**CNN IP**

**Design Document**

by Lee Seung Yeon

**Table of Contents**

1. Introduction 2

**1.1.** **What’s CNN?**  **3**

**1.2.** **Purpose**  **3**

2. Design Considerations 5

**2.1.** **Design Overview** **5**

**2.2.** **Approach** **5**

**2.3.** **About Interface** **6**

**2.4.** **Memory map** **8**

3. Architectural Strategies 9

**3.1.** **State diagram** **9**

1. Introduction

1.1 What’s cnn?

**Definition**

The name “convolutional neural network” indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers

**Design**

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a RELU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution. The final convolution, in turn, often involves backpropagation in order to more accurately weight the end product.

Though the layers are colloquially referred to as convolutions, this is only by convention. Mathematically, it is technically a sliding dot product or cross-correlation. This has significance for the indices in the matrix, in that it affects how weight is determined at a specific index point.

**Convolutional**

When programming a CNN, each convolutional layer within a neural network should have the following attributes:

Input is a tensor with shape (number of images) x (image width) x (image height) x (image depth).

Convolutional kernels whose width and height are hyper-parameters, and whose depth must be equal to that of the image. Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus.

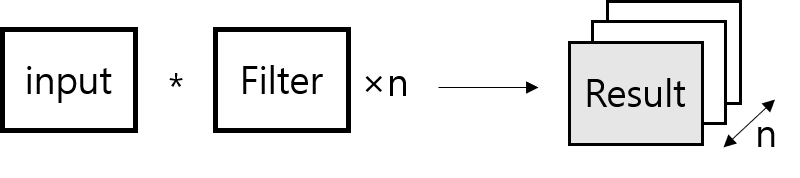
Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters. For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters. In this way, it resolves the vanishing or exploding gradients problem in training traditional multi-layer neural networks with many layers by using backpropagation.

**Pooling**

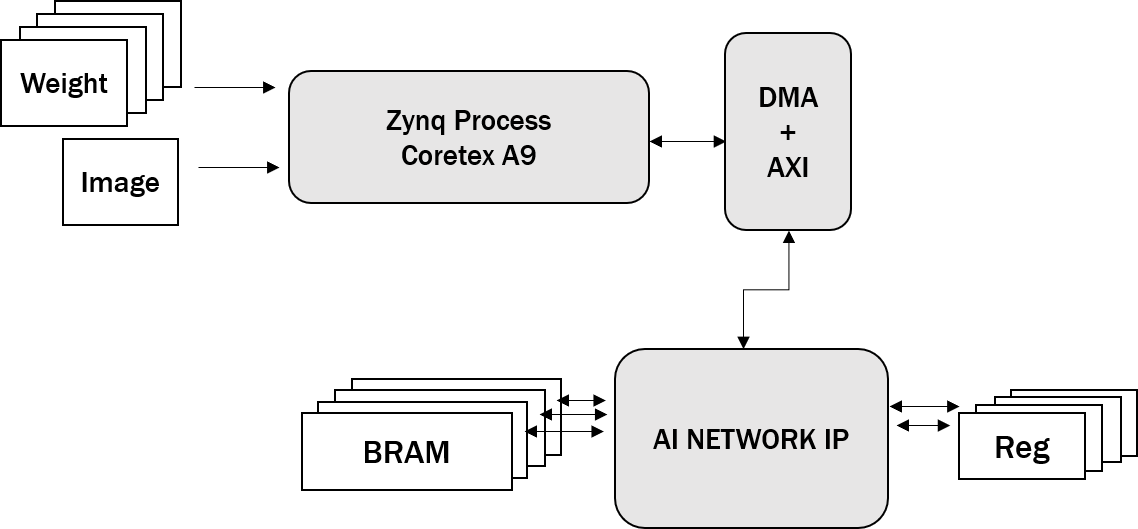
Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer. In addition, pooling may compute a max or an average. (but my design only include Max pooling.) Max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Average pooling uses the average value from each of a cluster of neurons at the prior layer.

1.2 purpose

We have convolution mono channel with multiple depth.



The top module is configured like this.

