首先是設計的測資，如圖:

```
1   1 0 0 0 0
2   3 0 0 3 0
3   1 0 0 1 0
4   2 0 1 0 1
```

在這個短短的 pattern 中，我製造了投錢足夠，但找錢時機器內零錢不夠的 case(每一行都是)，然後讓他執行成果，可以看出第一個測資(第二個 pattern)進去後，我們預期他要找 8 塊錢，由 1 個 5 員跟 3 個 1 元組成，然而一開始的機器，1 塊錢的數量只有 2 個，此時就會出現找錢錯誤的情形，如圖:

```
==========================================
= cycle 9
==========================================
serviceTypeOut= 2
itemTypeOut= 3
coinOutNTD_1= 2
coinOutNTD_5= 1
coinOutNTD_10= 0
coinOutNTD_50= 0
p= 0
==========================================
= cycle 10
==========================================
serviceTypeOut= 0
itemTypeOut= 0
coinOutNTD_1= 0
coinOutNTD_5= 0
coinOutNTD_10= 0
coinOutNTD_50= 0
p= 0
==========================================
= cycle 11
==========================================
serviceTypeOut= 1
itemTypeOut= 0
coinOutNTD_1= 0
coinOutNTD_5= 0
coinOutNTD_10= 0
coinOutNTD_50= 0
p= 1
==========================================
```

前幾個 cycle 是計算每個額度零錢找的數量，這邊就不特別截圖放上來了，在最後的第 9 cycle，他成功計算出了該找錢的數量(5+1+1+1)，然而販賣機裡面沒這麼多 1 塊錢，所以 p 在第 11 cycle 輸出了 1，因為輸出只能最多 7 塊錢而非 8 塊錢。

接著是 debug 環節，對於這樣的錯誤，我的猜想是:他在計算完零錢數目後，沒有設計成零錢不足就強制退款。此時我回頭檢查關於 vending.v 這部分的 code，如圖:

```
222          `NTD_1  : begin
223            if (serviceValue >= `VALUE_NTD_1) begin
224              if (countNTD_1 == 3'd0) begin
225                serviceValue_w    = inputValue;
226                itemTypeOut_w     = `ITEM_NONE;
227                serviceCoinType_w = `NTD_50;
228                countNTD_50_w     = countNTD_50 + coinOutNTD_50;
229                countNTD_10_w     = countNTD_10 + coinOutNTD_10;
230                countNTD_5_w      = countNTD_5 + coinOutNTD_5;
231                countNTD_1_w      = countNTD_1 + coinOutNTD_1;
232                coinOutNTD_50_w   = 3'd0;
233                coinOutNTD_10_w   = 3'd0;
234                coinOutNTD_5_w    = 3'd0;
235                coinOutNTD_1_w    = 3'd0;
236                serviceTypeOut_w  = `SERVICE_OFF;
237              end else begin
```

這邊他在輸出的時候，可以看到 serviceTypeOut_w 被設成 SERVICE_OFF，視為是輸出退幣成果，然而，上方幾行卻設置退幣數量皆為 0，導致了他判斷這是有意義的輸出，4 種硬幣輸出的值為 0。

```
(outExchange != inputValue); // catch the bug
```

這邊是 p 的最後一項，可以看到第一個測資，input 自然不是 0，跟 0 比對就會得到不同的結果，所以 p 此時為 1。對於這樣的問題，我們要做的，是讓他判斷無法退幣時就不要輸出，所以我做出了以下改動，如圖:

```
234                coinOutNTD_5_w    = 3'd0;
235                coinOutNTD_1_w    = 3'd0;
236                // serviceTypeOut_w  = `SERVICE_OFF;
```

我把設置輸出的 SERVICE_OFF 給註解掉，以此確保 p 的 outexchange 在此階段不會視為是 0。改動後重新執行的成果如圖:

```
==========================================
= cycle 16
==========================================
serviceTypeOut= 2
itemTypeOut= 0
coinOutNTD_1= 0
coinOutNTD_5= 0
coinOutNTD_10= 3
coinOutNTD_50= 0
p= 0
==========================================
= cycle 17
==========================================
serviceTypeOut= 0
itemTypeOut= 0
coinOutNTD_1= 0
coinOutNTD_5= 0
coinOutNTD_10= 3
coinOutNTD_50= 0
p= 0
==========================================
= cycle 18
==========================================
serviceTypeOut= 1
itemTypeOut= 0
coinOutNTD_1= 0
coinOutNTD_5= 0
coinOutNTD_10= 0
coinOutNTD_50= 0
p= 0
==========================================
```

修正完之後，p 的值就都保持為 0 了(30=30)，且根據前面的 cycle，可以看出他有成功進入到退回原金錢的部分(input 為 3 個 10 塊)，算是修正了 bug。