

# Department of Electrical Engineering, IIT Jodhpur

## BTP Report

### November 2022

---

**TITLE : Convolution Neural Network(CNN) implementation On  
FPGA for Object Detection**

Name of the Students :

- 1) Ramresh Meena (B19EE070)**
- 2) Rahul Dhayal (B19EE069)**

Name of the Supervisor :

**Dr. Binod Kumar**

### **Introduction :**

Our B-Tech Project is to implement a neural network on FPGA (Field Programmable Gate Arrays) using CNN model (Convolution Neural Network). This project aims to design a system to find the number of times the test pattern appears in the input image. Given a test image and a test pattern, the designed system can successfully detect the pattern and the number of times it occurs in the test image.

Convolutional Neural Network (CNN) is used for implementing the framework of the system. The layers of CNN used are Convolutional Layer, Max Pooling Layer.

### **MOTIVATION :**

---

---

The main goal of object detection is to scan digital images or real-life scenarios to locate instances of every object, separate them, and analyze their necessary features for real-time predictions. Object detection is a part of the overall data architecture of a company. Why is there a need for object detection? The main purpose of object detection is to identify and locate one or more effective targets from still image or video data. It comprehensively includes a variety of important techniques, such as image processing, pattern recognition, artificial intelligence and machine learning.

Object detection is a computer vision technique that allows us to identify and locate objects in an image or video. With this kind of identification and localization, object detection can be used to **count objects in a scene and determine and track their precise locations, all while accurately labeling them.**

This project is based on object detection using the **Convolution Neural Network** (CNN) on FPGA in Verilog. We already had experience in working with Verilog language, Python language and Machine learning. We had a good knowledge of working in both fields because we have already worked with both ML and Verilog in our design credit Projects. So we want to continue our flaws.

Our project's main task was to detect a pattern in an image that shows how many times a particular pattern is occurring in the image and then showing this on an FPGA using PYNQ Z2 Board.

## **METHODOLOGY :**

### **Converting image dataset and pattern into binary text file -**

For this part, first we converted the image and pattern into a binary .txt file using python code.

---

### Details of dataset:

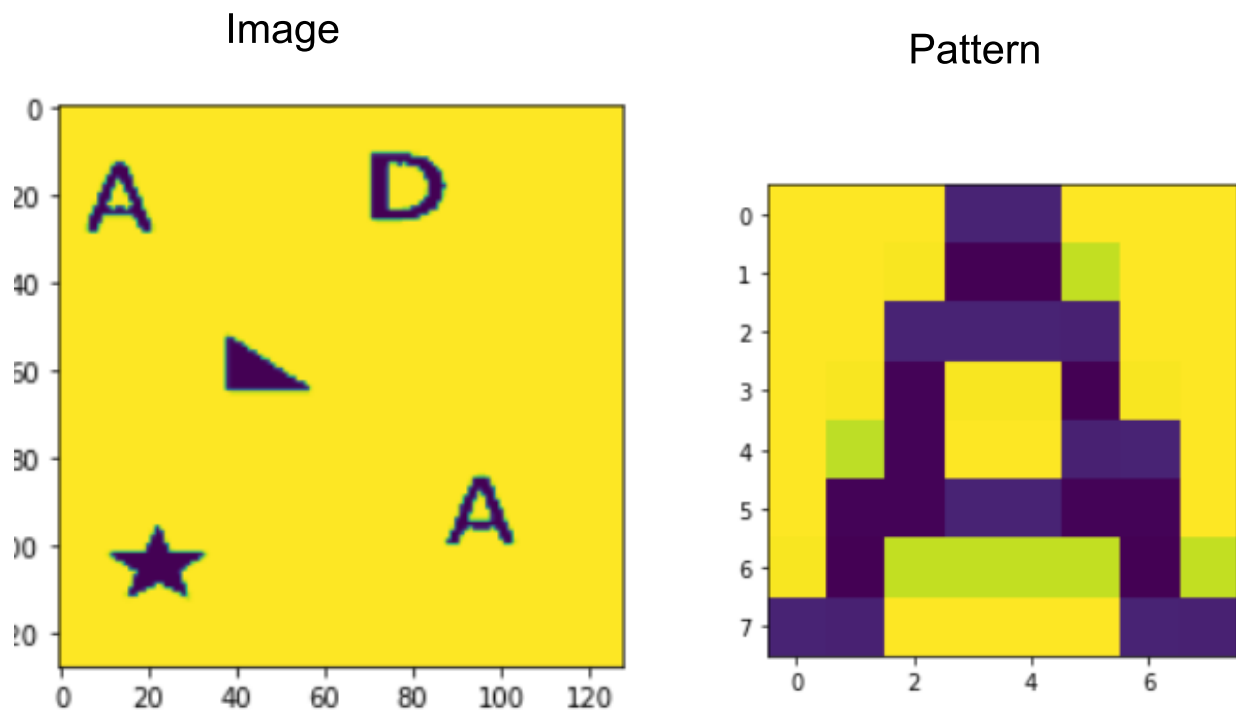
Image size: 128x128

Pattern size: 8x8

Color space: RGB

File format: JPG

Below is given one such Image with different patterns and next to is given the actual Pattern that we are going to detect in the given image.



