UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

IN-MEMORY-COMPUTING CNN ACCELERATOR EMPLOYING CHARGE-DOMAIN COMPUTE

A Master's Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Unai Echeverria Olaiz

In partial fulfilment

of the requirements for the degree of

MASTER IN ELECTRONICS ENGINEERING

Advisor: Francesc de Borja Moll Echeto

Codirector: Xavier Aragones Cervera

Barcelona, November 2020

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

telecos BCN

Title of the thesis:

In-Memory-Computing CNN accelerator employing charge-domain compute

Author:

Unai Echeverria Olaiz

Advisor:

Francesc de Borja Moll Echeto

Advisor:

Xavier Aragones Cervera

Abstract

High-dimensional matrix-vector-multiplications (MVM) are the main operations of deep neural networks (DNN). As the size of DNNs increases, data movement becomes a problem and limits their performance. Analog in-memory computing accelerators are one of the most promising solutions to reduce this problem.

This project designs an in-memory computing solution that employs charge-domain compute using 22FDX technology. The design, called multiplying bit cell (M-BC), consists of an 8T bit cell and a MOM capacitor. The design is part of an architecture of 8x8 = 64 neuron tiles that performs the filtering operation of up to 3x3x512 input activation (IA). Each neuron tile is composed of 64x64 = 4096 neuron patches. The design achieves energy efficiency of 1170 TOPS/W and throughput of 18876 GOPS.

Keywords

AI, ML, DL, NN, CNN, SRAM, M-BC, MVM, MAC

Acknowledgements

First of all, I would like to thank my advisor and co-director Francesc Moll and Xavier Aragones for all the help and support given during the realization of the thesis. I would especially like to thank you for the support offered and the trustful shown in the first weeks of the project.

I would also like to thank Isidro Martín and Francesc Rey for his great professionalism and for always helping me solving any doubts related the master's degree and the entire procedure of the final thesis. Also thank you for all the advice offered in the search for the project.

Finally, I would like to thank my family, my friends and my colleagues, especially to Paula, for the unconditional support.

Revision history and approval record

Revision	Date	Purpose
0	07/09/2020	Document creation
1	13/11/2020	Document revision
2	16/11/2020	Document revision
3	18/11/2020	Document approved

Written by:		Reviewed and approved by:	
Date	17/11/2020	Date	18/11/2020
Name	Unai Echeverria Olaiz	Name	Francesc Moll
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation

Artificial Intelligence (AI) is the science of producing machines that have some similarities with human intelligence. It is very difficult to determine the exact date AI was born, but many people claim that it was created in 1956 by the mathematician John McCarthy. From that date until present days, AI has reached unimaginable levels, and its evolution is expected to continue increasing in the coming years. According to an Accenture research, "AI has the potential to boost rates of profitability by an average of 38% by 2035 and lead to an economic boost of US$14 trillion across 16 industries in 12 economies by 2035" [44]. This increase is largely due to digitalization of information and the improvement of computers. AI can be divided into different subfields, but in recent years many companies and researchers are focusing on a branch of AI called Machine Learning (ML). Machine learning is an AI technique that gives systems the ability to learn and improve automatically from experience. ML is divided into many subsets that are on raise in recent years, such as Neural Networks (NN). Normally, when talking about NN, it is normal to talk about Deep Learning (DL), since they are closely related. DL is a type of ML that through algorithms inspired by the way the biological brain operates, learn from a large amount of data. Nowadays, this technique is used in countless applications such as autonomous driving, computer vision or image recognition. Figure 1 shows the taxonomy of AI [45, 46,47].

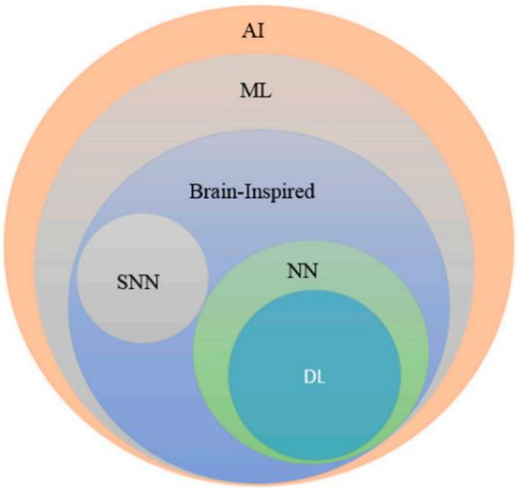


Figure 1 Taxonomy of AI [48]

In recent years, neural networks have become essential in many applications, especially in those that work with a large amount of data, such as image and character recognition. A standard NN is composed by an input layer, several hidden layers and an output layer. In each of its hidden layers, the main operation of a NN is performed: the Matrix-Vector-Multiplication (MVM). The MVMs are performed between the weight matrixes, that are generally stored in SRAM memories, and the input vectors of each layer. In a standard solution, the SRAM memories are separated from the computation units, causing massive movement of data when performing the MVMs. This problem is known as "memory wall".

The evolution and performance of NNs have been so great in recent years that they have exceeded the capabilities of humans to perform some tasks. This achievement is very important for some applications, such as image recognition. For example, the image

recognition system used in vehicles as an Advanced Driver Assistance System (ADAS), requires very high accuracy, because a small failure can have critical consequences. To obtain high accuracy, it is necessary to have very complex networks. That is, to work with a huge amount of data. When this happens, the time and energy spent moving the data becomes a problem. Therefore, it becomes a necessity to fix the “memory wall” problem.

To avoid this problem, several researchers have focused on finding solutions to minimize the movement of data to perform this operation. Nowadays, there are several NN accelerator designs that manage to reduce the impact of this problem. The most promising NN accelerators are those that use the concept of in-memory computing. The main idea of in-memory computing solutions, as the name suggests, is to move computing to memory and thus eliminate the expensive data movement between the memory and the computation units [33, 48].

In addition to the implementation of in-memory computing solutions, the power consumption problem of NN can be reduced using Fully Depleted Silicon-On Insulator (FDSOI) technologies instead bulk CMOS technologies. This mainly happens because FDSOI devices present much lower leakage than regular devices [41].

1.2. Objectives

The main objective of this thesis is to analyze and design (schematic and layout) a 3x3 column of an in-memory computing Convolutional Neural Network (CNN) accelerator in 22 nm FDSOI process. Apart from this, the secondary objectives of the thesis are listed below:

- Get familiar with 22FDX technology
- Try to design standard layouts so that they can be reused without big modifications.
- Improve the results obtained by the reference design

This thesis is based on the design proposed in [30]. This design is made with a 65 nm CMOS technology.

1.3. Thesis Organization

The remainder of this document is organized as follows:

1. **State of the art and Context:** This chapter is divided into two main parts. In the first part, it begins explaining the basic concepts and the structure of CNNs and then continuous reviewing the state of the art in different CNN architectures and in different types of CNNs. The second part reviews the state of the art in in-memory computing designs.
2. **In-Memory Computing CNN Accelerator Structure:** This chapter starts explaining the operation of the design on which this thesis is based. Then, the two important parts of the circuit are explained, the SRAM and the M-BC. For each part, the circuit, the working principle, and some important criteria to design are mentioned. Finally, some important characteristics of the technology used to design the M-BC are explained.
3. **M-BC Schematic Design:** This chapter is divided into two main parts. In the first part, three different SRAM designs are analyzed, simulated, and compared. In the second part, the M-BC design is analyzed and simulated.

4. **M-BC Layout and Post-Layout Results:** This chapter starts explaining how the layout of the M-BC is made and what characteristics are obtained. Then, the results of the design obtained before and after the layout are analyzed and compared. Finally, the results of the project are compared with the results achieved by the reference design.
5. **Conclusions and future improvements:** The last chapter discusses the conclusions of the project and the development of future modifications that could improve the performance of the design**.**

2. State of the art and context

2.1. Neural Networks

As mentioned, deep learning uses algorithms inspired by the way the biological brain operates. Therefore, it is necessary a structure that is based on the structure of the biological brain. The fundamental units of the brain are the neuron cells. A neuron cell is composed mainly of three parts: dendrites (inputs), an axon (output), and a cell body. When a neuron cell receives an input from another neuron, if the signal is accepted, the neuron may fire. If this happens, a signal is transmitted over the axon of the fired neuron to another neuron. This way of processing information was tried to mimic in the late 1950s by the psychologist Frank Rosenblatt with the invention of the perceptron. Figure 2 shows an illustration of a neuron cell and a perceptron [45, 49].

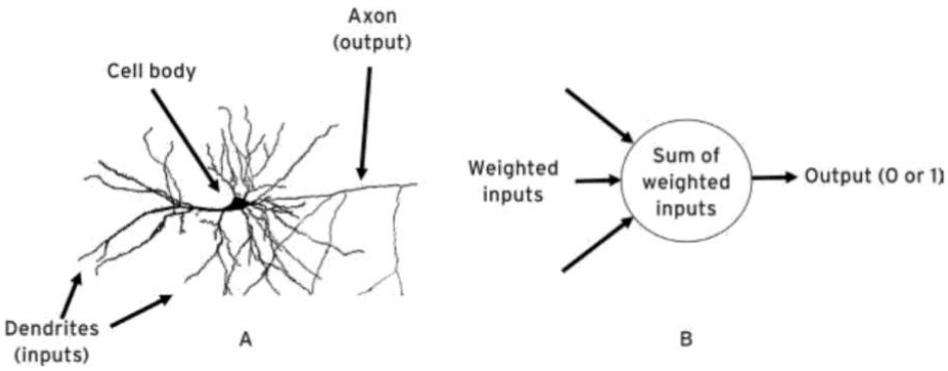


Figure 2 A, a neuron cell; B, a simple perceptron [45]

As it happens in neuron cells, the perceptron adds its inputs and, if the sum is equal or greater than the threshold value of the perceptron, the perceptron will “fire” (its output will have a value of ‘1’). Since the perceptron was introduced, a wide variety of artificial neuron models have been developed [45].

Nowadays, for most machine learning and deep learning applications, NNs are used, which, as the name suggests, are inspired in the neural structure of the biological brain. NNs are typically organized in many layers that are interconnected with each other like neurons. Depending on the number of neurons and how the connections are made, there are several types of NNs such as feedforward NN, recurrent NN, and convolutional NN, being the convolutional neural network one of the most popular networks for deep learning. Figure 3 shows several NN architectures [46, 49].

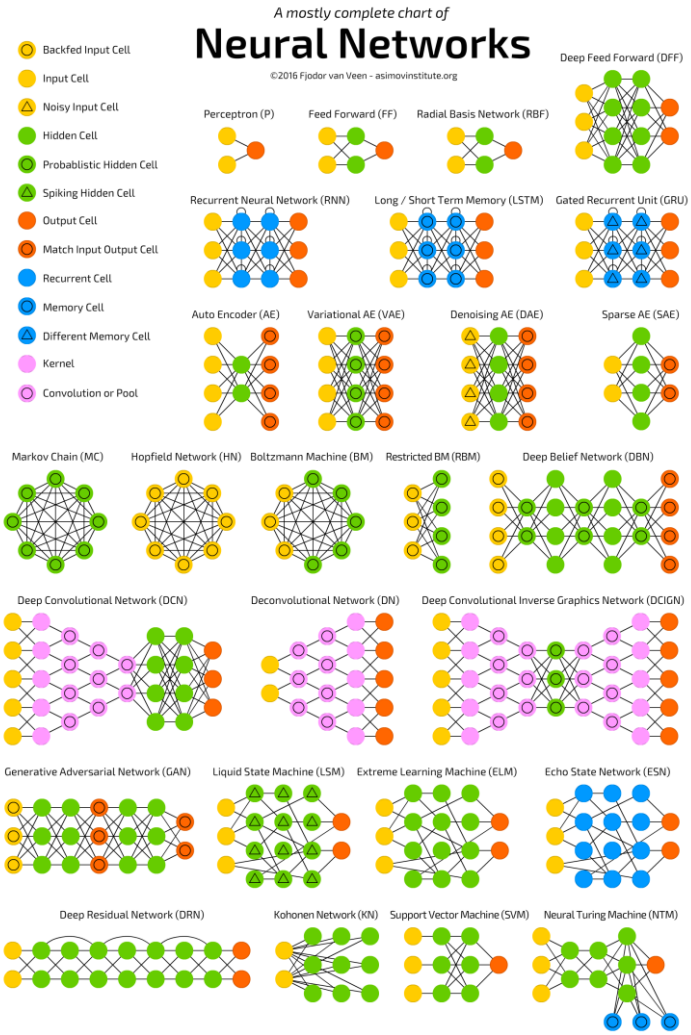


Figure 3 Several Neural Network architectures [50]

2.1.1. Convolutional Neural Network

The convolutional neural network is a specific type of neural network that receives this name because it performs a linear mathematical operation called convolution.

This type of network is used to process ordered data such as images and it is one of the most popular types of NNs, especially for applications that work with high-dimensional data [1]. The main advantage of CNNs is that they require less variables than other types of NNs. In fact, it would be impossible to use another type of NN, such as a Fully Connected Network, to process high resolution images or videos, because the number of parameters required would be unmanageable for any practical application [2].

In this type of network, as it can be deduced from its name, the convolution operation is its main characteristic. This operation is performed in a specific layer inside the network, called convolutional layer. Apart from this layer, the CNN is composed by other layers. In general, the standard model of a CNN is composed by an input layer, a convolutional layer, a pooling layer, a non-linear layer and a small number of fully connected layers that are different depending on the application of the network [3]. Figure 4 shows the standard model of a CNN.

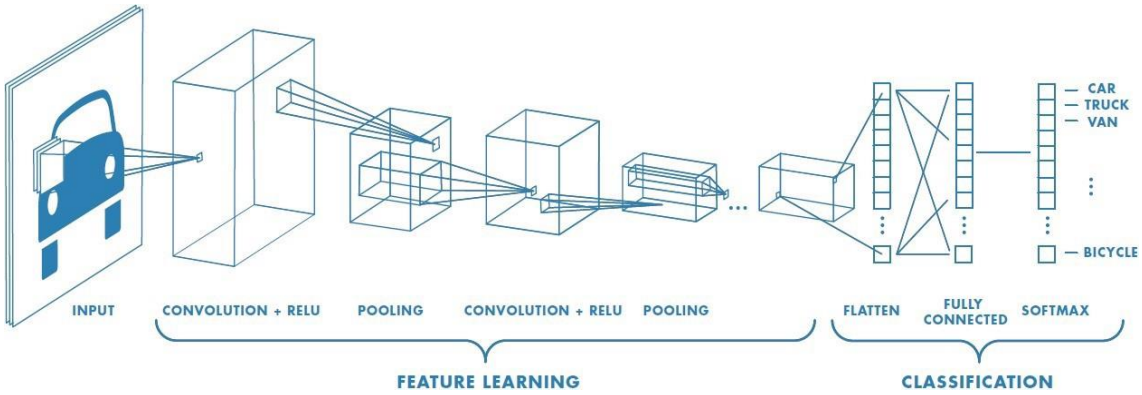


Figure 4 Standard model of a CNN [4]

2.1.1.1. Input layer

In general, the input of a CNN is a multidimensional array of data that introduces data to the network. Depending on the application of the network, the input data can be a color image represented by a three-dimensional matrix, a video signal or a higher dimensional array composed of training examples [5].

2.1.1.2. Convolutional layer

As introduced before, this layer is the principal block of CNNs. The main objective of this layer is to detect local features from the previous layers and generate a feature map with the information obtained. To obtain high-level features of an image it is necessary to use several convolutional layers. The first layers are responsible for capturing low-level features, such as lines or edges, and higher layers are responsible for capturing high-level features, such as paws or eyes to identify the object. To obtain the features of an image, parts of the input are convolved with one or several filters or kernels.

Convolution process

In general terms, convolution is a mathematical operation that combines two signals forming a third signal. This combination can be understood as the superposition of the input signal and a translated and inverted secondary signal. So, in general, for discrete signals, the convolution of x[n] and h[n] is expressed in (1).

$$x[n] * h[n] = \sum_{m=-\infty}^{\infty} x[m] \cdot h[n-m] \tag{1}$$

Working with images, the convolution can be understood as the sum of all the multiplications performed between the kernel weights and the corresponding pixel value of the input image. This operation is commonly known as matrix-vector-multiplication. To simplify this operation, a two-dimensional image and a unique kernel of M x N dimensions will be considered. The convolution operation working with images is represented in (2), where the input image is represented by **I**, the kernel weights by **W** and the output feature map by **F**. The kernel is placed in the pixel i, j of the input image.

$$f_{i,j} = \sum_{m=1}^{M} \sum_{n=1}^{N} i_{i+m-1,j+n-1} \cdot w_{m,n} \tag{2}$$

Once the expression is explained, an illustrative example of the calculation will be shown. Figure 5 shows the input image and the kernel used in the example.

5x5 Input

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

3x3 Kernel

1	0	1
0	1	0
1	0	1

Figure 5 Input image and kernel [6]

To perform the convolution operation, the kernel is shifted over all the input. At every location, it will be applied (2), obtaining a unique value that will be placed in the feature map. Figure 6 shows the convolution operation and the result obtained when the kernel is placed in the pixel 1,1 of the input.

Input x Kernel

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

Feature Map

4		

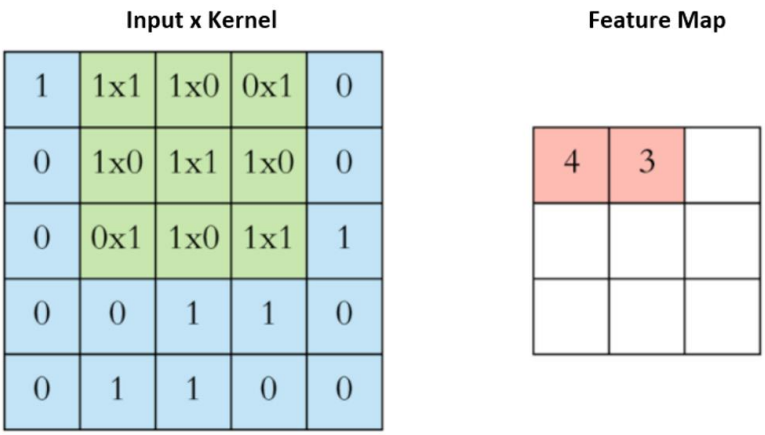
$$f_{1,1} = \sum_{m=1}^{3}\sum_{n=1}^{3} i_{1+m-1,1+n-1} \cdot w_{m,n}$$

$$f_{1,1} = 1 \cdot 1 + 1 \cdot 0 + 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 + 1 \cdot 0 + 0 \cdot 1 + 0 \cdot 0 + 1 \cdot 1$$

$$f_{1,1} = 4$$

Figure 6 Convolution operation in the pixel 1,1 [6]

By sliding the kernel into the right one position, a new value is obtained in the feature map as can be seen in Fig. 7.



$$f_{2,1} = \sum_{m=1}^{3}\sum_{n=1}^{3} i_{2+m-1,1+n-1} \cdot w_{m,n}$$

$$f_{1,1} = 1 \cdot 1 + 1 \cdot 0 + 0 \cdot 1 + 1 \cdot 0 + 1 \cdot 1 + 1 \cdot 0 + 0 \cdot 1 + 1 \cdot 0 + 1 \cdot 1$$

$$f_{1,1} = 3$$

Figure 7 Convolution operation in the pixel 2,1 [6]

This process is carried out until the kernel has been placed in all possible positions over the input. Once this is done, the complete feature map is obtained. Figure 8 shows the obtained feature map for this example.

Figure 8 Final Feature Map [6]

As it can be deduced, the dimension of the feature map tends to be reduced since it is not possible to apply the kernel in all positions of the input image. It can also be expected that when the larger the size of kernel, the larger the reduction of the feature map. The reduction of the feature map can become a problem in some applications where a dense prediction at pixel level is required [1]. So, to avoid this reduction, a technique called padding is used. This technique consists of increasing the size of the input by adding rows and columns of pixels and therefore, enabling the kernel to reach all positions. The added pixels can take any value, however, in most cases they take a value of 0. When this happens, the technique is called zero-padding. Figure 9 shows an input image where zero-padding has been applied.

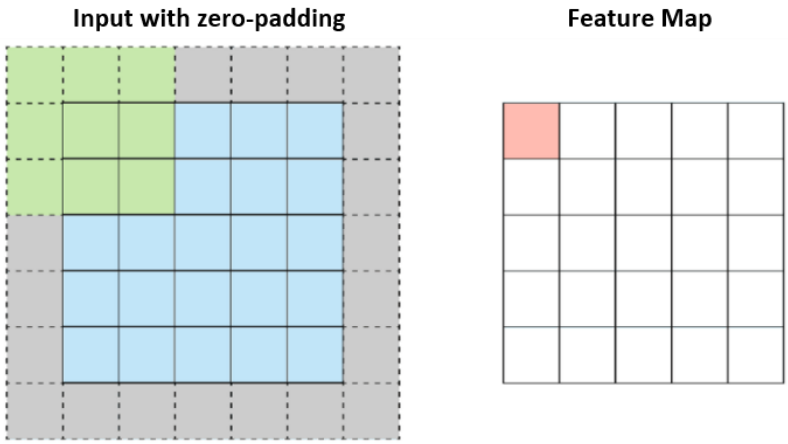


Figure 9 Zero-padding technique [6]

If instead of working with a two-dimensional image, a color image is used, that is, a three-dimensional image, the kernel used must be three-dimensional. The movement of this kernel over the input image is the same as for two-dimensional images. However, the calculation of the feature map is different in this case since the depth must be considered.

Equation (3) shows how the feature map is calculated in a color image, placing the kernel in a specific pixel position, in this case, in pixel i, j, d. In (3), the input image is represented by **I**, the kernel with size M x N x D by **W** and the output feature map by **F**.

$$f_{i,j,d} = \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{d=1}^{D} i_{i+m-1,j+n-1,d} \cdot w_{m,n,d} \tag{3}$$

Figure 10 shows an illustrative example of the calculation explained above.

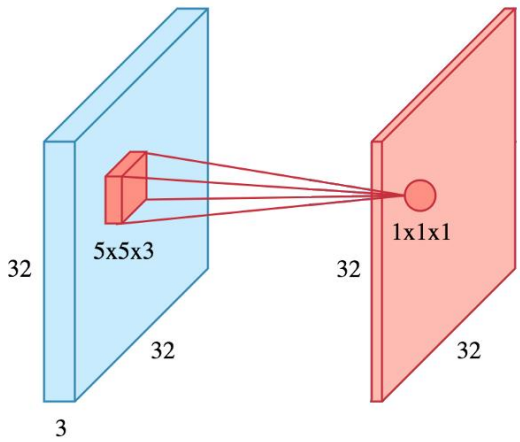


Figure 10 Example of a convolution in three-dimensional images [6]

In all the previous examples only one kernel has been used to obtain the feature map. However, in a real image processing application, multiple kernels are used to process a single image, because each kernel is usually focused on finding a specific feature of the image. For example, in the first convolutional layer, a kernel can be in charge of finding circles and another kernel finding crosses. As the NN gets deeper, the number of kernels used in each convolutional layer tends to increase, since the level of the features also increases. Figure 11 shows an example of a convolution in a three-dimensional image where more than one kernel has been used.

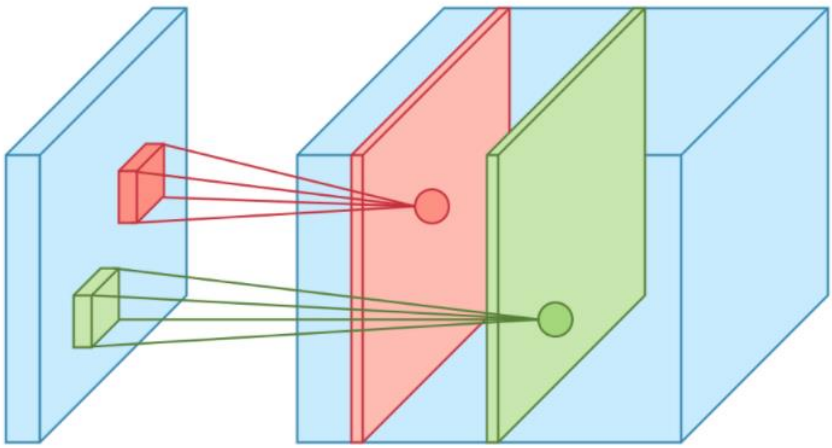


Figure 11 Example of a convolution in three-dimensional image with many kernels [6]

As it can be seen above, the convolution operation for each kernel is performed independently and the feature maps obtained with each kernel are accumulated along the depth dimension, creating a "dense" feature map [6].

Non-linearity

A nonlinear activation function is usually applied after the convolution operation. The main objective of this function is to take the feature map created in the previous layer and to generate the activation map, which will be the new output. For that, the function takes each value of the input feature map and changes it according to the type of nonlinear function. For example, some functions modify the original values by enclosing them within a small range of values such as [0,1] or [-1,1]. One of the nonlinear functions that is becoming popular in recent years is the rectified linear unit (ReLU). The ReLU is a simple activation function which is popular due to its simplicity and its fast computation [1, 5, 7]. This function "maps the input to a 0 if it is negative and keeps its value unchanged if it is positive" [1]. ReLU function is represented in (4).

$$f_{relu}(x) = \max(0, x) \tag{4}$$

2.1.1.3. Pooling Layer

Typically, after each convolutional layer, a pooling layer is added. The main objective of this layer is to reduce the dimensionality of each activation map separately, by applying one of the reduction techniques. By doing this, it makes the network more robust to small variations and vibrations. Additionally, applying this layer ensures that the network focuses on the most important features [5,7].

As mentioned before, in general, pooling layers down sample the input activation map independently, that is, the reduction technique is applied to each activation map obtained with each kernel separately. Therefore, only the width and height are reduced, not the depth. Figure 12 shows an example where it can be seen the result of applying this layer.

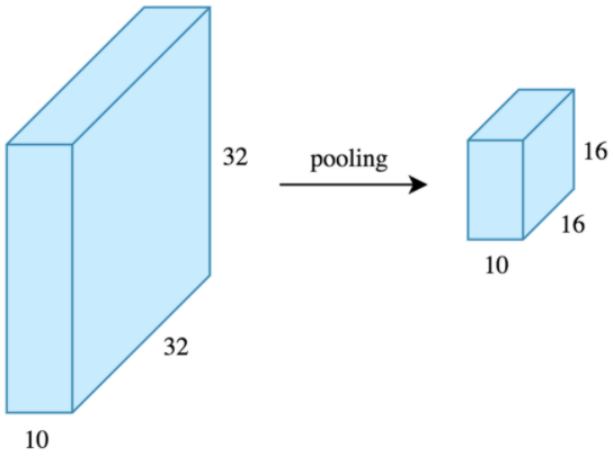


Figure 12 Example of applying a pooling layer [6]

Nowadays there are several techniques to apply this reduction, but the most popular technique is max pooling. This technique consists of taking the maximum value of the area where the pooling window has been applied. As with the kernels in the convolutional layer, the pooling window will scroll through the input map, choosing the maximum value in each area. However, an important difference between the movement of a kernel and a pooling window is the stride.

The stride specifies how much the window is moved at each step. Since the objective of pooling is to reduce the size of the input feature map, the stride of the window should be bigger than 1. Usually a 2x2 pooling window is used with a stride of 2, since using a larger window size means losing a lot of information. Figure 13 shows an example of applying max-pooling technique with a pooling window with the characteristics described above.

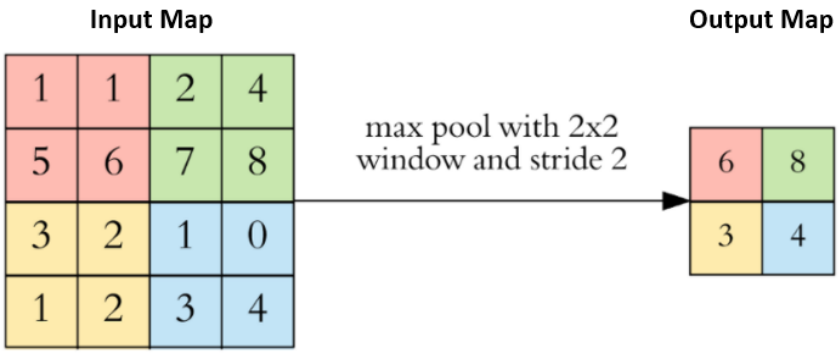


Figure 13 Max-pooling [6]

2.1.1.4. Fully Connected layers

This is the last stage of a CNN. Usually, after all the previous layers, a couple of fully connected layers are applied. As these layers expect as an input a one-dimension feature map, it is necessary to flatten the output of the previous pooling layer. As convolutional layers, fully connected layers also apply kernels to the flattened input map. Therefore, the kernels used in this layer are of size 1x1 [1].

2.1.2. Review of Architectures

Convolutional neural networks are one of the most popular neural networks used in complex applications such as image recognition and AI models for computer vision, due to their efficiency and performance. However, to obtain an efficient and well-performing network, the architecture of these CNNs is decisive. Usually for this type of applications, how layers are structured, how many layers are used, how these layers are designed and the number of kernels used in each layer directly affects the speed and accuracy of the network.

Over the years, CNNs have evolved considerably, achieving results that a few years ago not many people would have imagined. All this evolution has brought a wide variety of architectures. However, most of these modern architectures have a base architecture that has been the benchmark for many years. This first important architecture is known by the name of LeNet-5.

2.1.2.1. LeNet-5

LeNet was introduced in 1989 and was one of the first CNN architectures defined. This architecture had several versions, being the LeNet-5 the most popular. LeNet-5 was created by Yann LeCun in 1998 and it was used to digitize 32x32 pixel grey scale images such as hand-written numbers recognition. Figure 14 shows the LeNet-5 architecture.

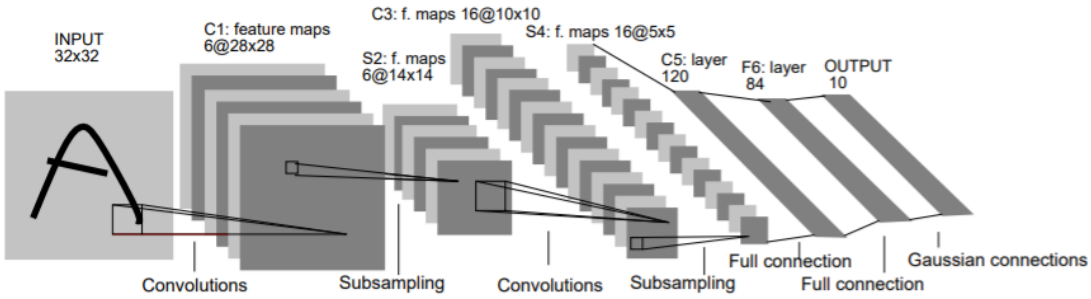


Figure 14 LeNet-5 architecture [8]

LeNet-5 architecture is a simple architecture composed by 7 layers, not counting the input. The first 5 layers are 3 convolutional layers and 2 subsampling or pooling layers that alternate as follows: C1-S2-C3-S4-C5. The number of kernels, as explained in section 2.1.1, increases as the depth of the network increases. In LeNet-5 the number of kernels is 5, 16 and 120 from the first to the last convolutional layer. The size of the kernel is 5x5 and no padding is applied in these convolutional layers. For non-linearity, in these layers, a sigmoid function is used. In the pooling layers, instead of using the max polling reduction technique explained in section 2.1.1, an average pooling reduction technique is used. This technique, instead of choosing the maximum value, performs the average of all the values. The last 2 layer are fully connected layers [8].

