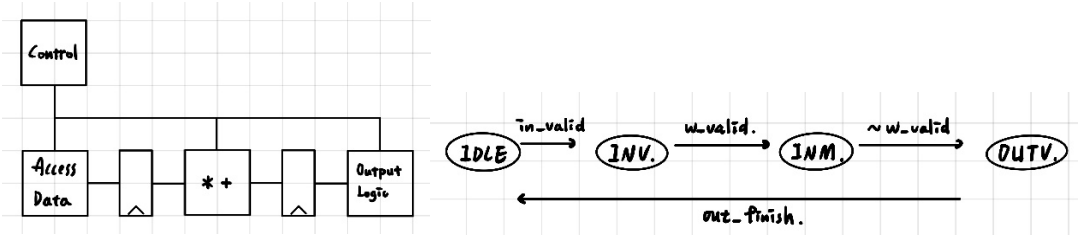
Report_dcs015 110652041 黃鵬宇

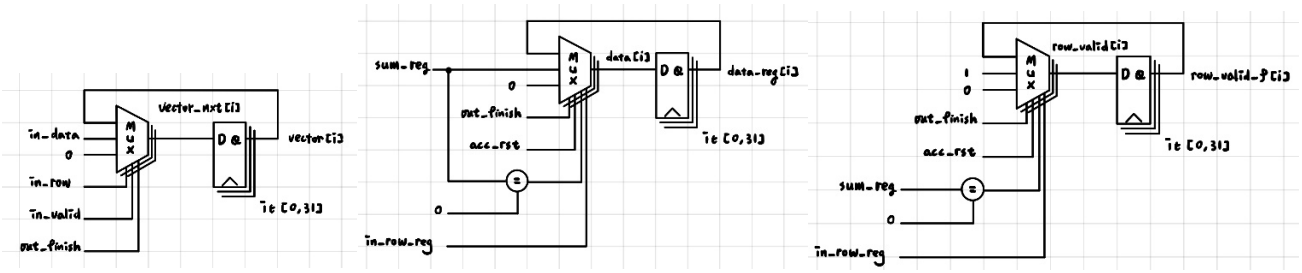
設計方法 & Block diagram

這次 HW 做稀疏矩陣的乘法計算，輸入訊號只有在每個 cycle 給一個非零值與其座標位置，又因為是先給向量再給矩陣，所以不必開多個平行的運算單元，只需要一個乘加器，計算公式：data[mtx_row] = acc + vector[mtx_col] * matrix[mtx_row][mtx_col]

以這個運算架構為主軸去設計控制訊號抓取正確對應的數據，以及需要的暫存器數量。依照 spec 要求與輸入輸出的方式，可以知道此次設計不需要將整個矩陣資料存起來，只需要輸入向量跟輸出向量。並且用 pipeline 架構，縮短 critical path。

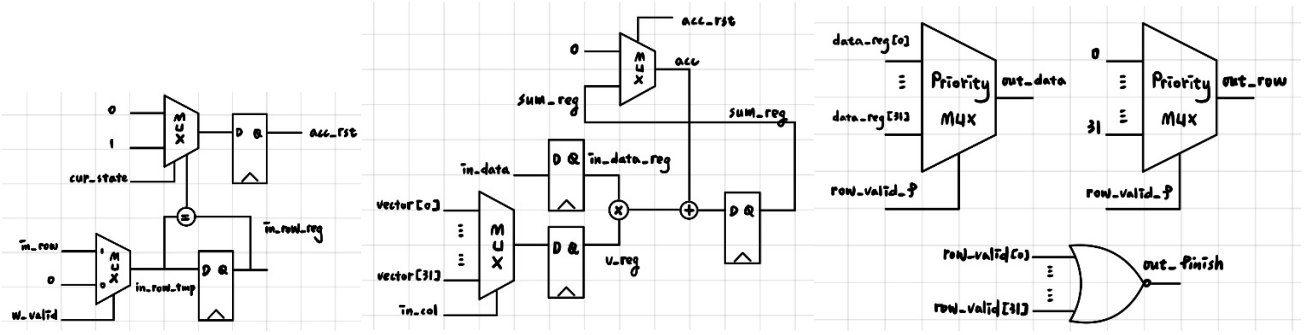


Sequential logic array：對應的 valid 為 enable 訊號更新暫存值，用 out_finish 做 reset，row_valid 是存 data array 的狀態，表示是否有非 0 值存入，並作為輸出的 select 訊號。



Arithmetic, Control, & Output

acc_rst 代表輸出 row 算完，換下個 row，將 acc 清 0，並將算好的 sum 存到 data array。



優化方法

輸入與輸出向量要存，需要 data 與 row 的資料，由於向量的 size 是 32，所以我以 data array 的 index 代表 row，這樣可以省下 row 部分的暫存器面積，不過這樣輸出向量就沒辦法用 shift register 做，需要做一個很大的 priority mux 去決定輸出，但我兩種版本都嘗試後發現省一組 row array 的做法面積比較小。

由於這次 data processing 的過程有多 bits 的乘法運算，所以我有切 pipeline，將輸入的 matrix element 過一級 DFF 再做乘加，主要目的是避免 critical path 吃到 input external delay，在 data required time 較充裕的情況下，design compiler 才能全力將乘法器合出較小面積的電路，兩種版本嘗試後發現這樣改動面積能夠省一萬多。

除此之外，將軟體寫法的 for 迴圈改成 genvar.的硬體寫法，有時也會對電路不管是面積還是 timing 做到優化。我認為比較硬體的寫法能讓 design compiler 更準確解讀 code。

心得與困難

這次 HW 計算的 data 滿大的，運算部分的組合邏輯就直接用*交給 design compiler 處理，而控制訊號，尤其在有切 pipeline 的情況下，data 與 index 那些在 timestamp 的控制要能對應到正確的輸出。Debug 比較常遇到的問題是在 weight_valid 邊界的控制與計算，以及 acc 累加值的邏輯那邊。還有要注意 data 為 0 的情況。