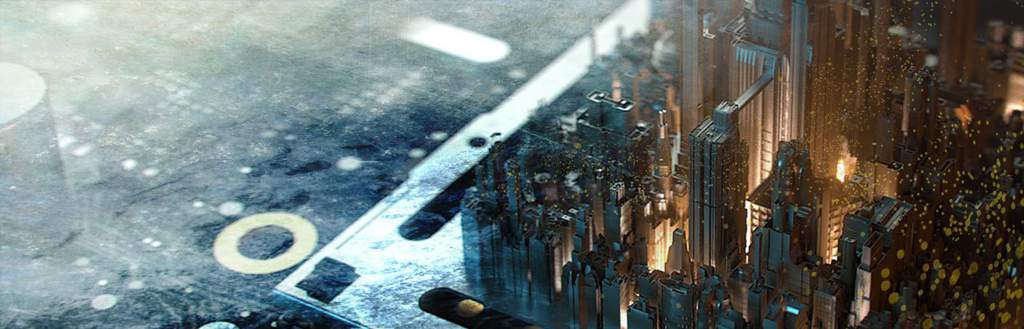
****

**Lab 5 report**

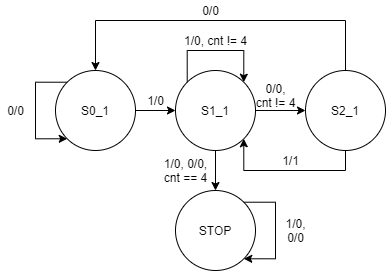
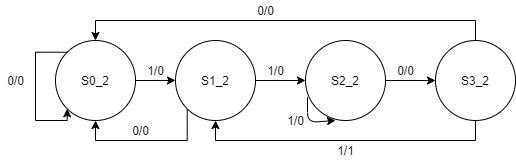
**2020/11/26**

**108062125 高敦晉**

**108062229 陳皇佑**

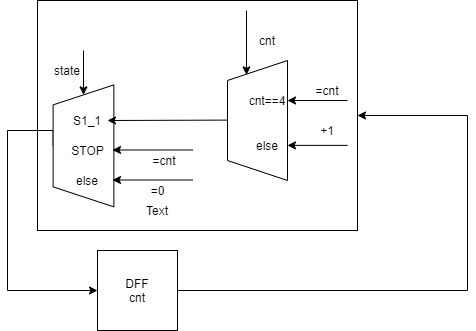
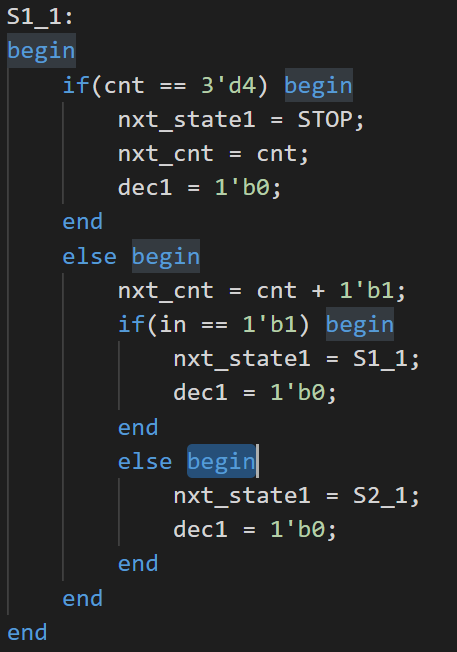
**Advance Question 1**

**State-transition Diagram**

**** dec1 statedec2 state

**設計構想：**

先寫出dec1和dec2的state transition diagram，因為dec1 →101，dec2→1101的最後一個和數列第一個皆為1，因此可重複使用state S1\_1和S1\_2。再另外寫一個counter紀錄state1在S1\_1待了幾次，如果已經4次了，state1就轉到STOP state，並在那裏自轉。



counter

**Testbench設計構想：**

從0000~1111跑過各種可能，並在1111後加入幾個101測試dec1。

**開發過程中的問題/學習：**

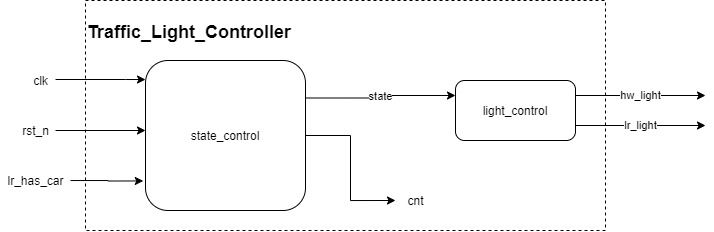
測試連續數列，可以重複利用前面的state節省空間，而如果state轉換有額外的條件，只需要在轉換state的combinational circuit寫好條件判斷即可。

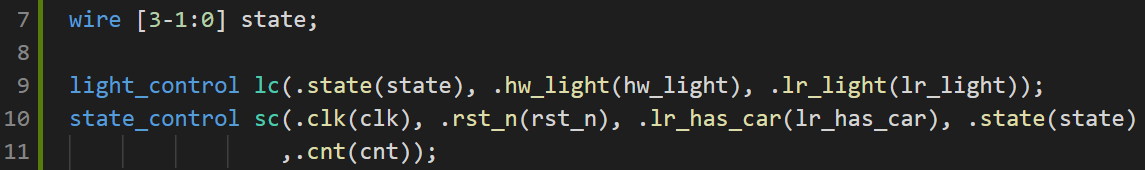
**Advance Question 2**

**設計構想：**

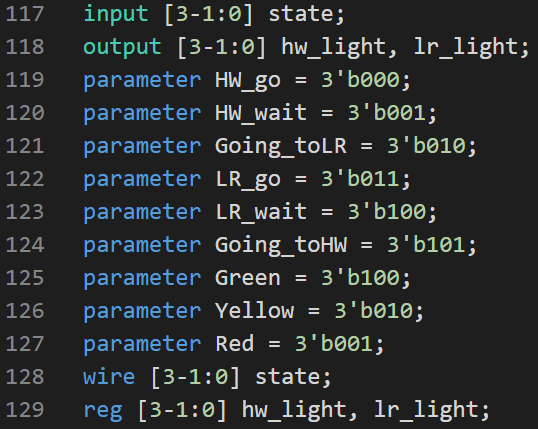
整體來說分成light\_control unit跟state\_control unit，一邊純處理state，一邊純根據state處理output。