During those years, the performance of CNNs used for image recognition was not very realistic. This was mainly because the labelled image datasets were very small, on the order of tens of thousands of images. Therefore, the number of images per item used to train the network was very poor. To solve this problem, in 2009 a new labelled image database called ImageNet was introduced. At the beginning, the database contained a total of 3.2 million images. However, in 2019 the number of images increased to 14 million divided in 22 thousand visual categories [9].

During those years, due to the advances made in computing of modern computers and in memory technology, several new architectures were created. To check the efficiency and performance of these architectures, ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was created in 2010. This challenge is an annual computer vision competition that evaluates networks for image recognition and classification at large scale. In this competition, two error rates are reported: top-1 error rate and top-5 error rate. The top-5 error rate is the fraction of test images for which the correct label is not included among the five most probable labels chosen by the architecture. In the first competitions, the best top-5 error rates were around 26%. In 2012 a new CNN architecture called AlexNet reduced the previous best top-5 error rate result by 10% [10].

2.1.2.2. AlexNet

AlexNet was created in 2012 by Alex Krizhevsky, Ilya Sutskever and Geoffrey E.Hinton. In that year, AlexNet was the first CNN to win the ILSVRC 2012 with an incredible top-5 error rate of 15.3 % and a top-1 error rate of 36.7 %. Figure 15 shows the AlexNet architecture.

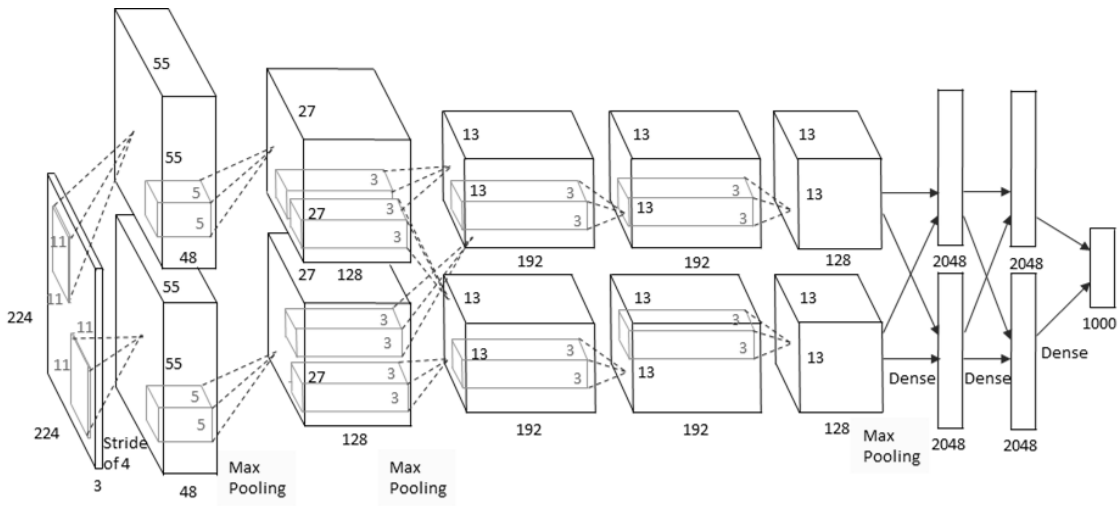


Figure 15 AlexNet Architecture [11]

The general scheme of AlexNet is very similar to LeNet-5, but it has some important differences. AlexNet architecture is much larger and deeper than LeNet-5, and instead of working with grey scale images, it works with color images. AlexNet is composed by 11 layers, not counting the input. The first 8 layers are 5 convolutional layers and 3 pooling layers that alternate as follows: C1-P2-C3-P4-C5-C6-C7-P8. The number and size of the kernels the in convolutional layers varies depending on the depth of the layer. As it can be seen in Fig.15. the number of kernels varies from 96 to 384 and the size from 11x11x3 to 3x3x3. In the pooling layer, unlike LeNet-5, AlexNet uses the max pooling reduction technique. The last 3 layers are fully connected layers [10].

Apart from these differences, there are two more important modifications that considerably increase the computing speed of the network. The main difference between AlexNet and its more popular predecessor is the application of a ReLU non-linear activation function in all the outputs of the convolution and fully connected layers. The second difference is that in the first convolutional layer the feature maps are split into two groups as it can be seen in Fig.15 [7].

After AlexNet achieved successful results at the ILSVRC, many researchers began designing new CNN architectures for image recognition and classification. The most popular CNN architectures will be mentioned below.

2.1.2.3. VGG-16

VGG-16 was created in 2014 by Karen Simonyan and Adrew Zisserman. This architecture participated in the ILSVRC 2014 obtaining the second place with a top-5 error rate of 7.3% and a top-1 error rate of 24.7%. This architecture became popular due to its homogeneity and its simplicity. Figure 16 shows the VGG-16 architecture.

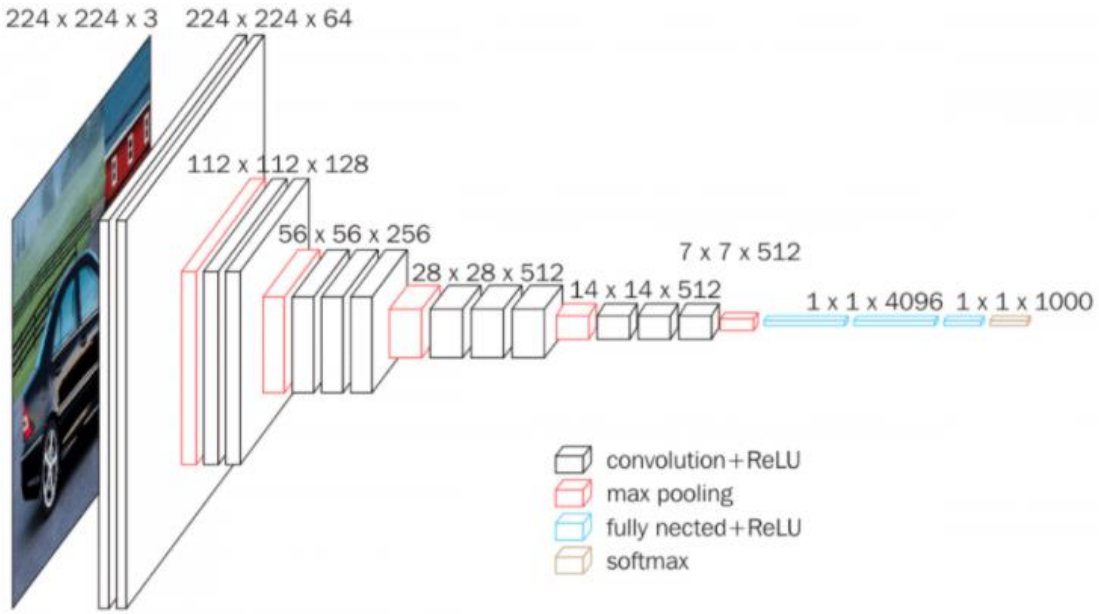


Figure 16 VGG-16 architecture [12]

The architecture of VGG-16 shares certain similarities with AlexNet, but this new architecture goes much deeper. VGG-16 is composed of 21 layers, not counting the input. The first 18 layers are 13 convolutional layers and 5 pooling layers that alternate as follows: [C1-C2-P3]x2-[C7-C8-C9-P10]x3. The number of kernels used in the convolutional layers, as in previously analyzed architectures, increases depending on the depth of the layer. In this case it varies from 64 to 512. One of the important differences between VGG-16 and AlexNet is the size of the kernels. As the VGG-16 network goes very deep, the number of weights and the MVMs to process is huge. To balance this problem, large kernels, such as the 11x11 or 5x5 kernels used in AlexNet, are built by multiplying smaller kernels of 3x3. So, all the kernels used in the convolutional layers in VGG-16 are of equal size. In the pooling layers, a pooling window of 2x2 and a stride of 2 is used. As in AlexNet, the last 3 fully connected layers have the same configuration [7, 13].

2.1.2.4. GoogleNet

GoogleNet or Inception-V1 was created in 2014 and is the winner of the ILSVRC 2014 with a top-5 error rate of 6.7%. The main goal of this design is to achieve high accuracy with a lower computational cost. This architecture is quite different from the previously analyzed architectures. The main difference is that instead of having a single serial connection, GoogleNet introduces inception modules which are composed of parallel connections. Figure 17 shows the inception module.

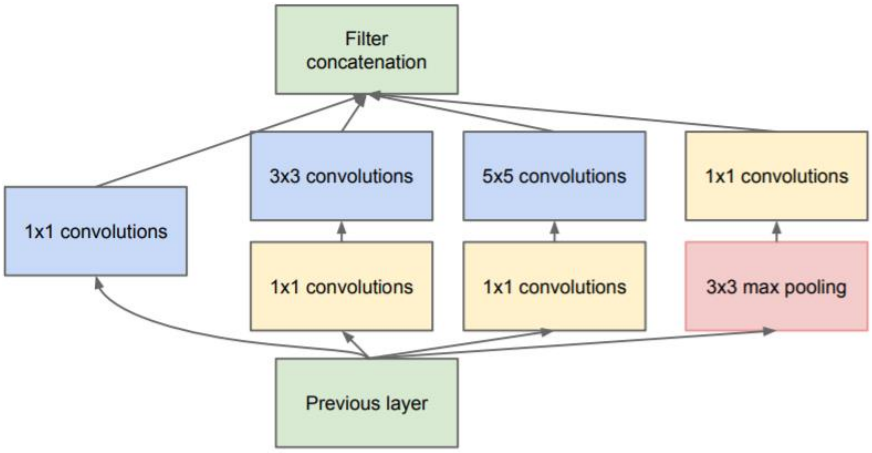


Figure 17 Inception module [14]

In the inception modules the input that comes from the previous layer is connected in parallel with three different kinds of convolution and pooling operations. In each convolutional layer (blue blocks) a different sized kernel is used; 1x1, 3x3 and 5x5. For the pooling operation (red block) a 3x3 max pooling technique is used. The output of these operations is concatenated together depth-wise. To reduce the number of weights and therefore, the computational cost, “bottleneck” layers are used (yellow blocks”). These blocks perform a 1x1 convolution to reduce the depth of the output feature map, avoiding this depth from increasing much after the application of each inception module [14].

Once the inception module has been analyzed, the general scheme of GoogleNet architecture is going to be explained. Figure 18 shows the GoogleNet architecture.

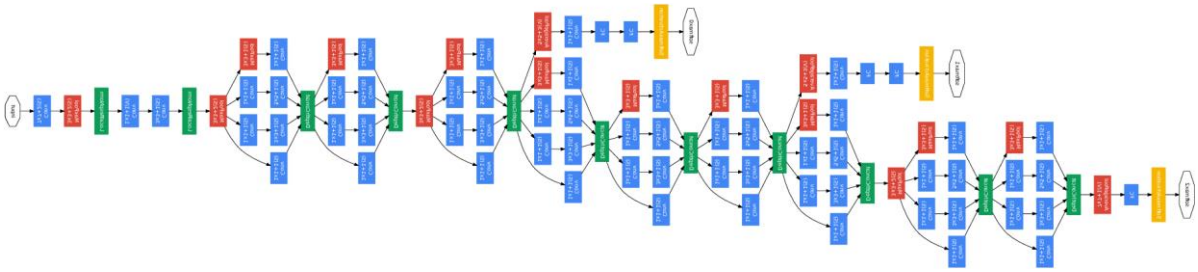


Figure 18 GoogleNet architecture [14]

The architecture of GoogleNet is composed by more than 100 individual layers, but regarding the depth, GoogleNet is 27 layers deep, not counting the input. The first 5 layers are 3 convolutional layers and 2 pooling layers that alternate as follows: C1-P2-C3-C4-P5. In future versions of this architecture, this group of 5 layers is named as stem module. The next 21 layers are 9 inception modules and 3 pooling layers. The last layer is a fully connected layer, that is the classifier output. So, in comparison with previous analyzed architectures, the computationally expensive fully connected layers were removed. All the convolution and fully connected layers are followed by a ReLU activation function. Apart from this, two auxiliary classification outputs are used. The goal of theses auxiliary outputs is to speed up the convergence rate and to combat the vanishing gradient problem [14, 15].

Since the creation of AlexNet, the trend was to increase the depth to develop more complex networks and to increase their performance and accuracy classifying and recognizing images. However, researchers found that increasing depth makes training more difficult. This happens because it is more difficult to propagate the gradients to the end of the

network and problems such as the vanishing gradient problem appeared. As a result of this problem, the accuracy of the networks gets saturated and then degrades rapidly. In order to eliminate these problems, in 2015 ResNet was published [16].

2.1.2.5. ResNet

ResNet or Residual Net was created in 2015 by Kaiming He et al. and is the winner of the ILSVRC 2015 with a top-5 error rate of 3.57%, the first CNN in ILSVRC exceeding the human-level accuracy. The main difference of this architecture is the use of residual blocks. These residual blocks solve the gradient vanishing problem and the accuracy problem mentioned before. Figure 19 shows a general scheme of the residual block used in ResNet.

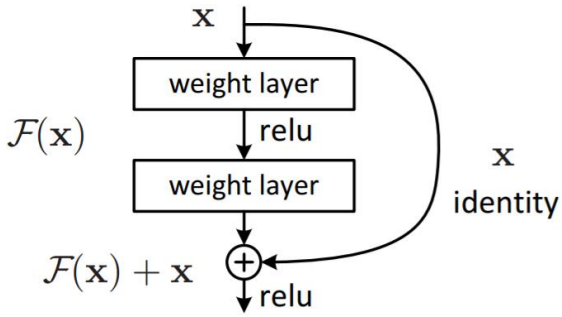


Figure 19 Residual block [16]

Residual blocks introduce an identity or skip connection that allows skipping one or more weight layers such as a convolutional layer. In case of ResNet, the main idea is to backpropagate through the identity function using a vector addition. With this technique the gradient is preserved in each layer avoiding the vanishing of the gradient. To this residual block some "bottleneck" layers are added as it happens in the inception modules in GoogleNet. As it can be seen in Fig. 20, the bottleneck design contains an extra 1x1 convolutional layer. This layer is responsible for increasing and decreasing the dimensions of the 3x3 convolutional layers [7,16].

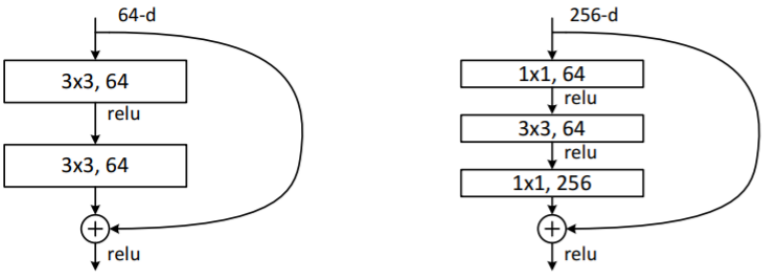


Figure 20 ResNet residual blocks [16]

Once the residual block has been analyzed, the general scheme of ResNet architecture is going to be explained. Figure 21 shows the ResNet architecture. As it can be seen in the upper right corner of Fig.21, depending on the number of total layers, the number of each cfg block will vary.

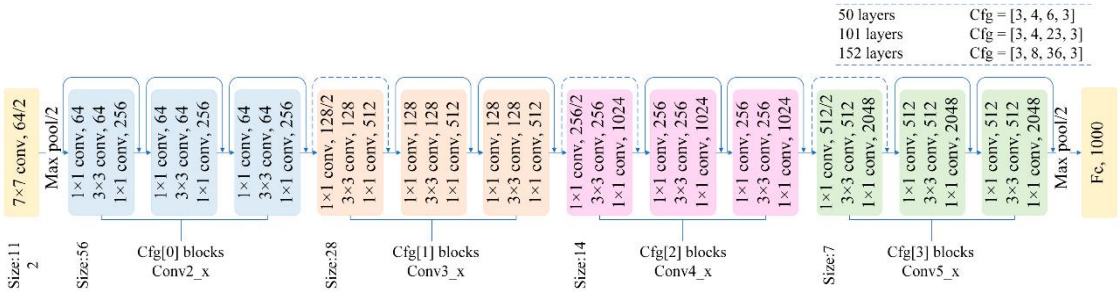


Figure 21 ResNet architecture [17]

The architecture of ResNet is composed by up to 152 layers. The first layer is made up of a 7x7 convolutional layer that uses 64 kernels and a 3x3 max-pooling layer. The next 150 layer are made up with 4 types of residual block. The last layer is a fully connected layer, that is the classifier output. Apart from these residual blocks, in this architecture two important techniques were applied to reduce the problem of gradient vanishing. The first technique is the ReLU activation function, a technique already used by several previously analyzed architectures. However, the second technique was used for the first time in this architecture. ResNet applies batch normalization technique after each convolutional layer. Batch normalization was applied to speed up learning and to obtain a faster convergence, thus reducing the problem of gradient vanishing. This is achieved by normalizing the output of the convolutional layer to obtain a zero mean and a unit standard deviation. [16, 54]

Nowadays, the researchers are focused on the creation of simpler architectures to be able to integrate them into low power devices such as mobile phones. The architecture analyzed above require a high computational cost, which makes their implementation in this type of system impossible. In the next chapter, a solution to this problem will be discussed.

2.1.3. Binarized Neural Network

During the last years, the growth of AI in countless applications such as sound or image recognition has been huge. This increase is mainly due to the evolution of deep neural networks. As explained before, these neural networks have been able to achieve great accuracies carrying out certain tasks, such as image recognition. However, in order to achieve this performance, the computational power and storage capacity required by the network has been impaired. Nowadays, these powerful deep neural networks are generally trained on GPUs with enormous power consumption, making their implementation almost impossible in low-power or limited-power devices such as IoT systems or mobile devices. This is one of the most challenging goals of researchers; being able to create a neuronal network keeping all the capabilities that deep neural networks offer, while reducing computational power and memory. To date, several proposals are being studied to satisfy these demanding requirements, being binary neural networks (BNN) one of the most promising solutions.

The main difference of BNNs over the CNNs is that the weights and activations are binarized to ±1. This considerably reduces the memory requirement and computational complexity to perform the matrix-vector-multiplication operation, since binarized weights and activations, convert the MVM operation into a bitwise operation. So, by binarizing the weights and activations, the multiplication operation is equivalent to a simple XNOR operation, as it can be seen in Fig. 22. To perform this operation, first the signed binary values are encoded with a 0 for -1 and 1 for +1 [18,19].

Encoding (Value)		XNOR (Multiply)
0 (−1)	0 (−1)	1 (+1)
0 (−1)	1 (+1)	0 (−1)
1 (+1)	0 (−1)	0 (−1)
1 (+1)	1 (+1)	1 (+1)

Figure 22 XNOR operation in BNN [1]

To perform the MVM operation, the results obtained in the XNOR operations must be added. This summation is performed by using a population count (popcnt) instruction. This instruction counts the number of 1s obtained in a group of XNOR operations. Once this value is known, it is multiplied by 2 and subtracted by the amount of XNOR operations done in the group. This operation is represented in (5), where **a** and **b** are two vectors with length n, and the elements of this vector are encoded binarized values (0,1) [18, 20].

$$\langle a, b \rangle = n - 2 \times popcnt(XNOR(a, b)) \quad (5)$$

This bitwise operation is much simpler and more efficient than the Multiply-and-Accumulate (MAC) operation used in a deep neural network, providing faster execution times with less hardware resources [18].

After this operation, a Batch Normalization (BN) layer is used. BN is very important for BNNs, because apart from accelerating the training it also reduces the overall impact of the weight scale. Usually, BN layers are combined with activation layers [18, 21].

Another important feature of BNNs is the robustness against external perturbations. In DNN when an input image shows small disturbances, it can provoke an erroneous classification of the image, especially in CNNs. However, in BNNs, these small perturbances will have a lower impact on the network activations because discrete values are used [18].

BNNs also have some negative aspects that are important to mentioned. The main drawback of this type of network is the accuracy. In general, the accuracy shown by BNNs in image recognition challenges, such as ILSVRC, are much lower than any of the CNNs analyzed in previous sections. However, in recent years researchers have proposed several improvements that manage to increase the accuracy of this type of network to more reasonable values in comparison to CNNs. For example, the BNN proposed in [21] shows a top-5 accuracy rate of 60.1% using AlexNet architecture, whereas the original AlexNet architecture achieves a top-5 accuracy rate of 84.6%. [18, 22].

In conclusion, BNNs can lead to great improvements related to power consumption, computation speed and required memory and accesses. These improvements are mainly due to the fact that the arithmetic operations of MVM are replaced by bitwise operations. Some studies indicate that power efficiency can be improved by more than one order of magnitude. However, the precision of these networks, despite of the improvements, do not reach the levels of CNNs. So, BNNs can be considered as an interesting solution to implement neural networks in low-power devices [18, 20, 23].

2.1.4. Digital vs Analog

In 1945 von Neumann presented a computer architecture that has been used and will continue to be used in modern computer systems. The von Neumann architecture is a simple computer architecture that consists of separating the central processor unit (CPU) from the main memory. So, during data processing, data movement is needed between the CPU and the storage device through a transmission line. This principle of operation has not changed in all these years. Despite of the advancement of technology, this architecture has become more complex. However, since in the last years the data volume has increased, the high latency and the energy consumption of data transmission have become the architecture bottleneck, called, “von Neumann bottleneck” or “memory wall” [24].

This problem has a direct impact on deep neural networks, where the amount of data to transfer between the CPU and the memory is huge, especially performing the matrix-vector-multiplication, one of the most important computational tasks. That is why in recent years, several researchers have been trying to solve this problem looking for different alternatives and finding new computational methodologies. In this search, most researchers have focused on two types of solutions.

The first type of solutions is in favor of keeping the von Neumann architecture. These solutions have been proposed and implemented for several years. The main idea of all these proposals consists of reducing or eliminating the limitations that von Neumann’s architecture presents, maintaining its principle of operation. This type of solutions is usually called digital accelerators. One of the first solutions to mitigate the von Neumann bottleneck was the implementation of fast GPUs. With the introduction of these devices, researchers were able to train neural networks 10-20 times faster [25]. Until present days, more solutions have been studied, such as using compressing techniques to reduce the amount of data that has to be transferred, but the most popular solution is known as near-memory computing and dates to the 1990s. Near-memory computing consists of placing processing and memory in a single package, thus greatly reducing the power required to move data. The concept known as near-memory computing has been improved in recent years due to advances in die stacking technology and commercialization of advanced memory modules such as hybrid memory cube (HMC), high bandwidth memory (HBM) and Wide I/O. However, despite of improving the characteristics of neural networks, there is still physical separation between memory and compute units [26, 27, 28].

The second type of solutions works on a new computer architecture where the expensive data movement between the memory and the computation units is eliminated. This solution is called in-memory computing and it is one of the most promising alternatives to von Neumann architecture. The main difference between near-memory computing and in-memory computing is that the first one focuses on moving data to computing, whereas the second one focuses on moving computing to data. In-memory computing solutions are usually called analog accelerators because when the processing is moved into the memory, analog computing is typically used. To perform analog computing, a large variety of nanoscale memory devices are investigated and each one offers different advantages and disadvantages. These circuits will be explained in detail in the next section.

In general, In-memory computing reduces power consumption and computing times by orders of magnitude compared to some digital accelerators, especially for layers with a lot of weights per neuron, like fully connected layers. For example, Ambragio S. et al in [29], demonstrated that the computational energy efficiency and the throughput per area obtained with their analogue in-memory computing design exceeds today’s GPUs by two

orders of magnitude. However, not everything is positive. The complexity of this type of memory is much greater than the ones that are available in the market. In addition, as it is a relatively new technology, there are many problems that need to be solved before the true potential of these memories is reached. That is why there is still a lot of work to do, so that in-memory computing can replace the solutions offered nowadays.

Figure 23 shows a general architecture of a conventional computing system and of an in-memory computing system.

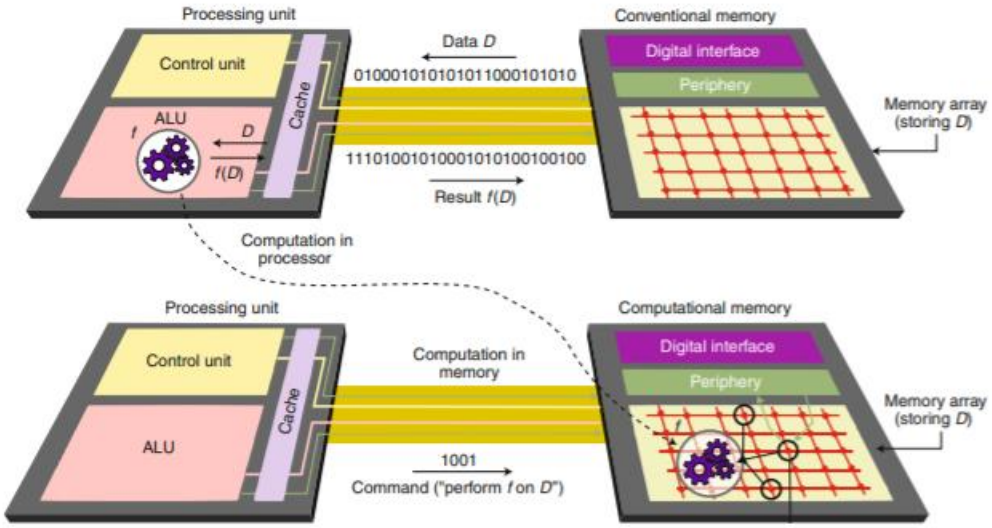


Figure 23 Conventional computing system vs in-memory computing system [27]

2.2. In-Memory Computing

Over the last years, the amount of information required by AI or machine learning systems to perform all tasks increases. This change has caused a dramatic increase in the amount of data stored in memories and especially in the amount of data transferred between different devices, such as from the memory to compute unit and vice versa. As technology evolved and the size of transistors decreased, the energy consumption due to computation decreased too. This reduction has reached a point where the energy consumption due to computing is orders of magnitude smaller than the energy consumption due to data transfer. So, to reduce the enormous cost of energy and delay produced by this data movement, many companies and researchers focused on this new concept.

The concept of in-memory computing exists since several years. However, it is starting to become real nowadays. This is mainly due to two reasons. The first reason is the introduction of new memory technologies and new memory architectures, which facilitate the implementation of in-memory computing. The other reason is that machine learning, AI and neural networks are becoming more and more important, and this in-memory computing concept can improve the performance of these systems.

The main idea of in-memory computing is to eliminate the data movement between the memory and computing unit by modifying the memory cell to embed the computation in the memory array. To achieve this, most circuits use analog rather than digital computing.

Analog circuits limit robustness, present nonidealities that limit computational SNR and in general, they are more complex.

Nowadays, there are three different ways of performing analog computing in this type of architecture: charge-mode analog in-memory computing, current-mode analog in-memory computing, and voltage-mode analog in memory computing. The main difference between these modes is the way in which they perform the MAC operation. However, even though there are solutions for each mode mentioned before, nowadays, the charge-mode is usually used to perform analog in-memory computing.

As mentioned in previous sections, the principal operation in neural network is the high-dimensionality matrix-vector-multiplications. To perform this MVMs, a lot of data needs to be moved from the memory to the computation, resulting in excessive power consumption. That is why most researchers focus on designing different circuits to perform this MVMs. Nowadays, there are two main groups of memories that perform this type of operation. Figure 24 shows the concept of in-memory computing and the two main groups of memory types.

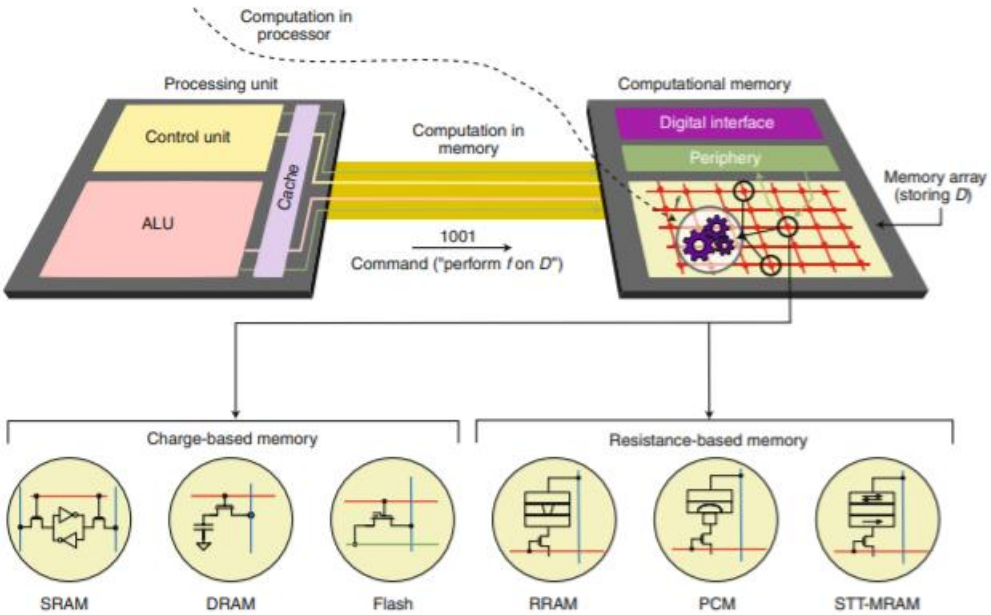


Figure 24 Memory types for In-Memory Computing [27]

2.2.1. Charge-based memory

The charge-based memories are divided into three principal types: Static Random-Access Memory (SRAM), Dynamic Random-Access Memory (DRAM) and Flash memories. From these three types of memories, the most used for in-memory computing is the SRAM, which gives 12 times greater energy savings compared to off-chip SRAM [7]. This type of memory can be implemented using the computing modes mentioned above.

