

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KỸ THUẬT MÁY TÍNH

SINH VIÊN

TRẦN ĐĂNG HẬU 20521302

LÊ QUANG PHONG 20521744

BÁO CÁO MÔN ĐỒ ÁN 2

THỰC HIỆN MẠNG TÍCH CHẬP TRÊN FPGA

GIẢNG VIÊN HƯỚNG DẪN

TRƯƠNG VĂN CƯỜNG

TP. HỒ CHÍ MINH

LỜI CẢM ƠN

MỤC LỤC

Chương 1.	Tổng quan đề tài.	1
1.1.	Giới thiệu đề tài.	1
1.2.	Tổng quan đề tài	1
1.3.	Mục tiêu đề tài	2
Chương 2.	Mô hình thiết kế.	2
2.1.	Các thành phần chính của mô hình mạng tích chập	2
2.1.1.	Convolution Layer.....	2
2.1.2.	Pooling layer.....	2
2.1.3.	Fully Connected Layer	3
2.2.	Kỹ thuật lượng tử hóa.....	3
2.3.	Mô hình mạng tích chập.	3
Chương 3.	Hiện thực thiết kế.	4
3.1.	Thiết kế bộ đệm dữ liệu.....	4
3.2.	Thiết kế bộ convolution.....	5
3.2.1.	Khối cnn_controller.....	8
Chương 4.	Kiểm tra và đánh giá thiết kế.....	9
4.1.	Mô phỏng pre-synthesis khối conv.....	9
4.2.	Mô phỏng post-synthesis khối conv.	14
Chương 5.	Tổng kết.....	18

DANH MỤC HÌNH

Hình 2.1: Mô hình CNN với bộ dữ liệu MNIST	4
Hình 2.2: Các khối CNN IP đối với dữ liệu MNIST	4
Hình 3.1: Thiết kế của khối <i>cnn_buffer</i>	5
Hình 3.2: Thiết kế khối <i>conv</i> đi kèm <i>fifo</i>	5
Hình 3.3: Các thành phần trong khối <i>conv</i>	6
Hình 3.4: Sơ đồ khối <i>pe_conv_mac</i>	6
Hình 3.5: Sơ đồ kiến trúc của khối <i>pe_conv_mac_datapath</i>	7
Hình 3.6: Sơ đồ chuyển trạng thái	8
Hình 3.7: Ví dụ hình ảnh test với <i>pPadding</i> là 1	9
Hình 3.8: Minh họa việc dữ liệu đã đủ để nhân tích chập	9
Hình 4.1: Cấu hình các tham số cho việc mô phỏng	10
Hình 4.2: Pixel hình ảnh test có giá trị tăng dần.....	10
Hình 4.3: Các giai đoạn khi một khối tích chập thực hiện	11
Hình 4.4: Đối chiếu kết quả của việc nhân và cộng dồn	11
Hình 4.5: Giá trị của 32 kênh đầu ra sau khi chạy <i>pre-synthesis</i>	12
Hình 4.6: Giá trị cụ thể của từng kênh đầu ra.....	13
Hình 4.7: Số lượng tài nguyên sử dụng khi <i>post-synthesis</i>	14
Hình 4.8: Phần trăm tài nguyên sử dụng khi <i>post-synthesis</i>	14
Hình 4.9: Tài nguyên sử dụng sau khi <i>post-implementation</i>	15
Hình 4.10: Phần trăm tài nguyên sử dụng sau khi <i>post-implementation</i>	15
Hình 4.11: Thời gian setup và hold của mô hình.....	16
Hình 4.12: Chạy mô phỏng <i>post-implementation</i>	16
Hình 4.13: Đưa các pixel có giá trị tăng dần vào khối tích chập.....	16
Hình 4.14: Kiểm tra kết quả của khối <i>adder</i> sau khi nhân và cộng dồn	16
Hình 4.15: Giá trị của 32 kênh đầu ra sau khi chạy <i>post-implementation</i>	17
Hình 4.16: So sánh kết quả của 2 quá trình tại kênh đầu ra thứ 0.	18
Hình 4.17: So sánh kết quả của 2 quá trình tại kênh đầu ra thứ 30.	18

DANH MỤC BẢNG

DANH MỤC TỪ VIẾT TẮT

Chương 1. Tổng quan đề tài.

1.1. Giới thiệu đề tài.

Convolution Neural Network (CNN) là một trong những loại mạng nơ-ron ứng dụng trong xử lý ảnh và thị giác máy tính. CNN đã đạt được nhiều thành tựu lớn trong nhiều ứng dụng như nhận diện khuôn mặt, xe, động vật, phân loại hình ảnh và nhiều nhiệm vụ liên quan đến xử lý ảnh khác như lái xe tự động, nhận diện đối tượng, xử lý ảnh Y Sinh,... Hiện nay, CNN ngày càng trở nên đa dạng và phức tạp hơn, dẫn đến việc triển khai CNN trên CPU hay GPU trở nên giới hạn về tốc độ. Vậy nên, việc hiện thực mạng CNN trên FPGA (Field Programmable Gate Array) giúp việc khả năng xử lý song song làm cho việc tính toán nhanh hơn đồng thời tiêu thụ ít năng lượng. Nhưng song với đó việc triển khai CNN trên FPGA khó hơn so với việc triển khai trên CPU và GPU. Vì vậy, cần có kỹ thuật và kiến thức để có thể triển khai CNN trên FPGA một cách hiệu quả và đạt được hiệu suất tối ưu nhất.

1.2. Tổng quan đề tài

Trong đề tài này, nhóm tập trung nghiên cứu và thiết kế các lớp trong mô hình học sâu. Để tối ưu hóa tài nguyên và hiệu suất cho mô hình, nhóm áp dụng kỹ thuật lượng tử hóa và tận dụng việc xử lý song song của FPGA. Trong đề tài này nhóm tập trung vào thiết kế các khối Convolutional, Pooling, Fully Connected bằng kỹ thuật lượng tử hóa, khả năng xử lý song song và viết testbench để kiểm tra chức năng các khối đó.

Sau khi thiết kế các khối ở trên, nhóm tìm hiểu và thiết kế mạng mô hình học sâu cho bài toán phân loại chữ số viết tay trên bộ dữ liệu MNIST. Sau đó tiến hành huấn luyện, lượng tử hóa mô hình và trích xuất trọng số để nạp xuống thiết kế FPGA. Cuối cùng tiếp hành mô phỏng trên tập dữ liệu test, pre-synthesis, post-synthesis và nạp xuống kit FPGA.

1.3. Mục tiêu đề tài

Mục tiêu của đề tài là thiết kế được các khối sử dụng trong mô hình học sâu để có thể triển khai được mô hình CNN theo hướng học sâu xuống kit FPGA.

Chương 2. Mô hình thiết kế.

