ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KỸ THUẬT MÁY TÍNH

BÁO CÁO ĐỒ ÁN MÔN HỌC

MÔN CHUYÊN ĐỀ THIẾT KẾ VI MẠCH 1

Đề tài:

HIỆN THỰC MÔ HÌNH MẠNG NƠ RON TÍCH CHẬP BẰNG NGÔN NGỮ PHẦN CỨNG VERILOG HDL

____INCEPTION RESNET V2____

GVHD: Trương Văn Cương

Nhóm sinh viên thực hiện:

1. Trần Cao Khải MSSV: 18520877
2. Nguyễn Hoàng Nghĩa MSSV: 18521143

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……, ngày…….....tháng……năm 2021

Người nhận xét

(Ký tên và ghi rõ họ tên)

LỜI MỞ ĐẦU

Thế giới hiện đang trong giai đoạn phát triển về công nghệ rất mạnh mẽ. Với cuộc cách mạng 4.0, các thiết bị điện tử đã và đang trở nên hiện đại hơn rất nhiều. Những chiếc máy tính không còn là những công cụ tính toán đơn thuần, chúng đã trở thành công cụ hỗ trợ con người rất hữu ích. Để có thể giúp con người hoàn thành những công việc mà từ trước đến nay chỉ có con người mới làm được, những chiếc máy tính cần phải nhìn nhận sự vật, hiện tượng như con người, suy nghĩ về sự việc như con người. Một trong những nghiên cứu để ứng dụng máy tính trong những công việc mà chỉ có con người mới có thể làm đó chính là trí tuệ nhân tạo (AI – Artificial Intelligence).

Phần không thể thiếu khi nhắc đến trí tuệ nhân tạo là thị giác máy tính. Và những mô hình mạng nơ ron tích chập (CNN/ConvNet – Convolutional Neural Network) sẽ giúp cho máy tính có thể phân tích được những hình ảnh, xử lý những thông tin có trên hình ảnh đó, dần dần tiệm cận đến khả năng xử lý hình ảnh của não bộ con người.

Mô hình nhóm chúng em lựa chọn cho đồ án môn học này là: Inception Resnet V2.

Sau đây sẽ là các chương về quá trình nghiên cứu, thực nghiệm và tổng kết của đề tài này:

- Chương 1: Giới Thiệu Chung
- Chương 2: Nội Dung Chính
- Chương 3: Tổng Kết

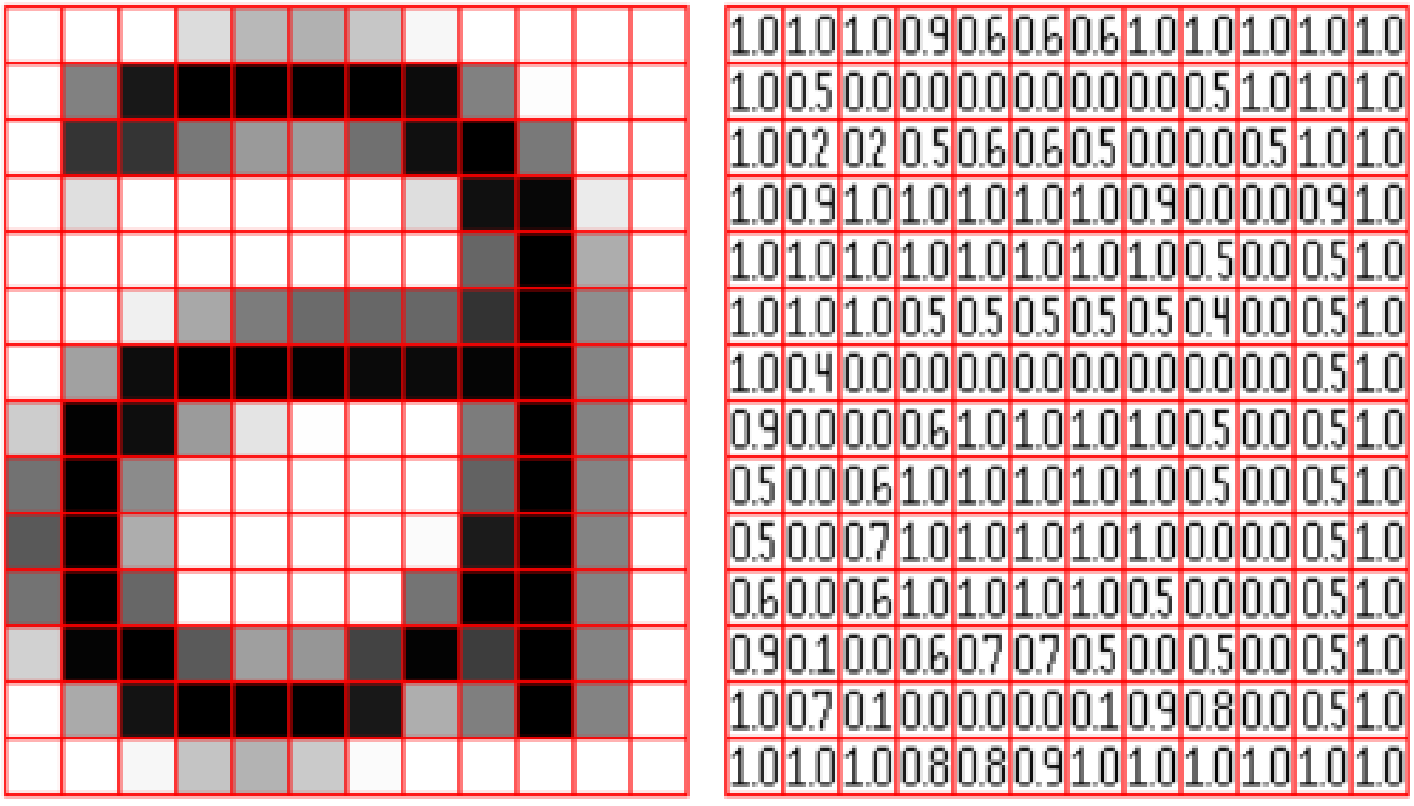
MỤC LỤC

Chương 1: Giới Thiệu Chung

1.1 Các định nghĩa về mạng nơ ron tích chập:

1.1.1 Định nghĩa mô hình mạng nơ ron tích chập (CNN):

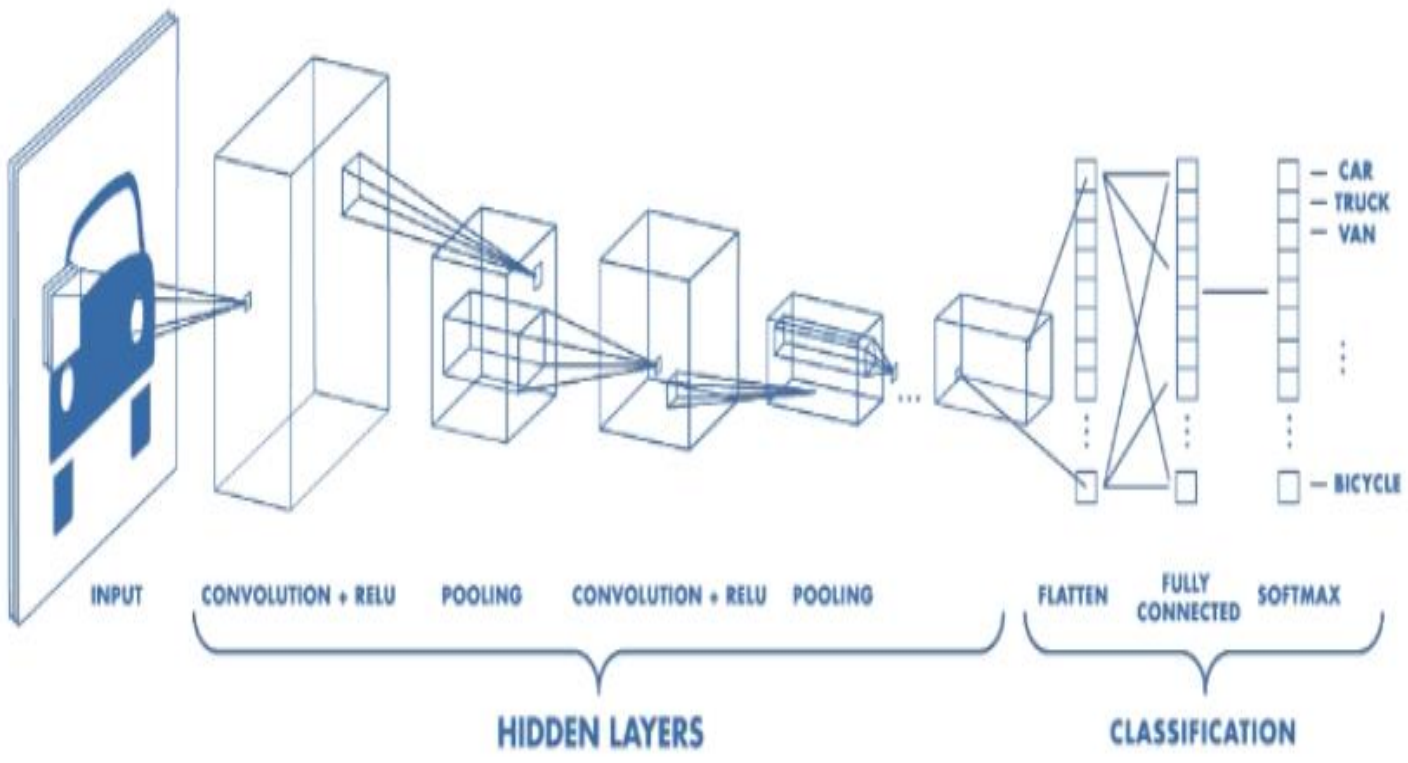
Mạng nơ ron tích chập, được viết tắt là: CNN hoặc ConvNet, là một lớp của mạng nơ ron được chuyên dụng trong lĩnh vực xử lý dữ liệu có liên kết dạng lưới, ví dụ như hình ảnh. Một bức ảnh kỹ thuật số thực chất là dạng biểu diễn số nhị phân của dữ liệu trực quan. Bức ảnh đó chứa các chuỗi với nhiều điểm ảnh (pixel) được sắp xếp ở dạng lưới, và trong các chuỗi đó là các điểm ảnh với giá trị của chúng sẽ tương ứng với độ sáng, màu sắc của điểm ảnh đó.



Hình 1.1.1.1 Biểu diễn dạng lưới các điểm ảnh của một hình ảnh.

Não bộ của con người xử lý lượng lớn dữ liệu ngay khi chúng ta nhìn thấy một hình ảnh nào đó. Mỗi nơ ron của não bộ hoạt động trong khả năng tiếp nhận của chúng và liên kết với các nơ ron khác để có thể hiểu được toàn bộ bức ảnh đó. Dựa trên cơ chế đó, các nơ ron trong mô hình mạng nơ ron tích chập sẽ có hoạt động tương tự với các nơ ron của não bộ con người. Trong mô hình CNN, sẽ có những lớp được sắp xếp theo thứ tự nhất định để giúp mô hình đó nhận biết được những khuôn mẫu đơn giản trước (đường thẳng, đường cong, v.v), sau đó sẽ là những khuôn mẫu phức tạp hơn (khuôn mặt, vật thể, v.v). Bằng cách áp dụng mô hình CNN, ta có thể giúp máy tính có được khả năng thị giác gần giống với con người.

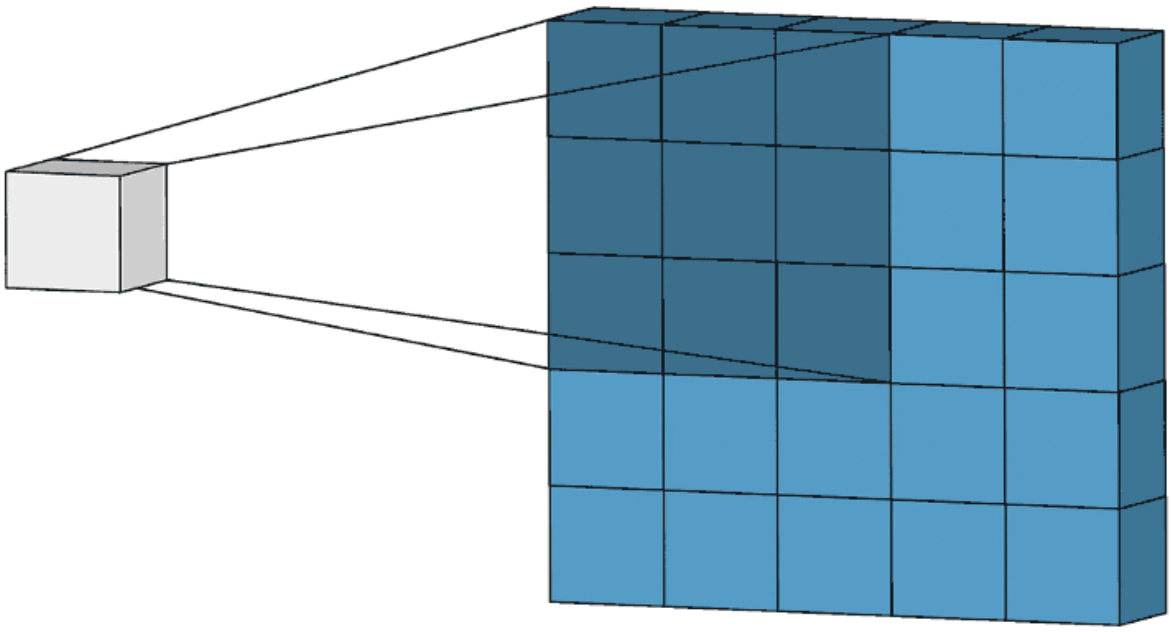
Một mô hình CNN thường sẽ bao gồm 3 lớp: lớp tích chập (convolutional layer), lớp pooling, lớp kết nối (fully connected layer).