2.2.1.1. Charge-Mode Computing

One of the main advantages of the charge-mode SRAM is that the additional components required to perform the MAC operation can be directly introduced into the bit cell of an SRAM without significantly increasing the size of the cell. Figure 25 shows a modified SRAM bit cell, called multiplying bit cell (M-BC), based on charge-mode computation [30].

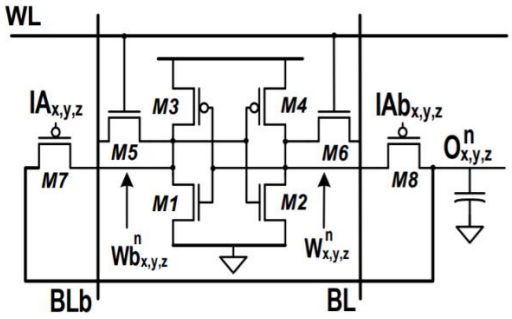


Figure 25 Multiplying Bit Cell circuit [30]

The M-BC consists of a six-transistor (6T) SRAM, 2 PMOS and a MOM capacitor. The binary values -1 and +1 are represented as GND and VDD respectively. To perform the MAC operation, the weights are stored in the standard 6T-SRAM cell. By adding the PMOS transistors driven by the input activation (IA) signals, an XNOR is built. This XNOR performs the multiplication operation between the IA and the weight. Since the weights and the input activations are binarized to ±1, the XNOR operation is equivalent to a multiplication. Then, the result obtained in this operation is sampled as charge on a MOM capacitor. The use of this MOM capacitor is due to two main reasons. The first reason is that it does not occupy any additional area since the capacitor is placed above the bit cell. The second reason is that the MOM capacitors allow highly linear and stable operations as they present excellent matching characteristics. This creates a design that achieves a high computational SNR [30].

2.2.1.2. Current-Mode Computing

As previously mentioned, most of the SRAM memories used for in-memory computing are based on charge-mode computation. However, there are other solutions such as those based on current-mode computation. In this solution, the standard 6T-SRAM cell is not modified, but to perform the MAC operation it is necessary to introduce some external components. Figure 26 shows this architecture.

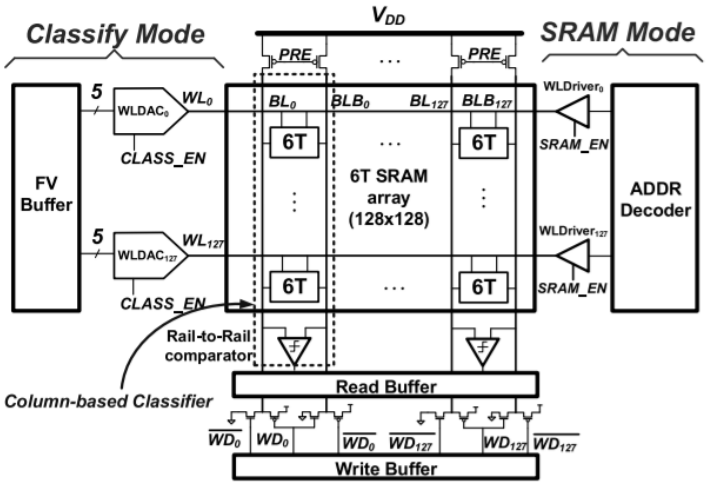


Figure 26 SRAM architecture based on current-mode in-memory computing [31]

As it can be seen in Fig.26, the architecture has two operation modes. In the SRAM Mode, the weights (w_i) are stored in the 6T bit cells as in a standard digital SRAM cell. In the Classify Mode, the MAC operation between the binary weights stored in SRAM and the 5-

bit input (x_i) is performed. For that, first the BL/BLB are precharged to V_{DD}. Then, the 5-bit input is converted to analog with a current-mode DAC (WLDAC). Figure 27 shows the working principle of this operation and the WLDAC circuitry.

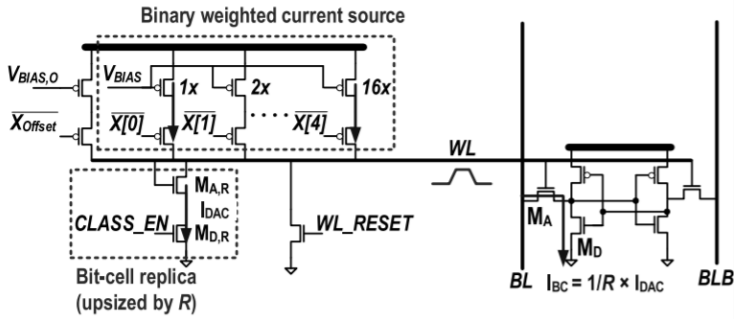


Figure 27 WLDAC circuitry and MAC operation [31]

The 5-b digital values X[4:0] are introduced in the WLDAC, formed by binary-weighted PMOS current sources. Then, the total current, I_{DAC}, runs through an upsized bit cell replica, formed by transistors $M_{A,R}$ and $M_{D,R}$. This circuitry only works in the Classify Mode, when the signal CLASS_EN is V_{DD}, activating the driver transistor replica $M_{D,R}$ and thus, representing a pull-down condition in a bit cell. The access transistor replica $M_{A,R}$, is self-biased to create a WL voltage related to I_{DAC}, creating a current mirror of this current as it can be seen in Fig.27. So, with this circuitry, an I_{BC} current that is roughly linear with the inputted digital value is achieved. Depending on the weight value stored in the SRAM, the cell will generate this discharging current I_{BC} in BL or BLB, as it can be seen in Fig.28. To finish the MAC operation, all the currents generated by the cells in a column will be added and the sense amplifier will compare the voltages in BL and BLB [31].

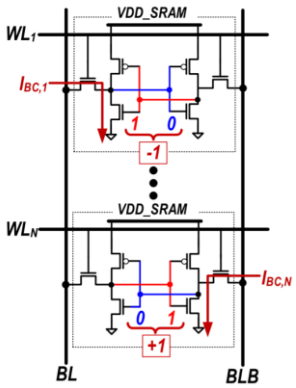


Figure 28 Column-based classifier [31]

This architecture, as the charge-mode architecture, shows several advantages in terms of latency and bandwidth over traditional architectures. However, the current-mode architecture presents an important drawback compared to the charge-mode architecture. This problem is related to the SNR. As explained before, in the charge-domain architecture, with the introduction of a MOM capacitor a high computational SNR is achieved despite of working with an analog circuit. However, in current-mode architecture, the important non-linearities, offsets and variations provoke the degradation of the SNR. To reduce the effect of these disturbances some techniques are used such as some training algorithms, or the Error-Adaptive Classifier Boosting (EACB) [31].

2.2.1.3. Voltage-Mode Computing

To finish with the SRAM memories used for in-memory computing, a solution based on voltage-mode computation will be discussed next. In this solution, as in the current-mode solution, the architecture has two operation modes. In the Memory Mode, the weights are stored in the standard 6T-SRAM cell, as a regular SRAM. In the XNOR Mode, it performs the binary MAC operations activating all rows simultaneously. One important characteristic of this architecture is that it allows binary and ternary inputs, but only the operation with binary inputs will be analyzed. Figure 29 shows the voltage-mode architecture [32].

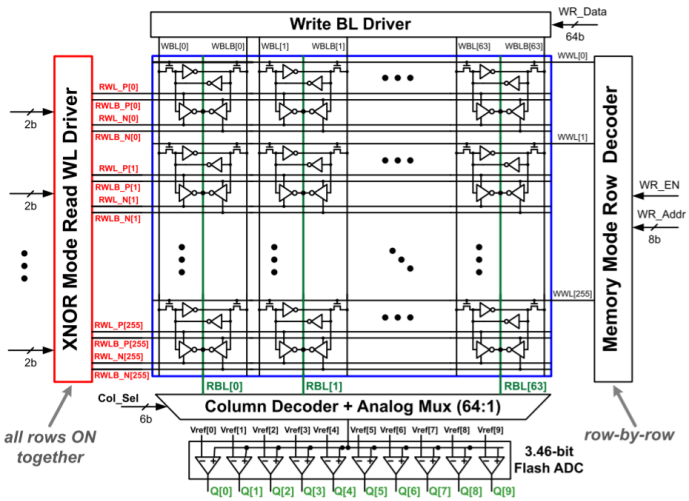
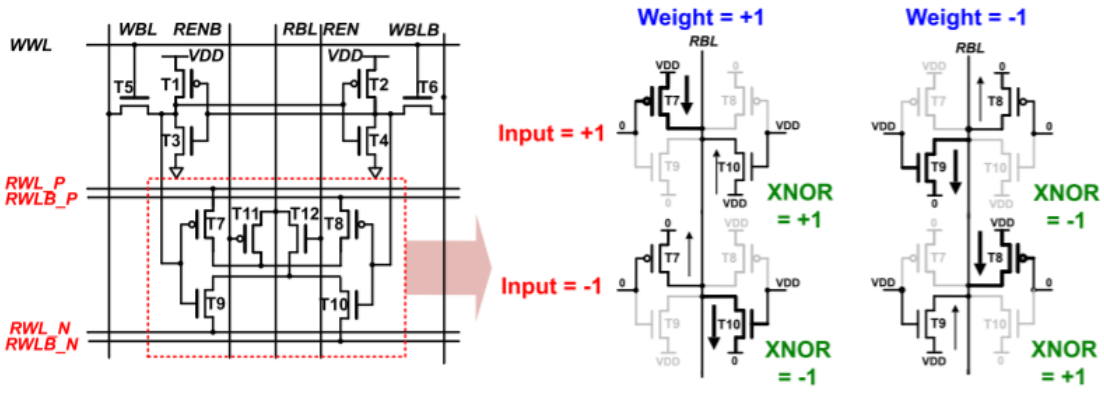


Figure 29 SRAM architecture based on voltage-mode in-memory computing [32]

As it can be seen in Fig.29, this architecture consists of 256x64 bit cell array and peripherals. To perform the binary MAC operation, this solution uses a 12T bit cell: a standard 6T-SRAM cell composed of transistor T1-T6, an additional pull-up (PU) and pull-down (PD) network formed by transistors T7-T10 and transistors T11 and T12, which are used to enable or disable the PU/PD circuits. Figure 30 shows a detailed view of the bit cell and how the MAC operation is performed [32].



RWL Driver Logic

RWL \ Input	+1	-1
RWL_P	VDD	0
RWL_N	VDD	0
RWBL_P	0	VDD
RWLB_N	0	VDD

XNOR Value Mapping Table

XNOR \ Input	Strong pull-up	Weak pull-up	Strong pull-down	Weak pull-down
+1	1	1	0	0
-1	0	0	1	1

Figure 30 Bit cell design and MAC operation [32]

As it can be seen in the RWL Driver Logic table above, depending on the value of each input bit, the WL controller will set values to 4 different read wordlines (RWL). Then, the value of the weight stored in each 6T-SRAM cell will activate or deactivate some transistors in the PU/PD network. For example, if the stored weight is +1, then the transistor T7 and T10 will be activated. Once the PU/PD network is activated, the XNOR operation between the RWLs and the binary weights is performed, producing a "+1" with a strong (PMOS) or weak (NMOS) PU or producing a "-1" with a strong (NMOS) or weak (PMOS) PD, as it can be seen in the "XNOR Value Mapping Table" above. So, in other words, when the XNOR operation is "1" the bit cell will behave as a PU and when the XNOR operation is "0" as a PD [32].

As mentioned before, in this architecture, 256 bit cells are connected to RBL in a column. So, each bit cell in the column will behave as a PU or a PD. Figure 31 shows a general circuit to calculate the final output voltage, V_{RBL}.

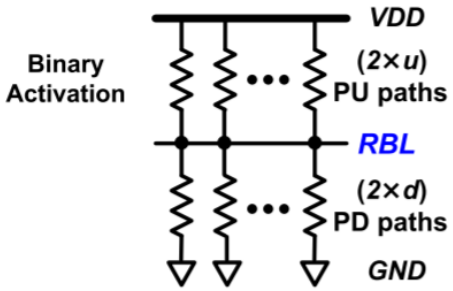


Figure 31 PU/PD general circuit to calculate VRBL [32]

If the number of rows is N, the result of the MAC operation will range from -N to +N. Assuming that *u* indicates the number of bit cells that behaves as a PU among N cells and *d* indicates the number of bit cells that behaves as PD, (5) and (6) can be considered. Observing Fig.31, V_{RBL} can be represented as in (7), as a voltage divider, that shows a linear relationship with the MAC value. Finally, each V_{RBL} is read by a 3.46-bit flash ADC [32].

$$N = u + d \tag{5}$$

$$MAC = u - d \tag{6}$$

$$V_{RBL} = \frac{2u}{2u + 2d} = \frac{MAC + N}{2N} \tag{7}$$

The principal problem of this architecture is the mismatch, and the non-linearity produced by the PU/PD resistances. Some of these problems are solved by placing non-linear ADC levels and by tuning the PMOS body bias in the bit cell array. Compared to charge-mode architecture, energy efficiency is approximately 38% lower. However, the voltage-mode architecture shows higher accuracy and smaller area than the other architectures analyzed [30, 31, 32].

2.2.2. Resistance-based memory

The resistance-based memories are divided into three principal types: Resistive Random-Access Memory (RRAM), Phase Change Memory (PCM) and Spin Transfer Torque Magnetoresistive Random Access Memory (STT-MRAM). These devices have different

trade-offs in terms on speed, write current, density, and endurance. However, the main advantage of all these types of memories is that they show multilevel programmability by applying electrical pulses, which makes it ideal for in-memory computing applications [7, 27].

2.2.2.1. RRAM

RRAM technology dates back to at least the 1960s and nowadays is the most popular and mature memory among those mentioned above. RRAM consists of a metal-insulator-metal (MIM) structure, where the formation and dissolution of conductive filaments (CF) through the insulator provides a high or low conductance state, in other words, a low or high resistance state, respectively. The CF formation and dissolution are reversible, and it is induced by an electric pulse. Figure 32 shows an RRAM device in the low resistance state and some general characteristics of this type of devices [27, 33].

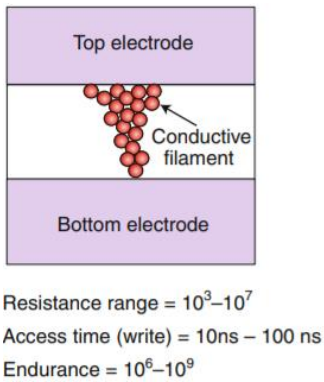


Figure 32 RRAM device [27]

2.2.2.2. PCM

PCM technology also dates back to 1960s. PCM is based on the property of certain types of materials known as chalcogenide material, such as $Ge_2Sb_2Te_5$. When these materials are exposed to the Joule effect, the material switches in a reversible and rapid manner from a highly resistive amorphous phase to a highly conductive crystalline phase. Usually, the bottom electrode confines heat and current. Figure 33 shows a typical mushroom-type PCM device in the high resistance state and some general characteristics of this type of devices [27, 33].

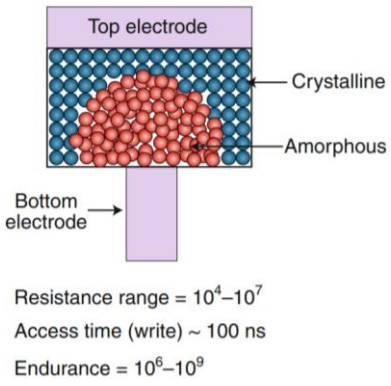


Figure 33 PCM device [27]

2.2.2.3. STT-MRAM

STT-MRAM is a relatively new technology. STT-MRAM consists of a magnetic tunnel junction structure composed by two ferromagnetic metal layers, such as CoFeB, separated by a thin tunnel oxide such as MgO. One of the metal layers (pinned layer) has the magnetic polarization structurally fixed whereas the polarization of the other one (free layer) is free. The pinned layer behaves as a reference. So, depending on whether the two magnetic polarizations are parallel or antiparallel, the device assumes a low or high resistance, respectively. To change the magnetic orientation of the free layer, an opposite voltage is applied. Figure 34 shows an STT-MRAM device and some general characteristics of this type of devices [27].

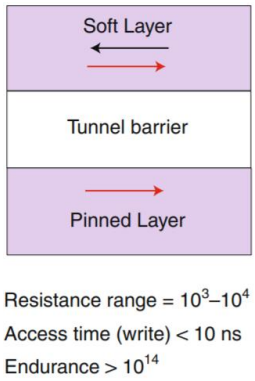


Figure 34 STT-MRAM device [27]

Each one of these memories has its own advantages and disadvantages. RRAM has the advantage of using well known materials in semiconductor manufacturing. Apart from this, RRAM and PCM have larger resistance ranges than STT-MRAM, allowing to store intermediate resistance at array level. However, these two memory types present worse cycling endurance and access times or write speeds than STT-MRAM [27].

Resistance-based memories can be used to perform in-memory computing, concretely to perform the MAC operation, since these memories can be used as programmable resistive elements by applying SET or RESET voltage pulses. The most compact and popular architecture used to perform this operation is a crossbar array. Figure 35 shows a memristor crossbar architecture used to perform MVM operation achieving high levels of parallelism performing all the dot products in a single step.

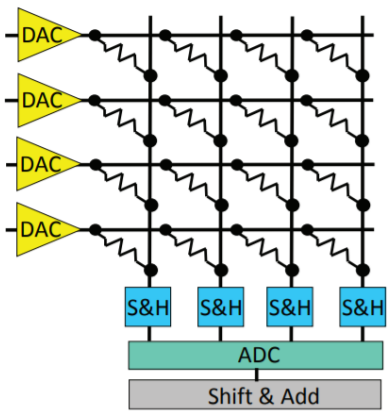


Figure 35 Memristor crossbar architecture [34]

As it can be seen above, every bitline is connected to every wordline via resistive memory cells. To perform the multiplication the weight is stored as the conductance of the resistor, $G_i = \frac{1}{R_i}$, and the input activation is the signal V_i entering the resistance. So, the current that goes through each resistor represents the multiplication between the input activation and the weight. The MAC operation finishes by adding all the currents that go through each memristor in a column. Figure 36 shows a detailed view of how this MAC operation is performed [34].

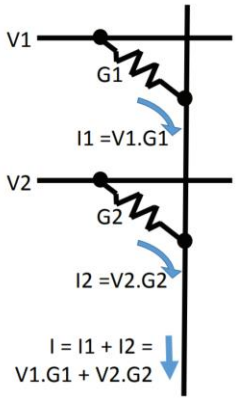


Figure 36 MAC operation with memristors [34]

There are several works where cells composed of resistive memory technology are used in convolutional neural network. One of the most interesting proposals is ISAAC architecture [34]. Figure 35 shows this architecture. ISAAC researchers used RRAM memories to develop their cells because it allows better precision than PCM [34].

Once the most important computation modes have been analyzed, the characteristics of the previously analyzed solutions will be compared. Table 1 compares some general characteristics of four architectures that have been explained previously.

Device	Charge-Mode SRAM	Current-Mode SRAM	Voltage-Mode SRAM	RRAM (AnIA)
CMOS technology	65 nm	130 nm	65 nm	22 nm
Accuracy MNIST/CIFAR-10	98.6 %/84.1 %	90%	98.8 %/88.8 %	-
Energy efficiency (TOPS/W)	866	11.5	403	1050-1500
Throughput (GOPS)	18876	57.1	600	23500
Reference	[30]	[31]	[32]	[53]

Table 1 Comparison of different types of in-memory computing solutions

As it can be seen in the table above, the energy efficiency, throughput and the accuracy are considerably worse for the architecture using current-mode SRAM. These values would improve using a smaller CMOS technology, but they would surely still be much worse than the other architectures, especially the energy efficiency. As for the other two types of in-memory computing SRAM architectures, it can be seen that the charge-mode architecture offers the best features, especially in terms of throughput.

Regarding the RRAM, it can be seen that the energy efficiency and throughput obtained are better than those obtained in the three in-memory computing SRAM solutions. In addition, as previously mentioned, area is one of the most important advantages of this type of solutions compared to SRAM solutions. There is no information about the accuracy that this solution provides, but based on [53], this solution does not improve the accuracy with respect to other solutions. However, this comparison is not entirely fair, since the technology used in the design of the RRAM solution is much smaller than the technology used in the SRAM solutions. Even so, the characteristics shown by this solution are impressive.

3. In-Memory-Computing CNN Accelerator Structure

In recent years, researchers have been focusing on finding solutions to perform the dominant computational task in CNNs, the MVM, in the most efficient way. Previous studies demonstrated that in order to avoid high energy consumption and delay caused by the movement of data in CNNs, it is necessary to introduce the concept of in-memory computing. In the previous section, different ways of performing MVM using the concept of in-memory computing have been explained. In this section, a solution to perform MVM using the concept of in-memory computing will be explained and designed.

The CNN accelerator designed in this thesis is the charge-domain architecture that has been analyzed in the previous section. This architecture was proposed by Hossein Valavi et al. in 2018 [30]. In this proposal, the MVM is performed in the neuron array, concretely in the neuron tile, inside the Hidden Layer (HL) of a deep CNN. Figure 37 shows the architecture of this hidden layer.

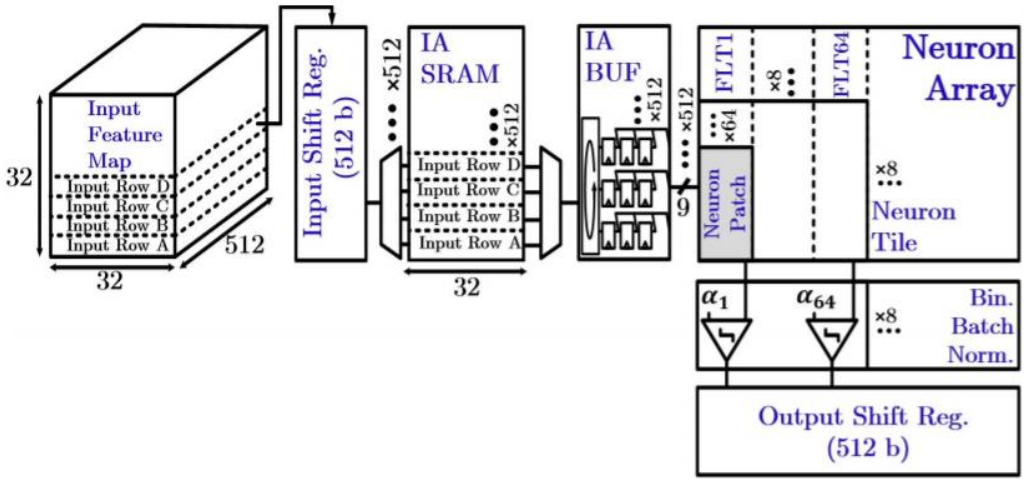


Figure 37 Architecture of Hidden Layer (HL) [30]

As it can be seen above, before performing neuron filtering to each pixel, of depth up to 512, of the input feature map, these pixels must be restructured. This is done through 3 components. The input shift register is used to facilitate the testing interface. The IA SRAM works as a line buffer and consists of 4 sets of 512 columns. Then, the three out of four IA columns to be processed are shifted to an IA buffer which consists of 3-b shift registers with round-robin input interface. Finally, the 3x3xd (depth up to 512) IAs to be filtered are transmitted in parallel over the neuron array, where the MVM operation is performed. Once the MVM operation is performed, the 512 computed pre-activations (PA) are introduced in parallel to a binarizing batch normalization circuit to obtain the binary output activation for the pixel [30].

The neuron array consists of 8x8 neuron tiles and each neuron tile consist of 64x64 neuron patches. Each of the 64 columns of the neuron tile corresponds to a different neuron filter, so the 64 neuron patches in each column belong to a single neuron filter. Figure 38 shows the structure of a neuron tile and a neuron patch.

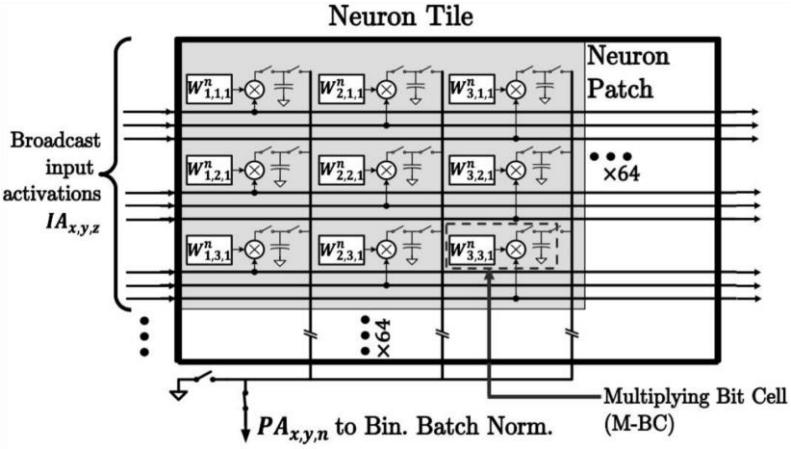


Figure 38 Structure of a neuron tile [30]

As it can be seen above, each neuron patch processes 3x3 binary input activations (IAs) from the IA BUF. For that, each IA is processed by a multiplying bit cell. The M-BC multiplies the corresponding 1-b IA with a previously stored 1-b filter weight and stores the result as charge on a local capacitor. When all the multiplications are performed, all capacitors are shorted together to perform the charge accumulation completing the MVM operation [30].

As explained in section 2.2.1, the M-BC is a modified SRAM based on charge-mode analog in-memory computing. The M-BC has two main functions: to store a 1-b filter weight and to perform the MAC operation. For that, the M-BC consist of a 6T-SRAM, 2 additional PMOS transistors and a MOM capacitor. Figure 25 shows the schematic of the M-BC. The 6T-SRAM takes care of storing the filter weight for the MAC operation, whereas the 2 PMOS and the MOM capacitor are used to perform the multiplication and accumulation, respectively. So, when designing the M-BC it is very important to consider these two functions [30].

As introduced, the MAC operation starts with the storage of pre-trained weights on each M-BC of the neuron array. Once the pre-trained weights are stored on the neuron array, these values will not change again until the neural network is retrained again. This operation is performed via SRAM read/write circuitry. In the next section, the design of a SRAM cell is going to be explained.

3.1. Static Random-Access Memory (SRAM)

3.1.1. Introduction

Nowadays SRAM occupies a large segment of modern System-on-Chips (SoC) and microprocessors, and they are widely used in many applications. The SRAM memories are composed by a memory cell and some peripheral circuits which are in charge of controlling the write and read operations of the memory. During write operations, incoming data is driven on the bitlines to be stored in the enabled bit cell. During read operations, the internal data of the previously enabled bit cell is sensed in the bitlines and translated to the output [35, 36].

The conventional architecture for an SRAM consists of an array of memory cells along with the circuitry needed to access each bit cell in the array and to perform the read write operations. Each row in the array corresponds to a concrete address in the array, and each

pair of columns are set to different voltages to perform write and read operation. For read operation, usually a sense amplifier is placed at the end of each pair of columns to detect voltage differences rapidly. Figure 39 shows a conventional architecture of an SRAM memory array.

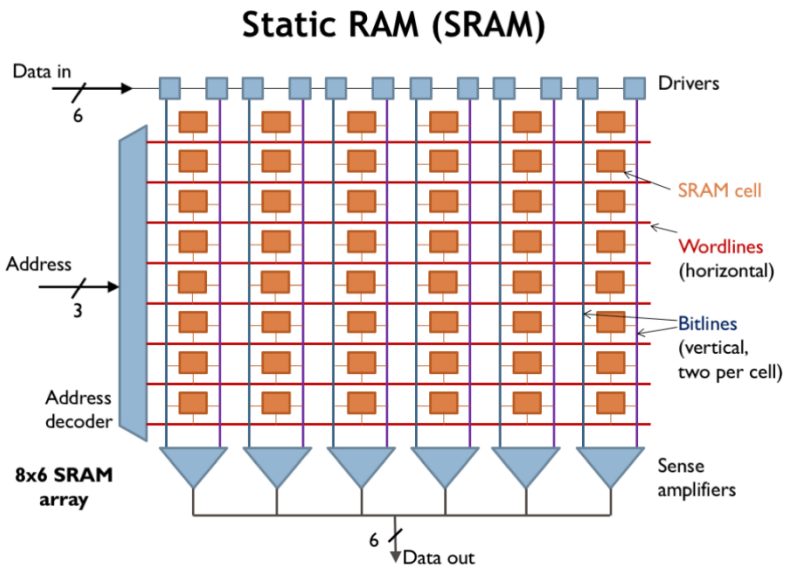


Figure 39 SRAM memory array [37]

The most important element of the SRAM is the memory cell. This memory cell must be able to read, write and hold data as long as the power is applied. There are several types of SRAM cells, but the most common is the 6T-SRAM. The 6T-SRAM cell contains a pair of cross-coupled inverters wired in a positive feedback loop creating a bistable storage element and a pair of access or pass transistors (A_1, A_2) controlled by a wordline and connected to the two bitlines to read and write the state. Figure 40 shows the schematic and the block diagram of a 6T-SRAM [35].

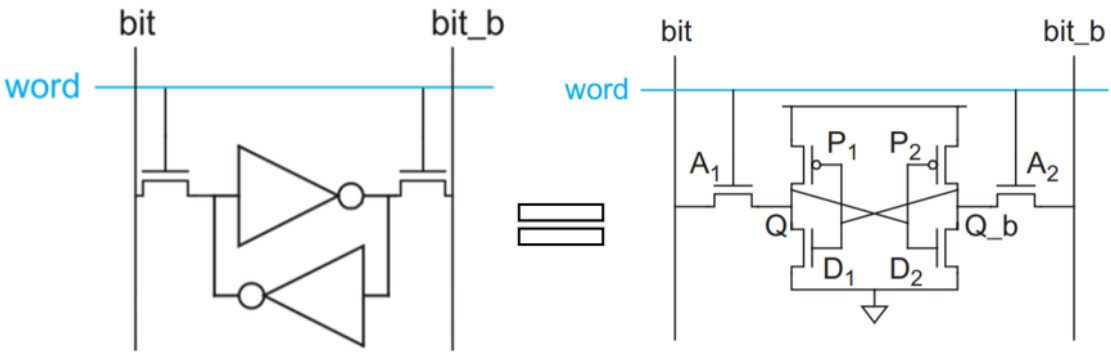


Figure 40 Block diagram and schematic of a 6T-SRAM [35]

As introduced before, the SRAM bit cell must perform three operations: hold data, read data and write data. Next, these three operations will be explained in detail.