2.1. Các thành phần chính của mô hình mạng tích chập

2.1.1. Convolution Layer

Là một phép trượt qua các điểm trong tất cả điểm của hình ảnh để tìm ra các đặc trưng của hình ảnh bằng cách sử dụng các bộ lọc (kernel). Các bộ lọc là một ma trận nhỏ thường có kích thước là 3x3 hoặc 5x5 được trượt qua các dữ liệu đầu vào để thực hiện phép toán convolution. Một mạng CNN thường có nhiều lớp convolution. Đầu ra của các lớp convolution được gọi là feature map, nó đại diện cho các đặc trưng đã học. Các đặc trưng thường là những đặc trưng về cạnh, góc, hoặc có các mẫu phức tạp hơn.

Việc thực hiện phép toán convolution với bộ lọc kernel sẽ làm giảm kích thước nên việc thêm các giá trị 0 vào xung quanh dữ liệu đầu vào để duy trì kích thước của đầu ra trước khi thực hiện phép convolution được gọi là padding. Việc thêm bao nhiêu lớp padding phụ thuộc vào kích thước của kernel.

2.1.2. Pooling layer

Hàm pooling layer làm giảm kích thước của feature map bằng cách chọn các giá trị lớn nhất (max pooling), giá trị trung bình (average pooling) trong các vùng nhất định của feature map.

Mục tiêu của pooling layer là làm giảm các tham số trong mô hình đồng thời cũng làm giảm số lượng tính toán trong mô hình CNN. Lớp pooling đóng vai trò trong việc cải thiện hiệu suất và hiệu quả của mô hình.

2.1.3. Fully Connected Layer

Lớp Fully Connected là một thành phần quan trọng của mô hình CNN. Lớp này thường ở cuối mô hình để chuyển đổi feature map thành đầu ra cuối cùng phù hợp với nhiệm vụ của mô hình CNN.

2.2. Kỹ thuật lượng tử hóa

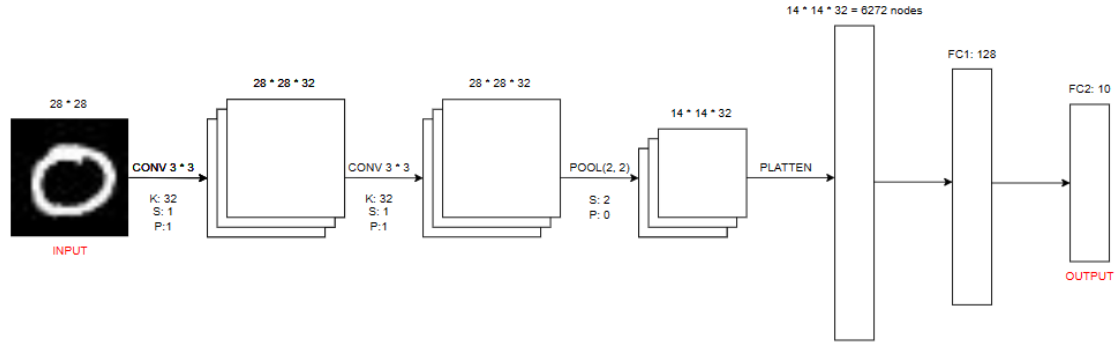
Lượng tử hóa là kỹ thuật thực hiện tính toán và lưu trữ các đặc trưng bằng kiểu dữ liệu có độ chính xác thấp hơn so với độ chính xác của kiểu dữ liệu số chấm động. Một mô hình được lượng tử hóa thực hiện một số hoặc tất cả các phép tính trên các đặc trưng với giá trị nguyên thay vì giá trị số chấm động. Kỹ thuật này đặc biệt hữu ích trong quá trình xử lý, vì nó giảm thiểu chi phí tính toán rất nhiều mà không làm giảm độ chính xác của kết quả quá nhiều.

Tuy nhiên bản chất của việc lượng tử hóa là làm giảm kích thước mô hình bằng cách chuyển trọng số từ số chấm động thành số nguyên sử dụng ít bit biểu diễn hơn. Trong lúc thực hiện các phép tính ta vẫn phải thực hiện việc chuyển đổi số nguyên thành số chấm động để thực hiện.

Như vậy, việc áp dụng kỹ thuật lượng tử hóa là việc dequantize và quantize dữ liệu liên tục giữa các lớp. Dữ liệu đầu vào được dequantize thành 32bit, qua hàm kích hoạt (Sigmoid, Relu,...) và sau đó được quantize lại thành 8 bit và đưa tới lớp tiếp theo.

2.3. Mô hình mạng tích chập.

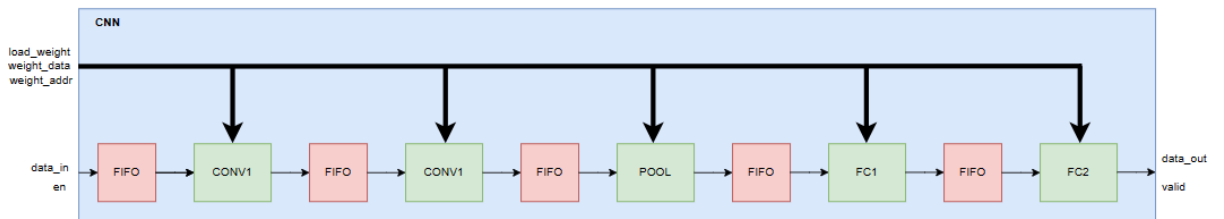
Với các thành phần chính của mô hình mạng tích chập vừa nêu trên, nhóm đưa ra thiết kế một mạng tích chập thực hiện trên bộ dữ liệu MNIST.



Hình 2.1: Mô hình CNN với bộ dữ liệu MNIST

Với mô hình CNN ở trên, thiết kế của khối CNN cũng có các khối như conv1, conv2, pool, fc_1 và fc_2. Trước mỗi khối trong thiết kế đều được trang bị một bộ fifo để có thể truyền dữ liệu và quản lý dữ liệu đầu ra của mỗi lớp convolution hay fully connected giúp tránh trường hợp các dữ liệu truyền không chính xác.

Mỗi khối conv hay fc trong thiết kế đều có một địa chỉ để nạp trọng số riêng, địa chỉ này phụ thuộc vào số lượng trọng số của tầng đó và địa chỉ của tầng trước.