---

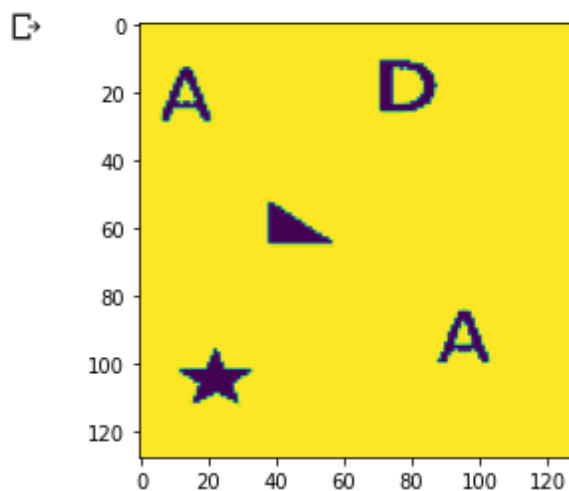
Python Code from RGB to Binary is given below :

---

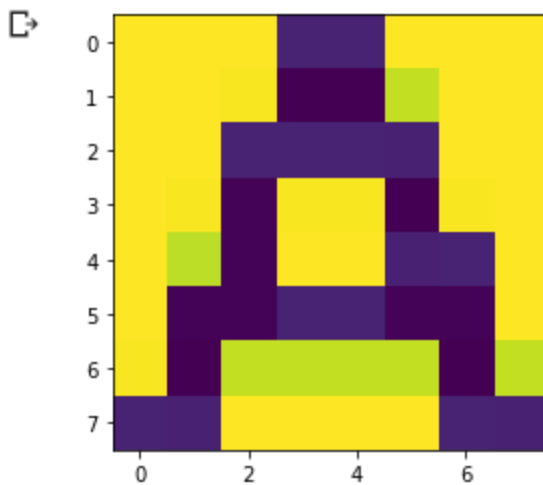
✓ 0s [2] `import numpy as np`  
`import pandas as pd`  
`import cv2`  
`import sys`  
`import io`  
`from matplotlib import pyplot as plt`  
`from google.colab.patches import cv2_imshow`

✓ 0s [3] `img = cv2.imread('Image.png', 2)`  
`ret, bw_img = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)`  
`bw = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)`  
  
`img = cv2.imread('ptn.png', 2)`  
`ret, bw_img = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)`  
`bw = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)`

✓ 0s  `import numpy as np`  
`from matplotlib import pylab as plt`  
`A = np.fromfile("/content/image1.txt", dtype='int16', sep="\n")`  
`A = A.reshape([128, 128])`  
`plt.imshow(A)`  
`import matplotlib.image`  
`matplotlib.image.imsave('image.png', A)`



```
plt.imshow(A)
import matplotlib.image
matplotlib.image.imsave('ptn.png', A)
```



```
X=[]
for i in img:
    X.append(i)

import struct
def float_to_bin(num):
    bits, = struct.unpack('!I', struct.pack('!f', num))
    return "{:04b}".format(bits)
```

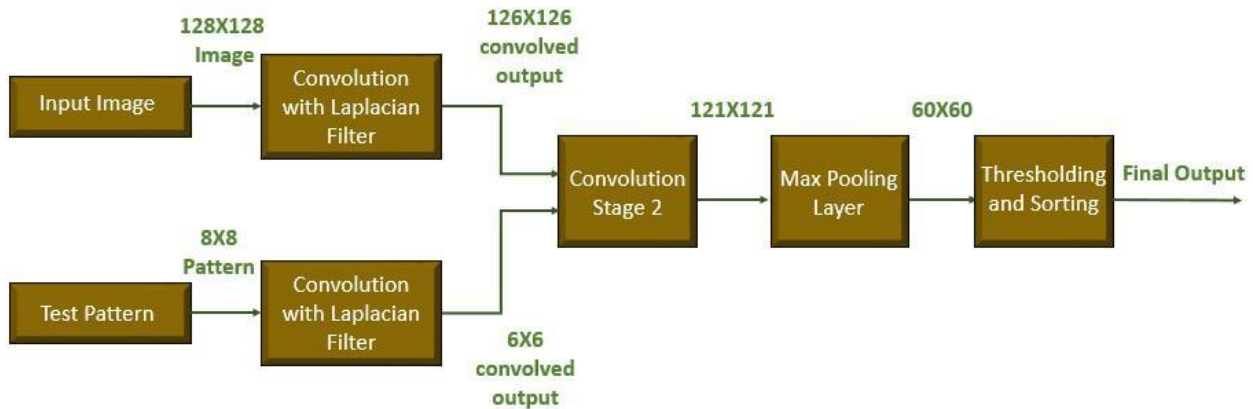
```
[15] bin_image=[]
for i in range(len(img)):
    temp=[]
    for j in range(len(img[0])):
        num=img[i][j]
        binary=float_to_bin(num)
        temp.append(binary)
    bin_image.append(temp)

df=pd.DataFrame(bin_image,index=None)
bin_image=np.array(bin_image)
df.to_csv("bin_img.txt",sep=" ",header=None,index=None)
```

---

## Implementation :

### Block Diagram -



### Methodology -

1. **Input Image convolved with the Laplacian filter** - The size of the image is (128X128). It is convolved with the Laplacian Filter to get a matrix of 126X126 dimensions.

