一、 主要設計概念

此次設計是 MAC & Convolution 運算。初次看兩種運算時，會覺得兩種運算沒什麼相似的地方，但仔細思考後，會發現許多相似處。這次設計，我使用了**平行化、Pipeline 技巧**。

二、 運算原理

1. 計算模式：由於輸入、輸出皆只能一行 or 一列，故我設計**每次處理輸出矩陣的一個元素**。此作法約為 **64+p cycles (p: Pipeline 的級數)**。

2. 取值：兩種運算皆以 Row 為單位輸出(可以讓 Out_idx 恆為 0)。

a.矩陣乘法:

取值方式為 Active Matrix 取 nth row, Weight Matrix 取 mth column。用 6-Bit counter 來想像，counter 分解成{n[2:0], m[2:0]}做 counter，此做法能明瞭的輪換矩陣元素。

b.Convolution:

取值方式為 Active Matrix 取 nth col, Weight Matrix 取任意值。我們可以共用 6-Bit counter，只是分解不同，counter 分解成{m[2:0], n[2:0]}，此做法能共用面積，並且這裡的 m 可以當作 Output of mth row，充分使用面積。

三、 特殊方法

1. 在 Convolution 移動 Active Matrix 的設計(減低 Delay 與 Idle 機會)

由上可知，Convolution 設定為輸出一個 row。當切換 row 時，Active Matrix 需要有 3*2 小矩陣元素運算。但因為一次只能讀進一個 col 的數值，故 col 0 資料輸入需要額外 1 個 Cycle。所以使用圖一的方式解決。

2. 8 Numbers 乘法使用 Parallel 與 Pipeline(減低 Delay 與 Cycle time)

Parallel:用我的讀值方式，代表 8 個輸入元素乘法可以同時進行，所以開設了 8 個等價乘法器。

Pipeline:乘法運算中，我分了兩個階層，且此過程只需兩個加法器。第一級：將 Active 元素*2'b11 (意思 $A_{i,j}$+{$A_{i,j}$,1'b0})，再利用 Weight 元素判斷下一級要相加的數值。第二級：將傳遞的數值相加。舉例如圖二。

3. 8 Numbers 加法使用 Pipeline (減少 Cycle time)

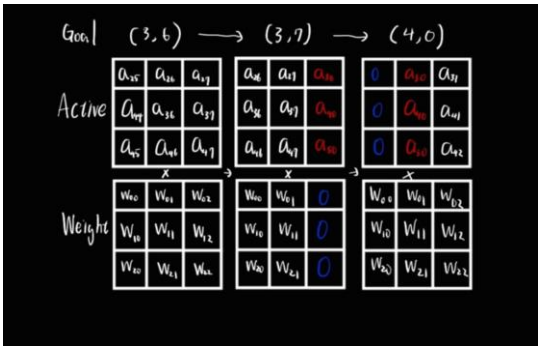
加法運算中，我切了 3(4 in convolution) Stage。截短 Critical Path 的長度來提高 Cycle time。

4.計算每個變數所需的 Bit 數

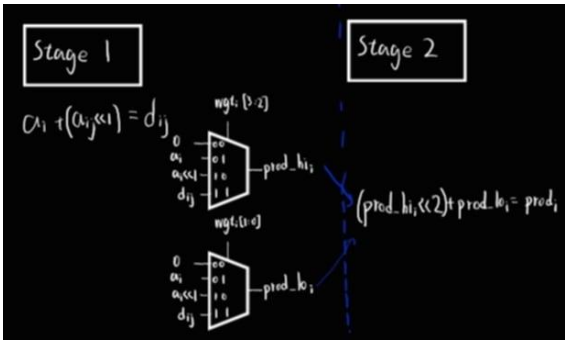
計算過程中所有加法器的 Bit 數，**節省加法器、DFF bit，因為這兩個是佔面積的裝置**。像是 Out_data 僅需 11 位，因為 $9\times(2^4-1)\times(2^4-1)=2^{11}-2^5+2^3+1<2^{11}$，故 Out_data[11]可以設定恆為 0。

