HW10

資訊所 P76134082 陳冠言

1. 結果是否正確

```
start initial!

initial finish!

all finish!
Congradulation ! ALL pass
```

上圖為 FPGA 系統產生的輸出結果與作業提供的 golden.hex 檔案進行比對。
比對結果是正確的

2. convolution ip 狀態機

state 0 (IDLE): 初始狀態或等待狀態。
當 start 信號為高電位時，初始化內部計數器與索引（array_index, window_index, weight_index, window_head, move_state, cal_state, write_back_index），並將 send 和 stall 設為 1，然後轉換到 state。

state 1 (INITIAL_WEIGHT): 初始化權重 (kernel) 值。
透過 M0 記憶體介面讀取權重。作業說明中 kernel 有 9 個值。您的程式碼中 weight_index 會計數到 9（讀取 10 個值，weight 到 weight[9]）。這可能是包含了 9 個 kernel 值以及一個額外的值，或者 weight[9] 未被使用，因為偏置 (bias) 是在 state 2 單獨讀取的。
send 信號控制讀取請求的發送與資料的接收。每讀取一個權重值後，weight_index 增加。 完成後轉換到 state 2。

state 2 (INITIAL_BIAS): 初始化偏置 (bias) 值。
從 M0 記憶體的特定位址 (793*4) 讀取偏置值。
完成後轉換到 state 3。

state 3 (INITIAL_WINDOW_ELEMENTS): 初始化卷 window 的初始元素。
作業系統架構圖暗示輸入影像資料儲存在 BRAM 中。
此狀態從 M0 記憶體讀取輸入影像的部分初始像素值填入 window 陣列。
window_index 和 array_index 控制讀取位址和儲存位置。

state 4 (TAKE_NEW_ELEMENTS_AND_MOVE): 為卷積窗口取新的元素並準備移動窗口。根據 move_state 的值，從 M0 記憶體讀取三個新的像素點，這些像素點通常是當前卷積窗口右側的新一列數據。

state 5 (CALCULATE): 執行卷積運算。此狀態分為三個子步驟，由 cal_state 控制： cal_state 0 (Product): 計算 window 中的 9 個元素與對應 weight 的乘積，結果存於 product 陣列 (64 位元)。 cal_state 1 (Rounding): 根據作業要求，對 64 位元的乘積結果取 [47:16] 的部分，並進行四捨五入（如果 [15:0] 大於等於 16'h8000 則進位）。結果更新回 product 陣列的低 32 位元。 cal_state 2 (Summing): 將 9 個經過處理的乘積結果累加，並加上偏置 bias，最終結果存於 ans。 完成後轉換到 state 6。

state 6 (WRITE_BACK_ANS): 將計算得到的卷積結果 ans 寫回 M1 記憶體。M1 的寫入位址由 write_back_index 控制，每次寫入後遞增。

state 7 (SHIFT_WINDOW_OR_FINISH): 移動卷積窗口的元素，為下一次計算做準備，並檢查是否所有運算已完成。如果 write_back_index 達到 676 (26x26 的輸出)，則設置 finish 信號為高電位，表示運算結束。否則，向前移動 window 陣列中的元素（window <= window[1], window[1] <= window[2] 等）。

state 8 (RESET_NEW_SIX_ELEMENTS_FOR_NEW_ROW): 當卷積窗口移動到新的一行時，重新初始化窗口的部分元素（類似 state 3，但可能是針對新行的起始部分）。 透過 M0 記憶體讀取數據填充 window 陣列。完成後轉換回 state 4 繼續正常滑動窗口運算。