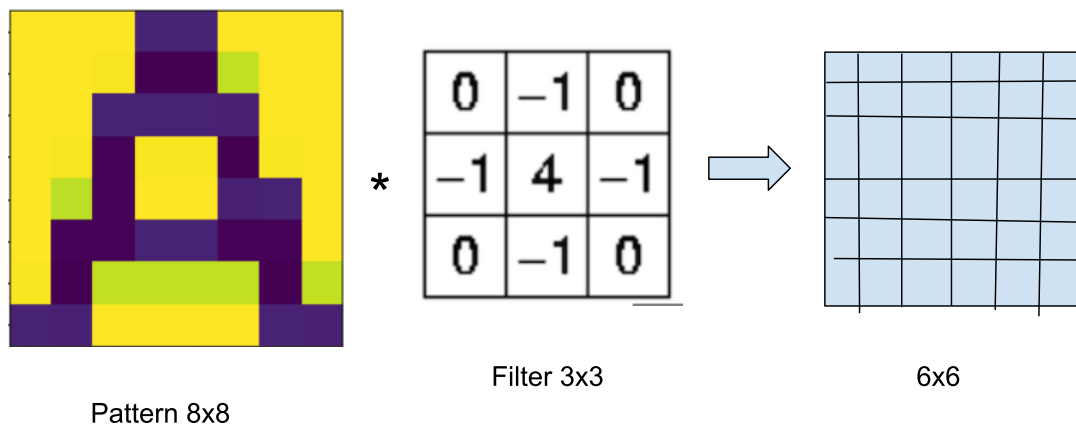
Laplacian filter is a second-order derivative filter used in **edge detection**, in digital image processing. using the Laplacian filter we detect the edges in the whole image at once.

The Laplacian Operator/Filter is =  $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ , here the central value of the filter is negative.

Or

Filter is =  $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ , here the central value of filter is positive. The sum of all values of the filter is always 0.

- 
2. **Test Pattern convolved with the Laplacian Filter** - The size of the Pattern to be detected is (8X8). It is convolved with the Laplacian Filter to get a matrix of 6X6.



3. **Convolution of the Matrices obtained from step (1) and step (2)** - The output matrices from step (1) and step (2) are convolved with each other. The output matrix obtained is of the dimension 121X121.

4. **Max Pooling on the Output of step (3)** - The factor used for Max Pooling is 2. The output matrix obtained from this layer is of the size 60X60.

The pooling operation involves sliding a two-dimensional filter over each channel of the feature map and summarizing the features lying within the region covered by the filter. Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

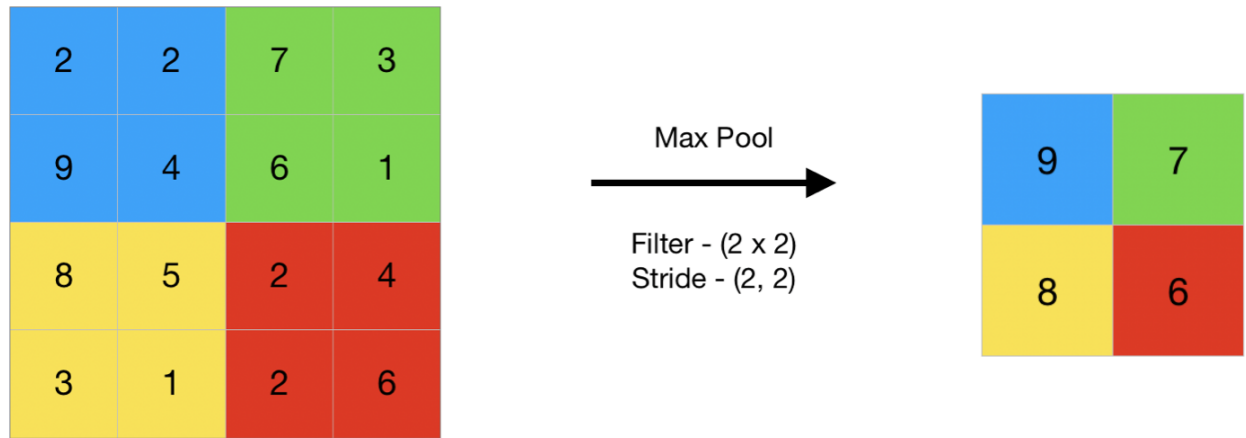
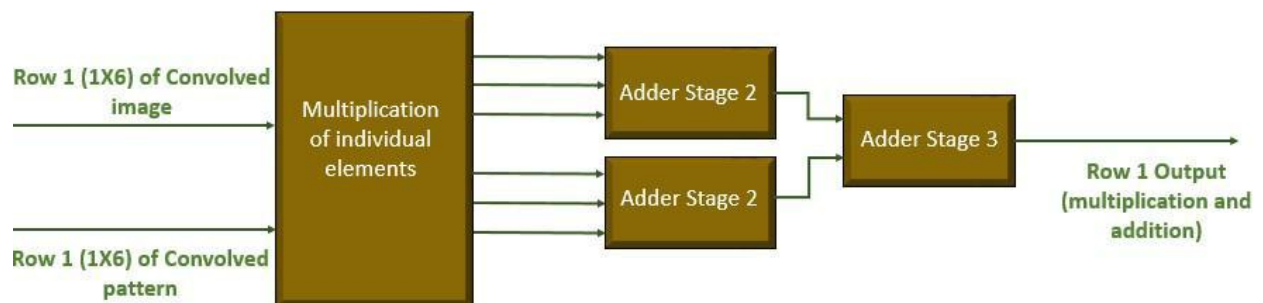


Image source : [Geeksforgeeks.com](https://www.geeksforgeeks.com/)

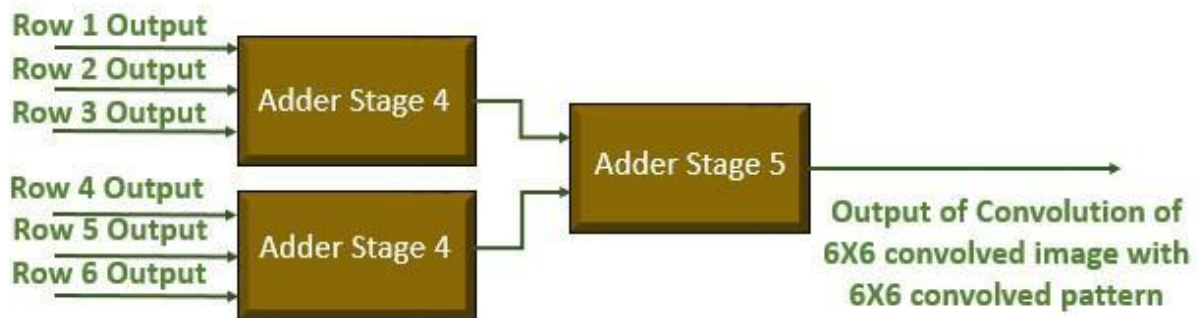
5. **Thresholding and Counting number of times the pattern is detected** - In the final step, using the thresholding, each element in the matrix obtained at step (4) is compared with the value of convolution of the pattern with itself. Values greater than 25% are considered a matched pattern.