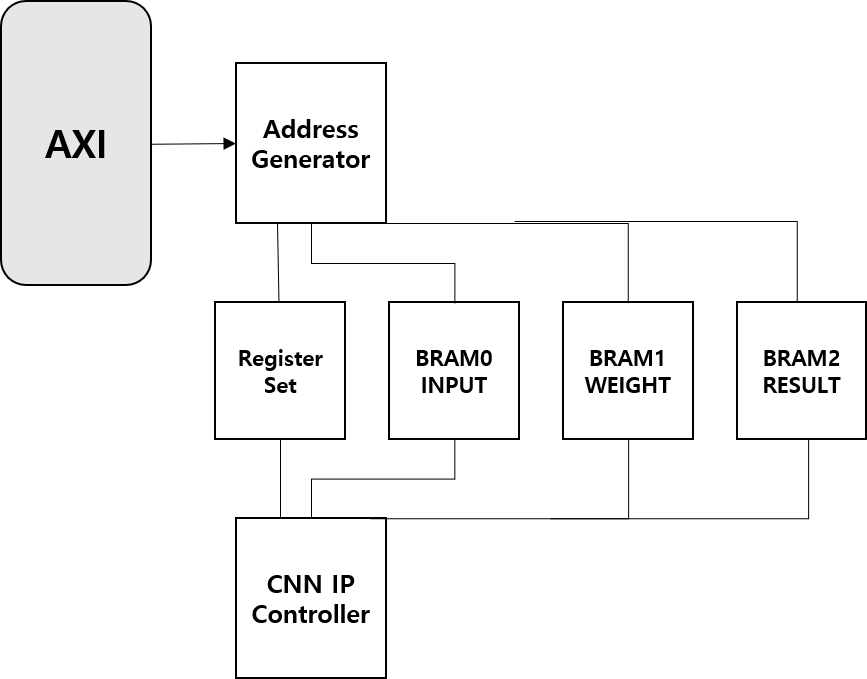


The role of DMA is to reduce the ARM CORE load using an interrupts method.

DMA is needed when there are multiple channels. My design has designed a single channel, not multiple channels.

2. Design Considerations

2.1 Design Overview



2.2 Approach

Convolution Design Plan

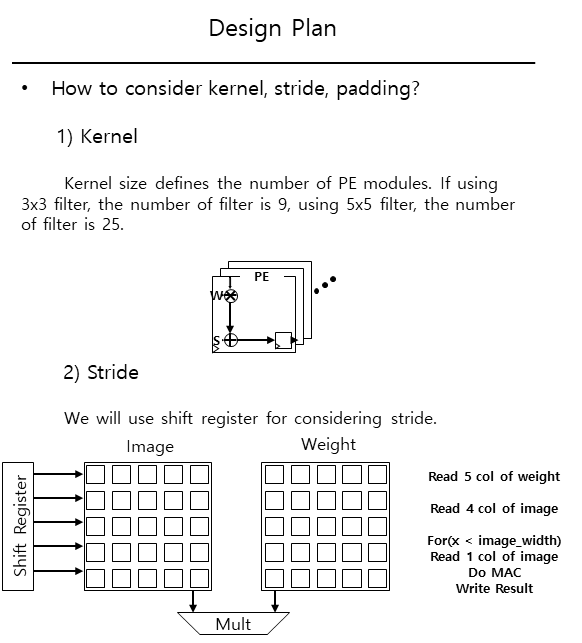
※ How to consider kernel, stride padding?

