Report_dcs015 110652041 黃鵬宇

Design method

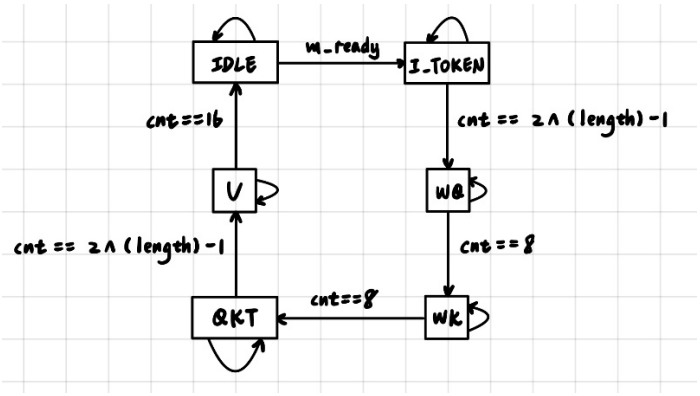
Final project 要實作一種 AI 加速器，兼容多種 size 的矩陣運算，以及優化設計資料流的傳遞，以達到在完成整體運算功能下，最好的電路架構。為縮減 latency，我嘗試用多開平行運算的矩陣乘法架構一個週期產一整個 column 去算 i_token*w 以及 q*k^t，節省 latency，而且每個輸出的 entry 都是 size=8 的內積運算；至於 score*v 的部分，因為涉及到 size=32 的內積，而且因為只要取 o_token 的最後一個 row 輸出，所以這邊的矩陣運算加上 SLT 直接以 score 的最後一個 row 跟 v 做 8 個 size=32 的內積，因為 size 很大以及配合輸出要求，不必在這裡開平行化運算，就一組 size=32 的內積做 8 個週期後輸出就行了。

而這次很特別的是與先前的 HW4 類似，要自己去 access 虛擬記憶體當中的 data，i_token 一次要一個 column，w 一次要一個 row，所以必須設計 FSM 搭配 counter 去決定每個周期要 access 什麼 data，要做那些計算。至於暫存器陣列的部分，我是用 shift reg 實現，這樣可以分清楚整體架構的 Controll logic & Data arthimatic & Matrix register 的部分，以上設計周全後，只需要再將剩下的 routing 部分實現即可。

