Lab 5: Keyboard and Audio Modules

Group 3 : 109060013 張芯瑜　109062328 吳邦寧

Design Explanation

1. Sliding Window sequence detector

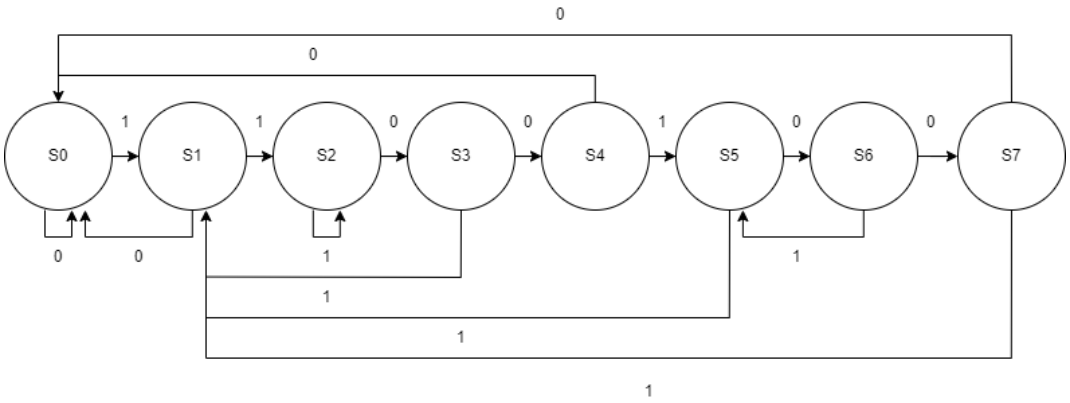
(1) 設計說明：

利用 finite state machine 實作，偵測輸入進來的數串是否符合題目需求。

S0	S1	S2	S3	S4	S5	S6	S7
1	1	0	0	1	0	0	1

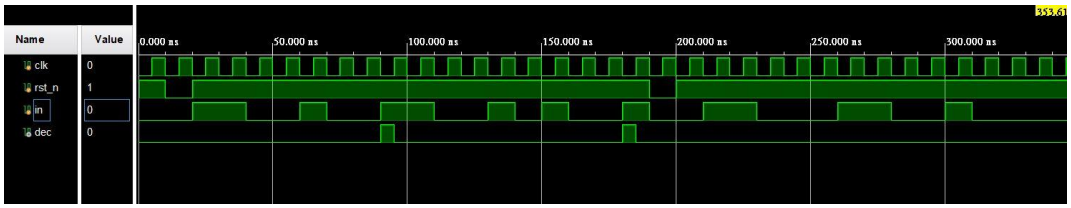
當 state 達到 S7 切 in=1，dec 被拉起，並在下一個 posedge clk 時歸零。

(2) State Diagram：



(3) 驗證：

嘗試所有可能的狀況，跑過所有的 state 並能夠得到正確的 dec 訊號。



2. Traffic light controller

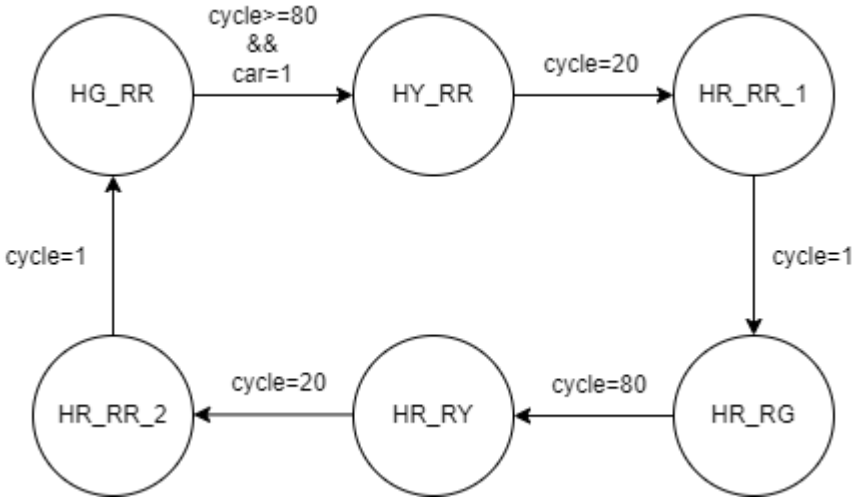
(1) 設計說明：

按照題目要求設計，並分成下面六個 state。

- HG_RR：Highway 綠燈/Local Road 紅燈。
- HY_RR：Highway 黃燈/Local Road 紅燈。
- HR_RR_1：Highway 紅燈/Local Road 紅燈第一次。
- HR_RG：Highway 紅燈/Local Road 綠燈。
- HR_RY：Highway 紅燈/Local Road 黃燈。
- HR_RR_2：Highway 紅燈/Local Road 紅燈第二次。