Hình 1.1.1.2 Kiến trúc tổng quan của mô hình CNN.

1.1.2 Định nghĩa lớp tích chập (Convolutional Layer):

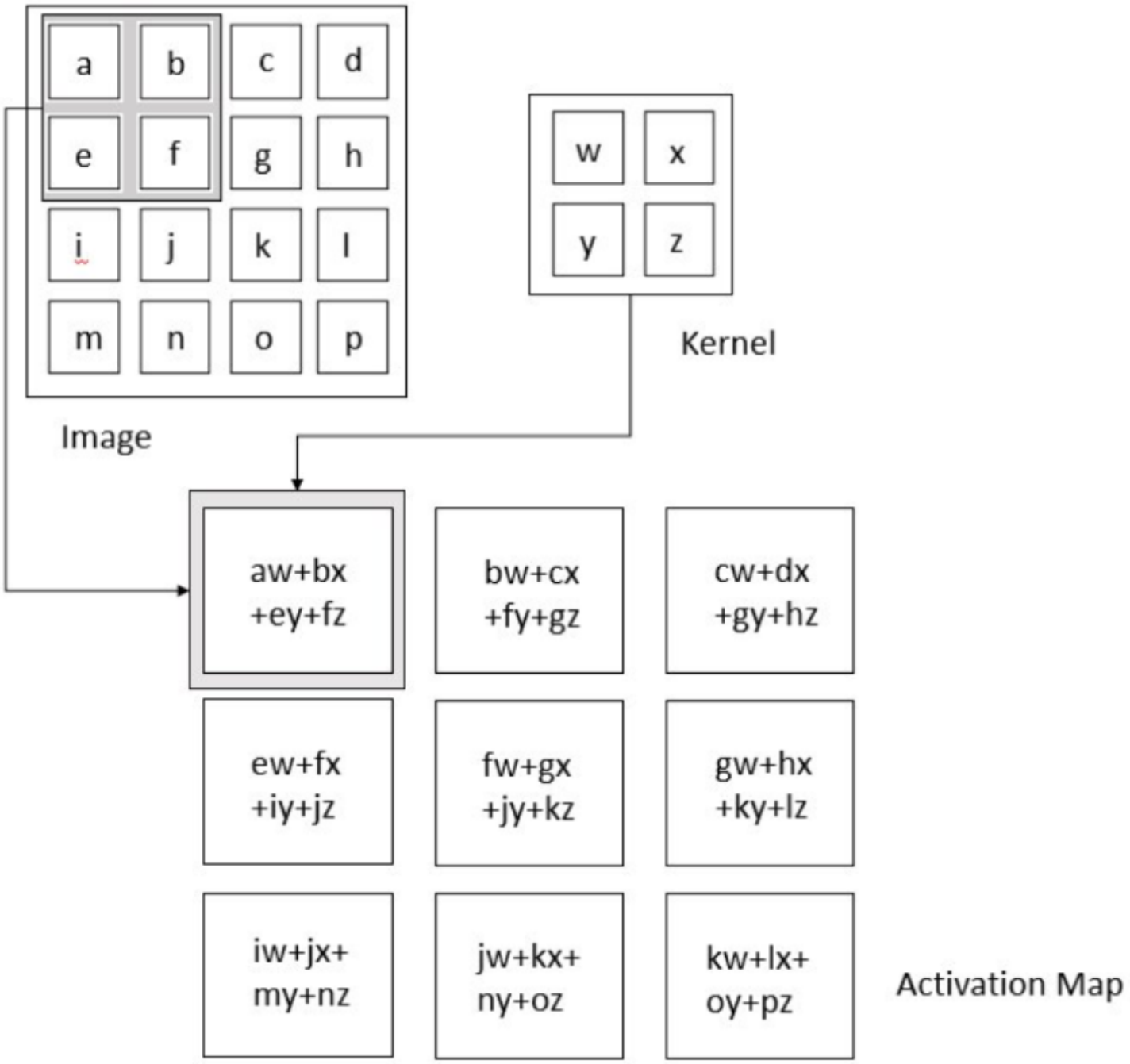
Lớp tích chập đóng vai trò chính trong một mô hình CNN, đây là lớp sẽ thực hiện hầu hết các phép tính toán khi xử lý dữ liệu đầu vào của mô hình đó. Lớp này sẽ thực hiện phép tích vô hướng giữa hai ma trận, với một ma trận là tập hợp các tham số, hay còn được gọi là kernel, ma trận còn lại sẽ là phần giá trị bị giới hạn lại trong phạm vi tiếp nhận của các giá trị trong kernel. Một kernel sẽ có kích thước nhỏ hơn bức ảnh nhưng lớn hơn về chiều sâu. Nói một cách dễ hiểu hơn, nếu như một bức ảnh có ba kênh màu (Đỏ, Xanh lá, Xanh dương – RGB), chiều cao và chiều rộng của kernel sẽ nhỏ hơn bức ảnh đó, nhưng chiều sâu của kernel được tăng lên tương ứng với ba kênh màu đó.



Hình 1.1.2.1 Mô tả hoạt động của lớp tích chập.

Trong quá trình hoạt động, kernel sẽ “trượt” đi trên bức ảnh và tạo ra các giá trị khác nhau. Các giá trị được sắp xếp thành ma trận 2 chiều với vị trí tương ứng của kernel khi trượt đi trên bức ảnh đầu vào. Khoảng cách trượt của kernel sẽ được gọi là stride.

Hình 1.1.2.2 dưới đây sẽ minh họa chi tiết hơn về cách mà lớp tích chập hoạt động:

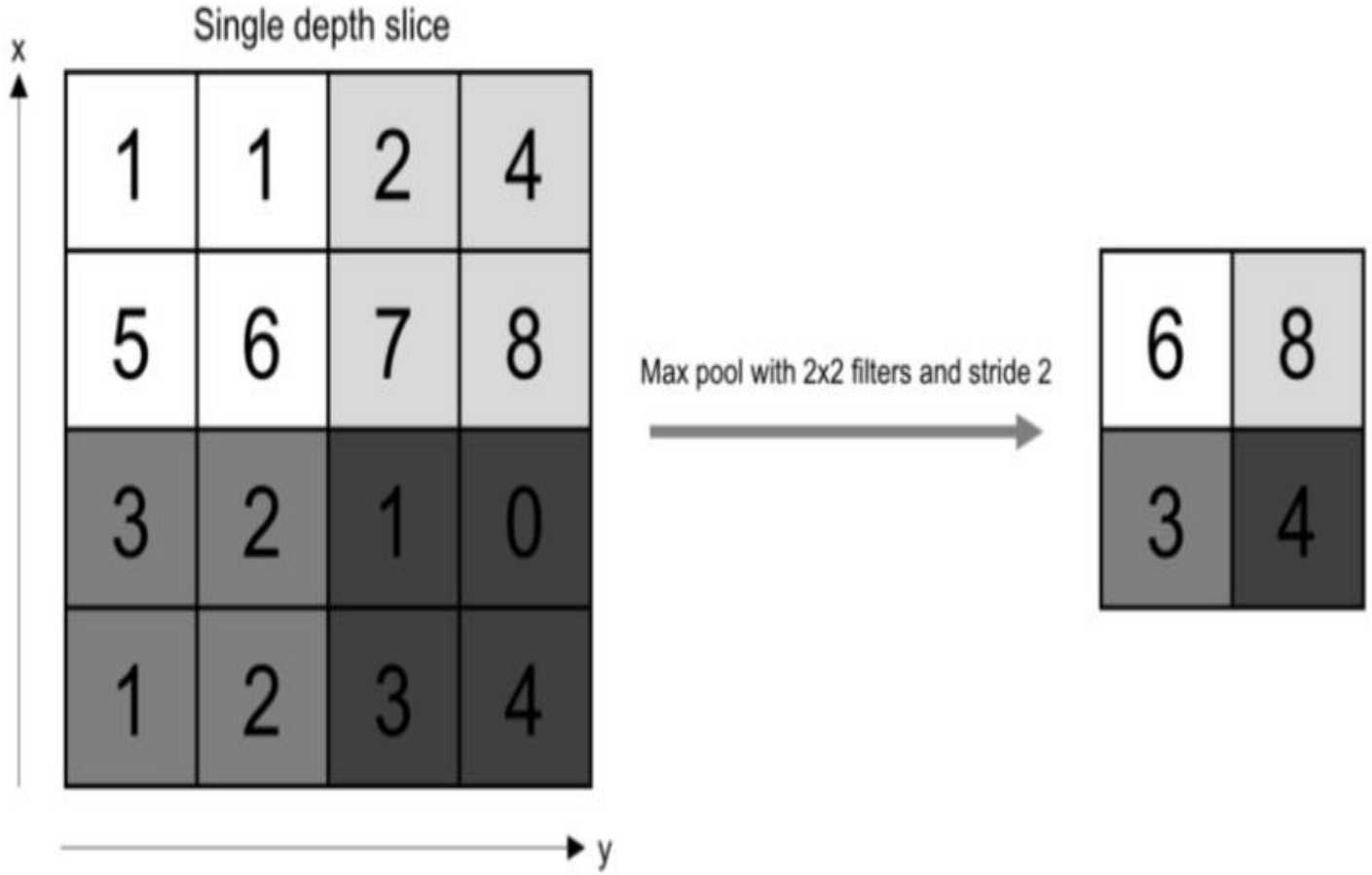


Hình 1.1.2.2.

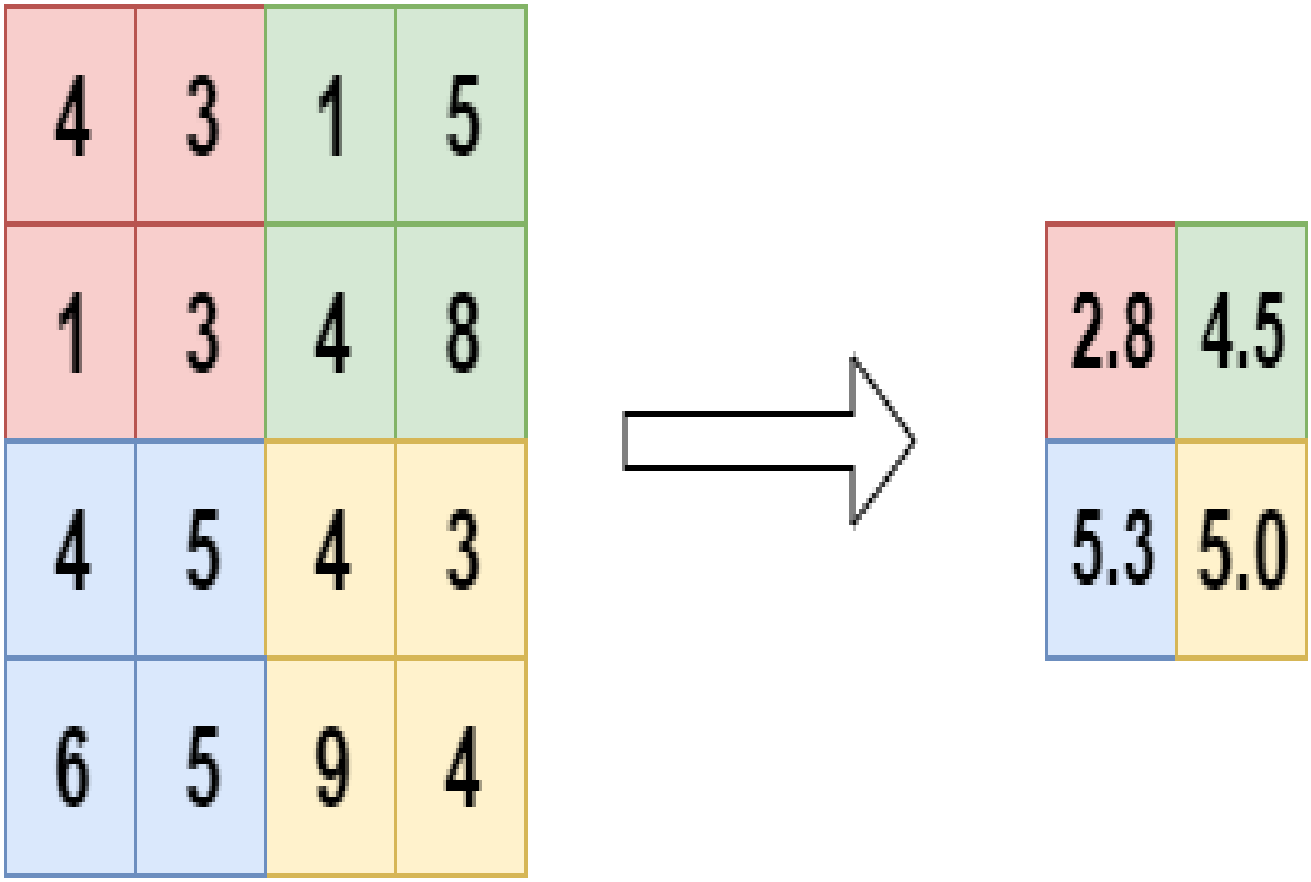
1.1.3 Định nghĩa lớp pooling (Pooling Layer):

Lớp pooling sẽ thay thế các giá trị đầu ra của mạng CNN ở các vị trí bằng cách trích xuất hoặc tính toán với các giá trị xung quanh chúng. Điều này giúp làm giảm kích thước của ma trận 2 chiều của ảnh khi đi qua khối tích chập, từ đó sẽ hạn chế đi bớt những phép tính và tham số cho các khối phía sau.