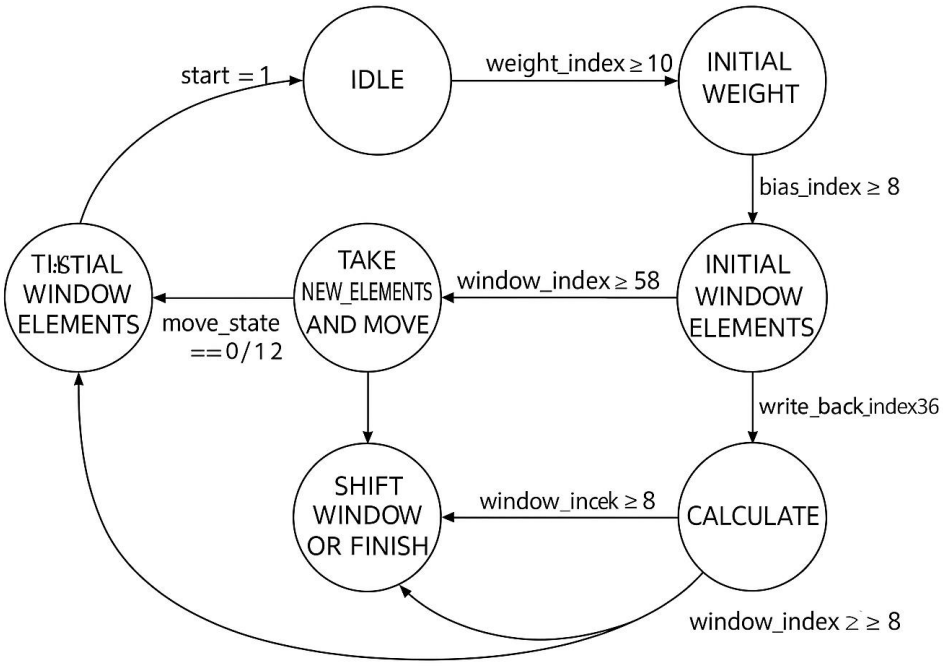
3. 學到什麼?

完成這個專案，您應該能學到以下知識與技能：

1. **卷積運算原理與硬體實現：**
    - 理解卷積神經網路 (CNN) 中卷積層的基本運算過程。
    - 學習如何將卷積演算法（包括乘法、截位/四捨五入、累加）映射到硬體邏輯。
2. **FPGA 設計與 Verilog HDL：**
    - 使用 Verilog HDL 設計複雜的數位邏輯系統。
    - FSM (Finite State Machine) 設計：學習如何使用狀態機來控制複雜的循序操作流程。
    - 記憶體介面設計：理解並實作與 BRAM (Block RAM) 等記憶體模組的讀寫交互，包括處理讀取延遲。
    - 資料路徑設計：規劃資料如何在不同運算單元間流動和處理。
3. **系統層級設計：**
    - IP (Intellectual Property) Core 的概念與設計：將特定功能（如卷積運算）模組化為一個可重複使用的 IP 核。

- 軟硬體協同設計初步概念：雖然軟體主要用於資料搬移和驗證，但也體現了軟硬體如何分工合作。

4. **FPGA 開發流程與工具（間接）：**
   - 雖然未直接操作，但作業要求提交 .bit 和 .tcl 檔案，暗示了需要經過合成 (Synthesis)、實作 (Implementation) 等 FPGA 開發流程。Vivado 是常用工具之一。
5. **硬體驗證與除錯：**
   - 學習如何透過與「黃金參考 (golden reference)」比對來驗證硬體設計的正確性。
6. **硬體加速概念：**
   - 體會到 FPGA 在加速如卷積運算這類計算密集型任務上的潛力，尤其是在邊緣 AI 應用中。
7. **效能考量（初步）：**
   - 透過窗口緩衝 (window buffer) 等技巧來優化資料重複使用，減少記憶體頻寬需求。



Convolution-IP-state transition diagram

4. 更多

我認為此設計的挑戰與優化方向：

- 記憶體頻寬：卷積運算通常是記憶體頻寬受限的，如何有效地管理片上記憶體 (BRAM, Distributed RAM) 和外部記憶體之間的資料搬移至關重要。

- 資源利用率： FPGA 上的邏輯資源（如 LUT, FF, DSP）是有限的，設計時需要在效能和資源消耗之間取得平衡。
- 平行化策略： 可以透過輸入通道平行、輸出通道平行、卷積核內平行等多種方式提高運算通量。您的設計主要體現在 MAC 單元的循序處理，但更高效的設計會使用平行 MAC 單元。
- 管線化 (Pipelining)： 將複雜運算分解為多個階段，並讓這些階段重疊執行，以提高時脈頻率和處理效率。