JUNIA Grande école d'ingénieurs
HEI·ISEN·ISA

Université de Lille

iemn Institut d'Électronique, de Microélectronique et de Nanotechnologie
UMR CNRS 8520

THÈSE

Présentée à l'Université de Lille

École doctorale 632 : Science de l'ingénierie et des systèmes

en vue d'obtenir le grade de

DOCTEUR

en

Électronique, Microélectronique, Nanoélectronique et Micro-ondes

---

Mixed-Signal In-Memory Matrix-Vector Multiplication for Ultra-Low Power Embedded Machine Learning

Multiplication de vecteur-matrice à signaux mixtes dans la mémoire à très faible consommation pour l'apprentissage machine embarqué

Présentée et soutenue publiquement par :

Kévin HÉRISSÉ

le 16 Décembre 2022

Jury

M. Ian O'CONNOR	Pr. - École Centrale de Lyon	Rapporteur, Président du jury
M. Yong LIAN	Pr. - York University	Rapporteur
Sylvain SAIGHI	Pr. - Université de Bordeaux	Examinateur
Mme Édith BEIGNÉ	DR - Meta	Examinatrice
M. Sylvain CLERC	Ingénieur - STMicroelectronics	Examinateur
Mme Andreia CATHELIN	Dr. HDR - STMicroelectronics	Examinatrice
M. Benoit LARRAS	Dr. - Junia, Université de Lille	Invité
M. Antoine FRAPPÉ	Dr. HDR - Junia, Université de Lille	Co-Directeur de Thèse
M. Andreas KAISER	DR - Junia, Université de Lille	Directeur de Thèse

Acknowledgements

This manuscript is the result of 4 years of work. I was fortunate enough to be able to go through this thesis thanks to all the people around me that support me and push me higher and I would like to thank them.

First, I would like to thank all the members of my jury, it was an honour to present you my work. I want to give special thanks to Pr. Ian O'Connor and Pr. Yong Lian who read this manuscript, your comments help me to do better work, thank you for this.

None of this work could have been possible without the support of my thesis directors Antoine Frappé, Andreas Kaiser, and my advisor Benoit Larras. I'm grateful to have you by my side during this Ph.D., I wish more Ph.D. students could have an advisor team like you.

I want to thank my colleagues for their support, their humour, and their kindness. You were once my teachers and I'm happy to know you more. Thanks to Bruno, Jean-Marc, Florence, Jean-François, Emmanuel, Axel, Valérie, Justine and, Etienne.

A special thanks to my Ph.D. colleagues, Julien, Mathieu, Antoine, Nicolas, Jean-Baptiste and Soufiane. You were great support and an incredible team to work with.

During these 4 years, I was glad enough to have been surrounded by great people. Great friends like Pierre-Raphaël, Marie-Jeanne, Emma G., Perrine, Quentin T., Mathilde, Quentin L., Raphaël, Alexis, Axelle, Claire, Jeremie, Meline, Marwane, Sarah, and Augustin.

I got the chance to work on other projects alongside the thesis with my friends Thomas, Eric and Marie-Camille. I learned a lot by your side.

Of course, my family was a great support. First, my parents, Isabelle and Franck they support me and push me to follow my instinct since I'm born. My sister, Margaux, was an equally great support. Thank you for everything.

A special thanks to my grandparents, Maurice and Marie-Thérèse, for their support during my studies. Romain, Louise, Cathy, Jean-Pierre, Jacky, Brigitte, Martine, Laurent, and all the family members I can't name, thank you for listening to me, making me laugh, and lifting my spirit when needed.

Finally, I will end this acknowledgment and thank my girlfriend Coralie. She is the only one who knows really what the thesis was like for me. She's great, she's brilliant and was and still is a great support in life. Thank you for everything. I love you.

I should probably thank my therapist, but I'm not sure it's something that we do.

Remerciements

Ce manuscrit est le résultat de 4 années de travail. J'ai eu la chance de pouvoir mener à bien cette thèse grâce à toutes les personnes de mon entourage qui me soutiennent et me tirent vers le haut et je tiens à les remercier.

Tout d'abord, je tiens à remercier tous les membres de mon jury, ce fut un honneur de vous présenter mon travail. Je tiens à remercier tout particulièrement le Pr. Ian O'Connor et Pr. Yong Lian qui ont lu ce manuscrit, vos commentaires m'ont aidé à faire un meilleur travail, je vous en remercie.

Aucun de ces travaux n'aurait été possible sans le soutien de mes directeurs de thèse, Antoine Frappé, Andreas Kaiser, et de mon encadrant Benoit Larras. Je suis reconnaissant de vous avoir eu à mes côtés durant ce doctorat, je souhaite que plus de doctorants puissent avoir une équipe d'encadrants comme vous.

Je tiens à remercier mes collègues pour leur soutien, leur humour et leur gentillesse. Vous avez été mes professeurs et je suis heureux de vous connaître davantage. Merci à Bruno, Jean-Marc, Florence, Jean-François, Emmanuel, Axel, Valérie, Justine et Etienne.

Un merci spécial à mes collègues de doctorat, Julien, Mathieu, Antoine, Nicolas, Jean-Baptiste et Soufiane. Vous avez été d'un grand soutien et une équipe incroyable avec laquelle travailler.

Durant ces 4 années, j'ai eu le bonheur d'être entourée de personnes formidables. De grands amis comme Pierre-Raphaël, Marie-Jeanne, Emma G., Perrine, Quentin T., Mathilde, Quentin L., Raphaël, Alexis, Axelle, Claire, Jérémie, Méline, Marwane, Sarah, et Augustin.

J'ai eu la chance de travailler sur d'autres projets, accompagnés de mes amis Thomas, Eric et Marie-Camille. J'ai beaucoup appris à vos côtés.

Bien sûr, ma famille a été d'un grand soutien. Tout d'abord, mes parents, Isabelle et Franck, ils me soutiennent et me poussent à suivre mon instinct. Ma sœur, Margaux, a été un soutien tout aussi important. Merci pour tout.

Un merci particulier à mes grands-parents, Maurice et Marie-Thérèse, pour leur soutien durant mes études. Louise, Romain, Cathy, Jean-Pierre, Jacky, Brigitte, Martine, Laurent et tous les membres de la famille que je ne peux pas nommer, merci de m'écouter, de me faire rire et de me remonter le moral quand il le faut.

Enfin, je terminerai en remerciant ma compagne Coralie. Elle est la seule à savoir réellement ce qu'a été la thèse pour moi. Elle est géniale, elle est brillante et a été et est toujours un grand soutien dans la vie. Merci pour tout. Je t'aime.

Je devrais probablement remercier ma psy, mais je ne sais pas si c'est quelque chose qui se fait.

Abstract

The applications for embedded artificial intelligence are numerous and cover multiple domains, such as consumer electronics, home automation, health, and industry. They require dedicated chips bringing intelligence close to the sensor while maintaining a low energy consumption. Although many types of neural networks (NN) exist, they all rely on the same basic computations which are Matrix-Vector Multiplications (MVM) composed of Multiply-and-Accumulate (MAC) operations. Optimizing the energy efficiency of MAC operations is a great lever to reduce global power consumption. In a classic Von Neumann architecture, the limitation implied by data access caps the efficiency at 10 TOPS/W considering a 50 fJ/byte energy consumption for data movement. In-memory computing (IMC) helps reduce the energy overhead for accessing data by processing them close to where they are stored. This thesis analyses the state-of-the-art NN architectures and the works for Voice Activity Detection (VAD) and Keyword Spotting (KWS), to show that energy consumption and accuracy are more important parameters than throughput for embedded applications. Furthermore, analysis of the state-of-the-art of IMC shows that the available time to perform NN operations can be advantageously leveraged. This work presents a time- and current-based analog IMC concept, where current sources charge/discharge a capacitive line during a time pondered by the product of two numbers, therefore performing multi-bit MAC oper-

ations through time. An implementation of the proposed architecture in a 28 nm FDSOI CMOS technology is presented. The integrated circuit prototype integrates 4 neurons with 100 inputs and 5-bit inputs and weights. The structure performs the multi-bit MVM using the proposed time- and current-based analogue IMC method within a maximum latency of 4.5 µs, perfectly suitable with most embedded applications. The measured energy efficiency allows envisioning >50 TOPS/W if deployed over a 100-neuron array.

Résumé

Les applications de l'intelligence artificielle embarquée sont nombreuses et couvrent de multiples domaines, tels que l'électronique grand public, la domotique, la santé et l'industrie. Elles nécessitent des puces dédiées apportant l'intelligence à proximité du capteur tout en maintenant une faible consommation d'énergie. Bien qu'il existe de nombreux types de réseaux neuronaux (Neural Networks - NN), ils reposent tous sur les mêmes calculs de base, à savoir des multiplications matricielles et vectorielles (MMV) composées d'opérations de multiplication et d'accumulation (MAC). L'optimisation de l'efficacité énergétique des opérations MAC est un excellent levier pour réduire la consommation énergétique globale. Dans une architecture Von Neumann classique, la limitation liée à l'accès aux données plafonne l'efficacité à 10 TOPS/W en considérant une consommation d'énergie de 50 fJ/byte pour le déplacement des données. Le traitement en mémoire (In-Memory Computing - IMC) permet de réduire la surcharge énergétique liée à l'accès aux données en les traitant à proximité de l'endroit où elles sont stockées. Cette thèse analyse l'état de l'art des architectures NN et les travaux pour la détection d'activité vocale (Vocal Activity Detection - VAD) et le repérage de mots-clés (Keyword Spotting - KWS), pour montrer que la consommation d'énergie et la précision sont des paramètres plus importants que le débit pour les applications embarquées. En outre, l'analyse de l'état de l'art de l'IMC montre que le

temps disponible pour effectuer les opérations du NN peut être avantageusement exploité. Ce travail présente un concept d'IMC analogique basé sur le temps et le courant, où des sources de courant chargent/déchargent une ligne capacitive pendant un temps pondéré par le produit de deux nombres, réalisant ainsi des opérations MAC multi-bits à travers le temps. Une mise en œuvre de l'architecture proposée dans une technologie FDSOI de 28 nm est présentée. Le prototype de circuit intégré intègre 4 neurones avec 100 entrées et des entrées et poids de 5 bits. La structure exécute le MMV multi-bits en utilisant la méthode IMC analogique proposée, basée sur le temps et le courant, avec une latence maximale de 4,5 µs, parfaitement adaptée à la plupart des applications embarquées. L'efficacité énergétique mesurée permet d'envisager une efficacité supérieure à 50 TOPS/W s'il est déployé sur un réseau de 100 neurones.

Contents

List of Figures

List of Tables

Acronym

ADC Analog to Digital Converter.

AI Artificial Intelligence.

AIMC Analog In-Memory Computing.

ASR Automatic Speech Recognition.