利用每一 clk 加一的方式計算 cycle 數， 當 cycle 數符合跳到下一 state 的條件時則將 next_state 設為下一個對應的 state。

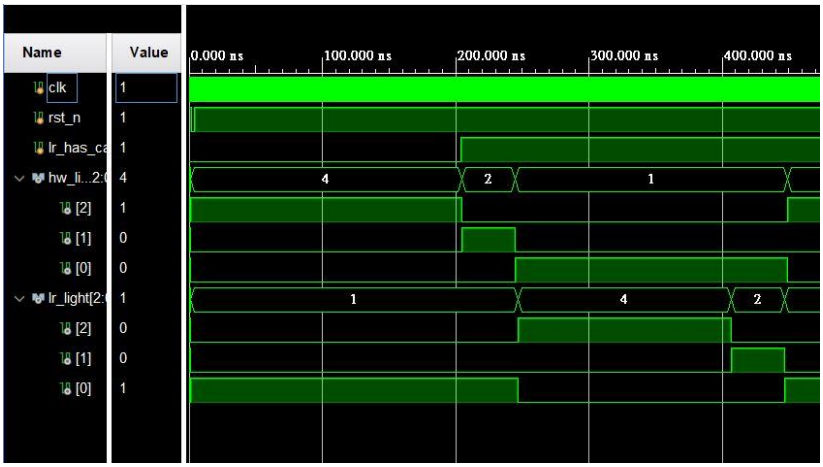
(2) State Diagram：



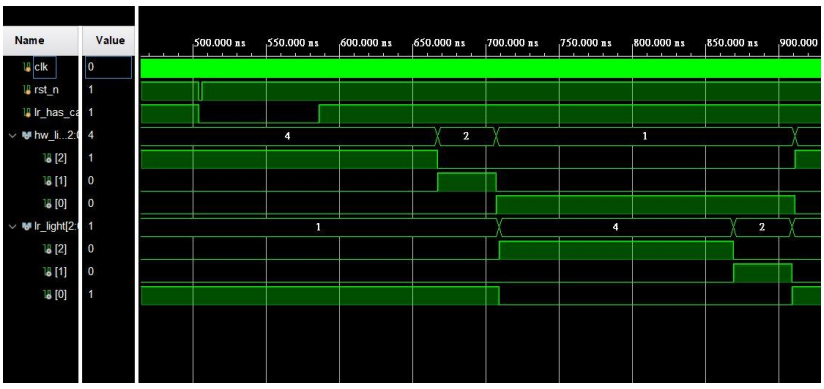
(3) 驗證：

測試兩種狀況

- Highway 綠燈先過 80cycle，Local Road 才有車。



- Highway 綠燈未過 80cycle，Load Road 就有車。



3. Greatest Common Divisor

(1) 設計說明：

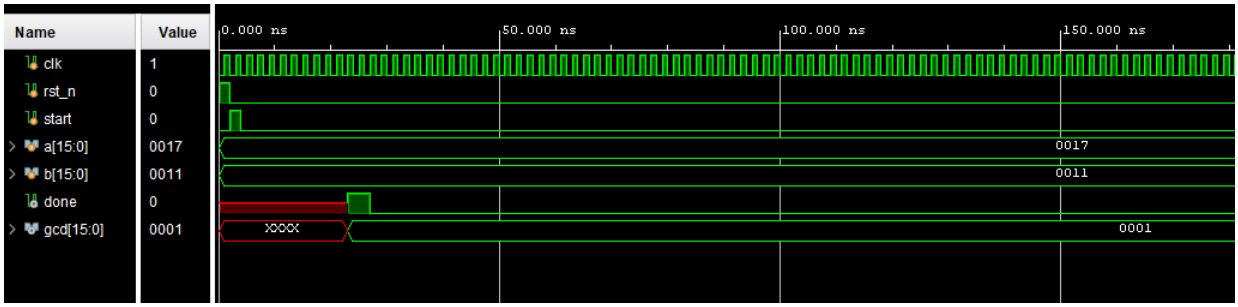
依照助教給定的虛擬碼進行實作，在重設時改變狀態到 WAIT，在啟動時改變狀態到 CAL，完成運算時（也就是 A,B 其中之一歸零時）改變狀態到 FINISH。