#### A. Convolution with laplacian filter (first stage) -

Below block diagrams represent the pipelined implementation of the stage 2 of convolution of outputs obtained at stage 1

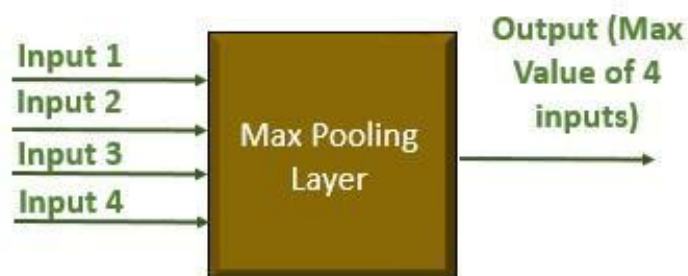






In the pipelined implementation, same numbered rows are given as input to the multiplication block, which gives 6 output and addition is also performed in a pipelined manner. Overall, a 6X6 convolution operation takes 5 clock cycles, because of use of pipelining and hardware parallelism. Once an output is obtained, each clock cycle produces one new output.

#### B. Max Pooling Layer (Stage 3) -

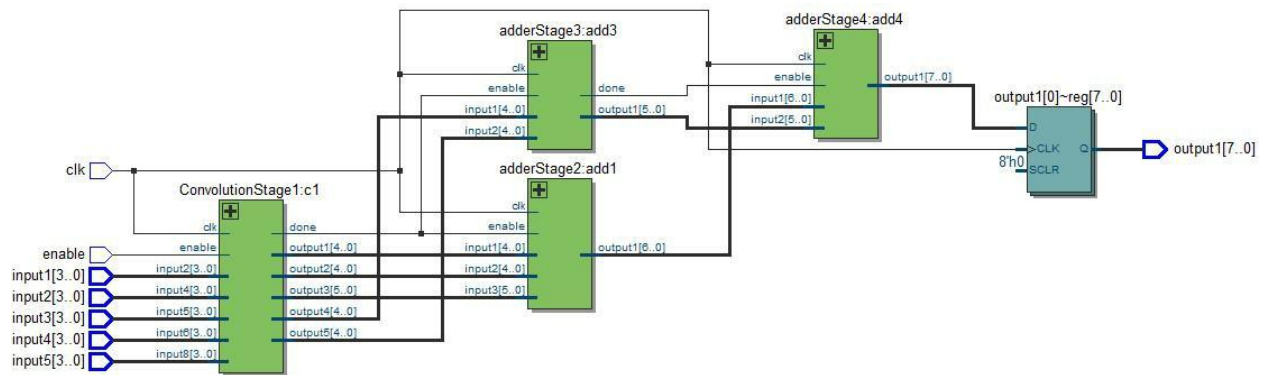


#### C. Thresholding -

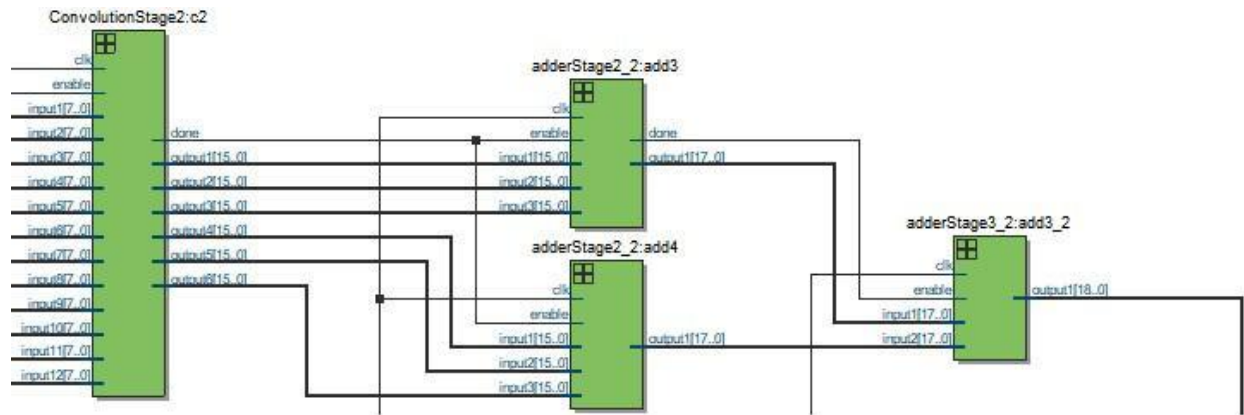
In the direct layer, instead of sorting and then comparing the values for finding the number of times a pattern appears in the image, direct comparison is done to avoid the time-consuming activity of sorting.