BEOL Back-end of line.

BPF Band Pass Filter.

CNN Convolution Neural Network.

CPU Central Processing Unit.

DAC Digital to Analog Converter.

DIMC Digital In-Memory Computing.

DNN Deep Neural Network.

DTX Discontinuous transmission.

FDSOI Fully Depleted Silicon On Insulator.

fefet Fe Field Effect Transistor.

FEOL Front-end of line.

FFT Fast Fourier Transform.

FPGA Field-Programmable Gate Array.

GOPS Giga Operation Per Second.

GPU Graphic Processing Unit.

HMM Hidden Markov Model.

IAF Integrate And Fire.

IMC In-Memory Computing.

IoT Internet of Things.

IROC Instant Rate Of Change.

KWS Keyword Spotting.

LNA Low Noise Amplifier.

LSTM Long Short-Term Memory.

MAC Multiplication and Accumulation.

MVM Matrix-Vector Multiplication.

OA Operational Amplifier.

PCM Phase change Memory.

RRAM Resistive Random-Access Memory.

SNR Signal to Noise Ratio.

SPI Serial Peripheral Interface.

SRAM Static Random-Access Memory.

STTRAM Spin-transfer-torque Random-Access Memory.

TinyML Tiny Machine Learning.

TOPS Tera Operation Per Second.

VAD Voice Activity Detection.

WL Wordline.

Associated Publications

Conferences

Kévin Hérissé et al. "Mixed-Signal In-Memory Multi-bit Matrix-Vector Multiplication". In: *15ème Colloque National du GDR SOC2.* Rennes, France: Jun. 2021.

Kévin Hérissé et al. "Mixed-Signal In-Memory Multi-bit Matrix-Vector Multiplication". In: *IBM IEEE CAS/EDS – AI Compute Symposium.* Virtual, United States: Oct. 2021.
Award of the third best poster at IBM IEEE CAS/EDS - AI Compute Symposium 2021.

Kévin Hérissé et al. "Keyword Spotting System using Low-complexity Feature Extraction and Quantized LSTM". In: *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS).* Dubai, United Arab Emirates: IEEE, Nov. 2021.
Paper present in Appendix A.

Patent

Patent application for the time- and current-based mixed-signal principle. **Kévin Hérissé**, Benoit Larras, Antoine Frappé, Andreas Kaiser.

Journal

In preparation : **Kévin Hérissé** et al. "A 99.2 TOPS/W In-memory Vector Multiplication Macro using Mixed-signal Incremental Time-Based MACs". In: *IEEE Transactions on Circuits and Systems I: Regular Papers*.

This work was supported in part by the French National Research Agency under Grant ANR-18-CE24-0006-01 LEOPAR and in part by the Nano 2022 - IPCEI program.

Introduction

The applications for embedded artificial intelligence are numerous and cover multiple domains, such as consumer electronics, home automation, health, and industry. They require dedicated chips bringing intelligence close to the sensor while maintaining a low energy consumption. Although many types of neural networks (NN) exist, they all rely on the same basic computations which are Matrix-Vector Multiplications (MVM) composed of Multiply-and-Accumulate (MAC) operations. Reducing the number of operations, and quantizing the weights of the NN can offer some efficiency gain. However, optimizing the energy efficiency of MAC operations is a great lever to reduce global power consumption. In a classic Von Neumann architecture, the limitation implied by data access caps the efficiency at 10 TOPS/W considering a 50 fJ/byte energy consumption for data movement. In-memory computing (IMC) helps reduce the energy overhead for accessing data by processing them close to where they are stored.The contributions of this thesis are the following :

- Analysis of state-of-the-art NN architectures, highlighting that the main computation blocks of artificial intelligence algorithms are MVM and MAC operations.

- Analysis of state-of-the-art works for Voice Activity Detection (VAD) and Keyword Spotting (KWS) applications. One of the outcomes of this analysis is that, for embedded applications, energy consumption and accuracy are more important parameters than throughput. This is a fundamental difference with accelerators for deep learning.

- Analysis of state-of-the-art works on IMC. One of the outcomes of this analysis is that the time available to perform the operations can be advantageously leveraged for embedded applications. A time-based multiplication scheme and a concurrent current combination for performing multi-bit MAC operations are then proposed, with promising efficiency levels.

- Description of the time- and current-based analogue IMC, where current sources charge/discharge an accumulation line during a time weighted by the product of two numbers, therefore performing multi-bit MAC operations through time.

- Implementation of the proposed architecture in a 28 nm FDSOI CMOS technology. The integrated circuit prototype integrates 4 neurons with 100 inputs and 5-bit coding of inputs and weights. The structure performs the multi-bit MVM using the proposed time- and current-based analogue IMC method within a maximum latency of 4.5 µs, perfectly suitable for most embedded applications. The measured energy efficiency allows envisioning a 99.2 TOPS/W if deployed over a 100-neuron array.

The manuscript is structured as follows:
In **Chapter 1** we will introduce machine learning by presenting the basic structure of a perceptron neuron, the matrix-vector multiplication and the multiply and accumulate (MAC) operation. This informative chapter will then describe how a simple neural network such as the Feed Forward neural network can *learn*. Finally, more advanced networks like CNN and LSTM will be presented as well as the different datasets used to benchmark them.
Thanks to this information, **Chapter 2** will introduce the TinyML environment and describe how a complex system can be built for integration on energy-constrained devices. The work especially focuses on audio applications such as Voice Activity Detection (VAD) and Key-

word Spotting (KWS), how the feature extraction and the computation are performed and highlight the need for low-consumption neural networks.

The **Chapter 3** present an overview of the State-of-the-art works on IMC, from Digital to Mixed-Signal architecture and identifies the pro and cons of each solution. This highlight the promising perspective of the proposed time- and current-based analog IMC architecture. This time- and current-based concept is presented in **Chapter 4** and its implementation in 28 nm FDSOI is detailed in **Chapter 5**. The measurement results are presented in **Chapter 6** where the functioning of the circuits is validated against Matlab simulations and efficiency and accuracy are compared to the State-of-the-art.

Finally, we conclude this work with future perspectives of improvements of the system in terms of accuracy and energy consumption.

Chapter 1

Neural Networks

1.1 Introduction

Artificial Intelligence (AI) is used in a lot of applications in our everyday life. AI algorithms are used on social networks for targeted advertising. They are running on our phones to predict the next word we will type, or search for a specific object inside our photo library. They are used to drive autonomous cars, behind Instagram filters, and to identify skin cancer from a picture of a patient's back. Either for entertainment or more pragmatic applications, AI is everywhere. These applications work thanks to Machine Learning (ML), which is a field of AI where a computer can learn to perform a task without specific instructions, only from datasets. ML allows neural networks to *learn* how to predict or classify data. There are mainly three different approaches to training a neural network:

- **Supervised Learning**
  This algorithm works with an input dataset and the corresponding classes of this data. With each data going through the network, the algorithm will compare the prediction to the target class. The parameters will then be updated so that the prediction is closer to the target class.

- **Unsupervised Learning**
  This algorithm works with datasets where we don't have the target class. Unsupervised learning is mainly used to create a cluster of data. This type of learning allows segmenting a set of customers, for example.

- **Reinforcement Learning**
  The reinforcement learning algorithm consists of learning continuously by trial and error, trying to reach the highest reward value inside a set environment. Through multiple experiences, it will try to find the best behavior that maximizes the reward.

Complex networks, with millions of parameters like AlexNet [1], uses supervised learning and can classify images with 15.3% top-5 error rates among 1,000 classes, meaning that 84.7% of the time the right class is in the top-5 predictions on an image. Others can classify audio signals and are able to identify 10 keywords, silence, and background noise with 93.09% accuracy [2]. Neural networks are also used to identify cancerous lungs cells with 94% accuracy from images [3].
The variety of architectures available to classify and predict data makes it difficult from an application point of view to spot how we can reduce the consumption of such a network to implement it on ASIC. By studying different architectures, it is possible to identify one common block we can optimize for better efficiency. In this chapter we will:

- Study the main building blocks of neural networks and introduce the Multiply and Accumulate (MAC) and the Matrix-Vector Multiplication (MVM) operations.
- Study the Feed Forward Neural Network and the backpropagation principle with the gradient descent algorithm that allows the training of the networks.
- Present the different metrics used to compare neural networks.
- Present different types of Neural Networks: CNN, LSTM, and describe their architecture.
- Present the datasets used for benchmarking Neural Networks in this work.

1.2 The perceptron and MAC operation

1.2.1 The perceptron

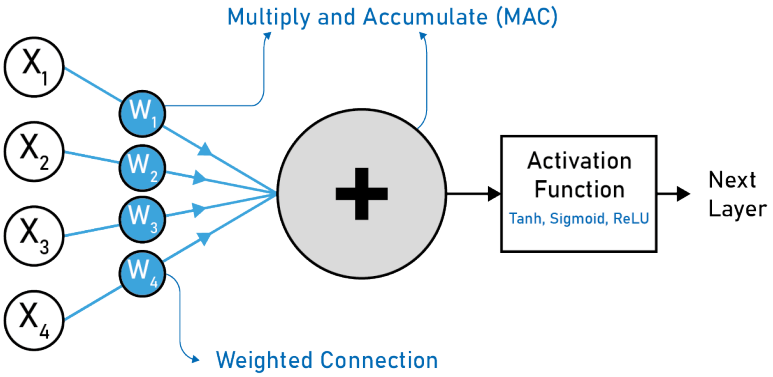
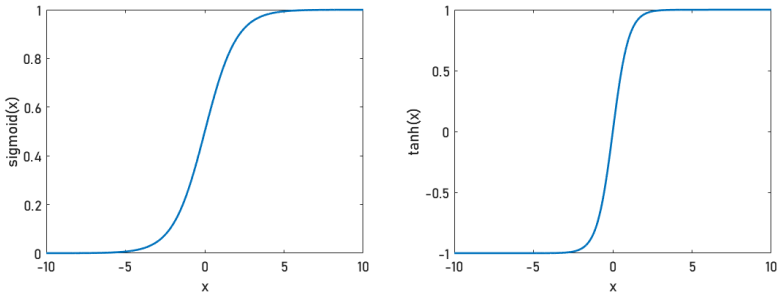


Figure 1.1: Perceptron

The perceptron is the basic neuron structure used in most neural networks. Figure 1.1 shows the basic structure of a 4-input perceptron. The input values X_1 to X_4 are first multiplied by the respective weights W_1 to W_4. These values are then accumulated before going through an activation function, often noted σ, that can be ReLU (equation 1.1), Sigmoid (Figure 1.2a), hyperbolic tangent (Figure 1.2b), or similar functions. Finally, the output is sent to the input of the next layer. The weights values are learned by the network to classify data.

$$ReLU(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{1.1}$$



(a) Sigmoid function **(b)** Tanh function

Figure 1.2: Activation Functions

1.2.2 MAC Operations

The main computation of a perceptron is called a MAC (Multiply and ACcumulate). Figure 1.1 presents a 4-MAC perceptron. This computation could also be presented as the multiplication of two vectors. In the case of a layer of multiple perceptrons in parallel, the computation consists of an input vector multiplied by a weights matrix. This operation is called Matrix-Vector Multiplication (MVM) and the main operator of an MVM is a MAC.

$$layer\ output\ vector = \sigma(\begin{bmatrix} X_1 & X_2 & X_3 & X_4 \end{bmatrix} \begin{bmatrix} W_{11} & \dots & W_{1n} \\ W_{21} & \dots & W_{2n} \\ W_{31} & \dots & W_{3n} \\ W_{41} & \dots & W_{4n} \end{bmatrix}) \tag{1.2}$$

This is the basic operation that we can find in nearly all neural networks, including Feed Forward, CNN, and LSTM.

