Table of Contents

[Abstract 4](#_Toc42959907)

[1.Introduction 5](#_Toc42959908)

[1.1. Introduction 5](#_Toc42959909)

[1.2. History and Background 5](#_Toc42959910)

[1.3. Problem Definition 9](#_Toc42959911)

[1.4. Objective 9](#_Toc42959912)

[1.5. Related Work 12](#_Toc42959913)

[2.The proposed solution and Methodology 12](#_Toc42959914)

[2.1. Proposed Solution/technique/architecture: architecture or flowchart 12](#_Toc42959915)

[2.2. Methodology: flowchart or block-diagram or steps 13](#_Toc42959916)

[2.2.1 Software Solution 14](#_Toc42959917)

[2.2.2 Hardware Solution 15](#_Toc42959918)

[3. Implementation 17](#_Toc42959919)

[3.1. Software implementation 17](#_Toc42959920)

[Folder structure 17](#_Toc42959921)

[3.2. Hardware implementation 18](#_Toc42959922)

[3.2.1 Convolution Layer 18](#_Toc42959923)

[3.2.2 Padding 20](#_Toc42959924)

[3.2.3 Max Pooling Layer 21](#_Toc42959925)

[3.2.4 ReLU Layer 24](#_Toc42959926)

[3.2.5 Integration of padding and convolution 26](#_Toc42959927)

[3.2.6 Integration of (Padding-Convolution-Maxpooling) 26](#_Toc42959928)

[3.3. An illustrative example 26](#_Toc42959929)

[4.Validation and results 30](#_Toc42959930)

[4.1. Software experimental Results 30](#_Toc42959931)

[4.2. Hardware experimental Results 32](#_Toc42959932)

[4.2.1 Convolution Layer 32](#_Toc42959933)

[4.2.2 Padding 35](#_Toc42959934)

[4.2.3 Max Pooling Layer 37](#_Toc42959935)

[4.2.4 ReLU Layer 40](#_Toc42959936)

[4.2.5 Integration of padding and convolution 42](#_Toc42959937)

[*4.2.6 Integration of (Padding-Convolution-Maxpooling)* 43](#_Toc42959938)

[4.3. Case studies 43](#_Toc42959939)

[4.4. Performance evaluation 43](#_Toc42959940)

[4.5. Performance metrics 44](#_Toc42959941)

[4.5.1 Convolution Layer 44](#_Toc42959942)

[4.5.2 Padding Layer 45](#_Toc42959943)

[4.5.3 Integration of padding and convolution 46](#_Toc42959944)

[4.5.4 Integration of (Padding-Convolution-Maxpooling) 46](#_Toc42959945)

[4.6. Comparison with related work 48](#_Toc42959946)

[4.7. Conventional versus proposed 48](#_Toc42959947)

[4.8. Others work versus proposed 49](#_Toc42959948)

[4.9. Limitations 49](#_Toc42959949)

[4.10. Demo (if applicable) ~ 3 min 49](#_Toc42959950)

[link 49](#_Toc42959951)

[5 .Conclusions 49](#_Toc42959952)

[References 50](#_Toc42959953)

# Abstract

This project is mainly for implementing an artificial intelligent neural network (A neural network is a type of machine learning which consists of a collection of building blocks each is called a neuron, it models itself after the human brain ) it is designed for autonomous driving vehicles, the purpose of this project is to solve many problems that occur in our daily life this includes : traffic problems , car accidents , traffic overflow , avoiding high death rates due to traffic accidents etc.., we studied carefully how a basic neural network operates with the aid of different references and papers through the internet this includes : mathematics of the neural networks , different architectures, algorithms , types and the problems that may occur in any neural network, the results is that we are now aware of the main idea of implementing a neural network, how it works and the main requirements of the design , This will be the first time to implement faster RCNN on hardware as it was always used as software code not hardware components , in conclusion the final architecture to be implemented is typically the Faster RCNN( which is to be implemented on a hardware chip

# 1.Introduction

## 1.1. Introduction

Autonomous cars are very closely associated with Industrial IoT.  IoT combined with other technologies such as machine learning, artificial intelligence, local computing etc are providing the essential technologies for autonomous cars.  Very inquisitive questions for many is how are these autonomous cars functioning. What actually is working inside to make them work without drivers taking control of the wheel.  Very well known that these days cars are equipped with a lot of sensors, actuators, and controllers. These end devices are driven by software sitting on various function-specific software running on ECUs ( Electronic Control Units).  Machine learning software is also part of this set.

One of the main tasks of any machine learning algorithm in the self-driving car is a continuous rendering of the surrounding environment and the prediction of possible changes to those surroundings. These tasks are mainly divided into four sub-tasks:

* Object detection
* Object Identification or recognition Object classification
* Object localization and prediction of movement

## 1.2. History and Background

An autonomous car is a vehicle that can guide itself without human conduction. This kind of vehicle has become a concrete reality and may pave the way for future systems where computers take over the art of driving. An autonomous car is also known as a driverless car, robot car, self-driving car or autonomous vehicle.

**Autonomous Car Components**

The monitoring system of autonomous cars that consists of powerful sensors and LiDAR’s view the objects ahead, with much more clarity, precision and a 360-degree view than the human eye and then chart the route of the car accordingly. While autonomous cars are good at identifying hurdles ahead, it cannot distinguish between pedestrians and inanimate objects.

vehicles? Nevertheless, the metrics of safety considerations cannot overrule autonomous cars because, in any given time and situation, they are statistically safer than human-driven cars, and intelligent AI-powered systems are immune to the foibles and temperamental indiscretions of humans.

What is needed is setting public expectations and gradually erasing doubts that the citizens have. Along with companies like Waymo, governments should also step in to allay the fears and disseminate the advantages of autonomous cars — that range from reduction in fuel consumption, less traffic congestion to the decreased probability of a collision. A framework conducive to emerging enterprises and minimal regulations that promote and incentivize innovation is needed, in addition to a dedicated task force and participatory approach between governments and private industry. No system is engineered to be free from the probability of error margin. The error margin can only be made negligible and then the cause of the error can be identified and rectified with a permanent effect. In autonomous cars, because of extensive research and cutting-edge technical equipment used, the error margin is already very low, and it is possible that it may be obliterated altogether once the test phase is over and the technology incorporates new innovations. Considering the enormous potential of autonomous car technology and the benefits it has in store for the society, to write it off and spell doom is akin to throwing the baby along with the bathwater.

Self-driving cars might seem like a recent thing, but experiments of this nature have been taking place for nearly 100 years. In fact, centuries earlier, Leonardo Da Vinci designed a self-propelling cart hailed by some as the world’s first robot. At the World’s Fair in 1939, a theatrical and industrial designer named [Norman Bel Geddes](https://en.wikipedia.org/wiki/Norman_Bel_Geddes) put forth a ride-on exhibit called Futurama, which depicted a city of the future featuring automated highways. However, the first self-driving cars didn’t arrive until a series of projects in the 80s undertaken by Carnegie Mellon University, Bundeswehr University, and Mercedes-Benz. The [Eureka Prometheus Project](https://en.wikipedia.org/wiki/Eureka_Prometheus_Project) proposed an automated road system in which cars moved independently, with cities linked by vast expressways. At the time, it was the largest R&D project ever in the field of driverless cars.

Since 2013, US states Nevada, Florida, California, and Michigan have all passed laws permitting autonomous cars; more will surely follow. Autonomous vehicle components. Ouster’s OS-1 3D [LiDAR](https://levelfivesupplies.com/glossary/lidar/) sensor captures point cloud data and [camera](https://levelfivesupplies.com/glossary/camera/)-like images to help autonomous vehicles ‘see’ their environment How does an [autonomous vehicle](https://levelfivesupplies.com/glossary/autonomous-vehicle/) operate and make sense of what it sees? It comes down to a powerful combination of technologies, which can be roughly divided into hardware and software. Hardware allows the car to see, move and communicate through a series of cameras, sensors and V2V/V2I technology, while software processes information and informs moment-by-moment decisions, like whether to slow down. If hardware is the human body, software is the brain.

