**Hardware Design and Lab: Lab6**

**Team 28**

**111060013 EECS 26' 劉祐廷**

**111060002 EECS 26' 李侑霖**

**Catalog**

**1. FPGA Question:**

**Slot Machine…………………………………….…………….……P3**

**2. FPGA Question:**

**Chip2chip…………………………………………..…...………….P7**

**3. FPGA Question**

**Car…………………………………………………………….P8**

1. **FPGA Question: Slot Machine**
2. **Block Diagram**

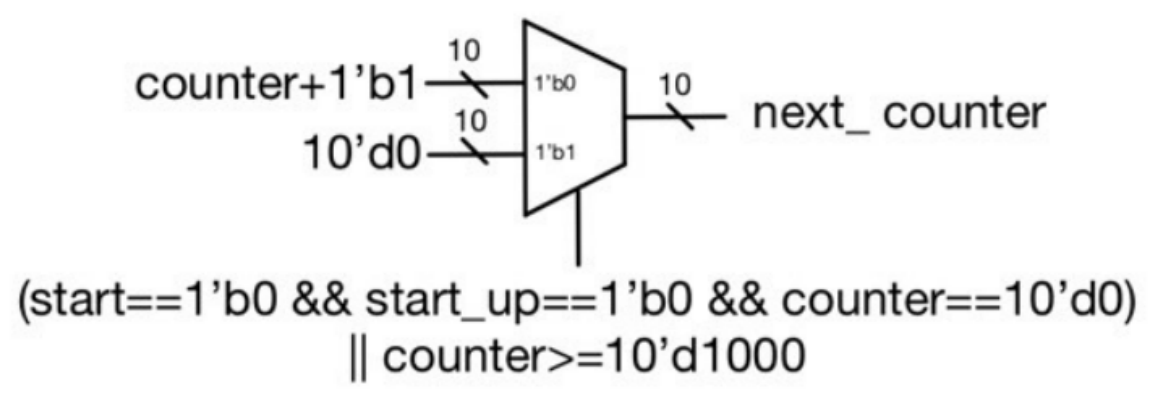
****

Figure 1.1

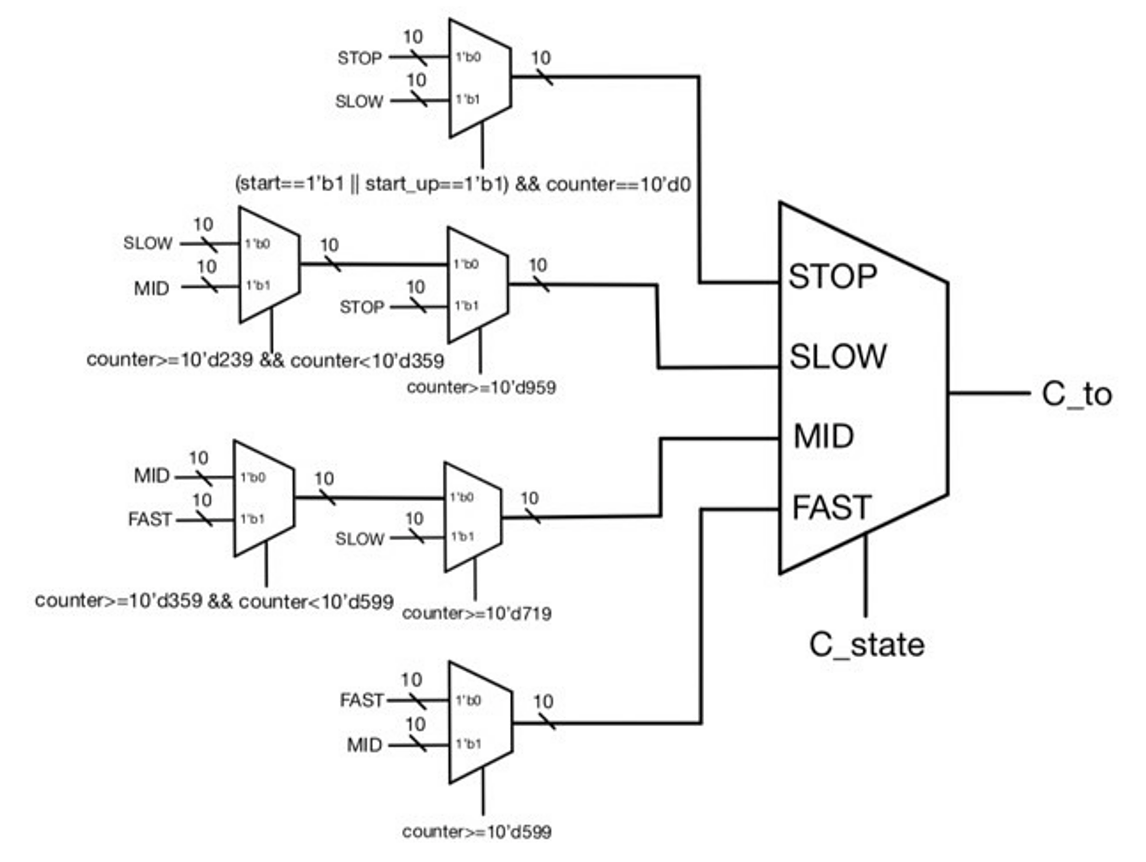
****

Figure 1.2

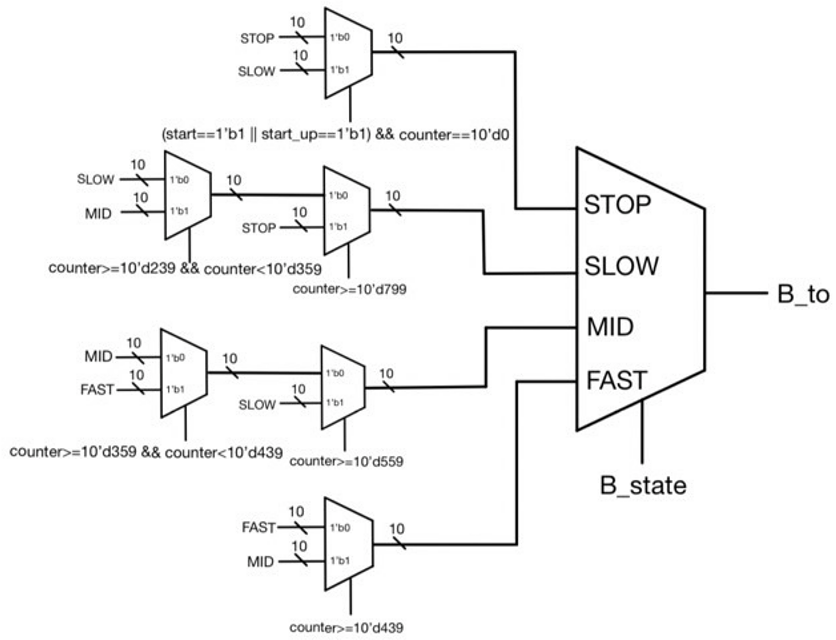
****

Figure 1.3