1.3 Basic Neural Networks and Learning Principle

1.3.1 Feedforward

Feed-forward neural networks are composed of one input layer, one output layer, and multiple hidden layers. This neural network is one of the most basic and ancient models used to classify data [4]. Each layer itself is composed of neurons that are perceptrons. They are Fully Connected (FC), meaning that each neuron composing a layer is connected to all the neurons of the next layer but not to the neurons inside the layer. A larger number of hidden layers allow finer processing of the information, therefore the number of layers depends on the complexity of the application. The connections between the neurons are weighted and unidirectional, hence the name feed *forward*. Figure 1.3 presents an example of a two hidden layers feed-forward neural network with two outputs, that is able to discriminate input data between two classes.

The feed-forward neural network can be defined by the following equations:

$$z_{L1} = x \times W_{L1} + b_{L1} \tag{1.3}$$

$$\alpha_{L1} = \sigma(z_{L1}) \tag{1.4}$$

$$z_{L2} = \alpha_{L1} \times W_{L2} + b_{L2} \tag{1.5}$$

$$\alpha_{L2} = \sigma(z_{L2}) \tag{1.6}$$

$$z_{L3} = \alpha_{L2} \times W_{L3} + b_{L3} \tag{1.7}$$

$$\alpha_{L3} = \sigma(z_{L3}) \tag{1.8}$$

With x the inputs vector, W_{L*} the weight matrix of a layer (the weighted connection), and b_{L*} the biases that are added to the weights (not represented in Figure 1.3 for simplicity). z_{L*} is the result of the accumulation of each neuron and α_{L*} is the output of the neurons after going through an activation function. $L1$ is the first hidden layer, $L2$ is the second one and $L3$ is the output layer.

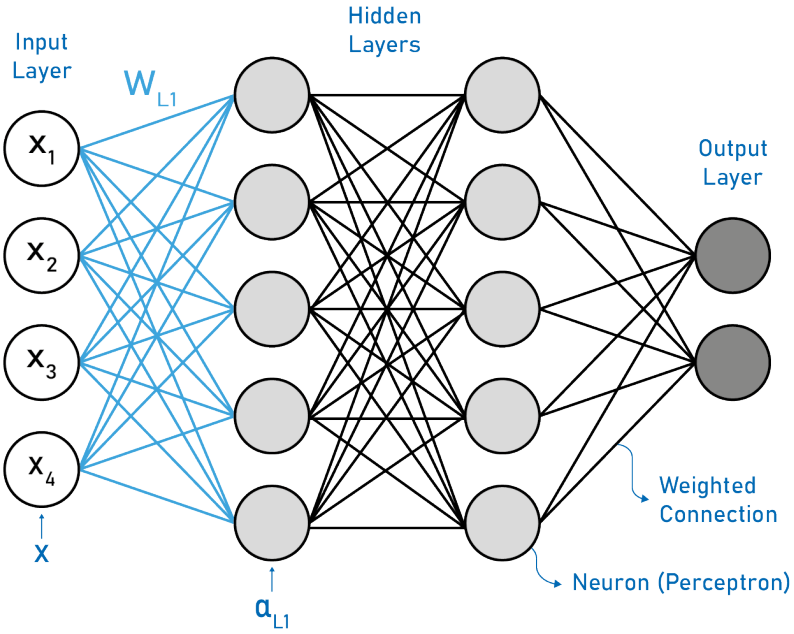


Figure 1.3: 2 Hidden Layers Feed Forward Neural Network

1.3.2 Backpropagation

Backpropagation algorithms are used to update the weights of a network [5, 6]. The principle is to define an error function and use gradient descent to find weights that optimize performance for a particular task which can be measured thanks to the error function. To update weight values, we will compute the gradient of the error with respect to the weight. Here is an example of a feed-forward neural network. We have an error function ϵ (that can be cross-entropy, for example, see Section 1.3.3), and we will compute the local gradient for W_{L3} thanks to the partial derivative:

$$\frac{\partial \epsilon}{\partial W_{L3}} = \frac{\partial \epsilon}{\partial \alpha_{L3}} \frac{\partial \alpha_{L3}}{\partial z_{L3}} \frac{\partial z_{L3}}{\partial W_{L3}} \tag{1.9}$$

The chain rule allows us to compute the gradient of the error with respect to the weight with a set of intermediary derivatives. This value is then subtracted from the actual weight value, scaled by a factor called the *learning rate* that will allow smoothing the weight variations:

$$W_{L3} = W_{L3} - \textit{learning rate} \times \frac{\partial \epsilon}{\partial W_{L3}} \tag{1.10}$$

By going backward and propagating the gradient through the network using partial derivatives and chain rule, we are able to update all the weights and biases of the network and minimize the error. This method is called the gradient descent algorithm.

1.3.3 Metrics to evaluate Neural Networks

To compare and evaluate different networks for a specific application, we need metrics. We already used a pretty obvious one, the classification accuracy. The following list presents other metrics that will help to understand how neural network works and how they learn.

Classification Accuracy

$$accuracy = \frac{Number\ of\ correct\ predictions}{Number\ of\ predictions} \tag{1.11}$$

The classification accuracy is relevant only if there is an equal number of samples in each class for the training dataset.

Logarithmic Loss

Logarithmic Loss works well for multi-class prediction. According to the training set distribution, each sample is given a probability to belong to a class. The Logarithmic Loss is given by

$$loss = \frac{-1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} \times \log p_{ij} \tag{1.12}$$

where N is the number of samples and M is the number of classes. y_{ij} indicates if the sample i belongs to class j or not, and p_{ij} indicates the probability i belongs to class j. The logarithmic loss has a range $[0, \infty)$. A value close to zero indicates a high accuracy.

Mean Absolute Error (MAE)

The mean absolute error indicates how far the prediction is from the actual target. It is the average difference between the target value and the predicted one. However, there is no information about whether we under or over-predict the data. The MAE is defined as

$$MAE = \frac{1}{N} \sum_{j=1}^{N} |y_i - \hat{y}_i| \tag{1.13}$$

where N is the number of samples, y_i is the target value and $\hat{y}_i$ is the predicted value.

Mean Squared Error (MSE)

The mean squared error is quite similar to MAE, the only difference is that this metric uses the square of the difference between the actual value and the predicted value:

$$MSE = \frac{1}{N} \sum_{j=1}^{N} (y_i - \hat{y}_i)^2 \tag{1.14}$$

Cross-Entropy

The cross-entropy function is useful as a loss function because the loss is high for bad prediction and close to zero for good prediction. It helps the gradient descent algorithm used in the backpropagation process to converge toward accurate precision. It is given by

$$X_{entropy} = -(y \log(p) + (1 - y) \log(1 - p)) \tag{1.15}$$

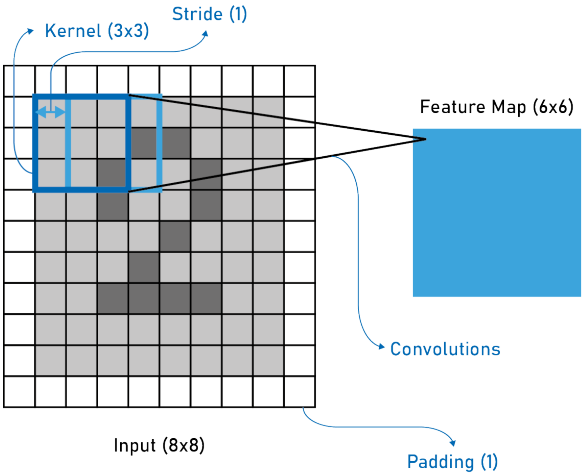
where y is a binary indicator (0 or 1) if the class label is the correct classification for observation and p is the predicted probability observation.

Number of operations

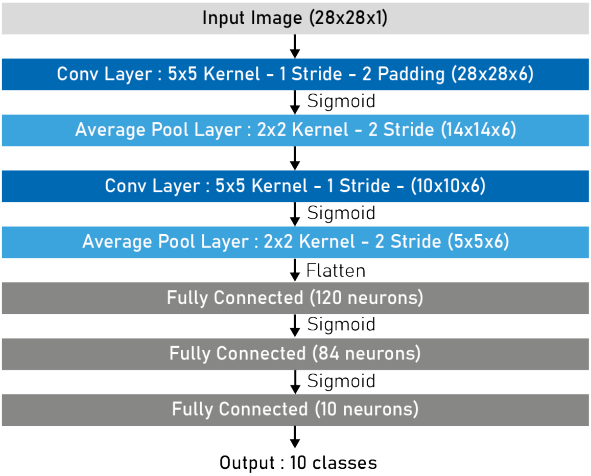
The number of operations needed to classify an input is a comparison point of the size and complexity of the neural network classifying the same data with equivalent accuracy. It is to note that the bit width of the operations needs to be associated with this metric. Performing a binary multiplication is not equivalent to an 8-bit multiplication. An important metric for the efficiency of a system is Tera Operation Per Second Per Watts (TOPS/W), it is noted that a MAC is counted as two operations, one multiplication, and one addition. It is a more hardware-oriented metric but still a good indicator of the complexity of a system.

1.4 Convolutional Neural Network

In 1989, LeCun [5] presented a Convolutional Neural Network (CNN) trained thanks to backpropagation and gradient descent algorithms. This network was able to classify handwritten zip-code digits from the MNIST dataset (See Section 1.6.3). As suggested by the name, CNN uses convolutional layers (Conv Layer). The parameters of a Conv Layer are learnable filters (Kernels) that have a small receptive field but that are convolved across the input dimensions. Figure 1.4a shows an example of the convolution process. The 3 by 3 kernel convolves the input by moving the kernel by one ”pixel” at a time which is called the stride value. The input value is sometimes padded to allow the kernel to have the input's edge pixel centered. The resulting feature maps are 6 by 6 and are used as the input of the next convolutional layers. In addition to conv layers, CNNs often use pooling layers that are used to



(a) Example of a convolution



(b) LeNet CNN

Figure 1.4: CNN.

reduce the data dimensions. The two common poolings are max pooling and average pooling that are outputting the max or average value of a small cluster of the feature maps. Figure 1.4b presents LeNet, the network used by LeCun to recognize hand-written digits. The last layers of the networks are 3 Fully Connected feed-forward neural network layers. The output classes are digits from 0 to 9. This type of network is well suited to classify images and find specific objects or people in an image.