Có rất nhiều những phương pháp pooling khác nhau như: max pooling, average pooling, L2 norm pooling, v.v. Tuy nhiên, các phương pháp thường xuyên được sử dụng vì sự đơn giản của chúng sẽ là: max pooling (trích xuất giá trị lớn nhất trong các giá trị xung quanh để làm đầu ra) và average pooling (tính trung bình cộng các giá trị xung quanh để làm đầu ra).



Hình 1.1.3.1 Mô tả hoạt động của lớp max pooling.



Hình 1.1.3.2 Mô tả hoạt động của lớp average pooling.

Đối với tất cả các phương pháp pooling, lớp này sẽ đóng vai trò như một trình biên dịch, phân tích, giúp máy tính nhận biết được các vật thể có trong một khung ảnh, bất kể vị trí của vật thể đó nằm ở đâu.

1.1.4 Định nghĩa lớp kết nối (Fully Connected Layer):

Các nơ ron ở lớp này sẽ được liên kết với cả những nơ ron ở các lớp trước và sau nó, vì vậy nên lớp này có thể sử dụng phương pháp nhân ma trận để tính toán. Lớp kết nối này sẽ giúp sắp xếp, định dạng lại ma trận đầu ra để tương thích với ma trận ảnh đầu vào của mạng CNN.

1.1.5 Các lớp không tuyến tính (Non-linear Layers):

Bởi vì bản chất của tích chập là tuyến tính, nhưng hình ảnh thì lại không phải như vậy. Cho nên các lớp không tuyến tính sẽ được đặt trực tiếp ngay sau lớp tích chập, để giúp ma trận ảnh sau khi được tính toán sẽ không còn tuyến tính để giống hơn so với bức ảnh đầu vào.

Có khá nhiều những lớp không tuyến tính và dưới đây sẽ là một vài lớp nổi bật:

1. **Softmax:**

   Lớp không tuyến tính này có phương thức hoạt động tương tự với phương thức hoạt động của Sigmoid.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

   Đầu vào của lớp softmax có thể là số âm, số dương, 0, số lớn hơn 0, … nhưng tất cả sẽ được chuyển thành giá trị nằm trong khoảng từ 0 đến 1 và được xem như là những khả năng có thể xảy ra. Giá trị đầu vào của softmax càng lớn, thì đầu ra của nó càng lớn, tương tự với các giá trị nhỏ hơn, nhưng đầu ra sẽ vẫn nằm trong khoảng từ 0 tới 1.

   Mạng CNN thường kết thúc với rất nhiều lớp khác nhau, kéo theo đó sẽ có rất nhiều đầu ra khác nhau. Điều này khiến cho việc phân tích sẽ rất khó khăn. Đây sẽ là nơi mà lớp softmax phát huy tác dụng, lớp này sẽ lấy các giá trị đầu vào khác nhau, sau đó cho ra các giá trị đầu ra đồng nhất, giúp cho việc nghiên cứu, phân tích, hoặc làm đầu vào cho các hệ thống khác trở nên rất dễ dàng. Chính vì

lý do này mà lớp softmax thường sẽ nằm ở cuối cùng của một mô hình mạng nơ ron tích chập.

2. Tanh:

Đây thực chất là công thức lượng giác Tanh(x), chức năng của lớp này cũng giống với lớp Softmax. Tuy nhiên giá trị đầu ra của lớp này sẽ nằm ở khoảng từ -1 đến 1.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

3. ReLU:

Lớp chỉnh sửa tuyến tính (Rectified Linear Unit – ReLU) đã trở nên rất phổ biến trong vài năm trở lại đây. Chức năng của khối này thực chất là sự so sánh giữa giá trị đầu vào với giá trị 0. Khi so sánh với lớp Softmax và Tanh thì lớp ReLU có hiệu năng và khả năng tổng hợp giá trị tốt hơn gấp 6 lần.

$$f(x) = \begin{cases} 0, if\ x < 0 \\ x, if\ x \geq 0 \end{cases}$$

1.2 Các ứng dụng phổ biến của mô hình mạng nơ ron tích chập:

Dưới đây là một số ứng dụng phổ biến đang được sử dụng của mạng CNN:

1. Nhận diện vật thể: Từ mô hình mạng CNN, hiện nay đã có rất nhiều các mô hình phức tạp hơn như: R-CNN, Fast-R-CNN, Faster-R-CNN. Các mô hình hiện đại này được phát triển để nhận diện các vật thể mức khó hơn và được

tích hợp trong các phương tiện giao thông tự động, các thiết bị nhận diện khuôn mặt, v.v

2. Phân lớp hình ảnh: tên của ứng dụng này là phân lớp, nhưng bản chất hoạt động của nó lại là kết nối các pixel trong ảnh lại với nhau thành cùng một nhãn. Những nhãn này có thể là con người, xe cộ, cây cối, đường phố, động vật, v.v. Chính nhờ những nhãn này mà hiện nay ra đời các phương tiện giao thông tự lái, vì chúng có khả năng phân tích hình ảnh chúng lấy vào thành các vật thể khác nhau.
3. Chú thích hình ảnh: các mô hình CNN được ứng dụng trong việc ghi chú cho các hình ảnh và video clip. Ứng dụng này được sử dụng rất rộng rãi, ví dụ như nhận biết hoạt động, mô tả hình ảnh và video clip. Đặc biệt, chú thích hình ảnh có thể được ứng dụng trong việc hỗ trợ người khiếm thị. Ứng dụng này đang được đội ngũ Microsoft phát triển để sớm đưa vào bộ công cụ Office, và hiện đã có mặt trên ứng dụng mạng xã hội Twitter.

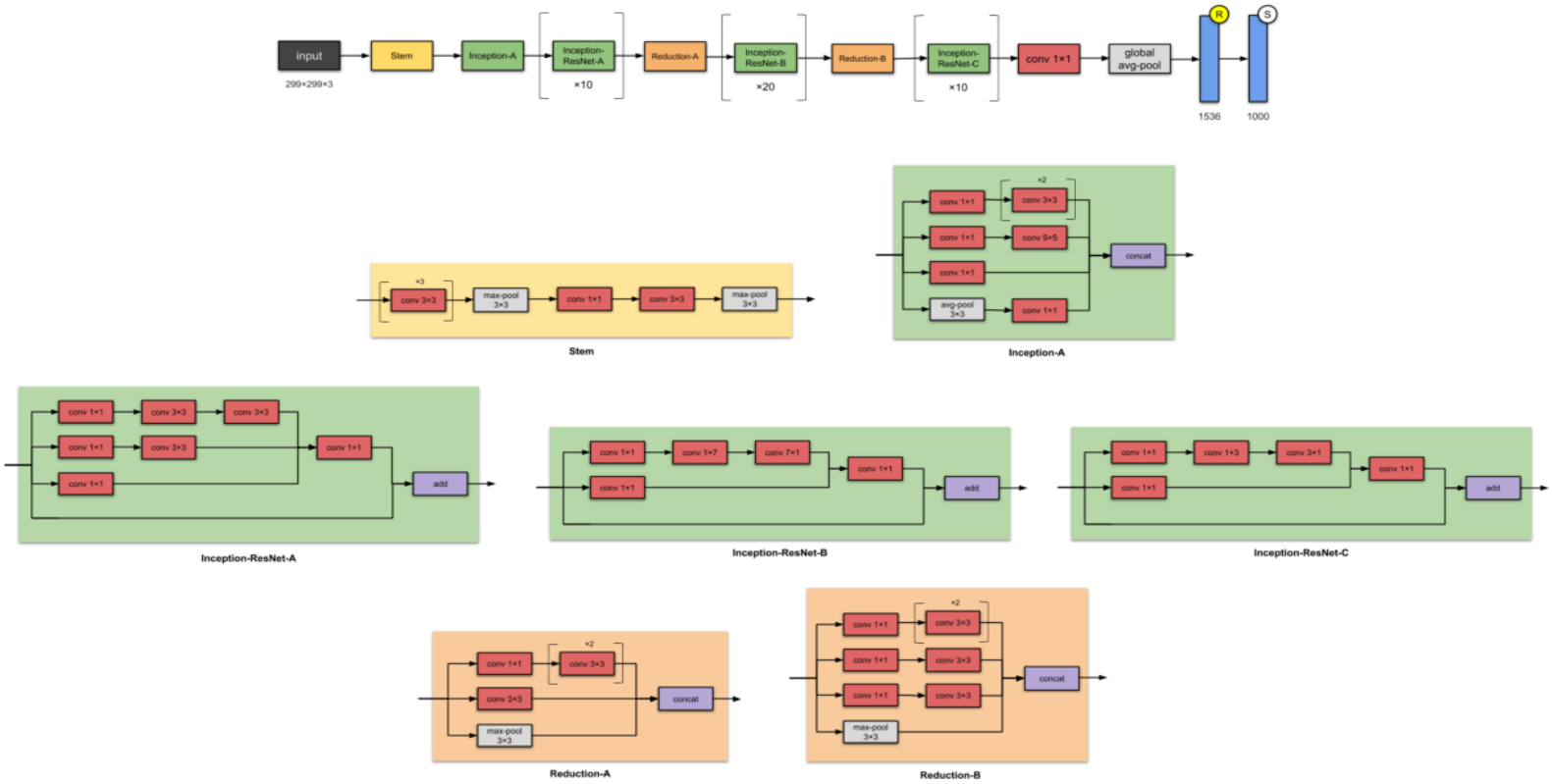
Mô hình CNN giúp chúng ta khám phá ra được lượng thông tin khổng lồ, kể cả ở phiên bản đơn giản nhất, các mô hình này vẫn hỗ trợ con người rất nhiều. CNN hiện đang và sẽ tiếp tục định hình cách chúng ta nhìn nhận thế giới xung quanh – Hãy thử tưởng tượng chúng ta sẽ gặp khó khăn đến mức nào khi Google không có ứng dụng tìm kiếm hình ảnh với mô tả bằng chữ, cùng với đó là rất nhiều trở ngại khác. Và đó chính là cách mà các mô hình mạng nơ ron tích chập hiện nay đang thay đổi cuộc sống của chúng ta.

Chương 2: Nội Dung Chính

2.1 Mô hình mạng Inception Resnet V2:

Inception Resnet V2 được huấn luyện trên cơ sở dữ liệu với hàng triệu bức ảnh khác nhau của Imagenet. Mô hình mạng này có tổng cộng 164 lớp về chiều sâu và có thể phân loại lên đến 1000 vật thể khác nhau như: bàn phím, chuột máy tính, bút chì và nhiều loài động vật khác. Inception Resnet V2 có kích thước ảnh đầu vào là 299*299 pixel, với số lượng tham số vào khoảng 56 triệu giá trị.

Inception Resnet V2 là một mô hình thuộc Inception, những mô hình thuộc Inception được thiết kế cho mục đích phân loại hình ảnh. Inception Resnet V2 có chi phí tính toán tương đương với của Inception 4 và là một biến thể của Inception V3.