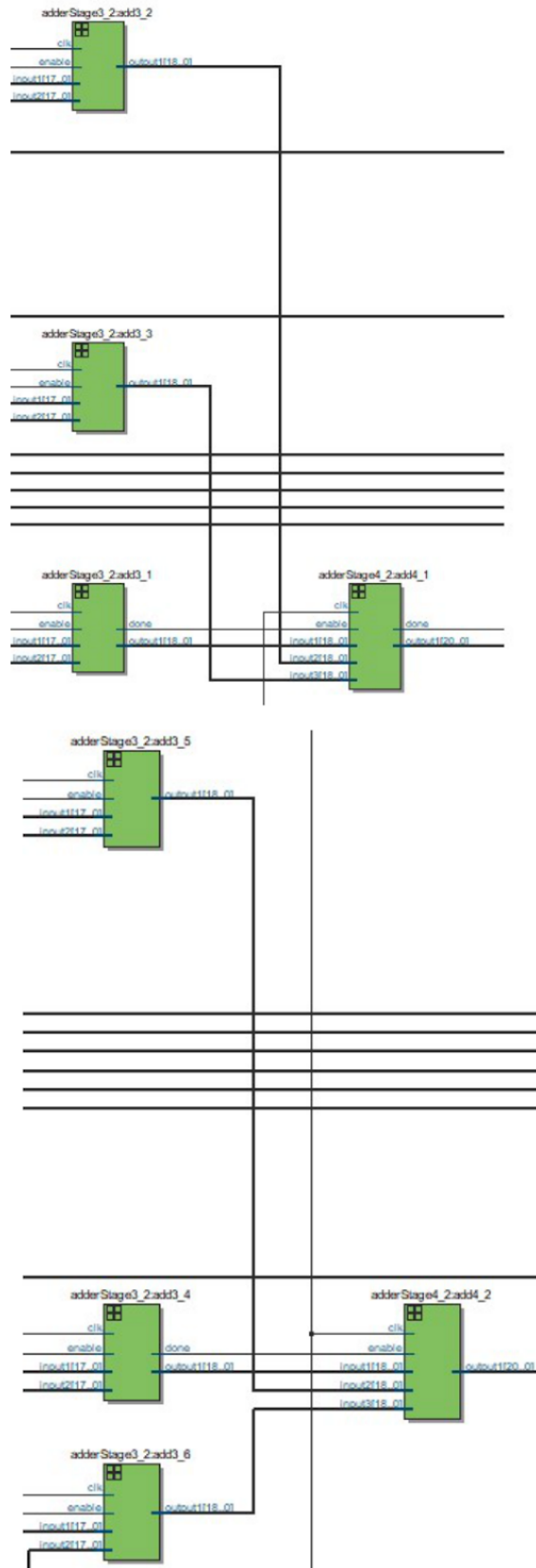
## RTL netlist for individual blocks -

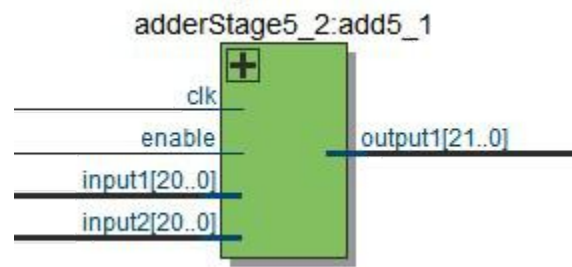
### Convolution (First Stage)



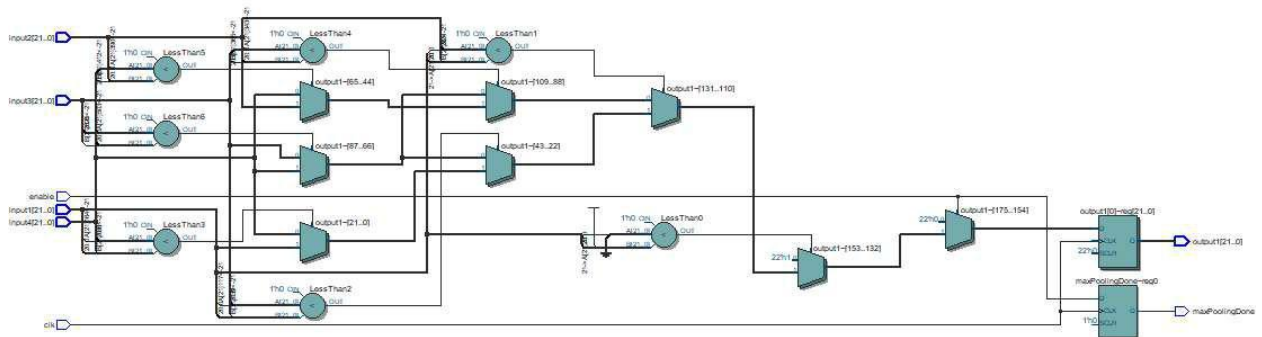
### Convolution (Second Stage)