1.5 Long Short-Term Memory

Long Short-Term Memories (LSTM) are recurrent neural networks. The output of the inference is used as an input in the next inference. They were introduced in 1997 by Hochreiter and Schmidhuber [7]. They perform well on sequential data thanks to their capacity to store information in time. LSTMs are composed of intermediate sets of feed-forward neural networks called *gates*: the input gates i_t, the forget gates f_t, the candidate gate g_t, and the output gate o_t. These gates are arranged according to the schematic of Figure 1.5. The new input data and the last hidden vector of the network are fed to the gates, and the outputs of those gates are combined to form the state vector c_t and hidden vector h_t. The equations of the LSTM are as follows:

$$f_t = \sigma_s(W_f x_t + R_f h_{t-1} + b_f) \tag{1.16}$$

$$i_t = \sigma_s(W_i x_t + R_i h_{t-1} + b_i) \tag{1.17}$$

$$o_t = \sigma_s(W_o x_t + R_o h_{t-1} + b_o) \tag{1.18}$$

$$g_t = \sigma_h(W_g x_t + R_g h_{t-1} + b_g) \tag{1.19}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ g_t \tag{1.20}$$

$$h_t = o_t \circ \sigma_h(c_t) \tag{1.21}$$

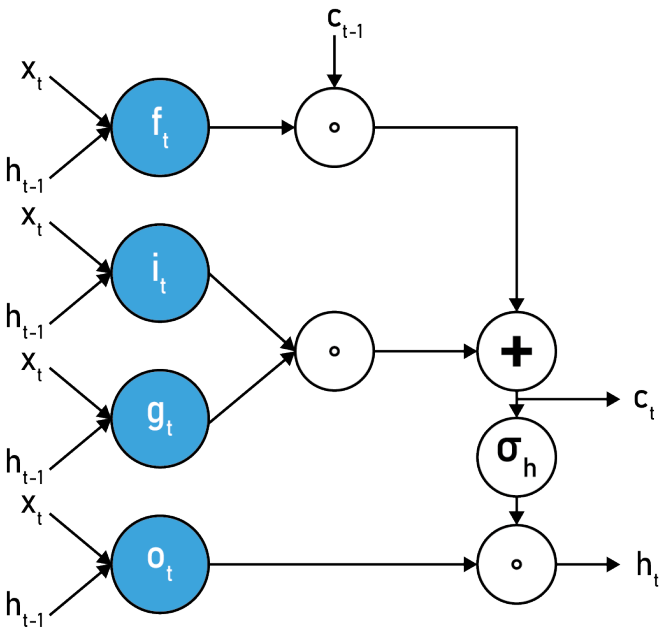


Figure 1.5: LSTM Neural Network

where W_* and R_* are weight matrices for each gate and b_* biasing values that are obtained by training the neural networks. σ_s and σ_h are sigmoid and hyperbolic tangent activation functions respectively. $\circ$ indicates an element-wise multiplication. The states and hidden vectors allow storing information in time at each inference, giving LSTM the faculty to remember some information that will influence the output. In many cases, the LSTM is followed by a Feed Forward and a Softmax layer. The Feed Forward allows classifying the data thanks to the hidden vector of the LSTM, and the Softmax layer computes the probability of the input data belonging to a certain class. Figure 1.6 presents a schematic of this network. Increasing the number of hidden units (increasing the length of the internal vectors) generally increases the accuracy of the NN. Reducing this number allows a shal-

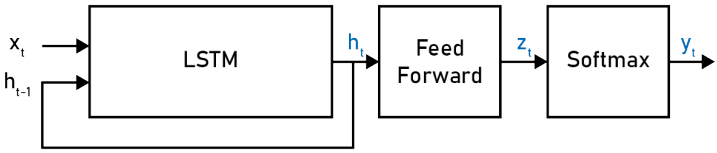


Figure 1.6: LSTM Neural Network With Feed Forward and Softmax Layers

lower network which is easier to implement on hardware. The number of hidden units is chosen by keeping in mind the balance between the number of operations and the accuracy which is mainly constrained by the application.

1.6 Datasets

Multiple datasets emerge and are used to benchmark neural networks on different tasks, such as keyword spotting, image recognition and handwritten digit recognition. This section presents a list of common datasets that are used in state-of-the-art works.

1.6.1 Google Speech Command Dataset

This dataset [8] is composed of 65,000 one-second long utterances of 30 short words pronounced by 1,000 different peoples. The number of keywords varies but one common task to perform using this dataset is to classify 10 keywords, unknown and background noise.

1.6.2 TIMIT

The TIMIT [9] corpus of read speech is designed to provide speech data for acoustic-phonetic studies. The dataset contains recordings of 630 speakers of eight major dialects of American English, each reading ten phonetically rich sentences. The TIMIT corpus includes time-aligned orthographic, phonetic, and word transcriptions as well as a 16-bit, 16 kHz speech waveform file for each utterance. It is used for Voice Activity Detection or for Keyword Spotting.

1.6.3 MNIST

The Modified National Institute of Standards and Technology database is composed of 60,000 training images and 10,000 testing images of handwritten digits (see Figure 1.7. This dataset is used to compare neural networks on simple image classification applications [10].

Figure 1.7: MNIST Dataset Example

1.6.4 CIFAR

The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. CIFAR-100 is the same as CIFAR-10 except it has 100 classes containing 600 images each.

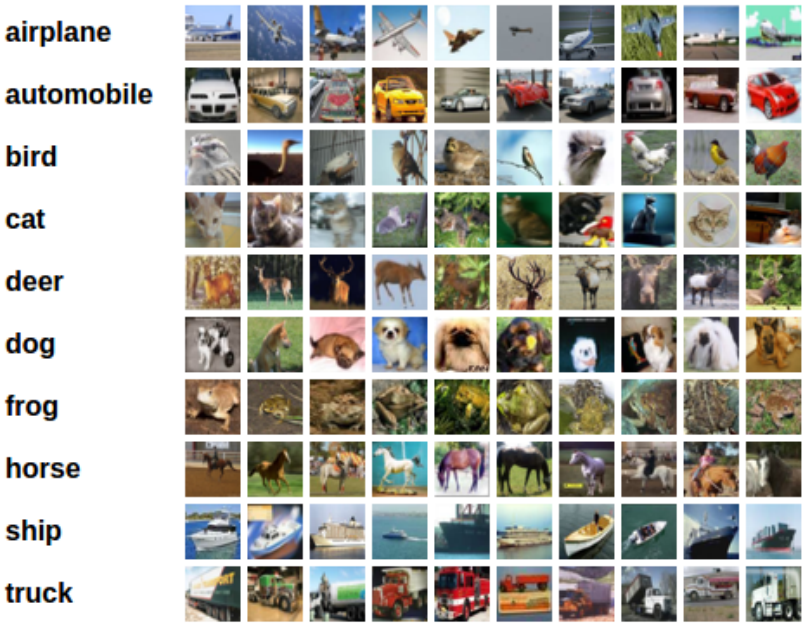


Figure 1.8: CIFAR10 Dataset Example

1.6.5 ImageNet

ImageNet is a dataset of millions of images that are classified by nouns, there are more than 20,000 classes in 2022. This dataset, regularly updated, is a reference for large neural network image classification. AlexNet [1] was the first to use a GPU during training and reached 15.3% top-5 error rates among 1,000 classes, meaning that 84.7% of the time, the right class is in the top-5 predictions on an image.

1.7 Conclusion

Artificial Intelligence and especially Machine Learning allow Neural Networks to predict and classify data in a wide range of applications. With a training dataset, the networks are able to *learn* how to classify data thanks to backpropagation and gradient descent algorithms. Different architectures, like CNNs, are well suited to classify images. LSTMs are well suited to classify sequential data like audio. Simple networks, like feed-forward NNs, are often used inside complex architectures to perform dimension reduction. Although built with different architectures, they are all based on Matrix-Vector Multiplication. The MAC operations performed to compute MVM constitute the foundational blocks of such structures. In the next chapter, we will introduce the Tiny Machine Learning environment and explore new ways to compute MAC operations.

Chapter 2

Embedded Machine Learning for Audio Applications

2.1 Introduction

Neural networks are able to classify data and make predictions in a wide variety of application fields. Recent developments in IoT require operating these neural networks on battery-constrained and computation-limited devices. Therefore, artificial intelligence on embedded devices relies on connectivity, local computing, or a combination of both to perform prediction. The neural network algorithms can be run on three different environments:

- The **Cloud** environment is constituted of data centers that are distributed across the world to store and compute information. They have computational power allowing them to run resource-demanding AI models. These servers are equipped with Graphic Processor Units (GPU) or specific hardware to accelerate the training of AI algorithms. They can be used to evaluate large image datasets or to detect objects in images at a rate suitable for video applications, for example. They are used to run and train algorithms.

- The **On-Device** environment represents any phone, tablet, or IoT device that runs on batteries. They are able to run and train medium-sized models, thanks to robust Central Processing Units (CPU) or specific hardware accelerators used to compute tasks locally, such as Augmented Reality (AR) or typing prediction. The models operating on these devices need to be chosen according to user usage and battery capacity.

- The **Tiny Machine Leaning** (TinyML) environment includes ultra-low consumption application-specific integrated circuits. They perform only one task and are designed to be deployed close to the sensor. It aims at specific application fields that require low maintenance and long battery life. Dedicated chips

for specific applications allow for fine-tuned architectures which in turn allow for ultra-low consumption. These models can be used to perform sound/noise detection, voice activity detection, and keyword spotting and can also be used in health applications for electrocardiogram (ECG) monitoring and heart arrhythmia detection. TinyML is used for inference only, the training of the model is done on a CPU/GPU and transferred to the chip when reaching suitable accuracy.

This work addresses the Tiny Machine Learning environment, specifically studying ultra-low power integrated circuits. This new trend has not been deployed yet in an industrial context, but already presents promising results in terms of consumption and accuracy. On the one hand, the number of operations performed in a neural network needs to be reduced, without sacrificing accuracy. On the other hand, each operation itself needs to be energy efficient. To address this matter, in the following pages, we will:

- Present the advantages of ultra-low power integrated circuits and TinyML as a preprocessing unit.

- Explore how neural networks are implemented on chips and how they compare in terms of accuracy and power consumption.

- Introduce the memory-wall bottleneck and explore new ways of implementing efficient MAC operations to overcome it.

