EECS 2070 02 Digital Design Labs 2019 Lab 5
學號：107062314　姓名：陳柏均

0. 前言

Vending Machine 一開始聽到就覺得是一個滿複雜的設計，要模擬一個簡單的販賣機我想有一定的困難度，這一次 lab5 只有一題加一 bonus，但花的時間卻沒比以前少，當然收穫同時也是比以前多。

經過之前的 lab 的之後，也同時運用之前所學及不容易再犯以前的相同錯誤。 這次 lab5 我認為也很容易在小地方出錯，而在打程式時，我也看著 FPGA 板上顯示出來的錯誤慢慢去做修正，以得到正確的答案。

1. 實作過程

一.

這次 lab5 我打的方式與之前的 lab 有些不同，以前我主要都是看著題目先把全部的 code 先打完然後呈現在 FPGA 板上，之後在看著 bug 慢慢去 debug，但我看完題目之後，我便決定改變我主要打的方式。

我大致上使用的方法是一個一個功能打，然後有錯誤便可以馬上更正。雖然說燒板子真的挺浪費時間的，但是以這種方式對我而言反而比較好。這樣打可以幫助我較容易達成題目的要求，同時也讓頭腦的思考能清晰且有條理性。

如 state 的部分，我一開始打的時候只有 initial 與 deposit 兩個 state，我將其個別的功能先做好後來再去處理後面 buy 及 change 這些比較麻煩的部分。以前的 lab 有較多題但像採取循序漸進的方式，一題跟一題之間通常都有一定的關係，而照著順序寫對於整個寫 code 的流程真的是很有幫助，前一題的想法通常可以用在下一題，犯過的錯誤也幾乎不會再發生。

由於這次與之前的複雜度和循序漸進式不同，若是一次將全部打完再 debug，對於我有一定的困難，頭腦也無法一次運作那麼多的東西。這時候我相信井然有序的思緒更顯重要，而這次的這種方法幫助我甚大，加快我除錯的速度。

二.

我使用的 state 與題目的一樣，分別是 INITIAL state, DEPOSIT state, BUY state 和 CHANGE state。在 INITIAL state 時我給全部我需要的變數初始值，而在

DEPOSIT state 的時候我用了 if else 去判斷按鈕，當然這些按鈕皆有做過 one_pulse，以避免按按鍵的時候下壓會讓數字一直跑；上次 lab 我處理這個問題很久，而這次並無什麼問題，由此可見上次學習到的錯誤真的可以大大加快這次這部分的速度。而在加錢的時候，我同樣使用 if else 去判斷現在的存款並去加錢。

另外這個 state 最難的部份我覺得是在顯示價錢的時候，我用了一個類似 counter 的東西在 always block 裡面去記按過幾下，當該飲料顯示價錢的狀況下，若再按一下且目前存款是足夠的，則就進到下一個 state，也就是 buy state 去進行購買的動作。

```
if(countA==1'b1&&enough_A==1'b1)
begin
    DetoBuy=1'b1; nextcoke=1'b1; nextcof
    nextBCD3=BCD3; nextBCD2=BCD2; nextBC
    //nexttmpBCD0=tmpBCD0; nexttmpBCD1=t
end
```

(註:使用一個 reg 去記住按過了幾次，價錢若足夠且按兩次則進入購買狀態)

三.

在 buy state 的時候，我看了 pdf 上給的英文字母對照表選自己想要的去呈現，在 lab4 時就有學到相關知識，因此沒什麼大困難。 但我這裡有件我認為挺有趣之事，那就是我從無想過七段顯示器竟然可以顯示英文，雖然老實說真的要發揮一些想像力才能夠聯想到一些顯示出來的到底是哪個英文字母，我認為有人竟然能想到使用七段顯示器去表示全部英文字母真的是十分厲害且有意思。

在這個 state 我用了 clk26(clk/2^26)，與題目的建議一樣，使飲料的名字能在四個顯示器上顯示大概一秒後進到下一個 CHANGE state，進行找錢的模擬動作。因為最多只能有四個字母可以顯示，因此在這裡我所選的飲料是 cofe 代表 coffee 以及 coke 兩種以讓我在 buy 的時候可以顯示出這兩個飲料。

(ii)

```
4'b1010:begin display=7'b1110010; end //c=10
4'b1011:begin display=7'b1100010; end //o=11
4'b1100:begin display=7'b0111000; end //f=12
4'b1101:begin display=7'b0110000; end //e=13
4'b1110:begin display=7'b0101000; end //k=14
```

(註:除了數字的顯示之外，我多加了五種可以顯示英文的 seven segments)

四.

