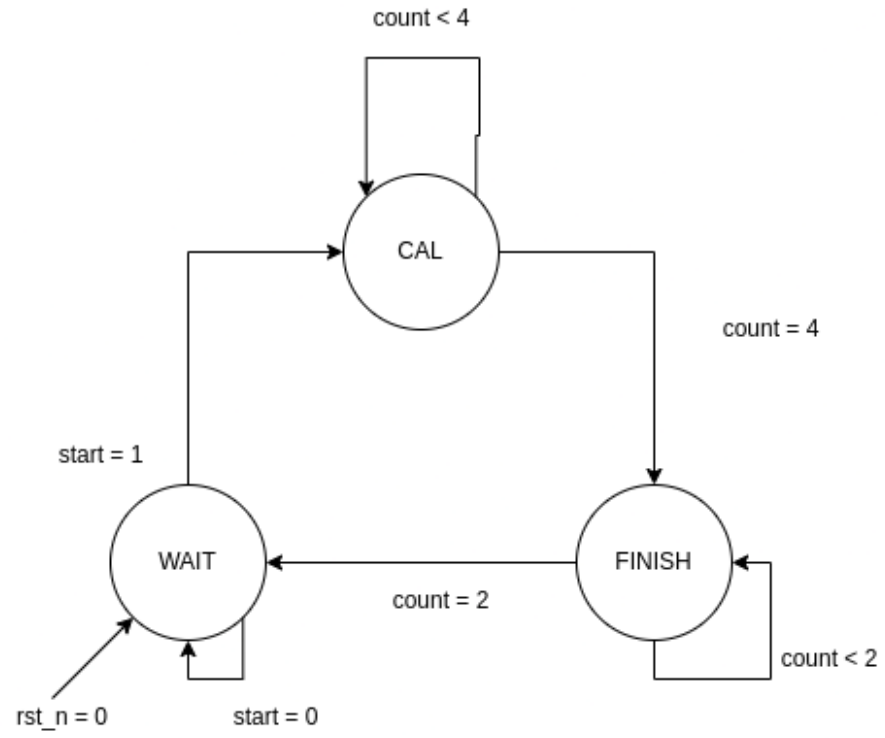
利用 $random 隨機生成 a 跟 b, 再去看他的結果是不是對的。
因為我們不知道gcd會在哪一個clock算好, 所以只能將test的clock數拉長到出現結果。

Q4 :

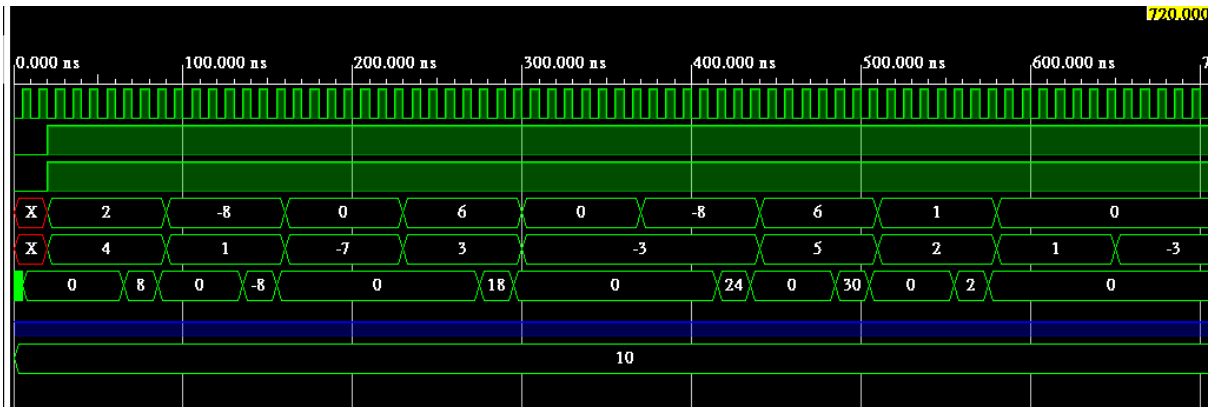
1. Block Diagram :



