1. 這次作業特別有感受到自己在修改面積時更加得心應手了，我原本寫完的面積是大約 25000，後來修改完就剩下約 15000。由上次的經驗，我這次將很多能夠重複利用的東西寫在一起。下面舉算找錢的部分為例下面是原本的寫法，上面是最後的寫法。透過這樣我可以將乘法器和減法器共用。20~1 用判斷的是因為合成完發現這樣會比全部都用乘法小。

```
always_comb begin
    case(cnt_nxt)
        5: Subtract = 50 * Coin50_reg;
        4: Subtract = (Coin20_reg == 2) ? 40 : (Coin20_reg == 1) ? 20 : 0;
        3: Subtract = (Coin10_reg) ? 10 : 0;
        2: Subtract = (Coin5_reg) ? 5 : 0;
        1: Subtract = Coin1_reg;
        default: Subtract = 0;
    endcase
end
```

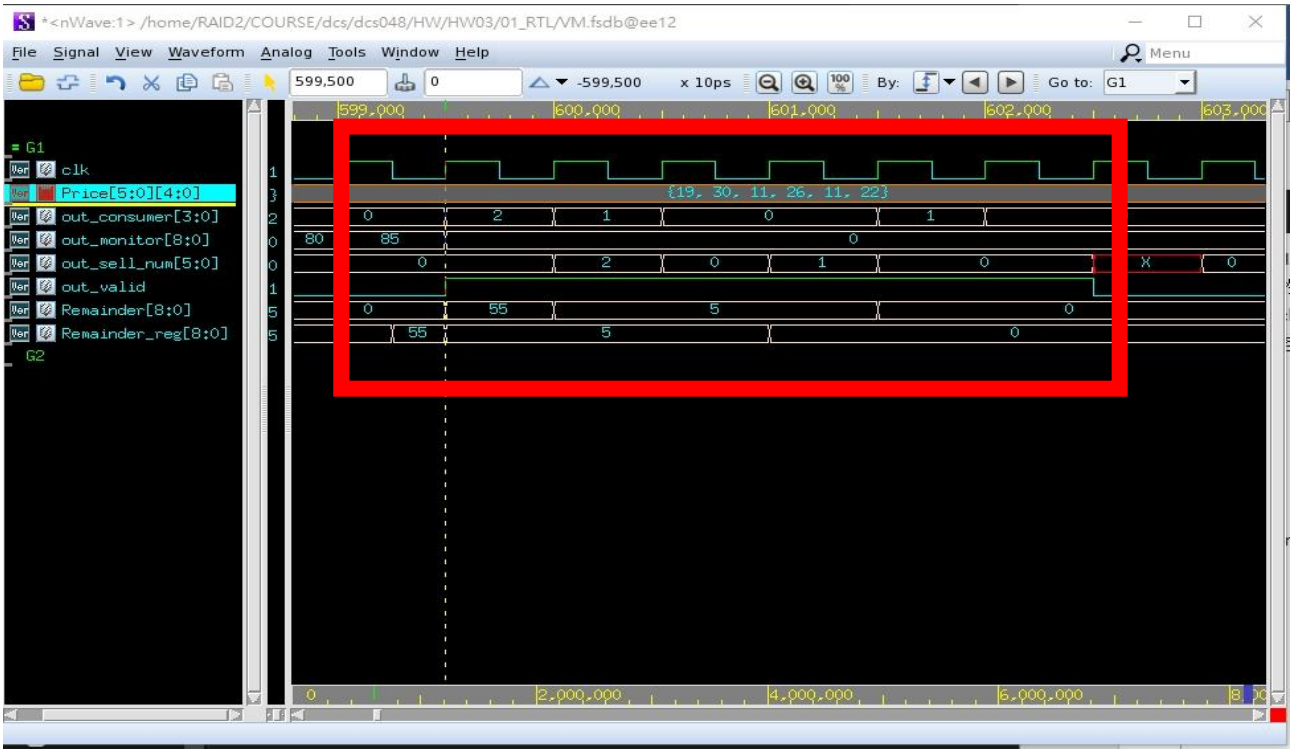
```
always_comb begin
    case(cnt_nxt)
        5: Remainder_reg = out_monitor - 50 * Coin50_reg;
        4: Remainder_reg = out_monitor - 20 * Coin20_reg;
        //...
    endcase
end
```