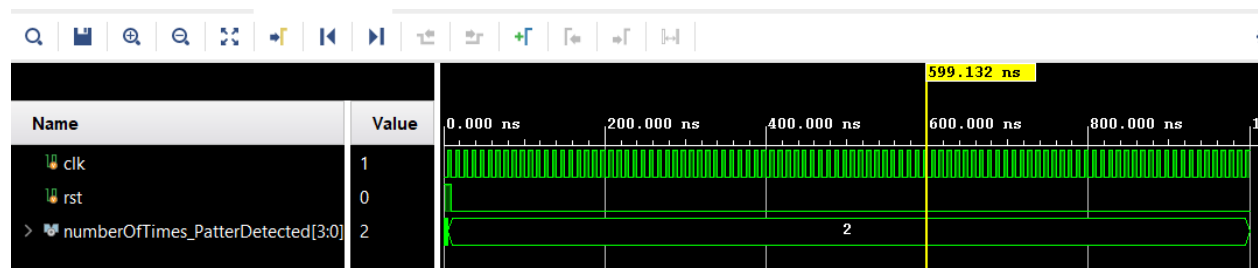
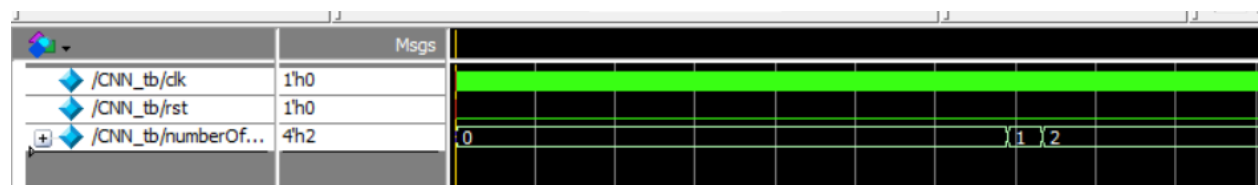
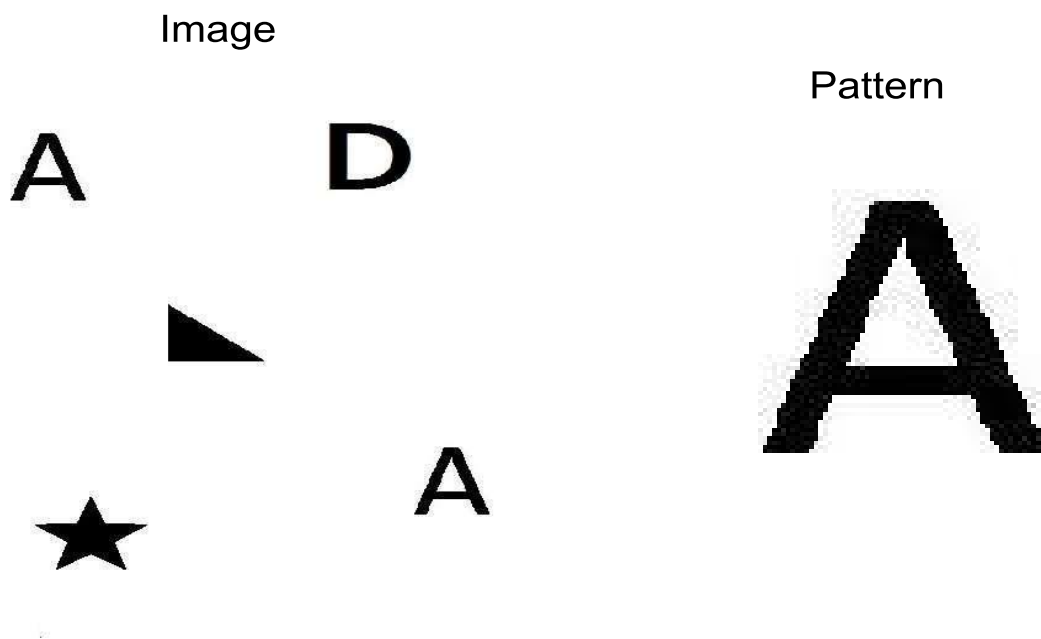
Max pooling layer :



## Results -

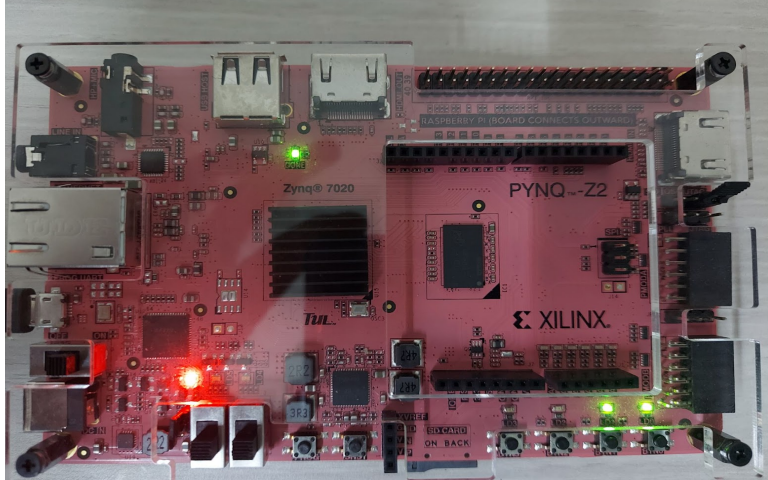
### Test - 1 :

Pattern A is occurring two times in the Image given below so the output of the simulation must be 2.



Output from simulation(number of times the pattern appears in the image is 2)

PynQ Z2 Board Result -

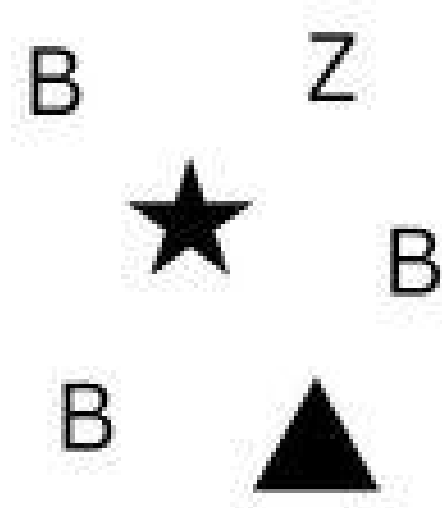


Two LEDs are glowing because the pattern is detected 2 times.

