EECS 2070 02 Digital Design Labs 2019 Lab 5
學號：107062107　姓名：王領崧

1.前言

此次 lab 為實作一個簡易的販賣機，設置兩種飲料與兩種投錢機制(5 塊與 10 塊)，並在購買飲料後可以自動退錢。Bonus 部分則是當使用者一段時間都沒有操作 FPGA 板的話，則自動進入退錢模式，退還所有錢並將數字歸零。

2.實作過程

底下的實作過程將主要利用 4 個 state(Initial、Deposit、Buy、Change)來做描述。

`Initial state

這次使用了 4 個 register 來記錄，分別是所投入的零錢(balance、next_balance)，顯示飲料的價錢(price、next_price)、主架構 state(state、next_state)，以及購買的狀態(buy_state、next_buystate)。Initial state 主要是將所有的數值做初始化，將價錢與零錢歸零，並將 state 轉為下一個 Deposit state，開始進行各種判斷。

```
`Initial: begin
    next_state = `Deposit;
    next_buy_state = `Idle;
    next_price = 8'd0;
    next_balance = 8'd0;
end
```

`Deposit state

首先，先將 4 個 reg 給予初始值，兩個 LED 燈則是判斷投入金額是否足夠購買該種飲料，如果可以就維持在亮的狀態，反之則不亮。

接下來，先判斷兩個投入金額按鈕，如果已投入金額(balance)尚未達到上限，則依照按鈕不同，增加相對應的金額，反之如果已達上限，則不進行更改。

然後再判斷兩個顯示飲料金額的按鈕，依據按鈕來顯示不同的價錢，並且進一步判斷如果顯示金額(price)已經為該種飲料的價錢，代表此次為第二次按壓按鈕，因此將 next_state 轉變為`Buy state，next_buy_state 設定為購買該種飲料。

最後判斷取消的按鈕，如果按下取消，next_state 直接變成`Change state，顯示飲料金額為 0，進行退錢的動作。

```
`Deposit: begin
    next_state = `Deposit;
    next_buy_state = `Idle;
    next_balance = balance;
    next_price = price;
    enough_A = (balance >= 20) ? 1'b1 : 1'b0;
    enough_B = (balance >= 25) ? 1'b1 : 1'b0;
    if (money_10_pulse == 1) begin
        next_balance = (balance >= 8'd90) ? balance : balance + 8'd10;
    end
    if (money_5_pulse == 1) begin
        next_balance = (balance >= 8'd95) ? balance : balance + 8'd5;
    end
```

```
if (drink_A_pulse == 1) begin
    next_price = 8'd20;
    if (price == 8'd20 && balance >= 8'd20) begin
        next_buy_state = `BuyDrinkA;
        next_state = `Buy;
    end
end
if (drink_B_pulse == 1) begin
    next_price = 8'd25;
    if (price == 8'd25 && balance >= 8'd25) begin
        next_buy_state = `BuyDrinkB;
        next_state = `Buy;
    end
end
if (cancel_pulse == 1 || num > 15'd9000) begin
    next_price = 8'd0;
    next_state = `Change;
    next_buy_state = `Idle;
end
```

`Buy state

Buy state 只維持一個 cycle，作用為從投入金額扣除飲料錢，因此 next_state 直接轉變為 change state，飲料價錢(next_price)歸零，購買狀態(next_buy_state)重置為`Idle，最後依據 buy_state，來扣除該種飲料的價錢。

```
`Buy: begin
    next_state = `Change;
    next_price = 8'd0;
    next_buy_state = `Idle;
    if (buy_state == `BuyDrinkA) begin
        next_balance = balance - 8'd20;
    end
    else begin
        next_balance = balance - 8'd25;
    end
end
```

`Change state

Change state 為退錢的環節，因此飲料價錢(next_price)維持在零的狀態，購買狀態(next_buy_state)也一樣為`Idle，接著開始判斷剩餘金額，依序由 10 塊、5 塊做退款的動作，LED 依照扣款金額來決定亮燈的數量，next_state 維持在扣款 state，而金額(next_balance)隨著扣款而減少，如果判斷剩餘金額為 0 時，表示退錢完成，next_state 則改變為最初的`Initial state，重新整個過程。

```
`Change: begin
    next_price = 8'd0;
    next_buy_state = `Idle;
    if (balance >= 10) begin
        next_balance = balance - 8'd10;
        drop_money = 10'b1111111111;
        next_state = `Change;
    end
    else if (balance >= 5) begin
        next_balance = balance - 8'd5;
        drop_money = 10'b1111100000;
        next_state = `Change;
    end
    else begin
        next_balance = balance;
        next_state = `Initial;
    end
end
```

Assign 4 個數字 part

FPGA 板子上的 4 個 digit，如果目前不是在`Buy state，則依照投入金額與顯示飲料價錢分別顯示，如果為`Buy state，則根據飲料種類來輸出品名。

```
assign p1 = (state == `Buy) ? ((buy_state == `BuyDrinkA) ? 8'd13 : 8'd14) : price / 8'd10;
assign p2 = (state == `Buy) ? ((buy_state == `BuyDrinkA) ? 8'd15 : 8'd12) : price % 8'd10;
assign b1 = (state == `Buy) ? ((buy_state == `BuyDrinkA) ? 8'd11 : 8'd10) : balance / 8'd10;
assign b2 = (state == `Buy) ? ((buy_state == `BuyDrinkA) ? 8'd10 : 8'd10) : balance % 8'd10;
```

Bonus 部分

因為 5 秒鐘大約是 2^29，因此我同步於 state transition 的 clk，利用一個 2^16 trigger 的 always block，製作一個大約加至 2^13(2^29/2^16)的 counter，當 state 為`Deposit，而且沒有按鈕被按下的話，代表為閒置的狀態，因此進行+1 的動作，反之如果有按鈕被按下，亦或者是 state 非為`Deposit，則將 num 設為 0。接著在主要 always block 中判斷，如果 num 加到 9000(大約 2^13)，代表閒置時間大約為 5 秒鐘，進行與取消按鈕相同的指令。

```
always @(posedge clk_div16) begin
    case(state)
        `Deposit : begin
            if(drink_A_pulse == 1 || drink_B_pulse == 1
            || money_10_pulse == 1 || money_5_pulse == 1)
                num = 15'd0;
            else
                num = num + 15'd1;
        end
        default:
            num = 15'd0;
    endcase
end
```

```
if (cancel_pulse == 1 || num > 15'd9000) begin
    next_price = 8'd0;
    next_state = `Change;
    next_buy_state = `Idle;
end
```

3.學習到的東西與困難

此次 lab 感謝助教提供完整且詳細的 spec，按照指示一步一步的做，基本上蠻順利地，沒遇到什麼 bug 就完成了。這次 state 轉換的 always block 中，採用先將所有 next 值給予初始值，接著在 case 和 if 的判斷式中，再去更改需要變化的 wire 與 reg。我覺得這種寫法可以減少重複的程式碼，縮減多餘的 else，也不會遺漏給值造成 latch 的發生，對於寫程式非常有幫助。也希望自己藉由每次 lab 的練習，程式能夠越寫越好。

4.想對老師和助教說的話

無