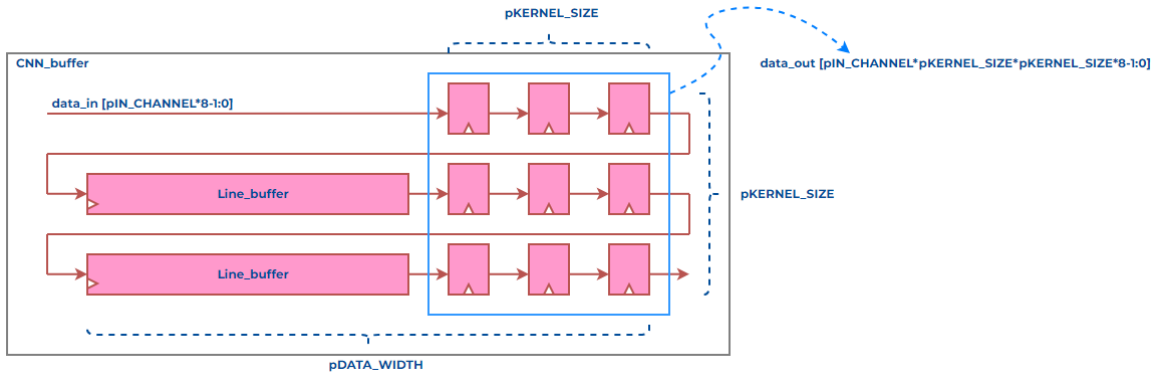
Hình 2.2: Các khối CNN IP đối với dữ liệu MNIST

Chương 3. Hiện thực thiết kế.

3.1. Thiết kế bộ đệm dữ liệu.

Trước khi thực hiện tích chập thì dữ liệu pixel phải đảm bảo đủ số lượng và có thứ tự chính xác, dựa trên hình ảnh nhân tích chập thường thấy như hình 3.8, nhóm thiết kế một bộ đệm có hình dạng mô phỏng tương tự như bên dưới. Cấu tạo gồm các D Flip Flop nối tiếp nhau theo dạng shift register, ở đây số lượng DFF có thể thay đổi

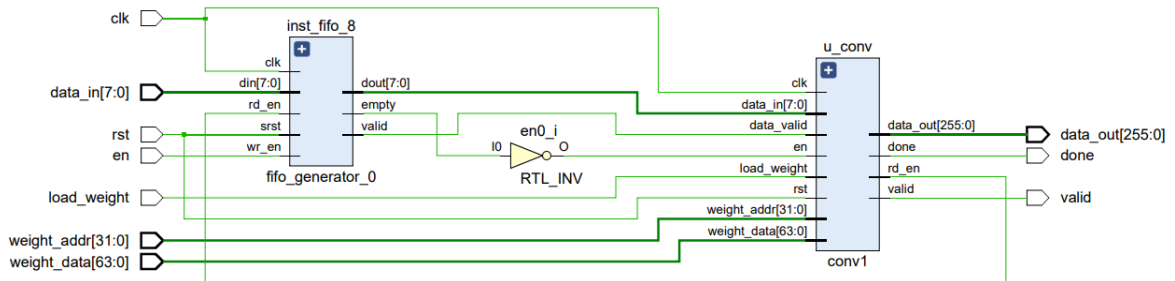
dựa trên các tham số $pKERNEL_SIZE$ và $pDATA_WIDTH$. Chính vì vậy việc sử dụng kernel có kích thước 3x3 hay 5x5,... có thể được đáp ứng bởi thiết kế này.



Hình 3.1: Thiết kế của khối *cnn_buffer*

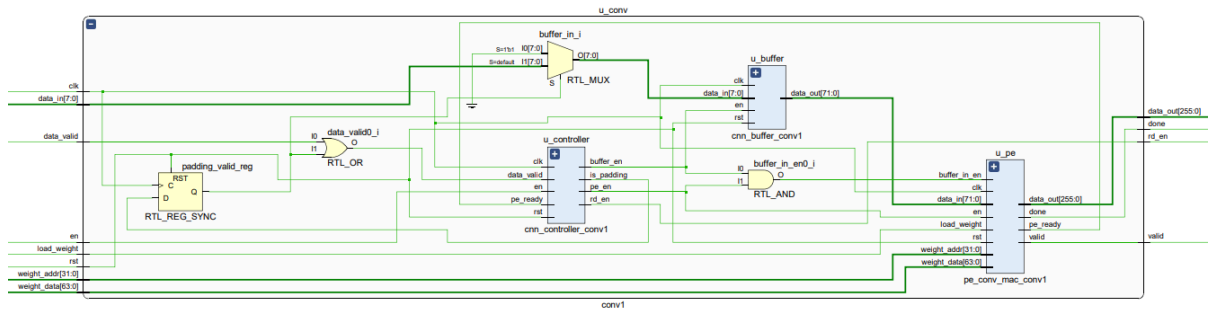
3.2. Thiết kế bộ convolution.

Như đã trình bày ở trên, cần một khối fifo đứng trước để đệm dữ liệu vào khối conv để tránh thất thoát dữ liệu. Khi tín hiệu $empty = 0$ báo hiệu rằng khối tích chập đã có dữ liệu và có thể bắt đầu thực hiện phép tính. Tín hiệu rd_en và $valid$ của conv dùng để báo cho fifo biết là khối conv có thể đọc dữ liệu của khối fifo truyền vào.



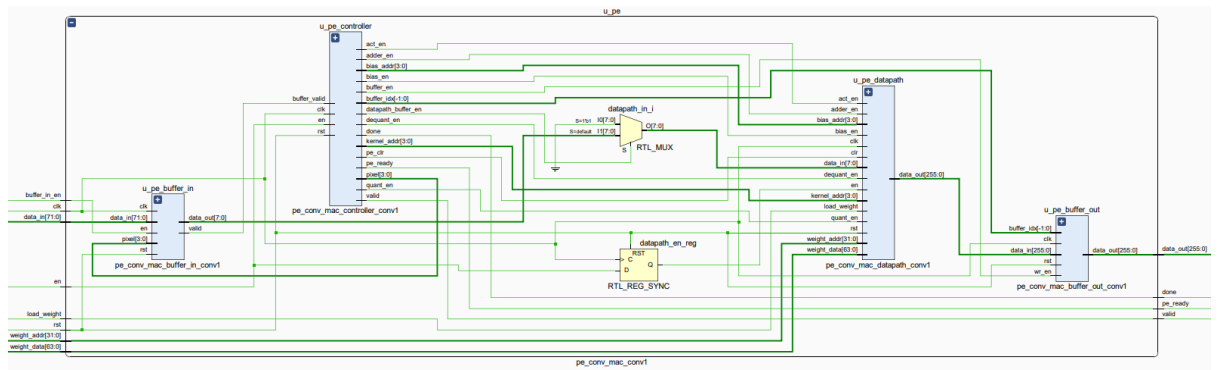
Hình 3.2: Thiết kế khối conv đi kèm fifo

Thành phần của khối convolution bao gồm một khối **buffer_conv** để đệm dữ liệu đầu vào, một khối thực hiện phép tích chập (**pe_conv_mac**) và một khối **controller** (**controller_conv**) để điều khiển việc hoạt động của 2 khối trên.