Hình 2.1.1 Kiến trúc mô hình Inception Resnet V2.

```
Stem:   Conv33, stride 2, pad 0                 299*299*3 -> 149*149*32
        Conv33, stride 1, pad 0                 149*149*32 -> 147*147*32
        Conv33, stride 1, pad 1                 147*147*32 -> 147*147*64
        MaxPool33, stride 2, pad 0              147*147*64 -> 73*73*64
        Conv11, stride 1, pad 0                 73*73*64 -> 73*73*80
        Conv33, stride 1, pad 0                 73*73*80 -> 71*71*192
        MaxPool33, stride 2, pad 0              71*71*192 -> 35*35*192

Inception A:    Conv11, stride 1, pad 0         35*35*192 -> 35*35*48
                Conv33, stride 1, pad 1         35*35*48 -> 35*35*96
                Conv33, stride 1, pad 1         35*35*96 -> 35*35*96
                Conv11, stride 1, pad 0         35*35*192 -> 35*35*48
                Conv55, stride 1, pad 2         35*35*48 -> 35*35*64
                Conv11, stride 1, pad 0         35*35*192 -> 35*35*96
                AvgPool33, stride 1, pad 1      35*35*192 -> 35*35*48
                Conv11, stride 1, pad 0         35*35*48 -> 35*35*64

Inception Resnet A:     Conv11, stride 1, pad 0         35*35*192 -> 35*35*32
                        Conv33, stride 1, pad 1         35*35*32 -> 35*35*48
                        Conv33, stride 1, pad 1         35*35*48 -> 35*35*64
                        Conv11, stride 1, pad 0         35*35*192 -> 35*35*32
                        Conv33, stride 1, pad 1         35*35*32 -> 35*35*32
                        Conv11, stride 1, pad 0         35*35*192 -> 35*35*32
                        Conv11, stride 1, pad 0         35*35*128 -> 35*35*320

Reduction A:    Conv11, stride 1, pad 0                 35*35*320 -> 35*35*256
                Conv33, stride 1, pad 1                 35*35*256 -> 35*35*256
                Conv33, stride 2, pad 0                 35*35*256 -> 17*17*384
                Conv33, stride 2, pad 0                 35*35*320 -> 17*17*384
                MaxPool33, stride 2, pad 0              35*35*320 -> 17*17*320

Inception Resnet B:     Conv11, stride 1, pad 0         17*17*1088 -> 17*17*128
                Conv17, stride 1, custom pad 3          17*17*128 -> 17*17*160
                Conv71, stride 1, custom pad 3          17*17*160 -> 17*17*192
                Conv11, stride 1, pad 0                 17*17*1088 -> 17*17*192
                Conv11, stride 1, pad 0                 17*17*384 -> 17*17*1088

Reduction B:    Conv11, stride 1, pad 0                 17*17*1088 -> 17*17*256
                Conv33, stride 1, pad 1                 17*17*256 -> 17*17*288
                Conv33, stride 2, pad 0                 17*17*288 -> 8*8*320
                Conv11, stride 1, pad 0                 17*17*1088 -> 17*17*256
                Conv33, stride 2, pad 0                 17*17*256 -> 8*8*288
                Conv11, stride 1, pad 0                 17*17*1088 -> 17*17*256
                Conv33, stride 2, pad 0                 17*17*256 -> 8*8*384
                MaxPool 33, stride 2, pad 0             17*17*1088 -> 8*8*1088

Inception Resnet C:     Conv11, stride 1, pad 0                 8*8*2080 -> 8*8*192
                        Conv13, stride 1, custom pad 1          8*8*192 -> 8*8*224
                        Conv31, stride 1, custom pad 1          8*8*224 -> 8*8*256
                        Conv11, stride 1, pad 0                 8*8*2080 -> 8*8*192
                        Conv11, stride 1, pad 0                 8*8*448 -> 8*8*2080
```

Hình 2.1.2 Mô hình Inception Resnet V2 với đầu vào và đầu ra của từng khối tính toán thành phần bên trong.

2.2 Quá trình hiện thực mô hình mạng:

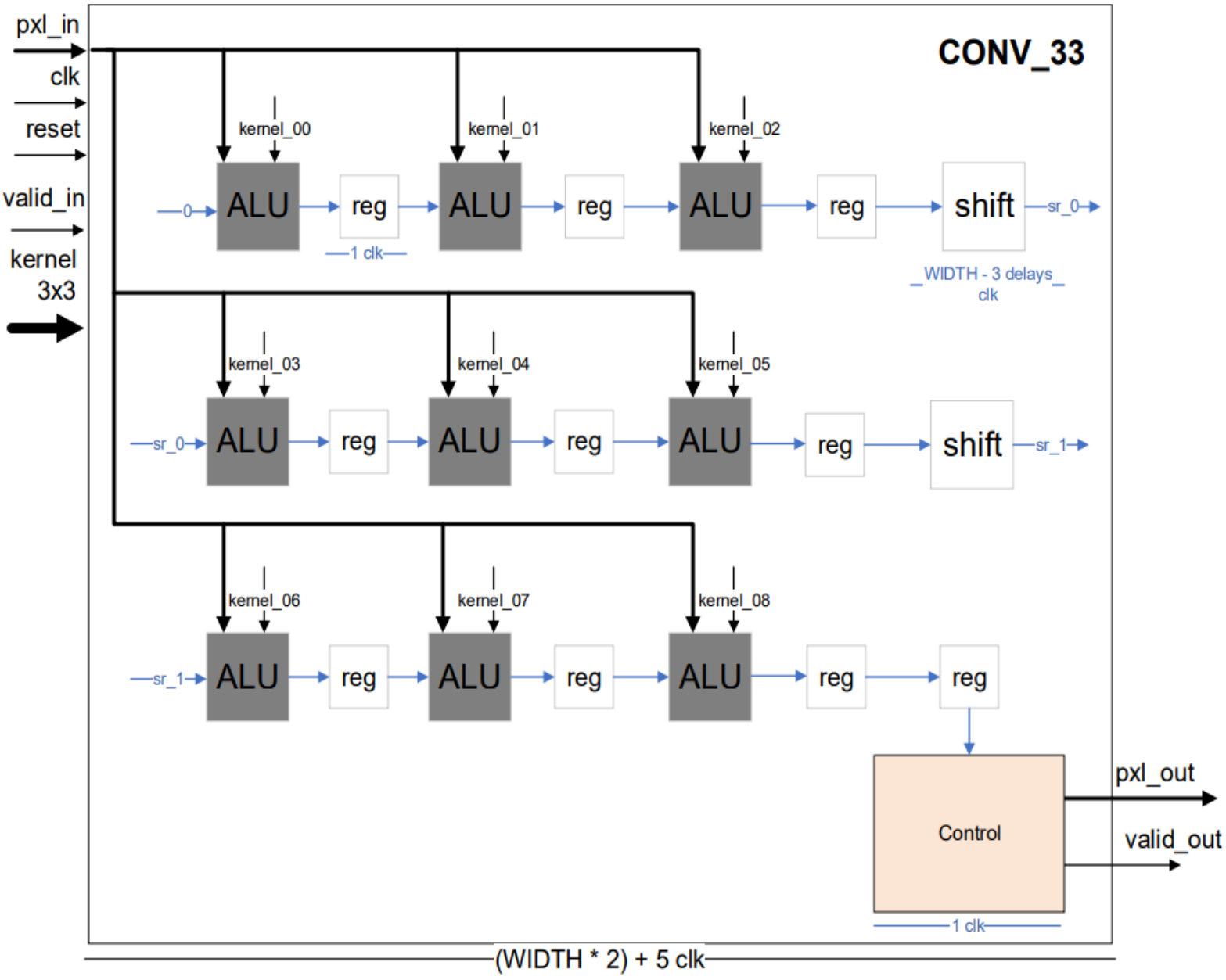
Khi đã có được kiến trúc của Inception Resnet V2, nhóm chúng em tiếp tục thiết kế đến các khối thành phần nhỏ hơn bên trong, bao gồm có: Convolution 3x3, 5x5, 1x3, 3x1, 1x7, 7x1, Max pool 3x3, Average Pool 3x3, …

Với các khối tích chập, dữ liệu đưa vào của hàng thứ nhất trong ma trận ảnh sau khi được tính toán sẽ được cho vào FIFO (kích thước của FIFO sẽ bằng độ rộng của ma trận ảnh ban đầu trừ cho độ rộng của kernel). Nhằm mục đích tạo độ trễ cho dữ liệu của hàng kế tiếp, để khi vừa tính toán xong dữ liệu hàng đầu cũng vừa lúc FIFO đã đầy dữ liệu và chuẩn bị được đọc ra. Vào thời điểm đó, khi dữ liệu đã tính toán xong của hàng đầu được đọc ra từ FIFO, những dữ liệu đó sẽ tiếp tục được tính toán với hàng tiếp theo của ma trận ảnh được đưa vào. Dữ liệu đưa vào sẽ liên tục được tính toán mà không gặp bất cứ trở ngại nào. Kích thước của kernel cũng sẽ tỉ lệ thuận với số hàng tính trong khối tích chập này. Ví dụ, kernel có kích thước 3x3 thì sẽ có 3 hàng tính toán với 3 khối cộng mỗi hàng, kernel có kích thước 5x5 sẽ có 5 hàng tính toán với 5 khối cộng mỗi hàng, … tương tự với các khối tích chập có kích thước khác nhau.

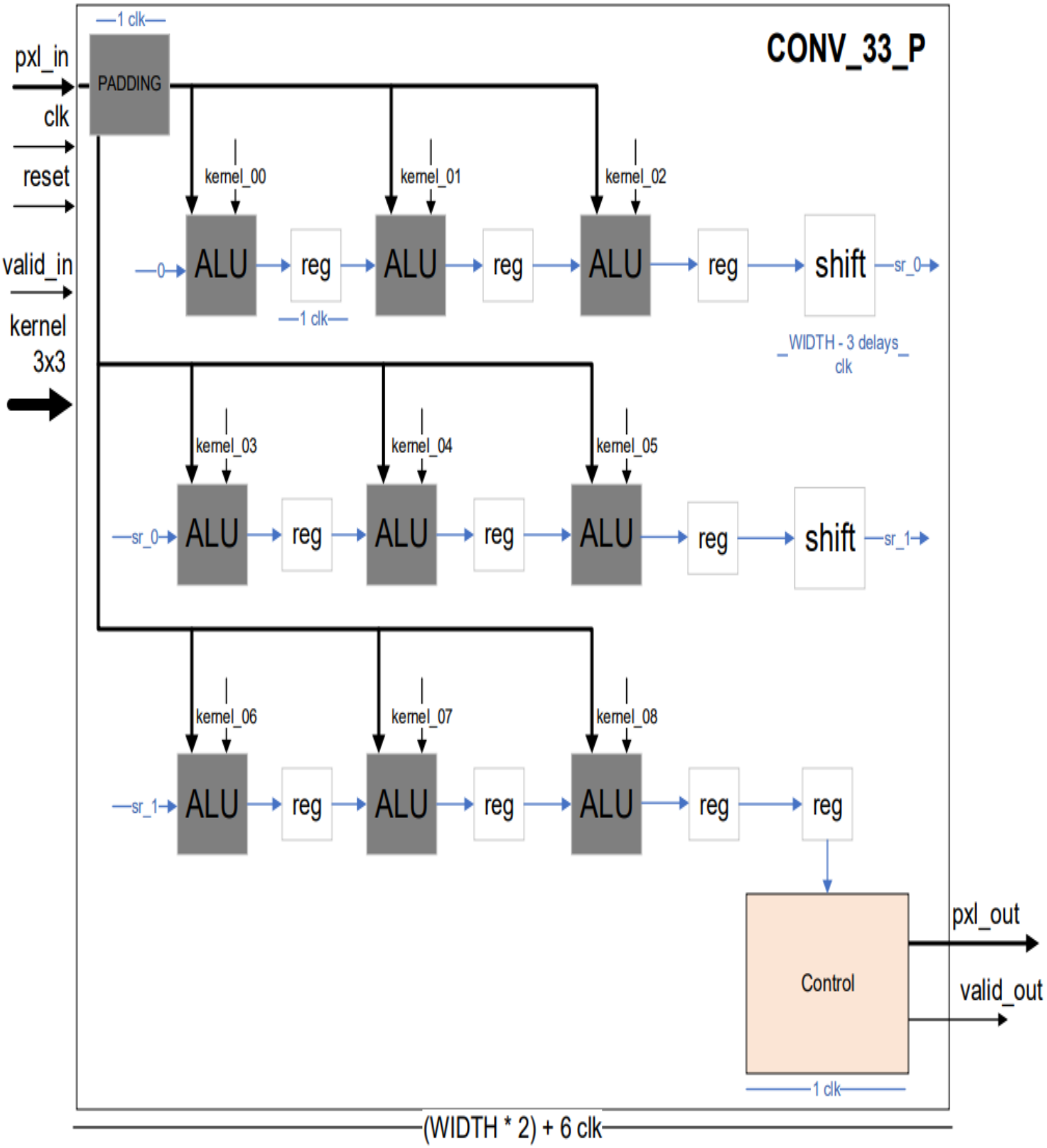
Các khối pooling có kiến trúc tương tự với các khối tích chập, tuy nhiên sẽ khác ở các khối tính toán. Ví dự như max pooling, dữ liệu đầu vào sẽ không được xử lý qua các khối tính toán, mà thay vào đó sẽ đi qua một bộ lọc để lọc ra giá trị lớn nhất và sau đó sẽ được truyền vào FIFO. Average pooling thì sẽ được cộng lại với nhau và sẽ đi qua một bộ chia để tính trung bình cộng trước khi được truyền vào bên trong FIFO.