整體的hierarchy如下。





**Light Control Unit:**

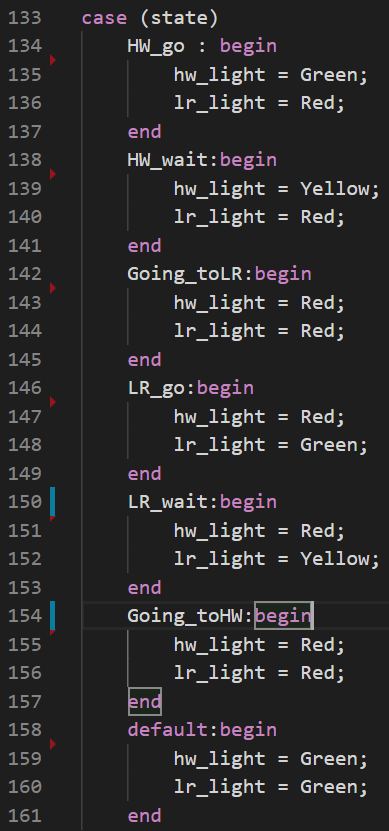
將state用parameter定義出來

HW\_go/LR\_go: 東西/南北向通車。

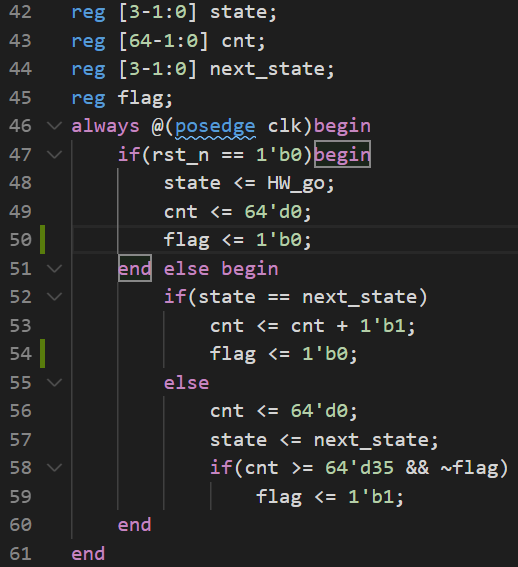
HW\_wait/LR\_wait : 東西/南北向轉黃燈，南北/東西向維持紅燈

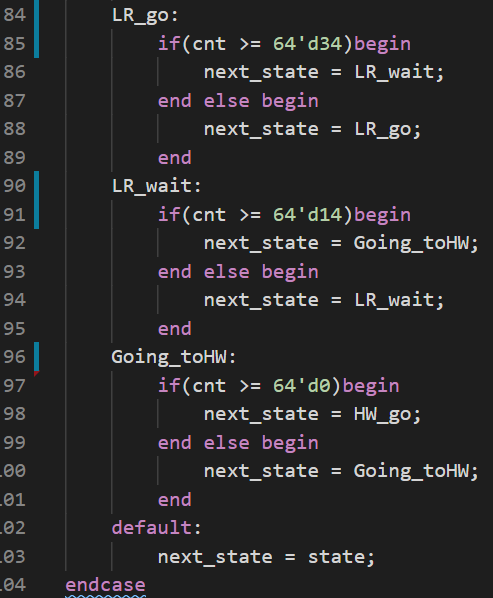
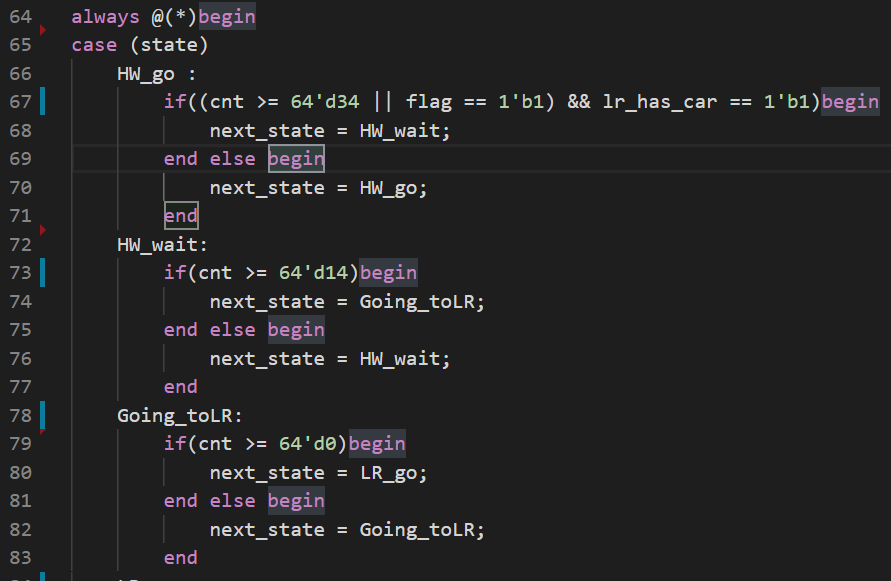
Going\_toLR/Going\_toHW:全部紅燈，下一個要讓南北/東西向通車

Green/Yellow/Red分別代表哪個燈亮。

根據目前的state定義每個燈號。

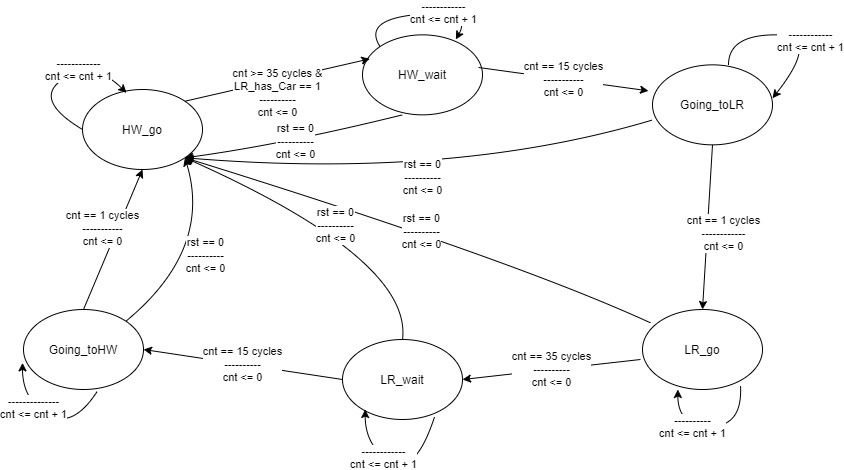
**State Control Unit :**

用cnt來數有幾個cycle，如果下一個state換與現在的state不同，就重置cnt。因為怕cnt有天會溢位然後以為沒超過35，因此多設一個flag來表示已經數超過一次35了。