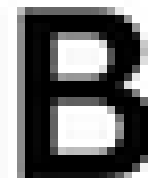
**test - 2:**

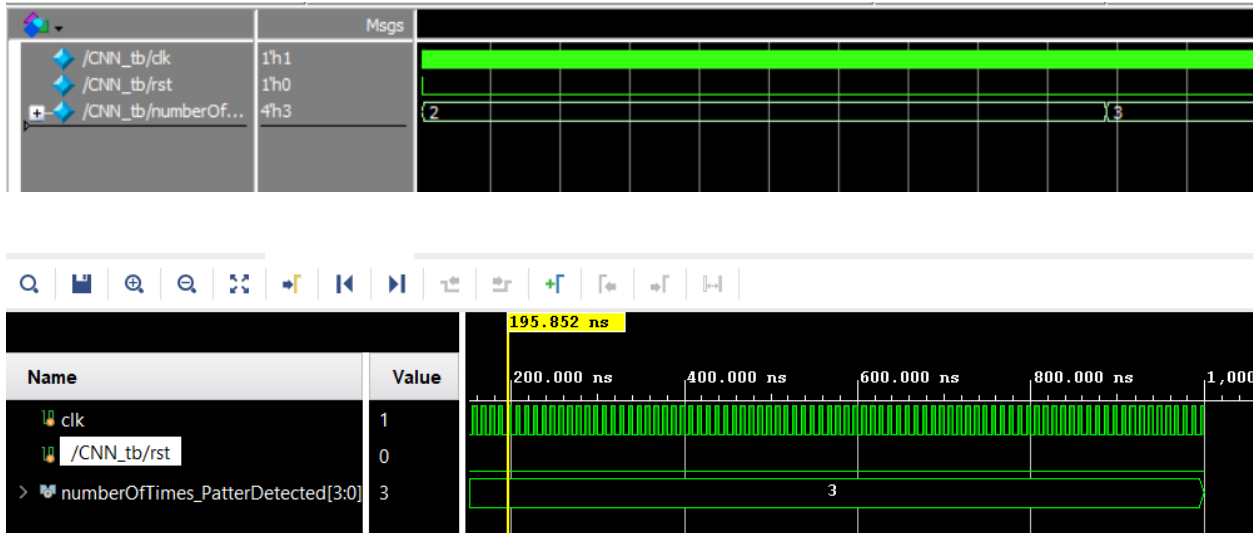
Pattern B is occurring three times in the Image given below so the output of the simulation must be 3.

Image



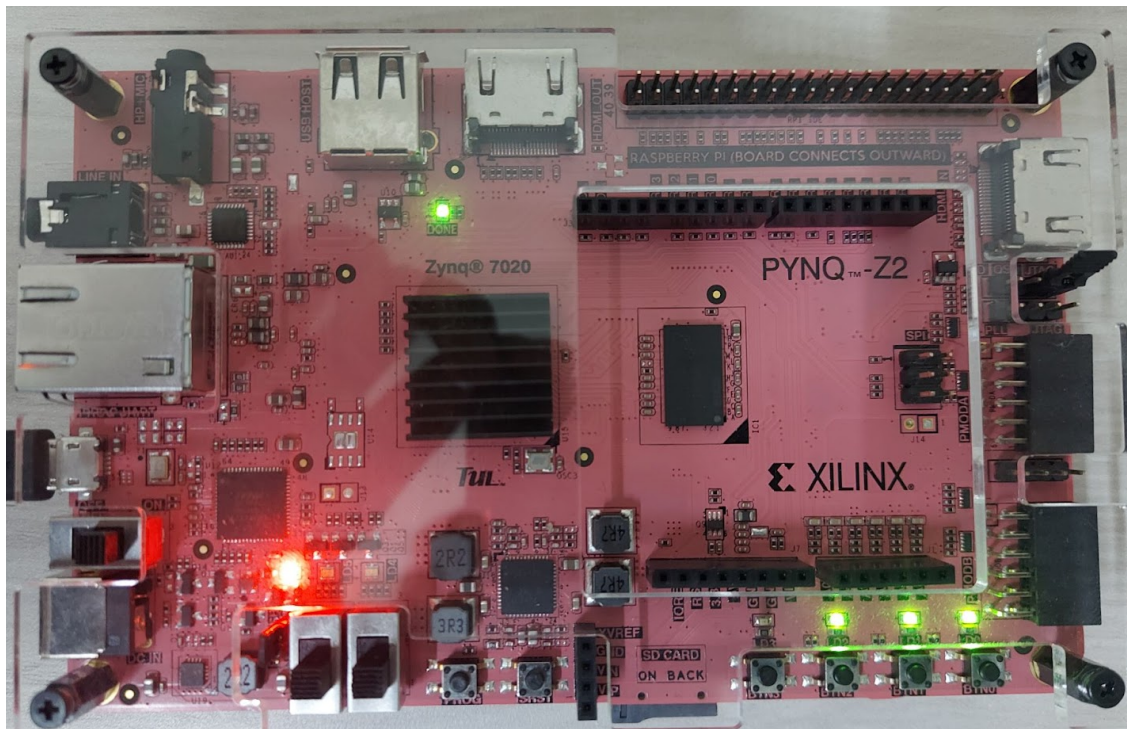
Pattern





Output from simulation (number of times the pattern appears in the image is 3)

PynQ Z2 Board result



Three LEDs are glowing or Blinking because the pattern was detected three times.

