VGG16 硬體加速器

壹.電路合成與結果分析

第一部分: 設計原理

一、設計原理

因為有兩層要做，所以設計成兩個模式，Mode Layer1、Model layer2，使用 8 個 PE。

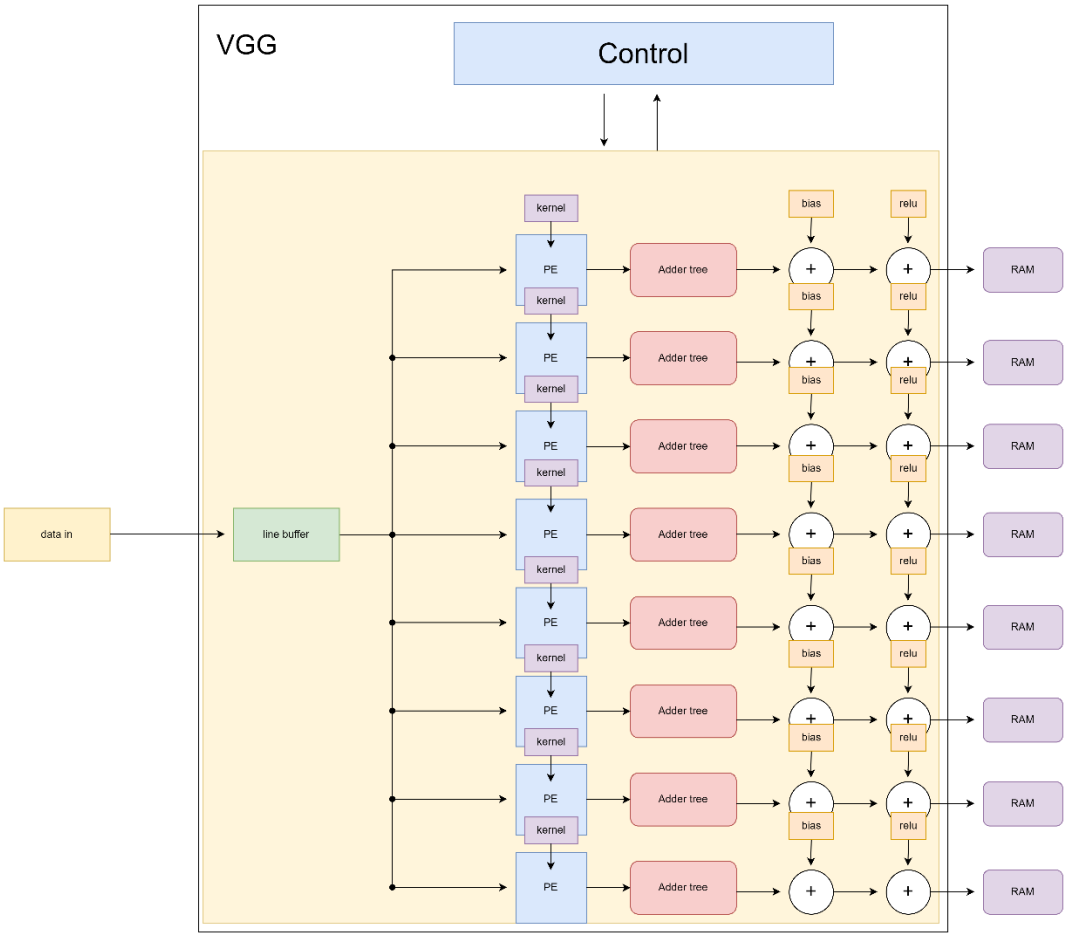
- Mode Layer1:

輸入是 1 張圖像，分為 RGB，做完 Padding 後。先做第 1 到 8 張圖，第一次將 R 放入 Line Buffer，做完 Convolution 後將直存入 RAM。第二次，更換成 G 的 Kernel，將 G 放入 Line Buffer，做完時讀入 R 的結果做相加，再存入 RAM。第三次，更換成 B 的 Kernel，將 B 放入 Line Buffer，做完時讀入 R+G 的結果做相加，再加 Bias，做 ReLu。此時已完成 1 到 8 張圖，換做 9-16 張，換新的 Bias，依此類推，直到做完第 64 張。