將每個state根據當前的input以及cnt，指定下一個轉換的state。

下圖附上

state transition diagram

“-------"的上面代表條件，如果沒有寫就是default往這裡走。Rst的條件優先度最高。下面代表會作的操作，主要是處理cnt的計算。

**開發過程中的問題/學習：**

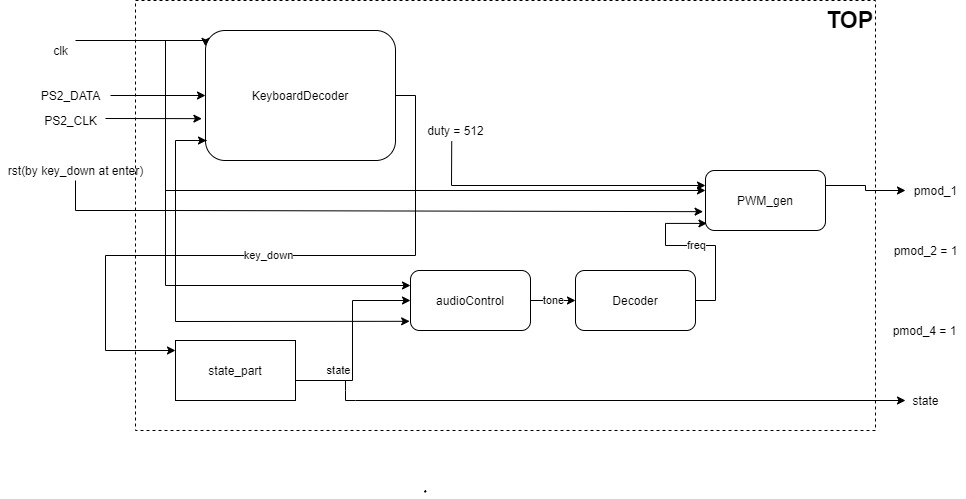
這題主要學到的部分是在估算cnt時的條件式，要寫成 cnt >= (期望的cycle次數 – 1)由於cnt是從0開始數的。

此外就是練習FSM的規劃以及實作，在這次的Top module 只簡單的放入了兩個submodule，在做測試時，可以很輕鬆的更改來接出wire，可謂體會FSM的強大。