At Level Five Supplies, our tech is broadly categorized as follows:

* [Data storage](https://levelfivesupplies.com/product-category/data-storage/)
* [Drive-by-wire](https://levelfivesupplies.com/product-category/drive-by-wire/)
* [Positioning](https://levelfivesupplies.com/product-category/positioning/)
* [Power](https://levelfivesupplies.com/product-category/power/)
* [Processing](https://levelfivesupplies.com/product-category/processing/)
* [Sensors](https://levelfivesupplies.com/product-category/sensors/)
  + [Camera](https://levelfivesupplies.com/product-category/camera/)
  + [LiDAR](https://levelfivesupplies.com/product-category/lidar/)
  + [Radar](https://levelfivesupplies.com/product-category/radar/)
  + [Ultrasonic](https://levelfivesupplies.com/product-category/ultrasonic/)
* [Software](https://levelfivesupplies.com/product-category/software/)

Autonomous vehicles rely on sophisticated algorithms running on powerful processors. These processors make second-by-second decisions based on real-time data coming from an [assortment of sensors](https://levelfivesupplies.com/sensors-used-in-autonomous-vehicles/). Millions of test miles have refined the technology and driven considerable progress – but there is still a way to go.Driverless car technology companies the race to be the first self-driving car on the road is heating up. [Level 5](https://levelfivesupplies.com/glossary/level-5/) autonomy is still some time away, but there is plenty else happening in the autonomous space, with aspects of driverless tech already making an appearance in today’s mass-produced cars. Early adopters will enjoy benefits such as automatic parking and driving in steady, single-lane traffic.

**Manufacturing Companies**

* Google – presently leading the charge via Waymo, its self-driving subsidiary
* Tesla – models are now being fitted with hardware designed to improve [Tesla Autopilot](https://www.tesla.com/autopilot)
* Baidu – in the process of developing [Level 4](https://levelfivesupplies.com/glossary/level-4/) automated vehicles
* General Motors – developed the first production-ready autonomous vehicle
* Toyota – working with ride-hailing service Uber to bring about autonomous ridesharing
* Nvidia – created the world’s first commercially available [Level 2](https://levelfivesupplies.com/glossary/level-2/)+ system
* Ford – planning to have their fully autonomous vehicle in operation by 2021
* nuTonomy – first company to launch a fleet of self-driving taxis in Singapore
* BMW – has teamed up with Intel and Mobileye to release a fully driverless car by 2021
* Oxbotica – will begin trialing autonomous cars in London in December 2019

Challenges in autonomous vehicle testing and validation

As we can see, there are many different autonomous vehicle projects in various stages of development. But there’s a big difference between creating a test vehicle that’ll run in reasonably tame conditions and building a multi-million strong fleet of cars that can handle the volatility and randomness of the real world. One of the biggest challenges is putting the computer in charge of everything, including exception handling. By exceptions, we mean variable circumstances like bad weather, traffic violations and environmental hazards. Fog, snow, a deer leaping into the road – how does a fully autonomous vehicle interpret and react to these situations? When we take the driver out of the picture completely, automation complexity soars compared to lower-level systems. The software must handle everything. Rigorous testing is essential, but it won’t be enough on its own. Alternative methods such as [simulation](https://levelfivesupplies.com/product/sim-lab-driving-simulator/), bootstrapping, field experience and human reviews will also be necessary to ensure the safety of the vehicle.

For the time being, it means implementing each new capability carefully and gradually: hence why it will be a good few years before we see Level 5 vehicles on the road. Global perceptions of autonomous car technology How does the general public feel about the onset of autonomous cars? Generally, perception is positive, but there’s still a way to go. Not everyone is convinced. The main bone of contention is safety. Autonomous car manufacturers must prove beyond doubt the safety of their vehicles before they can hope for widespread adoption. [One survey](https://www.information-age.com/autonomous-car-technology-china-123470870/) conducted across China, Germany and the US found that drivers want to decide for themselves when to let a car drive autonomously and when to take over. The survey also found that trust in autonomous cars is nearly twice as high in China.[A second survey](https://www.autonomousvehicletech.com/articles/673-survey-examines-global-consumer-perceptions-of-autonomous-vehicle-safety) discovered that the higher the level of automation, the higher the doubt. There are still reservations about giving the vehicle total control, despite having a positive view on autonomous vehicles generally. Another factor playing on people’s minds is cybercrime: fear of personal data falling into the wrong hands. People are generally happy to share data for safety reasons, but less so when it ends up being sold to related service providers.

## 1.3. Problem Definition

**Road Accidents**

It is worth asking at this point whether distrust of autonomous technology has a semblance of the truth and is it backed by some authoritative research. According to the WHO, approximately 1 million people die due to road accidents every year. And most of the accidents are caused due to avoidable reasons. Considering the fact that more than 90% of fatal car accidents are caused due to [human negligence](https://cyberlaw.stanford.edu/blog/2013/12/human-error-cause-vehicle-crashes), the assertion that autonomous cars would not lead to a significant decline in the number of car crashes is simply absurd and defies even the most elementary logic.

## 1.4. Objective

**General Idea of The Project**

In this chapter we will discuss a general idea about the project and what we want to achieve, and the steps done to achieve the project:

**Machine Learning and Neural Networks**

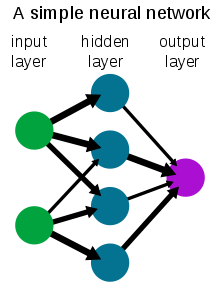
What is machine learning?

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

What are neural networks?

A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes. Thus, a neural network is either a biological neural network, made up of real biological neurons, or an artificial neural network, for solving artificial intelligence (AI) problems. The connections of the biological neuron are modelled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed. This activity is referred as a linear combination. Finally, an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be −1 and 1.

These artificial networks may be used for predictive modelling, adaptive control and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks, which can derive conclusions from a complex and seemingly unrelated set of information



Figure

Why do we need neural networks in the project?

We will use a dataset of common objects we see normally in our daily life in streets to train our neural network to be able to recognise these objects in any future image that would be our input to the neural network.

Examples:

1. Cars
2. Human bodies
3. Cats
4. Dogs
5. Buses
6. Traffic posts

**Phases of The Project**

Our project consists mainly of 2 main phases:

1- 1st phase is to find a suitable architecture of a neural network that would be able to satisfy our constrains (delay, complexity, etc..) and implement this architecture using python machine learning modules like TensorFlow and PyTorch

2- 2nd phase after simulating our neural network we will get the weights resulted from the simulation (more details on that later) and use it to implement a hardware chip on a FPGA development kit

We decided to implement the project on a FPGA kit as hardware is a lot faster than software simulations and in object detection in autonomous cars we need very fast response as it is very critical as very small delays may lead to hazardous events also, we favoured it over ASIC due to reconfigurability

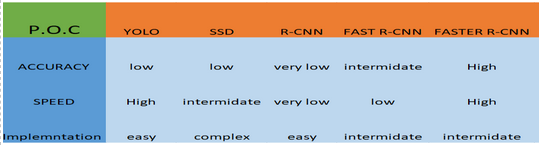
## 1.5. Related Work

**- If there is no related work “For the best knowledge of the authors, this is the first Contribution in…….”**

**Organization of the remaining chapters**

# 2.The proposed solution and Methodology

## 2.1. Proposed Solution/technique/architecture: architecture or flowchart



Figure

We have seen the suggested architectures in the convolutional neural networks, and we chose the faster RCNN as it is the best according to our constrains