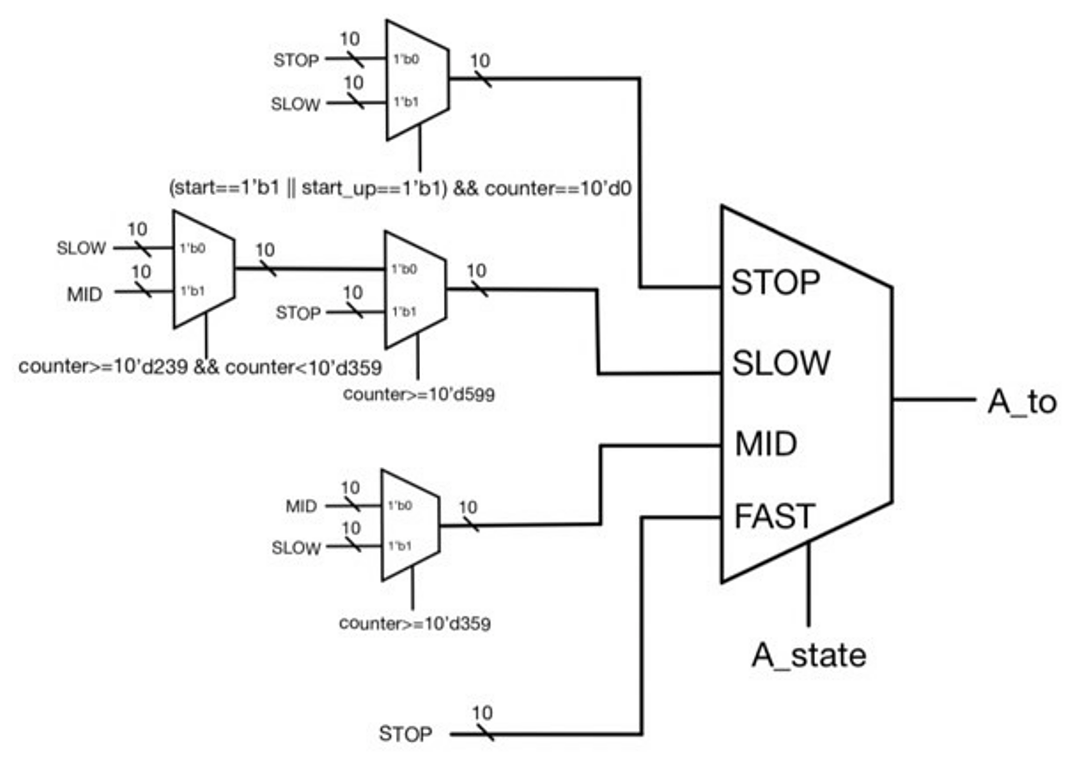
****

Figure 1.4

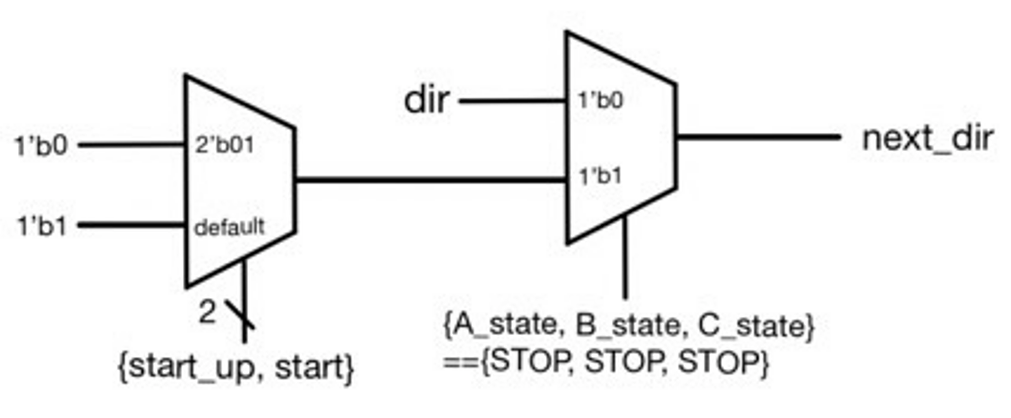
****

Figure 1.5

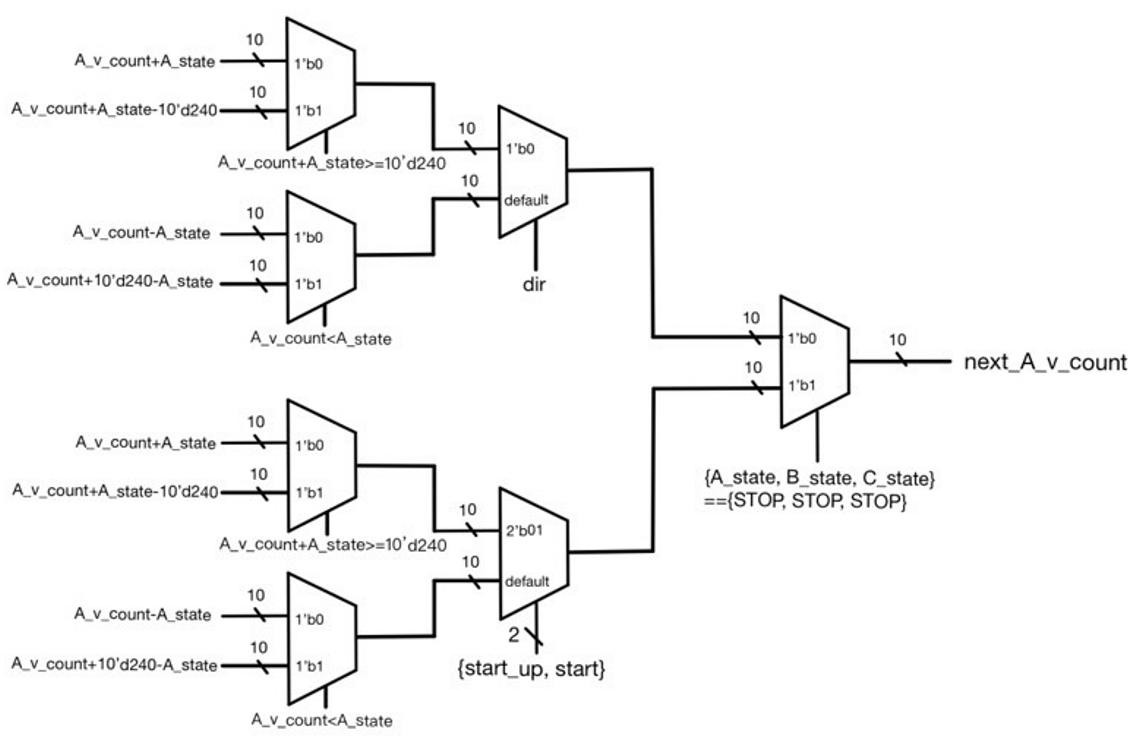
****

Figure 1.6

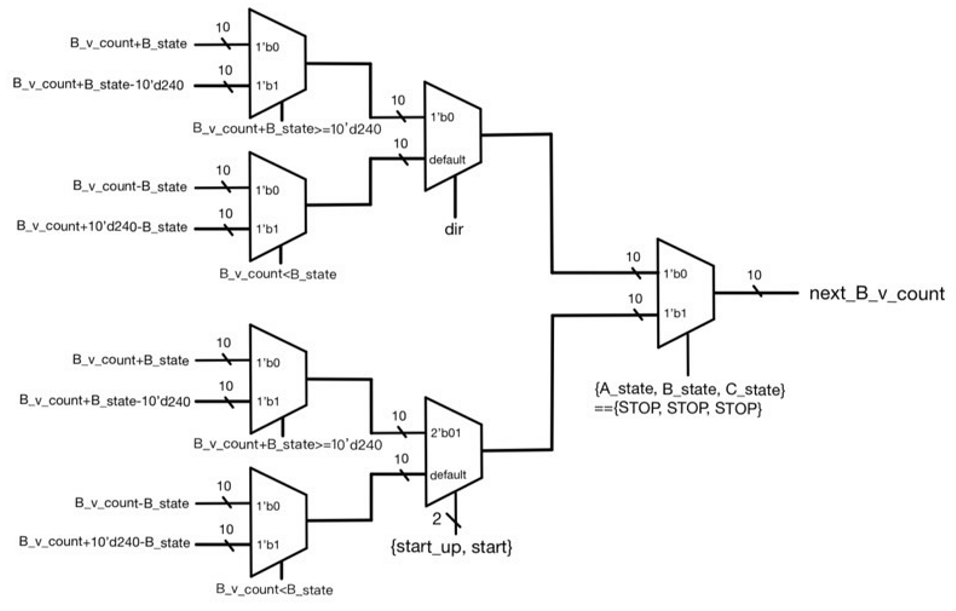
****

Figure 1.7

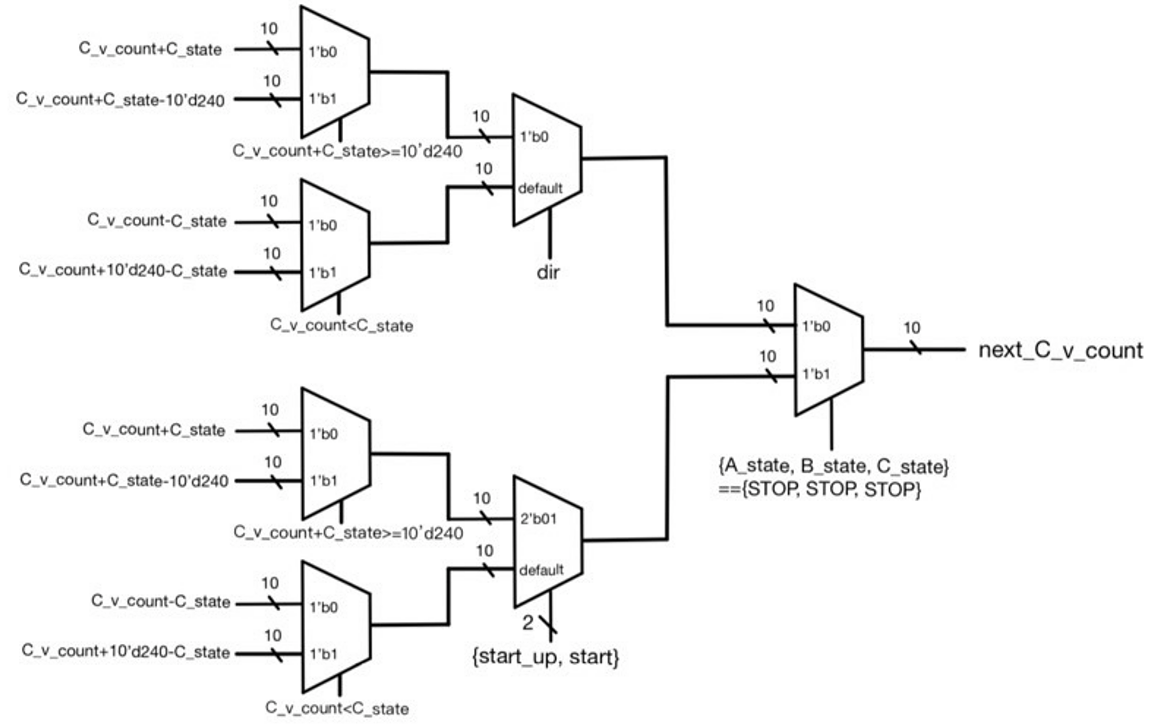
****

Figure 1.8

1. **Explanation**

在Slot Machine這題一開始的時候，我們注意到了在每次向下運轉後，都需要再按一次reset才能重新向下旋轉，經過trace code之後，我們發現 (Figure 1.1) counter 原本的運作邏輯有問題，因此我們將它改成當沒有任何start button被按下且counter為0時 (代表slot machine目前在停止狀態)，或是當counter數到1000時 (代表slot machine剛運轉完)，counter會被重新設為0，也就是原本等待button被按下時的狀態，以此達成不需要reset就可以連續使用向上旋轉或向下旋轉的功能。

為了實現向上旋轉的功能，我們新增了一個叫做start\_up的input來接受向上旋轉button的輸入，原本向下旋轉button的input就繼續沿用原本start這個名稱。因為加入了start\_up，所以對於A、B、C開始數數的條件我們改成了只要start或是start\_up為1時，並且counter是0時 (slot machine目前在停止狀態)，counter才會開始數數 (Figure 1.2 ~ Figure 1.4)。