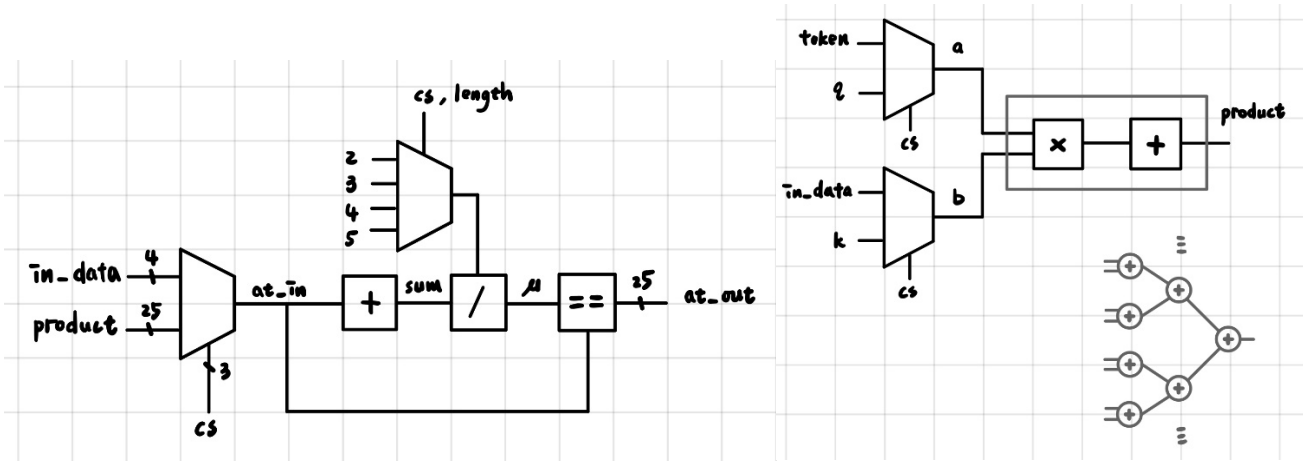
Block diagram

先講 FSM 的部分，如圖所示，接收到 m_ready 訊號之後，進入到 I_TOKEN，這個 state 跟 VROM 拿 i_token，並直接做 RAT 得到；做完就進入 WQ，這邊拿到 wq 後跟 token 做矩陣乘法得到 q，接著進入 WK，跟上個 state 做一樣的事情；再來，有了 q & k 之後，進入 QKT 計算 q*(k^t) 然後直接做 CAT 得到 score；再來是進入 V，算完 token*wv = v 後，直接取 score 的最後一個 row

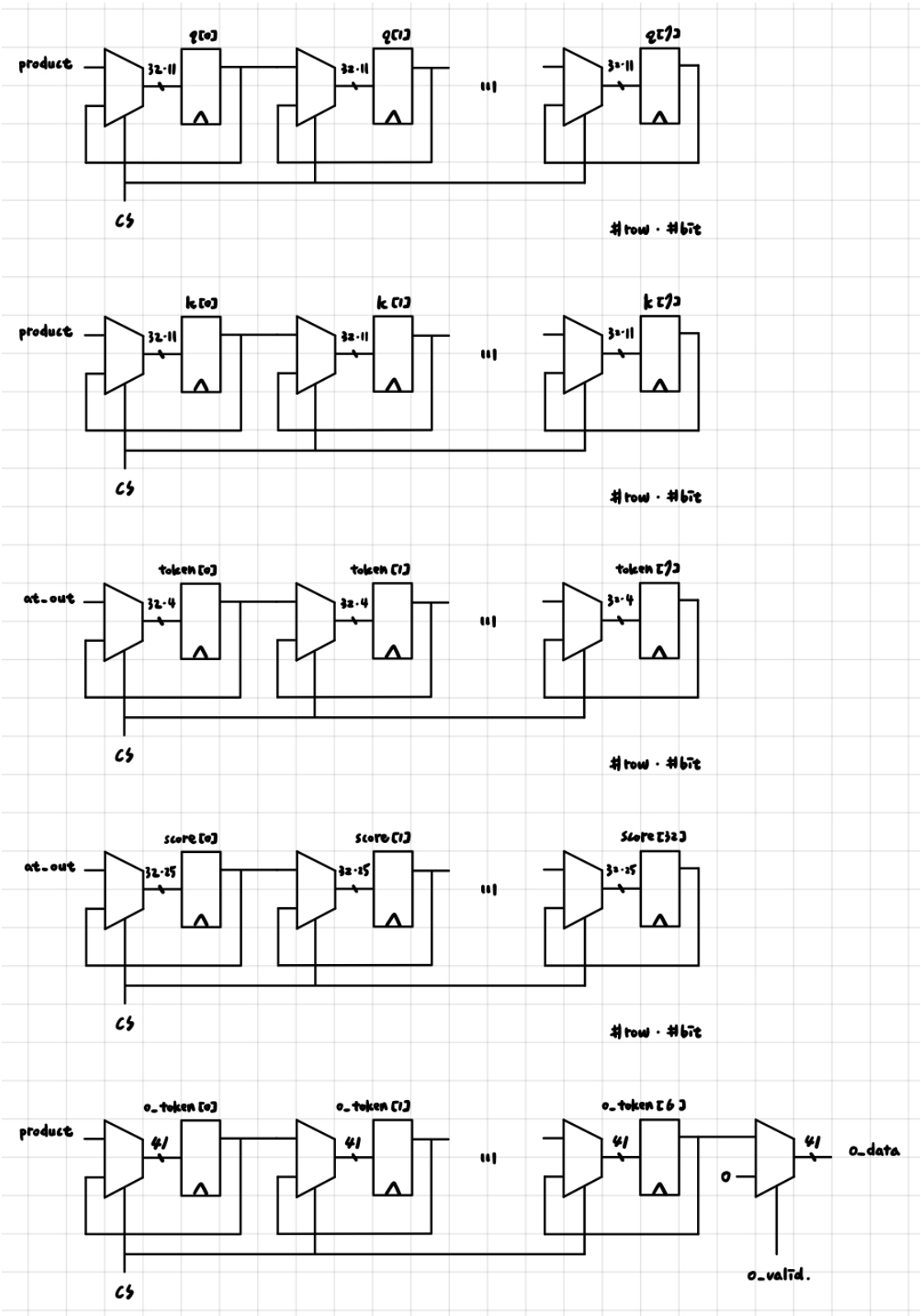
去算 SLT(score*v)的 o_token，然後等 m_ready 掉下來後依序輸出每個 o_data。