- Mode Layer2:

輸入是 64 張圖像，做完 Padding 後，因只有一個 Line Buffer，先將第 1 張圖放入 Line Buffer，做第 1 到 8 張圖，存入 RAM，再換下 8 張，直到做完 64 張。改放第 2 張到 Line Buffer，算出的結果與第一張的相加，再存入 RAM，直到做完 64 張。依此類推，當放入 Line Buffer 的是第 63 張，算出的結果，加上 Bias 後做 ReLu，輸出第二層的圖。

第二部分: 架構圖



說明:

VGG : 整體電路

Control : FSM 控制電路。

Line buffer : 共三組空間，分別存入 RGB 圖像，每組為 226*226+32。

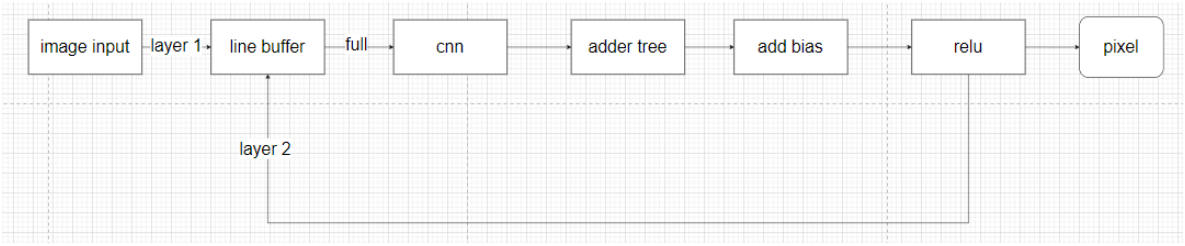
PE : 輸入 kernel 資料，進行捲積運算，共 9 個乘法器。

Adder tree : 兩兩進行相加。

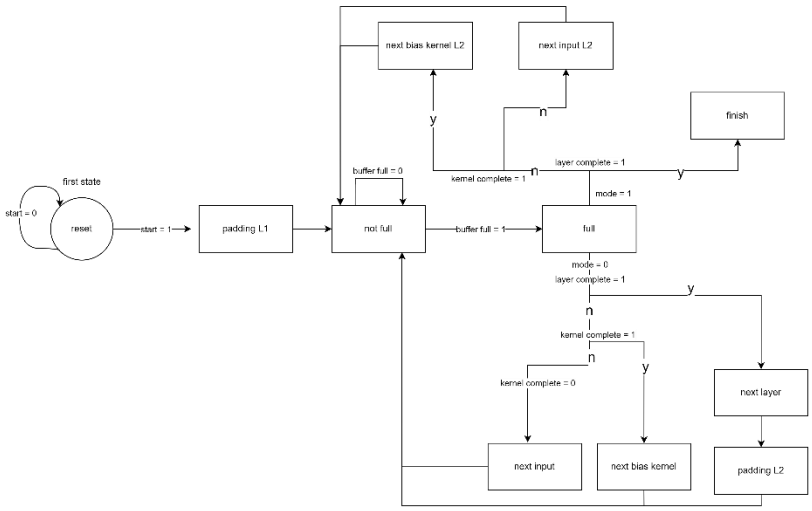
Bias : 加上 bias。

Relu : 進行 relu 處理

- 流程圖



■ 狀態機



貳、 模擬結果

一、合成結果 - area、power、delay

Comparison of Synthesis Results

	Area (um2)			Timing	Critical path	Power(W)		
Opt type	Combonational	Seqential	Total	(ns)	(ns)	Dynamic	Leakage	Total
area	46470.41	20343.96	66814.37	5.5	5.39	671.38 u	3.47m	4.1 m