為了要避免在旋轉狀態時，按下反方向的button導致slot machine出錯，我們多設了一個register叫做dir，用來儲存目前的狀態，如Figure 1.5所示，在slot machine為停止狀態時，才會依照按下的button去改變旋轉的方向，以此避免前述提到的問題。

Figure 1.6 ~ Figure 1.8展示了我們如何實現向上或向下旋轉，我們將原本的遞增改成了遞減，並稍微修改了A\_v\_count、B\_v\_count、C\_v\_count超出範圍的條件，並透過dir來保持運作時的方向，已完成向上旋轉的功能。

1. **FPGA Question: Chip2chip**
2. **Finite State Diagram**

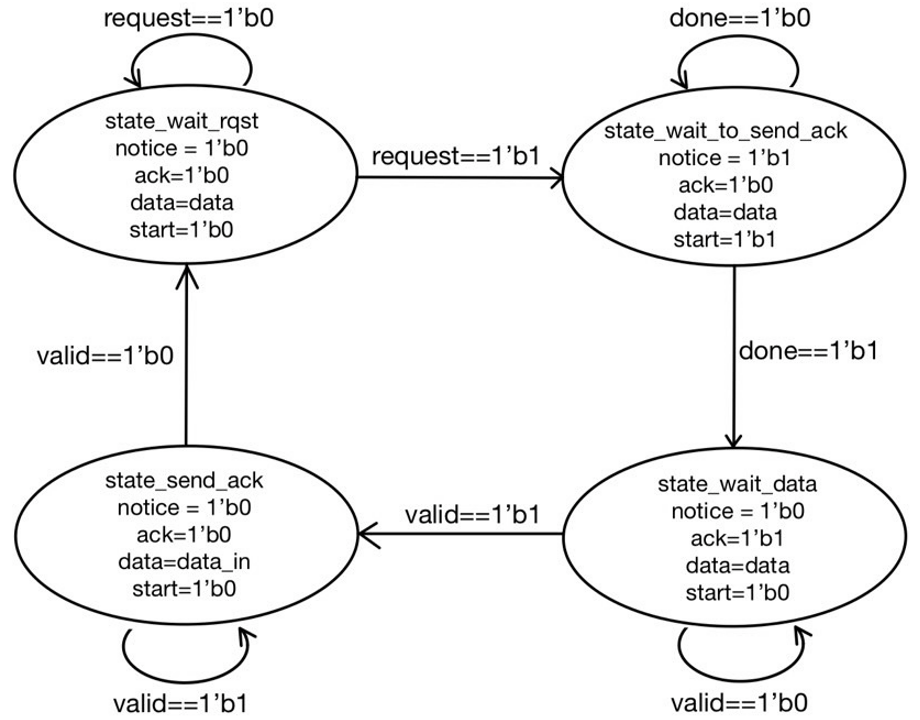
****

Figure 2.1

1. **Explanation**

如**Figure 2.1**所示我將使用狀態分成兩種，第一種是**INSERT\_MONEY**，一開始reset會進到這個狀態，等待使用者投錢，並用七段顯示器顯示使用者投入了多少錢，一旦使用者選擇了飲料而且投進去的錢足夠多，或是使用者按了**cancel**按鈕，則會進入第二種狀態**RETURN\_MONEY**，在這個狀態之下，販賣機對於投錢、選擇飲料和**cancel**按鈕不會做出反應，七段顯示器會以每秒減少5元的速度改變數值，直到**sumMoney**變為0，接者會回到**INSERT\_MONEY**等待後續的操作。整體的電路圖如**Figure 2.4**所示，其中**OnePulse**與**Debounce**是沿用上一個Lab使用的module (**Figure 2.2, Figure 2.3**)。Money這個module的電路圖為**Figure 2.5 ~ Figure 2.7**，大致分為三塊電路：**state, sumMoney, count**。其中**state**用來記錄狀態變化，**sumMoney**用來記錄販賣機內有多少錢，**count**則是計數器，用來devide clock。**SevenSegment**這個module的電路圖為**Figure 2.8 ~ Figure 2.14**，**mux2, mux1, mux0**分別代表百位數、十位數與個位數。