The demonstration will be based on audio applications, as many devices are now voice-activated [11]. This example offers a great overview of a complex system relying on on-device computation and connectivity (4G/5G, WiFi, etc.) to send data to the cloud for complex computations.

2.2 Hierarchical Architectures

2.2.1 Preprocessing Unit

In an architecture composed of multiple sensors and one processor, the processor always needs to be on to process the input signals. However, the main processor is consuming a lot of energy. Figure 2.1 illustrates this example with an audio signal as an input. The processor needs to digitize the signal and send the data to the cloud to be processed and classified on algorithms running inside data centers.

The neural network could be implemented on the processor. However, this method is not efficient and still consumes a lot of energy, as a large amount of data need to be moved between memory and processor (more details in Section 2.5). A solution to improve the efficiency of a system is using a preprocessing unit that will process and classify the data locally, as seen in Figure 2.2. The main processor is in standby

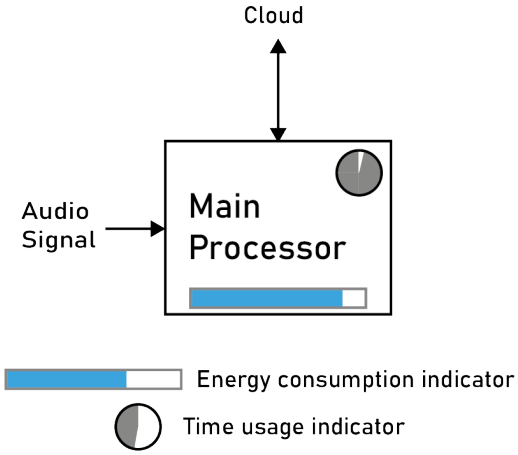


Figure 2.1: Classic architecture, consuming a lot of energy.

mode most of the time, while the preprocessing unit is always on. It exploits a specific architecture for feature extraction and classification. When the preprocessing unit detects or classifies a relevant pattern on the input data, the main processor is woken up to take over. A classic RISC-V processor embeds 47 instructions, neural networks need only two: multiplication and accumulation. We can therefore leverage novel solutions to perform MAC operations at high efficiency.

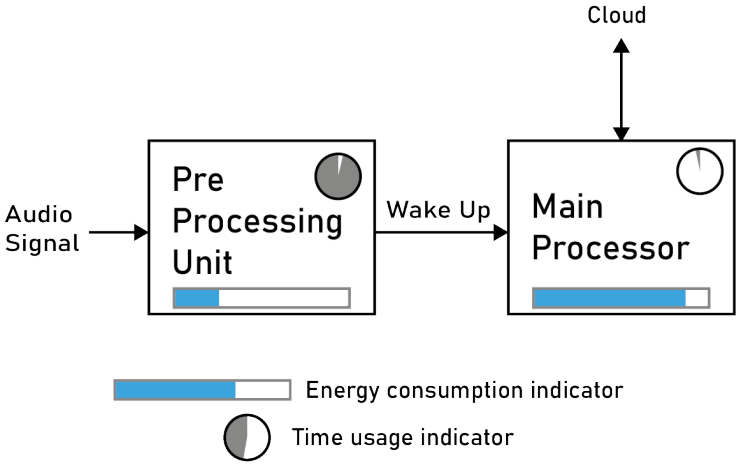


Figure 2.2: Preprocessing unit architecture.

2.2.2 An example of a vocal assistant

"OK Google" (Google) [12], "Hey Siri" (Apple), and "Alexa" (Amazon) are common words to be heard when using smart speakers, phones, or watches. They are able to activate only when one of these keywords is detected and then act accordingly to the user demand. Such a system always needs to be on to detect the activation keywords; therefore the energy consumption needs to be low, especially when implemented on energy-constrained devices. In addition, the system needs to be kept

simple to reduce the manufacturing cost, and it relies on connectivity to transfer complex actions to the cloud, keeping simpler tasks on the device and using a hierarchical architecture to distribute the computation load and reduce the overall energy consumption. The idea behind this architecture is to split the systems into different tasks that can be power gated. Figure 2.3 presents a vocal assistant architecture example implementing such a power-gating scheme. The preprocessing unit is composed of the Voice Activity Detection (VAD) block that will detect if a voice is present in the input signal. It will wake up the keyword spotting (KWS) block that is trained to detect specific keywords in an audio signal. This block will activate the speaker verification (SV) block which then turns on the main processor if the voice in the signal is from an authorized user. If the two conditions are met — an activation keyword has been pronounced and the voice comes from an authorized user — the remaining sentence will be sent to the cloud to be computed. The algorithm running in the data center performs Automatic Speech Recognition (ASR) to transcribe the audio in a sentence, then another algorithm identifies the actions needed to be taken and returns the information to the device. The whole process takes a couple of seconds to happen, thanks to the computational power of the data center. However, the energy consumption of our devices is deeply impacted by the data transmission to the cloud, therefore reducing the amount of information sent to the cloud is crucial to lowering global energy consumption.

In such a configuration, the role of the VAD is preponderant as it is responsible for the activation of the most consuming tasks of the system. Accordingly, the reduction of the consumption of the VAD needs to be performed without impacting the accuracy. In [13], M. Price introduces the following equation to model the averaged system power:

$$P_{AVG} = p_{VAD} + [(1 - p_M)D + p_{FA}(1 - D)]p_{downstream}$$

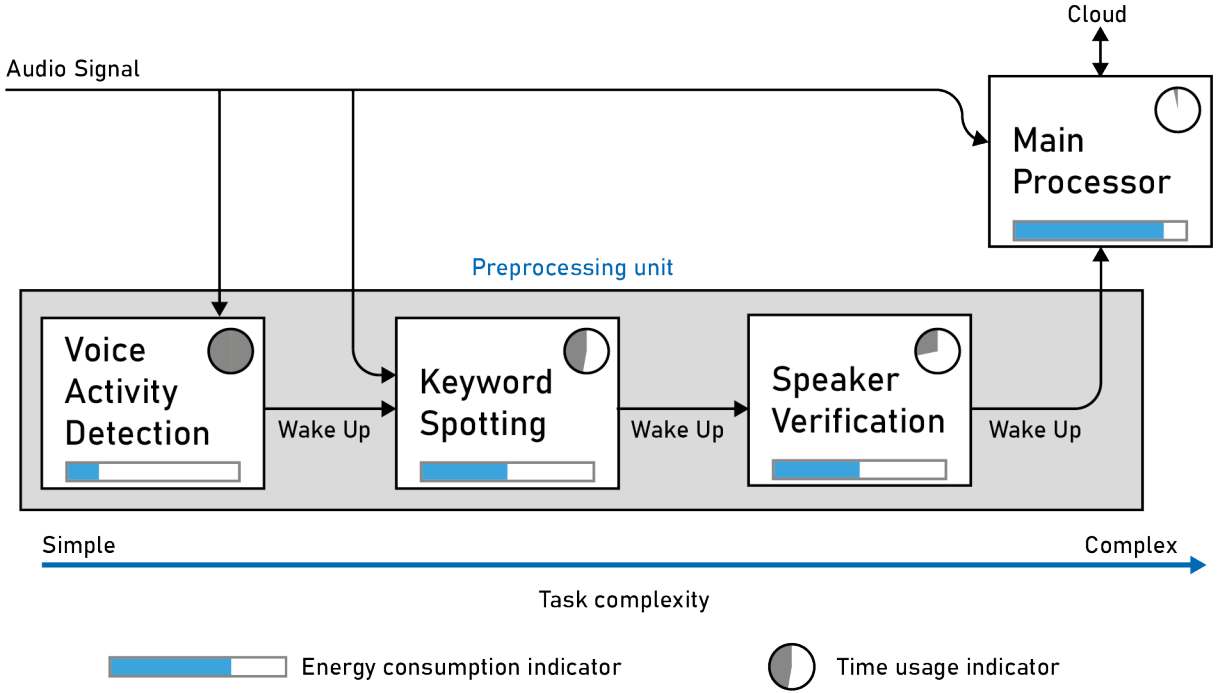


Figure 2.3: Example of a Vocal Assistant Architecture.

where D is the duty cycle of speech, p_M is the probability of misses, p_{FA} is the probability of false alarm, p_{VAD} is the power consumption of the VAD and $p_{downstream}$ the power of the bloc activated by the VAD. In this paper the author adds:

> "The coefficient $[(1-p_M)D+p_{FA}(1-D)]$ reflects how often the downstream system is enabled—a duty cycle that can be far higher than D if P_{FA} is significant. Differences in this contribution from the downstream system can far exceed the differences in power consumption between different VAD implementations."

For example, a reduction of the consumption of the VAD block by a factor of 2 leads to a reduction of the average power by only 18%, keeping the accuracy identical and a speech duty cycle of 60%. As we will see in the next section, VAD consumption decreased throughout the years thanks to neural networks, but if the false alarm rate is high, then the average energy consumption can still be high. Therefore, it is important to reduce the consumption and keep the accuracy constant but other parts of the system should not be neglected as they have a high impact on the average power consumption.

2.3 Detailed composition of a vocal assistant

2.3.1 Voice Activity Detection

Usage of VAD

Voice activity detectors detect if there is a voice or silence in a signal that can be disturbed by background noise. It is mainly used to extract additive noise characteristics on a signal to then be able to subtract it. It was first used in telecommunication for Discontinuous Transmission

(DTX), by detecting if the signal contains silence or voice. Two different coding algorithms were used: active speech codec and silence suppression. It helped to reduce the average bit rate by compressing the silences in the conversation. In addition, as we saw in the last section, VAD is often used as a power gating block inside more complex systems.

How it works

Voice activity detection algorithm uses decision trees, decision rules, or neural network algorithms. Table 2.1 compares different chips performing VAD and highlights the performance of neural networks in both accuracy and power efficiency for this task. Features need to be extracted from the raw audio signal to be computed and understood by the algorithm. We can discriminate these chips with their feature extraction method and their classifier model.

Some references [14–16] use decision trees or decision rules models to classify the audio signal as containing voice or not. The features used by this type of classifier are either noise estimation, computed in the digital domain, or signal energy in selected frequency region, computed in the analog domain. Decision rules work by comparing the chosen features to thresholds that are fixed or updated according to external parameters. In a decision tree, there are multiple comparisons of different features to thresholds.