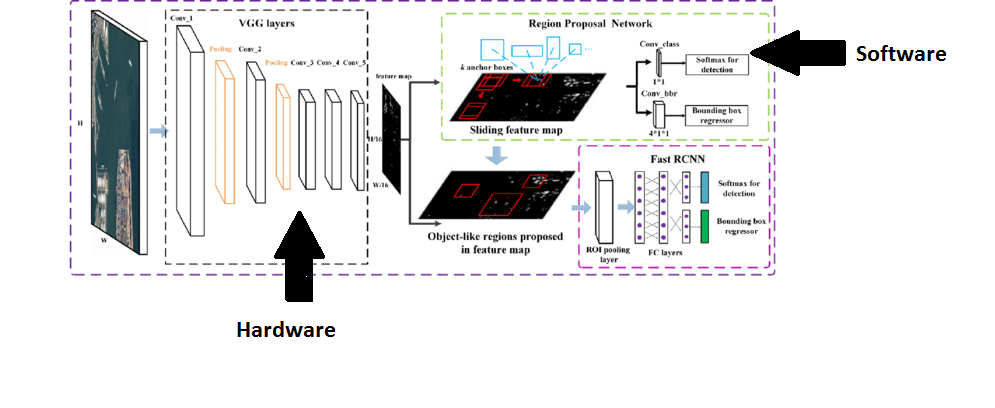
Faster RCNN is the modified version of Fast RCNN. The major difference between them is that Fast RCNN uses selective search for generating Regions of Interest, while Faster RCNN uses “Region Proposal Network”, aka RPN. RPN takes image feature maps as an input and generates a set of object proposals, each with an objectness score as output. The below steps are typically followed in a Faster RCNN approach: We take an image as input and pass it to the ConvNet which returns the feature map for that image. Region proposal network is applied on these feature maps. This returns the object proposals along with their objectness score. A RoI pooling layer is applied on these proposals to bring down all the proposals to the same size. Finally, the proposals are passed to a fully connected layer which has a SoftMax layer and a linear regression layer at its top, to classify and output the bounding boxes for objects. To begin with, Faster RCNN takes the feature maps from CNN and passes them on to the Region Proposal Network.

RPN uses a sliding window over these feature maps, and at each window, it generates k Anchor boxes of different shapes and sizes. Anchor boxes are fixed sized boundary boxes that are placed throughout the image and have different shapes and sizes. For each anchor, RPN predicts two things:

The first is the probability that an anchor is an object (it does not consider which class the object belongs to),Second is the bounding box regressor for adjusting the anchors to better fit the object. We now have bounding boxes of different shapes and sizes which are passed on to the RoI pooling layer. Now it might be possible that after the RPN step, there are proposals with no classes assigned to them. We can take each proposal and crop it so that each proposal contains an object. This is what the RoI pooling layer does. It extracts fixed sized feature maps for each anchor, then these feature maps are passed to a fully connected layer which has a SoftMax and a linear regression layer. It finally classifies the object and predicts the bounding boxes for the identified objects.

## 2.2. Methodology: flowchart or block-diagram or steps

**Block diagram for the whole project**



Figure

### 2.2.1 Software Solution

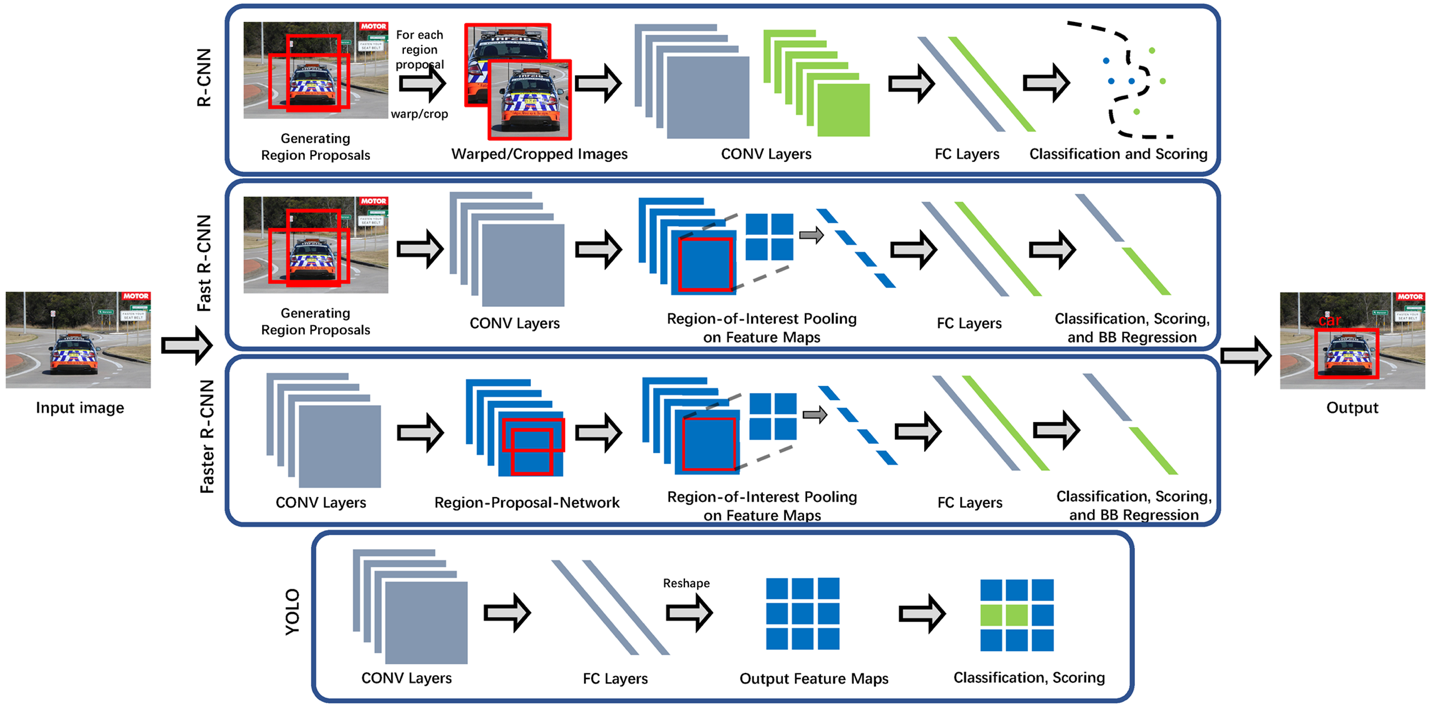
We know in the software industry that without an interface, libraries, and organized tools, software development proves a nightmare. When we combine these essentials, it becomes a framework or platform for easy, quick, and meaningful software development.

ML frameworks help ML developers to define ML models in precise, transparent, and concise ways. ML frameworks used to provide pre-built and optimize components to help in model building and other tasks. There are famous frameworks commonly used as TensorFlow and PyTorch, we expected to use both, so we wanted to know the differences between them.

TensorFlow: TensorFlow is an open source deep learning framework created by developers at Google and released in 2015, It supports regressions, classifications, and neural networks like complicated tasks and algorithms. You can run it on CPUs & GPUs both. TensorFlow creates a static graph, you first must define the entire computation graph of the model and then run your ML model. It’s more difficult than PyTorch to learn but it has larger community and easier to find solutions to your problem as it’s older than PyTorch.

PyTorch: PyTorch is based on Torch and has been developed by Facebook, it’s used by Facebook, IBM, Yandex, and Idiap Research Institute. PyTorch believes in a dynamic graph, you can define/manipulate your graph on-the-go. This is particularly helpful while using variable length inputs in RNNs. Torch is flexible and offers high-end efficiencies and speed. It offers a lot of pre-trained modules. It’s easier to learn but it is new compared to TensorFlow, so it has smaller community than TensorFlow.