① Kernel

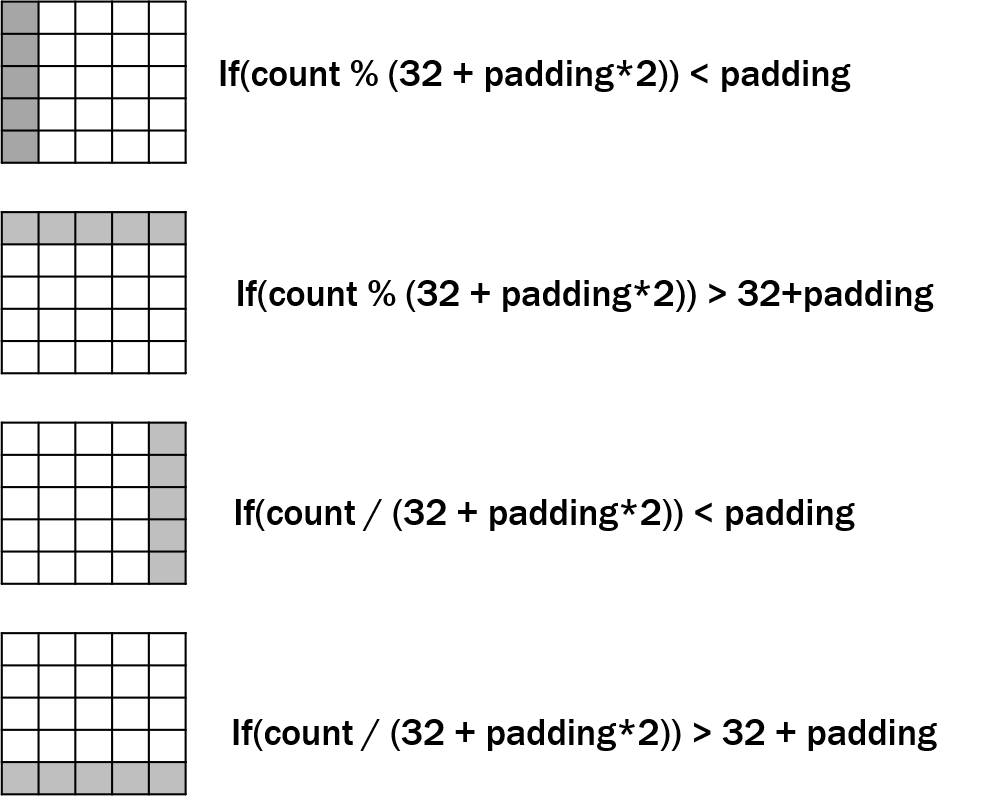
kernel size(filter size) defines the number of PE modules. and We have to consider two situations: when the filter is 3×3 and when the filter is 5×5. If using 3×3 filter, The number of filters is 9, using 5×5 filter, the number of filter is 25.

② Stride

we will use shift register for considering stride.



③ Padding



2.3 About INTERFACE

① AXI INTERFACE

AXI4 interface consist of five different channels:

• Read Address Channel

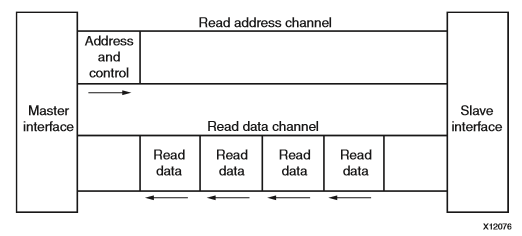
• Write Address Channel

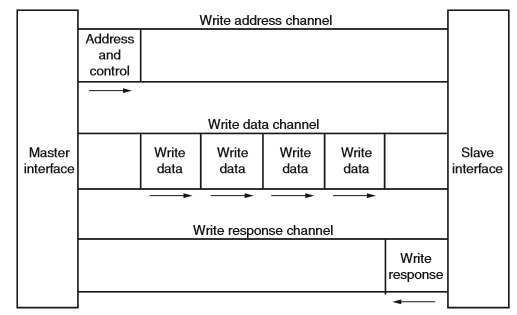
• Read Data Channel

• Write Data Channel

• Write Response Channel

Data can move in both directions between the master and slave simultaneously, and data transfer sizes can vary.