Đối với các khối có yêu cầu về bước đi (stride) lớn hơn 1, và có lớp đệm (padding) thì cấu trúc sẽ có một vài điều chỉnh để đáp ứng yêu cầu đó. Khi bước đi của kernel

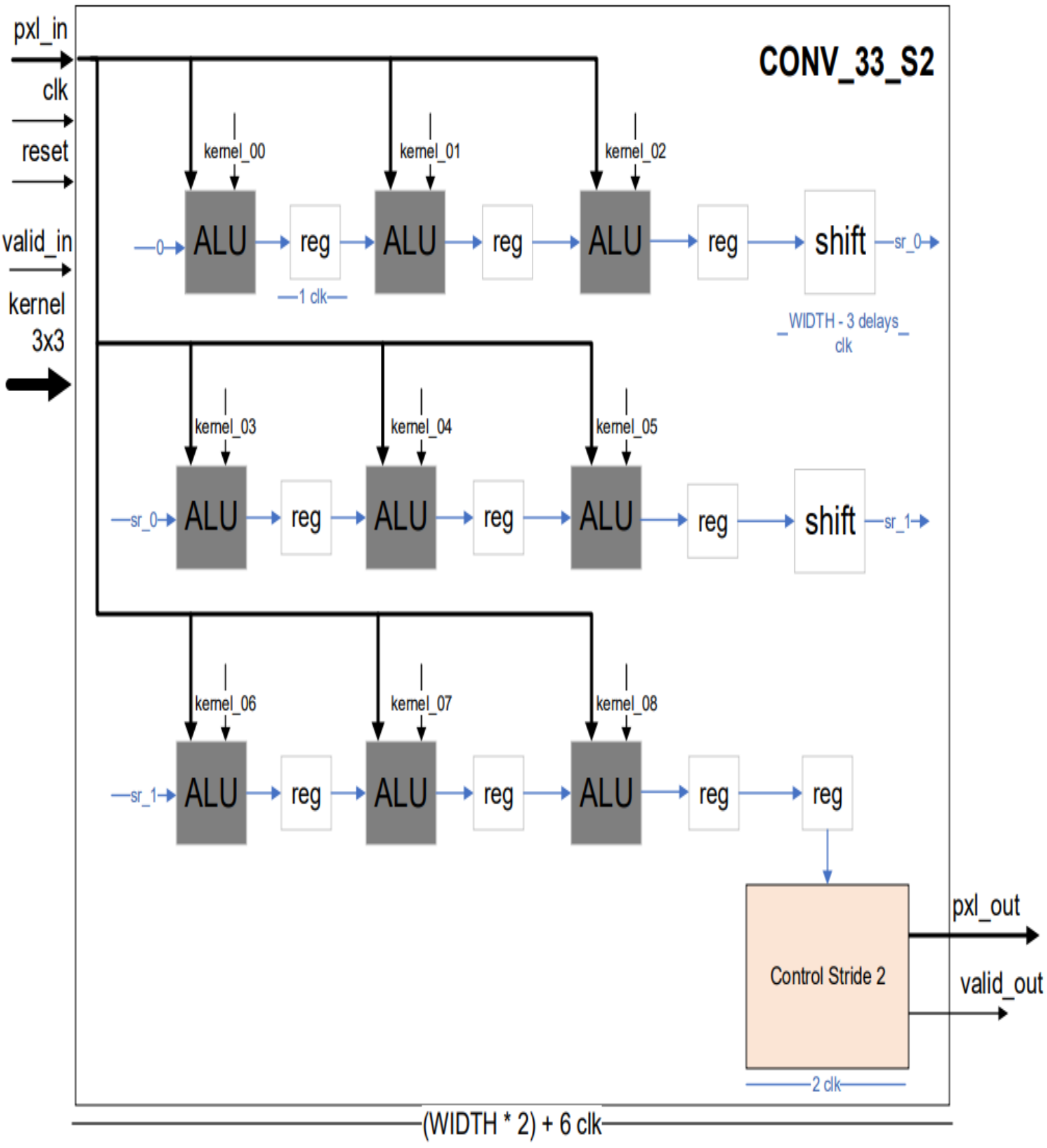
có giá trị lớn hơn 1, nhóm chúng em vẫn sử dụng kiến trúc mẫu của khối tích chập. Tuy nhiên, khi xuất kết quả sẽ có một biến đếm được tạo bên trong kiến trúc, biến đếm sẽ tương ứng với bước đi, để khi kết quả đến đầu ra thì sẽ chỉ lấy 1 giá trị và bỏ đi 1 vài giá trị tương ứng với số bước đi, rồi tiếp theo sẽ lấy tiếp 1 giá trị khác và lần lượt như thế. Còn đối với padding lớn hơn 0, dữ liệu đầu vào sẽ được đệm thêm số hàng và cột chứa giá trị 0 tương ứng với số padding.



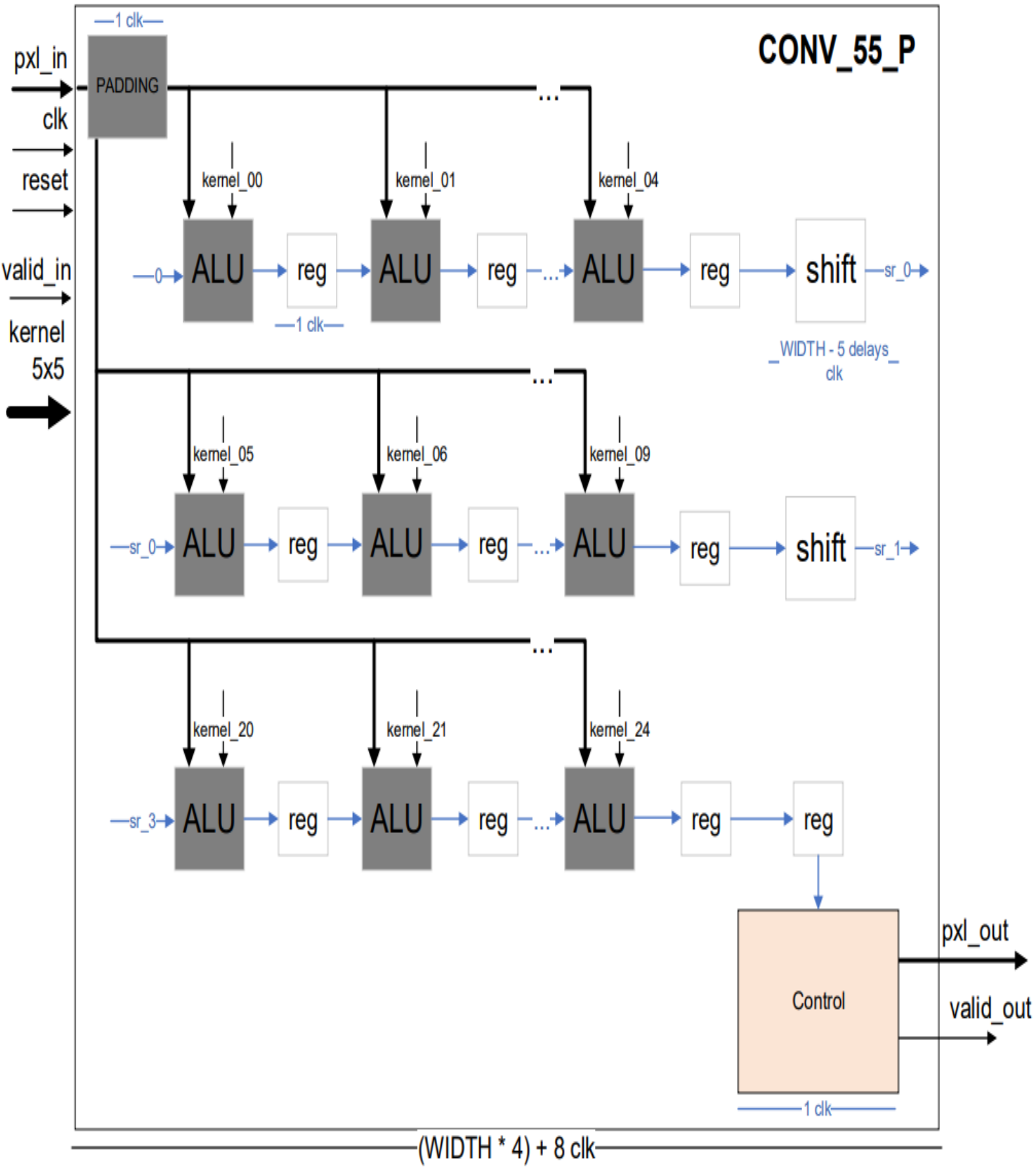
Hình 2.2.1 Kiến trúc của khối tích chập với kernel kích thước là 3x3 và không có padding.



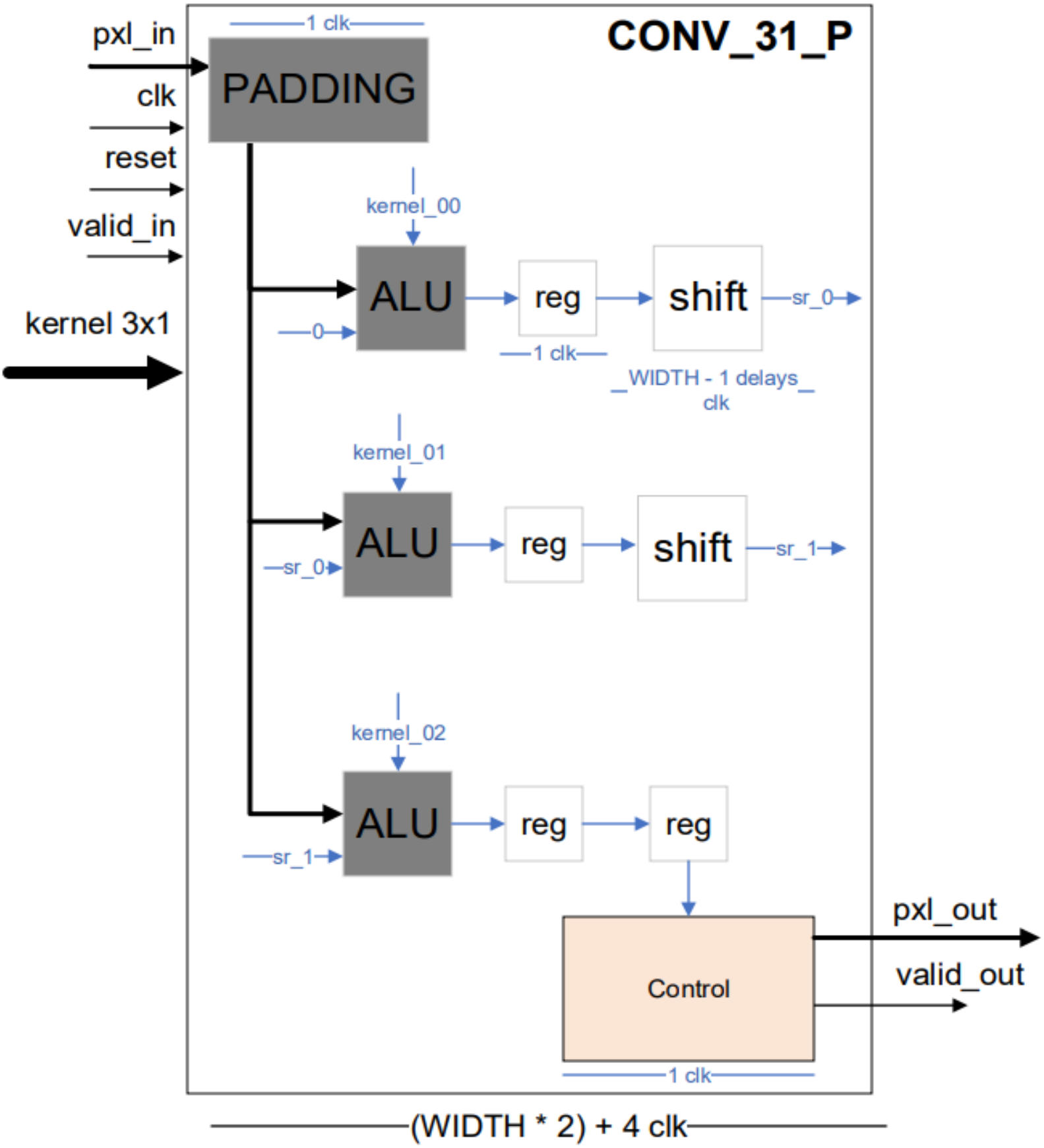
Hình 2.2.2 Kiến trúc của khối tích chập với kernel kích thước là 3x3 và có padding.