**Block diagram for software identification of objects**



Figure

### 2.2.2 Hardware Solution

Using vivado HLS tool because The tool interface is very easy We can write c++ code and turn it into Verilog

Through simulation wave forms can be obtained, We can target zynq fpga directly, Specify the type of memory easily (FIFIO ,ROM ,BRAM , RAM )

Using directives we can easily pipeline the codes, Give estimations to frequency , clock cycles and resources utilization, Compare different solutions at time.

Test the generated code against the c++ testbench and Useful flags inside the Verilog that help understanding the operation

**Block diagram of layers of VGG-16**



Figure

# 3. Implementation

## 3.1. Software implementation

We will be simulating the Faster-RCNN architecture using google colabs as this will fasten the training using google servers

we will use a unique custom dataset which is a subset of the VOCtrainval\_11-May-2012 that contains objects that is present on the road normally

This data set is found in the dataset folder where it is divided into 2 folders one which contains the images and the annotation files(XML format) and the other folder contains the tfrecord format that will be used in training ( explained later )

[Roboflow](https://roboflow.ai/) makes managing, preprocessing, augmenting, and versioning datasets for computer vision seamless. Developers reduce 50% of their code when using Roboflow's workflow, automate annotation quality assurance, save training time, and increase model reproducibility.

This website is used to change from one format to another to the annotation files and this will be used to generate the formats we will need in training in case you need to train on your custom dataset ( note the dataset attached in the dataset folder already has the correct format ) for the full tutorial on how to work with roboflow with your custom dataset please check the following link [Roboflow](https://blog.roboflow.ai/getting-started-with-roboflow/)

### Folder structure

The software folder contains 2 items and 2 folders:

1. Models contains the actual Faster-RCNN model being implemented
2. tensorflow-object-detection-faster-rcnn contains some .ipynb files that we can use to train the dataset on google colab
3. Faster-RCNN.ipynb this is the modified file that we will use for inference or training the Faster-RCNN model
4. frozen\_inference\_graph.pb this is the file that will be outputed from the Faster-RCNN.ipynb after training the model using the dataset in the folder, This file will be used in inference mode

## 3.2. Hardware implementation

**VGG-16 archeitecture**



Figure

### 3.2.1 Convolution Layer

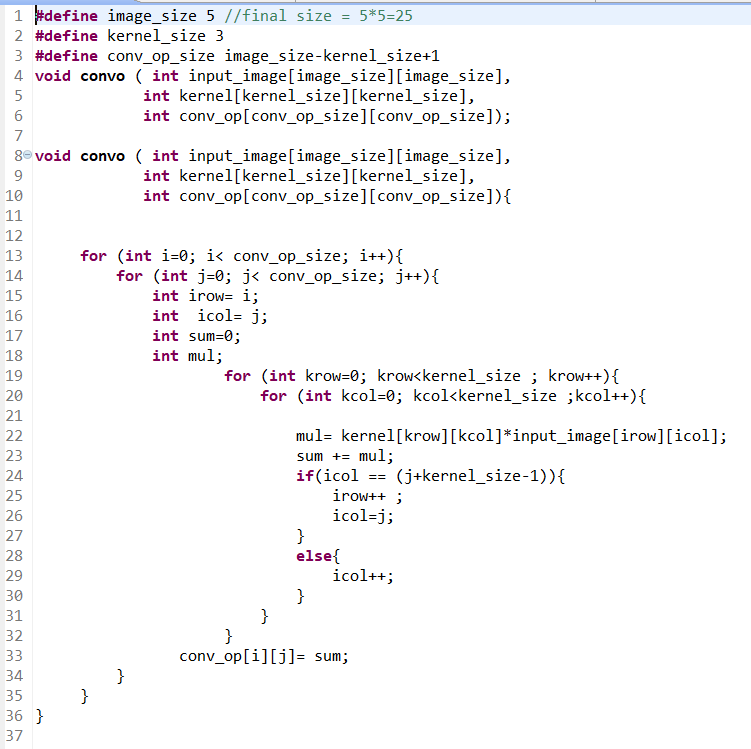
**Inputs:**

Taking the input RGB image as three dimensional array with (channel \* row \* column ) and the kernels number , channels and size as four dimension array (kernel number, kernel channel , row , column)

**Outputs:**

The feature maps the number of them is equal to the number of filters and the size is equal to image size –kernel size +1 if no padding is used

