**REPORT**

**Thiết kế và thực hiện khối tính nhân chập 2-D bằng phương pháp HLS**

Ver 1.0

08/11/2024

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Họ và tên (Full name)** | **Mã SV (ID)** | **Đóng góp (Contribution)** |
| Thành viên 1 (Member 1) | Long Trí Thái Sơn | 22022548 | Viết mã nguồn cho hàm nhân chập + testbench, viết báo cáo. |
| Thành viên 2  (Member 2) | Hoàng Đăng Khoa | 22022563 | Thêm dẫn hướng, phân tích kết quả, viết báo cáo. |
| Tên/Địa chỉ Repo trên Github | https://github.com/khoavpt/hardware-for-deep-learning-final-project | | |

|  |
| --- |
| **Abstract (from 5 to 10 lines)** |
| Báo cáo trình bày về quá trình thiết kế khôi nhân chập 2D bằng phương pháp HLS sử dụng ngôn ngữ C và tổng hợp phần cứng bằng bằng Vivado HLS.. Nhóm cũng áp dụng các kỹ thuật tối ưu hóa như phân chia mảng, pipeline và unroll giúp làm giảm đáng kể độ trễ và cải thiện hiệu năng hệ thống |

|  |
| --- |
| **Keywords** |
| Vivado HLS, tích chập 2D, tối ưu. |

|  |
| --- |
| **Hướng dẫn (Guide)** |
| Sinh viên điền vào báo cáo theo mẫu đính kèm. Sinh viên điền các mục:   * Thông tin sinh viên, mã số sinh viên * Mục *Đóng Góp* điền các công việc đã làm tương ứng của từng sinh viên. * Tên/Địa chỉ Repo trên Github hoặc Google Drive   Ngoại trừ phần thông tin sinh viên, mã số sinh viên và tên/địa chỉ Repo trên Github hoặc Google Drive ở đầu, sinh viên cần hoàn thành các phần nội dung (theo các mục đã được gợi ý – nhưng không hạn chế) trong phần báo cáo để mô tả các công việc nhóm đã thực hiện và các kết quả đã đạt được.  Sinh viên làm theo nhóm 2 người, chỉ cần **1 sinh viên đại diện nộp 1 bản báo cáo, sửa tên file thành tên của các thành viên trong nhóm (viết có dấu).**  Sinh viên nộp lại báo cáo này trước khi tới trình bày kết quả, **muộn nhất trước ngày thi hết môn một ngày**. **Ngày thi, SV cần mang máy tính laptop và sản phẩm để chạy demo!**  **Lưu ý: Nghiêm cấm mọi hình thức copy bài (bao gồm cả report và mã nguồn) của nhau. Nếu phát hiện sự giống nhau giữa 2 bài thì tùy mức độ mà có thể sẽ bị trừ điểm hoặc chia lấy điểm trung bình làm điểm của project.** |

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Time** | **Revised by** | **Description** |
| V1.0 | 08/11/2024 | Nguyễn Kiêm Hùng | Original Version |
| V2.0 | 18./12/2024 | Hoàng Đăng Khoa | Report added |
|  |  |  |  |

MỤC LỤC

[Document History 3](#_Toc181630605)

[MỤC LỤC 4](#_Toc181630606)

[1. Giới thiệu 5](#_Toc181630607)

[2. Yêu cầu 6](#_Toc181630608)

[2.1. Yêu cầu đối với thiết kế: 6](#_Toc181630609)

[3. Thuật toán 6](#_Toc181630610)

[4. Mô hình C và testbench 7](#_Toc181630611)

[5. Tối ưu thiết kế để nâng cao hiệu năng 7](#_Toc181630612)

[6. Mô phỏng/thực thi và đánh giá 7](#_Toc181630613)

[7. Kết luận 7](#_Toc181630614)

[Appendix A: 8](#_Toc181630615)

[Appendix B: VHDL Code 9](#_Toc181630616)

[(đóng gói thành tệp nén và gửi kèm báo cáo) 9](#_Toc181630617)

[Appendix C: 10](#_Toc181630618)