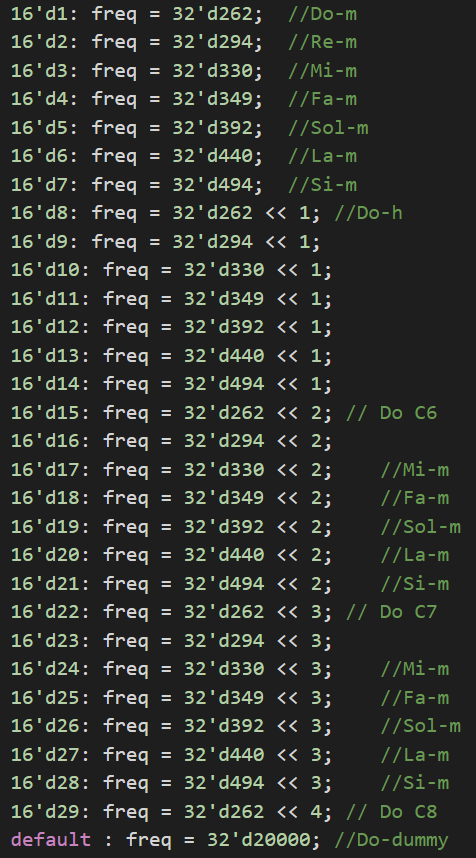
**FPGA-1**

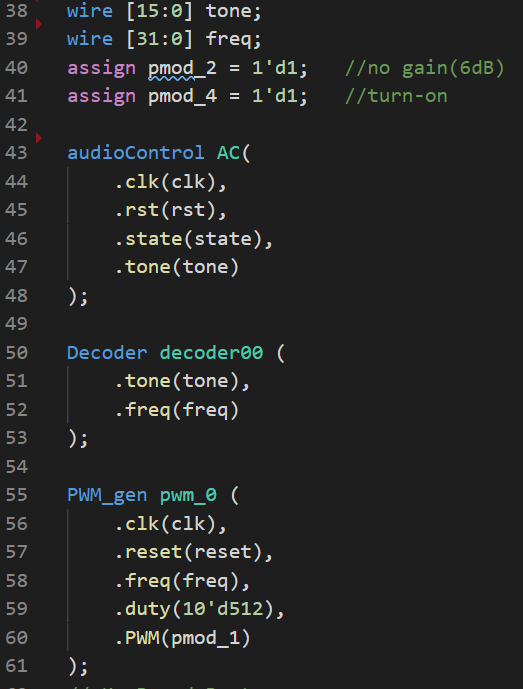
**設計構想：**

分成audio part, keyboard part, state part 三個部分

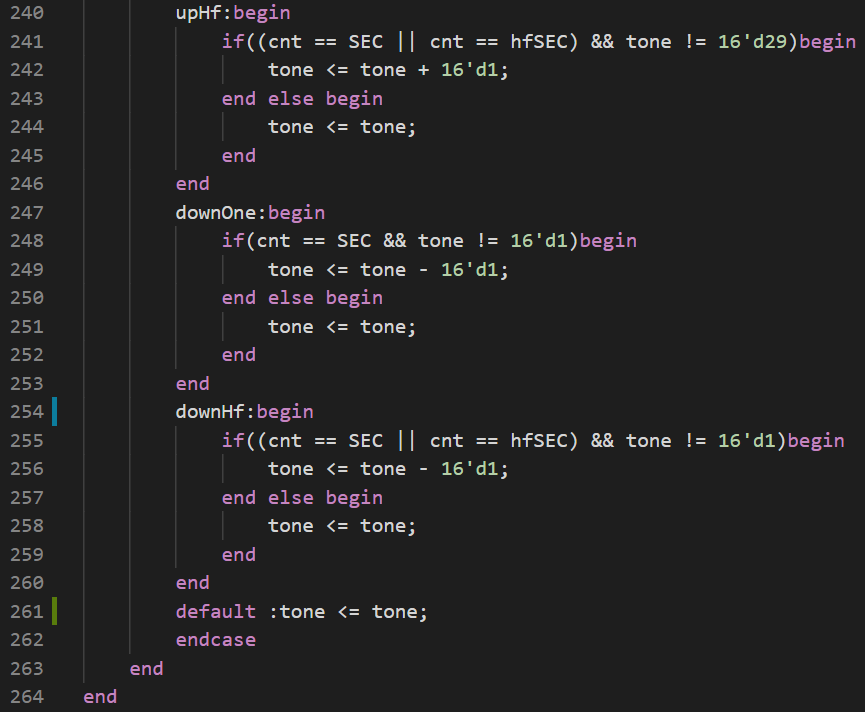
整體的hierarchy

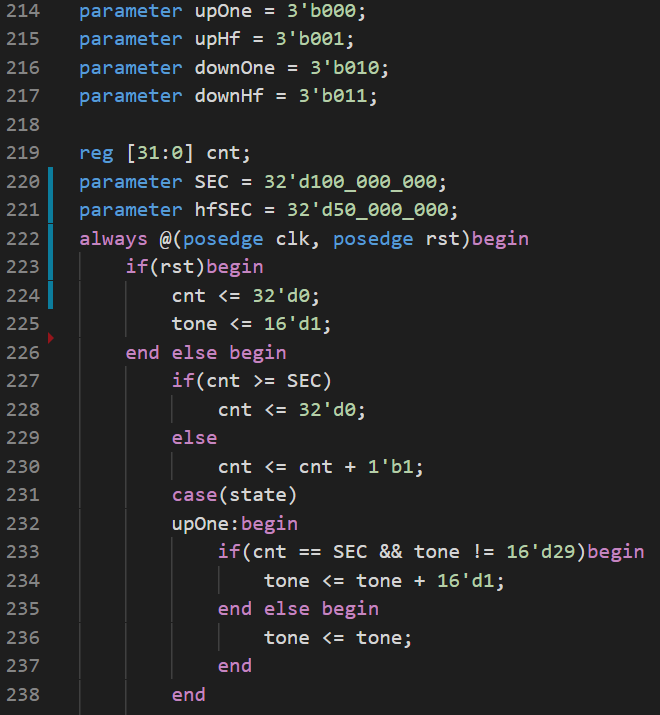
Audio part

**Audio Part :**

大致上跟 demo時用的code差不多，新增了一個audioControl來做根據state 輸出tone。

另外也更改了Decoder.v裏頭的freq音高。

**audioControl :**



state的定義：

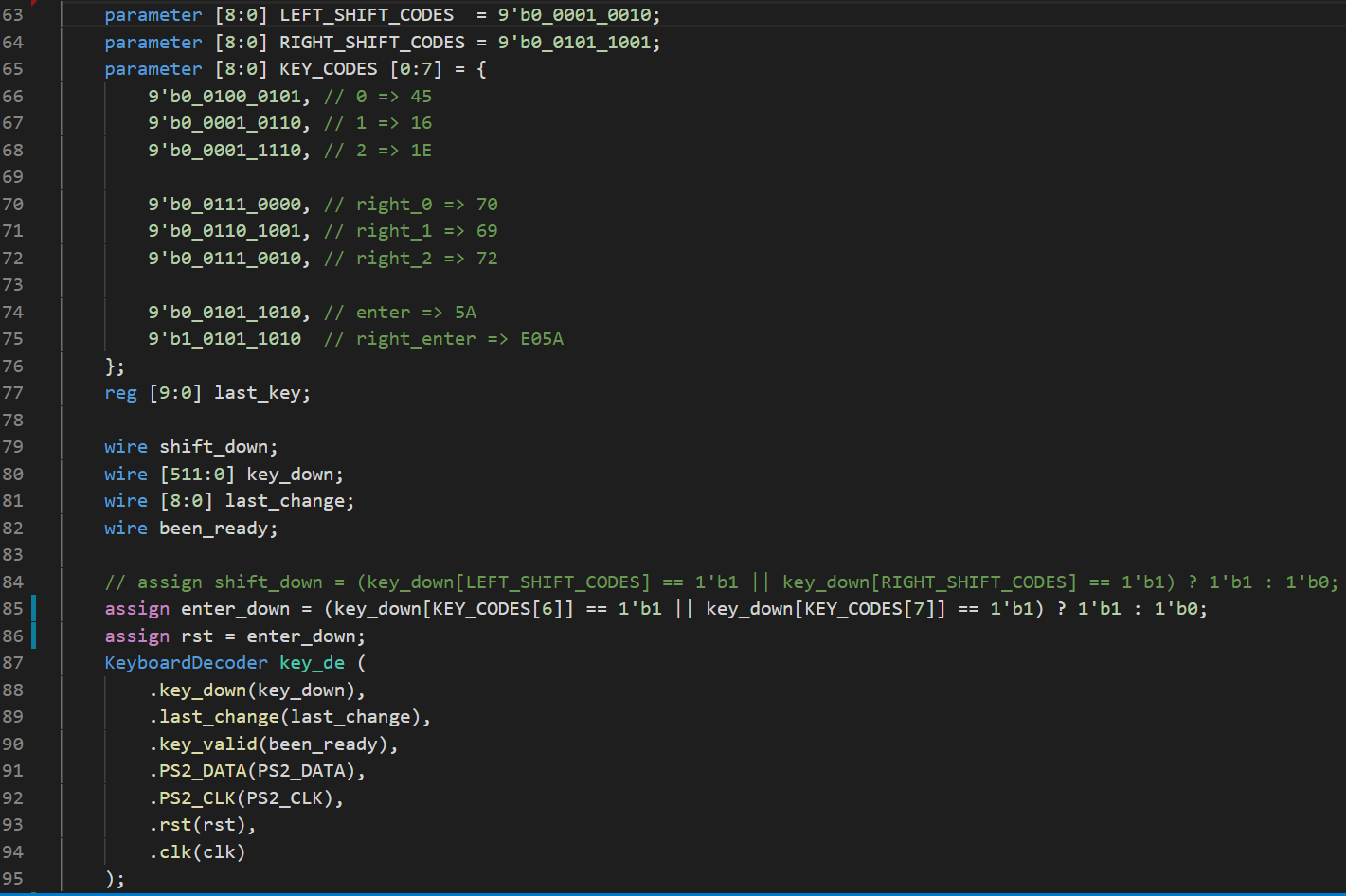
downOne : 往下1Hz數音高，downHf : 往下2Hz數音高。

upOne : 往上1Hz數音高，upHf : 往上2Hz數音高。

rst時，將tone直接設成1用cnt一直數到100,000,000，每數到100,000,000就增加/減少tone，製造出2Hz的方式是在state屬於Hf系列時，在條件式裡多增加 if(cnt == hfSEC)

