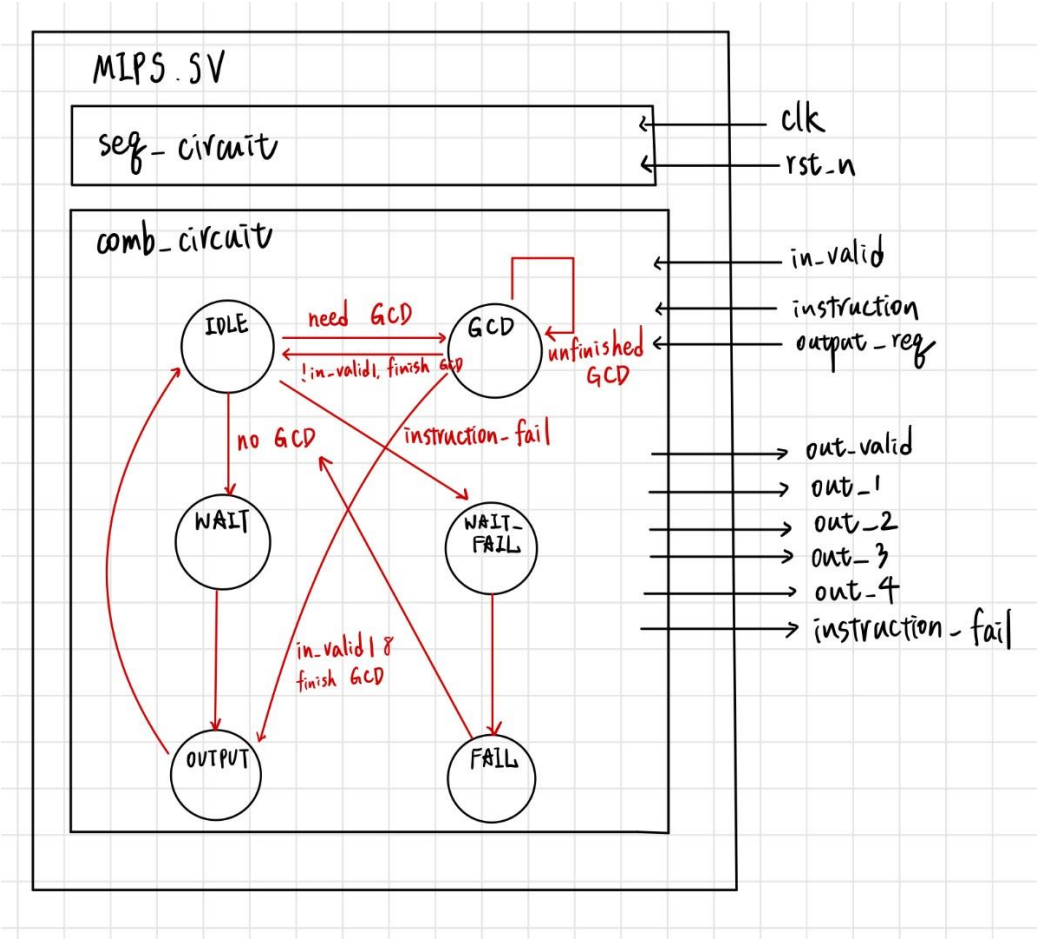
Report_dcs193 HW05 蔡東宏 110511277

1. 程式說明:

此題作業是執行 CPU 的簡單運算，我用了 6 個 state 的 FSM 去處理此問題，一開始 reset 後的 state 為 IDLE，當 in_valid 拉起來的時候，根據 instruction 特定位置的值判定須對哪一個暫存器執行哪一種指令，當判定到 instruction 的值不合理的時候，下一個 state 進到 WAITFAIL；而若合理的時候，且讀到的指令不為 GCD 時，根據 instruction 的值，將特定暫存器的值進行特定的指令並存到特定的暫存器，下一個 state 進到 WAIT；當需要進行 GCD 時，下一個 state 進到 GCD，若第一次進入 GCD 就有答案可以輸出的話，下一個 state 進到 OUTPUTP，若不是的話，GCD 有答案的話就可以直接輸出並回到 IDLE，若沒有答案的話則持續進行 GCD 直到有答案，若 state 為 WAIT 時，下一個 state 為 OUTPUT，根據 output_reg 的值選擇 out_1 到 out_4 要輸出哪一個暫存器，輸出完回到 IDLE； 若 state 為 WAITFAIL 時，下一個 state 為 FAIL，輸出 instruction_fail，輸出完回到 IDLE。