[List of Figures 11](#_Toc181630619)

[List of Tables 12](#_Toc181630620)

[References 13](#_Toc181630621)

# Giới thiệu

*(Introduction to the motivation, Objectives, and main Contents of the project)*

**Mục tiêu:** Vận dụng các kiến thức, kỹ năng đã được học đểthiết kế, mô phỏng và thực thi một mô-đun phần cứng thực hiện tính tích chập J = 2DConV(I, K) giữa hình ảnh lối vào I với một ma trận kernel K ([1] ). Trong đó, mỗi pixel trong hình ảnh tích phân J đại diện cho tổng tích lũy của tích điểm-điểm giữa ma trận K với một ma trận cùng kích thước được trích xuất từ ma trận đầu vào I. Phép tính tích chập được mô tả bằng ví dụ sau.

***Ví dụ:*** nếu hình ảnh đầu vào I là ma trận có kích thước 5×5 như sau:

I =

Và kernel có kích thước 3×3 như sau:

K =

thì kết quả tính toán của khối 2D-Convolution trả về hình ảnh J có kích thước 3×3 như sau:

C =

Quá trình tính toán được minh họa trong Hình 1.

A picture containing shape

Description automatically generated

Hình 1. Ví dụ minh họa phép tính tích chập 2D.

Lưu ý rằng pixel có tọa độ (r,c) – (hàng, cột) – tronh ảnh lối ra được tính bằng cách nhân chập điểm – điểm giữa ma trận kernel và ma trận 3×3 có tâm nằm ở vị trí có tọa độ (r + 1, c + 1) trong ảnh lối vào.

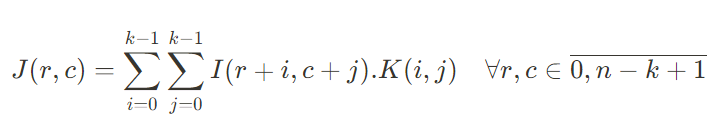
# Yêu cầu

## Yêu cầu đối với thiết kế:

* Xây dựng thuật toán thực hiện tính nhân chập 2D.
* Xây dựng bản mô tả mức cao cho thuật toán bằng ngôn ngữ C,
* Xây dựng testbench để kiểm chứng thuật toán bằng C simulation
* Tổng hợp bản mô tả C thành bản mô tả RTL bằng VHDL/Verilog bằng Vivado HLS
* Mô phỏng C/RTL co-simulation để kiểm chứng và phân tích hoạt động của phần cứng sau tổng hợp HLS
* Tối ưu mô hình C và thêm vào các dẫn hướng để cải thiện hiệu năng của thiết kế
* Thực hiện thiết kế trên ZynQ-7000 để đánh giá lượng tài nguyên sử dụng, hiệu năng.

# Thuật toán

Thuật toán được thực hiện cài đặt trong bài báo cáo này là phép nhân chập 2D (2D Convolution) để thực hiện tính nhân chập giữa ma trận hình ảnh đầu vào và ma trận kernel. Công thức tính tổng quát được mô tả như sau:



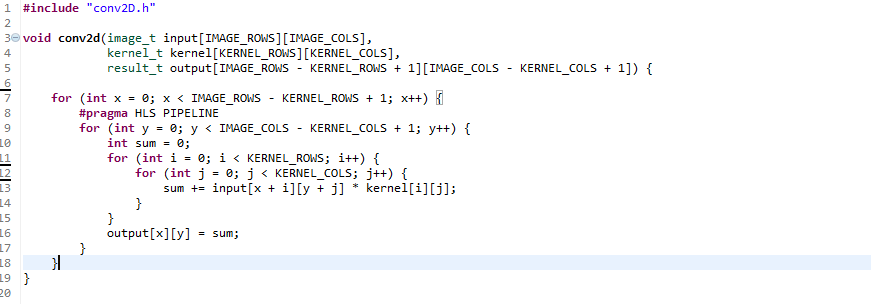
Trong đó :

* là ma trận ảnh input đầu vào với kích cỡ là
* là ma trận kernel với kích cỡ là
* là ma trận kết quả của phép nhân chập 2D với kích cỡ là