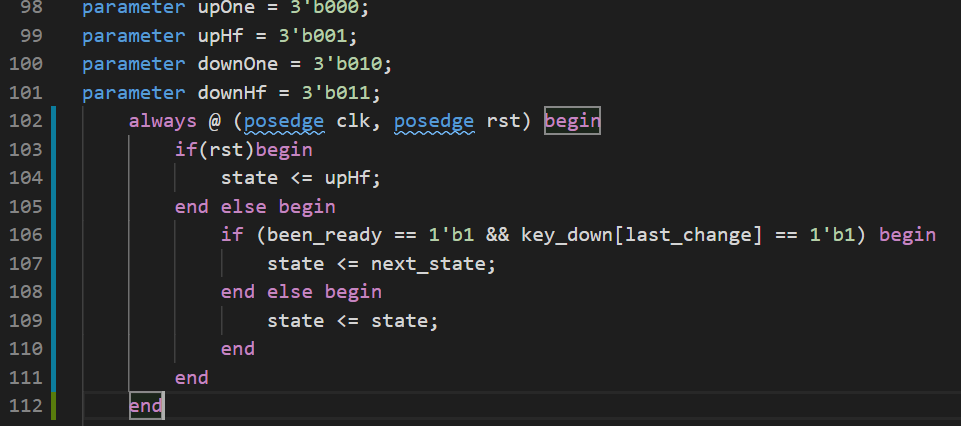
而hfSEC == SEC/2=50,000,000，如此便能只用一個cnt製造出半秒增加/減少的效果

**Keyboard Part :**

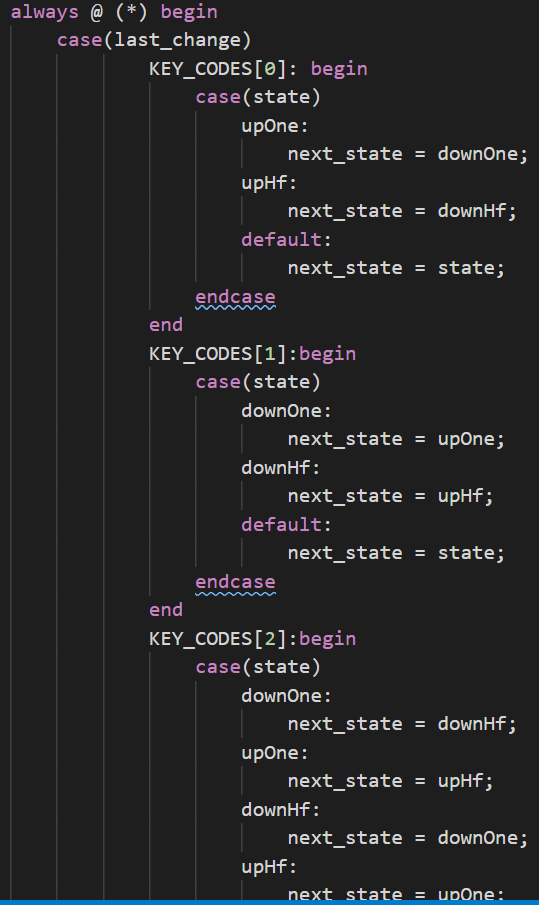
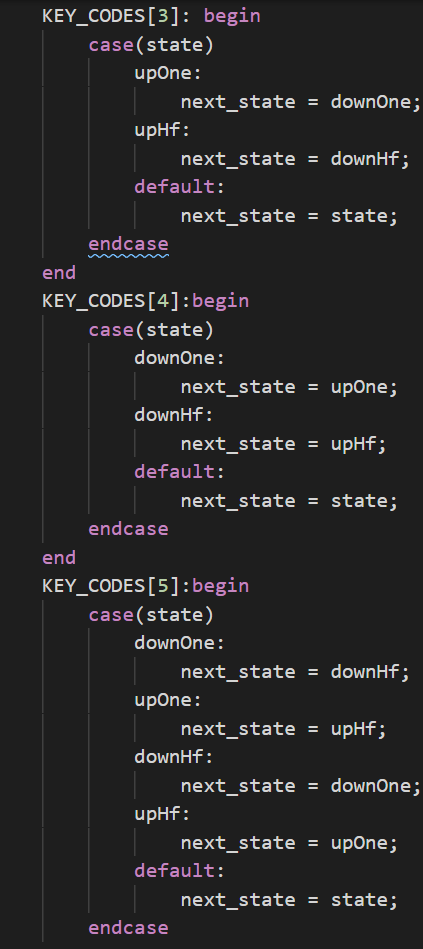
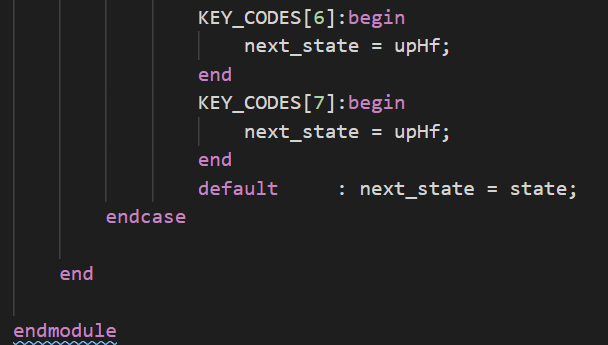
****