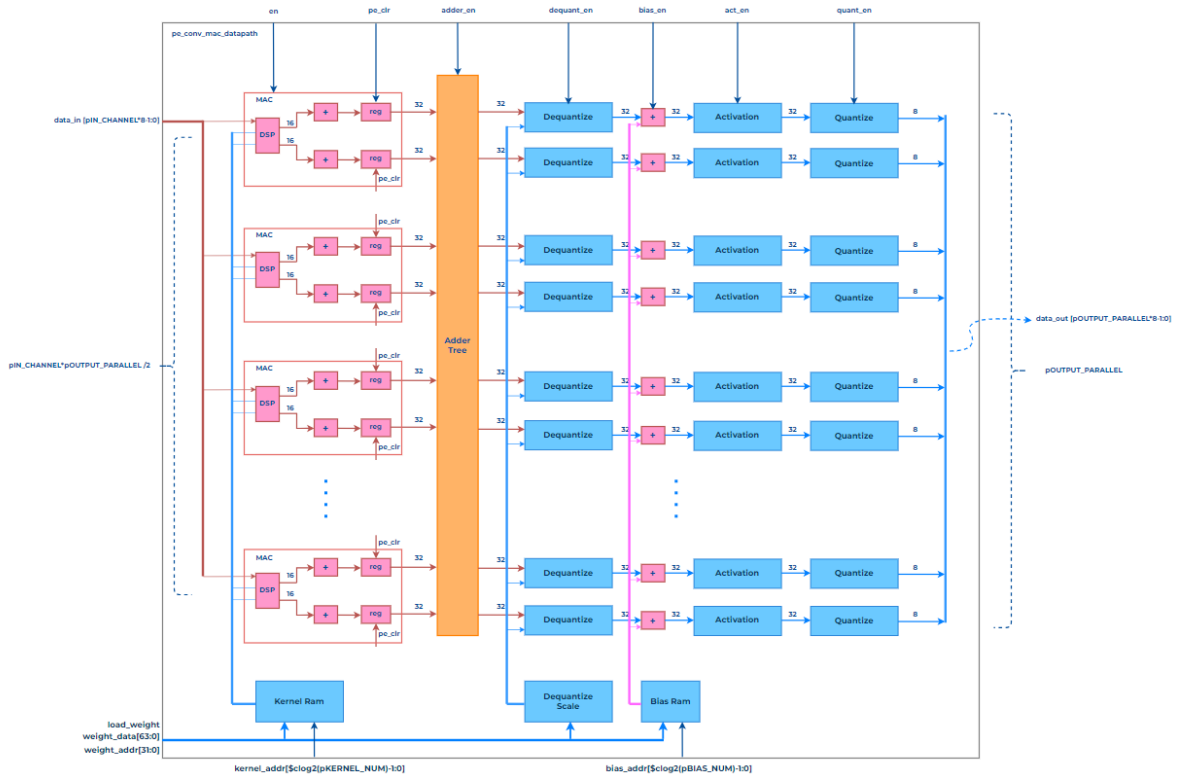


Hình 3.3: Các thành phần trong khối conv

Khối **pe_conv_mac** bao gồm khối **bufer_in**, **buffer_out**, **pe_conv_mac_controller**, **pe_conv_mac_datapath**, đây là nơi thực hiện phép tích chập. Khi tín hiệu en tích cực, khối đồng thời xử lý, tín hiệu valid, pe_ready ở mức cao khi khối tích chập thực hiện tính toán xong và đẩy kết quả ra qua tín hiệu data_out.



Hình 3.4: Sơ đồ khối pe_conv_mac



Hình 3.5: Sơ đồ kiến trúc của khối *pe_conv_mac_datapath*

Trước khi tính toán, toàn bộ trọng số kernel, bias, dequantize scale phải được lưu trữ. Ở đây chúng sẽ được nhóm lưu trữ ở trong các khối Block RAM và LUT, ngoài ra còn có thể lưu trữ trong ULTRA RAM nếu tài nguyên của board sử dụng có sẵn.

Khối *pe_conv_mac_datapath* là nơi xử lý toàn bộ quá trình tính toán, từ việc nhân, cộng dồn cho tới lượng tử hóa về lại 8bit. Đầu ra của khối *cnn_buffer* (có thể là 9 pixel hoặc nhiều hơn, tùy thuộc vào kích thước kernel mà chúng ta sử dụng), sẽ được đưa qua khối *buffer_in*, khối này có nhiệm vụ chọn từng pixel theo đúng thứ tự để đưa vào các DSP thực hiện nhân song song. Số DSP cần sử dụng sẽ được xác định dựa trên số *pIN_CHANNEL* và *pOUTPUT_PARALLEL*, ví dụ $pIN_CHANNEL = 1$, $pOUTPUT_PARALLEL = 32$ thì số DSP cần sử dụng là: $(1 \times 32) / 2 = 16$ tại vì mỗi DSP sẽ thực hiện 2 phép nhân đồng thời và cho ra 2 kết quả. Tương tự số bộ adder tree, dequantize, bias, activation, quantize sẽ được xác định dựa trên *pOUTPUT_PARALLEL*. Ngoài ra cần lưu ý về số bit đầu vào và đầu ra của mỗi khối, cũng như dấu của chúng, việc này đã được chú thích rõ trên hình

3.5. Từng khối trong sơ đồ trên thực hiện tính toán trong những chu kỳ nhất định khác nhau, tuy nhiên để chúng bắt đầu hoạt động thì các tín hiệu điều khiển như: `en`, `pe_clr`, `adder_en`, `dequant_en`, `bias_en`, `act_en`, `quant_en` phải tích cực. Những tín hiệu này sẽ được điều khiển bởi khối `pe_conv_mac_controller`, khối này cấu tạo gồm một chuỗi thanh ghi nối tiếp nhau theo dạng shift register, số thanh ghi giữa hai tín hiệu phụ thuộc vào khối đó cần bao nhiêu chu kỳ để tính toán. Ở đây khối DSP sẽ cần 5 chu kỳ để tính toán, khối cộng và lưu cần 1 chu kỳ, khối adder tree sẽ phụ thuộc vào `pINPUT_NUM` để xác định, khối dequantize sẽ cần 9 chu kỳ, khối bias cần 1 chu kỳ, khối activation cần 4 chu kỳ, khối quantize cần 1 chu kỳ.

3.2.1. Khối `cnn_controller`



Hình 3.6: Sơ đồ chuyển trạng thái

Về cơ bản, sẽ có 2 trạng thái hoạt động chính của khối tích chập:

+ Trạng thái IDLE: ở trạng thái này, các giá trị padding (nếu có) và giá trị pixel đọc từ fifo sẽ được đưa vào khối `cnn_buffer` cho đến khi đủ dữ liệu để phục vụ cho việc trượt kernel nhân tích chập (như được minh họa trong hình 3.8). Việc này được xác định dựa trên hai bộ đếm `col_cntr_r` và `row_cntr_r`.