二、截圖

- Timting

```
data arrival time                                                  5.39

clock clk (rise edge)                                 5.50         5.50
clock network delay (ideal)                           0.00         5.50
genblk1_4__bias_relu/pixel_write_reg_0_/CK (DFFRPQ_X2M_A9TR)
                                                      0.00         5.50 r
library setup time                                   -0.04         5.46
data required time                                                 5.46
-----------------------------------------------------------------------
data required time                                                 5.46
data arrival time                                                 -5.39
-----------------------------------------------------------------------
slack (MET)                                                        0.07
```

- Area

```
Combinational area:              46470.411786
Buf/Inv area:                     2228.083171
Noncombinational area:           20343.958991
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                 66814.370777
Total area:                 undefined
```

- Power

```
  Cell Internal Power  =   3.1048 mW   (90%)
  Net Switching Power  = 360.2630 uW   (10%)
                         ---------
Total Dynamic Power    =   3.4651 mW  (100%)

Cell Leakage Power     = 671.3828 uW
```

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)	
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)	
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)	
clock_network	0.0000	0.0000	0.0000	0.0000	(0.00%)	
register	2.5275	1.6463e-03	191.5534	2.7207	(65.78%)	
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)	
combinational	0.5773	0.3586	479.8076	1.4157	(34.22%)	
Total	3.1048 mW	0.3603 mW	671.3610 uW	4.1364 mW		

參、問題與討論

1. 如何加快運算速度: 提高平行度，但面積會上升。

2. 如何增加除錯效率 : 使用階層式撰寫，電路分成控制區與運作區，善用狀態機控制，單一電路的功能單一，資料寫出檔案檢視。

3. 資料輸入 line buffer 時，同時存入兩個 array:

如:

buffer[1] <= buffer[0]

buffer[0] <= data_in

理論上，buffer[0]與 buffer[1]的數值不會都是 data_in，但他在運作時，data_in 會同時給 buffer[0] and buffer[1]。

解決方法: 多給一個 array 空間，去緩衝同時給值的問題，治標但不治本。

肆、結論

神經網路是採交錯比對，本次電路在比對次數上 layer2 是 layer1 的 21 倍，且 layer1 計算就已經需要一段時間，layer2 更為可觀，因此硬體加速電路確實是必較的。

心得

VGG 相較於前幾次作業是很大的電路，而且只做了 VGG-16 的前兩層，控制是整體上最困難的步驟，製作 Process Element 並不困難，如: PE、linebuffer、adder tree。最初，我採用 TB 對電路進行控制，隨著開發的進展，我發現這樣的設計並不好，需要控制眾多位數龐大的 data input，picture padding、pixel writing、element state control，處理的功能過多導致電路複雜且難以掌握，所以後來我引入狀態機，雖然會增加電路面積，但是對電路的可控性有重要的幫助。

VGG layer1 的計算很快，但 VGG layer2 的計算很久。理論上，VGG layer1 處理的 kernel 次數為 1 pictures * 3 channels *64 cnn times * 224 scan times * 1 kernel = 43008 kernels；而 VGG layer2 處理的 kernel 次數為 64 channels * 64 cnn times * 224 scan times * 1 kernel = 717504 kernels。Layer2 比 layer1 需要 21 倍的計算次數，假設每次計算需要一個 clock，廣義上運算時間會多 21 倍。

運算圖像最重要的就是資料存取、傳輸要正確，因此需要時刻檢視資料是否正確，使用 terminal 檢視這麼龐大的資料並不方便，因此寫出檔案檢視是將對方便且有效率的做法。電路我採用階層式的寫法，在開發管控與除錯上比較方便，如果全寫在一起不好檢視也不好除錯。此外，因為控制電路並非寫在 TB，所以 post sim 如果要檢視狀態的話，除非有設置輸出狀態訊號到 TB，不然無法得知運作情況，只能等待他結束確認。

經過 VGG 的洗禮，我相當明白為甚麼神經網路需要硬體加速的設計，資料交錯比對的次數十分龐大，且這次電路只有前兩層神經網路，但 VGG16 有 16 層，簡單計算上至少目前電路 8 倍的時間，最大可能是 16 倍，開發時間會相當可觀。