**CODE:**



Figure

#### 3.2.1.1 Convolution (RGB Input images and kernel channels)

#### A screenshot of a social media post Description automatically generated

### 3.2.2 Padding

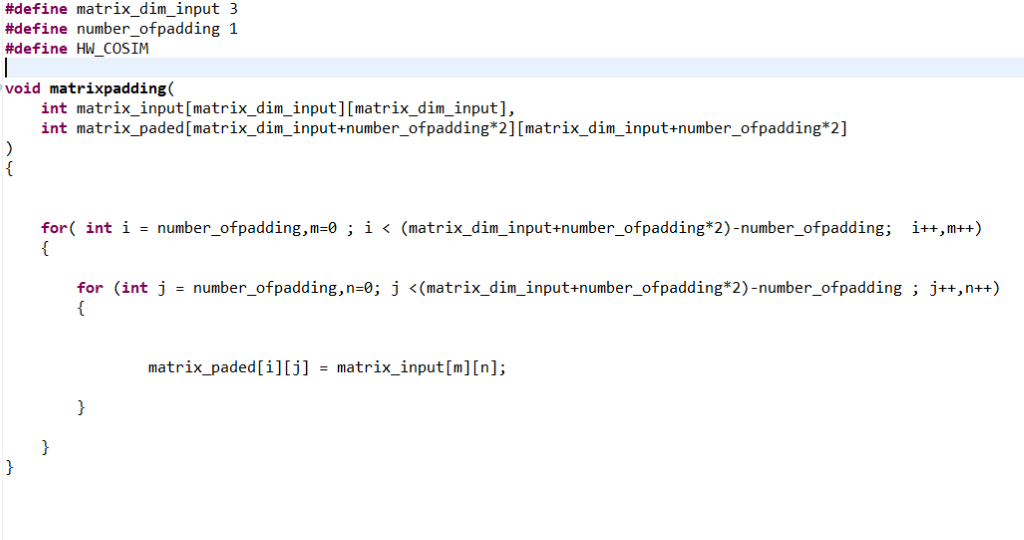
**Inputs:**

Input matrix of size 3x3

**Outputs:**

Output matrix of size 5x5 of the 3x3 matrix surrounded by zeros

**CODE:**



Figure

3.2.2.1 Padding ( padding image with multiple channels )

Code

A screenshot of a social media post

Description automatically generated

### 

### 3.2.3 Max Pooling Layer

**Inputs:**

Input matrix of 5x5

**Outputs:**

Output matrix of 3x3 but with the maximum numbers only of the input

**CODE:**





Figure

3.2.3.1 Max pooling (for multiple feature maps )

Code

A screenshot of a social media post

Description automatically generated

### 

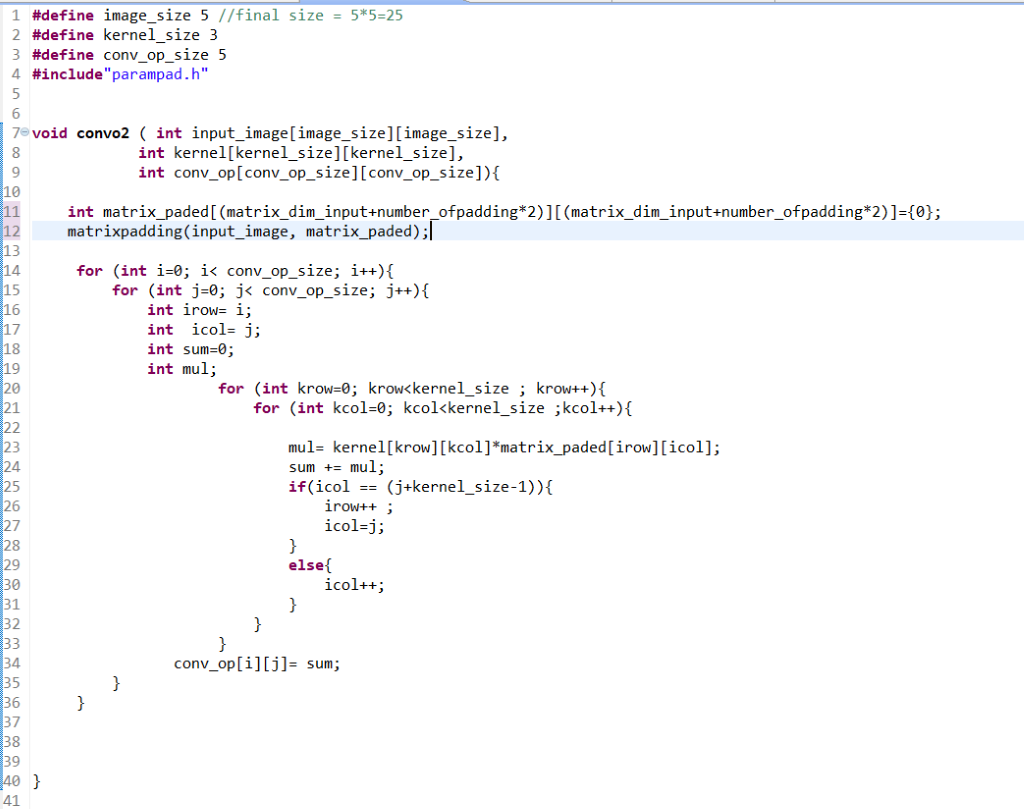
### 3.2.4 ReLU Layer

Code

A screenshot of a social media post

Description automatically generated

### 3.2.5 Integration of padding and convolution



Figure

### 3.2.6 Integration of (Padding-Convolution-Maxpooling)

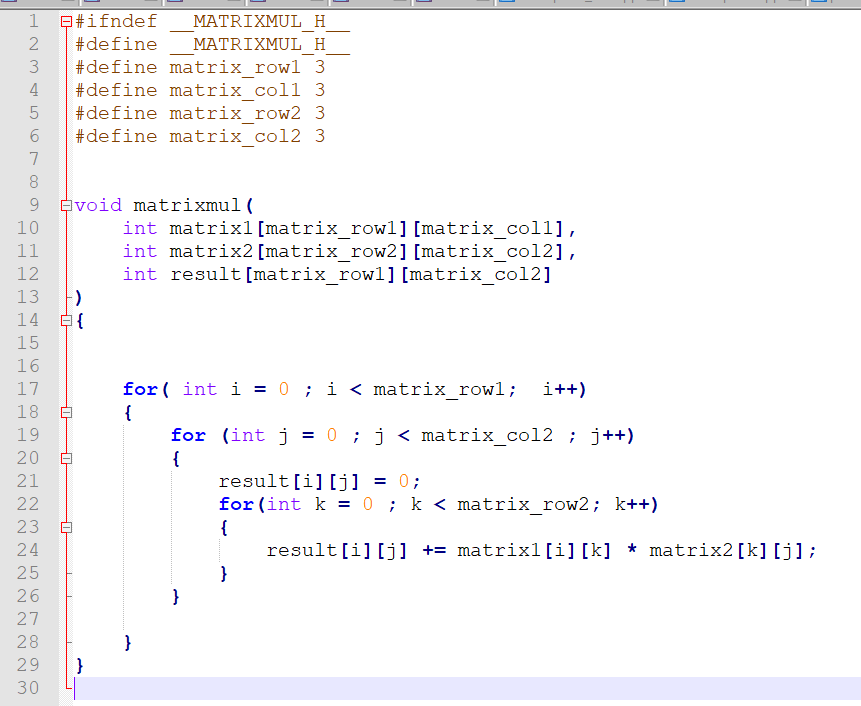
* Start the function by padding the input image so the feature map after convolution is same size if input image
* Then do the convolution with one image RGB ,2 filter with three channels each
* Then do max pooling 2\*2 which reduce the output feature map size by 2

## 

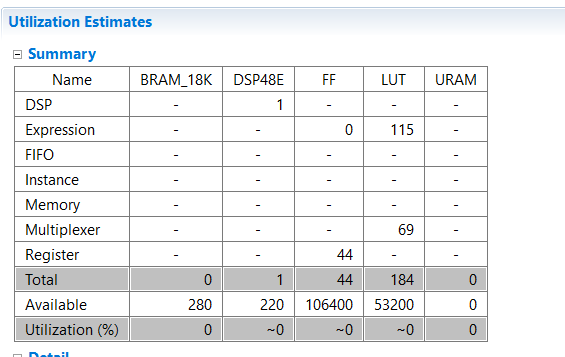
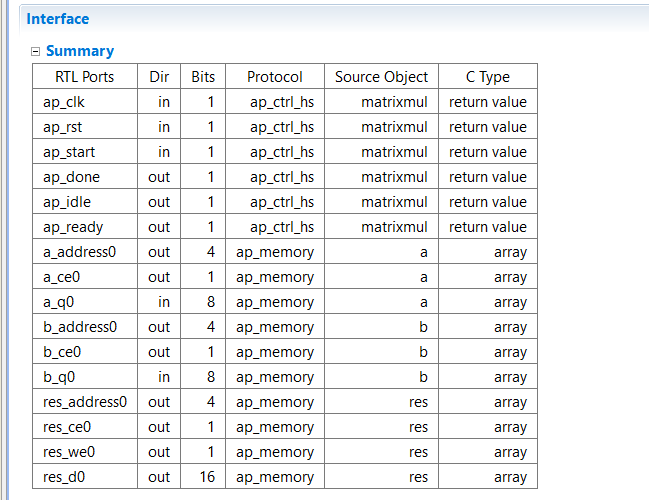
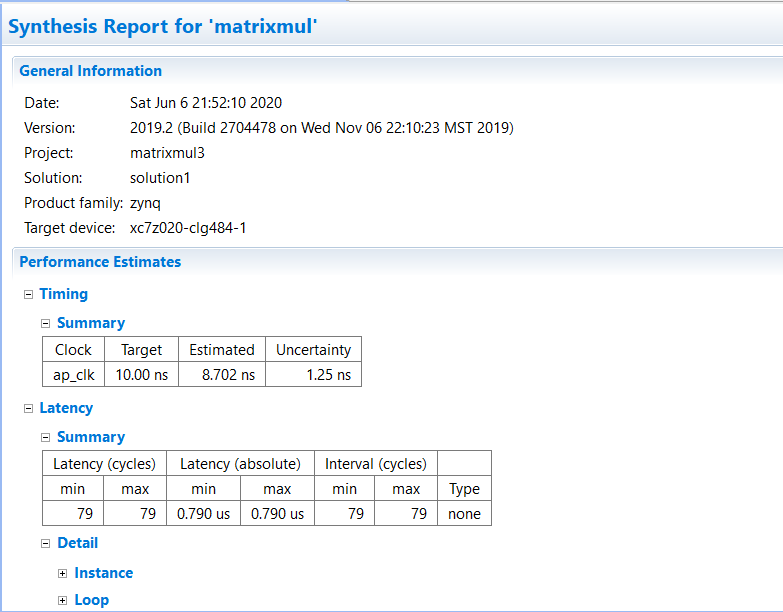
## 3.3. An illustrative example

**Matrix multiplication**

CODE :