3.1.2. Hold Data

As introduced before, the SRAM bit cell should provide a stable storage as long as there is power in the circuit. This work is performed by the two looped inverters when the access transistors are switch off, that is, where the wordlines are tied to ground. When this happens

the value of each storage node will be reinforced, ensuring that the value of the SRAM cell is maintained. To better explain the operation, it is assumed that in the storage node Q_b there is a one. As there is a one in Q_b, the lower inverter will drive the node Q to zero and this zero will drive Q_b to one, ensuring the initial values.

3.1.3. Read Operation

The read operation starts by precharging all the bitlines to V_{DD} and then disconnecting and leaving them floating at one. Once this is done, the wordline is raised activating the access transistors and connecting the storage nodes of all the bit cells in a row to their bitlines. When this happens, one of the bitlines in each memory cell will be pulled down to ground. This transition is normally slow because the line has a very large capacitance compared to the size of the inverter transistor. So, in order to not have to wait for the complete discharge of the bitline, a sense amplifier is usually placed on the periphery, since this device is capable of detecting voltage differences around 10% of the supply voltage.

When the wordline is raised, one of the bitlines will be discharged through the access and pull-down transistors. However, when this happens, the current going through the access transistor will try to raise the storage node to one. So, to minimize this effect, the pull-down transistor (D_1, D_2) needs to be much stronger than the pass transistor. This requirement is known as read stability and it must be considered when designing an SRAM, because it can cause many problems. Figure 41 shows the waveforms of the SRAM cell when reading the value '0' [35].

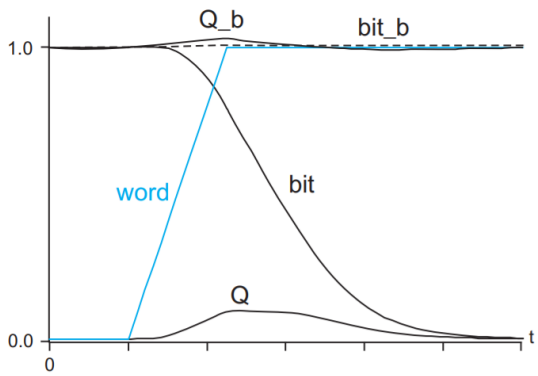


Figure 41 '0' read operation for SRAM cell [35]

3.1.4. Write Operation

To perform the write operation, instead of precharging the bitlines to V_{DD} and then leaving them floating, the bitlines are forced to the desired voltages. Once this is done, as in the read operation, the wordline raises, activating the access transistor and connecting the bitlines with the storage nodes. When trying to write the opposite value to the one already stored in the SRAM cell, the access transistor is forcing a value in the storage node while the inverter is trying to hold the previous value. In order to perform the write operation correctly, the access transistor should be stronger than the inverter. However, according to the read stability requirement, the access transistor should be weaker than the pull-down transistor of the inverter. Therefore, to ensure the proper performance during write operation, the access transistor must be stronger than the pull-up transistor (P_1, P_2) of the inverter. This requirement is known as writability or write stability. Figure 42 shows the waveforms of the SRAM cell when writing the value '1' [35].

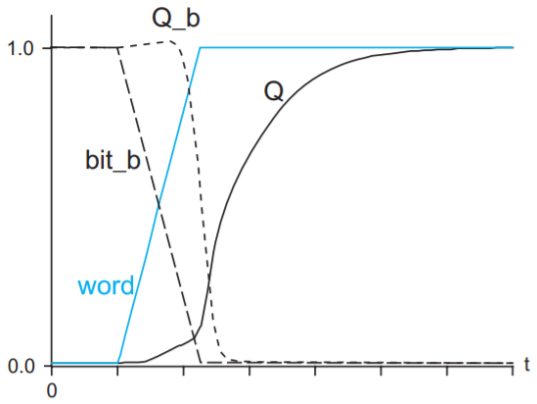


Figure 42 '1' write operation for SRAM cell [35]

Therefore, when designing an SRAM cell, it must be considered that during a read operation the access transistor has to modify as little as possible the storage node of the cell, and that during a write operation the access transistor has to be able to modify the value of the storage nodes imposed by the inverters. As mentioned before, this tradeoff is usually achieved by making inverters with weak pull-ups and strong pull-downs.

3.1.5. SRAM Cell Stability

One of the most important criteria when designing and SRAM cell is the stability, especially when reading or holding a value in the memory. If the SRAM cell is not stable during these operations, it can result in a loss of data. To measure the stability of the cell, the Static Noise Margin (SNM) is calculated. The SNM measures the amount of noise that can be applied to the inputs of both inverters before the stable state is lost. The most important measures according to the stability of the SRAM are the Hold Static Noise Margin (HSNM) and the Read Static Noise Margin (RSNM). Apart from the fact that the SRAM must be stable when reading and holding, it must be also stable when writing, although this measure is less important than the previous ones. The Write Static Noise Margin (WSNM) measures the minimum voltage required to flip the state of an SRAM to the desired value [35, 38].

3.1.5.1. Hold Static Noise Margin (HSNM)

During hold operation the pass transistors are off, so the bitlines and the storage nodes are not connected. Figure 43 shows the test circuit used to determine the HSNM. A noise source V_n will be applied at the input of each inverter. To measure the HSNM of the SRAM cell, first it is necessary to obtain the static voltage transfer characteristics (VTCs) of the two inverters. These curves are usually called butterfly curves due to its shape. Once the VTCs are obtained, the HSNM "will be determined by the length of the side of the largest square that can be inscribed between the curves" [35]. Figure 44 shows an example of the calculation of the HSNM.

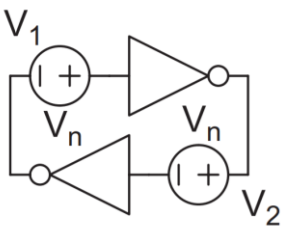


Figure 43 Circuit to find the Hold Static Noise Margin (HSNM) [35]

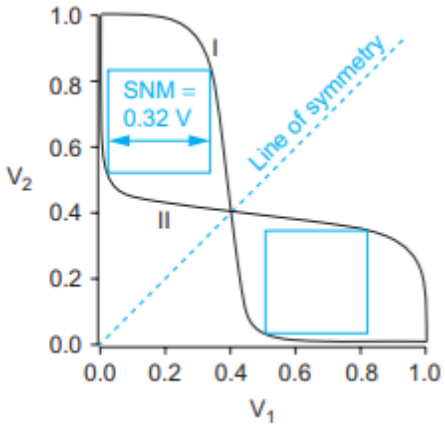


Figure 44 Butterfly curves indicating the Hold Static Noise Margin (HSNM) [35]

3.1.5.2. Read Static Noise Margin (RSNM)

During read operation the bitlines are precharged and the access transistors are activated to connect the bitlines with the storage nodes. Figure 45 shows the test circuit used to determine the RSNM. As mentioned before, during read operation, the pass transistor will try to pull the low storage node up. This effect will distort the VTCs and therefore, will reduce the SNM compared to HSNM. A solution to increase the RSNM of the SRAM, is to increase the size of the pull-down transistor of the inverters. However, this will also increase the total area of the SRAM cell. Figure 46 shows and example of the calculation of the RSNM [35].

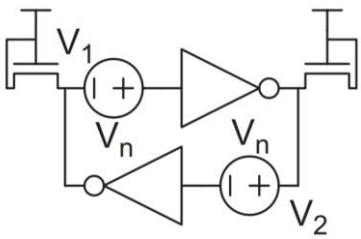


Figure 45 Circuit to find the Read Static Noise Margin (RSNM) [35]

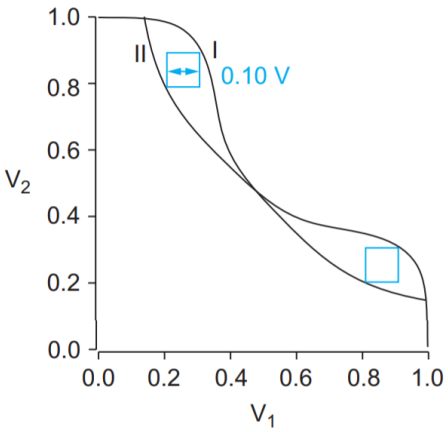


Figure 46 Butterfly curves indicating the Read Static Noise Margin (RSNM) [35]

3.1.5.3. Write Static Noise Margin (WSNM)

During write operation, the bitlines are set to a desired value and the pass transistors are activated to connect the bitlines with the storage nodes. Figure 47 shows the test circuit used to calculate the WSNM, that is very similar to the one used to calculate RSNM. As

mentioned before, during write operation, the pass transistor must overpower the pull up transistor of the inverter without breaching the read stability requirement. This will also modify the VTCs, reducing the SNM compared to HSNM. Unlike the previous two measurements, the WSNM will be “the size of the smallest square inscribed between the two curves” [35]. Figure 48 shows and example of the calculation of the WSNM.

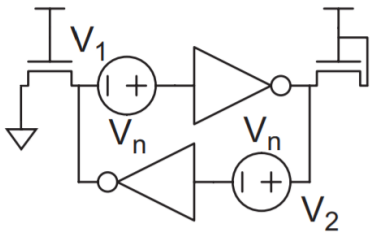


Figure 47 Circuit to find the Write Static Noise Margin (WSNM) [35]

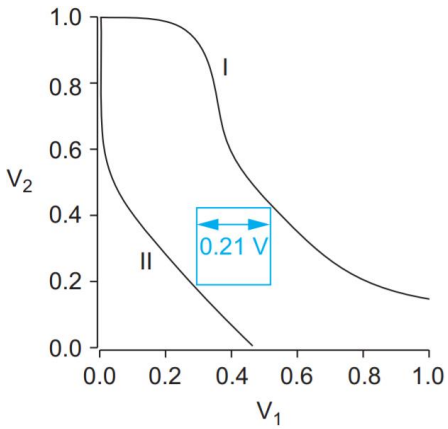


Figure 48 Butterfly curves indicating the Write Static Noise Margin (WSNM) [35]

3.1.6. SRAM Area and Power

Apart from the stability of the SRAM cell, it is very important to consider the area and the power consumption when designing an SRAM cell. As explained before, usually a memory array is composed of a large number of memory cells. For example, in the circuit being analyzed, the neuron array is composed of more than 2 million modified SRAM cells. So, a small difference in the size or power consumption of these modified SRAM cells can become very important. Usually, the energy per read and write operation is also calculated.

3.2. Multiplying Bit-Cell (M-BC)

3.2.1. Introduction

As mentioned before, to perform the MAC operation, apart from the SRAM cell, the design proposed by Hossein Valavi et al. [30] includes 3 extra elements: 2 PMOS transistors driven by the input activation signals and a 1.2 fF MOM capacitor. Figure 49 shows the schematic of the M-BC.

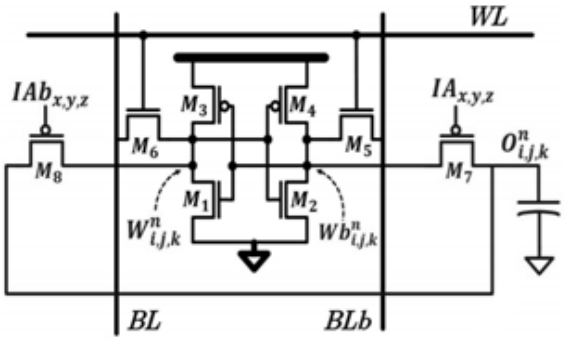


Figure 49 M-BC schematic [30]

Apart from these extra elements necessary to create the M-BC, two more components must be added to perform the MAC operation correctly. To store the result of the XNOR operation in the MOM capacitor, Valavi et al. proposed the use of PMOS transistor. These PMOS transistors will have no problem storing a '1' in the capacitor. However, when the XNOR result is a '0', the PMOS transistor will not be able to store a strong '0' in the capacitor. This is because when the transistor has to pass a '0', the parasitic capacitances of the source are discharged until $V_{GS} = |V_{thp}|$. At that moment, the transistor tuns off and the MOM capacitor will not reach a strong '0', reducing considerable the voltage range of the M-BC. To solve this problem, the paper adds a single NMOS transistor that discharges all the MOM capacitors of a neuron filter before performing the MAC operation. So, if the result of the operation is '0', the capacitor will be already discharged [30].

Apart from this discharge transistor, the paper also adds a transmission gate between the MOM capacitor and the PA signal. The main purpose of this component is to isolate the MOM capacitor from the output and thus not influence the XNOR operation of each M-BC. Figure 50 shows the schematic a M-BC and the necessary extra elements mentioned above [30].

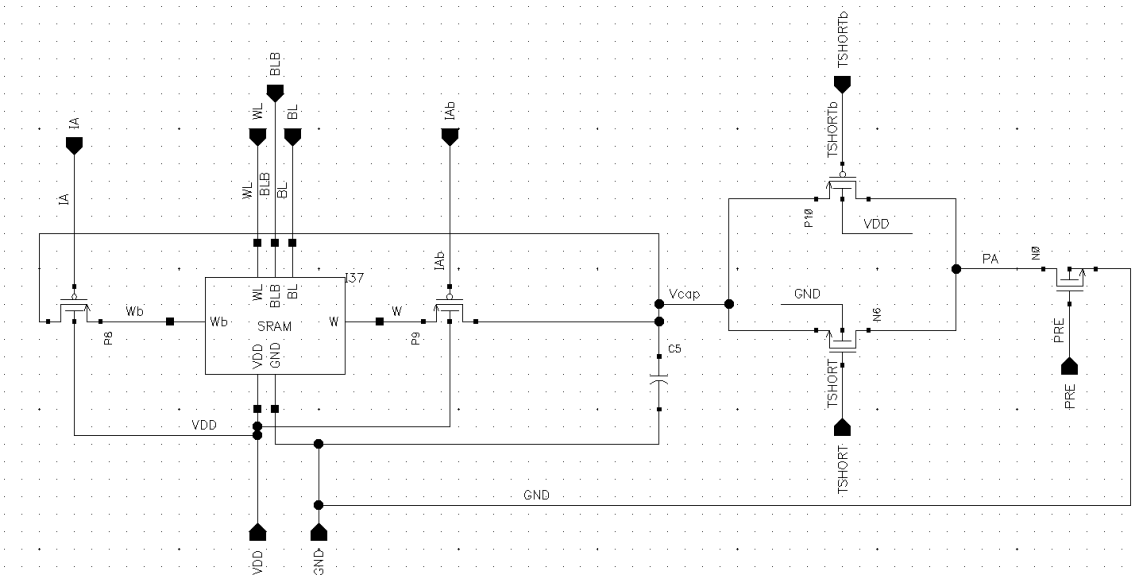


Figure 50 M-BC complete schematic

3.2.2. Multiply and Accumulate (MAC) operation

The MAC operation is performed in three steps. The total duration of this operation proposed in [30] is of 250 ns. These three steps will be explained below.

3.2.2.1. Reset Phase

First, as mentioned above, all MOM capacitors in a neuron filter are discharged to GND. For that, the transmission gate (via TSHORT/TSHORTb) and the discharge transistor (via PRE) are activated. This will connect all the MOM capacitors with the PA node. The SRAM will be disconnected during this phase by keeping IA and IAb high. This phase lasts 100 ns [30].

3.2.2.2. Binary-Multiply Phase

After discharging all the capacitors, the binary multiplication or XNOR operation will be performed. For this, the transmission gate and the discharge transistor will first be deactivated to not interfere with the operation. Then, IA and IAb will take their corresponding values, activating a single PMOS and charging or holding the capacitor to V_{DD} or GND respectively, depending on the result of the operation. This phase lasts 100 ns [30].

So, during this phase, depending on the result of the XNOR operation, a different charge will be stored on the capacitor. Equation (8) shows how this operation is performed, where out_i indicates the binary result of the XNOR operation.

$$Q_i = out_i \cdot V_{DD} \cdot C_{MOM} \qquad out_i \in \{0,1\} \tag{8}$$

3.2.2.3. Accumulate Phase

Once the results of every M-BC are stored in each MOM capacitor, the SRAM will be disconnected again by keeping IA and IAb high and the transmission gate will be activated connecting all the capacitors in a neuron filter. When this happens, the charges of all the MOM capacitors will be added depending on the result of each XNOR operation. This operation is represented in (9)

$$Q_{Output} = \sum_N Q_i = \sum_N (out_i \cdot V_{DD} \cdot C_{MOM}) = V_{DD} \cdot C_{MOM} \cdot \sum_N out_i \qquad out_i \in \{0,1\} \tag{9}$$

So, considering a column with a total of N M-BCs, the voltage of the entire output column is defined by (10).

$$V_{Output} = V_{PA} = \frac{Q_{Output}}{C_{total}} = \frac{V_{DD} \cdot C_{MOM} \cdot \sum_N out_i}{N \cdot C_{MOM}} = \frac{V_{DD} \cdot \sum_N out_i}{N} \tag{10}$$

As it can be observed, the output voltage only depends on the power supply, the number of M-BCs and the results of each XNOR operation. Therefore, considering the neuron filter in the deeper Hidden Layer, with depth equal to 512, PA can take $(3x3x512) + 1 = 4609$ levels between GND and V_{DD}, centered at mid-rail, which corresponds to an approximate resolution of 12-bits. This phase lasts 50 ns [30].

Figure 51 shows the phases explained above for three M-BCs.

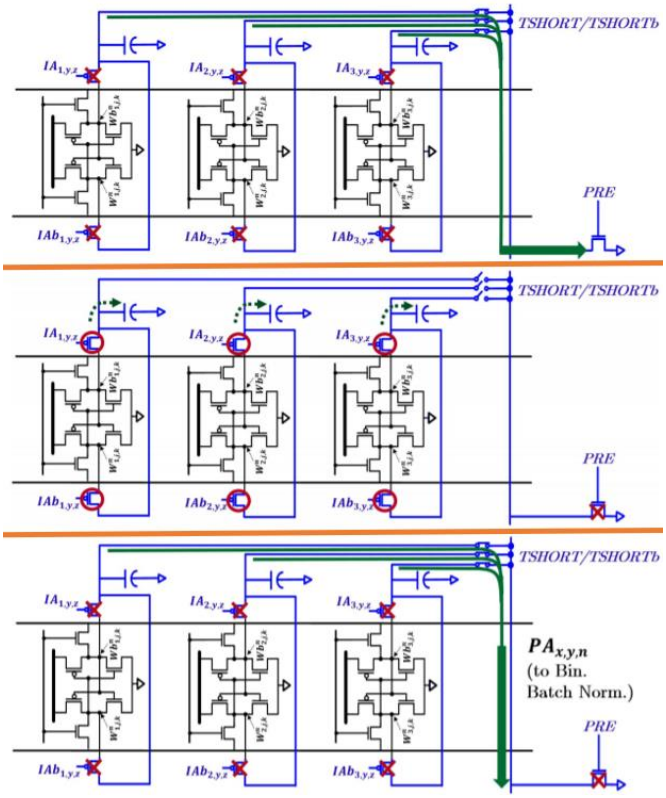


Figure 51 MAC operation performed by 3 M-BCs in 3 phases [30]

3.2.3. M-BC Stability

As mentioned before, the weights stored in the SRAM are pre-trained values that are not changed regularly. That is why the stability of these values is crucial for a good performance of the neural network. During reset phase, the storage nodes are exposed to a pull-down condition, so when designing the M-BC it must be taken into account that the node that stores a '1' does not invert its value [30].

Apart from the stability of the storage nodes, it is important to consider the parasitic elements that will be introduced, especially in the PA signal, due to the great length of the line and the routing. These parasitic elements are important, because they will produce a reduction in the dynamic range of the PA signal. This problem will be explained in more detail later [30].

3.2.4. M-BC Area and Power

As with the SRAM cell design, one of the most important criteria are the area and the power of the device. In order not to increase the area excessively, [30] proposes to add the capacitor above the bit cell, using higher metal layers. Apart from this, they have also decided to choose p-type transistors instead of n-type transistors to perform the XNOR operation. By choosing p-type transistors, the number of PMOS and NMOS transistors match and thus make the design denser. With the design proposed in [30], they have been able to design the M-BC only 80% larger than a standard SRAM cell.

3.3. CMOS Technology

3.3.1. Moore's Law

CMOS technology was invented in 1963 by Frank Wanlass and to this day it is the dominant technology in electronics, especially in integrated circuits. One of the most important benefits that CMOS technology shows is the ability to improve the performance with reduced power consumption by scaling the technology. Gordon Moore in 1965 published an article where he predicted the evolution of the electronics industry and it became one of the most popular "laws". It is known as Moore's law. The Moore's law stated that the number of transistors on an integrated circuit would double every 2 years (1 year at the beginning). So, to continue with this prediction, the transistor width, length, and oxide dimension were scaled down by 30% every two-three years. Through this scaling, not only a denser design is obtained, but per technological node, parasitic capacitances are reduced by 30%, energy and active power per transition are reduced by 65% and 50% respectively, and the gate delay is reduced by 43%, allowing a 30% increase in maximum clock frequency. Figure 52 shows how the CMOS technology has been reduced during the last years [39,40].

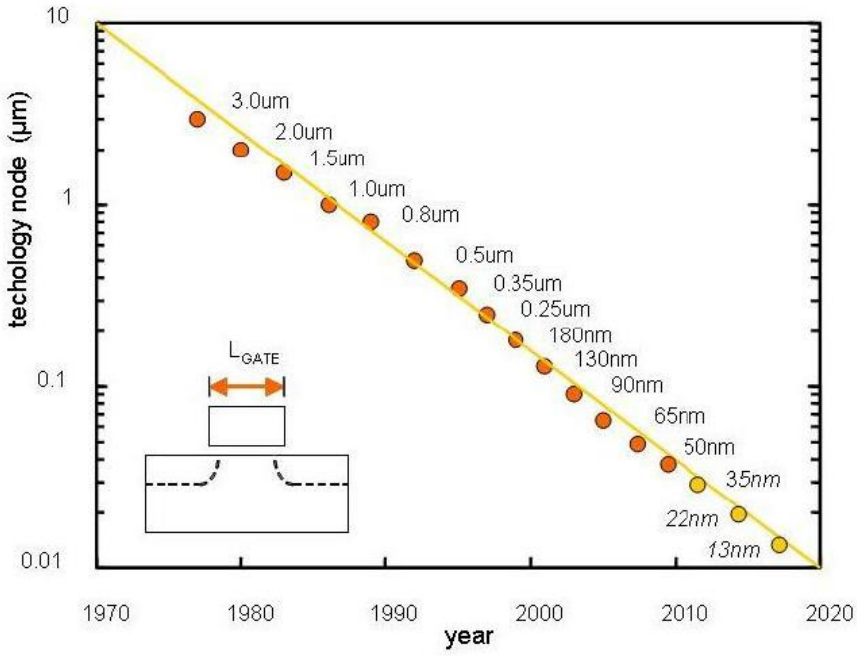


Figure 52 Transistors size reduction during the last years [51]

3.3.2. M-BC Technology Overview

The design of the M-BC explained above, was manufactured with 65nm CMOS technology, which is a technology that began to be used around 2003. In accordance with the Moore's law, in this project the M-BC circuit has been redesigned using a more recent CMOS technology to improve the characteristics of the original circuit. For that, a 22nm FDSOI technology has been used.

The technology used to design the M-BC circuit is 22FDX®, from GlobalFoundries. This technology employs Fully Depleted Silicon-On Insulator (FDSOI) devices which presents several advantages over common bulk CMOS devices. One of the main advantages is that

SOI technologies present much lower leakage than the bulk technologies which makes it ideal for IoT systems and also for M-BC as it will be explained later in the design section. To reduce the leakage, a Buried Oxide (BOX) layer is added below the diffusion regions creating a fully depleted channel. By placing this BOX layer, the electric flow between drain and source is confined in a smaller area that is very close to the gate, thus reducing the leakage. Also, by adding this BOX layer, the depth of the source and drain junction becomes smaller, which reduces the junction capacitances of the transistors. Figure 53 shows a cross section of the regular transistor and FDSOI transistor [41].

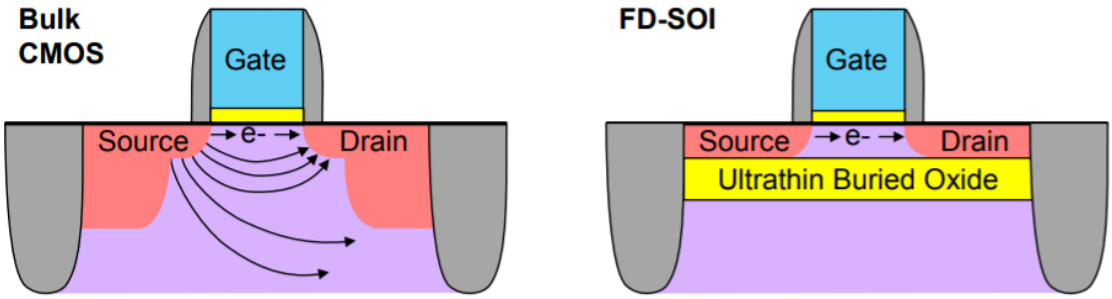


Figure 53 Cross section of the regular transistor and FDSOI transistor [42]

Another important advantage of SOI devices over bulk devices is that the body is electrically isolated from the drain and source, allowing the transistor threshold voltage to be controlled. In these devices, the body can be considered as a second gate, which depending on the voltage applied in the bulk, it can control the leakage and the switching frequency of the transistor. Figure 54 shows a cross section of the regular transistor and FDSOI transistor with bulk connection [42].

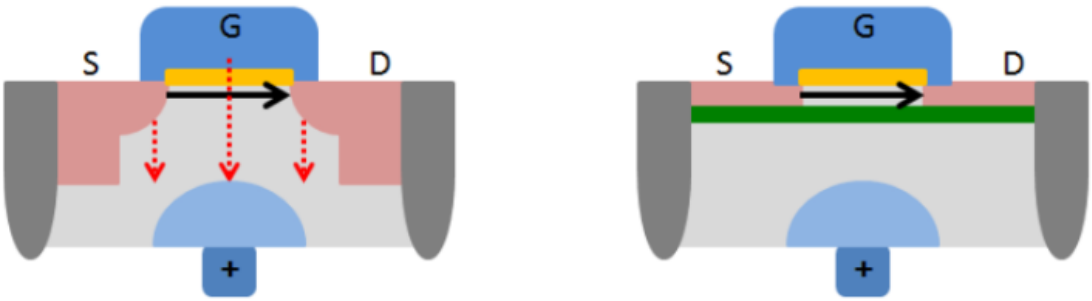


Figure 54 Body Biasing in regular transistor and FDSOI transistor [43]

4. M-BC Schematic Design

In this section the design of the CNN accelerator analyzed in the previous section will be explained. As mentioned, for this design a much smaller technology has been used. With this change, it is expected to improve the main characteristics of the design, such as area and energy efficiency. As technology has changed, certain general characteristics of the circuit have also been changed. Table 2 summarizes the general characteristics of the circuit.

Technology	22 nm FDSOI	65 nm CMOS
Clock frequency	100 MHz	100 MHz
Operating V_{DD}	0.8 V	0.94 V

Table 2 Summary of general design characteristics

At the end of the design, a comparison will be made between the original results obtained in the paper [30] and the results obtained in this project to see if these improvements have been achieved.

As previously mentioned, the M-BC has two main functions, to store the pre-trained weights and to perform the MAC operation. First, the SRAM has been designed, which will later be implemented in the M-BC.

4.1. SRAM Design

The SRAM is the responsible of storing the pre-trained weights (W, Wb) used for the filtering of the input activations. As mentioned before, the area, the stability and the power consumption have been considered for the memory design.

22FDX technology offers several types of transistors depending on the threshold voltage (V_{th}). All the transistors support a nominal voltage V_{DD} of $0.8\,V\ \pm 10\%$. During the SRAM design, two types of transistors have been used:

- Regular V_{th} (RVT: *nfet* and *pfet*): This type of transistors, as the name indicates, have a regular V_{th} compared with the rest of the transistor. Regarding the size, the minimum length of the transistor is 20nm and the minimum width is 80nm.
- Ultra-low Leakage High V_{th} (UHVT: *uhvtnfet* and *uhvtpfet*): This type of transistor, as the name suggests, have a higher V_{th} than the regular V_{th} transistors. Apart from this, this type of transistors offers lower leakage than the regular V_{th} transistors. Another difference between the two types is the minimum length of the transistor, which in this case is 28nm.

Using these transistors, two different SRAMs have been designed and analyzed in this project. However, there is another SRAM that has been analyzed with these designs. The technology offers three 6T-SRAM bit cell models where some specific transistors are used. 22FDX offers an Ultra-Low Leakage (ULL) SRAM, a High Density (HD) SRAM and a High Current (HC) SRAM. From these types of SRAMs, during the test carried out, two of them were discarded, because the behavior regarding the previously mentioned criteria was worse. Therefore, in the following sections three different SRAMs will be analyzed and

compared. Two of them designed in this project and the third one the ULL SRAM, offered by the technology. In order to be able to compare properly and to facilitate the design of the layout, it has been decided to use a length of 28nm for all transistors.

4.1.1. UHVT SRAM

The UHVT SRAM has been designed using UHVT transistors. As area and stability are one of the main criteria for the SRAM design, the width of the pull-up transistor of the SRAM has been set to the minimum value, which is 80nm. Once this value was set, the widths of the pull-down and pass-gate transistors were set to 110nm and 90nm respectively, taking into account the writability and read stability requirements. The width of the pull-down transistor was intended to be 150nm instead of 110nm, since the reading stability of the SRAM bit cell was better. However, due to some problems encountered during the layout design, the width of this transistor was changed. One of the main advantages of this SRAM is that the storage nodes remain constant because the transistors suffer low leakage. Figure 55 shows the schematic of UHVT SRAM.