為了維持有兩個時脈的 Done，在完成運算時，我們將 Done 設為高電位，並且設立一個 Dummy State，在 FINISH 狀態等候一個時脈後，轉移到 Dummy State，並且將 Done 設為低電位。藉由設立一個新的 Dummy State，就能輕鬆延長 Done 的維持時間。

(2) 驗證：

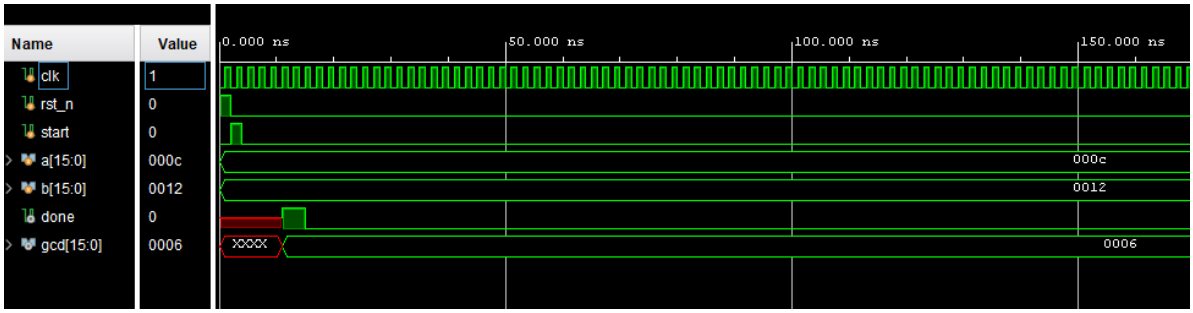
測試資料 #1

A=23, B=17



測試資料 #2

A=12 B=18



4. Booth Multipler

(1) 設計說明：

參考自維基百科

1. Determine the values of *A* and *S*, and the initial value of *P*. All of these numbers should have a length equal to (*x* + *y* + 1).
    1. A: Fill the most significant (leftmost) bits with the value of **m**. Fill the remaining (*y* + 1) bits with zeros.
    2. S: Fill the most significant bits with the value of (−**m**) in two's complement notation. Fill the remaining (*y* + 1) bits with zeros.
    3. P: Fill the most significant *x* bits with zeros. To the right of this, append the value of **r**. Fill the least significant (rightmost) bit with a zero.
2. Determine the two least significant (rightmost) bits of *P*.
    1. If they are 01, find the value of *P* + *A*. Ignore any overflow.
    2. If they are 10, find the value of *P* + *S*. Ignore any overflow.
    3. If they are 00, do nothing. Use *P* directly in the next step.
    4. If they are 11, do nothing. Use *P* directly in the next step.
3. Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let *P* now equal this new value.
4. Repeat steps 2 and 3 until they have been done *y* times.
5. Drop the least significant (rightmost) bit from *P*. This is the product of **m** and **r**.

該算法的核心精神為「利用二進位的性質，將一堆加法組合成少量的加法與減法」，如果要計算一個 N 位數布林乘法，該演算法需要 $O(N)$ 個時脈，並且需要 $O(N)$ 個電晶體。與暴力法相較，暴力法需要 $O(1)$ 個時脈，但是需要 $O(N^2)$ 個電晶體，由此可知該算法有「以時間換取空間」的特性。

假定 m 為最負整數，也就是在 N 位數二補數系統下的 -2^{N-1}，$-m$ 的值沒辦法在 N 位數二補數系統下表示，因為 $2^{N-1} > 2^{N-1} - 1$。為了避免溢位，我們將 m 轉為 $N+1$ 位數二補數系統，並在 A, S, P 前面 append 一個空格，藉此放入額外的位元。

從 N 位數二補數系統轉換到 $N+1$ 位數二補數系統的流程如下，我們分成兩個情況來討論，第一種情況是待轉換的數字 $x > 0$。在該情況下，僅需複製原本的數字，並在最高位補上 0 即可。