大致上也跟demo的範例keyboard code一樣，差在按鍵只剩下需要的0, 1, 2, enter code。

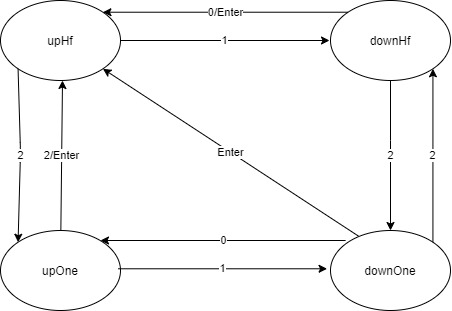
**State Part :**

****

處理state，如果rst == 1，state直接reset到upHf，如果有鍵被按下更新到next\_state，否則就保持當前的state。next\_state的處理在下方。

 處理next\_state的去處，根據KEY\_CODE來做變換，default是不變state，state transition diagram在下方。

這邊要注意的是由於前面已經有判斷是否有按鍵按下，故不用再處理沒按鍵的case。



**開發過程中的問題/學習：**

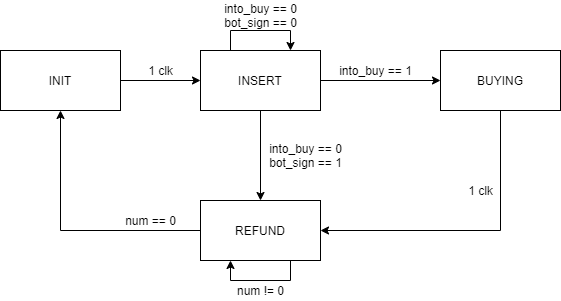
在寫這份題目之前，仔細的trace demo時的keyboard 以及 piano code一次，有練習到看懂整個架構的速度並擷取需要用的部分做使用、並能夠自行修改使其符合需要。

這個題目的設計結構中途大改了一次，原先我是希望能夠吧keyboard的訊號接成一個botton的感覺，然後再接進去一個stateControl的module，但是後來一直弄不好botton，所以就直接沿用piano的架構，拿Keydown以及last\_change作為state transition的依據。

再一次體會到FSM的重要性，因為中途改架構的關係，多虧一開始就有先想好整體state的結構，更改code時也方便許多，很多部分都是小改後就能直接沿用的。

**FPGA2**

**State-transition Diagram**

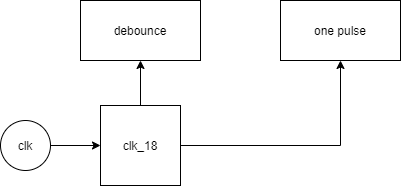


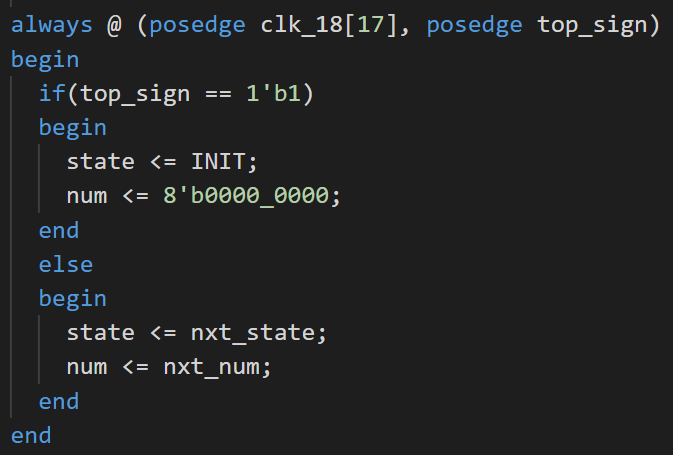
Vending machine

**設計構想：**

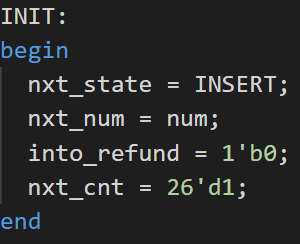
先把所有按鈕接上debounce跟one pulse，而state的dff,

Debounce, one\_pulse，都接上clockdevider(clk\_18)，因為debounce要接上延長過的clock才算真正除顫。

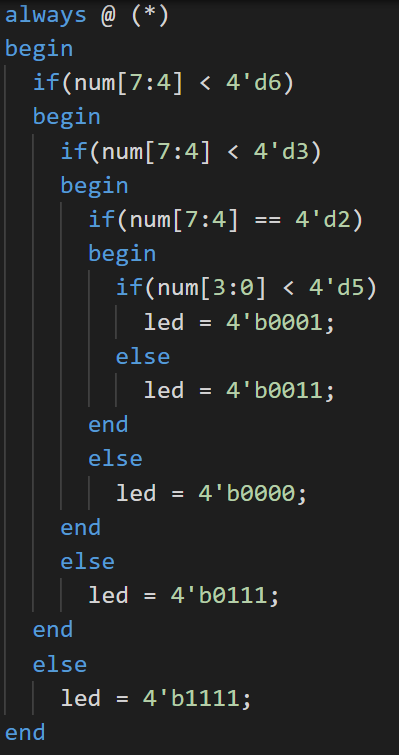
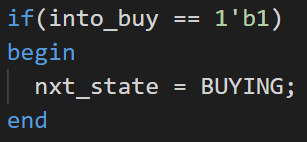
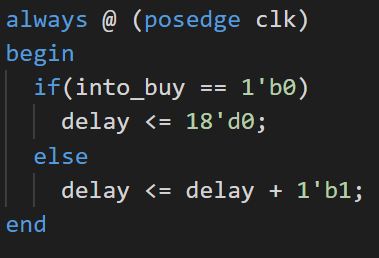




在INIT state 初始化num、cnt、into\_refund。



into\_buy == 1代表在INSERT狀態下asdf被按下，且金額足夠購買，下一個state為BUYING。而因為keyboard\_decoder的clk訊號是接原生clk，故into\_buy需要使用delay延長clk\_18個長度。當delay == 1時代表現在已經有一個購買正在進行，此時再按其他商品並不會進行購買。把按哪個商品用4bit drink\_code表達，哪些商品能夠購買用4bit led表示，只要drink\_code & led不為0000代表可以購買。