2. State Diagram :



3. Waveform :



4. Design :

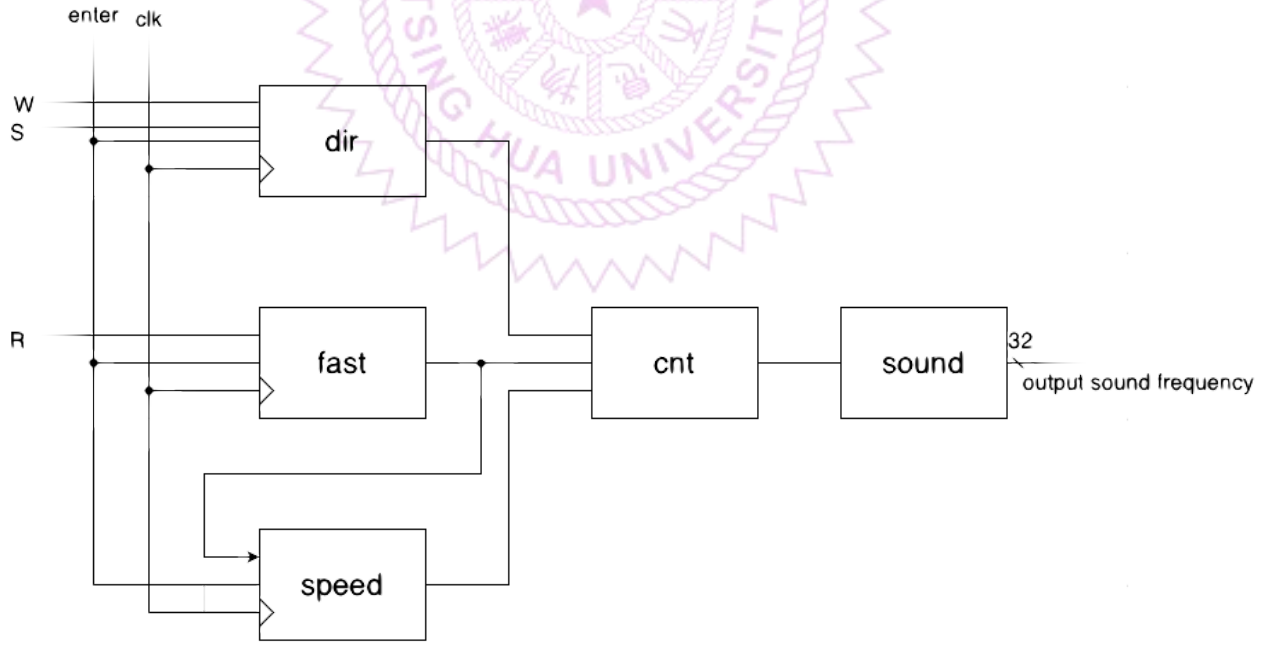
state的deisgn同Q3，只是在CAL的部份也使用一個Counter計算維持這個state多久了，而計算結果的部份，照著wiki上的寫法將A, S分別存入{a, 5'b0}, {-a, 5'b0}然後再執行四次運算，每次用P[1:0]來決定下一個clock的P會是P, P+A還是P+S的arithmetric shift。在FINISH state時一樣將P[8:1]輸出到p，就完成Booth Multiplier的design。

5. Test :

我們知道計算是每7個clock一個周期，那就每7個clock隨機給一組a, b方便檢驗。

FPGA 1:

1. Block Diagram :



2. Design :

在這一題主要的變數有
1. [31:0] sound [28:0], 用來記錄每一個頻率的聲音的
2. cnt, 用來記錄現在在哪一個聲音
2. fast, 用來記錄速度
3. dir 用來記錄現在聲音是往上還是往下
4. speed 用來記錄時間。
5. ONE_SECOND = (fast) ? 50000000 : 100000000

我的 rst 的寫法是判斷現在有沒有按下 enter, 他是個 wire, code 裡面長這樣。

```
assign rst = (been_ready && key_down[last_change] && last_change == ENTER);
```

一開始的初始值在方向向上，fast == 0，也就是每一秒換一個聲音。

如果我偵測到 W 或 S 被按下的話, 然後如果和現在的方向相反的話, 就會交換方向。
enter 被按下的話就會回到初始值。

我的 cnt 是跟著 dir 在走的, 如果現在往上, 他就會加一, 往下就減一。
輸出的地方則是 sound[cnt], 也就是現在 cnt 數到的位置。
speed 則是讀秒的 counter, 當 speed == ONE_SECOND的時候, 代表已經過了一秒, 也就是要換下一個音。這裡值得注意的是，如果 speed 數到大於 50000000 但是 fast 從 1 變成 0 的時候他會卡住, 所以要注意。

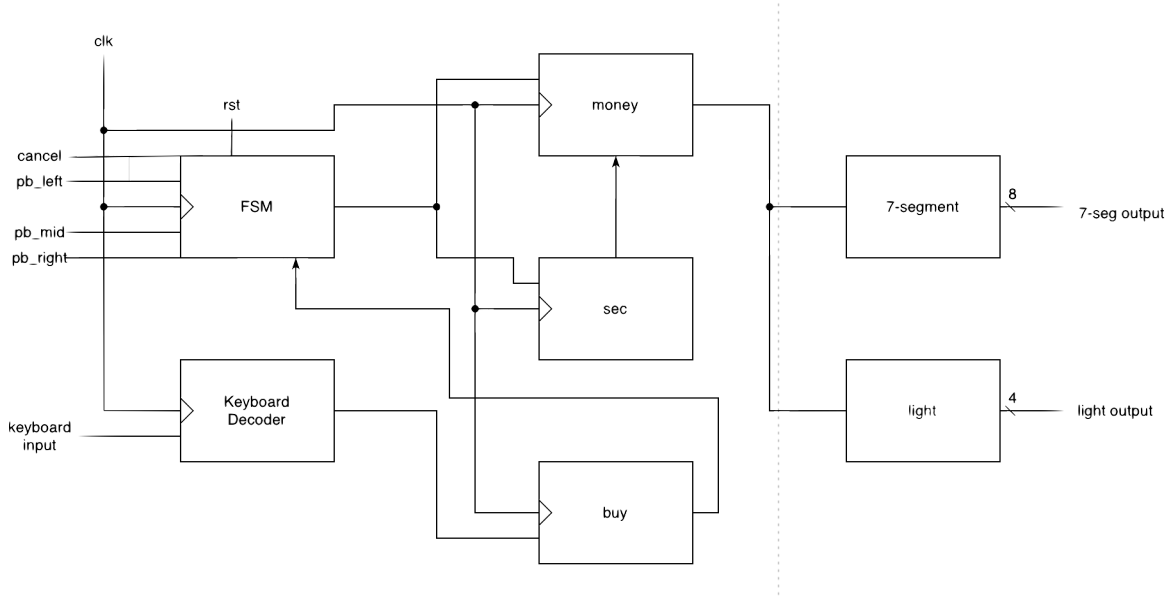
next_speed = (speed + 1 > ONE_SECOND) ? ONE_SECOND : speed + 1;