2. 另外我這次也遇到了 slack 的問題，在社團發問後，助教很有耐心ㄉ幫我們解惑了！就是答案不用急著在一個 cycle 裡面算完，也因此我後來把設計改成每個循環算出一個答案，並且另外開一個 register 存剩下多少錢要找零。因為 out_monitor 的輸出必須在一個 cycle 內決定，所以才要另外開一個存

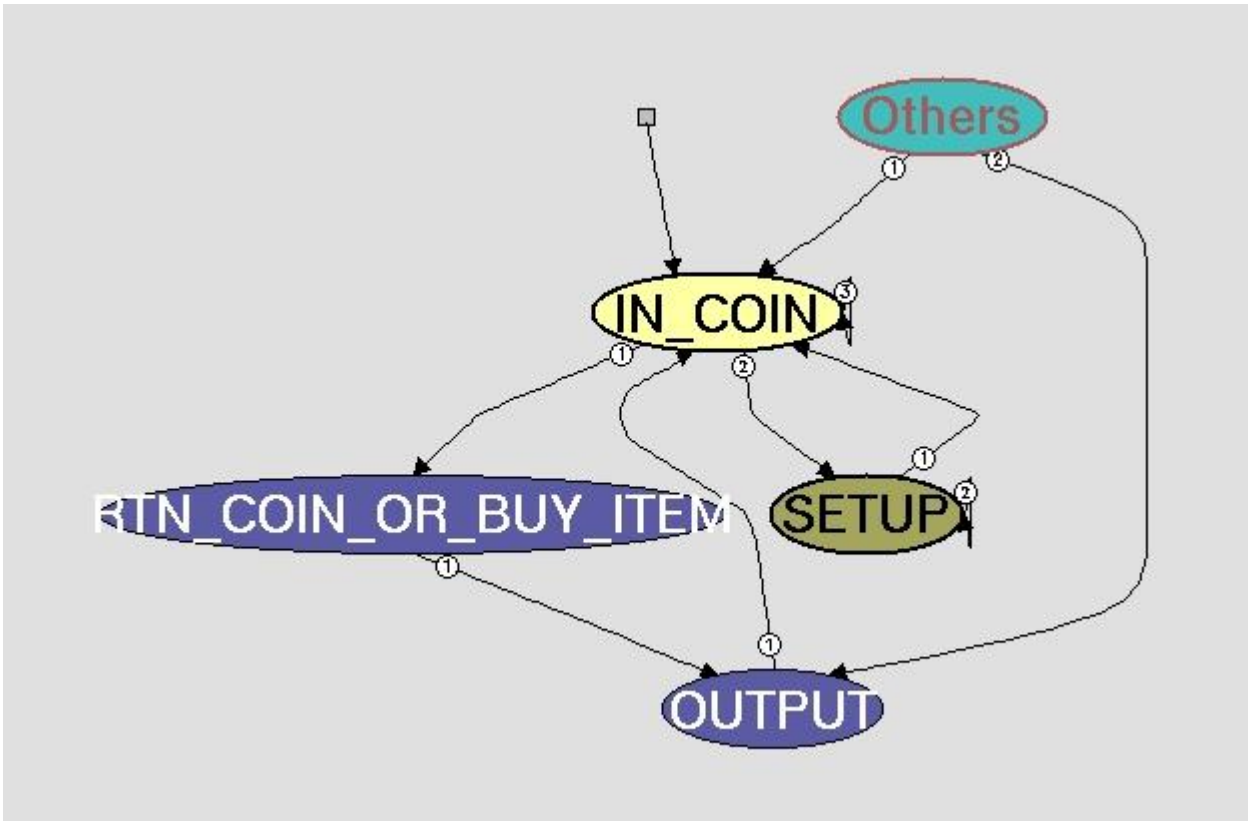
```
// Calculate Remainding Money
// And update Remaindersell
assign Remainder_reg = (state_nxt == RTN_COIN_OR_BUY_ITEM) ? out_monitor - Item : Remainder - Subtract;
```

