EECS 2070 02 Digital Design Labs 2019 Lab 5
學號：104021215　姓名：熊磊

1. 實作過程

- 先利用 clock_divider 做出需要的三種頻率，然後將 input 中的 cancel, money_10, money_5, drink_A, drink_B 做 debounce 及 one pulse 的處理。

```
52      clock_divider #(.n(26)) c26(.clk(clk), .clk_div(clk_display));
53      clock_divider #(.n(16)) c16(.clk(clk), .clk_div(clk_button));
54      clock_divider #(.n(13)) c13(.clk(clk), .clk_div(clk_d));
```

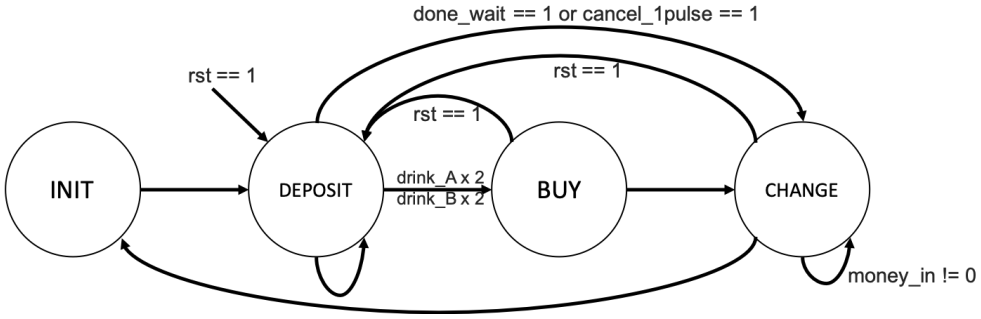
- 設一個 clk_used 的 wire，當 state 為 DEPOSIT 時，就跟著 clk_button，否則就跟著 clk_display。

```
71      assign clk_used = (state == `DEPOSIT) ? clk_button : clk_display;
```

- 把 posedge clk_used 及 posedge rst 當作一個 always block 被觸發的條件，在裡面會將①state, ②使用者投入的總金額, ③drink_A 的計數器, ④drink_B 的計數器, ⑤enough_A, ⑥ enough_B, ⑦飲料名稱 reset, ⑧餘額 reset 或是 update。

```
209     always@(posedge clk_used, posedge rst) begin
210         if(rst==1) begin
211             state <= `DEPOSIT;
212             money_in <= 0;
213             A_c <= 8'd0;
214             B_c <= 8'd0;
215             enough_A <= 0;
216             enough_B <= 0;
217             drop_money <= 10'b0000000000;
218             D0 <= 0;
219             D1 <= 0;
220             D2 <= 0;
221             D3 <= 0;
222         end
223         else begin
224             state <= next_state;
225             money_in <= money_in_next;
226             A_c <= A_c_next;
227             B_c <= B_c_next;
228             enough_A <= enough_A_next;
229             enough_B <= enough_B_next;
230             drop_money <= drop_money_next;
231             D0 <= D0_next;
232             D1 <= D1_next;
233             D2 <= D2_next;
234             D3 <= D3_next;
235         end
236     end
```

- 這次總共用到四個 state，大致的 FSM 切換 trigger 如圖所示



- 用一個@*的 always block 來維護所有我需要用到的值，包含上述 8 個值，以下將針對各個 state 討論。
- 在 INIT 時，在進到這個 state 時為一次新的交易，應將上述 8 個值歸零並可以直接進入 DEPOSIT。
- 在 DEPOSIT 時，如果使用者有選擇飲料，將對應的計數器設為 1，另一個飲料的計數器設為 0，並設定飲料的名稱。如果使用者投錢，增加 money_in。如果 money_in 的金額足以買 drink_A 或 drink_B，將 enough_A, enough_B 亮起。若使用者按了 cancel，就將下一個 state 設為 CHANGE。在這裡說明為什麼使用飲料的計數器，如此可以幫助我判斷使用者是否按兩次飲料的 button。我將進入 BUY 的條件設為判斷投入的錢是否大於 drink 的金額：A 數目*20 + B 數目*25，如果使用者按了其中一個，另一個就會是 0，那我要比較的金額就得到了，不用判斷他過去按的是 A 還是 B。
- 在 BUY 時，將對應的飲料釋出後，飲料計數器歸零，算出要還的錢，顯示飲料的名稱，然後將下一個 state 設為 CHANGE。
- 在 CHANGE 時，若有錢要還，則判斷要還的錢有沒有大於 10 元，若有，則將要還的錢減 10 元，以及將顯示的金額及 Led 燈更新，要還的錢為 5 元時，做類似的事。每次還錢都會判斷還有多少錢要還，如果已經還完，就回到 INIT，否則留在 CHANGE。
- 還有一個 default 的情形如下，state 設為 INIT，其他值設為 0。
- Bouns 的做法如下: 我每次在 DEPOSIT 有動作後，都將等待時間歸零，並開始計算 5 秒。做法是做一個 counter，用 2^{26} 來除頻，並用 en 來作為計時器的開始按鈕。而 en 在 DEPOSIT 的 state 下，只要有任何 button 被按就會在該 clock 中設成 0，而此時計時器會歸零。若沒有下個動作，下個 clock，en 又會被設成 1，並開始 counting。

```
319  module counter(clk, en, done);
320      parameter n = 26;
321      input clk, en;
322      output done;
323
324      reg [n:0] cnt;
325      wire [n:0] cnt_next;
326
327      always@(posedge clk) begin
328          if(en == 1) cnt <= cnt_next;
329          else cnt <= 0;
330      end
331
332      assign cnt_next = cnt + 1'b1;
333      assign done = cnt[n];
334  endmodule
```

2. 學到的東西與遇到的困難

   這次作業讓我對 finite state machine 更熟悉了一些，要設計好什麼時候要改變 state 以及每個值在哪個時間點會改變，否則很難做出題目要的樣子，過程中有發生錯誤也很難知道錯誤在哪裡。本來沒有妥善規劃每一個值造成發生錯誤了，我也看不出來哪裡有錯，後來規畫好之後，直接重寫一遍，才比較容易找到問題在哪，也才順利寫出這次作業。

3. 想對老師或助教說的話

   謝謝助教的幫忙☺