最後一個 state 為 change state，功用為退買家(FPGA 板操作者)錢，這個部分我用 DFF 來實作並進行類似 down counter 的操作以進行還錢。

前面當 BCD1(balance 的第二位)不為 0 時，則 balance 的十位數不斷進行一次減一的動作，而當只個位還剩五的時候，則個位數減五。

而 led 燈的部分，看助教的 demo 影片有注意到它會慢一個 cycle，因而我在給值的時候是給 drop_money 的下一個狀態值。

```
begin
    nextstate=`INITIAL;
    nextBCD0=BCD0;
    nextBCD1=BCD1;
    nextBCD2=4'b0000;
    nextBCD3=4'b0000;
    nextdrop_money=10'b1111100000;
end
```

(註:給 drop_money 正確值以亮燈)

起初我多寫了一個 state 去進行前面在 DEPOSIT 階段時要進行 cancel 的行為，而我後來才發現其實它跟 change 做的事情實際上是一模一樣的，可以併用一個 change state。

一個是從 deposit 這一個 state 直接到 change state，而另一個則是經由 buy state 再進到 change state。 且前者是將當前的 BCD(顯示器上呈現出來的東西)直接傳到下一個 state，後者就必須做一些變化。

(ii)

這次的 bonus 我認為跟上次頗像，目的是要讓 FPGA 板在使用者閒置它的狀態下直接退錢，也就是 change state 所做的事，而這題的重點我認為就是如何判斷閒置並且使它進到 change state。

由於題目的建議是 clk29(clk/2^29)，跟我在跑 DEPOSIT state 的 clk(clk/2^16)差了 2^13。我用了一個類似 counter 的變數去存目前的數字，每次當我對 FPGA 板有動作時(按按鈕)，我就讓這一個 counter 歸零，而當它在這個 cycle 沒有動作時，我就加一。

一直到這一個數字加到 14'b10000000000000 時，代表它已經閒置了大

約五秒，當遇到這一個情況時，就當使用者要退錢，下一個 state 直接進到 change 以執行。

```
else if(((cancel_pul==1'b1)||(idle==14'b10000000000000))
    begin
        DetoChange=1'b1;
        DetoBuy=1'b0;
        nextBCD0=BCD0;
        nextBCD1=BCD1;
        nextBCD2=4'b0000;
        nextBCD3=4'b0000;
        nextidle=0;
    end
```

(註:當 idle 到達 14'1b10000000000000 時，即代表已閒置五秒鐘而進行下一步)

2. 學到的東西與遇到的困難

(i)state diagram 的運用

這次的 lab 幫助我更加熟悉 state 運用的方法，我使用 case 去將不同的 state 分開寫，這樣就可以完全專心去做每一個 state 中需要做的事而不用去管別的 state 的東西。如上面所寫，我這一題共使用 4 個不同的 state, INITIAL, DEPOSIT, BUY 與 CHANGE。

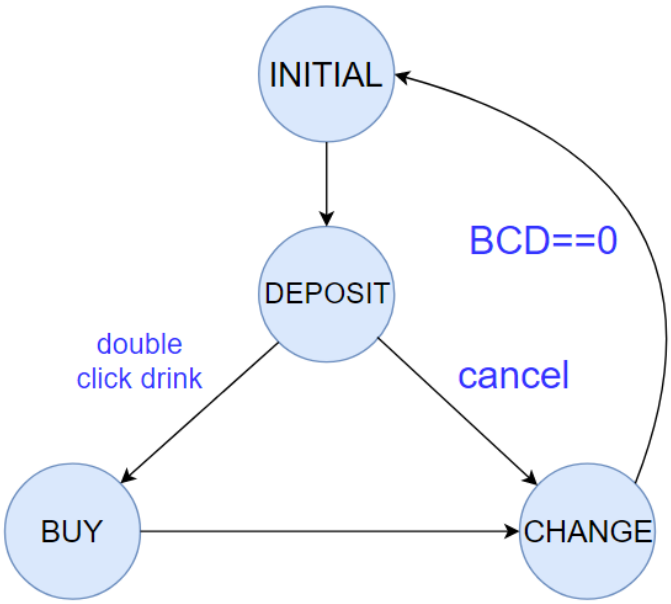
INITIAL 目的是回到初始值，而之後這一個 state 馬上無條件跳到 DEPOSIT，而在 DEPOSIT state 的時候是需要寫判斷最多的地方，若雙擊某一個飲料則跳到 BUY state 中表示進行購買。

BUY state 則是專門負責顯示目前買的飲料是哪一種，接下來便無條件跳到下一個 state CHANGE 以進行找錢的功能，並且此時下一個右邊兩個數字是存款去減掉飲料價格。

