一、 主要設計概念

此次設計是 Sparse Matrix 運算，所以會有許多元素為 0，所以可以依照 0 的數目來減少面積。

這次輸入的資料順序與硬體十分媒合。由於先輸入 Input vector 再輸入 Matrix element，所以只要儲存 Input Vector 就好，Matrix element 可以等它輸入後乘完就捨去，所以會減省許多硬體使用。認知到這種輸入資料的方式後，讓我在之後若為 SISO 的矩陣乘法，可以知道如何輸入資料，使硬體能優化。

在本次硬體設計，儲存數值也是一種學問，可以依照儲存資料的特性、個數、取出邏輯多寡來決定使用哪一個硬體設計。但在這次設計中，若想把 delay of out_valid 為 1，則會有 Input external delay，所以若想要不被懲罰 external delay，則必須使用 FF 來吃下 input，使得 Path 變短。

二、特殊方法

1. 儲存硬體方法：

針對需要儲存數值的部分，分別根據資料特性設計了兩種不同的儲存方式。首先，對於 Input Vector，由於每次輸入最多只有 14 個非零元素，因此採用堆疊式的暫存器陣列來儲存這些有效資料，同時記錄對應的行號，相較於直接配置 32 個暫存器，可以大幅減少硬體面積的浪費，能有效提升儲存效率。取值時，透過多級 MUX 邏輯與優先編碼器，根據輸入的 in_col 快速找到對應的資料，雖然硬體邏輯較為複雜，但能兼顧面積與效能。至於 Sum of each row，由於必須完整保存 32 行的計算結果，因此直接配置 32 個暫存器，讓每一行有固定的儲存位置，這樣不僅方便直接定址存取，也能簡化輸出時的檢索邏輯。整體而言，Input Vector 採用動態分配、關聯式查表的方式，節省硬體資源但增加了取值複雜度；而 Row Sum 則以固定分配、直接定址為主，確保資料完整且檢索快速。這兩種儲存架構的設計，分別在 Verilog 程式碼中以 invalue[13:0]、addr[13:0] 及 sumst[31:0] 等暫存器陣列存在。

2. 在 One-hot 使用 Casez：

在處理 One hot selector 時，如果使用原本的 case 用法，僅能生成對等比較(全部 bit 都要比較)，所以使用 Casez，則可以使用?來當作 Don't care，使得只要比較一個 bit 就好。

3. 思考是否必要增添 rst_n falling edge 歸零功能於模組內 DFF：

在模組內，有許多除了 Output 的 DFF，**這些 FF 不一定要添加 rst_n Reset 訊號**。若盲目加上，則會耗費許多空間。

三、改進

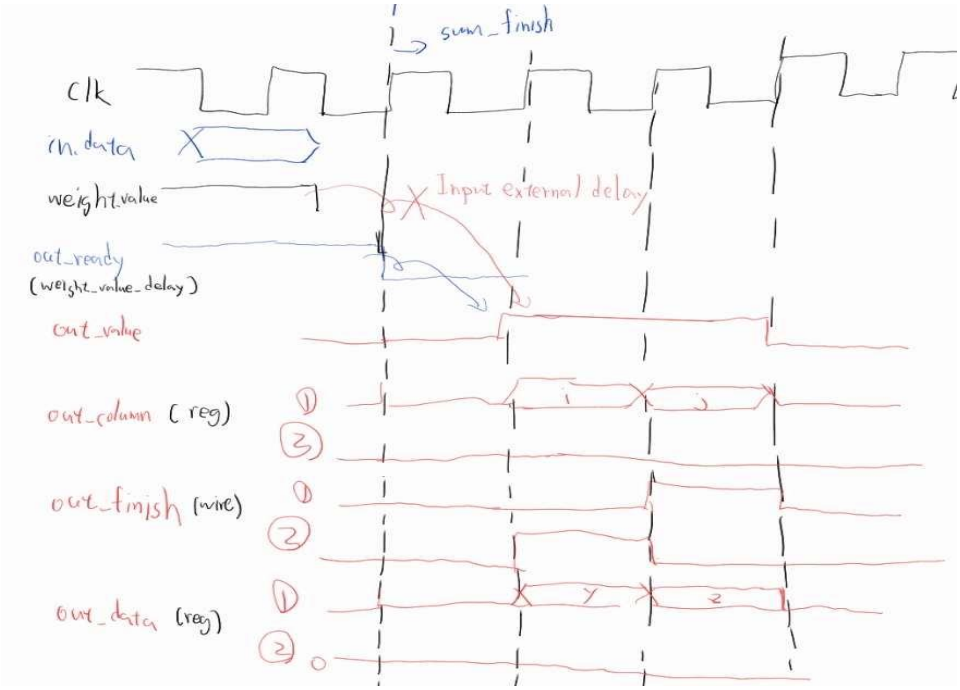
1. Mux 使用 Tri-state Buffer：

在課堂是，老師說使用 Tri-state Buffer 實現 Mux 可以減少面積，下次可以試試看。

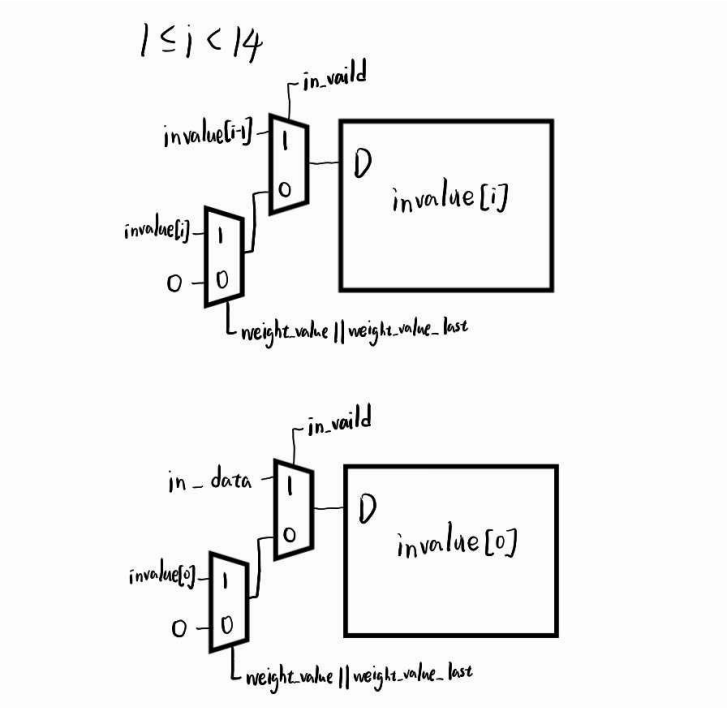
2. 可以將乘法與加法 Pipeline

在運算上，乘法器+加法器會耗掉許多時間，使得 Compiler 需要使用更多面積、低延遲的硬體架構來實現，所以下次有時間，可以試看看 Pipeline。

波型設計(General)



硬體設計架構(Input vector)



硬體設計架構(Matrix)