+ Trạng thái COMPUTATION: sau khi có đủ dữ liệu thì tín hiệu `pe_en` bật lên 1, cho phép toàn bộ khối `pe_conv_mac` hoạt động.

0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	0
0	8	9	10	11	12	13	14	0
0	15	16	17	18	19	20	21	0
0	22	23	24	25	26	27	28	0
0	29	30	31	32	33	34	35	0
0	36	37	38	39	40	41	42	0
0	43	44	45	46	47	48	49	0
0	0	0	0	0	0	0	0	0

Hình 3.7: Ví dụ hình ảnh test với $p\text{PADDING}$ là 1

0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	0
0	8	9	10	11	12	13	14	0
0	15	16	17	18	19	20	21	0
0	22	23	24	25	26	27	28	0
0	29	30	31	32	33	34	35	0
0	36	37	38	39	40	41	42	0
0	43	44	45	46	47	48	49	0
0	0	0	0	0	0	0	0	0

Hình 3.8: Minh họa việc dữ liệu đã đủ để nhân tích chập

Chương 4. Kiểm tra và đánh giá thiết kế.

4.1. Mô phỏng pre-synthesis khối conv.

Nhóm sẽ mô phỏng thiết kế trong trường hợp đầu tiên như hình 3.7 với các cấu hình tham số như sau:

```

localparam pPERIOD = 4;

localparam pData_WIDTH    = 8;

localparam pINPUT_WIDTH   = 7;
localparam pINPUT_HEIGHT  = 7;

localparam pWEIGHT_DATA_WIDTH = 64;
localparam pWEIGHT_BASE_ADDR = 32'h4000_0000;

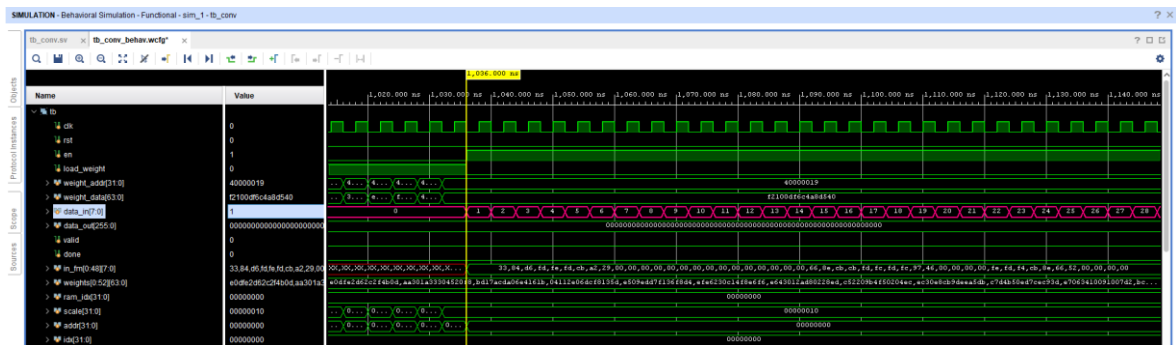
localparam pIN_CHANNEL    = 1;
localparam pOUT_CHANNEL   = 32;

localparam pKERNEL_SIZE  = 3;
localparam pPADDING       = 1;
localparam pSTRIDE        = 1;

localparam pACTIVATION    = "sigmoid";
localparam pOUTPUT_PARALLEL = 32;

```

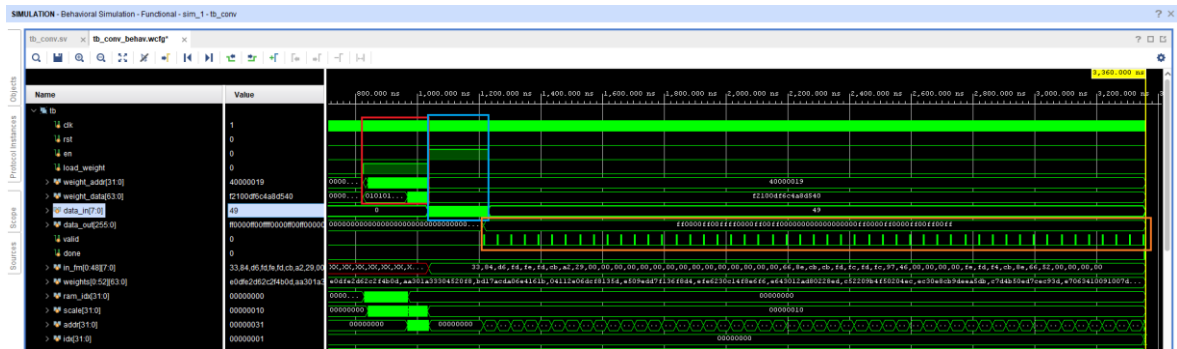
Hình 4.1: Cấu hình các tham số cho việc mô phỏng



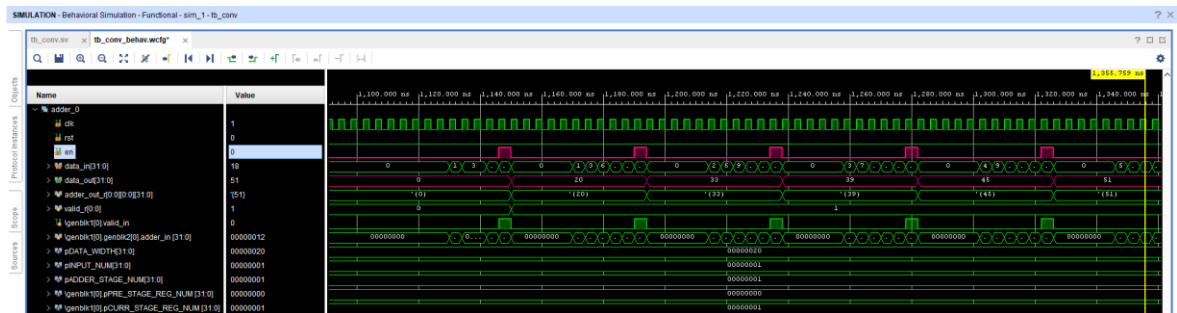
Hình 4.2: Pixel hình ảnh test có giá trị tăng dần

Mỗi lần khởi tích chập hoạt động sẽ có các giai đoạn như:

- Nạp các trọng số vào các khối lưu trữ (Block RAM, LUT,...) (mô tả bằng hình chữ nhật đỏ).
- Ghi các pixel vào fifo (mô tả bằng hình chữ nhật xanh).
- Chờ đợi mỗi lần valid lên 1 thì data_out của khối tích chập là hợp lệ. (mô tả bằng hình chữ nhật cam).