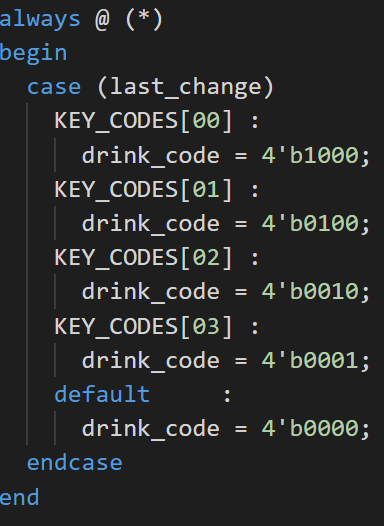


Led[3] 代表coffee

Led[2] 代表coke

Led[1] 代表oolong

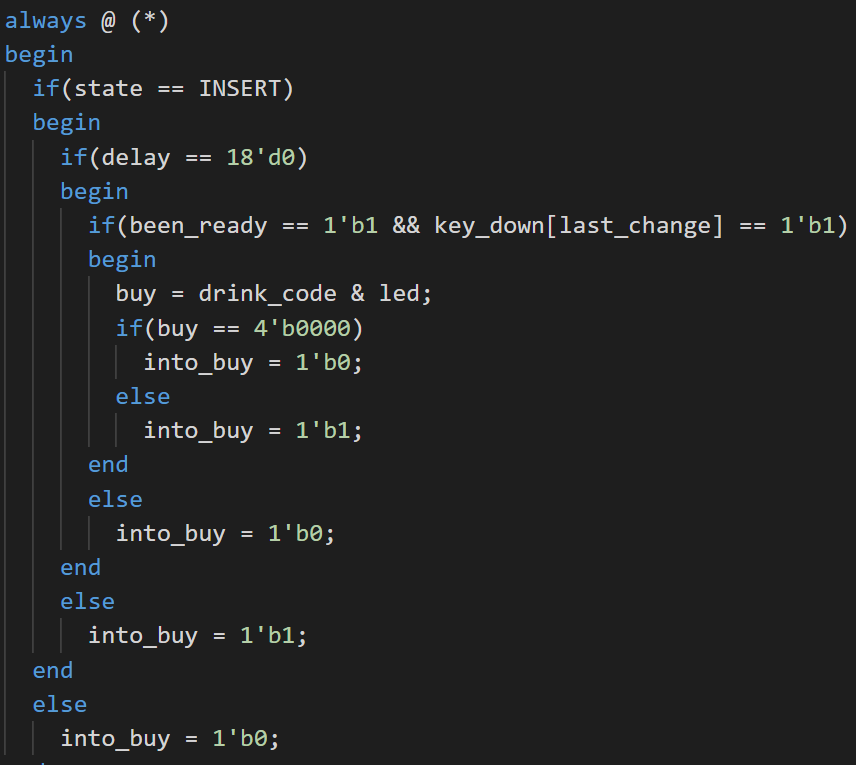
Led[0] 代表water

drink\_code[3] 代表coffee

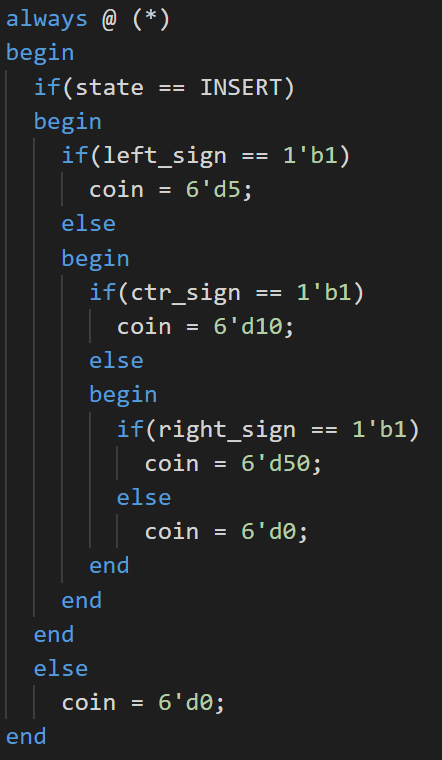
drink\_code [2] 代表coke

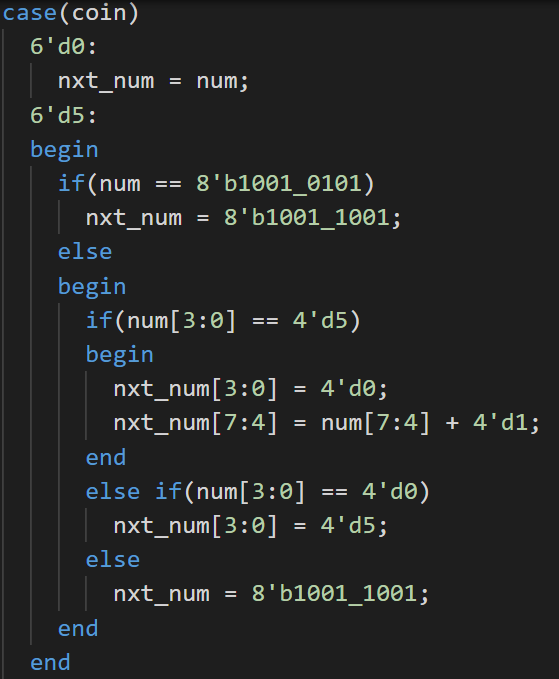
drink\_code [1] 代表oolong

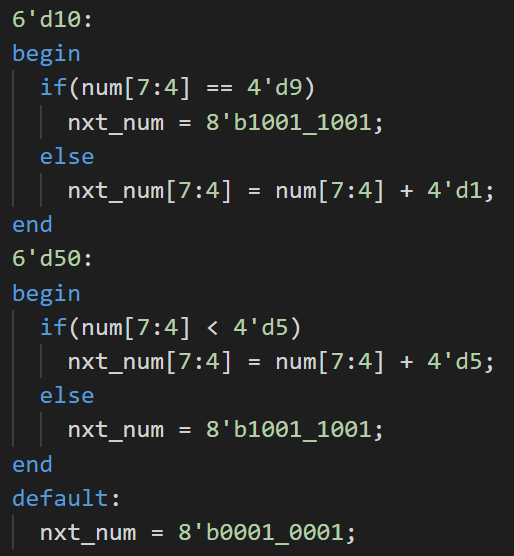
drink\_code [0] 代表water



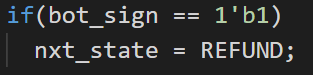
用coin紀錄投入多少錢。



接著在INSERT state中依coin改變num，

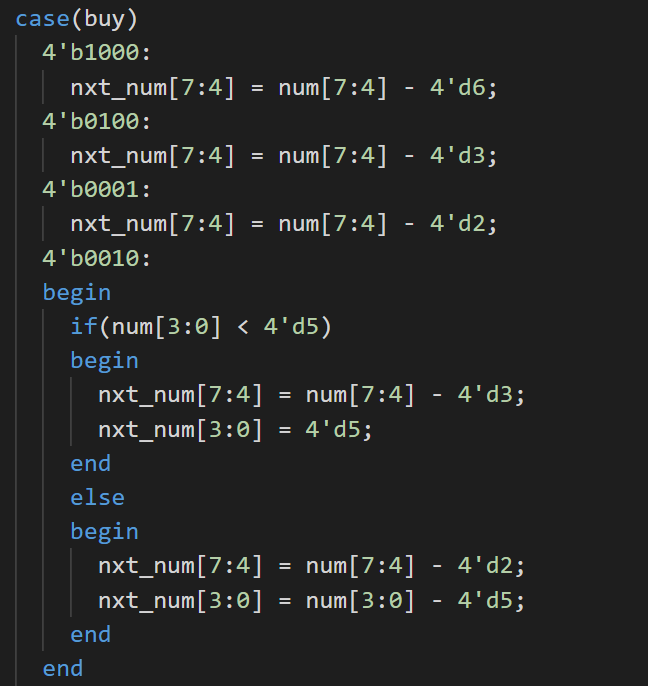


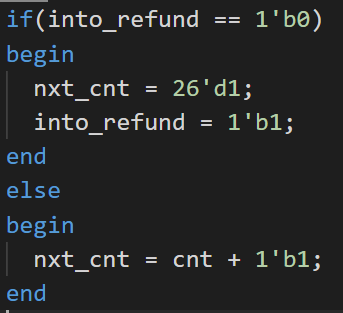
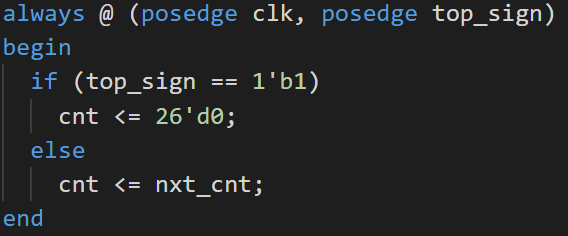