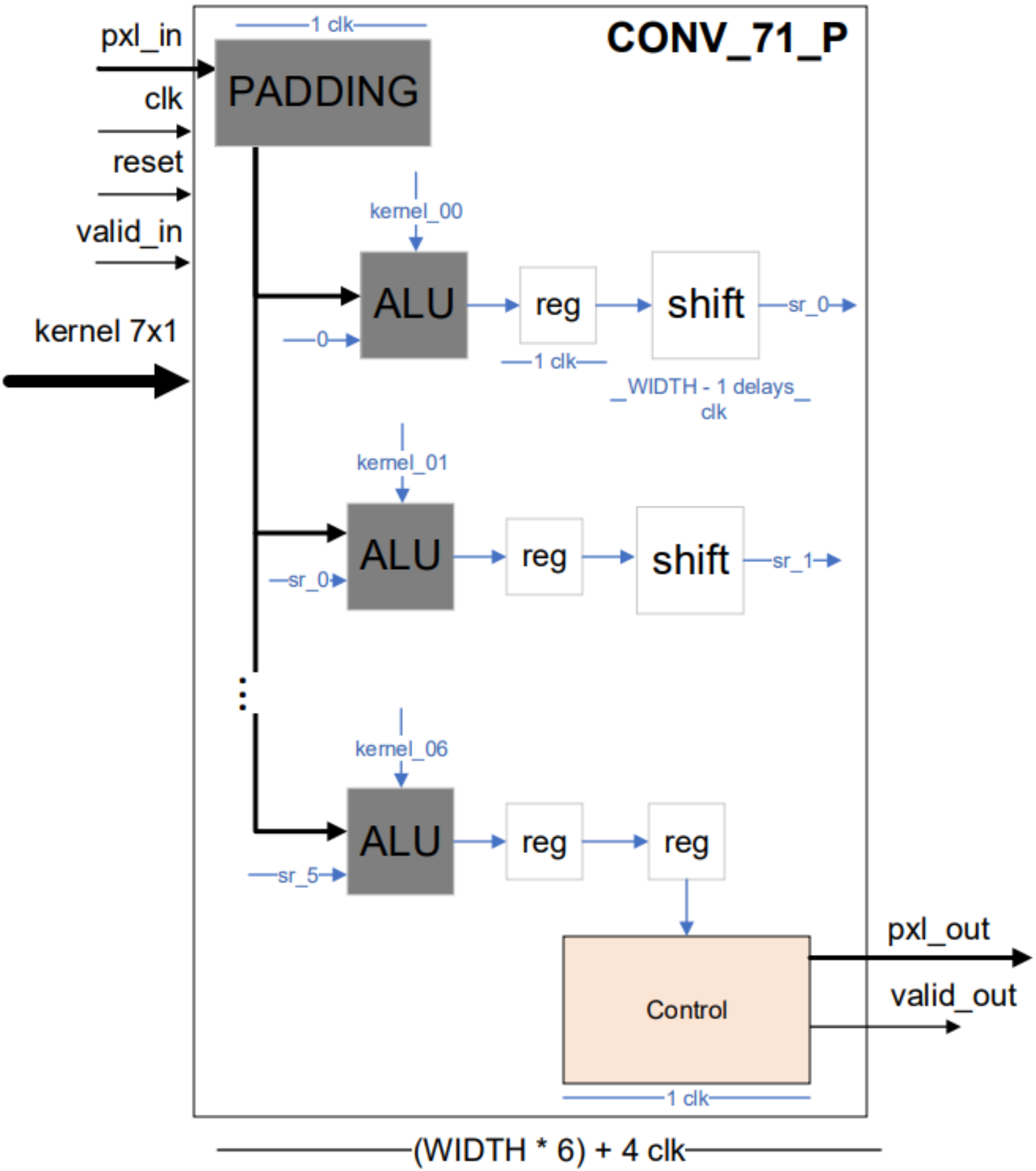
Hình 2.2.3 Kiến trúc của khối tích chập với kernel kích thước là 3x3, có stride và không có padding.



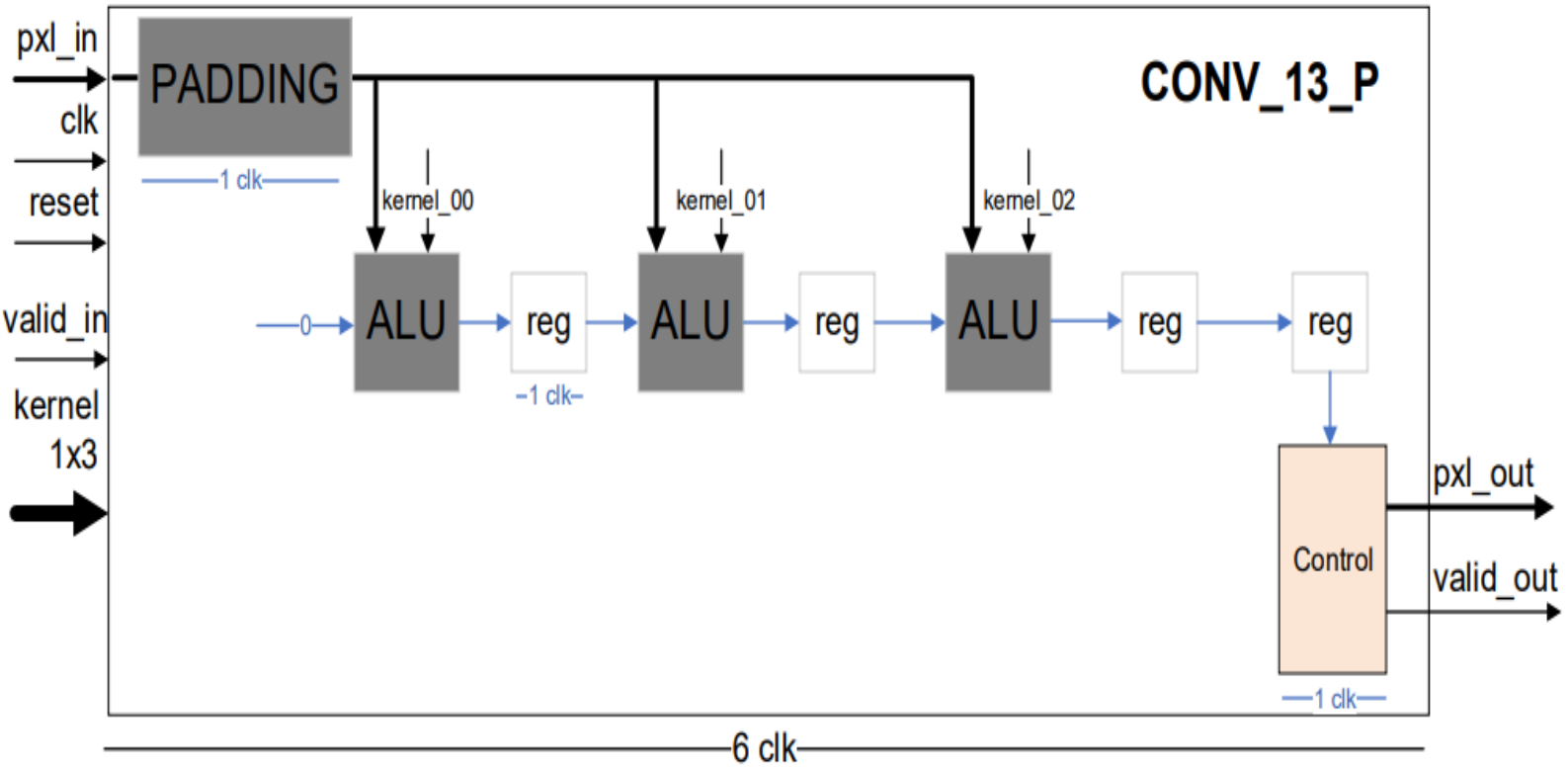
Hình 2.2.4 Kiến trúc của khối tích chập với kernel kích thước là 5x5 và có padding.



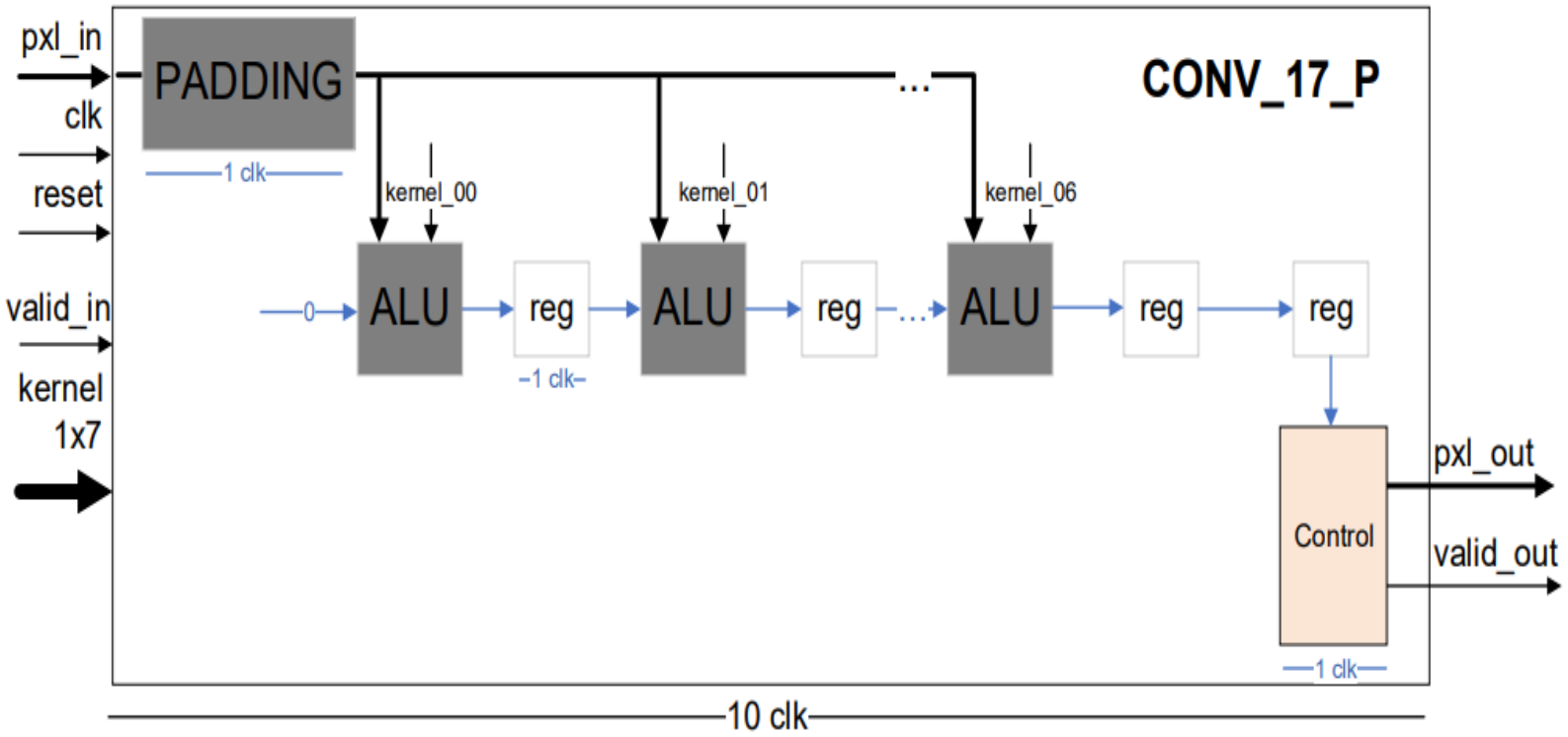
Hình 2.2.5 Kiến trúc của khối tích chập với kernel kích thước là 3x1 và có padding.