# Mô hình C và testbench

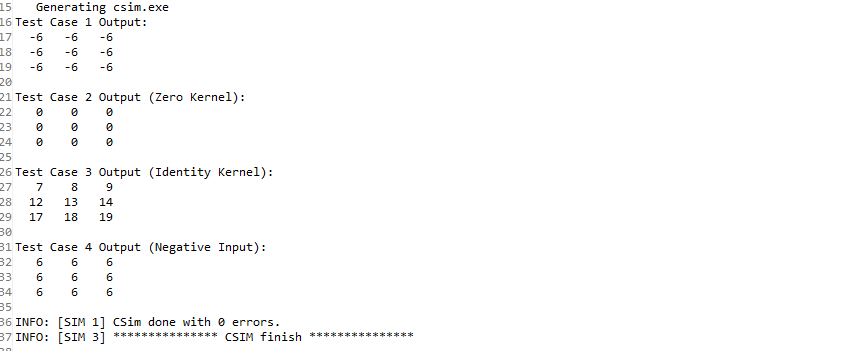
**Mô hình C:** Chương trình cài đặt phép tính nhân chập được cài đặt bằng ngôn ngữ C sử dụng 4 vòng lặp lồng nhau, trong đó:

* 2 vòng lặp ngoài duyệt qua từng pixel của ma trận đầu ra.
* 2 vòng lặp sau thực hiện phép nhân chập giữa ma trận kernel và vùng tương ứng trong ma trận đầu vào.



*Ảnh mã nguồn C (chưa áp dụng dẫn hướng)*

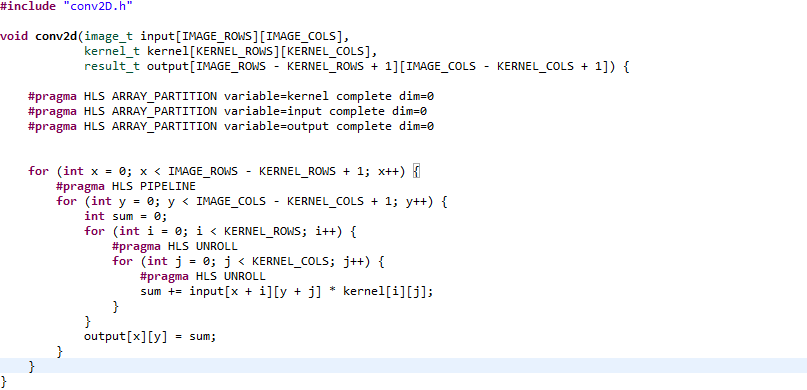
**Testbench:** Tính đúng đắn của khối nhân chập được kiểm chứng qua thử nghiệm 4 trường hợp (basic, zero kernel, identity kernel và negative input) với cá ma trận ảnh đầu vào với kích cỡ 5x5 và các kernel kích cỡ 3x3.



*Kết quả thử nghiệm cho thấy khối nhân chập hoàn thành tất cả test với 0 errors*

# Tối ưu thiết kế để nâng cao hiệu năng

Để nâng cao hiệu năng thiết kế phần cứng cho khối nhân chập 2D, nhóm áp dụng một số dẫn hướng (directives) như sau :



*Ảnh mã nguồn C (sau khi áp dụng dẫn hướng)*

1. Chia mảng kernel, output và input thành các phần tử độc lập, mỗi phần tử được lưu riêng biệt thay vì cùng trong một khối bộ nhớ.