而當在 DEPOSIT state 的時候假如直接按 cancel 的話，也表示要退錢的意思，而這時候只要將左邊兩個 digit 所呈現的價格歸零，也是做一樣退錢的步驟。而主要的差別就是在 BUY state 的時候必須再次給上上次之值，因為它上一個值是英文字母。

用 state diagram 對我幫助甚大，不但快速知道是哪一個 state 發生了

問題，更可以讓思路更加明瞭，雖說之前多少有用類似的方式去打，但是我認為這是 lab5 的最主要架構，相信也是老師及助教要我們所學。這個觀念以後一定也會常遇到，而這次 lab5 我深信我也更加熟悉之中的用法以及道理。



(註:自己畫的簡略 block diagram 以釐清 state 跟 state 之間的轉換)

(ii)暫時記住數字

由於在顯示的部分上，我全部皆是用同一種變數去改變當前的顯示，我用 BCD 來表示現在 FPGA 板上應該出現的數字。上次 lab 即使用這種方法去寫，也沒有什麼大問題，但當我做到從 BUY state 到 CHANGE state 的時候，我就碰到了一個困難；那就是我在從 DEPOSIT state 到 BUY state 時，我已經將 BCD 變數改成我想要呈現的英文字母了。

而在做的時候，如果我直接拿當下的數字去做計算，它就會在板子上不斷呈現數字即英文字母，這樣的錯誤很明顯不符合題目所望。而我想到這個問題的主要解決方法就是再從 BUY state 到 CHANGE state 的時候，我就去判斷然後便給一個暫存變數值以做記住數字值，如此一來，當我需要用到這個數字的時候，我就可以從這裡面去取值，解決了經過一個 state 卻而無法得到上上次數字的困境。

```
if(BCD2==4'b0101&&BCD0==4'b0000)
    begin
        nexttmpBCD0=4'b0101;
        nexttmpBCD1=BCD1-4'b0011;
    end
else
    begin
        nexttmpBCD0=BCD0-BCD2;
        nexttmpBCD1=BCD1-BCD3;
    end
```

(註:在 BUY state 的時候我用條件去判斷並給 price 跟 balance 的差)

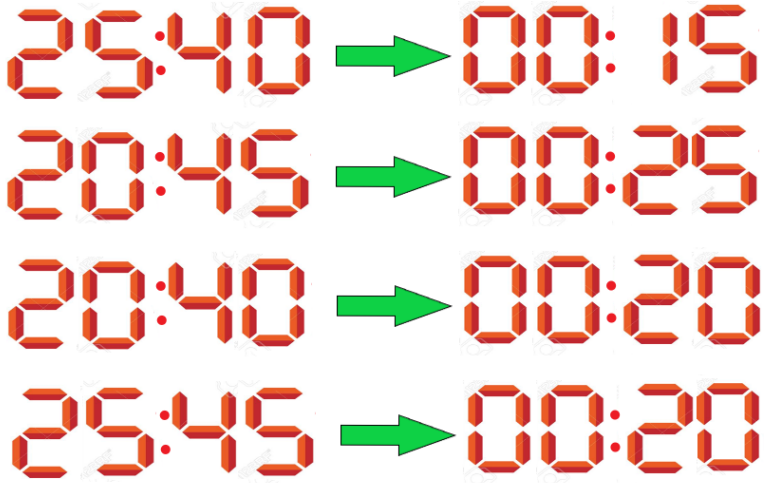
(iii)相減的狀況

在買飲料之後，必須將 price 及 balance 之差給 CHANGE state 以供退錢的動作，而在這裡我看 price 跟 balance 之值去給 tmpBCD0 及 tmpBCD1(暫時記住右邊兩位的變數)值。看個位數主要會有四種情況發生。

第一種是存的錢的個位數是 0，買的飲料價錢個位數是 5；第二種是存的錢個位為 5，而買的飲料個位為 0。而最後兩種分別為個位數皆為 0 以及個位數皆為 1 兩種情況。

經由仔細的判斷其實不難發現只有第一種狀況有需要借位的問題，也就是不能單看一個 digit 去進行計算，碰到這樣的情況時，就將 nexttmpBCD0 直接設為 5，而 nexttmpBCD1 減三(減二在減一)。

而其他第二三四種狀況，由於我們知道要 balance 的錢大於或等於 price，才能購買，也就是說這三種情況我只要將存的錢個別值(兩個 digits)直接扣掉飲料的錢(兩個 digits)就可以。



(註:算需要找的錢的時候需要判斷的情況)