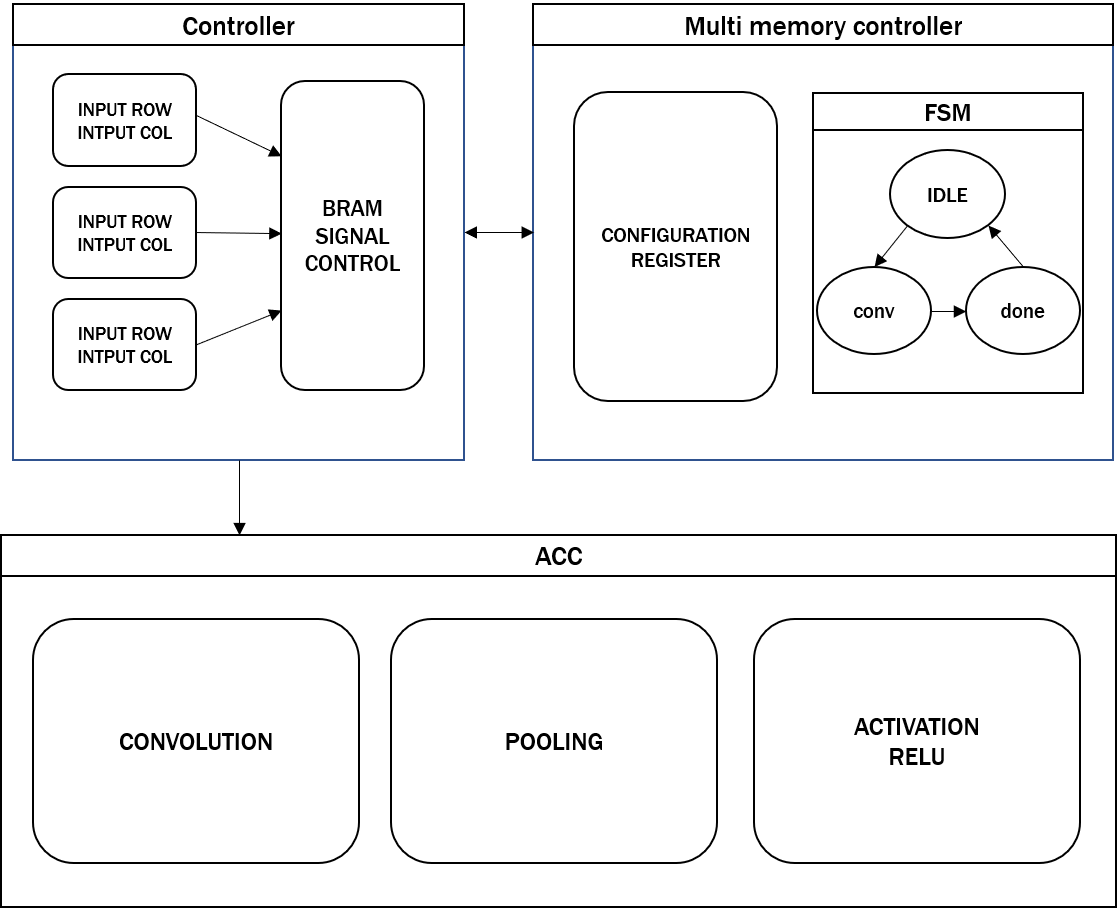




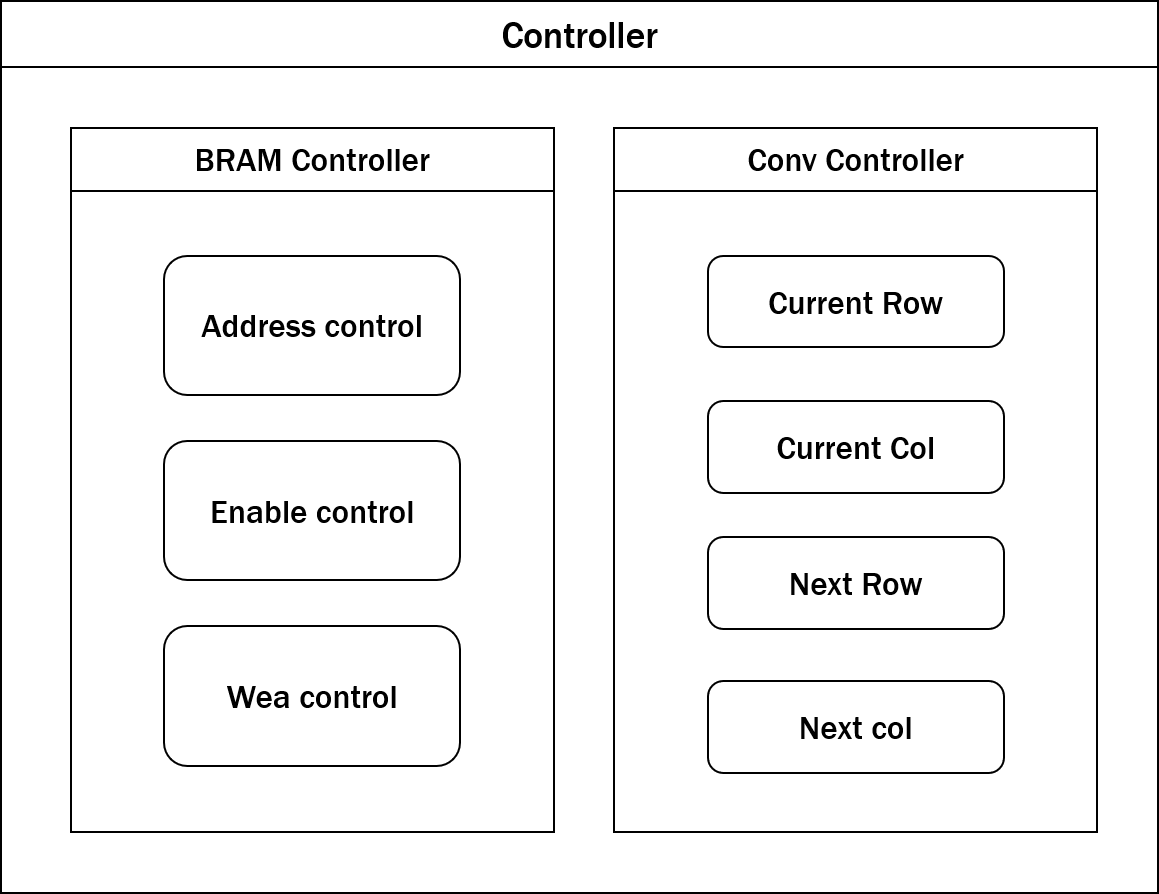
② ADDRESS GENERATOR

The role of the Address Generator is to connect to the register, bram0, bram1 or bram2 according to the address value.