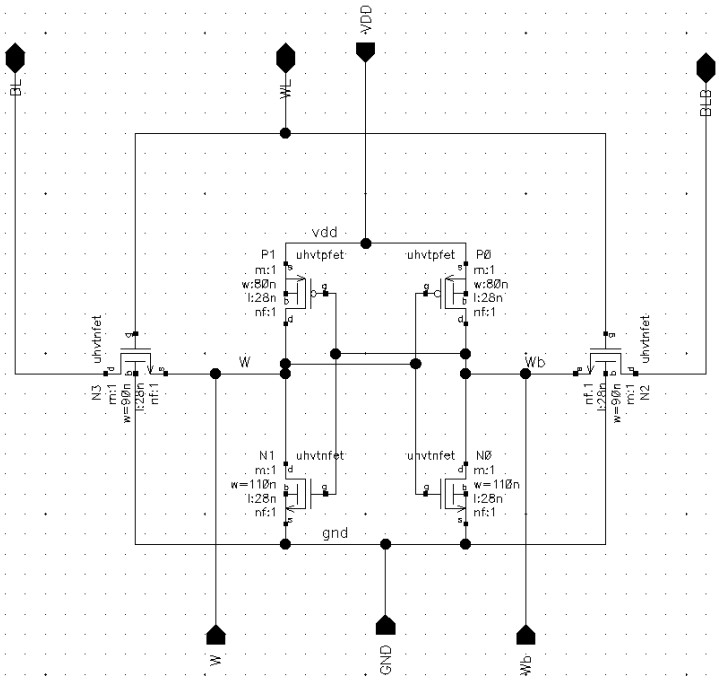


Figure 55 UHVT SRAM Schematic

4.1.2. UHVT and RVT SRAM

The UHVT and RVT SRAM has been designed using UHVT and RVT transistors. Specifically, the pass-gate transistors are UHVTs and the transistors that make up the bistable storage element are RVTs. This configuration offers several advantages. The UHVT pass-gate transistors will maintain the storage node constant and the RVT transistors will improve the read and hold stabilities of the SRAM bit cell. This happens because RVT transistors are stronger than UHVTs. So, the pull-down transistor does not let the current flowing through the access transistor raise the storage node. However, this configuration shows two important drawbacks over UHVT SRAM. On the one hand, since the RVT transistors are stronger, the power consumption is higher, especially when writing a value. On the other hand, the area of this SRAM is larger, since the necessary distance between two transistors of different type is greater than the distance between two

transistors of the same type. Regarding the size of the transistors, they are the same as those used in the UHVT SRAM. Figure 56 shows the schematic of UHVT and RVT SRAM.

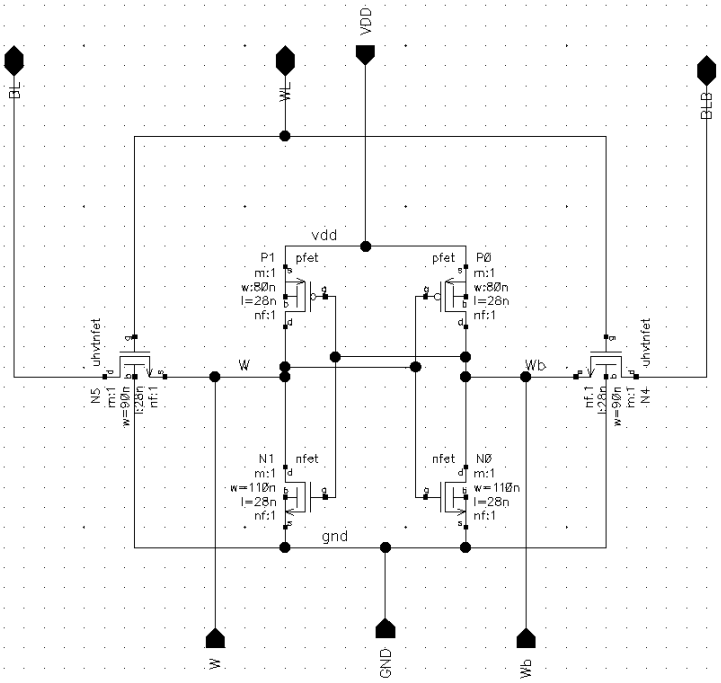


Figure 56 UHVT and RVT SRAM Schematic

4.1.3. ULL SRAM

The ULL SRAM is one of the designs that the technology offers. One of the main advantages of this SRAM is that the area is much smaller than the previous designs. This happens because the technology has some special rules for its SRAM that ignore or relax some design restrictions, such as the minimum width of the transistors. However, this is also a problem, because these rules only apply to SRAM and not to the additional transistors needed for the M-BC. So, the SRAM area is much smaller, but the M-BC area will be much larger than with other SRAM models. Figure 57 shows the schematic of ULL SRAM.

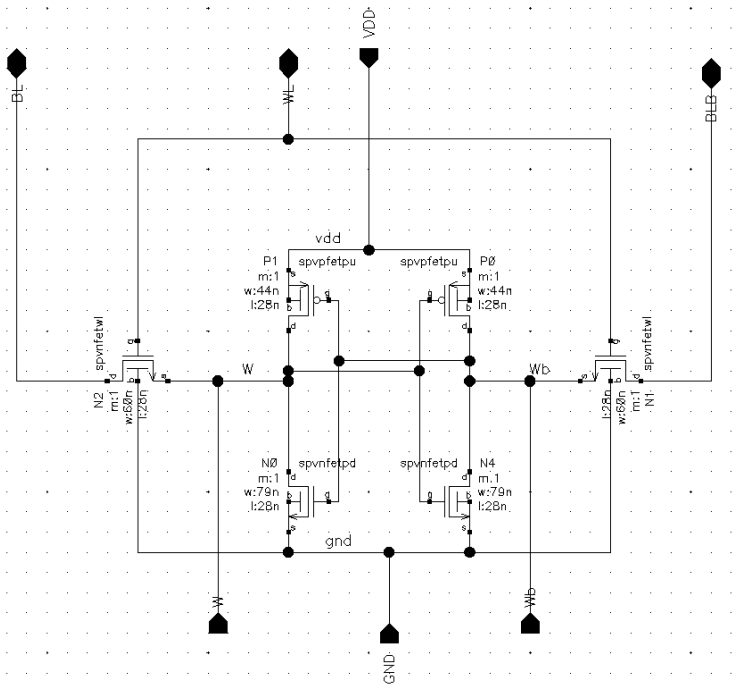


Figure 57 ULL SRAM Schematic

4.1.4. SRAM Testbench

To select the SRAM that will be used in the M-BC, the criteria mentioned in the previous section have been used. To measure all the necessary characteristics for the comparison, the testbench shown in Fig.58 has been used.

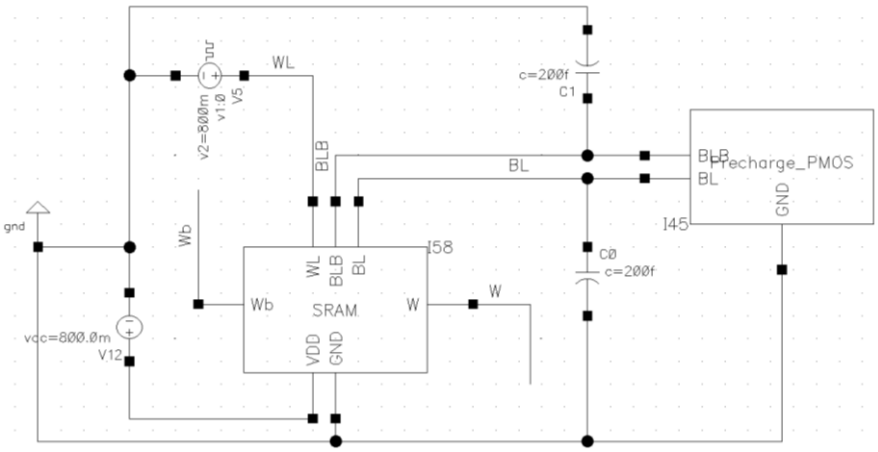


Figure 58 SRAM Testbench

As it can be seen in the testbench, the bitlines BL and BLB are connected to two capacitors that simulate the parasitic capacitors of the bitlines. The value of these capacitors is quite large because the bitlines are normally connected to several SRAM bit cells and their length is usually big. Apart from these capacitors, the bitlines are also connected to a circuit called "*Precharge_PMOS*". This circuit simulates the operation of the driver that sets the necessary values of the bitlines to perform read and write operation in an SRAM array. Figure 59 shows the schematic of the Precharge_PMOS circuit.

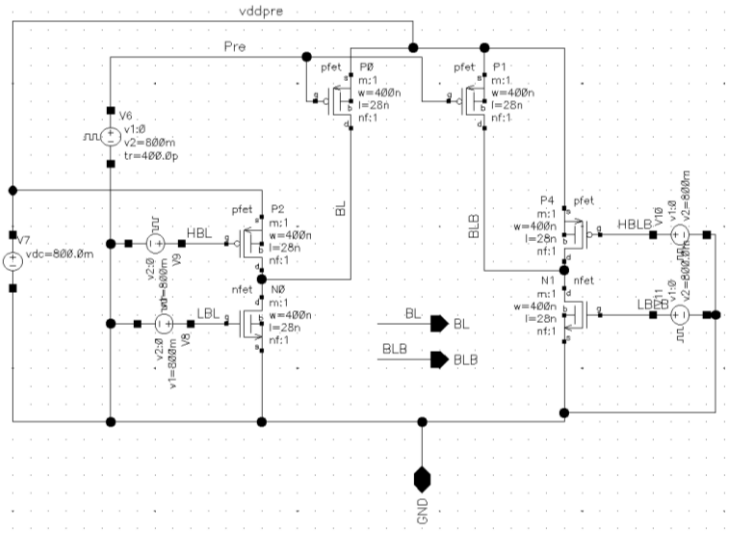


Figure 59 Precharge_PMOS Schematic

This circuit can be divided in two parts. The first part is formed by the transistors P0 and P1 and they are used to precharge the bitlines to start a reading operation. The second part is composed by the other transistors and they are used to set the desired value in the bitlines to perform write operations. Specifically, the transistors P2 and N0 sets BL to V_{DD} and GND respectively, and the transistors P3 and N1 sets BLB to V_{DD} and GND, respectively. During read operation, after the precharge of the bitlines, all transistors remain off. As it can be seen in the circuit above, the transistors are much bigger than the transistors used in the SRAM. This is because the transistors of the "driver" must be stronger than the SRAM transistors. Figure 60 shows the different waveforms of the signals used in the testbench.

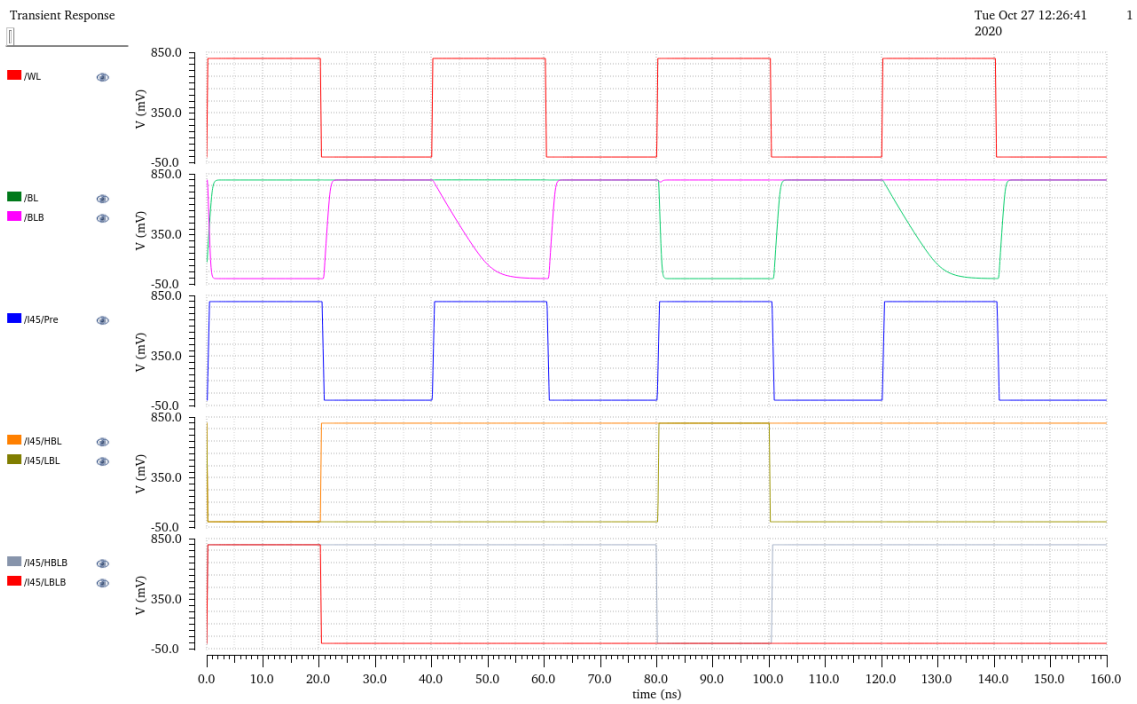


Figure 60 Signals used in SRAM Simulation

The simulation that can be seen in the figure above is divided in two parts. From 0 to 20ns, the bitlines BL and BLB are set to V_{DD} and GND respectively to write the value '1' in the memory. This is done with the signals *HBL* and *LBLB*. Then, from 20ns to 40ns, the reading operation begins by precharging the bitlines with the signal *Pre*. In the next 20 ns, it can be seen the discharge of bitline BLB indicating that there is a '1' stored in the SRAM. Then, from 80ns to 140ns the process is repeated but writing a '0'.

4.1.5. Comparison

Using the previous testbench, various features have been obtained from the SRAMs, such as the RSNM, the HSNM and the energy per read and write operation. In addition, the area of each SRAM bit cell has been estimated during layout process, since, as mentioned before, it is a very important characteristic. Table 3 shows the most important features for each type of SRAM.

SRAM Type	HSNM (mV)	RSNM (mV)	Energy per Read operation (aJ)	Energy per Write operation (aJ)	Estimated Area (μm^2)
UHVT	420	180	22.7	246.35	0.28
UHVT and RVT	390	290	32.4	295.7	0.37
ULL	415	210	22.1	284.9	0.11

Table 3 SRAM type comparison

As it can be seen in the table, each SRAM has its own advantages. The ULL SRAM shows very good values in all the parameters that have been calculated, especially in the area. However, as this SRAM is a cell given by the technology, during the layout process, it was realized that it was not feasible to design the M-BC with this SRAM. As for the other two designs, the UHVT SRAM presents better characteristics than the UHVT and RVT SRAM except the RSNM. Although this value is lower, it is an acceptable RSNM. So, after analyzing the three SRAM models, the UHVT SRAM has been chosen for the M-BC design.

4.1.6. Results

In this section, the results of the UHVT SRAM simulations are shown. For that, the testbench explained above is used. Figure 61 shows the waveforms of the storage nodes of the SRAM during the simulation.

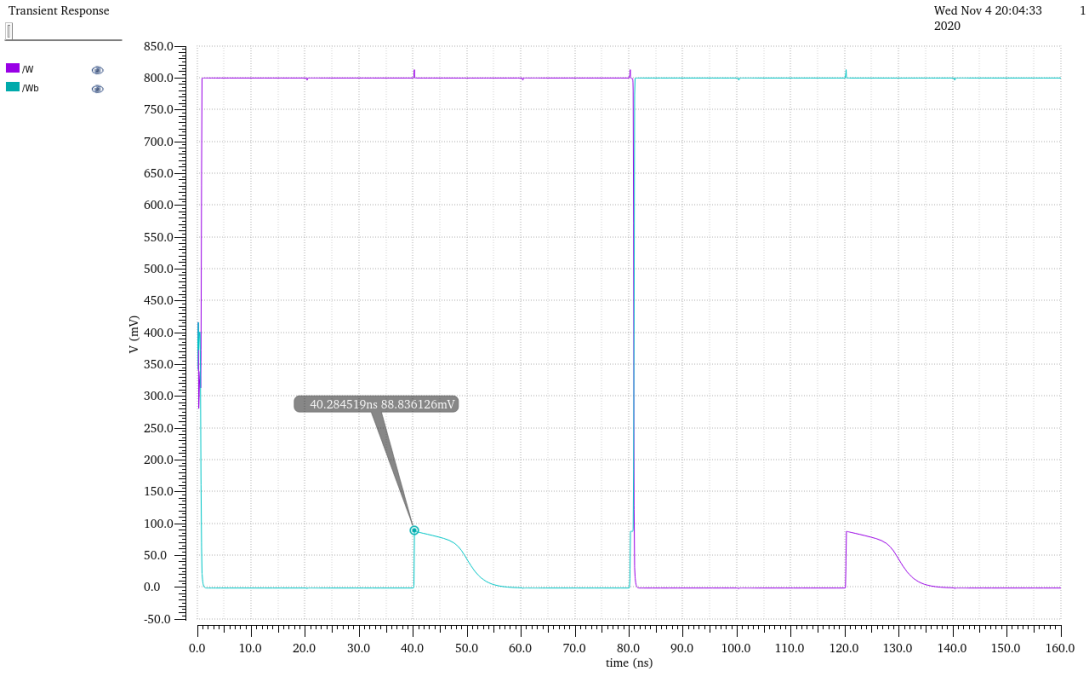


Figure 61 Storage nodes simulation in SRAM

As it can be seen above, the performance of the SRAM is as expected. During the read operation, it can be seen that as explained in section 3.1.3, the storage node with the low level tries to increase its value due to the current going through the pass transistor. However, this value is not critical as it is very far from the value necessary to change the memory state. Specifically, the peak reaches a maximum value of 88.8 mV, which is much lower than the V_{th} of the transistors used. In addition, it can be also seen that at 80ns, the SRAM state changes correctly, confirming that the SRAM meets the writability requirement. Figure 62 shows a zoom where it can be seen that the change of state in the SRAM lasts approximately 1 ns.

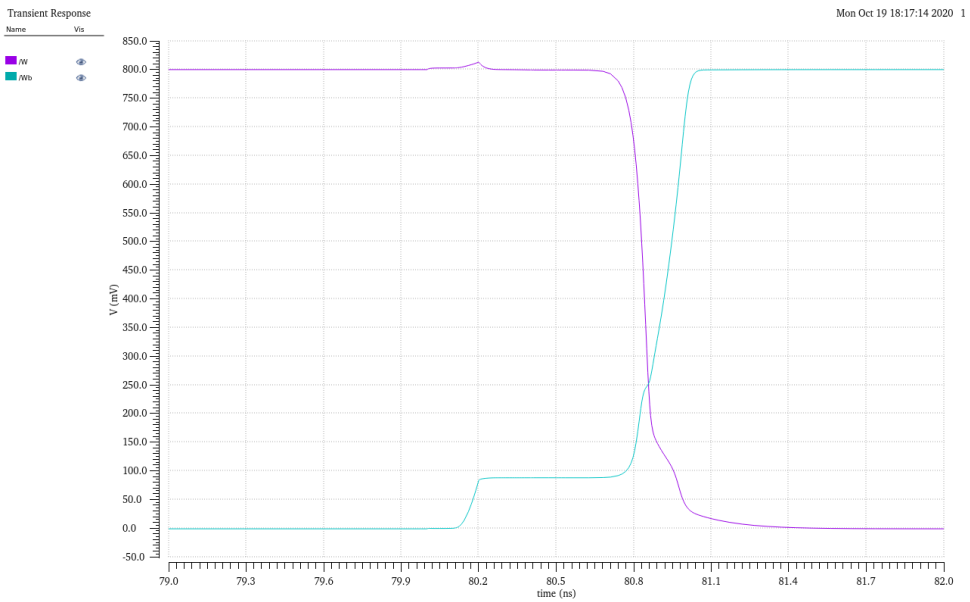


Figure 62 Zoomed view of state change in SRAM

In the comparison of the SRAM types, the HSNM and RSNM of each model have been taken into account. To obtain these values, as explained in section 3.1.5, the VTCs of the two inverters have been obtained. Figure 63 and Fig. 64 show the butterfly curves were the HSNM and RSNM have been obtained, respectively.

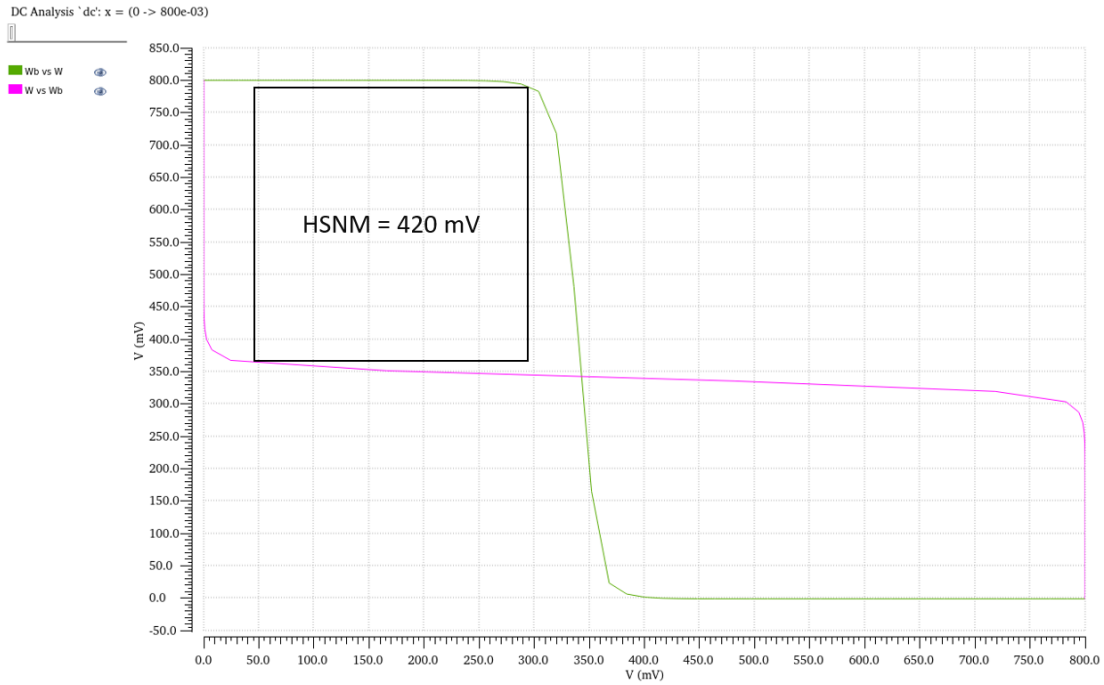


Figure 63 SRAM HSNM

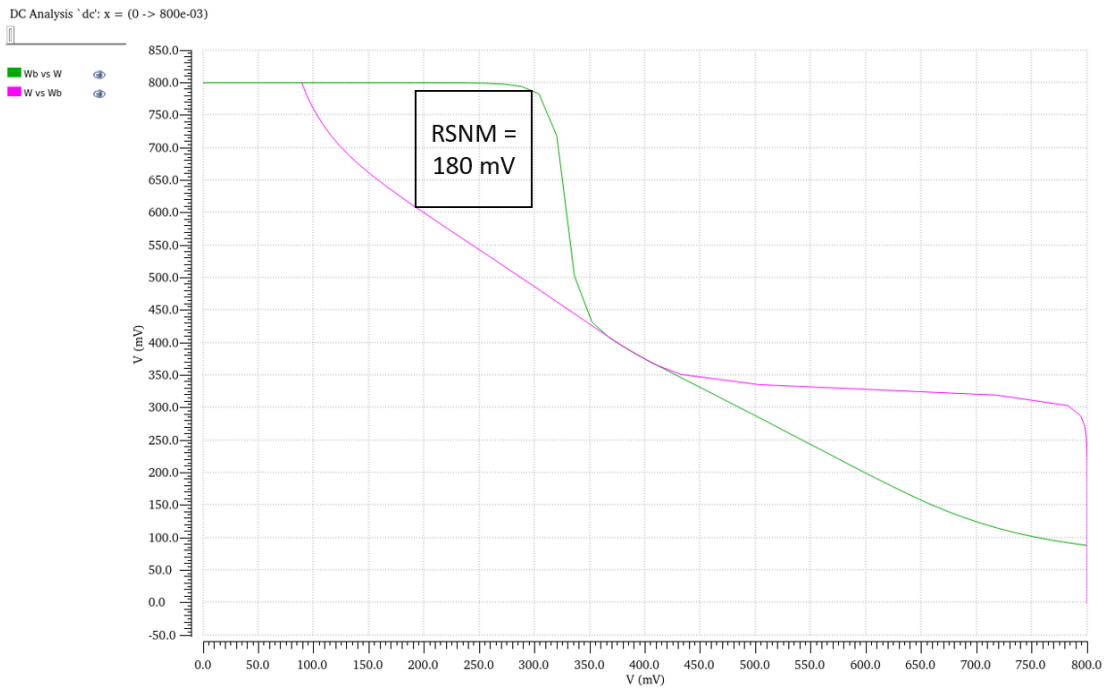


Figure 64 SRAM RSNM

4.2. M-BC Design

4.2.1. Single M-BC Design

After selecting the SRAM, the M-BC can be designed. One of the most important parts of the circuit is the charge stored in the MOM capacitor, which is the result of the XNOR operation. Therefore, it is important that this charge remains constant and that there are no losses during the binary-multiply phase and accumulate phase. If this does not happen, the PA obtained in the accumulation phase may be a wrong value, especially when there are many M-BCs connected. To ensure that the charge remains constant, UHVT type transistors have been used. The use of this type of transistors has another important advantage related to the area. As in SRAM, if different types of transistors were used in the design, the size of the M-BC would be much larger than if the same type of transistors were used. In addition, to reduce leakage and area even more, the PMOS transistors used to perform the XNOR operation and the transistors used in the transmission gate have a width of 80nm, which is the minimum width. The length of those transistors, as in SRAM, is 28nm.

When designing the M-BC, the MOM capacitor is also very important. In section 3.2.2 it was said that the PA voltage of the M-BC does not depend on the MOM capacitor. Specifically, the PA voltage is represented in (11).

$$V_{PA} = \frac{V_{DD} \cdot C_{MOM} \cdot \sum_N out_i}{N \cdot C_{MOM}} = \frac{V_{DD} \cdot \sum_N out_i}{N} \quad (11)$$

However, this only happens in an ideal case. In fact, the output voltage range of the M-BC is highly dependent on the value of the MOM. This happens because, as with bitlines, the parasitic at the output are usually important, mainly due to its length. Equation (12) shows the output voltage considering the parasitic capacitors.

$$V_{PA} = \frac{V_{DD} \cdot C_{MOM} \cdot \sum_N out_i}{N \cdot C_{MOM} + C_{par}} \quad (12)$$

These parasitic capacitors cause the reduction of the dynamic range of the PA, thus reducing the output voltage range of the M-BC. To mitigate this problem, the capacity of the MOM should be made as large as possible. However, increasing the capacity of the MOM means increasing the size of the MOM, that is, increasing the area of the M-BC. Apart from this, by increasing the MOM, the stability of the M-BC worsens. This is because the storage nodes are going to be exposed to a stronger pull-down condition. So, to choose a value for the MOM capacitor, it is necessary to take these conditions into account. The MOM capacitor value has been modified during the design phases, especially in the layout phase. Finally, the value of the MOM capacitor has been set to 2.84 fF, which is 136% bigger than the capacitor selected in the reference paper. Figure 65 shows the schematic of a M-BC.

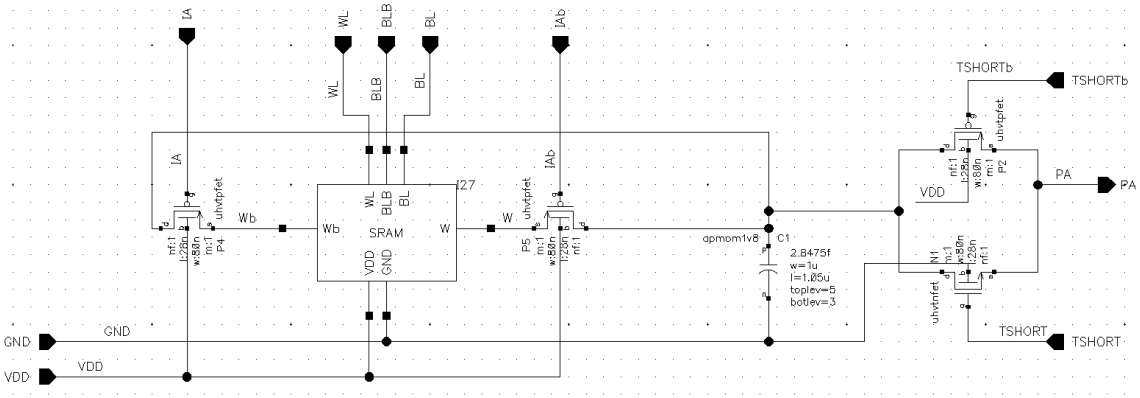


Figure 65 Single M-BC schematic

4.2.1.1. Single M-BC Testbench

To simulate the circuit, the testbench shown in Fig.66 has been used.

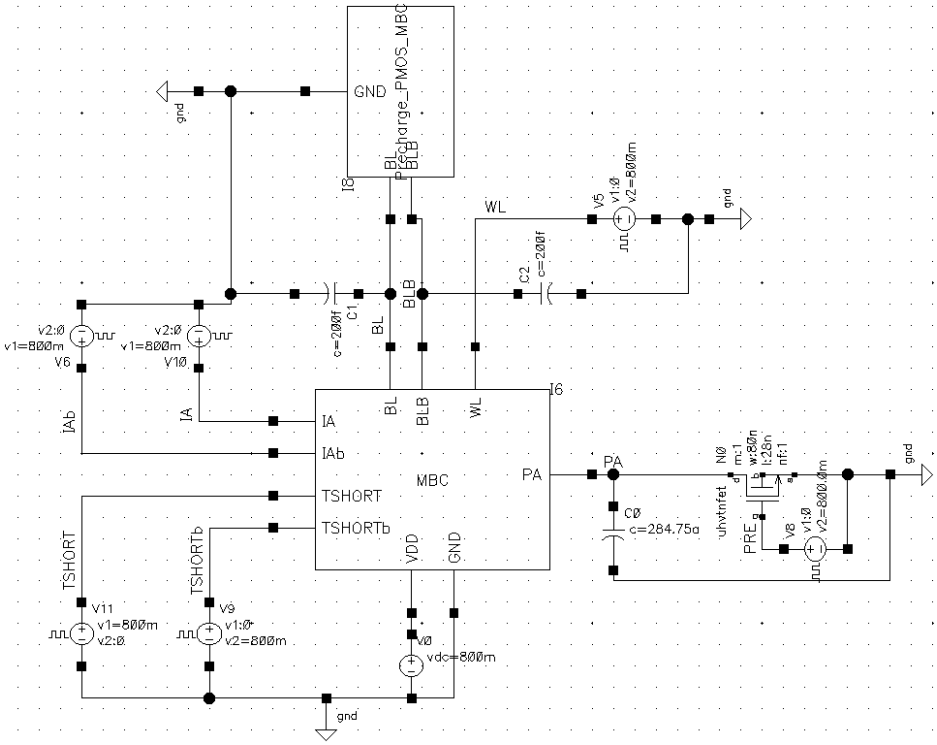


Figure 66 Single M-BC Testbench

As it can be seen in the testbench, the bitlines BL and BLB, as in the SRAM testbench, are connected to two capacitors and to a circuit that simulates a driver. However, the operation of this circuit is now different because to simulate the M-BC, it is only necessary to write a value in the SRAM. Therefore, this circuit is only used to write values in the SRAM. In this case, a '1' is stored in the memory.