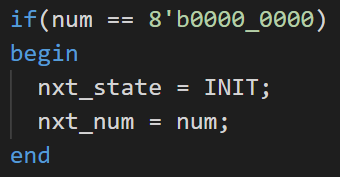
在INSERT中，按下bottom button進入REFUND state。

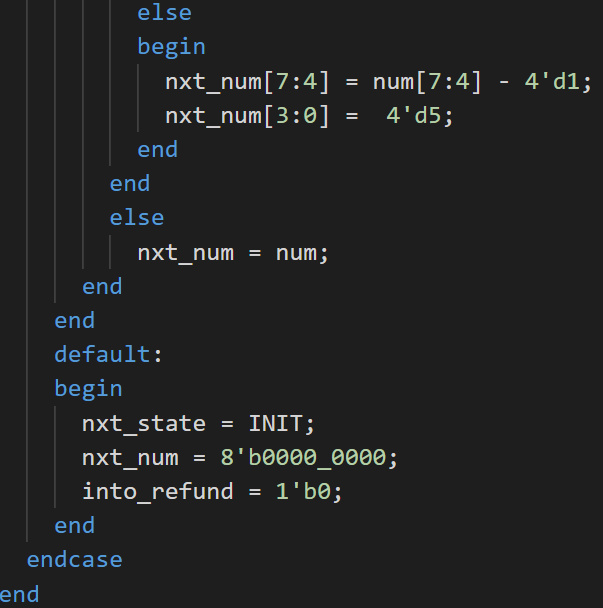
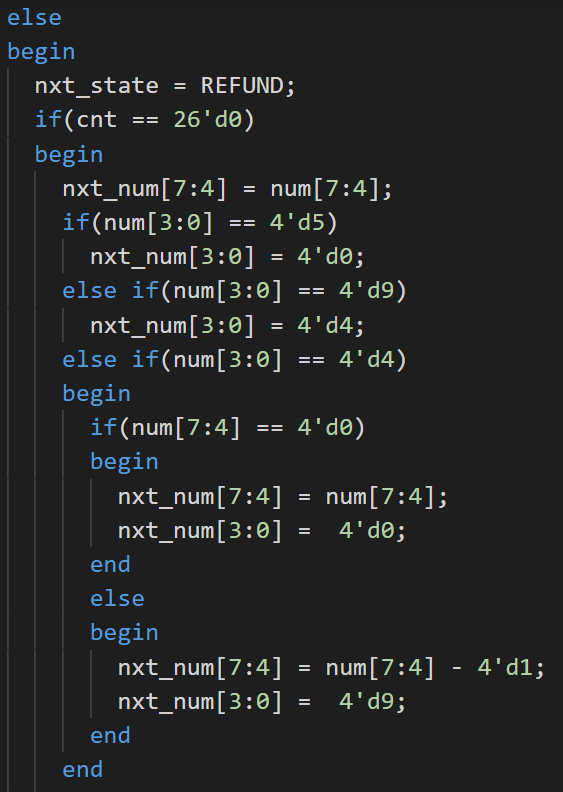


BUYING state中依買入商品更改num，並轉到REFUND state。





第一次進入REFUND，reset cnt，接著用cnt實現一秒變動一次的條件。接著看現在num為多少，將num減5。當num為0，下個state為INIT，num keep 原值。  
   




**開發過程中的問題/學習：**

一開始按reset時state就會卡住，之後才發現clock divider不能接reset，不然reset訊號會一直維持在1。一開始也讀不到鍵盤的訊號，才發覺因為keyboard\_decoder是接原生clk，訊號維持時間太短，需要將期延長。而在做數字加減時原本等號兩邊都放同一個變數，結果會一直跳warning，說會形成self-loop，不能generate bitstream，才知道好的coding習慣真的很重要。

分工 :

AQ1、FPGA2 ,report: 陳皇佑

AQ2、FPGA1, report&report合併 : 高敦晉