生成測資: input1.pattern

```
23 11
XXXXXXXXX0
01001000001
00XXXXXXX1
00XXXXXXX1
00XXXXXXX1
00XXXXXXX1
00XXXXXXX1
00XXXXXXX1
00XXXXXXX1
00XXXXXXX1
00XXXXXXX1
00XXXXXXX1
01001000001
00XXXXXXX1
00XXXXXXX1
00XXXXXXX1
00XXXXXXX1
00XXXXXXX1
00XXXXXXX1
00XXXXXXX1
00XXXXXXX1
00XXXXXXX1
00XXXXXXX1
```

原則上，我首先測試“如果投錢足夠，但找回餘額時，零錢不足”的情況．而猜測也確實正確，在模擬的結果會發生 p=1 的 flag，如同下圖所示．

#模擬結果: sim1.output

```
00000000000000000
01000000000000000
10010000000000000
10010000000000000
10010000000000000
10010000000000000
10010000000000000
10010010000000000
10010100000000000
00010100000000000
01000000000000000
01000000000000000
01000000000000000
10010000000000000
10010000000000000
10010000000000000
10010000000000000
10010000000000000
00000000000000001
01000000000000000
01000000000000000
01000000000000000
01000000000000000
```

p=1 的 flag 會在第 19 行成立．仔細探究 vending.v 這個 verilog 檔案，發現有一處邏輯錯誤．

#vending.v 部分程式碼: line 224 - 236

```
if (countNTD_1 == 3'd0) begin
  serviceValue_w    = inputValue;
  itemTypeOut_w     = `ITEM_NONE;
  serviceCoinType_w = `NTD_50;
  countNTD_50_w     = countNTD_50 + coinOutNTD_50;
  countNTD_10_w     = countNTD_10 + coinOutNTD_10;
  countNTD_5_w      = countNTD_5 + coinOutNTD_5;
  countNTD_1_w      = countNTD_1 + coinOutNTD_1;
  coinOutNTD_50_w   = 3'd0;
  coinOutNTD_10_w   = 3'd0;
  coinOutNTD_5_w    = 3'd0;
  coinOutNTD_1_w    = 3'd0;
  serviceTypeOut_w  = `SERVICE_OFF;
```

由於在截圖中，最後一行(236 行)的”serviceTypeOut_w = `SERVICE_OFF”，導致了 serviceTypeOut 在這個 cycle 的最後被設定成 SERVICE_OFF，並且輸出成退幣的結果，也就是不退幣也不退商品的下場.

#證據

```
// Property Logic
/***** whether the change is right *****/
//assign p = initialized && (serviceTypeOut == `SERVICE_OFF) && (itemTypeOut == `ITEM_NONE) && (outExchange != inputValue);
assign p = initialized && (serviceTypeOut == `SERVICE_OFF) && (itemTypeOut == `ITEM_NONE) && (outExchange == 8'b00000000);
```

為了證明我的推測，我把 p 的邏輯修改成上圖，結果跑 input1.pattern 以後的結果，與 sim1.output 完全相同，紀錄為 try1.output. 換句話說，輸出的金額之所以和投入的金額不同，就是因為機器吐了 0 元出來，導致與投幣金額不一致.

#Debug

我認為，將 236 行的”serviceTypeOut_w = `SERVICE_OFF”給註解掉，程式就會自動再執行第二輪的 switch，並且從 50 元額度開始退錢，進行購買失敗的退款程序. 因此，應該就會輸出正確的結果. 修改後的模擬結果如下: (sim1-fixed.output)

```
00000000000000000
01000000000000000
10010000000000000
10010000000000000
10010000000000000
10010000000000000
10010000000000000
10010010000000000
10010100000000000
00010100000000000
01000000000000000
01000000000000000
01000000000000000
10010000000000000
10010000000000000
10010000000000000
10010000000000000
10010000000000000
10000000000000000
10000000000000000
10000000000010000
10000000000010000
10000000000010000
```

Flag p=1 確實不再發生了. 問題解決!