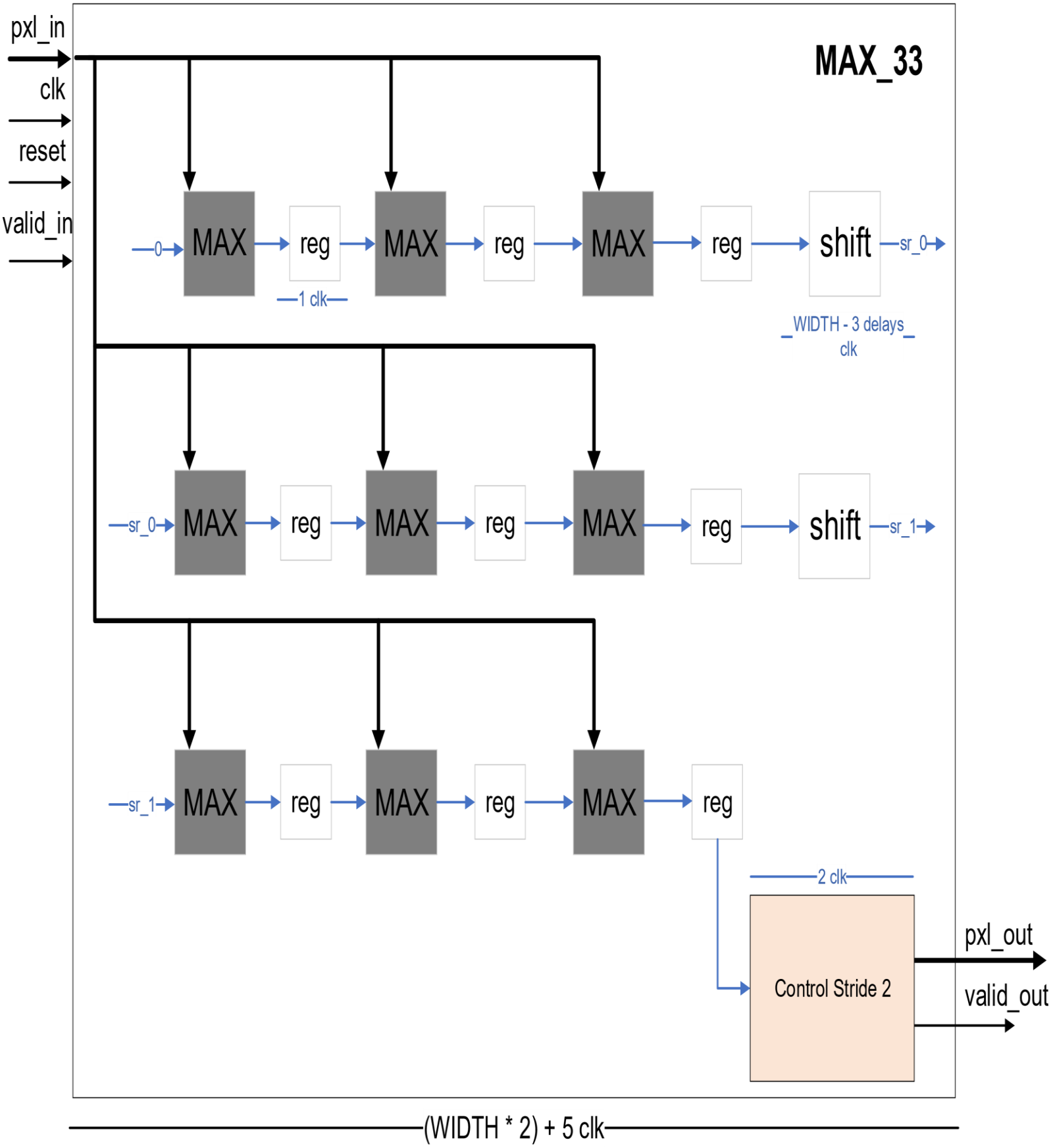
Hình 2.2.6 Kiến trúc của khối tích chập với kernel kích thước là 7x1 và có padding.



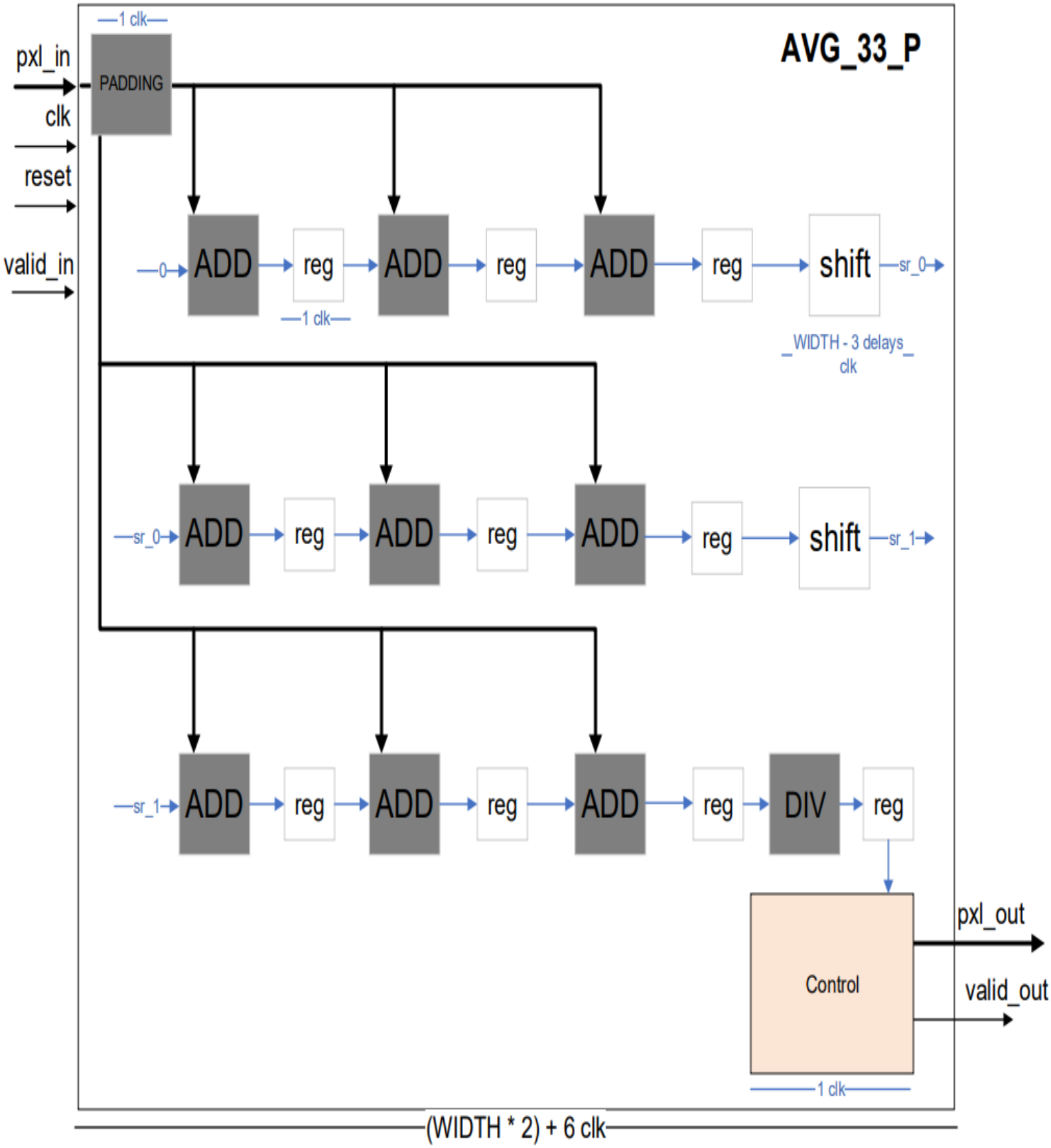
Hình 2.2.5 Kiến trúc của khối tích chập với kernel kích thước là 1x3 và có padding.



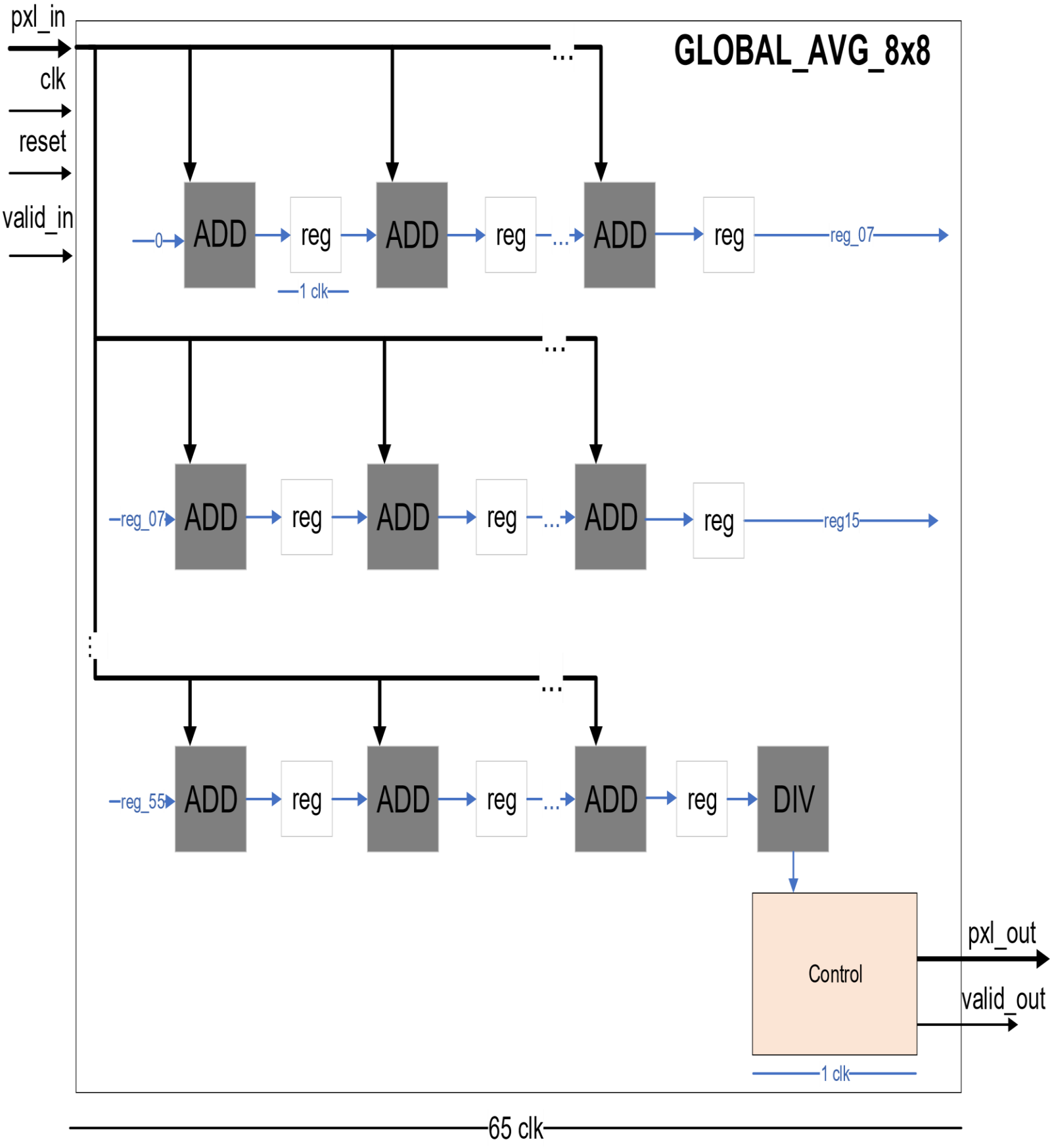
Hình 2.2.7 Kiến trúc của khối tích chập với kernel kích thước là 1x7 và có padding.



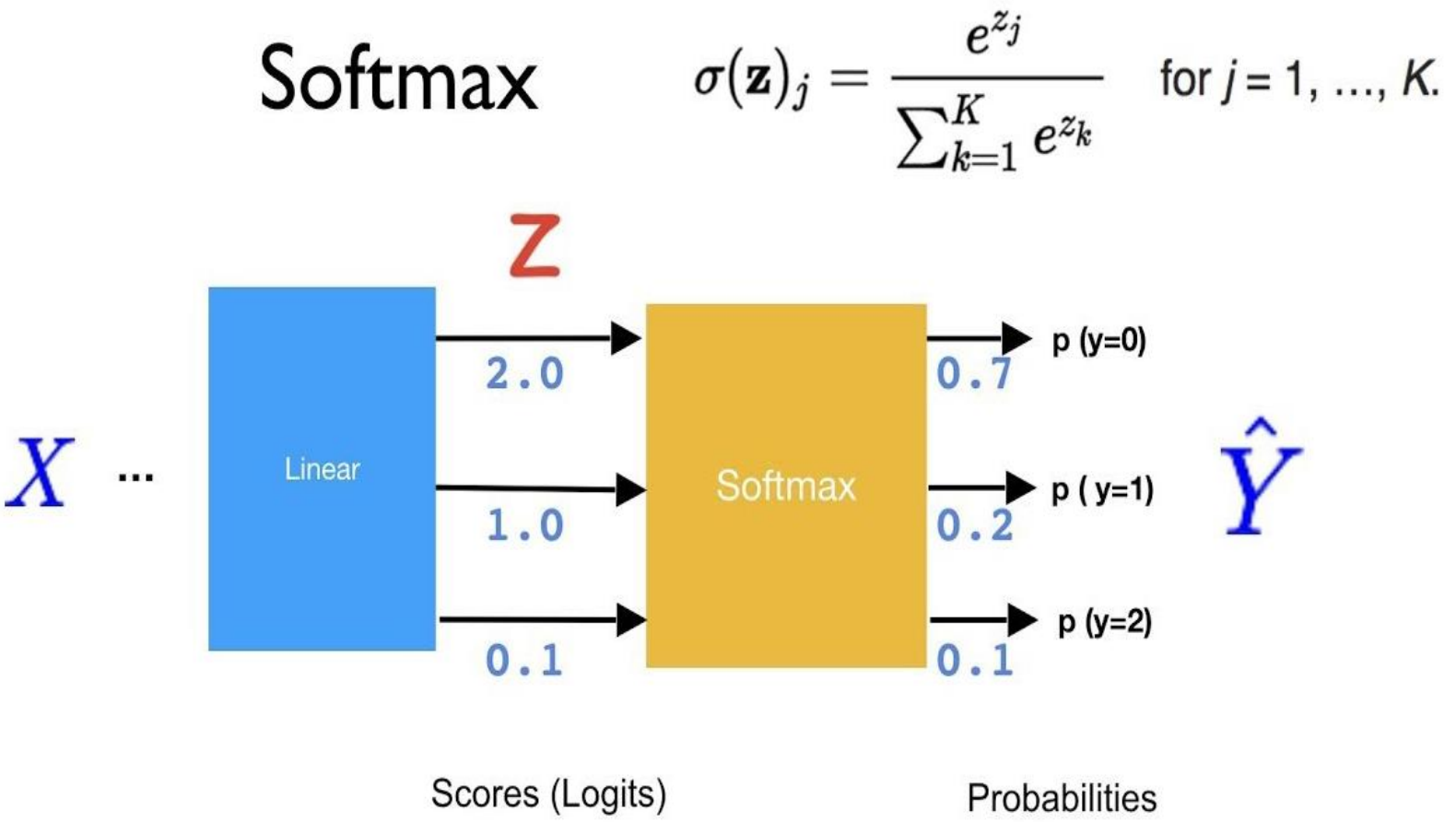
Hình 2.2.8 Kiến trúc của khối max pooling với kernel kích thước 3x3 và không có padding.