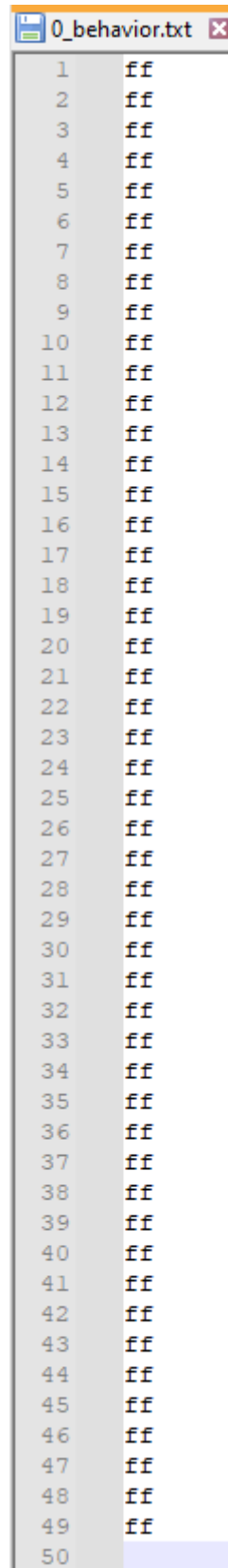
Hình 4.3: Các giai đoạn khi một khối tích chập thực hiện



Hình 4.4: Đối chiếu kết quả của việc nhân và cộng dồn

AN1 > Project > cnn-fpga-main_1 > design > software > results > fpga > conv > 15					Se
Name	^	Date modified	Type	Size	
0_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
1_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
2_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
3_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
4_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
5_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
6_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
7_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
8_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
9_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
10_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
11_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
12_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
13_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
14_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
15_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
16_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
17_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
18_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
19_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
20_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
21_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
22_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
23_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
24_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
25_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
26_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
27_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
28_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
29_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
30_behavior.txt		16/01/2024 02:14	TXT File	1 KB	
31_behavior.txt		16/01/2024 02:14	TXT File	1 KB	

Hình 4.5: Giá trị của 32 kênh đầu ra sau khi chạy pre-synthesis



1	ff
2	ff
3	ff
4	ff
5	ff
6	ff
7	ff
8	ff
9	ff
10	ff
11	ff
12	ff
13	ff
14	ff
15	ff
16	ff
17	ff
18	ff
19	ff
20	ff
21	ff
22	ff
23	ff
24	ff
25	ff
26	ff
27	ff
28	ff
29	ff
30	ff
31	ff
32	ff
33	ff
34	ff
35	ff
36	ff
37	ff
38	ff
39	ff
40	ff
41	ff
42	ff
43	ff
44	ff
45	ff
46	ff
47	ff
48	ff
49	ff
50	

Hình 4.6: Giá trị cụ thể của từng kênh đầu ra

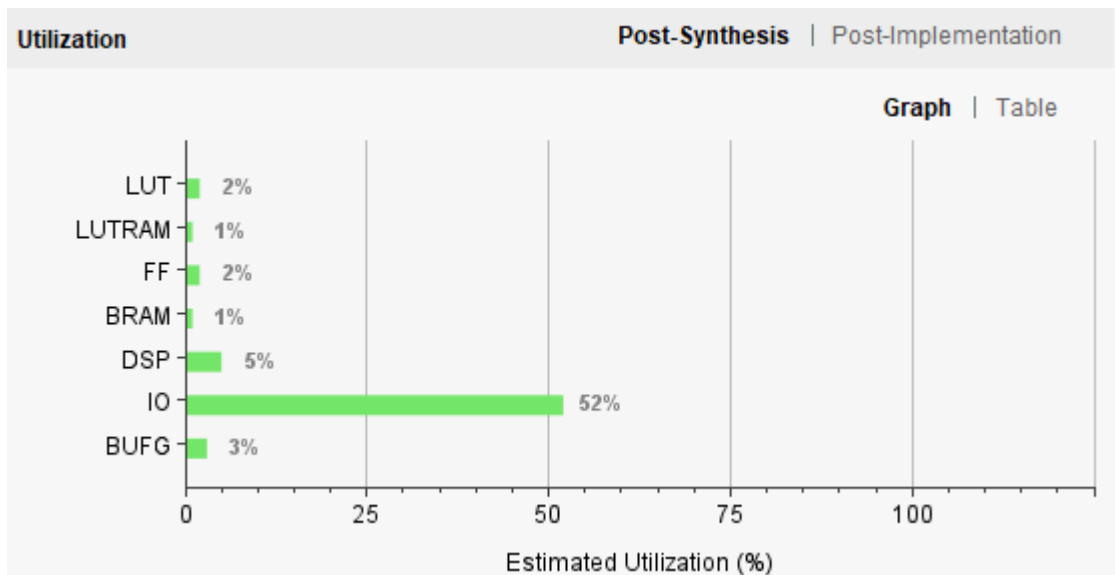
4.2. Mô phỏng post-synthesis khối conv.

Về cấu hình tham số và pixel hình ảnh test giống như quá trình pre-synthesis, ở quá trình post-synthesis, nhóm sẽ tiến hành tổng hợp, chạy implementation và sau đó mô phỏng lại chức năng của mạch xem có đúng với mong đợi hay không.

Sau khi tổng hợp, số lượng tài nguyên sử dụng trên board Virtex7-VC707 như sau:

Utilization				Post-Synthesis	Post-Implementation
				Graph	Table
Resource	Estimation	Available	Utilization %		
LUT	6732	303600	2.22		
LUTRAM	818	130800	0.63		
FF	13595	607200	2.24		
BRAM	0.50	1030	0.05		
DSP	144	2800	5.14		
IO	366	700	52.29		
BUFG	1	32	3.13		