RTN_COIN_OR_BUY_ITEM 就是在該循環要做交易或是找錢，所以是直接拿 out_monitor 去減掉所買商品的價錢。

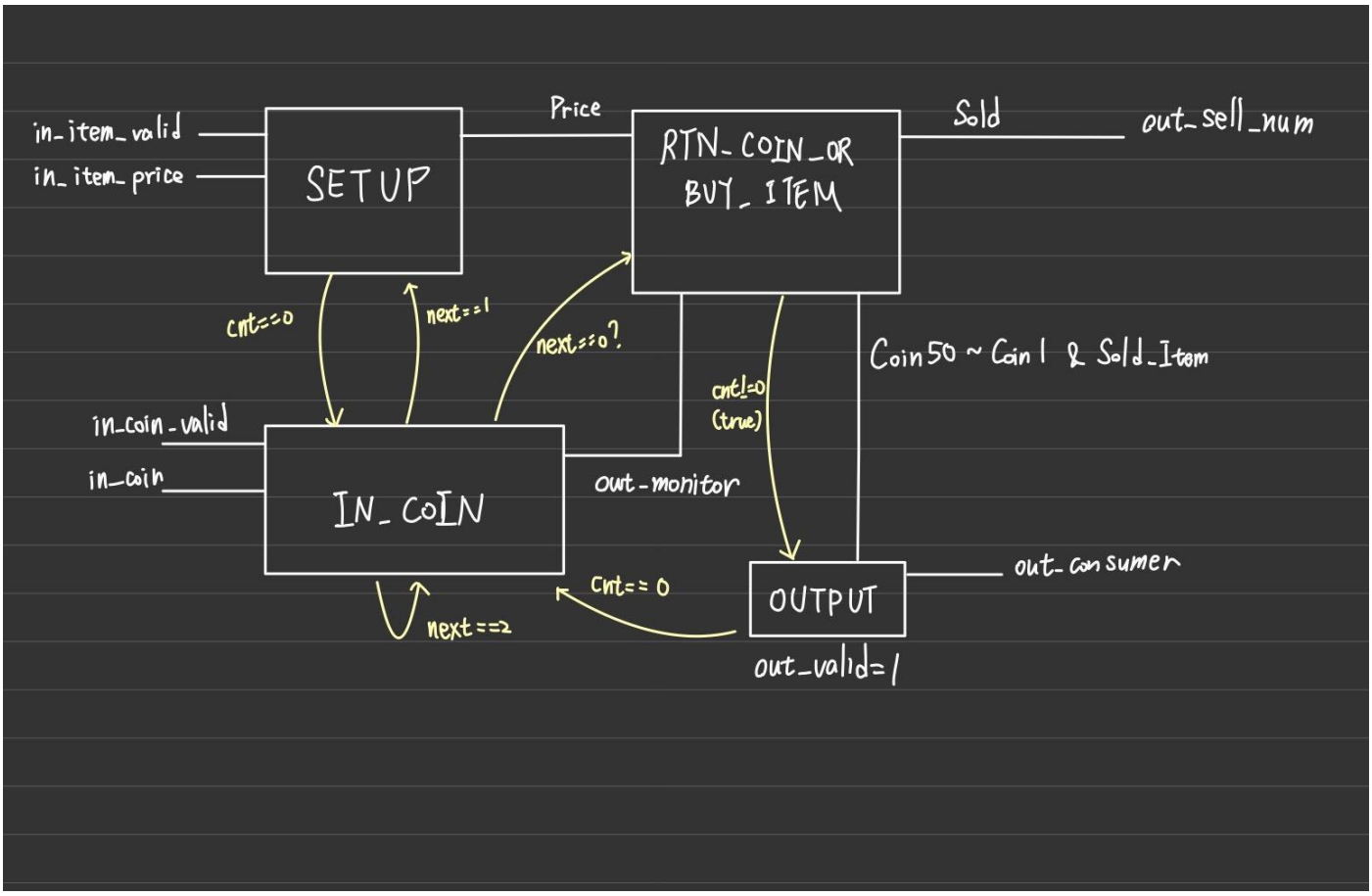
下個 cycle 之後會接著 OUTPUT，所以就直接拿 Remainder(此時已經存到 Remainder_reg 的數字)去減掉各個循環要減的值(計算方法在第一張圖)



state diagram (不清楚 others 為啥在那，我的 state 只有 4 種而已):



block diagram :



附上 Annotation 的截圖，以 state_nxt 和 state，以及所有 input output 為例