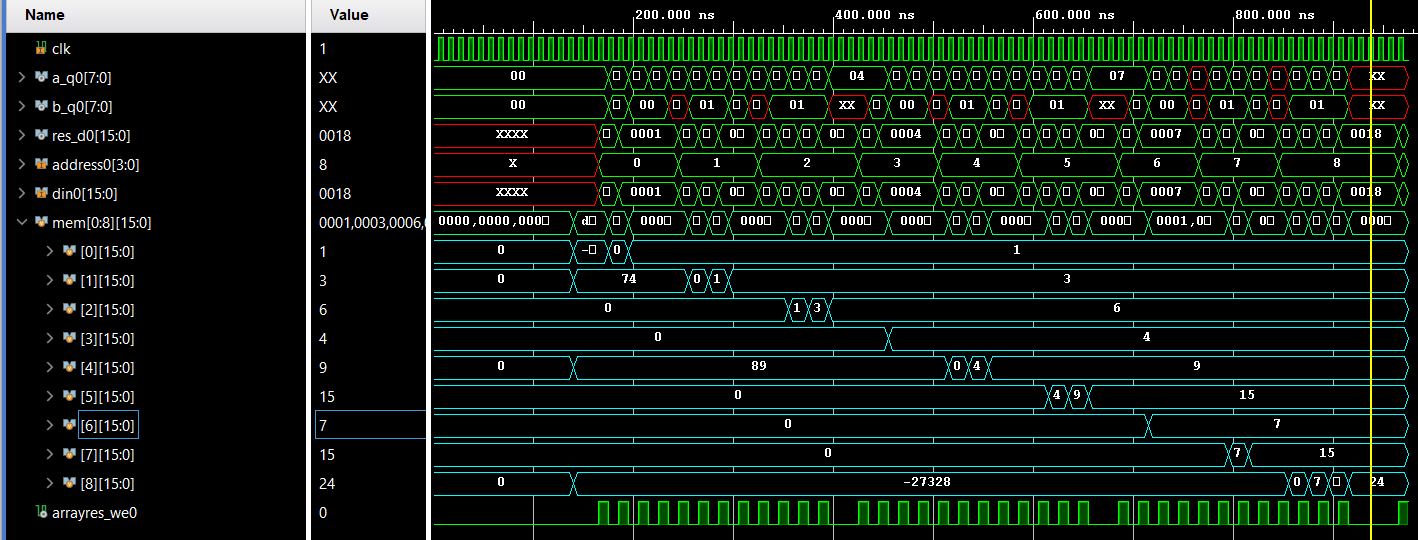


Figure

Synthesized code output report :

Figure

Simulation screenshot:

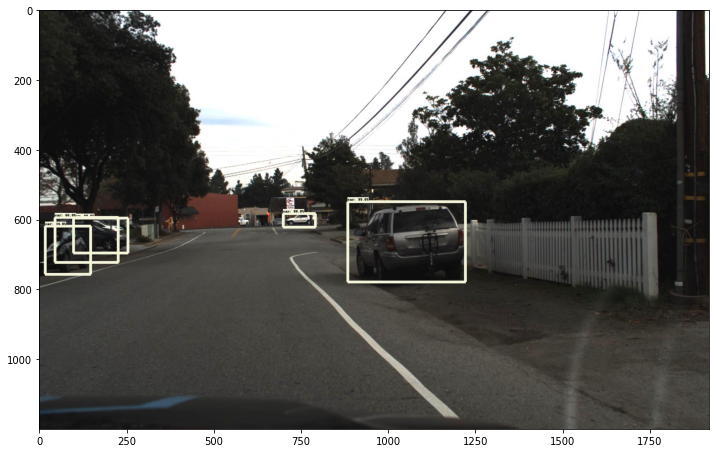


Figure

The memory in the aqua is output array that has the res\_d0 saved in it .It shows clearly that it is the expected output. By tracing the wave form of inputs and the result the timing of every thing goes as expected .the testbench of matrices are

# 4.Validation and results

## 4.1. Software experimental Results

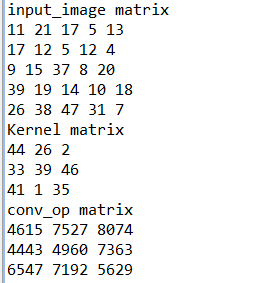


Figure

## 4.2. Hardware experimental Results

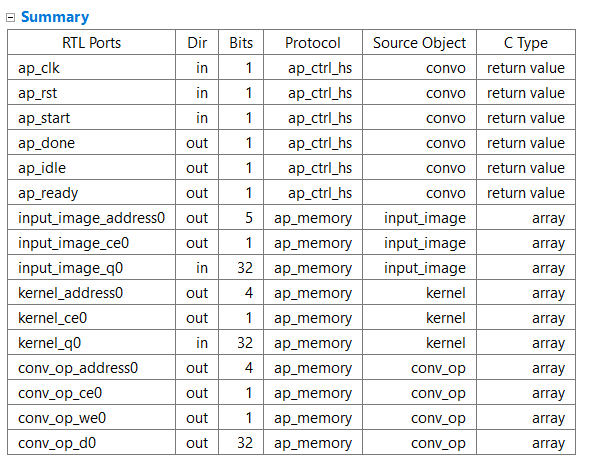
### 4.2.1 Convolution Layer

**Results**



Figure

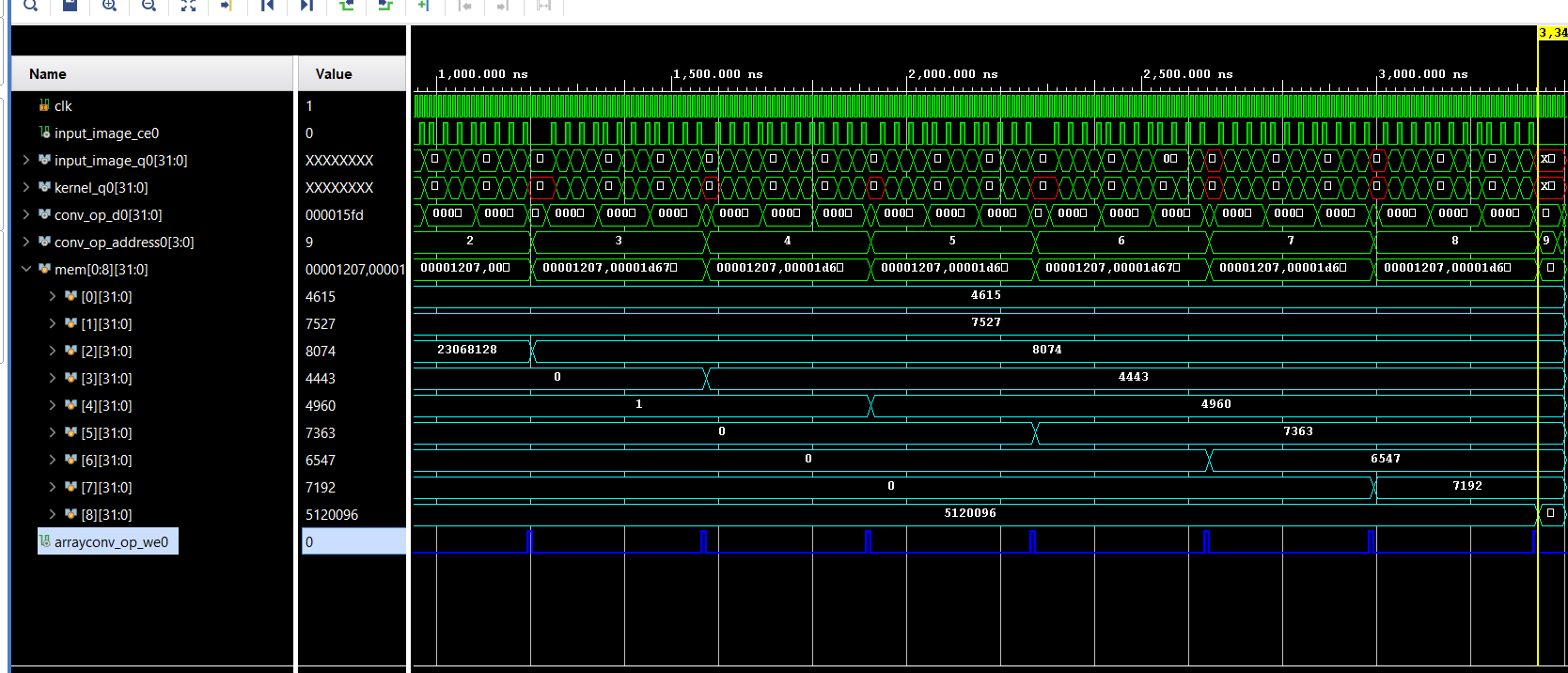
**Synthesized code output report**



Figure

**Simulation results**

The simulation shows that convolution process is carried out as intend in the c++ code and below output is output of c++ testbench and the output exactly the same. Noting that the we0 (write signal in the output memory (array)is high only when the correct result signals has arrived can be specified by the clock period during project )