Hình 2.2.9 Kiến trúc của khối avergage pooling với kernel kích thước 3x3 và có padding.



Hình 2.2.10 Kiến trúc của khối global average pooling với kernel có kích thước 8x8, stride = 1, padding = 0.



Hình 2.2.11 Kiến trúc của khối Softmax.

Sau khi đã có được hầu hết kiến trúc của các khối cần thiết, nhóm chúng em tiến hành tạo các khối đó, nối chúng lại với nhau thành các khối lớn của mạng, và nối các khối lớn đó lại với nhau thành mô hình mạng Inception Resnet V2 hoàn chỉnh bằng ngôn ngữ phần cứng Verilog HDL. Kết quả của các khối sau đó sẽ được ghi vào các tập tin để so sánh độ chính xác với kết quả từ việc mô phỏng bằng ngôn ngữ Python.

Nhóm chúng em dự định sẽ thực nghiệm mô hình mạng Inception Resnet V2 để nhận diện vật thể có trong 1 bức ảnh (chó hoặc mèo). Với cơ sở dữ liệu hình ảnh lấy từ Imagenet, nhóm chúng em đã lấy được các tập tin trọng số dùng cho việc huấn luyện mô hình. Mô hình mạng này sẽ có dữ liệu đầu vào là ma trận điểm ảnh của một bức ảnh, đầu ra sẽ là tên của vật thể nhận dạng được từ trong bức ảnh.

2.3 Kết quả thực nghiệm:

Hình 2.3.1 Bức ảnh thử nghiệm.

```
Loaded Model Weights!
Class is: African elephant, Loxodonta africana
Certainty is: 0.868498
```

Hình 2.3.2 Kết quả khi thực nghiệm mô hình trên Python.

Mô hình mạng Inception Resnet V2 khi được thực nghiệm trên Python cho ra kết quả chính xác, với mức độ chắc chắn cao. Dựa trên cơ sở này, nhóm đã tiến hành liên kết các khối tính toán lại với nhau để xây dựng lên mô hình mạng.

Vì bộ chia số floating point 32 bit của nhóm chúng em chưa thực sự tối ưu, cho nên kết quả của khối có phép chia (Average Pooling, Softmax) sẽ không thể đạt được kết quả tốt nhất. Còn lại các khối khác sẽ đạt được kết quả với sai số phần thập phân ở mức khá tốt.

Sai Số Verilog vs Python	Tổng Số Giá Trị
Lớn hơn 0.1	376 / 22201
Lớn hơn 0.01	376 / 22201
Lớn hơn 0.001	376 / 22201
Lớn hơn 0.0001	384 / 22201
Lớn hơn 0.00001	17246 / 22201
Lớn hơn 0.000001	21932 / 22201

Bảng 2.3.1 Sai số giữa kết quả mô phỏng bằng Verilog so với Python của khối tích chập với kernel có kích thước 3x3, stride = 2, padding = 0, đầu vào ảnh 299x299.

Sai Số Verilog vs Python	Tổng Số Giá Trị
Lớn hơn 0.1	1515 / 88209
Lớn hơn 0.01	1517 / 88209
Lớn hơn 0.001	1518 / 88209
Lớn hơn 0.0001	1552 / 88209
Lớn hơn 0.00001	68613 / 88209
Lớn hơn 0.000001	87151 / 88209
Lớn hơn 0.0000001	88080 / 88209

Bảng 2.3.2 Sai số giữa kết quả mô phỏng bằng Verilog so với Python của khối tích chập với kernel có kích thước 3x3, stride = 1, padding = 0, đầu vào ảnh 299x299.

Sai Số Verilog vs Python	Tổng Số Giá Trị
Lớn hơn 0.0001	0 / 22201
Lớn hơn 0.00001	0 / 22201
Lớn hơn 0.000001	15417 / 22201
Lớn hơn 0.0000001	21529 / 22201

Bảng 2.3.3 Sai số giữa kết quả mô phỏng bằng Verilog so với Python của khối Max Pooling với kernel có kích thước 3x3, stride = 2, padding = 0, đầu vào ảnh 299x299.

Sai Số Verilog vs Python	Tổng Số Giá Trị
Lớn hơn 0.1	22160 / 22210
Lớn hơn 0.2	21905 / 22210
Lớn hơn 0.3	20676 / 22210
Lớn hơn 0.4	17098 / 22210
Lớn hơn 0.5	10966 / 22210
Lớn hơn 0.6	4097 / 22210
Lớn hơn 0.7	1580 / 22210
Lớn hơn 0.8	353 / 22210
Lớn hơn 0.9	46 / 22210
Lớn hơn 1.0	0 / 22210

Bảng 2.3.4 Sai số giữa kết quả mô phỏng bằng Verilog so với Python của khối Average Pooling với kernel có kích thước 3x3, stride = 2, padding = 0, đầu vào ảnh 299x299.

Sai Số Verilog vs Python	Tổng Số Giá Trị
Lớn hơn 0.0001	0 / 89401
Lớn hơn 0.00001	57571 / 89401
Lớn hơn 0.000001	89323 / 89401

Bảng 2.3.5 Sai số giữa kết quả mô phỏng bằng Verilog so với Python của khối Average Pooling với kernel có kích thước 3x3, stride = 1, padding = 1, đầu vào ảnh 299x299.

Sai Số Verilog vs Python	Tổng Số Giá Trị
Lớn hơn 0.1	0 / 1536
Lớn hơn 0.01	74 / 1536
Lớn hơn 0.001	231 / 1536
Lớn hơn 0.0001	343 / 1536
Lớn hơn 0.00001	462 / 1536
Lớn hơn 0.000001	561 / 1536
Lớn hơn 0.0000001	658 / 1536
Lớn hơn 0.00000001	732 / 1536
Lớn hơn 0.000000001	832 / 1536
Lớn hơn 0.0000000001	926 / 1536
Lớn hơn 0.00000000001	1007 / 1536
Lớn hơn 0.000000000001	1074 / 1536
Lớn hơn 0.0000000000001	1156 / 1536
Lớn hơn 0.00000000000001	1213 / 1536
Lớn hơn 0.000000000000001	1275 / 1536
Lớn hơn 0.0000000000000001	1331 / 1536
Lớn hơn 0.00000000000000001	1372 / 1536
Lớn hơn 0.000000000000000001	1420 / 1536
Lớn hơn 0.0000000000000000001	1445 / 1536

Bảng 2.3.6 Sai số giữa kết quả mô phỏng bằng Verilog so với Python của khối Softmax, đầu vào có 1536 giá trị nằm trong khoảng từ 0 đến 55.

Khối	Sai Số Trung Bình
Stem	3.5475123332066664e-13
Inception A	1.7406126658548 35e-16
Inception Resnet A	4.457320079003828e-17
Reduction A	9.642003933592909e-17
Inception Resnet B	9.617398986238853e-17
Reduction B	1.174091297121499e-17
Inception Resnet C	1.1618286886670832e-17

Bảng 2.3.7 Sai số giữa kết quả mô phỏng bằng Verilog so với Python của các khối thành phần của mạng, đầu vào của khối sau sẽ là đầu ra của khối trước, với dữ liệu ban đầu là ma trận ảnh 299*299 không có chiều sâu (ảnh xám).

Do khả năng của các thành viên trong nhóm còn hạn chế, cho nên nhóm chúng em chỉ đang ở bước kiểm tra và kết nối các khối tính toán nhỏ thành khối lớn có chiều sâu của mô hình mạng, chứ chưa thể hoàn thành được toàn bộ đồ án.

Tài Liệu Tham Khảo

AN FPGA-BASED PROCESSOR FOR CONVOLUTIONAL NETWORKS: farabet-fpl-09.pdf (lecun.com) (Truy cập lần cuối: 25/7/2021).

[AI Interview] 12 câu hỏi phỏng vấn Deep Learning siêu hay không thể bỏ qua (viblo.asia) (Truy cập lần cuối: 25/7/2021).

Convolutional Neural Networks: CS231n Convolutional Neural Networks for Visual Recognition (Truy cập lần cuối: 25/7/2021).

Keras Inception Resnet V2: yuyang-huang/keras-inception-resnet-v2: The Inception-ResNet v2 model using Keras (with weight files) (github.com) (Truy cập lần cuối: 25/7/2021).

VIPUsingFPGA: My-UIT-Students/VIPUsingFPGA: Video And Image Processing Using FPGA (github.com) (Truy cập lần cuối: 25/7/2021).

Softmax Function: Softmax Function Definition | DeepAI (Truy cập lần cuối: 25/7/2021).

Models & Training Tutorial: models/research/slim at master · tensorflow/models (github.com) (Truy cập lần cuối: 25/7/2021).

Python Full Course: Python Full Course (***Free***) - YouTube (Truy cập lần cuối: 25/7/2021).

Deep Learning: Đắm mình vào học sâu — Đắm mình vào Học Sâu 0.14.4 documentation (aivivn.com) (Truy cập lần cuối: 25/7/2021).

Imagenet classes: https://zodoc.tech/posts/en/imagenet_classes (Truy cập lần cuối: 25/7/2021).

Những thông tin liên quan:

https://arxiv.org/pdf/1512.00567v3.pdf (Truy cập lần cuối: 25/7/2021).

https://arxiv.org/pdf/1602.07261v2.pdf (Truy cập lần cuối: 25/7/2021).

https://arxiv.org/pdf/1512.03385v1.pdf (Truy cập lần cuối: 25/7/2021).

https://arxiv.org/pdf/1603.05027.pdf (Truy cập lần cuối: 25/7/2021).

https://arxiv.org/pdf/1312.4400.pdf (Truy cập lần cuối: 25/7/2021).

http://www.ele.puc-rio.br/~raul/DL/CNN%20Architectures.pdf (Truy cập lần cuối: 25/7/2021).

http://yann.lecun.com/exdb/publis/pdf/farabet-fpl-09.pdf (Truy cập lần cuối: 25/7/2021).

https://nttuan8.com/bai-6-convolutional-neural-network/ (Truy cập lần cuối: 25/7/2021).

https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning

(Truy cập lần cuối: 25/7/2021).

https://aicurious.io/posts/2019-09-23-cac-ham-kich-hoat-activation-function-trong-neural-networks/ (Truy cập lần cuối: 25/7/2021).

10 CNN architectures: https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d (Truy cập lần cuối: 25/7/2021).

Imagenet Dataset:

https://image-net.org/challenges/LSVRC/2012/browse-synsets (Truy cập lần cuối: 25/7/2021).

https://image-net.org/challenges/LSVRC/2012/ (Truy cập lần cuối: 25/7/2021).

Chương 3: Tổng Kết

Tuy rằng chưa thể hoàn thành được toàn bộ đồ án môn học, nhưng nhóm chúng em cũng đã học hỏi được thêm rất nhiều kiến thức. Ngoài những kiến thức về tổ chức cấu trúc và lập trình với ngôn ngữ phần cứng Verilog HDL, còn có thêm kiến thức về ngôn ngữ Python và cách sử dụng các phần mềm ứng dụng hữu ích khác. Bên cạnh đó, nhóm chúng em cũng hiểu thêm về cách hoạt động của các mô hình mạng nơ ron tích chập và các ứng dụng của chúng.

Quá trình thực hiện đồ án cũng giúp cải thiện các kĩ năng mềm của các thành viên trong nhóm như: tinh thần và trách nhiệm làm việc nhóm, tìm và đọc hiểu các tài liệu tiếng anh, thiết kế bản trình chiếu và viết báo cáo, …

Trong tương lai, nhóm chúng em sẽ cố gắng hoàn thiện và thực nghiệm trên mô hình mạng để có sự so sánh về độ chính xác giữa việc hoàn thiện mạng bằng Python với việc hoàn thiện bằng Verilog.

Nhóm chúng em xin được cảm ơn thầy Trương Văn Cương vì những kiến thức, kinh nghiệm, sự hỗ trợ trong suốt quá trình môn học cũng như quá trình thực hiện đồ án mà thầy đã truyền dạy cho nhóm.

THE END!