第二種情況是待轉換的數字 $x < 0$，首先，我們要計算在 N 位數二補數系統的 $-x$，並將 $-x$ 執行位元反轉後再加一，最後在最高位補上 1 即可。

上述流程可以用這段簡潔的虛擬碼表示。

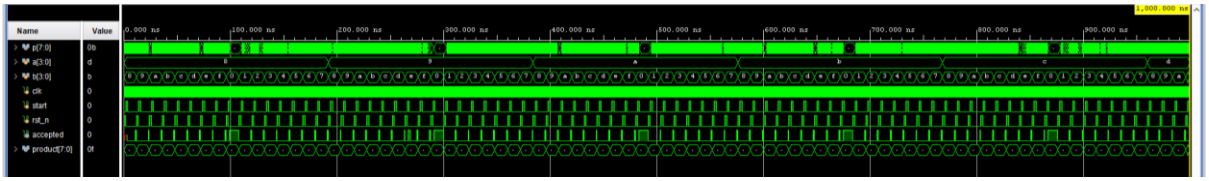
```
assign a_5bit = (a[3] == 1'b0 ?
    {
        1'b0, a[3:0]
    } :
    {
        1'b1, ~$unsigned(-a) + 1
    }
);
```

最後，跟著維基百科給定的方法，即可完成晶片設計。

(2) 驗證：

測試資料 #1

枚舉所有可能，並用肉眼觀察是否相等，Accept 為兩者是否相等。



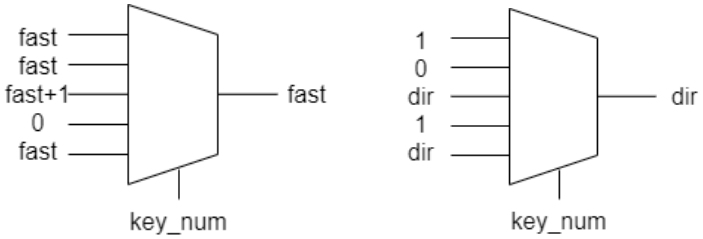
5. FPGA_Music

(1) 設計說明：

利用 KeyboardDecoder 得知現在所按的按鍵是哪一個，並分別將本次所使用的按鍵”W、S、R、Enter”，改變 key_num 的值進行辨認，利用 key_num 的改變，來執行動作。

每個 posedge clk 時當按鍵被按下，且 key_num 不等於 3’b100(也就是並非要用的按鍵)，則會判斷是哪個按鍵並進行動作。

- W：key_num = 3’b000。
  dir = 1 方向向上， fast = fast 速度不變。
- S：key_num = 3’b001。
  dir = 0 方向向下， fast = fast 速度不變。
- R：key_num = 3’b010。
  dir = dir 方向不變， fast = fast+1 速度變快/慢。
- Enter： key_num = 3’b011。
  dir = 1 方向向上， fast = 0 速度回到慢。



Rst 以 wire 的型態用 assign 進行操作，當 key_num = 3’b011 則 rst 進行。

音樂的部分，則是看 dir 的值，dir = 1 則上行，dir=0 則下行。速度則看 fast，fast = 1 時每 0.5 秒一個音，反之每 1 秒一個音。若以達最頂(C8)或達最底(C4)則維持在該音不變。

6. FPGA_Vending Machine

(1) 設計說明：

首先，我們將所有的輸入裝上 Debounce 以及 OnePulse

1. 鍵盤輸入 A, S, D, F
2. 按鈕輸入上、下、左、右、中

再來，構造兩個自動機狀態，輸入以及退錢。初始狀態為輸入中，當 A, S, D, F 或是取消被按下時，就轉移到退錢狀態，如果是左、右或是中被按下，那就增加金額。在退錢狀態時，每一秒扣五塊，歸零的時候，就轉移到初始狀態。

Contribution

1. 吳邦寧

第三、四題之實作及報告之撰寫、販賣機之實作。

2. 張芯瑜

第一、二題之實作及報告撰寫、音階盒之實作。

What have we learned?

1. 多去看討論區！
2. 瀑布開發的效率很高，非常適合趕工用。
3. 不要寫髒 Code，未來的自己會討厭你。
4. 盡量維持模組的可讀性，未來的自己會很感謝你。
5. VERILOG 是同時進行所有 MODULE，而不是相 C 一樣有順序，要注意!