The remaining references in Table 2.1, use deep neural networks (DNN) to classify the audio signal, meaning Feed Forward neural networks with a high number of hidden layers. They use features like Mel Frequency Cepstral Coefficient (MFCC) [13], spikes [17, 18] and Sequential Frequency Scanning [19]. The MFCC is performed in the digital domain, the audio signal is converted using an ADC before going through digital processing, and the ADC sampling frequency is set between 16 kHz and 48 kHz. Fast Fourier Transform (FFT) and Discrete Cosine Trans-

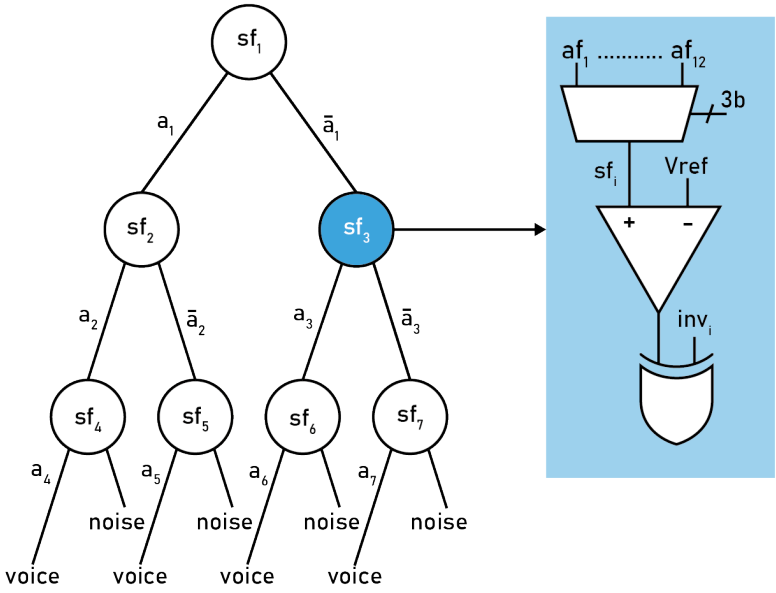


Figure 2.4: An example of a decision tree from [15].

form (DCT) are used to compute MFCC. Digital Feature Extraction (FE) represents a large part of the total consumption of a system, more than 50% of the total consumption in [14]. To reduce the consumption of the feature extraction block, other methods are proposed to perform the feature extraction in the analog domain, for the most part, using analog filters and rectifiers allowing using an ADC at the end of the processing line, requiring lower sampling frequency because it then depends on the input data latency, improving the efficiency of the system. Other methods use spikes that are created with Integrate And Fire (IAF) mechanism. The signal goes through a band-pass filter

(BPF), then through a Full-Wave Rectifier (FWR) before integration. When the resulting voltage crosses a threshold a spike is produced as an input to the deep neural network. The IAF plays the role of an event-driven ADC here, which allows for reduced consumption.

Power consumption and number of operations

The references relying on analog feature extraction consume far less energy than the ones using digitally extracted features. However, unlike analog feature extractors, digital one allows reconfiguring the systems as needed, which can motivate their use in some applications despite higher energy consumption. The mean accuracy is around 90 % speech rate at 10 dB SNR. DNN seems to allow for better efficiency, regarding accuracy vs. power consumption. Reference [16], which uses a decision rule model, reaches 99.5 % accuracy at 0.76 μW, the result can't be compared precisely with the other references since SNR is not indicated. Nonetheless, references using DNN clearly detached from the other works reaching $0.115\mu W$ [18] in 65 nm, and $0.142\mu W$ [19] in 180 nm technology node. This comparison seems to indicate the advantages to use deep neural networks to perform VAD, reaching high accuracy in a noisy environment and consuming a low amount of energy. The number of operations is low for the decision rules model as it consists mainly of comparisons, counted as one operation. The number of operations does not seem to be correlated with the accuracy. We can observe that two binarized DNNs presented by the same authors in 2019 and 2021 [17, 18] have nearly divided by 2 the number of operations with increased accuracy, which comes mainly from improved feature extraction and training.

Reference	Raychowdhury, JSSC'13 [14]	Badami, JSSC'16 [15]	Croce, CICC'20 [16]	Price, JSSC'18 [13]	Yang, JSSC'19 [17]	Yang, JSSC'21 [18]	Oh, JSSC'19 [19]
Technology	32nm	90nm	180nm	65nm	180nm	65nm	180nm
Area	-	$3mm^2$	$0.14mm^2$	$2.08mm^2$	$2.52mm^2$	$0.9mm^2$	$17.5mm^2$
Feature Extraction	Digital	Analog	Analog	Digital	Analog	Analog	Mixed-Signal
Feature Extraction Building Blocks	FFT, Mult/Filtr, Noise Estimation	LNA, BPF, FWR, LPF	BPF, PGA, Square Integrator	FFT, Log	LNA, BPF, FWR, IAF	LNA, BPF, CLIPA HWR, IAF	LNA, Mixer, LPF, DSP
Features	Noise Estimation	Average Signal	Signal Energy	MFCC	Spike	Spike	Sequential Frequency Scanning
Classifier Model	Digital Decision rule	Mixed-signal Decision Tree	Analog Decision Rule	Digital Fixed-Point DNN	Digital Binarized DNN	Digital Binarized DNN	Digital DNN
Dataset	-	NOISEUS	-	Aurora 2	Aurora4 + DEMAND	Aurora4 + DEMAND	LibriSpeech + NOISEX-92
Accuracy	97% Unspecified SNR	HR Speech 89% HR Non-Speech 85% 12dB SNR	99.5% Unspecified SNR	10% EER 7dB SNR	HR Speech 84% HR Non-Speech 85% 10dB SNR	HR Speech 90.9% HR Non-Speech 90.7% 10dB SNR	HR Speech 91.5% HR Non-Speech 90% 10dB SNR
Latency class/s	32,600	-	31.25	100	100	100	1.95
Power (FE + Classifier)	300μ W	6μ W	0.76μ W	8.5μ W	1μ W	0.115μ W	0.142μ W
OP/Classification	9	7	1	10,800	4606	2368	1568

Table 2.1: Comparison of VAD integrated circuits

2.3.2 Keyword Spotting

Usage of KWS

To be able to respond only when the user addresses them, voice assistants are activated thanks to a special keyword, such as ”OK Google”, ”Alexa” or ”Hey Siri”. The Keyword Spotting (KWS) block is trained to detect if the activation keyword is pronounced.

How it works

Table 2.2 compares multiple chips performing KWS. All the references presented in this table are using keywords from the Google Speech Command Dataset (GSCD) [8]. The number of spotted keywords varies greatly from one reference to another and some models are also trained to classify 10 keywords plus unknown words and background noise, resulting in 12 output classes. The main feature extracted for KWS is MFCC, which requires heavy computation: i) FFT, ii) Mel Filtering, iii) Compute Log, iv) DCT, it takes 50% and 66% of the total power consumption for [20] and [21] respectively. Unlike VAD, KWS is exclusively performed with neural networks, there are two types of neural networks used in Table 2.2: CNN and LSTM.

Power consumption and number of operation

It is to note that the power includes the contribution from the feature extraction block, except for [2, 22]. Reference [23] uses spike-based IROCs (Instant Rate Of Change) as features, that seem to allow for reduced consumption compared to [21] despite a gain brought by the technology node of 180 nm and 28 nm respectively for accuracy of 94 % and 97.3 %. [20] and [24] share the same amount of output classes, the first one is using an LSTM neural network in 65 nm tech node and the latest is using a CNN in 22 nm. In addition, [20] is not only

Reference	Schaefer, ISCAS'21 [22]	Giraldo, ISSCC'20 [20]	Wang, ISSCC'21 [23]	Shan, ISSCC'20 [21]	Bernardo, TCAD'20 [2]	Liu, TCASI'20 [24]
Technology	32nm (Simulation)	65nm	180nm	28nm	22nm	22nm
Area	-	$2.56mm^2$	-	$0.23mm^2$	$0.2mm^2$	$0.6mm^2$
Feature Extraction	MFCC (software)	MFCC	Spike-based IROC	MFCC	MFCC (software)	MFCC
Task	KWS	VAD-KWS-SV	General Purpose Wake Up	KWS	KWS	KWS
Latency	20ms	16ms	348ms	64ms	100ms	16ms
Model	LSTM	LSTM	CNN	DSCNN	TC-ResNet (CNN)	CNN
Bitwidth	5 - 6 bit	8 bit	8 bit	Binary	6 - 8 bit	16 bit - Binary
Accuracy	90%	90.87%	94%	97.3% / 94.6%	93.09%	87.9%
Number of classes	12	10-12	1	1/2	10	10
Power	-	10.6μ W	0.378μ W	0.51μ W	8.2μ W (FE not included)	15.1μ W
OP/Classification	131,072	47,432	350,020	47,232	2,480,000	271,080
OP/s	6.5×10^6	2.9×10^6	1×10^6	0.7×10^6	24×10^6	16×10^6

Table 2.2: Comparison of KWS integrated circuits

performing KWS but also VAD and SV, implementing a hierarchical architecture as we saw earlier at $10.6\mu W$ with 8-bit weights. Overall, the LSTM network, with 4 to 6 times less operation per second using 6 to 8-bit weights can reach similar accuracy to CNN. The latency is smaller for the LSTM as we run the model for every new audio frame, which is not the case for CNN where multiple frames are buffered to compute the sample. In reality, the computation might take less time than the latency displayed in the table as this value is set by the input data frequency. As it is usual for digital processors to reach high throughput, in the case of TinyML, we can leverage the latency to explore new ways of performing MAC operations and to take advantage of this available time. This table shows us that LSTM seems to offer a great opportunity to reduce consumption of KWS, as they use the least amount of operation for the same accuracy. However, this neural network requires a non-linear activation function that can impact this efficiency.

2.4 Speaker Verification and Automatic Speech Recognition

2.4.1 Speaker Verification

In Figure 2.3, Speaker Verification (SV) is activated by the VAD alongside the KWS. The role of this block is to detect if the user pronouncing the activation keyword is an authorized user. As it is possible to use our voice to perform sensitive tasks (online shopping, event booking, accessing private data) this security feature allows only the main user to perform this demand. [20] implements an SV algorithm in their chip thanks to a Gaussian Mixture Model (GMM) which according to the author:

> "offers the lowest computational load while maintaining high accuracy for typical recognition context."

GMM is a probabilistic model generated by a sum of multivariate Gaussians, which computes the probability of a feature vector belonging to a specific class depending on the set of trained parameters. This value is computed with a Universal Background Model (UBM) trained with several speakers, which allows them to model the average user. GMM and UBM are computed together and the difference between the two models is used to classify if the speaker is the targeted speaker.

2.4.2 Automatic Speech Recognition

ASR is an algorithm able to transcribe complete sentences from audio. These highly complex models are performed in the cloud as it would consume too much power to compute them locally. [13] implements an on-chip ASR, reaching 7.78 mW for 145k vocabulary using a Hidden Markov Model (HMM) and Viterbi search algorithm which is able to approximate the likelihood of the next word in a sentence. The memory bandwidth of such a network is 15 MB/s and therefore is not suited for low-cost, low-power embedded devices but is easily achievable on the cloud.