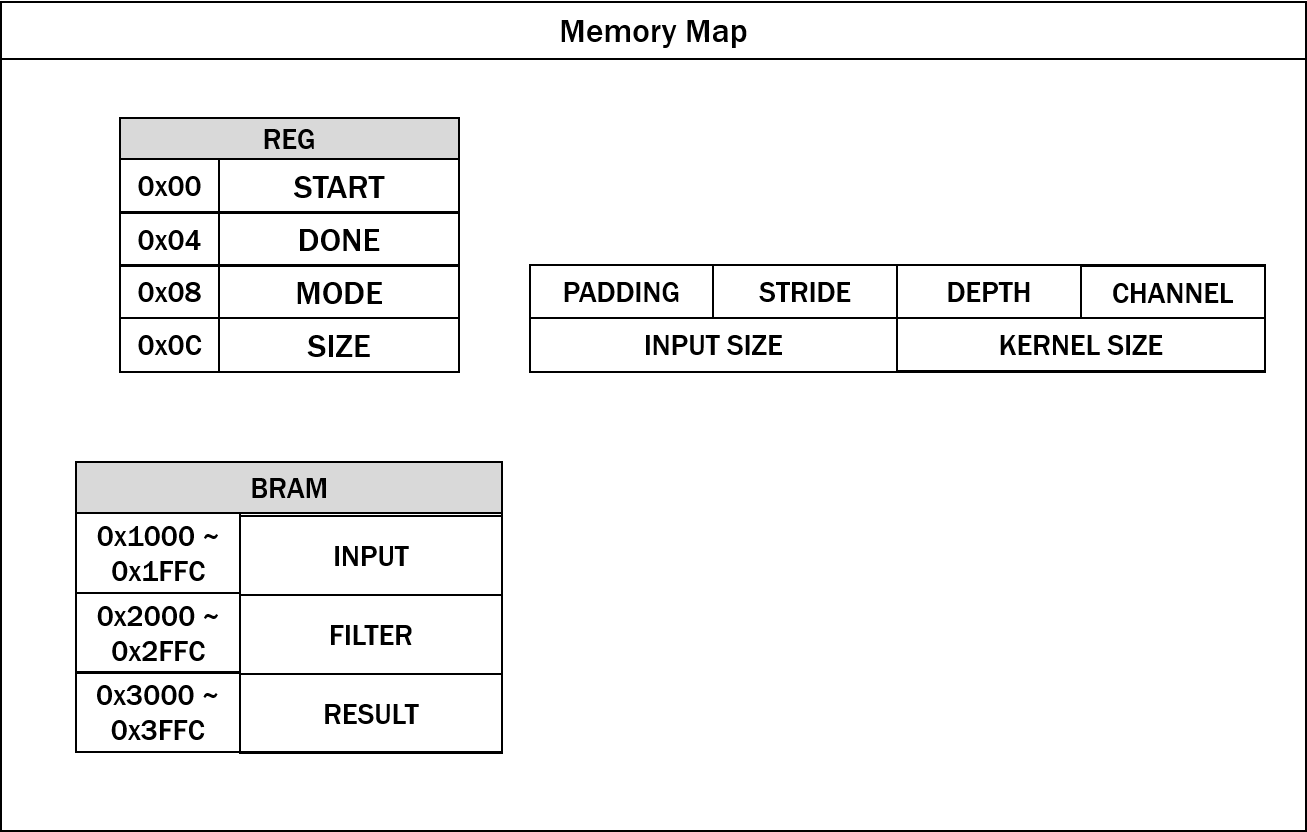
③ CNN IP CONTROLLER



I have illustrated the configuration of the controller. The picture below shows a simple controller configuration.



2.4 Memory Map

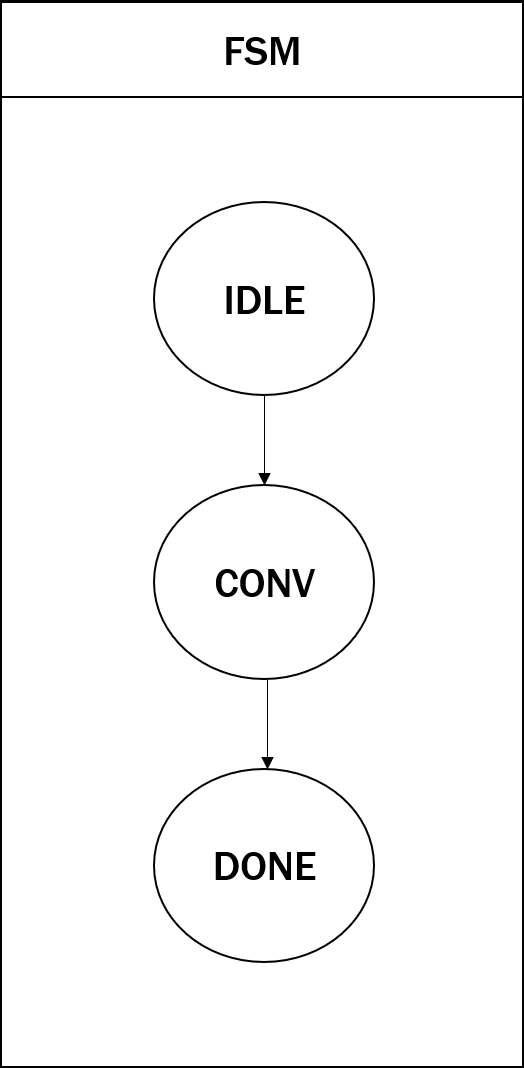


It's an Memory map I thought. The register contains signals(start and done) and information about padding, stride, channel, depth and input size and kernel size. BRAM should be able to read and write information. The storage space of the BRAM was distributed according to the address value, and each address was used to organize a memory map containing information of input , filter, and result.

3. Architectural Strategies

3.1. state diagram

The FSM of the controller that I thought was below.



My controller can be divided into three main states. The three states are divided into idle, conv, done, and status. In the idle state, the BRAM is accessible from the SDK. Enter the input and filter values into the BRAM. Next, in the CONV state, the value of REG0 goes from idle state to run state when it reaches 1. In this state, the BRAM signal is controlled from IP. After completion of the Convolution process, the signal is applied to REG1 (DONE signal) and goes from IDLE to DONE. Finally, in the DONE state, REG1(DONE signal) is continuously checked in the SDK. When this condition is finished, the system is designed to move from DONE state to IDLE state.