It can be also seen that the M-BC output is connected to the discharge transistor and to a capacitor. This capacitor, as the capacitors connected to the bitlines, simulates the parasitic capacitors of the output due to its length and routing. In this case, it has been decided that the value of the parasitic capacitor is 10% the value of the MOM. Surely, due to the technology that is being used, this percentage could be lower, but it has been decided to keep this value so that the simulation is not too optimistic.

4.2.1.2. Simulation

Figure 67 shows the waveforms of the external signals used to simulate the M-BC.

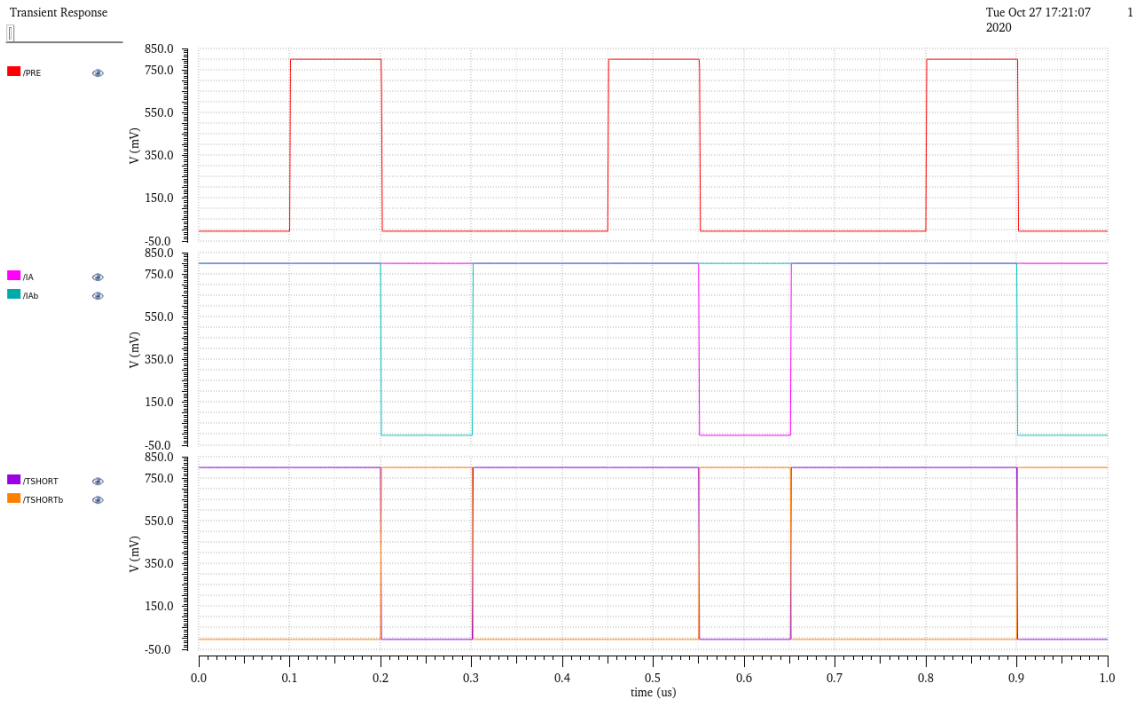


Figure 67 External signals used in M-BC Simulation

As mentioned before, the first 100 ns of the simulation are used to write the value '1' in the SRAM. Then, the MAC operation begins. As explained in section 3.2.2, in the first 100 ns, the discharge transistor is activated by PRE signal, discharging the MOM capacitor. Then, in the next 100 ns, the binary-multiplication is performed. For that, the transmission gate controlled by TSHORT/TSHORTb is deactivated and a PMOS is activated, in this case with the signal IAb. Therefore, in this case the output is represented in (13).

$$Out = W \cdot IA =' 1' \cdot' 1' = '1' \quad (13)$$

Finally, in the next 50 ns, the transmission gate is activated again, thus connecting the MOM capacitor with PA. This sequence is repeated but inverting the values of IA/IAb and obtaining a different output. This operation is represented in (14)

$$Out = W \cdot IA =' 1' \cdot' 0' = '0' \quad (14)$$

Between the two operations, there is a 100 ns wait. This is done to simplify the simulation and to separate the two different operation.

Figure 68 shows the voltage at the MOM capacitor and at the PA output obtained in the simulation.

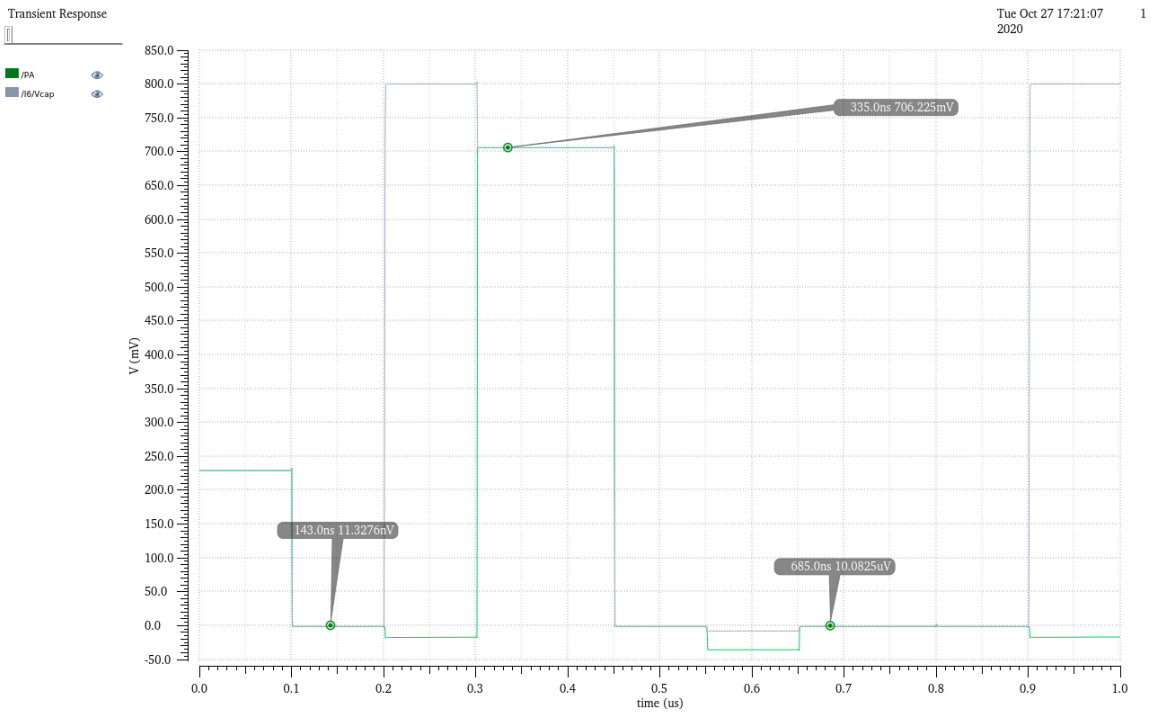


Figure 68 Single M-BC PA and Vcap pre-layout simulation

In the image above, it can be seen several things that have been already discussed. First of all, it can be seen that the MOM is discharged successfully. It can be also seen that when the result of the XNOR operation is '1', the PA is not able to reach V_{DD} due to the parasitic capacitors. In this case, the drop is a little higher than expected, because the discharge transistor is slightly oversized. Otherwise, the M-BC works as expected.

However, to ensure that the M-BC works properly, it is necessary to check the stability of the circuit. Figure 69 shows the storage nodes of the SRAM during the simulation.

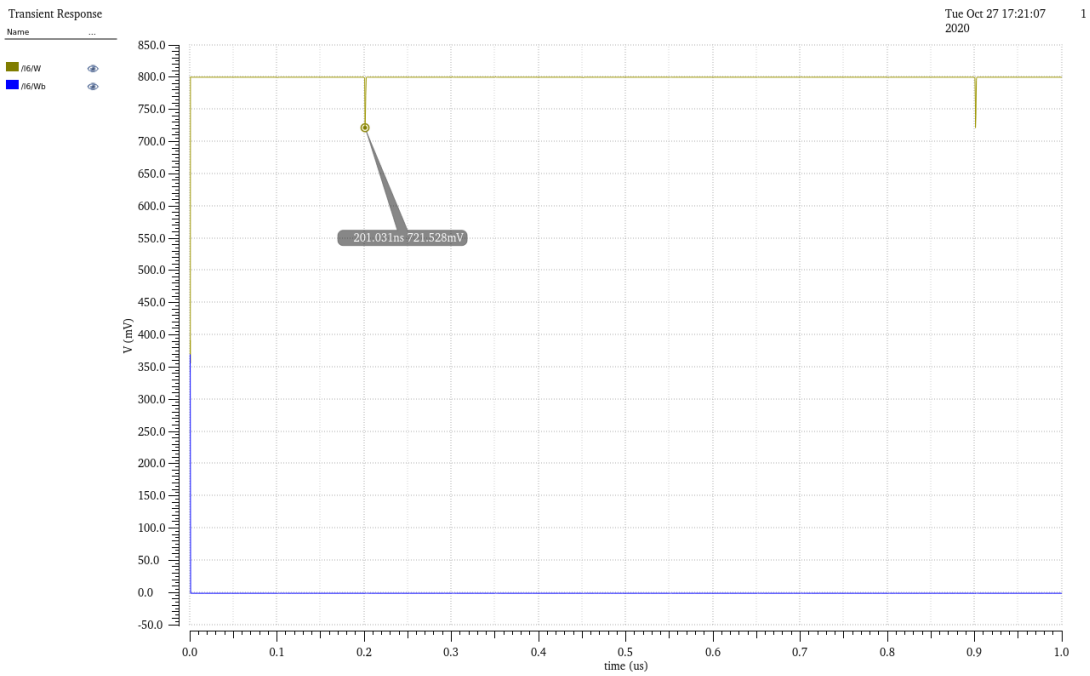


Figure 69 Storage nodes pre-layout simulation in M-BC

As it can be seen above, the storage node that stores the high value tends to decrease at the beginning of the binary multiplication phase due to the pull-down condition of the MOM. However, as it can be seen, this value only drops approximately 10% of its value and it recovers very quickly, without affecting the stability of the circuit.

4.2.1.3. Results

In this section the most important characteristics obtained in the simulation of a single M-BC are summarized. Table 4 shows these characteristics.

Pre-Layout Simulation	Maximum PA voltage (mV)	Minimum PA voltage (µV)	Maximum energy per operation (fJ)	Minimum energy per operation (fJ)
Single M-BC	706.22	10.08	2.07	0.05

Table 4 Single M-BC results in pre-layout simulation

The maximum PA voltage and maximum energy per operation correspond to the case in which the XNOR operation is '1' and the minimum values to the case in which it is '0'. This is consistent, since when the result of the operation is '1', the MOM capacitor has to be charged, whereas when the result is '0', it is already discharged.

4.2.2. Neuron Patch

Once the M-BC is designed, a 3x3 M-BC column can be designed, also called Neuron Patch. The circuit does not have any significant change since no new components have to be added. Figure 70 shows the schematic of a Neuron Patch.

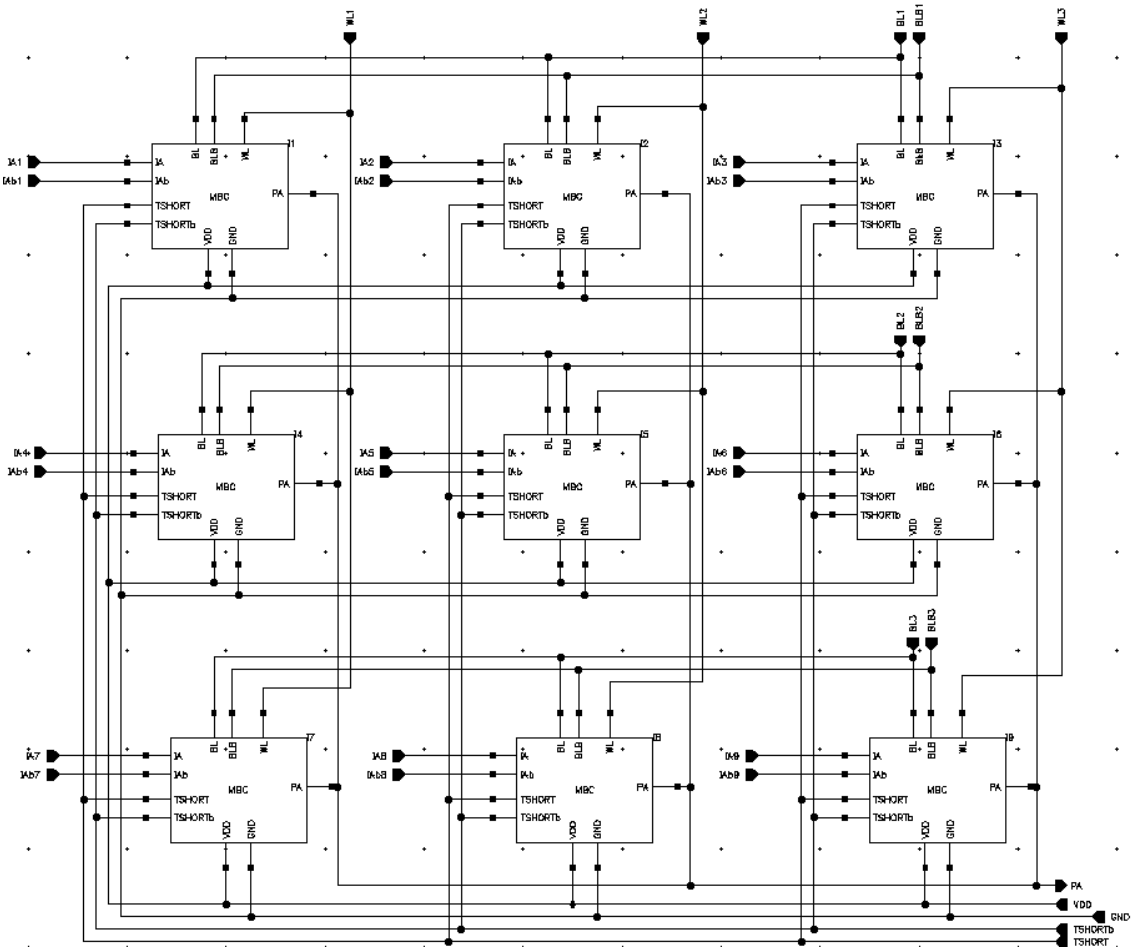


Figure 70 Neuron Patch Schematic

As it can be seen in the schematic, as in an SRAM array, each column inside the neuron patch has its own wordline and each row its pair of bitlines. Apart from this, the transmission gates of all the M-BCs in the neuron patch are controlled by the same TSHORT/TSHORTb signal. As it can be expected, the PA signals of each M-BC are connected to perform the accumulation operation and each M-BC in the neuron patch has its own input activation.

4.2.2.1. Neuron Patch Testbench

The testbench used to simulate the neuron patch is almost the same than the one used for a single M-BC. For example, if the total capacity at the output increases, the parasitic capacitor at the output increases by the same proportion. Figure 71 shows the testbench used to simulate the neuron patch.

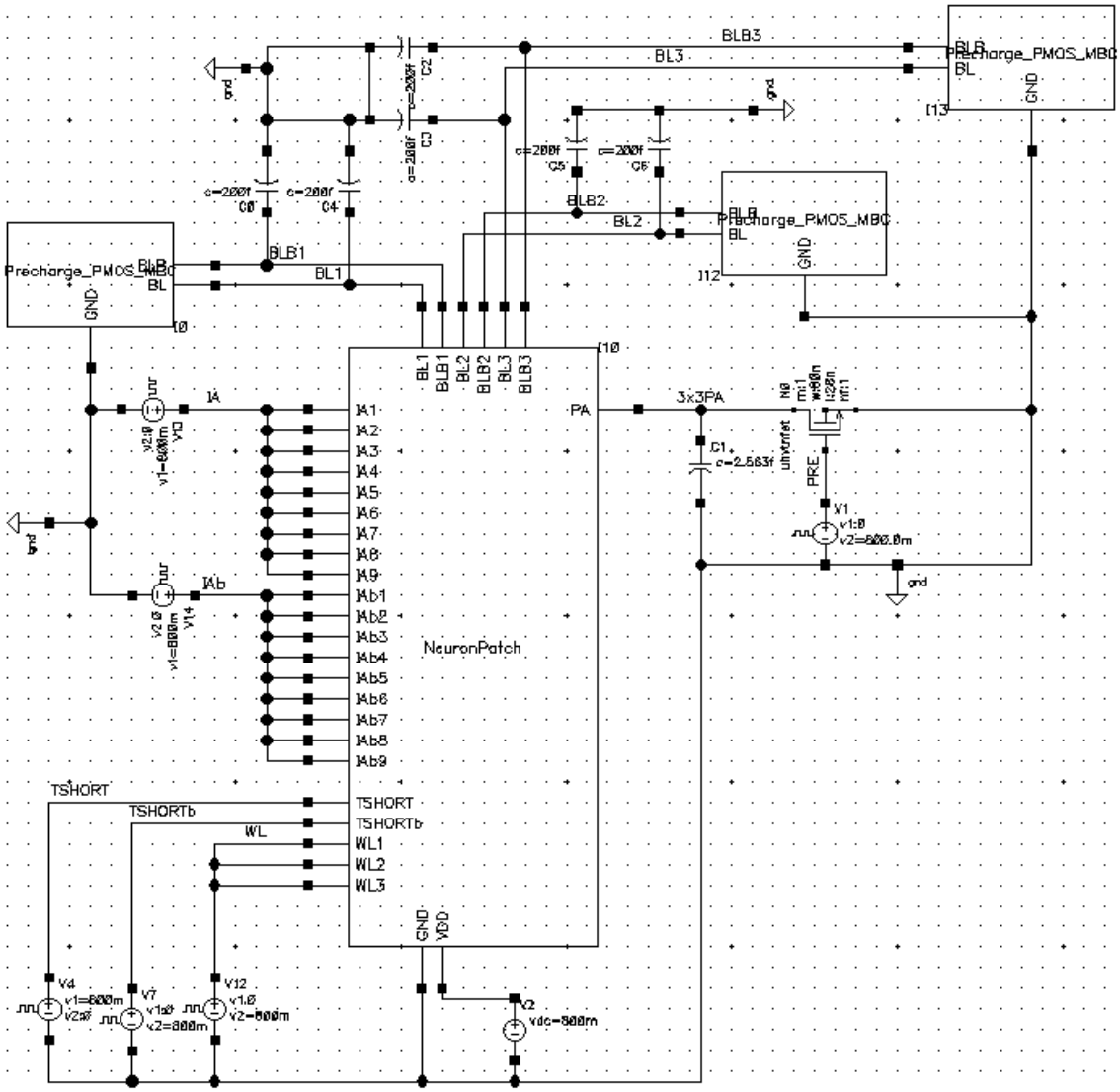


Figure 71 Neuron Patch Testbench

4.2.2.2. Simulation

The external signals (IA/IAb, PRE, TSHORT/TSHORTb) are the same as those used in the simulation of a single M-BC. These signals are represented in Fig. 67. Therefore, considering that all the weights stored in the M-BCs of the neuron patch are '1', the output obtained is shown in Fig. 72.

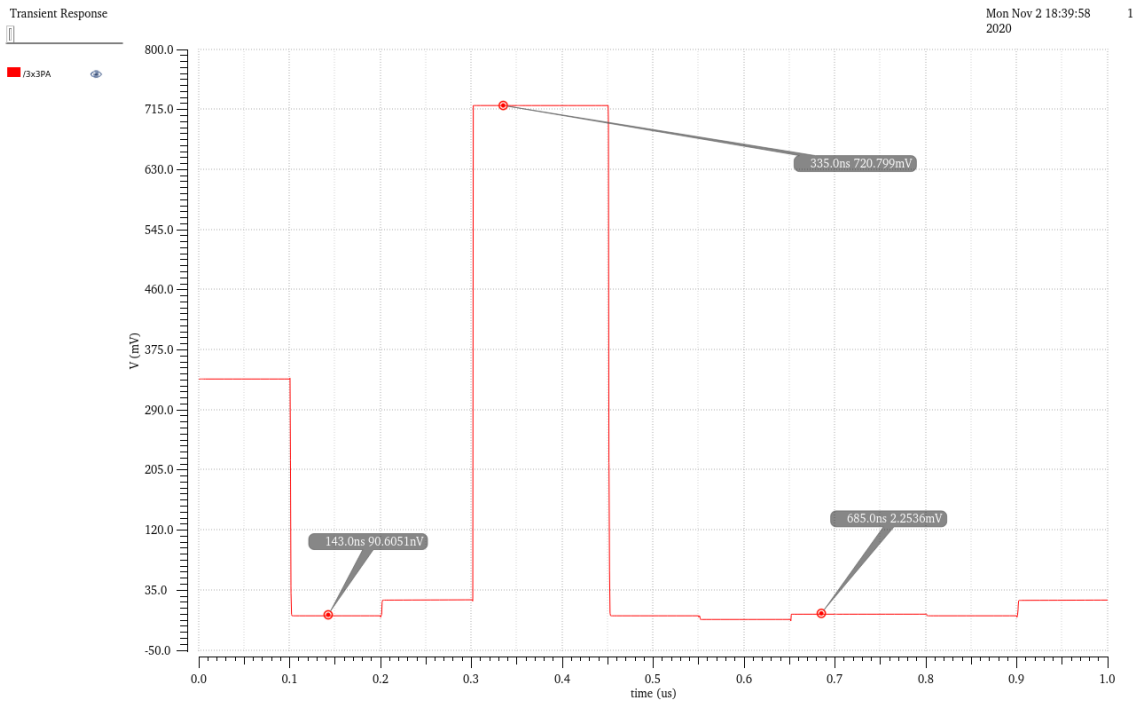


Figure 72 Pre-layout Neuron Patch PA simulation with all XNOR operation '1' and '0'

As it can be seen, when the result of the XNOR operation in all the M-BCs is '1', the output still does not reach V_{DD}, but the value obtained is better than when only one M-BC was simulated. However, the value obtained when the result of the XNOR operation in all the M-BCs is '0', is worse.

Figure 73 shows the resulting PA values obtained, when the result of the XNOR operation in the MBCs is not '1' or '0' for all of them. The specific cases that are shown in Fig. 73, are the following two:

- The first example simulates 5 XNOR operations resulting in '1' and the rest XNOR operations resulting in '0'.
- The second example simulates 4 XNOR operations resulting in '1' and the rest XNOR operations resulting in '0'

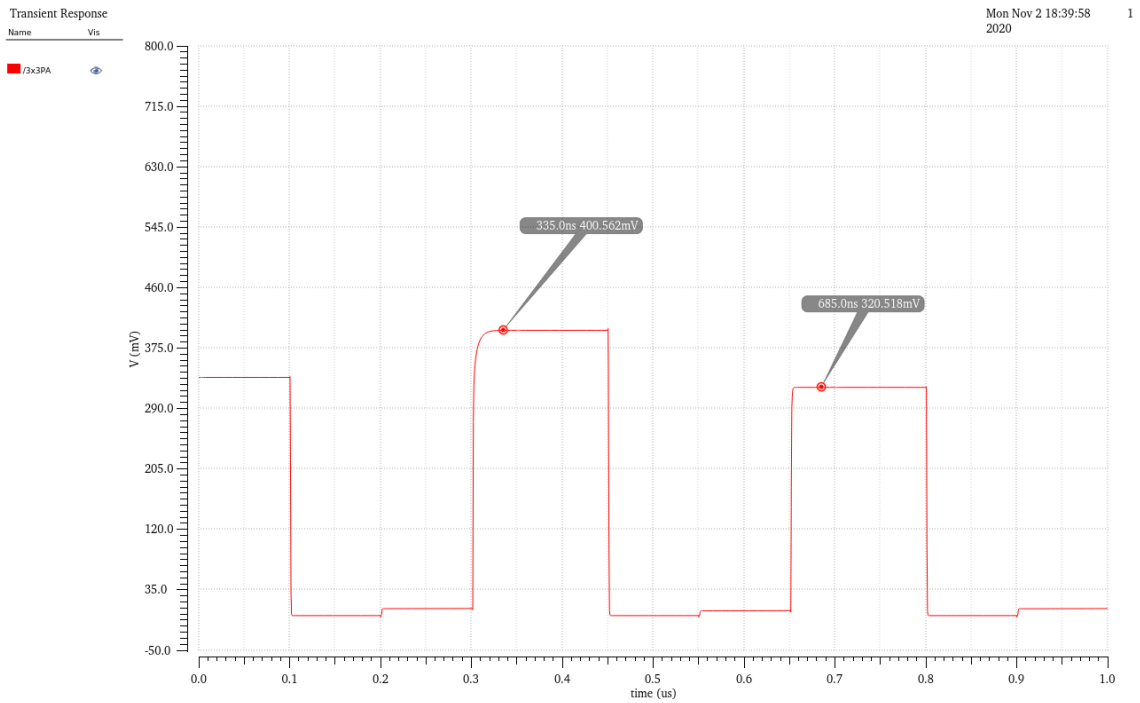


Figure 73 Pre-layout Neuron Patch PA simulation with 5 and 4 XNOR operations resulting '1'

4.2.2.3. Results and Comparison

In this section, the most important characteristics obtained in the simulation of a neuron patch are summarized. Table 5 shows these characteristics.

Pre-Layout Simulation	Maximum PA voltage (mV)	Minimum PA voltage (µV)	Maximum energy per operation (fJ)	Minimum energy per operation (fJ)
Single M-BC	706.22	10.08	2.07	0.05
Neuron Patch	720.8	2256	23.34	0.67

Table 5 Neuron Patch results in pre-layout simulation

As mentioned before, the maximum and the minimum PA voltages are higher than for a single M-BC. So, as expected, the dynamic voltage range of PA increases by adding more M-BC, since the effect of parasitic capacitors becomes smaller. Regarding energy, the increase in energy is consistent with the increase in M-BCs.

4.2.3. Neuron Filter

A neuron filter in the neuron array is composed of 4608 M-BCs. So, it is interesting to evaluate the PA that will enter in the binarizing batch normalization circuit to obtain the binary output activation. In order to simulate this circuit, it is necessary to make an important change. In previous simulations, the discharge transistor has always a width of 80nm. However, as the number of M-BCs increases, the width of this transistor must also increase, since otherwise it would not be able to discharge all the MOM capacitors. For the neuron filter, a discharge transistor with width of 1µm has been used. Figure 74 shows the output of a neuron filter.

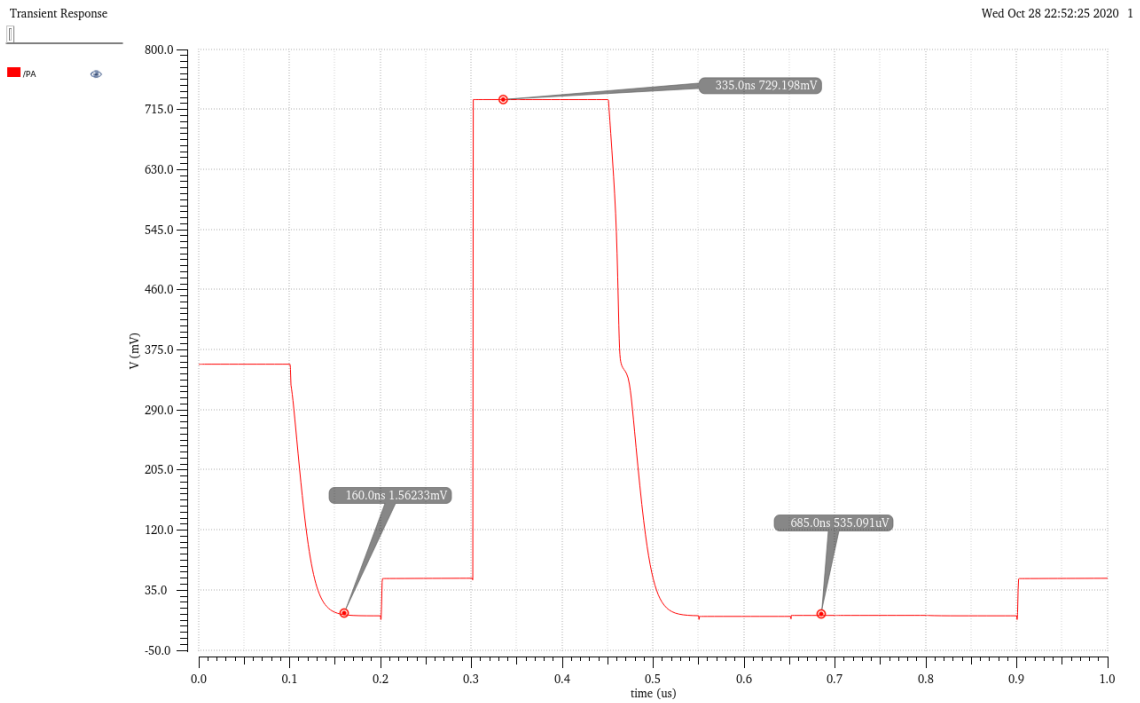


Figure 74 Neuron Filter PA pre-layout simulation with all XNOR operation '1' and '0'

As mentioned above, the PA signal is discharged much slower than in previous simulations, since the total capacitance in the output is much bigger. Despite this, increasing the width of the transistor, the MOM capacitors are discharged properly. Apart from this, it can be seen that the voltage range of the PA is slightly higher than in previous simulations.

4.2.3.1. Results and Comparison

In this section, the most important characteristics obtained in the simulation of a neuron filter are summarized. Table 6 shows these characteristics.

Pre-Layout Simulation	Maximum PA voltage (mV)	Minimum PA voltage (µV)	Maximum energy per operation (fJ)	Minimum energy per operation (fJ)
Single M-BC	706.22	10.08	2.07	0.05
Neuron Patch	720.8	2256	23.34	0.67
Neuron Filter	729.2	535.1	12150	349.8

Table 6 Neuron Patch results in pre-layout simulation

As mentioned before, the voltage range of PA is higher than in previous cases, but it still does not reach the full range. Regarding consumption, it can be seen that the filtering operation consumes a maximum of 12.15 pJ and a minimum of 0.35 pJ.

5. M-BC Layout and Post-Layout Results

5.1. M-BC Layout Design

After finishing the design of the circuit and verifying that it works correctly, the layout design has been carried out. The first idea was to use the first three metal layers to do the routing and use the next two metal layers to place the MOM. However, as explained in previous sections, by increasing the capacity of the MOM, the effects produced by parasitic capacitors at the output are also reduced. That is why it was decided to increase the capacity of the MOM using three metal layers instead of two. In this way, the capacity of the MOM is increased, without increasing its size.