    #pragma HLS ARRAY\_PARTITION variable=kernel complete dim=0

    #pragma HLS ARRAY\_PARTITION variable=input complete dim=0

#pragma HLS ARRAY\_PARTITION variable=input complete dim=0

* Cho phép truy cập đồng thời vào các phần tử của mảng => Cho phép áp dụng unroll để tính chập

2. Sử dụng pipeline cho vòng lặp x để giảm độ trễ giữa các lần lặp

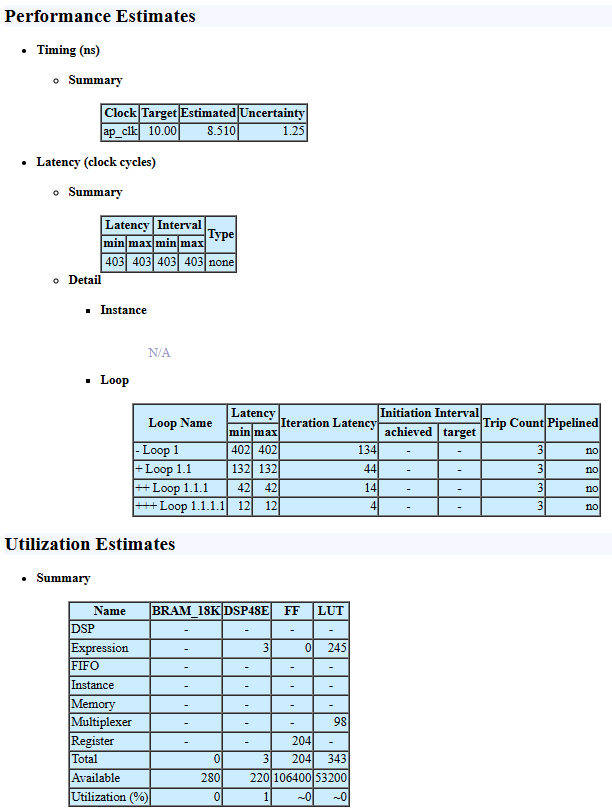
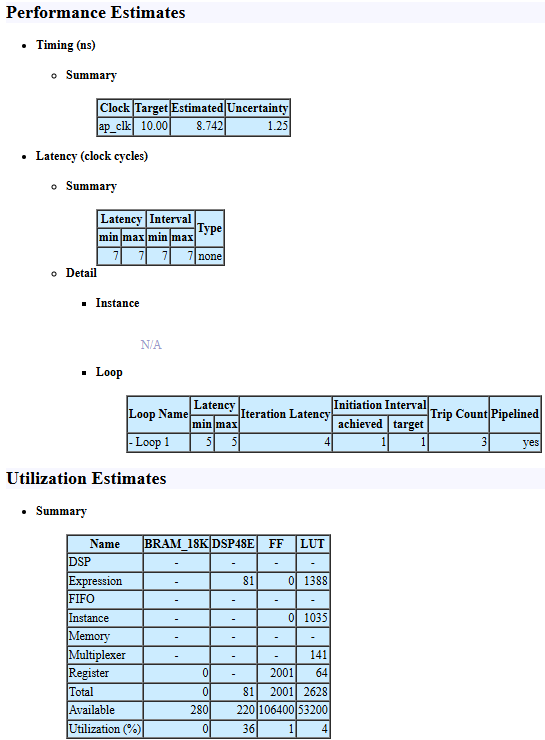
        #pragma HLS PIPELINE

3. Sử dụng unroll cho hai vòng lặp để cho phép thực thi song song bước tính nhân chập của từng vị trí ảnh.

#pragma HLS UNROLL

# Mô phỏng/thực thi và đánh giá

Nhóm đã tiến hành mô phỏng C (C simulation) và đồng mô phỏng C/RTL (C/RTL cosimulation) để kiểm tra tính đúng đắn của thiết kế. Kết quả mô phỏng cho thấy đầu ra từ mô hình C đúng với testbench và khớp hoàn toàn với đầu ra từ RTL. Không có lỗi xảy ra trong quá trình kiểm thử.