Figure

#### 4.2.1.1 Convolution (RGB Input images and kernel channels)

**Synthesized code output report**

A screenshot of a cell phone

Description automatically generated

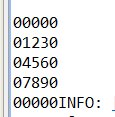
**Simulation results**

A screenshot of a computer

Description automatically generated

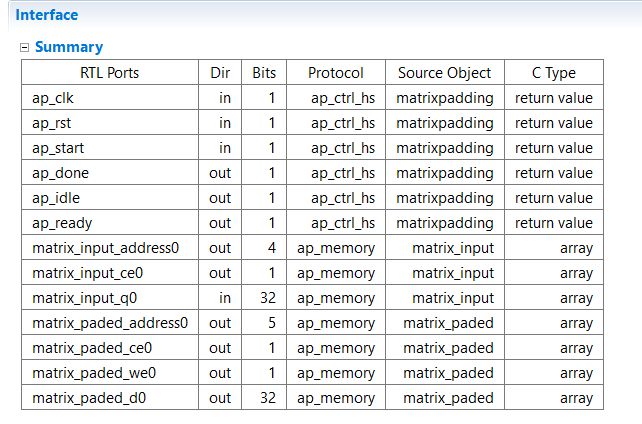
### 4.2.2 Padding

**Results**



Figure

**Synthesized code output report**



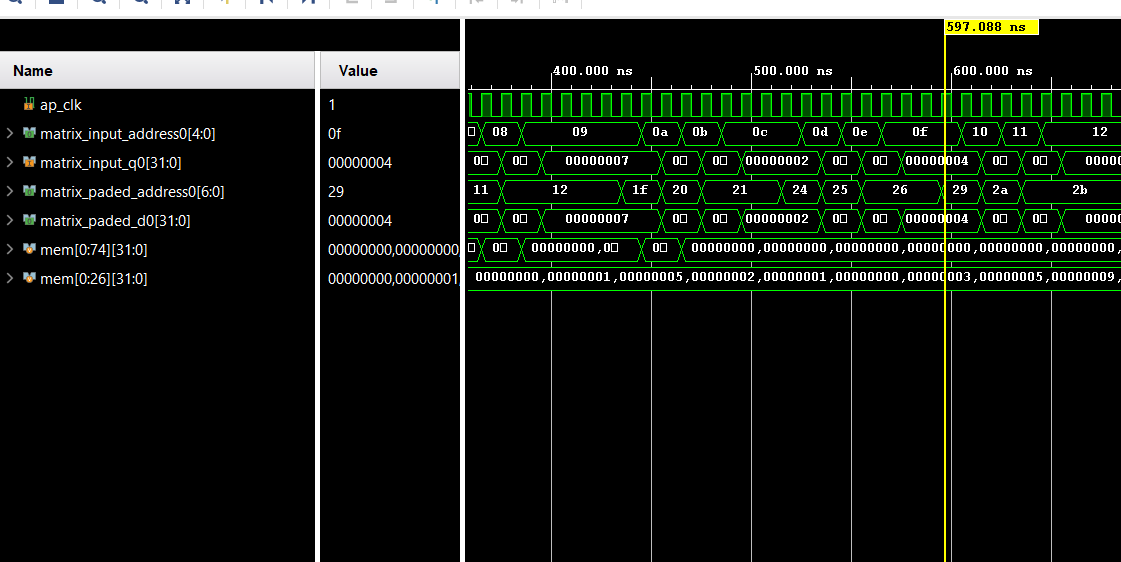
Figure

**Simulation results**



Figure

**When the input is 3d Matrix (channels \* columns\*rows)**



Padding ( padding image with multiple channels )

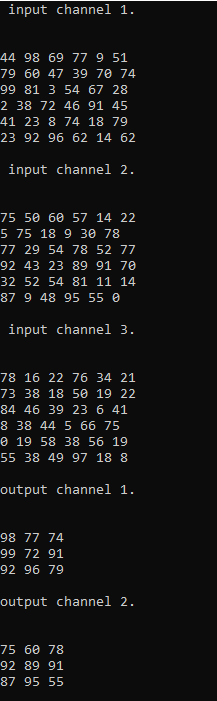
**Simulation results**

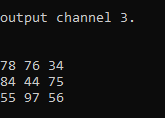


### 

### 4.2.3 Max Pooling Layer

**Results**





Figure

**Synthesized code output report**

A screenshot of a cell phone

Description automatically generated

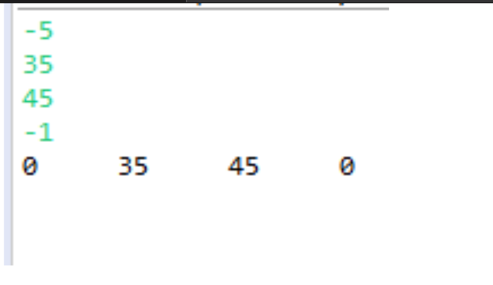
**Simulation results**

A screen shot of a computer

Description automatically generated

### 4.2.4 ReLU Layer

Results

****

**Synthesized code output report**

A screenshot of a cell phone

Description automatically generated

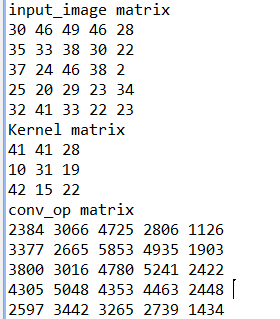
**Simulation results**

A screenshot of a computer screen

Description automatically generated

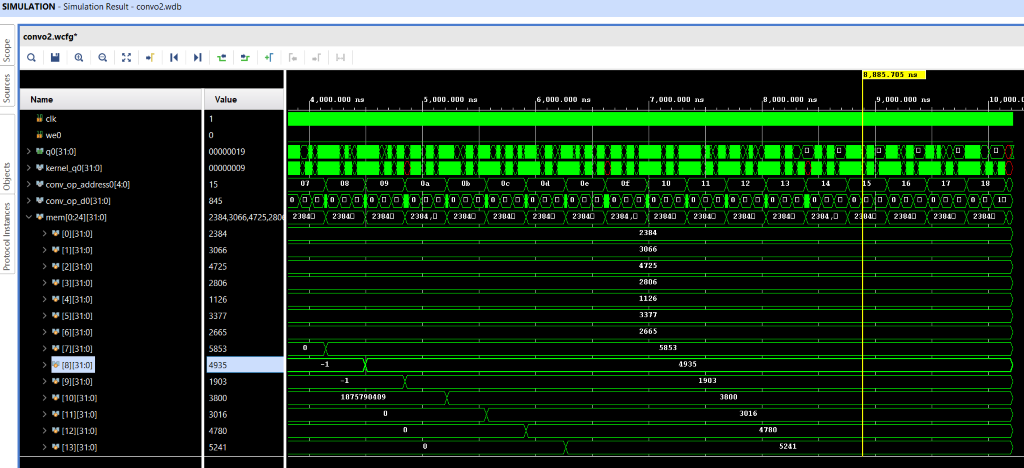
### 4.2.5 Integration of padding and convolution