By reducing the metal layers for routing, the area of the design increases. However, this increase is not so critical, since a technique called double patterning is applied in these first layers. This technique consists of dividing a layer into two different masks. The main advantage of having two masks is that the design rules between two different masks on the same layer are much more flexible than the design rules between the same mask on the same layer. These masks are differentiated by the subscript "*_E1*" and "*_E2*".

Table 7 shows the final configuration used for the circuit design.

	Metal Layer 1 (M1_E1, M1_E2)	Metal Layer 2 (M2_E1, M2_E2)	Metal Layer 3 (C1)	Metal Layer 4 (C2)	Metal Layer 5 (C3)
Routing	**Yes**	**Yes**	**No**	**No**	**No**
MOM capacitor	**No**	**No**	**Yes**	**Yes**	**Yes**

Table 7 Metal Layer configuration

5.1.1. Single M-BC

The design of the M-BC layout is largely determined by the size of the MOM capacitor. The MOM used in the design has a width of 1 µm and a length of 1.05 µm, which equates to a capacity of 2.84 fF. However, the dimensions of the MOM, specifically the length, is slightly bigger due to the thickness of the pins. Both the length and the width of the MOM are quite large compared to the rest of the components in the circuit, that is why they greatly affect the final size of M-BC.

Figure 75 shows the layout of a single M-BC.

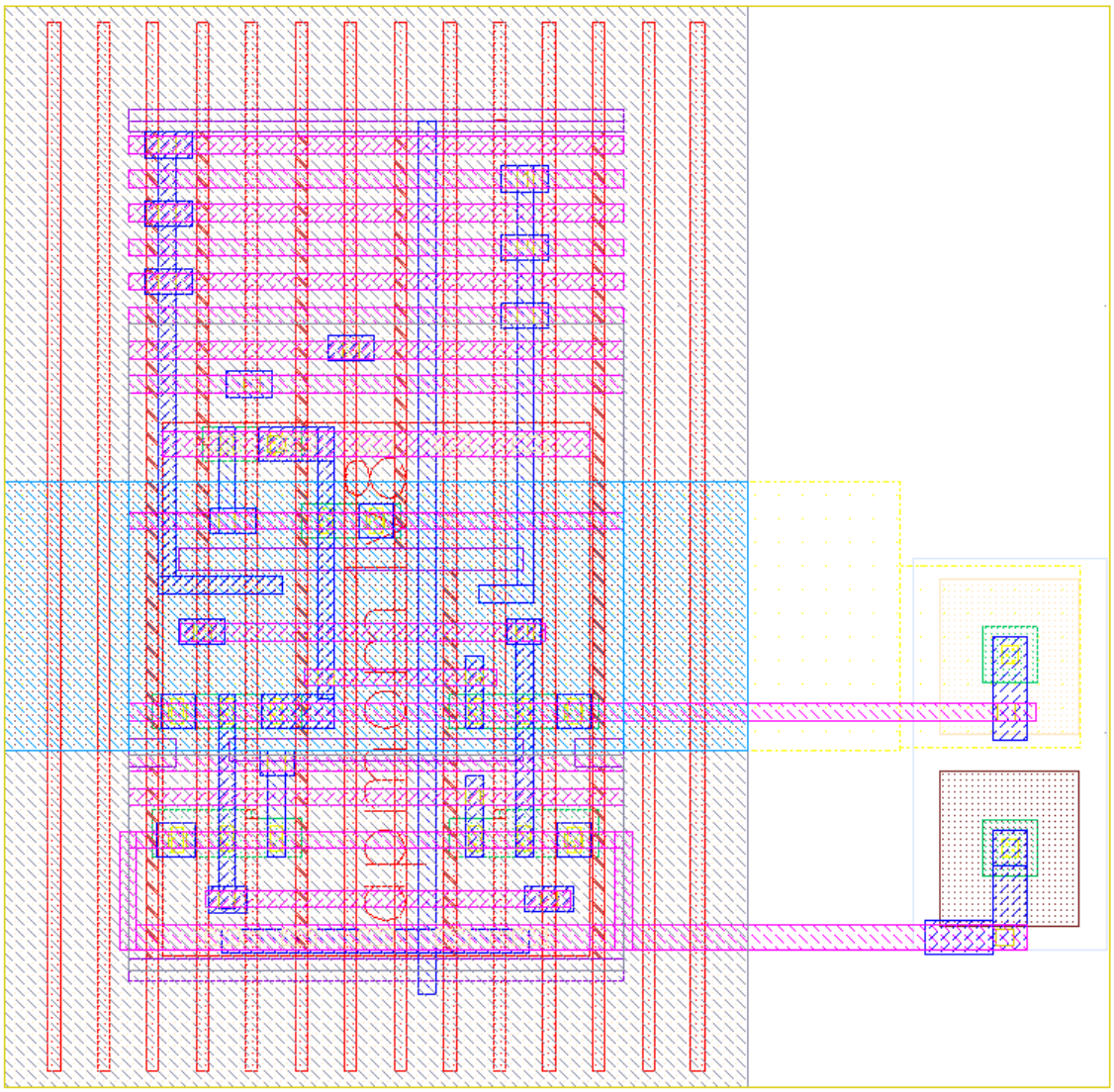


Figure 75 Single M-BC Layout

The red vertical lines are the polysilicon gate contacts. As it can be seen, there are several extra polysilicon on both sides of the design. This is because to perform the QRC (Cadence parasitic extraction tool) correctly, it is necessary to add some dummies on the sides. Apart from this, at the right side of the figure, it can be seen the body contacts for both types of transistors. This design is only used to simulate a single M-BC, since to extract the parasitic components of the circuit the QRC and layout-versus-schematic (LVS) must not have any error. However, the design shown above is not useful when several M-BCs have to be connected together. For that, it is necessary to create a standard design. Figure 76 shows the layout of a standard M-BC.

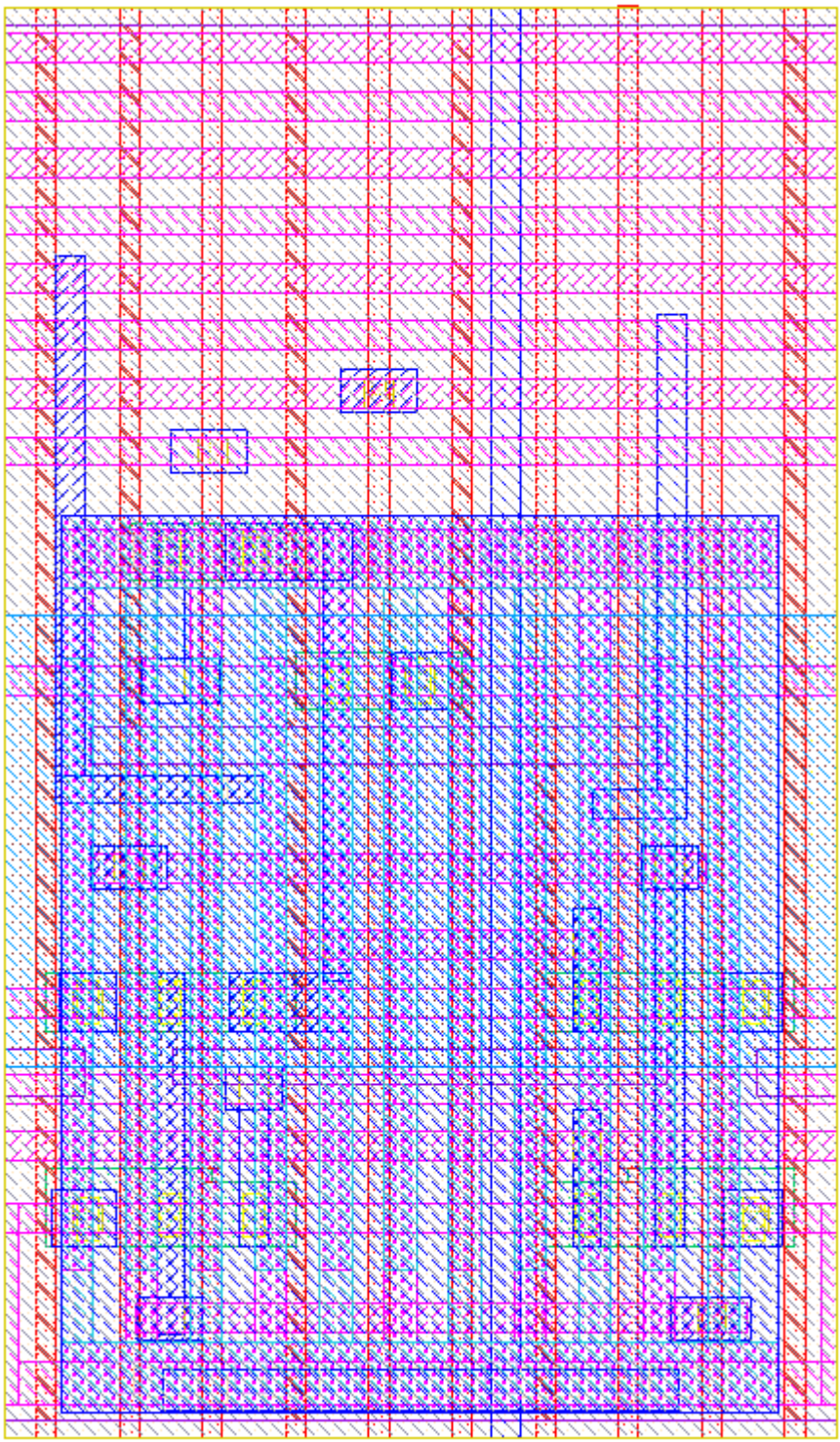


Figure 76 Single M-BC Standard Cell Layout

As it can be seen above, the design could be divided in two parts: the upper part where there are 6 horizontal tracks in parallel without any connection, and the rest of the circuit. The first part is used to connect each IA/IAb signal with each M-BC, and it can be considered as an external part of the M-BC. In other words, the standard cell could have been smaller by eliminating these 6 tracks, but later, when designing the Neuron Patch, it would be necessary to leave an equivalent space between M-BC rows to be able to make these connections. Therefore, introducing these tracks in the standard cell, the design of

the neuron patch is simplified. The second part of the layout is where the M-BC is located. However, in the image above it can be only seen the MOM capacitor since the routing and the transistors are located below it. Figure 77 shows the M-BC layout where some of the layers and the capacitor have been hidden.

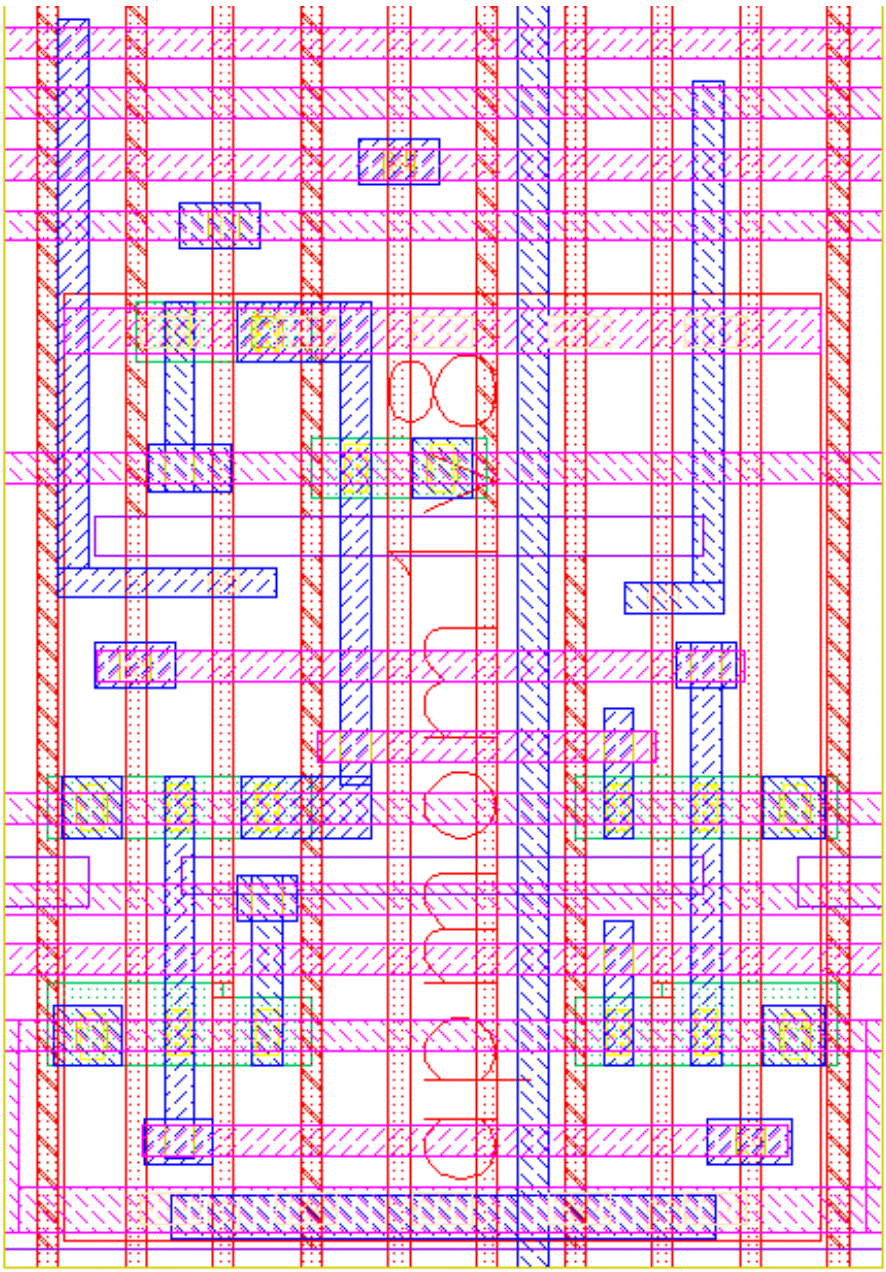


Figure 77 Zoomed and reduced view of single M-BC layout

In the image above, it can be seen a clearer view of the M-BC layout. Regarding the transistors, the two upper transistors correspond to the transmission gate, while the lower ones corresponds to the 6T SRAM and the multiplication transistors. Observing the shape of the capacitor, it can be also seen how the MOM covers almost completely the M-BC, except for the 6 tracks used for the connections of the input activations, and the connections of TSHORT and TSHORTb signals.

5.1.1.1. Results

Table 8 shows the dimensions of this single M-BC standard cell layout

	Width (μm)	Length (μm)	Area (μm^2)
Single M-BC Standard Cell	1.16	1.99	2.31

Table 8 Dimension of single M-BC Standard Cell

The dimensions of the M-BC obtained are slightly larger than expected. This is mainly due to two reasons. First of all, as mentioned at the beginning of this section, the size of the MOM has greatly influenced the dimensions of the M-BC, especially the width. As it can be seen in Fig. 77, in the center of the layout and at the top right there is some free space. This is because two extra polysilicon must be added to the design, so that the M-BC and the MOM have similar widths. It is important to remark that the layout of the MOM is offered by the technology, and the minimum width of this capacitor is 1μm. The second reason why the dimension of the cell, in this case the length, is greater than expected, is because the IA/IAb connections have been introduced in the standard cell. If these connections were made in the neuron patch layout, the M-BC standard cell would have a length of 1.51 μm and therefore an area of 1.76 μm^2.

5.1.2. Neuron Patch

Once the standard cell of the M-BC is designed, is relatively easy to design the layout of a neuron patch. Figure 78 shows the layout of a neuron patch.

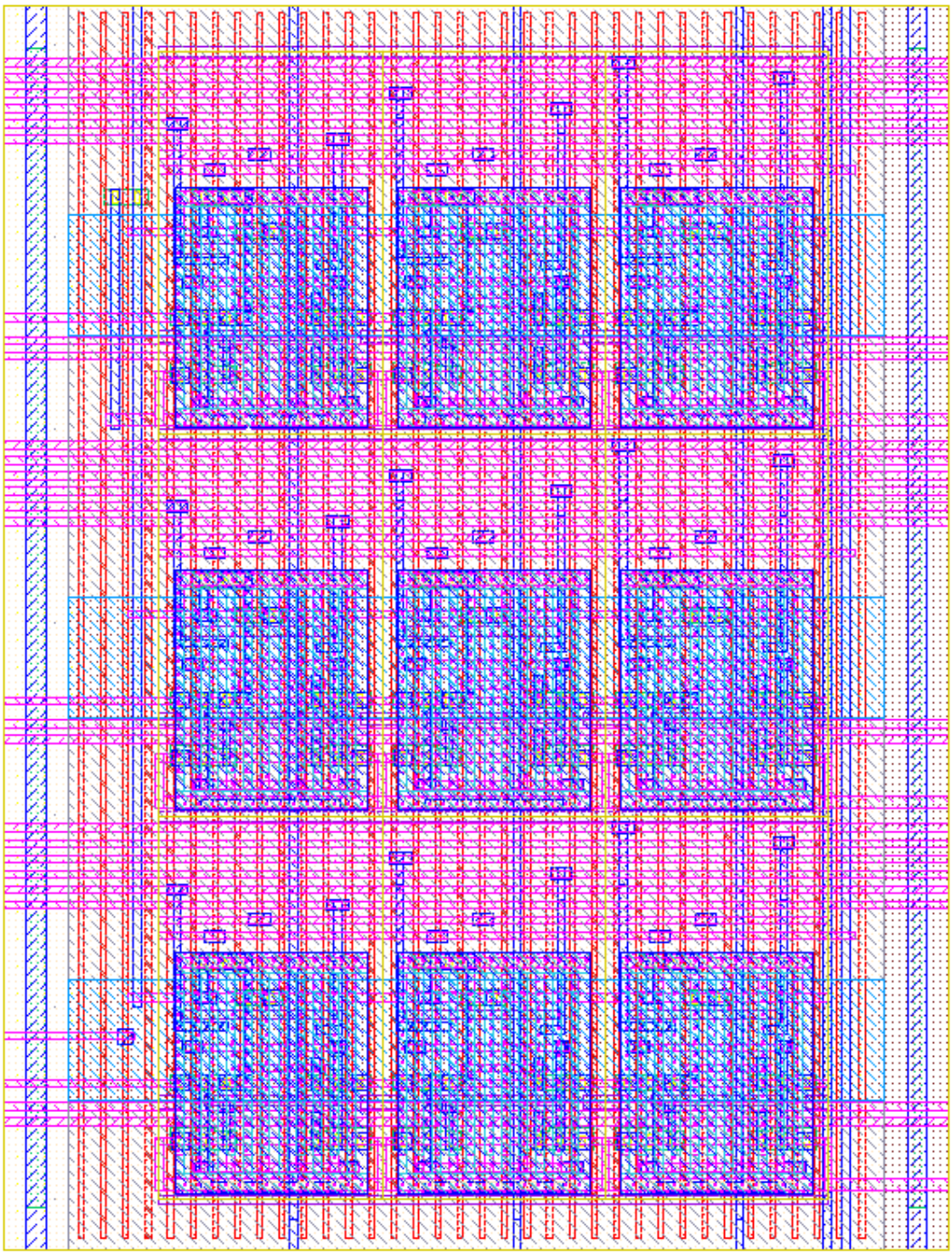


Figure 78 Neuron Patch Layout

Observing the 6 horizontal tracks used for the connections of the input activations, it can be seen how each M-BC in the same row connects to two different tracks. Figure 79 shows a zoomed view of these connections.

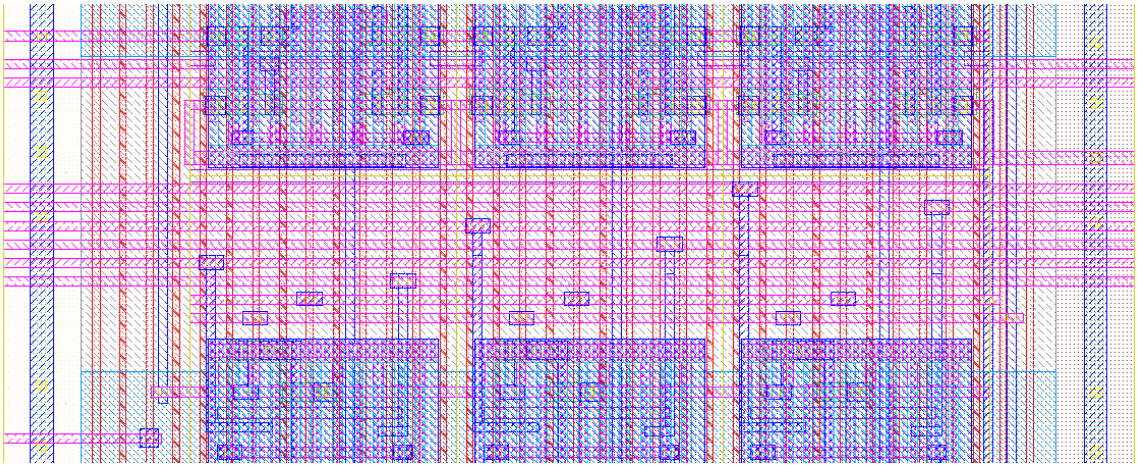


Figure 79 Zoomed view of input activations connections in a row

As it happens with the M-BC layout, in order to perform the post-layout simulation, it is necessary to complete the layout by adding the dummy polysilicon and the bulk connections. To make these body connections, the vertical tracks placed at both sides of the layout are used. In this layout, unlike in the layout of a single M-BC, what has been added is the discharge transistor. It is not necessary to add this component, since, as explained before, only one transistor is required in each neuron filter. However, this component has been added to the layout to observe what its position would be in the neuron filter. Finally, the discharge transistors have been placed in the upper left corner of the layout. Figure 80 shows a zoomed view of the position of the discharge transistor.

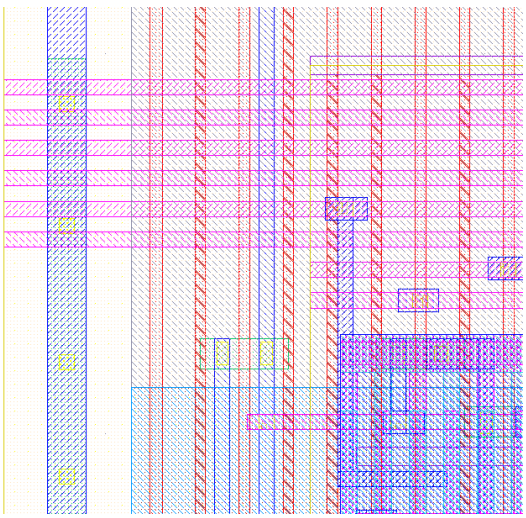


Figure 80 Zoomed view of the position of the discharge transistor

As it happens with the layout of a single M-BC, to connect several neuron patches together it is necessary to create a standard cell. However, in this case there are two types of standard cells. The only difference between them is that the neuron patches that are placed in the first row of the neuron array, have the discharge transistor incorporated. Figure 81 and 82 shows these two types of neuron patch standard cells.

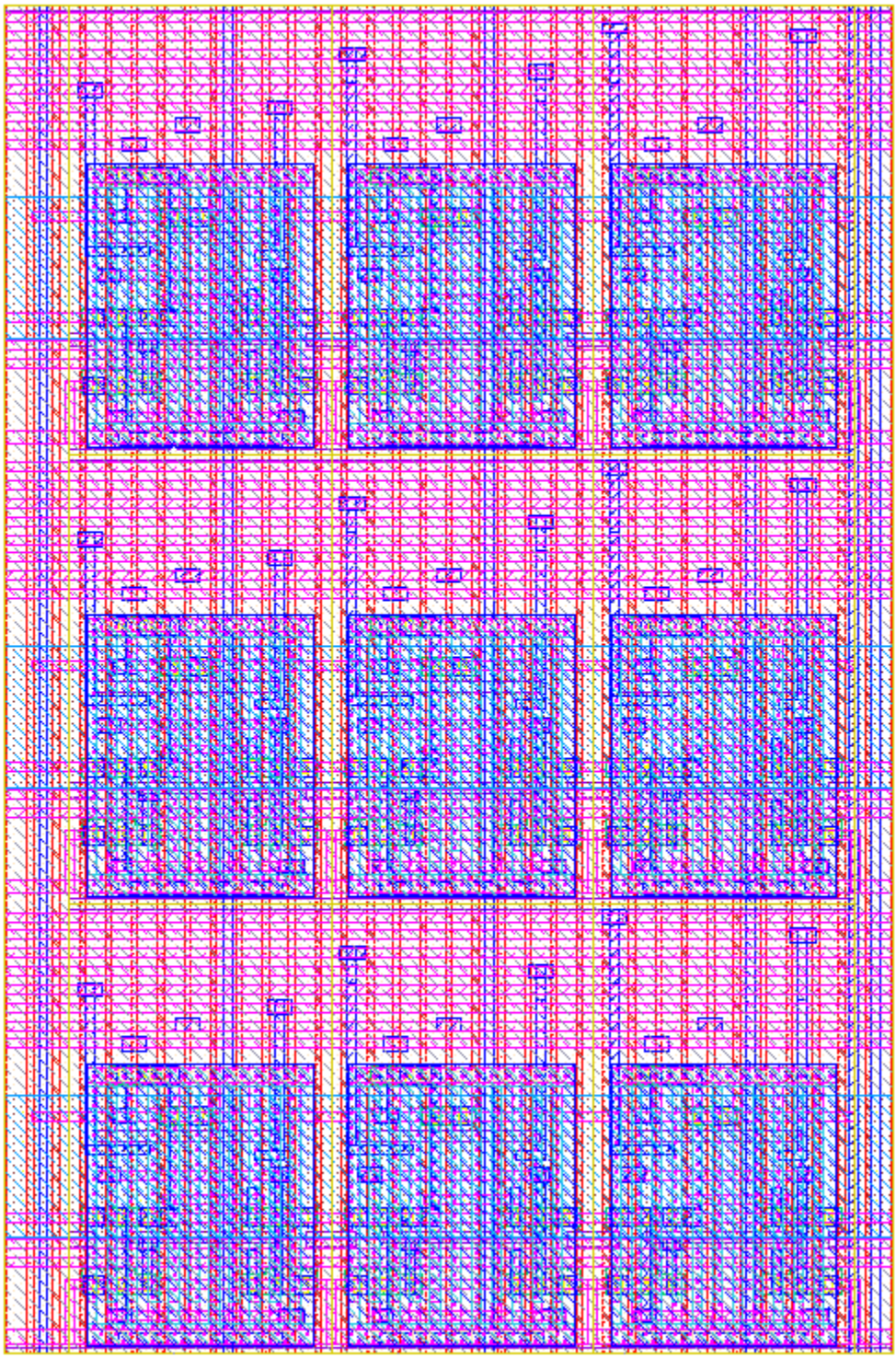


Figure 81 Neuron Patch Standard Cell Layout

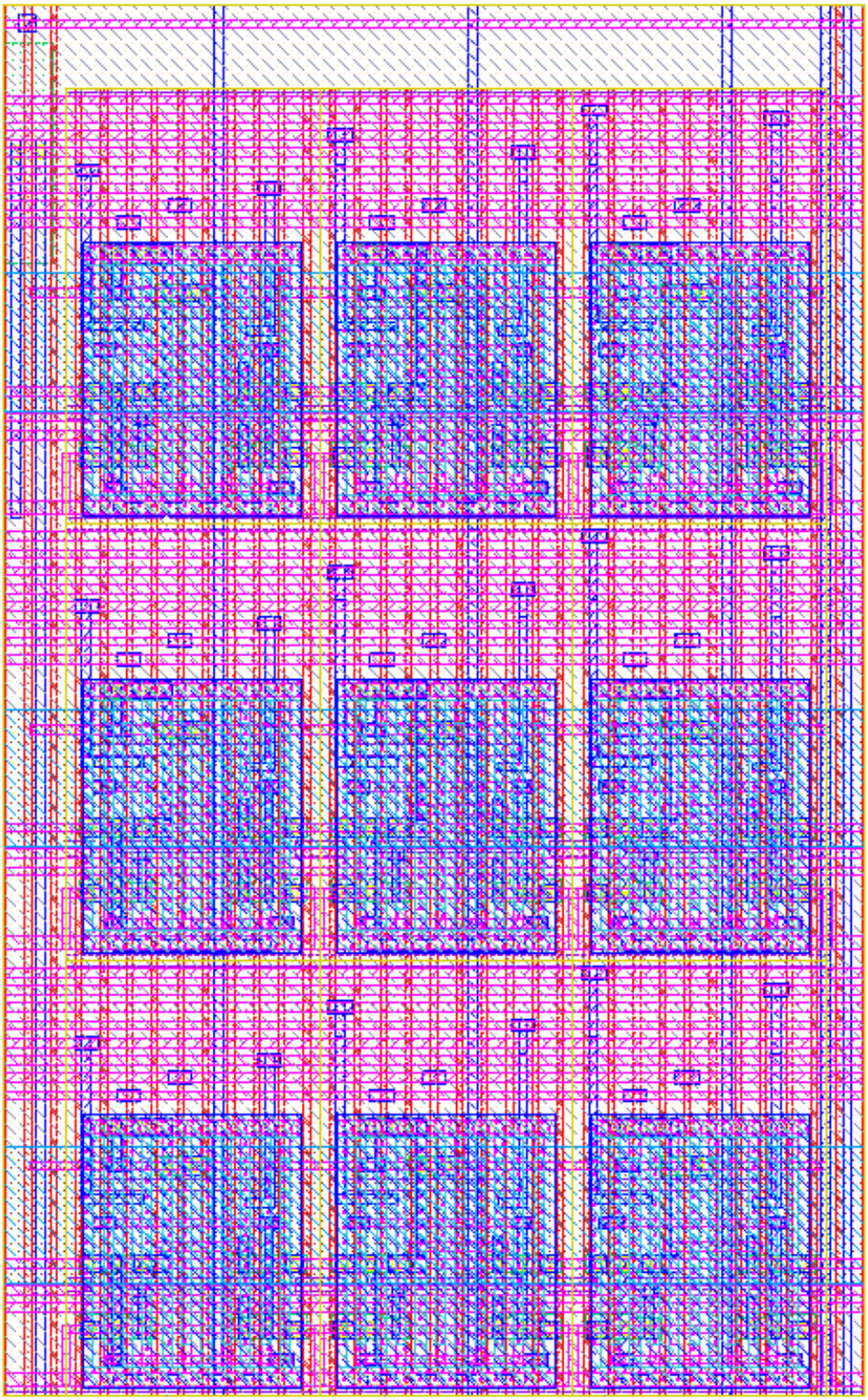


Figure 82 First row neuron patch Standard Cell Layout

5.1.2.1. Results

Table 9 shows the dimensions of these neuron patch standard cells layout.