(iv)進入 CHANGE 時餵給 BCD 值

由於在進入 CHANGE 這一個 state 之前，有兩種狀況會發生，因此我多加了一個變數 buyflag 去判斷上一個 state 為多少。若上一個 state 是 BUY 的話(buyflag==1)，表示我目前要扣的數字是上上次的數值，而非上次的英文字母；而此時我將 buyflag 設為 0，以表示下一次數字就可以直接拿上一次的進行運算。

同時我把之前記下來的數字餵給目前的 BCD0 與 BCD1，即右邊兩個 digits，以表示剩了多少錢並從那個值開始依題目條件扣。如此一來便可以回到一個我所要的循環。

```
if(buyflag==1'b1)
    begin
        nextstate=`CHANGE;
        nextbuyflag=1'b0;
        nextBCD3=4'b0000; nextBCD2=4'b0000;
        nextBCD1=tmpBCD1; nextBCD0=tmpBCD0;
        if(tmpBCD0==4'b0000&&tmpBCD1==4'b0000)
```

(註:用 buyflag 去決定要不要給 tmpBCD1 跟 tmpBCD0(上一次所記住的數字))

(v)初始位數的問題

在記右邊 balance 的數字時，我曾經遇到一個很大的 bug 導致我花很多時間在找錯誤，我發現一但我給它之前的值，它卻只能顯示 0 或 1。我一直想卻沒任何頭緒為何莫名會出現這樣的數字。

後來我才發現我犯了一個滿愚蠢的錯誤，那就是我不小心在 tmpBCD0 及 tmpBCD1 初始時，只給它們一個 bit，也難怪它顯示的時候只有 0 跟 1 兩個狀況，因為它們只有一個 bit。

```
reg [3:0]tmpBCD0;
reg [3:0]tmpBCD1;
reg [3:0]nexttmpBCD0;
reg [3:0]nexttmpBCD1;
```

```
reg tmpBCD0;
reg tmpBCD1;
reg nexttmpBCD0;
reg nexttmpBCD1;
```

(註:一開始我只給 1 個 bit，但實際上他們卻得存 4 個 bits)

(vi)判斷哪種飲料的方式

我 state 之中買的 state 只有 BUY, 但在此時我卻要依上一個狀況去顯示究竟是買了哪一種飲料, 而此時我一樣使用變數的方法去記住在上一個狀態時使用者買了哪一種飲料, 並依飲料去顯示。

```
if(coke==1'b1)
    begin
        nextBCD3=4'b1010;
        nextBCD2=4'b1011;
        nextBCD1=4'b1110;
        nextBCD0=4'b1101;
    end
else if(cofe==1'b1)
    begin
        nextBCD3=4'b1010;
        nextBCD2=4'b1011;
        nextBCD1=4'b1100;
        nextBCD0=4'b1101;
    end
```

(註:我用了 coke 及 cofe 兩個變數去判斷在板子上我該顯示哪一種)

3. 想對老師或助教說的話

這次 lab 花了我一整個假日才打出來, 初時看到題目時其實滿慌張的, 不知如何是好, 因為只有一題且條件算滿多的。但後來我發現就算是硬體語言其實也可以循序漸進, 慢慢地去做。

之前一股腦兒的就會將全部打完再除錯的方式似乎沒有很適合我。Lab5 雖然花了我更多時間, 但以此新方式, 我在做的過程中頭腦十分清晰, 一次只需要去管目前眼前所發生的事, 不必想太多。

比起之前數次 lab, Vending Machine 就更貼近生活, 這次的題目更帶給我一種學習的東西其實就在自己身旁的感覺, 不能否認過程中有數次撞牆期, 停滯不前, 感到沮喪; 但其實也是這些情緒使我的成就感昇華, 更加顯得珍貴而不凡。

完成 lab5 的當下, 身心是疲憊的, 但是所學到的新事物絕對是值得的, 我在 DEPOSIT 和 CHANGE state 兩個 state 中花了很久的時間去做, 但同時也是在這之中慢慢修正我錯誤的寫法, 以得到一個漂亮的呈現結果。

學習不就是如此? 若無經歷無數挑戰, 怎能有所進步? 我相信老師要我們從一次次 lab 中一點一滴的學習新知識, 而我知道我無形之中已經吸收了許多寶貴的學問了。

笑話時間:

丈夫和妻子走到購物廣場的許願池前。妻子拋下一個錢幣並許下一個願。

丈夫隨即也從口袋掏出一個錢幣拋下去，妻子問他許了什麼願。

『我的願望是』他莞爾地說，
『我付得起你剛才願望得到那件東西的價錢。』