2.4.3 Conclusion

In this section, we have shown that it is possible to achieve the same classification with different architecture. Although we've shown a reduced number of operations for LSTM neural network in the case of KWS. This conclusion might not be the same for other, more complex, applications. Therefore, choosing the right neural network is one path toward dedicated low-consumption ASICs but not the only one. In the

next section, the optimization of neural networks to target low-energy systems is presented. The software and hardware approaches are shown to be essential for reducing the number of operations and reducing the consumption of one operation.

2.5 Optimization of Neural Network

Equation 2.1, introduced by Murmann in [25], shows the impact of two factors.

$$Power = Rate \times \frac{Energy}{Inference} = Rate \times \frac{Operation}{Inference} \times \frac{Energy}{Operation} \quad (2.1)$$

The number of operations per inference corresponds to the number of MAC needed per inference. It is possible to reduce this number by optimizing the model, compressing the weights, or reducing the bit width by quantizing the network or pruning some elements. Another alternative option is to reduce the energy per operation. It means using an energy-efficient way to perform a MAC operation. In this section, we will review how an LSTM complexity can be reduced by optimizing the model and introduce the "Memory Wall" principle.

2.5.1 Reducing the number of operations

As seen earlier, the neural network is always preceded by a Feature Extraction block. The role of this block is to transform the input into a set of features used by the neural network. The feature extraction block usually represents nearly half of the total consumption of a system. In Appendix A, we propose a KWS solution that doesn't use energy-intensive feature extraction with a quantized LSTM on 8-9 bits [26]. This work was presented at the 28th International Conference on Electronics, Circuits, and Systems (ICECS) in 2021. We reached an 89.45% accuracy on 10 keywords, silence, and unknown classifica-

tion using post-quantization techniques and a filter bank for the feature extraction. By comparing the different filter scales we highlighted that Log, Mel, and Bark scales are all suitable. However, the number of bands has an impact but a filter with more than 16 bands shows no substantial increase in accuracy. In addition, our 64 hidden units LSTM with quantized weights compares well with the state-of-the-art using a simple quantization method that doesn't require to access the internal layer of the LSTM.

Other techniques are used to reduce the number of parameters and therefore the number of operations of neural networks:

- **Pruning** techniques [27] reduce the size of a neural network by removing parameters. The algorithm [28] consists of first training the neural network to convergence. Then each parameter is given a score and based on its score the weights are pruned.

- **Quantization Aware training** (QAT) techniques quantized the weights during training to reduce the complexity of neural networks [29]. [30] shows 3-bit weights on ResNet152 [31] with the same training accuracy as full-precision one and [32] shows 3 and 4-bit weights on MobileNetV2 and MobileNetV3 [33] trained on ImageNet without significant accuracy loss.

- **Post Training Quantization** solutions quantized the weights of fully trained neural networks. Although QAT performs better, there are specific cases where it is not possible to perform QAT, such as in the medical field where the dataset is sensitive and not available when training. [34–36] shows post-quantization techniques that allow for reducing the weight precision and show less than 1% accuracy loss on ResNet50 with 4-bit weights.

- **Binarization** techniques allow reducing the weights of neural networks to 1-bit but offer a tradeoff with extra memory and performance [37]. However, they offer opportunities, especially for digital implementation as seen in [17, 18] for high-efficiency ASICs.

As the software approach is useful to train smaller networks, with fewer parameters and less complex activation functions keeping the accuracy level high. The hardware approach will allow reducing the energy consumption of one operation, theoretically without affecting the accuracy, which will be analyzed in the next sections. Being able to implement configurable multi-bit neural networks is required to fit a large array of neural network architecture targetting a wide range of applications.

2.5.2 Reducing the consumption of one operation

In a classic "Von Neumann" architecture, it is necessary to move the data stored to do the computation step by step. In [25], the example of Figure 2.5 is presented. If we want to perform a MAC, we need to move the data 4 times. If we consider energy for the access around 50 fJ/byte, this is equivalent to 200 fJ/MAC, topping our maximum system efficiency at 10TOPS/W (1 MAC is counted as 2 operations).

This issue is called the "memory wall" because the consumption of our system is topped by the memory access cost (a limitation regarding access time is also identified [38]). However, it is possible to jump over the memory wall by introducing In-memory Computing (IMC) methods. These techniques allow to perform computations inside the memory and therefore, achieve lower energy consumption thanks to reduced memory access. Figure 2.6 presents the advantages of an IMC

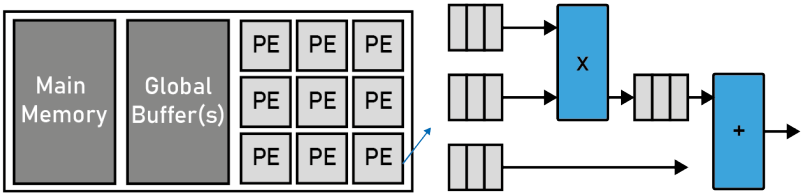


Figure 2.5: Von Neumann Architecture

architecture. Compared to a conventional all-digital implementation, which requires a huge amount of data transfer to/from the memory, the IMC implementation performs the computation inside the memory and therefore reduces greatly the access cost.

2.6 Conclusion

Hierarchical architectures are a great way to reduce the average energy consumption of a system by conditionally activating successive blocks so that they are powered only when relevant to the final task. However, in such a configuration, the accuracy of the always-on part can greatly impact the average power and therefore needs to be balanced with the power consumption of this element. We highlight the impact of the neural networks classifier for VAD and KWS and how the chosen feature extraction methods can have an impact on power consumption. The classical "Von Neumann" architecture prevents us from reaching high efficiency because of the data access cost. In the next chapter, a novel approach will be presented to overcome the memory wall and reach higher efficiency by computing the data inside the memory.

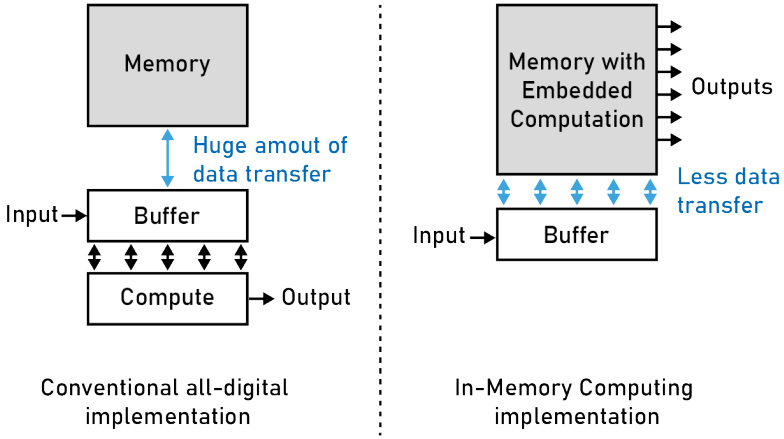


Figure 2.6: Comparison of digital implementation versus In-memory implementation from [39].

Chapter 3

In-Memory Matrix-Vector Multiplication

3.1 Introduction

The high energy consumption associated with memory access prevents classic digital systems to embed neural network architecture on battery-constrained devices. Although it is possible to reduce the number of operations needed by a neural network thanks to advanced quantization and optimization techniques (see Section 2.5.1) it is possible to reach higher energy efficiency thanks to hardware optimization to perform multi-bit MAC operation required to fit the inference of a wide range of neural networks.

In-memory computing is a great approach to reducing the energy consumption of a MAC operation. Solutions are using classic SRAM architectures with only slight modifications on the bitcell in search of high-density integration. Other works developed specific macros with custom processing elements that embed their own local memory. Using a digital or a mixed-signal approach the terminology used in the literature are "In-Memory Computing", "Compute In-Memory", "Process In-Memory", etc. In this document, the term "In-Memory Computing" is used without difference between a dense SRAM or the macro approaches.

This chapter aims to present the range of available In-Memory Computing methods and highlight their advantages and disadvantages with respect to their target application and neural network architectures. Therefore we will review:

- The macros that use digital in-memory computing architecture and highlight the constraints associated with higher precision parameters (input and weights $> 4\ 5 - bit$).

- The macros using Non-Volatile Memory (NVM) including RRAM, STTM-RAM, and Fe-FET that try to increase their GOPS/mm^2 metrics and mitigate the leakage associated with SRAM bitcell.

- The macros using SRAM charge-based in-memory computing including the dense SRAM mixed-signal architectures able to integrate multi-bit MAC operations on classical SRAM arrays.

- The macros using time-domain in-memory computing either with a fully digital architecture and a time-to-digital conversion or with a mixed-signal current-based approach requiring an analog-to-digital conversion.

3.2 SRAM-Based Digital In-Memory Computing

Multi-bit computations have been demonstrated in Digital In-Memory Computing (DIMC) structures. Local processing elements are implemented close to SRAM bit cells storing the data, minimizing the need for off-chip memory. They increase efficiency by reusing data and implementing novel data flows [40–42]. Reference [43] uses near-memory computing and a new design methodology including datatype optimization, pruning, quantization, and active hardware fault detection to lower the SRAM voltage to reduce the consumption by a factor of 8. However, the operations are implemented with RTL and are not explicitly targeted for power reduction. Reference [44] uses SRAM XOR/NOR gates, adder trees, and shifters and compute 64 4-bit MACs in parallel with high efficiency (See Figure 3.1). This technique, requiring a high-frequency clock (up to 1.4 GHz), leverages the technology node (5 nm FinFET) to reduce the consumption and is suitable for large chips and moderate-resolution applications. Reference [45] uses a systolic architecture to implement a 1 to 16-bit MAC using an XNOR

gate and a full adder. The digital operation offers impressive results for binary NN and doesn't suffer from PVT, however, for multi-bit operation (<8 bits) mixed-signal architectures provide better efficiency.

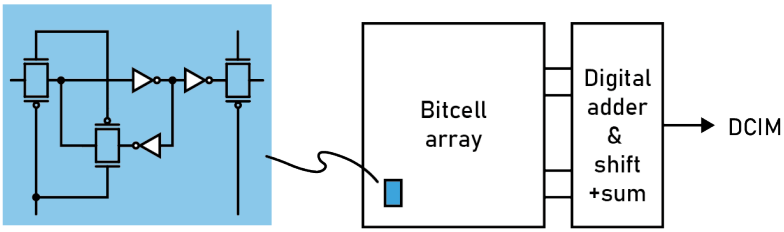


Figure 3.1: DIMC from [44]

3.3 Non-Volatile Memory Approaches

MVM dataflow at the edge of a Deep Neural Network (DNN) often requires fixed weights in a dense array to reach high memory capacity. On-chip non-volatile memories (NVM) are often used in Analog IMC applications for their high density and to try to mitigate the leakage of SRAM-based solutions [46]. The back-end of Line (BEOL) memories like Phase Change Memory (PCM) [47], Spin-transfer-torque RAM (STTRAM) [48] and Magnetic RAM (MRAM), and Resistive RRAM [49] are presented as well as a Front-end of Line (FEOL) FEFET technology [50, 51].