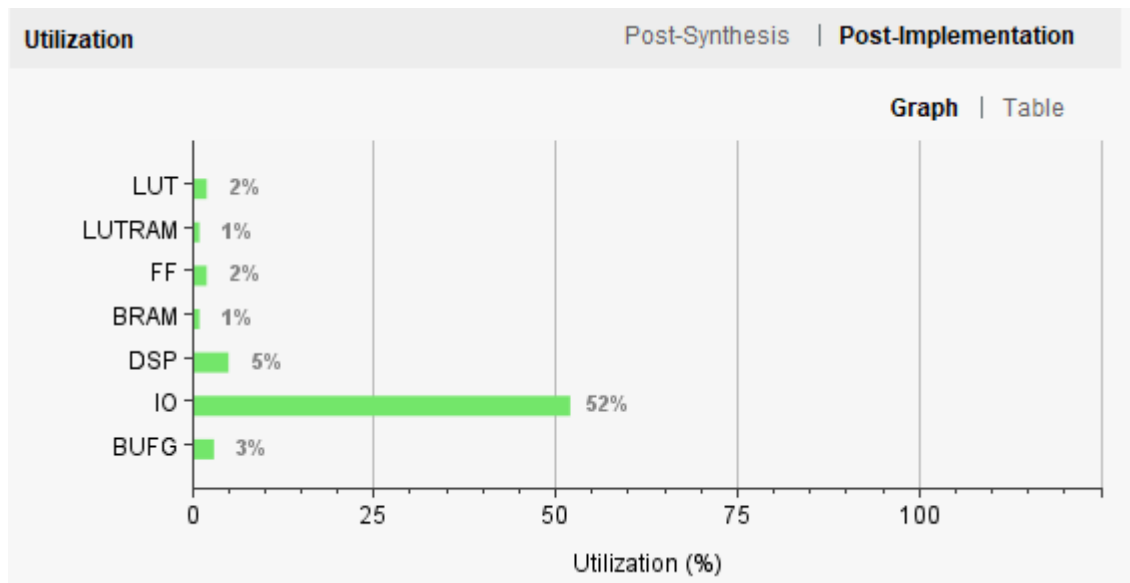
Hình 4.7: Số lượng tài nguyên sử dụng khi post-synthesis



Hình 4.8: Phần trăm tài nguyên sử dụng khi post-synthesis

Utilization				Post-Synthesis	Post-Implementation
				Graph	Table
Resource	Utilization	Available	Utilization %		
LUT	6632	303600	2.18		
LUTRAM	817	130800	0.62		
FF	13596	607200	2.24		
BRAM	0.50	1030	0.05		
DSP	144	2800	5.14		
IO	366	700	52.29		
BUFG	1	32	3.13		

Hình 4.9: Tài nguyên sử dụng sau khi post-implementation



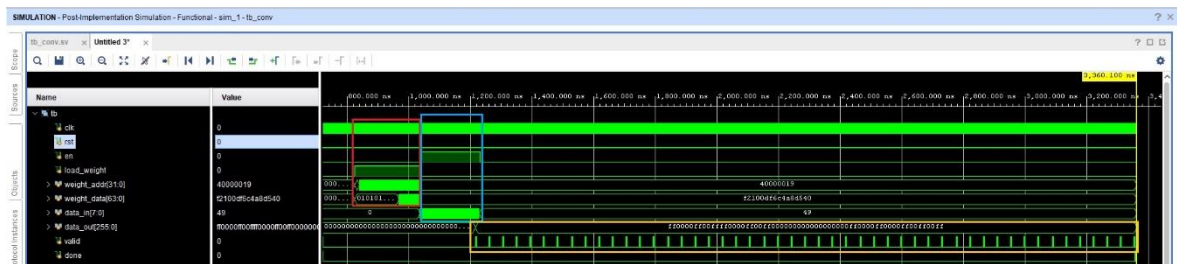
Hình 4.10: Phần trăm tài nguyên sử dụng sau khi post-implementation

Về vấn đề Timing: Không có violation nào xuất hiện, hiện chu kì đang được chỉ định là 4ns.

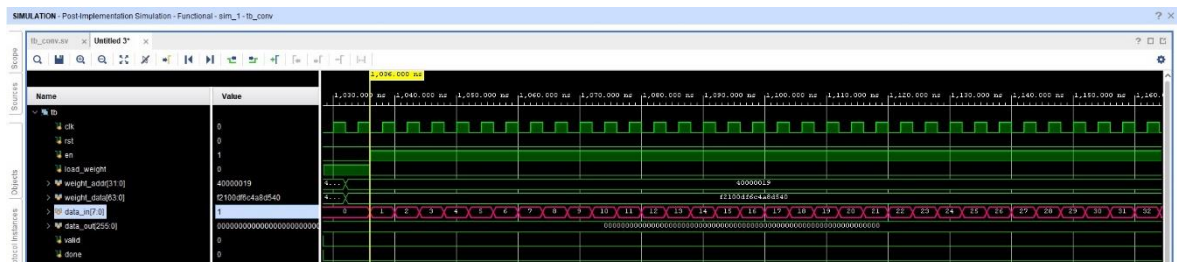
Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 0.127 ns	Worst Hold Slack (WHS): 0.059 ns	Worst Pulse Width Slack (WPWS): 1.228 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 32164	Total Number of Endpoints: 32164	Total Number of Endpoints: 15265

Hình 4.11: Thời gian setup và hold của mô hình

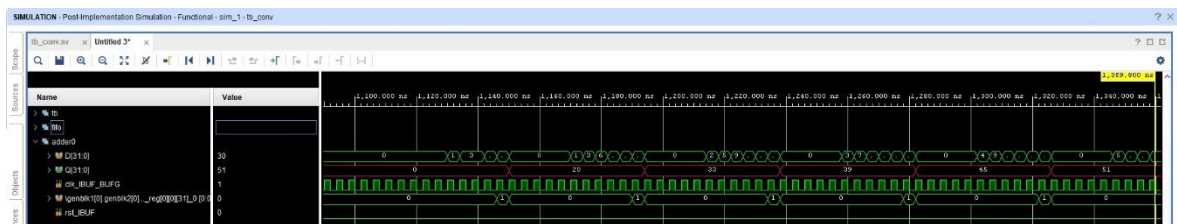
Sau khi đã hoàn thành các bước bên trên, nhóm tiến hành chạy post-implementation functional simulation.



