**Results**



Figure

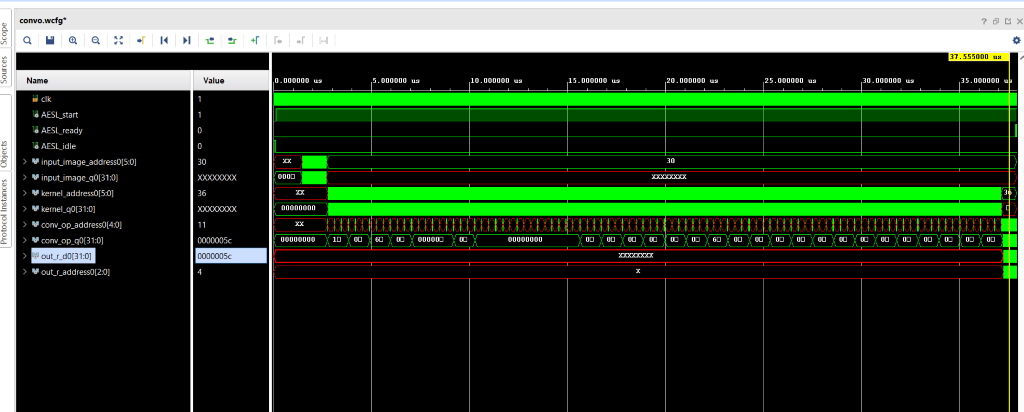
**Simulation results**



Figure

### 4.2.6 Integration of (Padding-Convolution-Maxpooling)

**Simulation results**



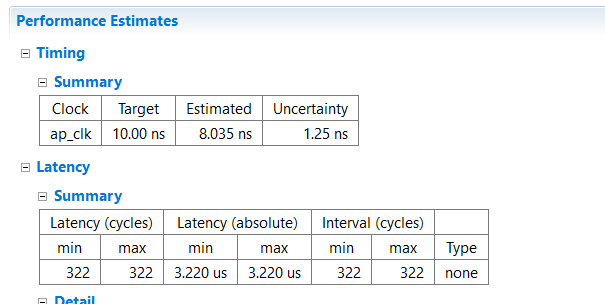
Figure

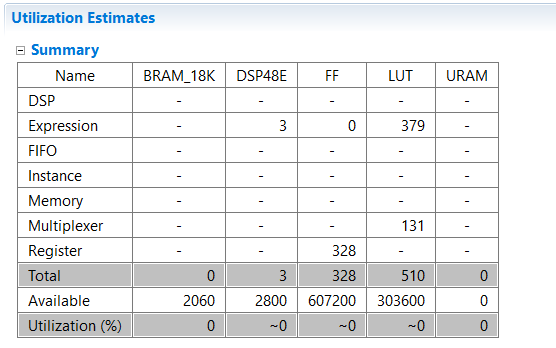
## 4.3. Case studies

## 4.4. Performance evaluation

## 4.5. Performance metrics

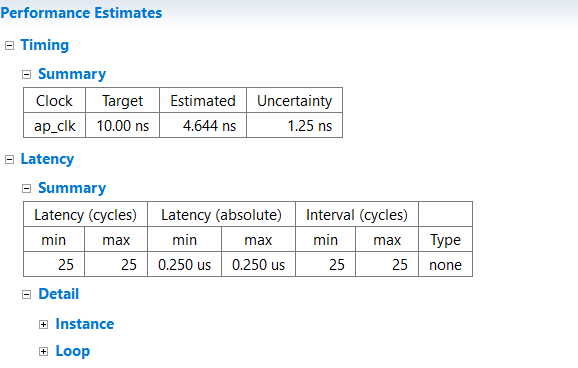
### 4.5.1 Convolution Layer

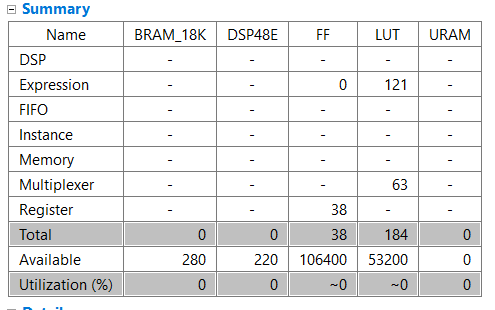




Figure

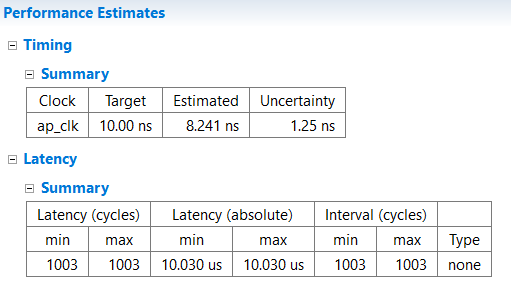
### 4.5.2 Padding Layer

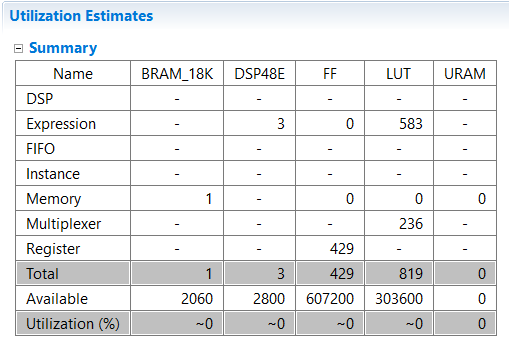




Figure

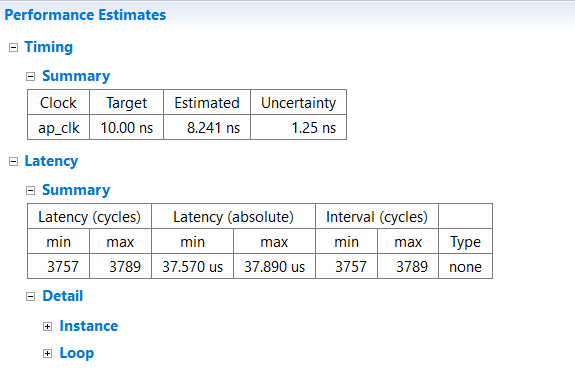
### 4.5.3 Integration of padding and convolution

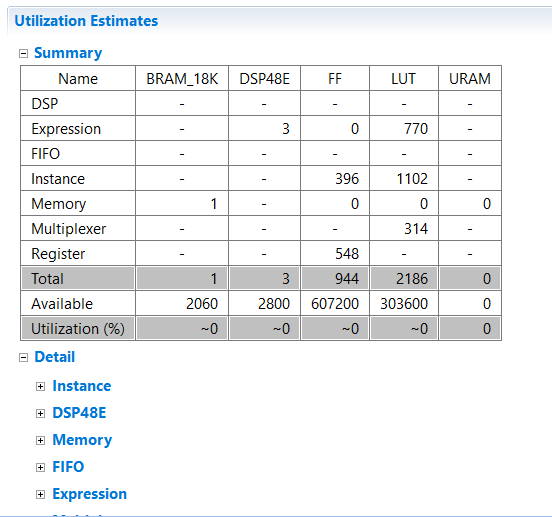




Figure

### 4.5.4 Integration of (Padding-Convolution-Maxpooling)





Figure

## 4.6. Comparison with related work

## 4.7. Conventional versus proposed

## 4.8. Others work versus proposed

## 4.9. Limitations

## 4.10. Demo (if applicable) ~ 3 min

## link

# 5 .Conclusions

# References