5.若簡單邏輯包含 Input、Output port，則不用理會 External delay

就算有一半 Cycle delay，對於簡單邏輯還有充裕時間。

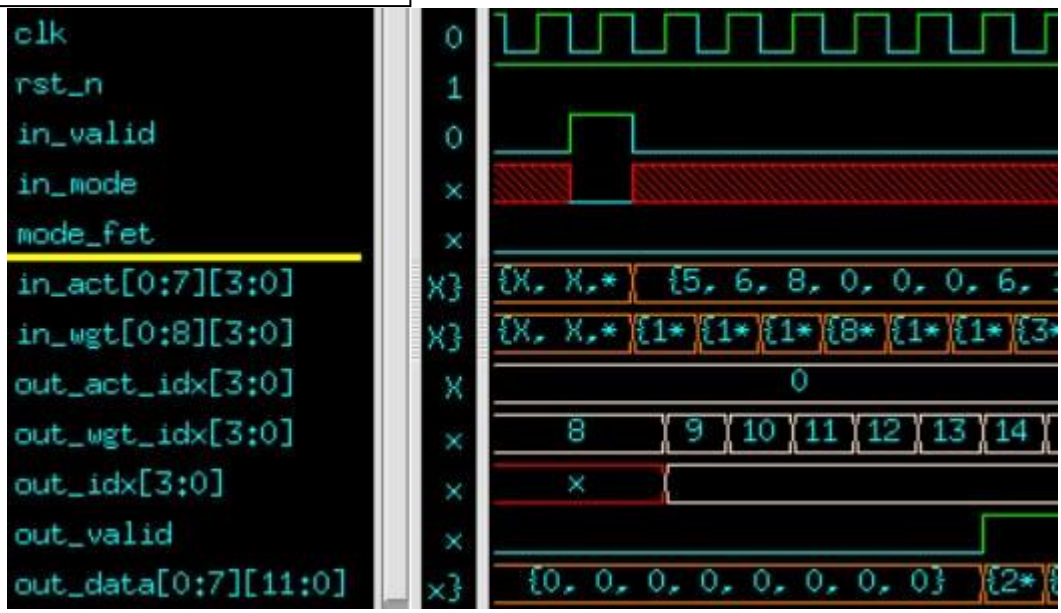


(圖一)



(圖二)

波型設計(用 verdi 圖，原本畫的圖太醜了)



1. 由 In_valid 至 Out_valid 可知，此設計從輸入到輸出有 6 Stage(7:Conv)。

2. 根據 HW4 的示範波型，可知 in_act 跟 in_wgt 在 in_valid 拉起後下一個 cycle 才會開始輸入，所以 out_act_idx 與 out_wgt_idx 調整至 in_valid 後 1.5cycle 開始遞增。

3. Mode_fet 初始設定為 0(Mul mode)。當 In_valid=1 時，Mode_fet 會隨著 In_mode 的值改變。

心得

在使用 Pipeline 時，原本以為要在各個 Stage 放置暫存器，所以會大幅增加面積，但**沒想到僅增加一些面積**，仔細想後，覺得是：把硬體切分成幾層後，若該層原本的 Critical Path 接近 Clock period，則與原本面積一樣；**若該層原本的 Critical Path 小於 Clock period，Design compiler 就會在達成 Critical Path 小於 Clock period 的條件下，換更小的硬體架構**，這是我覺得很酷的地方。

並且由於 Clock Rate 上升，Latency 會大幅下降，畢竟 Latency 可以利用 $(P + Ori) \times T_{clk}$做估算(P 為 Pipeline 級數，Ori 為設計沒有 Delay 時，經過多久 Cycle 可以輸出完成)。所以T_{clk}在這套公式下顯得重要。

令我感到神奇的第二點，是提高 Clock Rate 時，大部分情況面積會提高，**但有遇到是降低的**，這點我倒是想不太出來。

硬體設計架構(Element Switching:本次設計最複雜的部份)

右圖為 Active Matrix 在不同狀況下，輸入的邏輯，是我認為本次滿有挑戰性的地方。