Data path 的部分，我將 RAT 與 CAT 用同一個硬體實現，搭配 FSM & length 等 control 訊號，就可以在正確的時間將正確的資料送進去算出 threshold 並砍掉比 threshold 小的 data。而矩陣乘法分成兩塊，一個是 32 份平行化 size=8 的內積運算邏輯去算 token * wq, token * wk, token * wv,　q * (k^t)，另一個是沒有平行化的 size=32 內積運算邏輯輸入 score 與 v 然後算出 o_token。



至於暫存器的部分，我是用二維的 shift register 實現，基本上用 control logic 控制好何時該 reset 或是寫入 data 和推送，要特別注意的是由於 token 是 row-wise ordering，所以是 shift row；其他像是 q, k, score 都是 column-wise ordering，是 shift column。而 o_token 就將每個 entry 的 data 逐步推向 o_data。



Difficulties

這次的 design 比以往的 HW 大且複雜，需要在多個 block 之間做資料傳輸與溝通，剛開始有點不知道如何起步，而且有很多想法的時候也會慌亂，不知道該寫什麼。後來根據修過 iclab 學長給的

建議，對於較大的電路，可以先把比較簡單的架構先寫出來，跑過都沒問題之後再去進一步想優化。在完成這份 project 的過程中也體會到：先寫出個初始版本心裡也比較踏實，而且等到第一版出來，會對於近一步優化有更明確的方向，使得後續的優化更有效率。

除此之外，debug 的時候也遇到之前較少見的問題，由於這次使用到二維 shift register 要將矩陣 data 存起來，由於多層迴圈看得眼花撩亂，但不小心在 shift 推送資料的邏輯小細節弄錯，導致後續要計算的 data 吃到 X，剛開始乍看認為接線沒接錯到底怎麼會有 X，也是自己花很多時間才慢慢發現問題，像是陣列宣告跟數值位元寫反要 shift column 但寫成 shift row 等等。後來我用 generate 的方式寫迴圈就比較簡潔，也比較容易 debug。

還有像是這種比較大的 design，因為電路較複雜，訊號也比較多，所以我這次是邊寫 code 邊做測試，我先寫完 FSM 跟 counter 後就先跑 01 看看在超過最大 execution cycles 的模擬波形下先看看控制訊號是否如自己預先規劃的樣子。再來才是慢慢將運算的 data path 兜上去。

Thoughts

經過這次 FP，透過更大更複雜的 design，讓我意識到這與先前寫 lab 和 hw 的策略有所不同。我覺得更需要在開始寫 code 之前就先擬定好 control logic，再去想有哪些硬體可以共用，尤其這種比較大的 design 也更容易找到可以共用的方，像是這次 FP 當中 RAT 與 CAT 的部分，以及矩陣乘法；而在很長的 data path 當中，切 pipeline 時，只要每一筆測資不用接連著輸入與輸出，也可以考量到哪些站存陣列已經用完，就可以將後面的的訊號存進去，達到共用暫存器省 noncombinational 電路的面積。總結來說，這次寫比較大的電路，不僅可以銜接之後修 iclab，也更清楚怎樣的實作過程比較有效率。