還有如果 speed 數到上限之後要歸零。

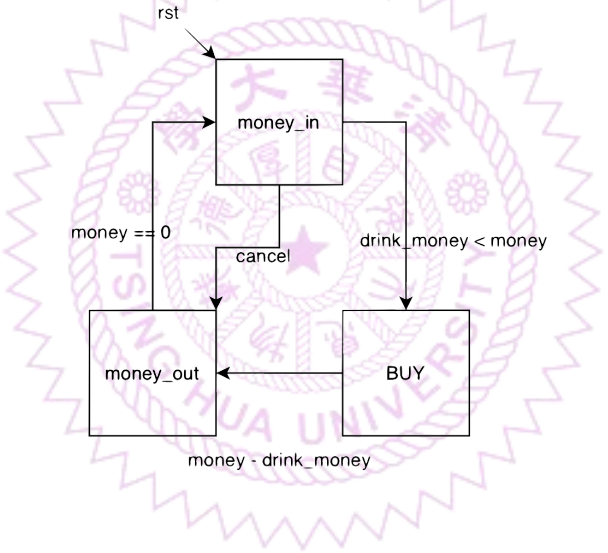
fast 的話就是如果 R 被按下, 就取not。

FPGA 2:

1. Block Diagram



2. State Diagram



3. Design

rst, pb_left, pb_mid, pb_right, cancel 這五個按鈕都有先做 debounce 還有 one pulse

我在 fpga2 題目中主要分成三個 state，分別是 money_in, BUY, money_out。

我主要的變數有 money, buy, state, sec。
money 是用來記錄現在有多少錢的，buy 是記錄上次按鍵按下去是按了哪一個飲料。
state 就是紀錄現在的 state，sec 則是在進到 money_out 的時候會開始的 counter，數到 1e8 會重頭開始數，是用來記錄什麼時候過了一秒的 counter，因為要退錢。

money_in :
當我按下 rst，state 會馬上進到 money_in。在 money_in 裡面，我會判斷有沒有錢被投進 Vendin Machine，pb_left, pb_mid, pb_right 分別是 5, 10, 50 塊錢，如果投錢進去之後 money > 100，我會讓 money 維持在 100。

如果鍵盤有按鍵被按下去(我用 been_ready && key_down[last_change] == 1 來判斷)我會判斷 buy，也就是我買的飲料的價格有沒有低於我現在投的錢，如果有的話就跳到 BUY 的 state，反之則無視，繼續留在 money_in。

這裡我也有寫 cancel 的功能，如果 cancel 被按下去，跳到moeny_out 的 state。

BUY :
在這個 state 裡面沒有什麼特別的，我用 buy 來判斷我買了什麼飲料，在一個 clk 的時間內讓 money = money - 飲料錢。並且在下一個 clk 跳到 money_out。

money_out :
這裡是要退錢的 state，我設計了一個叫做 sec 的 counter，表示一秒鐘的 counter，因為 clk 的頻率是每 10^(-8)秒一個，所以我定義 10^8 clk 為一秒，並在每次 counter 數到 10^8 的時候讓 money 減五。

當 money == 0，我會跳回 money_in。

ASDF 的 KEYCODE

```
parameter [8:0] KEY_CODES_A = 28;
parameter [8:0] KEY_CODES_S = 27;
parameter [8:0] KEY_CODES_D = 35;
parameter [8:0] KEY_CODES_F = 43;
```

What we have learnt :

在這次的 Lab 中，我們學習如何使用鍵盤還有喇叭，學會看懂老師提供的 code 也讓我瞭解了鍵盤跟喇叭是怎麼控制的，也在做完這次的 Lab 後對Finite state machine有了更深入的認識。透過Question 1 ~ 4照著已經設計好的FSM實作，讓我們在寫FPGA 2時對於如何設計FSM使得之後的實作loading減輕有了更為明確的想像。

而Booth Multiplier這個神奇的乘法器也讓我們再次認識到硬體設計的有趣之處，我們了解到一個良好設計的演算法能很大程度加速運算過程。

在使用課程提供的sample code的同時也增加了我們trace code跟運用的能力，使我們變得能更快地上手新的project，為了final project做好準備。

Suggestion :