Hình 4.12: Chạy mô phỏng post-implementation



Hình 4.13: Đưa các pixel có giá trị tăng dần vào khối tích chập

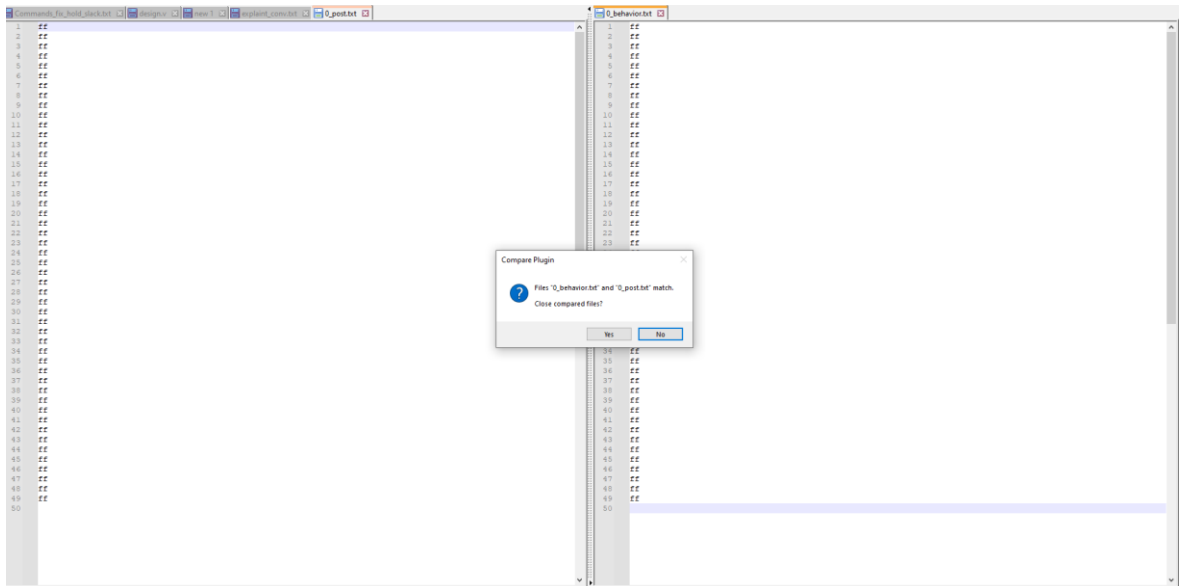


Hình 4.14: Kiểm tra kết quả của khối adder sau khi nhân và cộng dồn

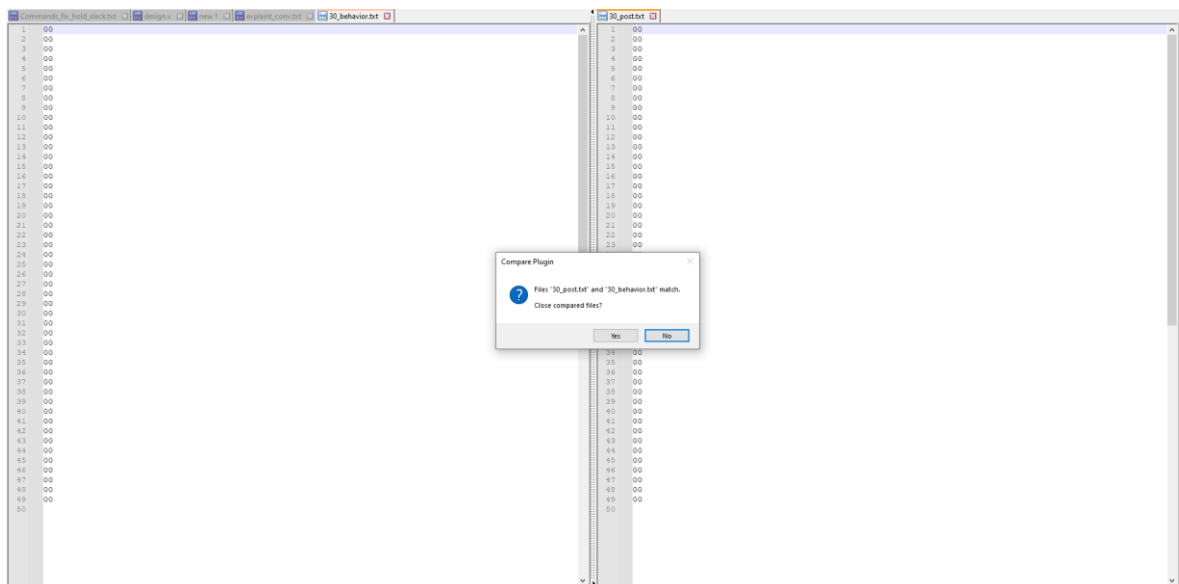
Name	Date modified	Type	Size
 0_post.txt	16/01/2024 03:34	TXT File	1 KB
 1_post.txt	16/01/2024 03:34	TXT File	1 KB
 2_post.txt	16/01/2024 03:34	TXT File	1 KB
 3_post.txt	16/01/2024 03:34	TXT File	1 KB
 4_post.txt	16/01/2024 03:34	TXT File	1 KB
 5_post.txt	16/01/2024 03:34	TXT File	1 KB
 6_post.txt	16/01/2024 03:34	TXT File	1 KB
 7_post.txt	16/01/2024 03:34	TXT File	1 KB
 8_post.txt	16/01/2024 03:34	TXT File	1 KB
 9_post.txt	16/01/2024 03:34	TXT File	1 KB
 10_post.txt	16/01/2024 03:34	TXT File	1 KB
 11_post.txt	16/01/2024 03:34	TXT File	1 KB
 12_post.txt	16/01/2024 03:34	TXT File	1 KB
 13_post.txt	16/01/2024 03:34	TXT File	1 KB
 14_post.txt	16/01/2024 03:34	TXT File	1 KB
 15_post.txt	16/01/2024 03:34	TXT File	1 KB
 16_post.txt	16/01/2024 03:34	TXT File	1 KB
 17_post.txt	16/01/2024 03:34	TXT File	1 KB
 18_post.txt	16/01/2024 03:34	TXT File	1 KB
 19_post.txt	16/01/2024 03:34	TXT File	1 KB
 20_post.txt	16/01/2024 03:34	TXT File	1 KB
 21_post.txt	16/01/2024 03:34	TXT File	1 KB
 22_post.txt	16/01/2024 03:34	TXT File	1 KB
 23_post.txt	16/01/2024 03:34	TXT File	1 KB
 24_post.txt	16/01/2024 03:34	TXT File	1 KB
 25_post.txt	16/01/2024 03:34	TXT File	1 KB
 26_post.txt	16/01/2024 03:34	TXT File	1 KB
 27_post.txt	16/01/2024 03:34	TXT File	1 KB
 28_post.txt	16/01/2024 03:34	TXT File	1 KB
 29_post.txt	16/01/2024 03:34	TXT File	1 KB
 30_post.txt	16/01/2024 03:34	TXT File	1 KB
 31_post.txt	16/01/2024 03:34	TXT File	1 KB

Hình 4.15: Giá trị của 32 kênh đầu ra sau khi chạy post-implementation

Tiếp theo, tiếp hành so sánh kết quả từng kênh đầu ra của quá trình chạy pre-synthesis và quá trình post-implementation.



Hình 4.16: So sánh kết quả của 2 quá trình tại kênh đầu ra thứ 0.



Hình 4.17: So sánh kết quả của 2 quá trình tại kênh đầu ra thứ 30.

Chương 5. Tổng kết.

Trong đề tài này, nhóm đã thiết kế được các khối tích chập, hàm activation sigmoid và kiểm tra thiết kế bằng cách hiện thực mạng mô hình học sâu phân loại chữ số viết tay trên tập dữ liệu MNIST.

- Thiết kế được khối tích chập hoạt động được với tần số 250Mhz, có các tham số có thể cấu hình để thay đổi linh hoạt, đáp ứng cho những thiết kế khác nhau

TÀI LIỆU THAM KHẢO

Theo chuẩn IEEE