1. **FPGA Question: Car**
2. **Finite State Diagram**

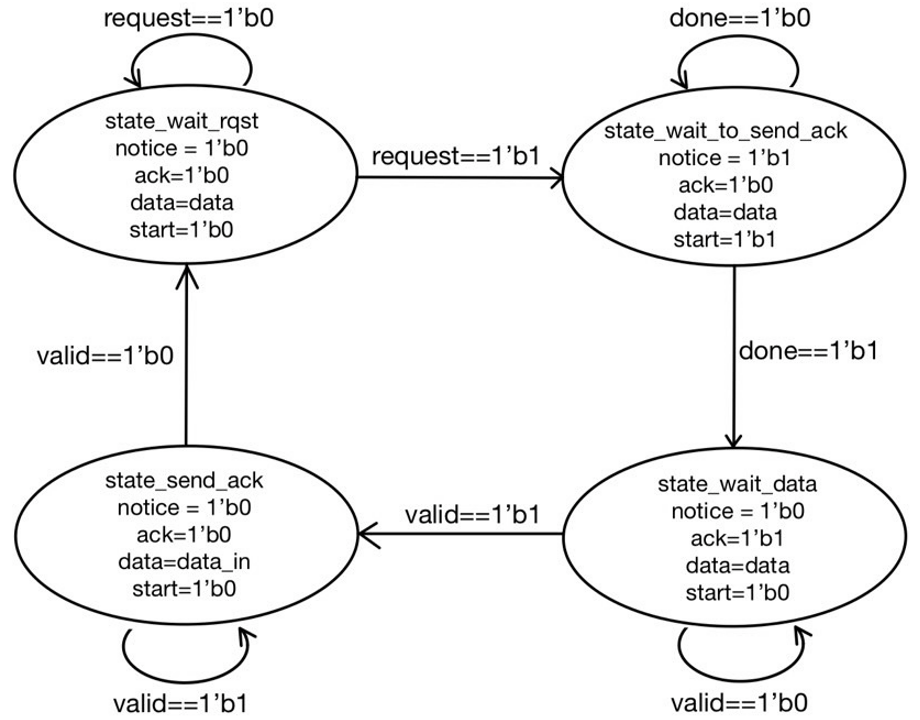
****

Figure 2.1

1. **Explanation**

如**Figure 2.1**所示我將使用狀態分成兩種，第一種是**INSERT\_MONEY**，一開始reset會進到這個狀態，等待使用者投錢，並用七段顯示器顯示使用者投入了多少錢，一旦使用者選擇了飲料而且投進去的錢足夠多，或是使用者按了**cancel**按鈕，則會進入第二種狀態**RETURN\_MONEY**，在這個狀態之下，販賣機對於投錢、選擇飲料和**cancel**按鈕不會做出反應，七段顯示器會以每秒減少5元的速度改變數值，直到**sumMoney**變為0，接者會回到**INSERT\_MONEY**等待後續的操作。整體的電路圖如**Figure 2.4**所示，其中**OnePulse**與**Debounce**是沿用上一個Lab使用的module (**Figure 2.2, Figure 2.3**)。Money這個module的電路圖為**Figure 2.5 ~ Figure 2.7**，大致分為三塊電路：**state, sumMoney, count**。其中**state**用來記錄狀態變化，**sumMoney**用來記錄販賣機內有多少錢，**count**則是計數器，用來devide clock。**SevenSegment**這個module的電路圖為**Figure 2.8 ~ Figure 2.14**，**mux2, mux1, mux0**分別代表百位數、十位數與個位數。

1. **What I Have Learned**

以前玩遊戲鍵盤會分成短按與長壓兩種操作，在這次lab我了解到原來這兩種狀態是使用按下與放開組合成的，或許在期末專題可以用這個特性做出比較多樣的操作組合。另外透過這次的題目，我也更了解如何去設計finite state machine，並且也更熟悉divided clock的使用方法與時機。