	Width (µm)	Length (µm)	Area (µm^2)
Neuron Patch Standard Cell	3.944	5.976	23.57
First Row Neuron Patch Standard Cell	3.944	6.357	25.07

Table 9 Dimension of Neuron Patch Standard Cells

Knowing the dimensions of the neuron patches and the number of neuron patches within the neuron array, it is possible to estimate the dimensions of the neuron array. The neuron array is composed of 512 x 512 neuron patches, where 512 of those neuron patches have the dimensions of the neuron patch in the first row. In addition to the neuron patches, it is necessary to add the bulk connections of all transistors and the polysilicon dummies. Taking all this into account, the result of the approximate dimensions of the neuron array can be seen in Table 10.

	Width (mm)	Length (mm)	Area (mm^2)
Neuron Array	2.1	3.1	6.5

Table 10 Approximate dimension of the Neuron Array

5.2. Post-Layout M-BC Results

After making the layout of the design, it is very important to check that the circuit continues working correctly. This is very important since when the layout is made, due to the length of the track, the proximity of different tracks and many other reasons, several parasitic elements are generated in the design. These parasitic elements are usually not considered during schematic design, and they can change the behavior of the circuit.

There are two types of extractions of parasitic capacitors: decoupled extraction and coupled extraction. In the decoupled extraction all the parasitic coupling capacitors are lumped to the ground. The simulations done with this extraction are less accurate because some important effects such as cross coupling cannot be simulated. However, in the coupled extraction all the parasitic capacitors are extracted, increasing the accuracy and the realism of the simulation. So, due to these differences, it has been decided to carry out the coupled extraction.

5.2.1. Single M-BC

All the signals and testbench used for the post-layout design are the same as those used in the schematic design. The only difference is that the parasitic capacitors extracted in the layout have been added to the circuit. Figure 83 shows the voltage at the MOM capacitor and at the PA output obtained in the simulation.

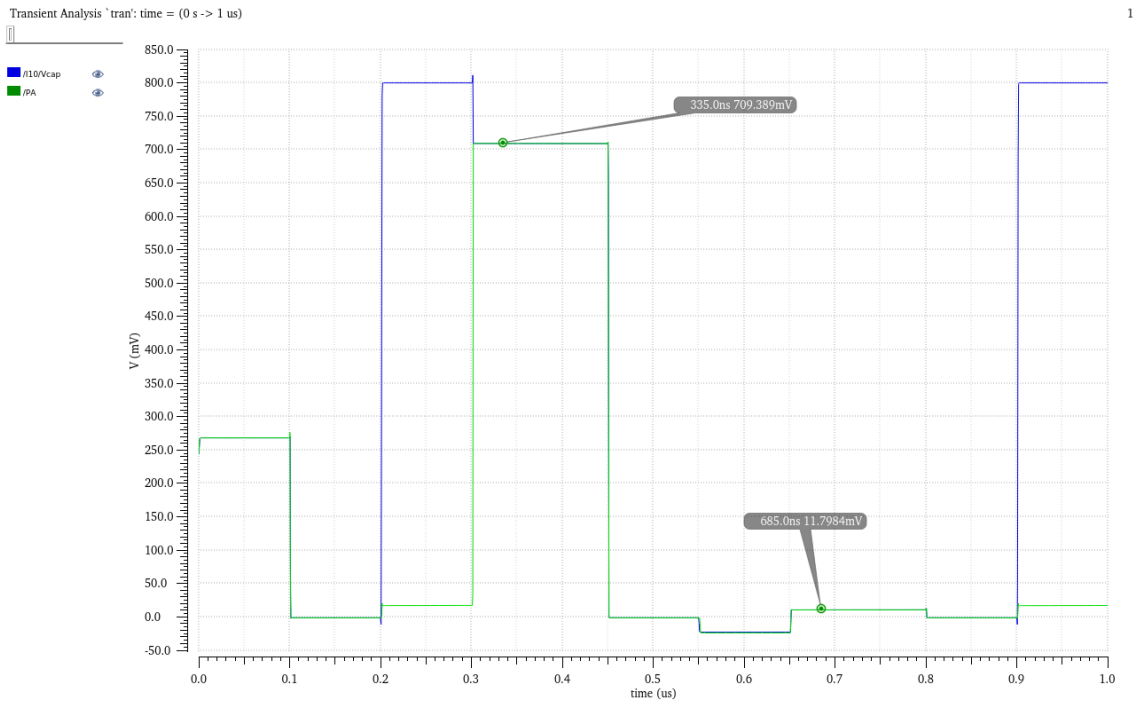


Figure 83 Single M-BC PA and Vcap post-layout simulation

As it can be seen in the image above, the behavior of the circuit is very similar to that obtained in the pre-layout simulation. Observing the voltage range of the PA output, it is very similar to that obtained in the pre-layout simulation.

To ensure that the M-BC works properly, it is necessary to check the stability of the circuit. Figure 84 shows the storage nodes of the SRAM during the simulation.

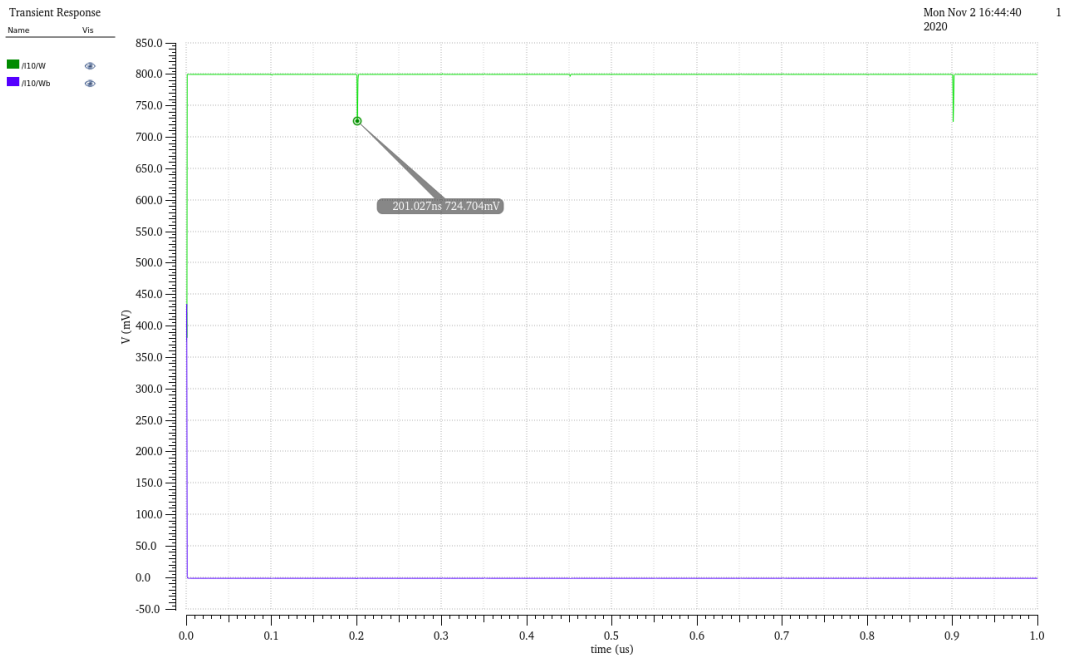


Figure 84 Storage nodes post-layout simulation in M-BC

As it can be seen, the stability of the M-BC has not been affected by the introduction of parasitic capacitors.

5.2.1.1. Results and comparison

In this section, the result obtained in the post-layout and pre-layout design will be compared. Table 11 shows these results.

Single M-BC	Maximum PA voltage (mV)	Minimum PA voltage (µV)	Maximum energy per operation (fJ)	Minimum energy per operation (fJ)
Pre-Layout	706.22	10.08	2.07	0.05
Post-Layout	709.39	11798	3.12	0.2

Table 11 Single M-BC pre-layout and post-layout comparison

As it can be seen, even though the maximum PA voltage has increased a little bit, the voltage range has decreased due to the minimum value of PA. However, this is not very significant, because, as it has been seen during the schematic design, as the number of M-BCs increases, the range of PA voltage also increases. What is more important is the increase in the energy per operation. This is expected because several new components have been added to the circuit. However, it can be seen that the increase has not been very large.

5.2.2. Neuron Patch

As in the case of a single M-BC, the signals used in the simulations are the same as in the schematic design simulations. However, since the pre-charge transistor has been added to the layout, a small modification has to be made in the testbench. Figure 85 shows an example of the PA output when the XNOR operation in all the M-BCs are '1' and '0'.

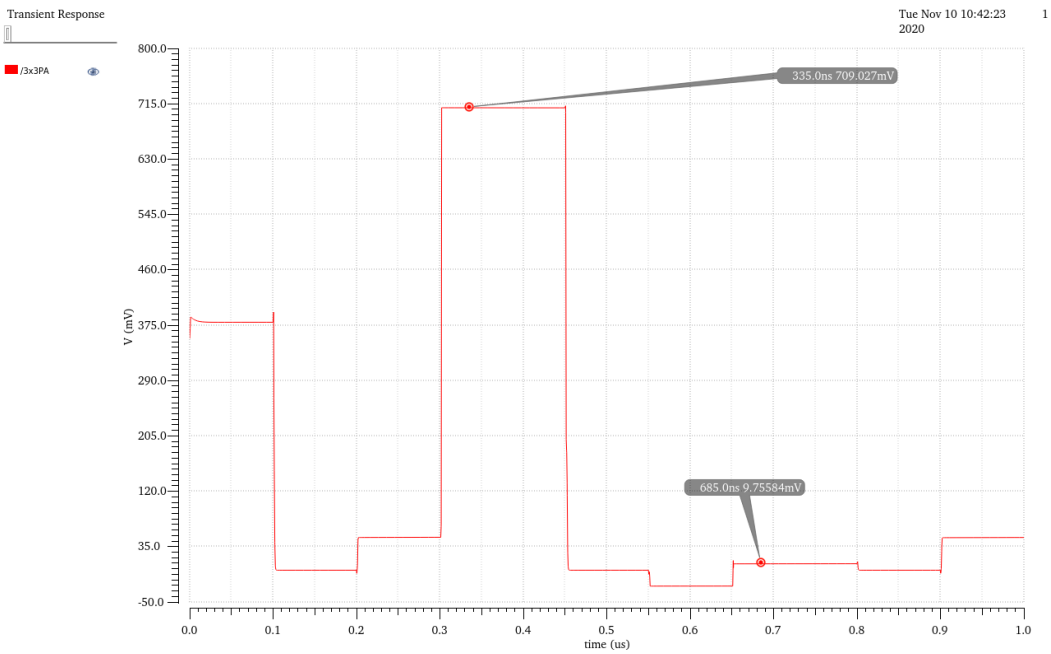


Figure 85 Post-layout Neuron Patch PA simulation with all XNOR operation '1' and '0'

It is very important that the PA signal is linear for a correct operation of the filtering of the input activations. For this, all possible PA voltage levels have been calculated. Figure 86 shows a curve that shows the linearity of the output.

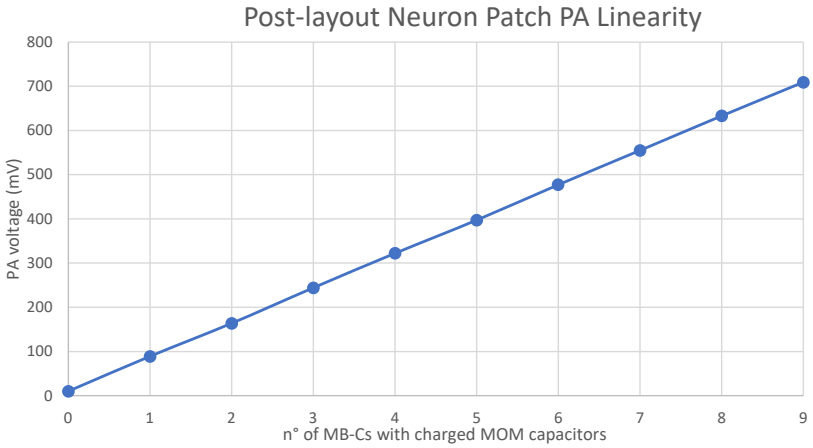


Figure 86 Linearity of post-layout neuron patch PA linearity

As it can be seen in the graph, the PA signal does not show any monotonicity problem and the size of the steps are almost equal. However, this is not enough to state that the design would not suffer from any linearity problems. For that, the linearity of a neuron filter would have to be analyzed.

5.2.2.1. Results and Comparison

In this section, the results obtained in the post-layout and pre-layout design will be compared. Table 12 shows these results.

Neuron Patch	Maximum PA voltage (mV)	Minimum PA voltage (mV)	Maximum energy per operation (fJ)	Minimum energy per operation (fJ)
Pre-Layout	720.8	2.26	23.34	0.67
Post-Layout	709.03	9.76	28.28	2.33

Table 12 Neuron Patch pre-layout and post-layout comparison

As it can be seen above, the PA voltage range decreases when performing the post-layout simulation. This is largely due to the fact that the number of parasitic capacitors is greater than in the schematic simulations. This happens, because in the post-layout simulations, two types of parasitic capacitors are considered, the capacitor added in the testbench (in both simulations) and the capacitors added by the layout (only post-layout simulation). The capacitor in the schematic, as previously explained, is introduced in the schematic to simulate the parasitic capacitance of the PA line. As before, the value of the parasitic capacitor is 10% of the sum of the MOM capacitors. In this case there are 3x3 MOMs.

Regarding energy per operation, as expected, it can be seen that it is higher when adding the layout to the simulation, since more elements have been added to the design. However, the increase is not critical.

5.3. Final Design vs Paper Design

After finishing the complete design of the M-BC and the 3x3 column of M-BCs, the results obtained are compared with the results of the reference paper [30]. However, before starting with the comparison, the results offered by the original paper will be analyzed. Table 13 indicates the most important results shown in the paper.

	Single M-BC Area (μm^2)	Neuron Array Area (mm^2)	Energy per filtering operation (pJ)	Energy Efficiency (TOPS/W)	Throughput (GOPS)
Paper design	1.8	≈ 9	10.6	866	18876

Table 13 Reference paper results

Some of the results shown above are not very well defined in the paper, that is why they will be explained. First, the paper indicated that the area of a single M-BC is 1.8 μm^2. However, if this value is multiplied by the total number of M-BCs in the neuron array, the area obtained is much smaller than the final area obtained in the paper. Therefore, this area value must refer to a smaller part of a M-BC. Figure 87 shows this part of the M-BC.

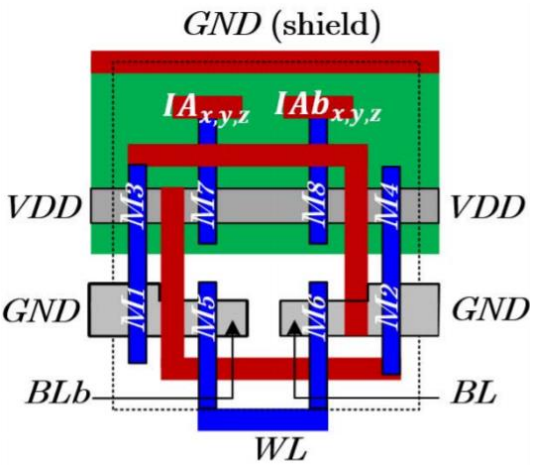


Figure 87 M-BC part for area comparison [30]

As it can be seen, the value of the area to which the paper refers is an M-BC part without the MOM capacitor, without the transmission gate, and without the IA/IAb, TSHORT/TSHORTb, WL and BL/BLB signals.

In addition, regarding the total area of the neuron array, it can be seen that the value shown in Table 13 is an approximate value. This is because the paper offers the dimensions of the chip, including the neuron array, the binarizing batch normalization, the IA SRAM and the IA BUF. So, to obtain an estimated area of the neuron array, a chip image has been used. Figure 88 shows the image of the chip.

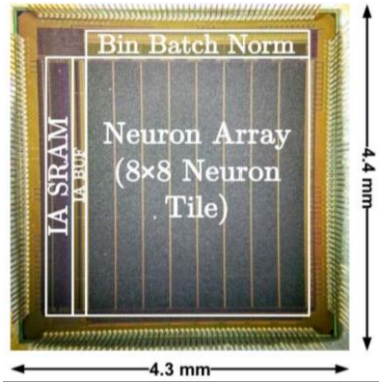


Figure 88 Paper chip dimensions [30]

The estimated dimensions of the neuron array are: 2.8 mm wide and 3.2 mm high. Considering these dimensions, the area of a complete M-BC could be obtained by dividing the total area of the neuron array by the total number of M-BCs. However, this result is not reliable because the inside distribution of this neuron array is unknown.

Finally, the paper indicates that the energy consumption for a filtering operation is 10.6 pJ. However, it does not specify whether it is an average value or a minimum or maximum value. Observing the values that have been obtained during the design, it is believed that the value shown in the paper refers to the average value.

Once the origin of some results of the paper design has been explained, the comparison of the two design will be made. Table 14 shows the most important results of both designs.

	Single M-BC Area (μm^2)	Neuron Array Area (mm^2)	Energy per filtering operation (pJ)	Energy Efficiency (TOPS/W)	Throughput (GOPS)
Paper design	1.8	≈ 9	10.6	866	18876
This project	0.87	≈ 6.5	≈ 7.9	≈ 1170	18876

Table 14 Comparison between this project design and paper design

As it can be seen in the table above, the results obtained in the project are much better than the results announced by the paper. Regarding the area, it has been possible to reduce a little more than 50% the area of a M-BC and approximately 28% the total area of the neuron array. As it can be seen, the reduction of a single M-BC is approximately twice the reduction of the neuron array. This may be due to the fact that the design of the paper uses more metallic layers for routing than this project. Regarding the energy consumption for a filtering operation, it has been reduced approximately 25%. This value has been obtained by scaling the energy consumption of a neuron patch, so the real improvement could differ a few percentages up or down. Something similar happens with energy efficiency, since it is inversely proportional to energy consumption per operation. The expected improvement of the energy efficiency of the design is approximately 35%. Finally, it can be seen that the throughput has not changed. This is because the original times of each phase of the MAC operation explained in section 3.2.2 have not been modified.

6. Conclusions and future improvements

6.1. Conclusions

The use of NNs has increased considerably in the past few years. Nowadays, they are used in many AI applications such as image recognition, robotics, and computer vision, and often achieving better accuracy than human. However, this accuracy is often related to very high computational complexity, which degrades the energy efficiency and the throughput of the NN. Therefore, NN processing techniques are necessary to obtain better results, and to be able to apply NNs in more applications. One of the most interesting techniques consists of moving the computation into the memory, saving the expensive data movement. This technique is called in-memory computing and there are many types of solutions where different circuits are used.

In this thesis the analysis and comparison of some of the most important in-memory computing solutions is carried out. After this analysis and comparison, the structure and operation of the CNN accelerator that has been taken as a reference is explained, emphasizing the most important criteria when designing. Finally, the design of the CNN accelerator is performed.

This thesis presents a 3x3 M-BCs column that shows much better results than the design on which the thesis is based. Specifically, the area and energy efficiency have been improved by 28% and 35 % respectively. These results are very good since area and energy efficiency are two of the most important characteristics in a CNN accelerator. This improvements are mainly due to the employment of the advanced 22 nm FDSOI technology instead of the 65nm CMOS technology used in [30].

In addition, two types of standard neuron patch or standard 3x3 M-BC column have been designed to facilitate the design of larger columns.

In conclusion, it can be said that the proposed design has fulfilled all the proposed objectives and the results obtained are satisfactory.

6.2. Future improvements

This thesis is focused on designing the neuron patch, a small part of the neuron array. However, the main challenge for future work is to design the neuron filter. This project has demonstrated the correct performance of the neuron patch after making the layout. However, it would be convenient to verify that the design works as expected when completing the entire neuron filter. Furthermore, it would also be convenient to perform a linearity analysis once the neuron filter design is completed. Finally, once this design is finished, it would be good to complete the CNN accelerator design with the output circuit designed in [52].

One of the most important improvements is related to throughput. In this design, despite reducing the size of the technology, it has been decided to maintain the MAC operation time, and therefore, the throughput. However, as seen in the simulations, this time could be reduced. Table 15 shows the suggested changes for a future design.

	Reset Phase (ns)	Binary-Multiply Phase (ns)	Accumulate Phase (ns)
This thesis	100	100	50
Future design	100	50	50

Table 15 Suggested future improvement to increase throughput

As it can be seen in the table, it is suggested to reduce the total time by 50 ns, which would increase throughput by 25%.

Another future work could be to design the M-BC using three metal layers instead of two. The clearest advantage that would be obtained with this modification is obvious; it would considerably reduce the area of the M-BC and the neuron array. However, it would also have clear negative consequences. As explained in section 5.1, in this design, five metallic layers have been used. So, by increasing the number of metallic layers used for routing, the number of metallic layers used to design the MOM capacitor gets reduced. By reducing the number of layers for the MOM capacitor design, the value of the MOM would decrease, and therefore, the voltage range of the PA. It would be interesting to make a comparison between these two designs to see if it would compensate the reduction in area with the reduction in the PA voltage range.

The final improvement is related with the MOM capacitor. As it has been mentioned during the thesis, the layout of the MOM capacitor used in the design is given by the technology, which is the biggest component inside the M-BC. That is why it would be interesting to design a MOM capacitor instead of using the one offered by the technology. The main advantage of this improvement is that the MOM capacitor would not impose any size restriction, because it could be designed with the desired dimensions. So, this would decrease the area of the M-BC. However, the biggest disadvantage of designing the MOM is that it would not be as accurately characterized as that offered by the technology.

Bibliography

[1] Khan, S., Rahmani, H., Shah, S. A. A., & Bennamoun, M. (2018). A guide to convolutional neural networks for computer vision. *Synthesis Lectures on Computer Vision*, *8*(1), 1-207.

[2] Aghdam, H. H., & Heravi, E. J. (2017). Guide to convolutional neural networks. *New York, NY: Springer*, *10*, 978-973.

[3] Namatēvs, I. (2017). Deep convolutional neural networks: Structure, feature extraction and training. *Information Technology and Management Science*, *20*(1), 40-47.

[4] Saha, S. (2018). A comprehensive guide to convolutional neural networks—the ELI5 way. *Towards Data Science*, *15*.

[5] Namatēvs, I. (2017). Deep convolutional neural networks: Structure, feature extraction and training. *Information Technology and Management Science*, *20*(1), 40-47.

[6] Dertat, A. (2017). Applied deep learning-part 4: Convolutional neural networks. *Towards Data Science. November*, *8*, 2017.

[7] Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, *105*(12), 2295-2329.

[8] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

[9] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248-255). Ieee.

[10] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[11] Tsang, S. (2018). Review: AlexNet, CaffeNet–Winner of ILSVRC 2012 (Image Classification). *A Medium Corporation*, *9*.

[12] Elhassouny, A., & Smarandache, F. (2019, July). Trends in deep convolutional neural Networks architectures: a review. In *2019 International Conference of Computer Science and Renewable Energies (ICCSRE)* (pp. 1-8). IEEE.

[13] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[14] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

[15] Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 1-62.

[16] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[17] Deng, L., Chu, H. H., Shi, P., Wang, W., & Kong, X. (2020). Region-Based CNN Method with Deformable Modules for Visually Classifying Concrete Cracks. *Applied Sciences*, *10*(7), 2528.

[18] Simons, T., & Lee, D. J. (2019). A review of binarized neural networks. *Electronics*, *8*(6), 661.

[19] Ghasemzadeh, M., Samragh, M., & Koushanfar, F. (2018, April). ReBNet: Residual binarized neural network. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)* (pp. 57-64). IEEE.

[20] Xiang, X., Qian, Y., & Yu, K. (2017, August). Binary Deep Neural Networks for Speech Recognition. In *INTERSPEECH* (pp. 533-537).

[21] Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., & Bengio, Y. (2016). Binarized neural networks. In *Advances in neural information processing systems* (pp. 4107-4115).

[22] Lin, X., Zhao, C., & Pan, W. (2017). Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems* (pp. 345-353).

[23] Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., & Bengio, Y. (2017). Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, *18*(1), 6869-6898.

[24] Yan, B., Li, B., Qiao, X., Xue, C. X., Chang, M. F., Chen, Y., & Li, H. (2019). Resistive Memory-Based In-Memory Computing: From Device and Large-Scale Integration System Perspectives. *Advanced Intelligent Systems*, *1*(7), 1900068.

[25] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.

[26] Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

[27] Sebastian, A., Le Gallo, M., Khaddam-Aljameh, R., & Eleftheriou, E. (2020). Memory devices and applications for in-memory computing. *Nature Nanotechnology*, 1-16.

[28] Singh, G., Chelini, L., Corda, S., Awan, A. J., Stuijk, S., Jordans, R., ... & Boonstra, A. J. (2019). Near-memory computing: Past, present, and future. *Microprocessors and Microsystems*, *71*, 102868.

[29] Ambrogio, S., Narayanan, P., Tsai, H., Shelby, R. M., Boybat, I., di Nolfo, C., ... & Killeen, B. (2018). Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature*, *558*(7708), 60-67.

[30] Valavi, H., Ramadge, P. J., Nestler, E., & Verma, N. (2019). A 64-tile 2.4-Mb in-memory-computing CNN accelerator employing charge-domain compute. *IEEE Journal of Solid-State Circuits*, *54*(6), 1789-1799.

[31] Zhang, J., Wang, Z., & Verma, N. (2017). In-memory computation of a machine-learning classifier in a standard 6T SRAM array. *IEEE Journal of Solid-State Circuits*, *52*(4), 915-924.

[32] Yin, S., Jiang, Z., Seo, J. S., & Seok, M. (2020). XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks. *IEEE Journal of Solid-State Circuits*, *55*(6), 1733-1743.

[33] Tsai, H., Ambrogio, S., Narayanan, P., Shelby, R. M., & Burr, G. W. (2018). Recent progress in analog memory-based accelerators for deep learning. *Journal of Physics D: Applied Physics*, *51*(28), 283001.

[34] Shafiee, A., Nag, A., Muralimanohar, N., Balasubramonian, R., Strachan, J. P., Hu, M., ... & Srikumar, V. (2016). ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News*, *44*(3), 14-26.

[35] Weste, N. H., & Harris, D. (2015). *CMOS VLSI design: a circuits and systems perspective*. Pearson Education India.

[36] Pavlov, A., & Sachdev, M. (2008). *CMOS SRAM circuit design and parametric test in nano-scaled technologies: process-aware SRAM design and test* (Vol. 40). Springer Science & Business Media.

[37] Therman, C. Computation Structures: The Memory Hierarchy. [Online] Available: https://computationstructures.org/lectures/caches/caches.html#1

[38] Saun, S., & Kumar, H. (2019, October). Design and performance analysis of 6T SRAM cell on different CMOS technologies with stability characterization. In *IOP Conference Series: Materials Science and Engineering* (Vol. 561, No. 1, p. 012093). IOP Publishing.

[39] Vassighi, A. (2004). *Heat and Power Management for High Performance Integrated Circuits*. University of Waterloo.

[40] Singh, A., & Singh, S. (2016). Evolution of CMOS Technology-Past Present and Future. *International Journal of Engineering Research & Technology (IJERT)*, *5*(02).

[41] Global Foundries. [Online] Available: https://www.globalfoundries.com/technology-solutions/cmos/fdx/22fdx

[42] Gwennap, L. (2016). FD-SOI offers alternative to FINFET. *Posted at https://www. globalfoundries. com/sites/default/files/fd-soi-offers-alternative-tofinfet. pdf*.

[43] Body - Bias Scaling for GLOBALFOUNDRIES 22FDx Technology New Dimension to Explore the Design. [Online] Available: https://docplayer.net/54679367-Body-bias-scaling-for-globalfoundries-22fdx-technology-new-dimension-to-explore-the-design.html

[44] Purdy, M., & Daugherty, P. (2017). How AI boosts industry profits and innovation. *Accenture Ltd, Dublin, Ireland ISBN12560543*.

[45] Mitchell, M. (2019). *Artificial intelligence: A guide for thinking humans*. Penguin UK.

[46] Baruah, L. (2017). Performance Comparison of Binarized Neural Network with Convolutional Neural Network.

[47] Ahmet, C. (2018). *Artificial Intelligence: How Advance Machine Learning Will Shape The Future Of Our World*. Shockwave Publishing via PublishDrive.

[48] Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., ... & Asari, V. K. (2019). A state-of-the-art survey on deep learning theory and architectures. *Electronics*, *8*(3), 292.

[49] Bergel, A. (2020). Agile Artificial Intelligence in Pharo.

[50] Tchircoff, A. (2017). The mostly complete chart of Neural Networks, explained. *Towards Data Science*, 1-29.

[51] VOVES, J. (2009). Nanoelectronics and nanolithography.

[52] Martinez P. (2020). Design of an output interface for an in-memory-computing CNN accelerator

[53] Ward-Foxton S. (2020) Processor-in-memory chip speeds AI computations. [Online] Availabe: https://www.embedded.com/processor-in-memory-chip-speeds-ai-computations/

[54] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Glossary

ADC	Analog to Digital Converter
AI	Artificial Intelligence
BN	Batch Normalization
BNN	Binary Neural Network
BOX	Buried Oxide
CNN	Convolutional Neural Network
CPU	Central Processor Unit
DL	Deep Learning
DNN	Deep Neural Network
DRAM	Dynamic Random-Access Memory
EACB	Error-Adaptive Classifier Boosting
FDSOI	Fully Depleted Silicon-On Insulator
GOPS	Giga operations per second
GPU	Graphics Processing Unit
HSNM	Hold Static Noise Margin
IA	Input-Activation
ILSVRC	ImageNet Large Scale Recognition Challenge
IoT	Internet of Things
LVS	Layout-Versus-Schematic
MAC	Multiply-and-Accumulate
M-BC	Multiplying Bit Cell
MIM	Metal-Insulator-Metal
ML	Machine Learning
MOM	Metal-Oxide-Metal

MVM	Matrix-Vector Multiplication
NN	Neural Network
PA	Pre-Activation
PCM	Phase change memory
ReLU	Rectified Linear Unit
RRAM	Resistive Random-Access Memory
RSNM	Read Static Noise Margin
RVT	Regular Threshold Voltage
SNM	Static Noise Margin
SNR	Signal-to-Noise-Ratio
SOI	Silicon-On Insulator
SRAM	Static Random-Access Memory
STT-MRAM	Spin Transfer Torque Magnetoresistive Random Access Memory
TOPS	Tera operations per second
UHVT	Ultra-Low Leakage High Threshold Voltage
ULL	Ultra-Low Leakage
VTC	Voltage Transfer Characteristic
V_{th}	Threshold Voltage
WSNM	Write Static Noise Margin