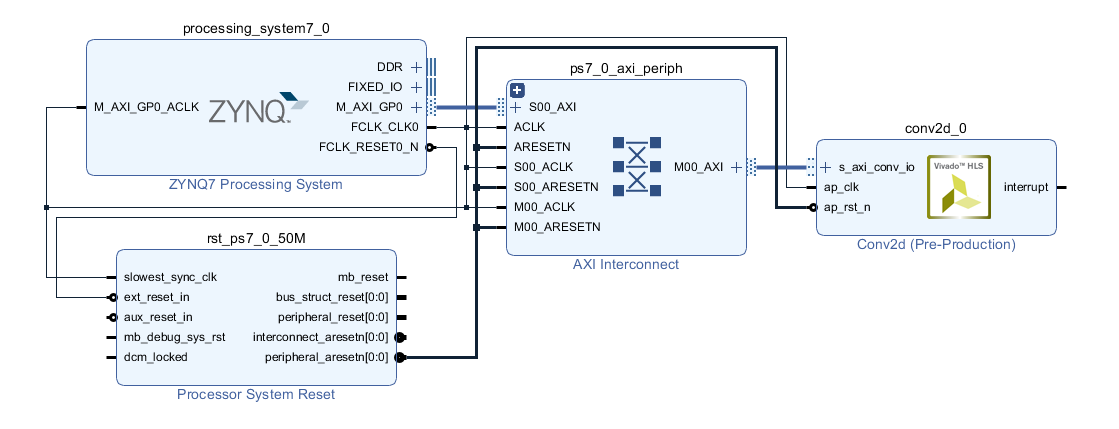
 

*Kết quả tổng hợp trước và sau khi sử dụng dẫn hướng*

Việc áp dụng các dẫn hướng cho ARRAY\_PARTITION, PIPELINE và UNROLL đã giúp giảm độ trễ trên testbench từ 403 xuống còn 7 chu kỳ xung nhịp. Tuy nhiên áp dụng song song hóa cũng dẫn đến sự tăng đáng kể trong tài nguyên sử dụng như:

* Registers (tăng vì cần lưu các biến tạm thời như thông tin mảng, vòng lặp): 204 -> 1947
* LUT (tăng vì cần thực hiện nhiều phép tính song song): 343 -> 2462
* DSP (tăng vì cần thực hiện nhiều phép tính song song): 3-> 81

Mặc dù việc song song hóa làm tăng đáng kể mức sử dụng tài nguyên, nhưng tỷ lệ sử dụng phần cứng vẫn ở mức chấp nhận được. Tuy nhiên nếu kích thước đầu vào tăng đáng kể cần xem xét thêm các kỹ thuật tối ưu khác để cân bằng giữa hiệu năng và mức tiêu thụ tài nguyên



*Sơ đồ khối thiết kế SoC tích hợp bộ tăng tốc Conv2D qua giao diện AXI*

# Kết luận

Nhóm đã thành công thiết kế và tối ưu hóa khối nhân chập 2D sử dụng VivadoHLS. Kết quả mô phỏng cho thấy thiết kế hạt động chính xác với độ trễ giảm đáng kể (từ 403 chu kỳ xuống 7 chu kỳ)

Mặc dù việc song song hóa làm tăng đáng kể mức sử dụng tài nguyên, nhưng tỷ lệ sử dụng phần cứng vẫn ở mức chấp nhận được. Tuy nhiên nếu kích thước đầu vào tăng đáng kể cần xem xét thêm các kỹ thuật tối ưu khác để cân bằng giữa hiệu năng và mức tiêu thụ tài nguyên

Nhóm xin gửi lời cảm ơn đến giảng viên đã hỗ trợ trong quá trình thực hiện báo cáo này.

Appendix A:

Appendix B: VHDL Code

(đóng gói thành tệp nén và gửi kèm báo cáo)

Appendix C:

Compress and email to hungnvnu@gmail.com

List of Figures

[Hình 1. Ví dụ minh họa quá trình sắp xếp một dãy gồm 4 phần tử theo trật tự tăng dần. 6](#_Toc496109799)

[Hình 2. Giao diện ghép nối I/O của đơn vị Sorting Unit. 7](#_Toc496109800)

[Hình 3: FSMD. 8](#_Toc496109801)

[Hình 4: Datapath. 8](#_Toc496109802)

[Hình 5: FSM of controller. 8](#_Toc496109803)

[Hình 5: Block diagram of whole sorting unit. 9](#_Toc496109804)

List of Tables

[Bảng 1: Mô tả các tín hiệu vào ra. 7](#_Toc496109879)

References

1. https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1