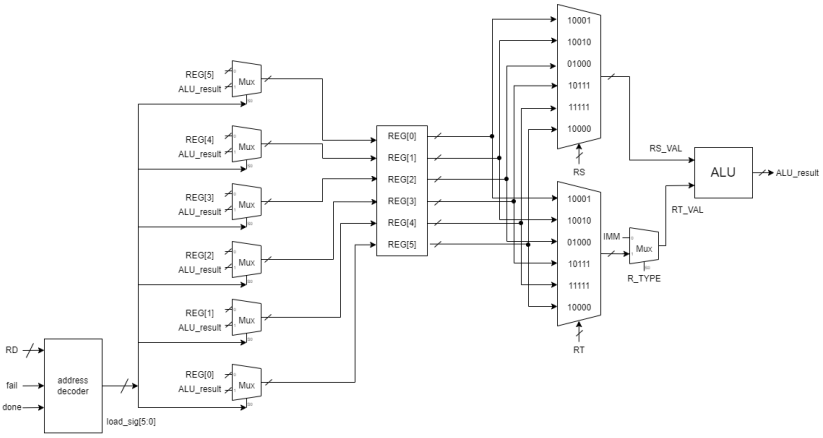
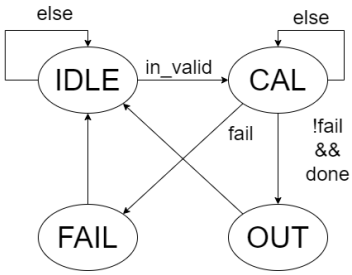
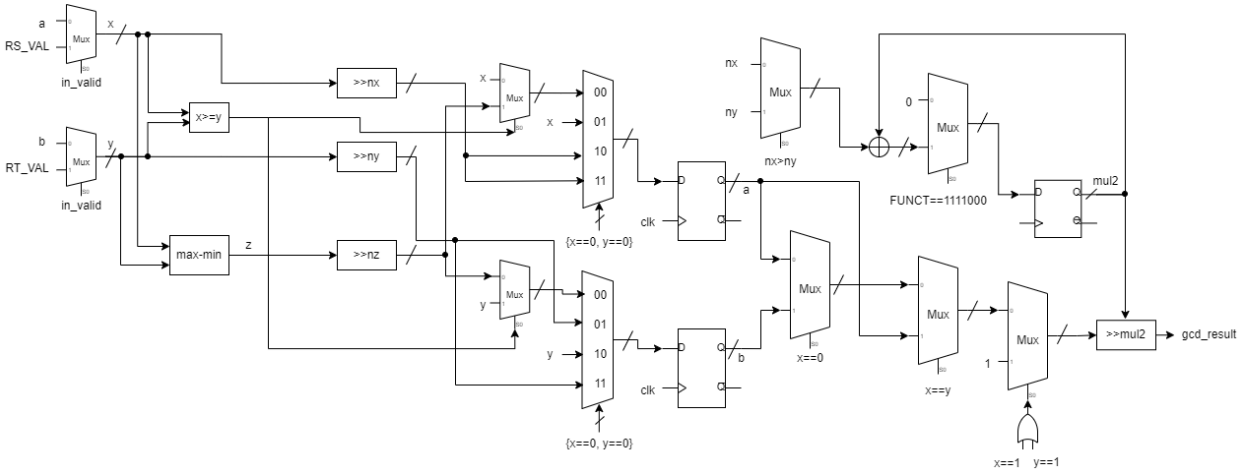
數電HW5



HW5的架構參考RISC-V CPU的架構，首先從指令中讀出RD、RS、RT地址和其他資訊，根據指令內容先從REG讀出對應的RS、RT值，輸入ALU運算完成後，再於狀態由CAL轉換到OUT時將其寫入RD，如此一來在OUT階段便可根據out_reg給的資訊輸出REG中的值。



上圖為 control fsm 的 state diagram，其中fail是各種不合法狀況的結果，done則表示ALU是否完成運算，除了gcd情況時done會等到迭代完成才為1之外，其他運算的done皆為1。



上圖為ALU中利用stein's algorithm 計算GCD的電路，為了降低迭代次數，我找出三種停止迭代條件:

1. x、y兩數其中一個為1→輸出1

2. x、y兩數其中一個為0→輸出另一個數
3. x、y兩數相等→輸出其中一個數

x、y兩數是組合邏輯的輸出，因此不用等待輸入值被存入a、b，就可以先判斷是否停止迭代。如果RS_VAL和RT_VAL輸入時已滿足停止條件，就可以在下個週期輸出。

```
casez(x)
        16'b????????????????10: shift_num_x = 1;
        16'b????????????????100: shift_num_x = 2;
        /* ..... 下略*/
        16'b??10000000000000: shift_num_x = 13;
        16'b?1000000000000000: shift_num_x = 14;
        16'b1000000000000000: shift_num_x = 15;
        default: shift_num_x = 0;
endcase
shifted_x = x >> shift_num_x;
```

此外，我用自製的pattern測試時發現一次只除以2的演算法，在遇到2^n的倍數時會非常耗時。因此我把演算法中所有需要除2的地方，都改成除以2的n次方。實現方式為: 若x的二進位表示中，最小的nx位數字都是0，就把x右移nx位，y和奇數相減也用同樣的方式處理。在10000筆測資的情況下，加了這一步可以將latency下修將近一倍。而面積僅上升約3成。

可以改進的地方:

在開發過程中我一直遇到timing violation的問題，原因都是從讀取RS_VAL到GCD的路徑延遲太長，我本來有想過用FF把RS_VAL存起來，讓GCD有更多時間可用。但後來為了省面積而沒有採用。但後來想想，那32個FF頂多占去3200的面積，若將RS_VAL存下來，GCD說不定還能串連兩極，讓latency少一半。