2. 架構圖:



3. 優化過程:

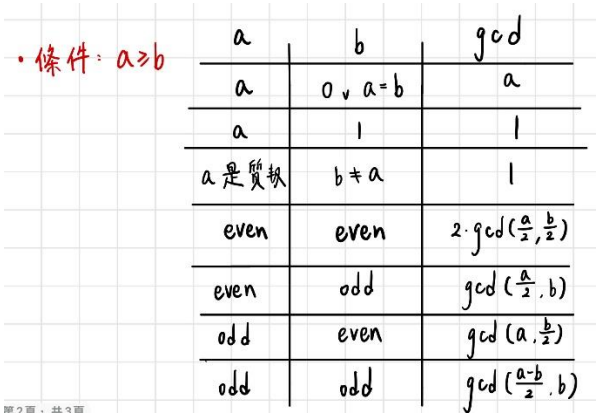
簡化 GCD: 一開始的時候，GCD 的計算我使用的是輾轉相除法，但最後會發現在合成電路的時候需要合成一個 16bit mod 16bit 的電路，不僅合成出來的面積超大，合成的時間還要很久。後來我改了方法，利用下表(GCD 簡化表)，我們根據 a 和 b 的基偶數，慢慢將 a 和 b 減小，當 a 和 b 分別符合前三行的條件及代表我們已經可以知道 gcd 的答案。所以判斷是否完成 GCD 的判斷式為右圖。

```
if(gcdb_reg<=1 || gcdb_reg==gcda_reg || (a&&gcdb_reg!=gcda_reg))begin
```

一開始的時候，我沒有判斷質因數的條件，所以導致算 GCD 的時候，即使大的數遇到質數時，還需要再多幾個 cycle 讓他有答案可以輸出，但後來發現可以將他列成一個表，當判斷到大的數為質數且小的數不等於大的數時，能夠直接得到他們的 GCD，如此一來，雖然會讓面積稍微變大，但 latency 會減少滿多的。

一開始的時候，若遇到兩個都是偶數，每次都只有除 2，若要連續除 2，會發現此作法會讓 latency 變很大，後來我改了方法，並結合兩個數其中有一個是偶數的情況，先判斷最後有幾個 0(shifta 和 shiftb)，即代表他為 2 的(shifta 和 shiftb)次方倍，最後將 a 和 b 分別除 2 的(shifta 和 shiftb)次方倍(我用 shift 來完成除法)，而因為若都是偶數須將 ans*2，所以我要判斷 shifta 和 shiftb 的大小)，並將答案乘 2 的 min(shifta,shiftb)次方倍，透過此方法可以大幅減少 latency 的大小。

```
casez (gcda_reg)
    16'b????????????????10:shifta= 1;
    16'b???????????????100:shifta= 2;
    16'b??????????????1000:shifta= 3;
    16'b?????????????10000:shifta= 4;
    16'b????????????100000:shifta= 5;
    16'b???????????1000000:shifta= 6;
    16'b????????10000000:shifta= 7;
    16'b???????100000000:shifta= 8;
    16'b??????1000000000:shifta=9;
    16'b?????10000000000:shifta=10;
    16'b????100000000000:shifta=11;
    16'b???1000000000000:shifta=12;
    16'b??10000000000000:shifta=13;
    16'b?100000000000000:shifta=14;
    16'b1000000000000000:shifta=15;
    default: shifta=0;
endcase
```



(GCD 簡化表)

4. 結論:

此題作業是執行 CPU 的簡單運算，為了減少面積，我避免使用輾轉相除法去處理 GCD，因為它需要用到 16bit mod 16bit 的電路，如此可以大幅減少電路面積，最後再利用一些方法減少 latency。