---

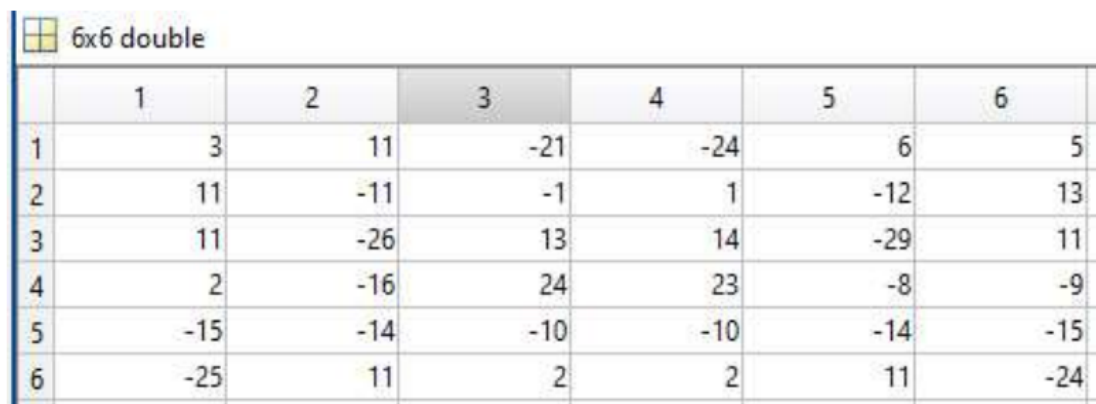
## Conclusion :

The results shown above verifies that using CNN we can detect the number of times a particular pattern is occurring in a given image.

Using the Pynq Z2 Board we can implement the same on the FPGA. on the board LED same number of LED as the number of times pattern detected will be Blown when Button is pressed

We can also verify results from Matlab using Laplacian filter matlab code. Comparison of outputs with results MATLAB implementation and RTL implementation (with example of convolution of Pattern with Laplacian Filter). As can be observed the results Below are exactly matching, with no error.

### Results from Matlab implementation -

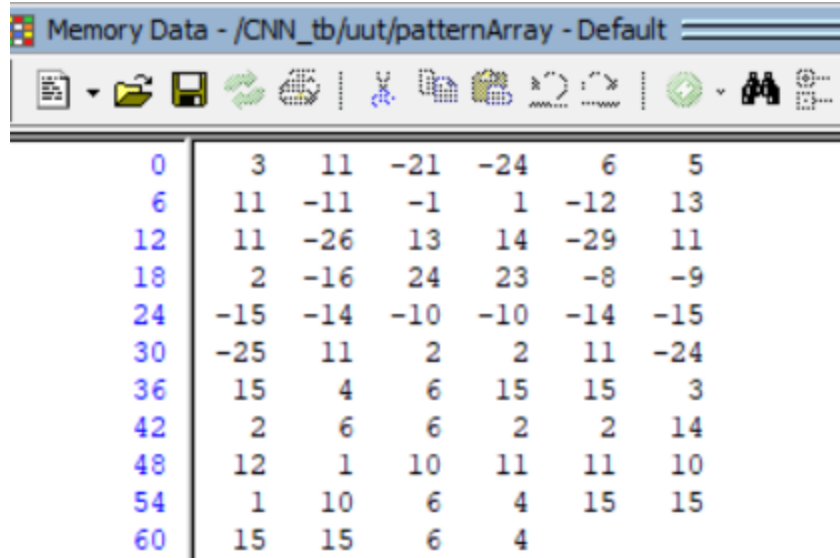


A screenshot of a MATLAB window showing a 6x6 double matrix. The matrix is displayed in a table with columns numbered 1 to 6 and rows numbered 1 to 6. The values are as follows:

	1	2	3	4	5	6
1	3	11	-21	-24	6	5
2	11	-11	-1	1	-12	13
3	11	-26	13	14	-29	11
4	2	-16	24	23	-8	-9
5	-15	-14	-10	-10	-14	-15
6	-25	11	2	2	11	-24

### Simulation Result from RTL implementation -





	0	6	12	18	24	30	36	42	48	54	60
0	3	11	-21	-24	6	5					
6	11	-11	-1	1	-12	13					
12	11	-26	13	14	-29	11					
18	2	-16	24	23	-8	-9					
24	-15	-14	-10	-10	-14	-15					
30	-25	11	2	2	11	-24					
36	15	4	6	15	15	3					
42	2	6	6	2	2	14					
48	12	1	10	11	11	10					
54	1	10	6	4	15	15					
60	15	15	6	4							

As it can be observed from above two tables that are matching that means convolution is happening without any error.

#### **Clock Cycles Required for each Stage -**

For all stages	37722
For Convolution of Image (128X128) with Laplacian Filter (3X3)	15880
For Convolution of pattern (8X8) with Laplacian Filter (3X3)	400
For Convolution of convolved image (126X126) with convolved pattern (6X6)	14647
For Max Pooling Layer (121X121)	3602
For Thresholding (direct layer)	3600

#### **Maximum clock speed achievable 155.67 MHz**

Verilog code link :

[https://drive.google.com/drive/folders/1mbTyqJxzbHI-QZqhAkSy8xY7CeMnomOT?usp=share\\_link](https://drive.google.com/drive/folders/1mbTyqJxzbHI-QZqhAkSy8xY7CeMnomOT?usp=share_link)

---

## References :

<https://blog.umer-farooq.com/a-pynq-z2-guide-for-absolute-dummies-part-ii-using-verilog-and-vivado-to-burn-code-on-pynq-d856f79948b1>

<https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/#:~:text=Max%20pooling%20is%20a%20pooling,of%20the%20previous%20feature%20map.>

<https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network>

<https://in.mathworks.com/matlabcentral/answers/383861-developing-laplacian-filter-and-apply-it-to-an-image>

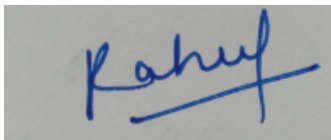
### SIGNATURES

#### Students

1. Ramresh Meena



2. Rahul Dhayal



#### Supervisors

1. Dr. Binod Kumar

