Lab5: (Infinite impulse response filter)IIR

104062321

劉樂永

一、設計架構

這次的設計和上次的 GCD 類似，以一個 FSM 來控制 datapath。但是這次我只設置兩個 module，control_FSM 和 datapath。

control_FSM 分成兩個 state，idle 和 read。idle 接收到 Start 訊號後，會進入 read，並拉起 load，把 RAddr 設成 0。接著在每一個 cycle，把 RAddr+=1。讀到 data_done 之後再回到 idle 階段，把 load 設成 0。

datapath 會接收 control_FSM 的 state，作為切換 state 的 command。當 control_FSM 的 state 為 read，datapath 就會進入 write，並將 WEN 設為 1。之後在 write state 裡，讓 WAddr 保持和 RAddr 相隔一個 cycle，方便做運算。等到 control_FSM 的 state 回到 idle 之後，就讓 WEN=0，並輸出 Finish。

二、coding 經過

有了 GCD 的設計經驗後，在兩個 module 之間的訊號傳遞、延遲控制等都熟練許多，很快就完成了基本架構。

實作 IIR 公式的時候就遇到了很多問題，主要是在負數處理和位數擴充的部分。最後發現，正確的方法是先把公式內每個需要運算的值都取絕對值之後，再乘以係數的絕對值，最後相加之前再從原先的正負號判斷是否需要再 * -1 還原。看來負數是不能用 shift 的方式模擬小數乘法的。

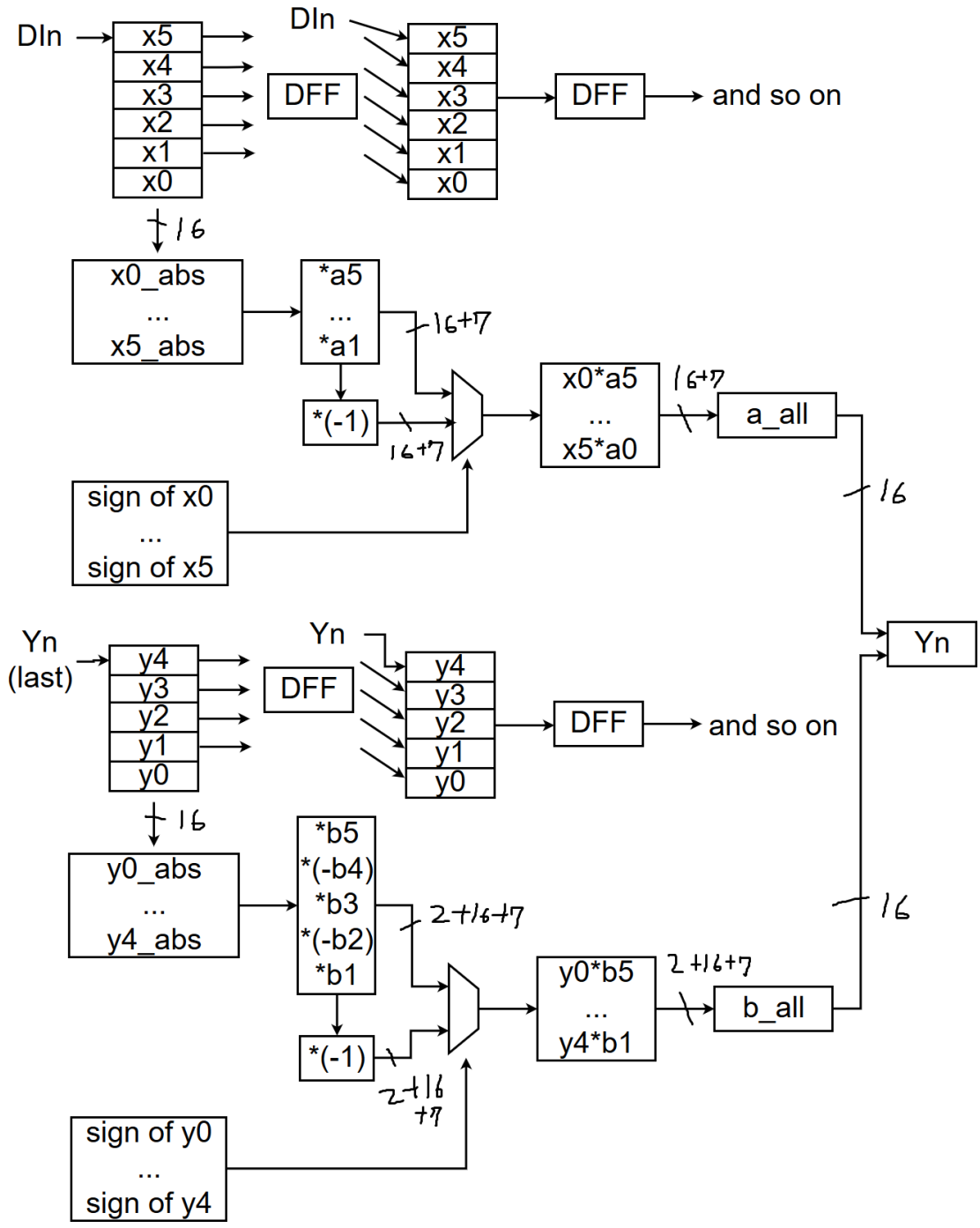
至於位數擴充的部分，我把所有乘法都往最低位擴充 7 bits。如果係數大於 0，就再往最高位擴充 2 bits。這樣濾出的聲音就沒問題了。

另外一開始還有一個粗心的錯誤，是在 DFF 裡出現像這樣的 code：

```
x0<=x1;

x1<=x2;

x2<=x3;

…….
```

但是這在一開始的模擬中居然沒有問題，只是造成後來的 timing violation。

三、datapath 構造



大致的構造如圖中所示。用 DFF 讓資料流動。每一次計算都先把所有值取出絕對值（其實這裡也是用一個 MUX），之後乘上係數，再還原回原本的 sign，最後加總，取出需要的 16 bits。

和實際作品不一樣的是，因為 b5 的值是 0.37959，小於 0，整數部分不需要擴充到 18-bit，因此在 code 中其實是在上半部運算的。

四、心得

寫過 GCD 之後，這次作業的架構相對上簡單很多，也能更熟練的用 nWave 找出訊號轉換的問題。另外在 coding 和 debug 的同時，也能在腦中更清晰的掌握整個電路的結構，因此這次作業需要的時間比之前少了很多，終於有比較上手的感覺了。