3.3.1 Phase-Change Memory

PCM is based on the reversible transition between a low and high resistance phase of chalcogenides. In [52], they are able to store up to 16 levels of conductance on an 8T4R bitcell, reaching 1587 GOPS/mm^2 and 10.5 TOPS/W with a 256x256 array implemented in 14 nm technology node. Reference [53] presents a spiking recurrent neural network with phase change memory neurons. The PCM cell is used as the integrating elements of a stochastic neuron to solve a Sudoku puzzle in hardware. The PCM devices are used as a source of true random noise for generating random spikes in an RNN. The PCM is a promising concept because of the high switching speed and low current operation but requires high voltages to tune the conductance value. Although compatible with CMOS process, they are not standard and widely available.

3.3.2 Spin-transfer-torque and Magnetic RAM

STTRAM improves the writing mechanism of conventional field-switching MRAM with spin transfer torques. The memory element is a magnetic tunnel junction (MTJ)with two ferromagnetic layers separated by a thin oxide. The parallel and antiparallel orientation of the two ferromagnetic layers allows tuning the conductance to low and high states thanks to recent achievements [54]. STTM-RAM solutions [55, 56] allows reaching energy efficiency between 5 and 25 TOPS/W and up to 176 TOPS/W [57] for low precision inputs and binary weights. STTM-RAM shows promising performance with write speed under 10 ns and $> 10^{12}$cycle endurance. However they are sensitive to process variation and are not yet available on standard CMOS process.

3.3.3 RRAM-Based Analog In-Memory Computing

The RRAM-based approach uses memristors to store the weights as conductance values to perform the multiplications in a crossbar array. The digital input vector is sent as analog voltages across all the rows of the array. The conductance value of each memristor modulates the current through an accumulation line. This approach requires DACs with large output currents, and a current comparator, as highlighted in [58]. It is possible to perform multi-bit operations with this approach by using up to 16 states of conductance per RRAM [59]. However, process variations prevent reaching a higher number of bits. Furthermore, in terms of energy dissipation, writing necessitates high current spikes (around 6 μA in [60]), while inference costs approximately 250 fJ/-MAC, according to [61]. It is noted that a trade-off exists between energy consumption and variability.

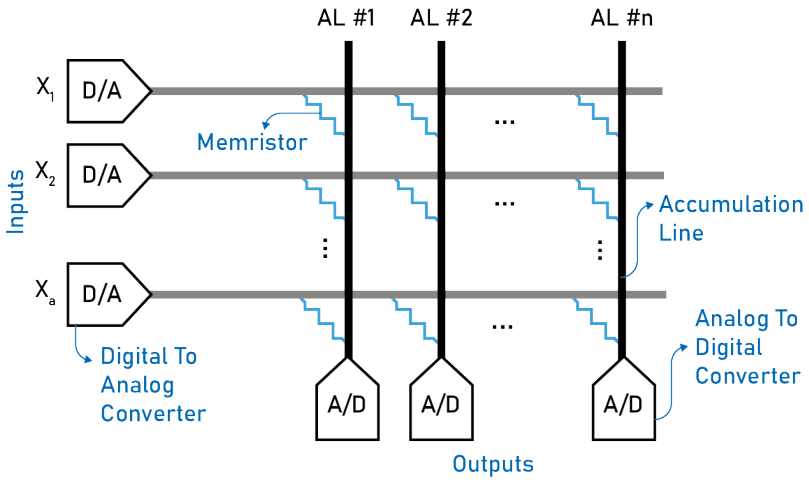


Figure 3.2: Example of RRAM-AIMC

3.3.4 FeFET

The Fe field effect transistor (FeFET) based approach consists in adding a Metal-FE-Metal (MFM) capacitor on the gate of the transistor. Thanks to two remnants polarization states caused by the ion displacement in an FE crystal lattice, it is possible to use MFM to store a bit [62]. Unlike BEOL memories, FeFET is fabricated in the FEOL with stricter material requirements. In [63] explore the usage of FeFET for hybrid memory solutions such as Ternary Content Addressable Memory (TCAM) and classic memory behavior. This technology is currently under development and offers promising high-density, low-leakage, low-latency and high-endurance memory, but suffers from device-to-device variations on deeply scaled FEFET that currently prevent their usage for matrix-vector-multiplication [64].

3.4 SRAM-Based Mixed-signal IMC

3.4.1 Charge-Based Analog IMC

In the charged-based approach, the multiplication is based on sharing charges or currents on a capacitive line, usually using SRAM and XNOR gates for binary multiplication (see Figure 3.3) [65]. Unit capacitors are charged according to each binary multiplication and charges are redistributed across all capacitors on an accumulation line, resulting in a voltage to be converted by the ADC [66–68]. To perform a multi-bit operation with switched capacitors, [61] shows a topology similar to a digital multiplier using one AL for each bit. However, this method needs additional circuitry to combine all the line's results, which is adequate for a reduced number of lines (<5) but dominates the power consumption for higher numbers of bits.

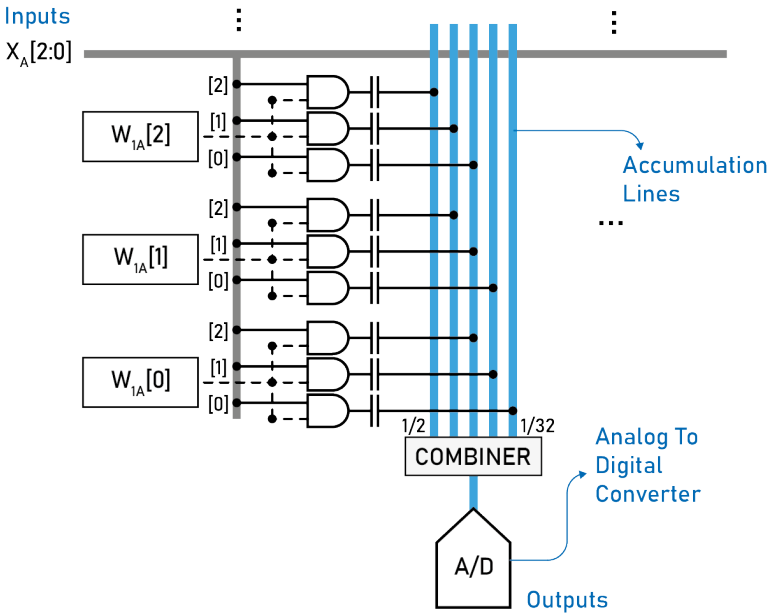


Figure 3.3: Example of Charge-Based Mixed-signal IMC

These charged-based SRAM arrays implement MACs as a weighted average of the bit-line voltage, which is proportional to the digital input values. The input is converted to a voltage and sent through the WL of a 6T SRAM bitcell. However, with this conventional approach [69, 70] where multiple wordlines are activated at once, the system suffers from writing disturbance where bits can be flipped if the level of the line is too low. Reference [39] proposes a 10T SRAM that decouples the memory write process and the MAC operation, therefore providing robustness but increasing the area overhead of the bitcell.

3.4.2 Time- and Current-Based Analog IMC

Reference [71] exploits Time-domain Near-Memory computing architecture using foundry-provided SRAM, this solution split an 8-b input into 4 voltage references applied to an edge delay cell that will create different delay according to the weight bit value, and a time-to-digital converter is then used to convert this delay in a partial MAC value. By spanning across the weight bit and using additional adders and shifters, it is possible to compute the total MAC value. However, high-speed clock (1GHz) is required, for the system to reach 84.45 TOPS/W with 50% input and weight sparsity. In [72] an SRAM with pulse width and amplitude-modulated WL access pulses is proposed to generate a bit line discharge proportional to the weighted sum of the stored weights bits W. The bit line voltage is then processed to perform the multiplication in time with the input X. The 8-bit multiplication is performed in two steps with 4 LSB and 4 MSB. The different values are finally aggregated via charge-sharing techniques and converted digitally. This method requires a specific PWM driver for each WL and is limited in weight precision due to the available bit line voltage range. Therefore increasing the bit width requires to add additional circuitry. Hence, increasing the area overhead. Current-Based Analog In-Memory Computing architectures use small currents (<1 nA) to charge an accumulation line, thanks to advanced CMOS technology that can drive small currents with enough precision for performing MVM. As shown in Figure 3.4, a current source is used to charge/discharge the line depending on the sign of the multiplication. An array of 100 accumulation lines, composed of 100 current sources charging a capacitive line with a 100 pA current, operating at 4 GOPS is equivalent to a 4,000 TOPS/W efficiency by taking into account only the current sources to perform the accumulation. This top efficiency will be degraded by the associated elements like the ADC and the gate performing the multiplication.

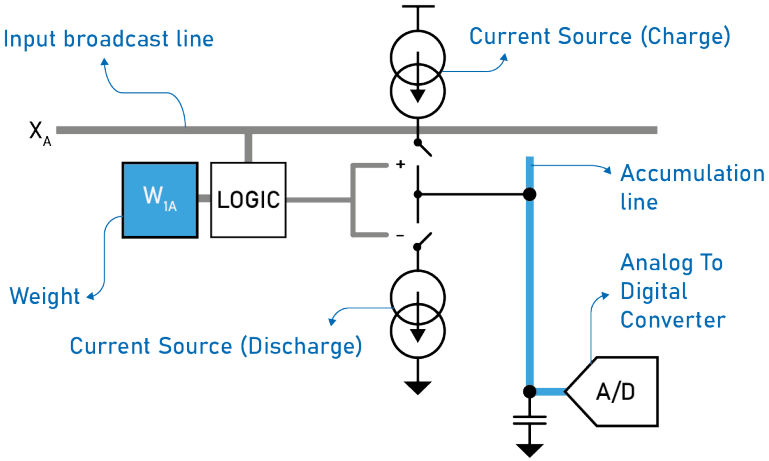


Figure 3.4: In-memory current source

However, this solution allows the computation of only ternary multiplications [1-bit W $\times$ (1-bit + sign) X]. To allow for multi-bit operations, it is possible to modulate the pulse width of the signal on the input broadcast line to represent the multi-bit multiplication. In this thesis, we proposed and designed a repetitive stretched pulse technique that allows performing a multi-bit operation using one current source per MAC.

3.5 Comparison of IMC architectures

In this section, we compare the pros and cons of the different architecture. The digital approach offers great results for binary MAC but suffers from an area overhead when working with multibit MAC. RRAM approaches offer the advantage of using the same elements for storage and computation but don't use the standard CMOS cell, need high currents, and are limited to 4 bits per cell (with current technology).

Architecture	SRAM DIMC [40–45]	NVM IMC [47–50, 52–57, 59, 60, 62, 63]	Charge Mixed-signal IMC [39, 61, 65–70]	Current + Time Mixed-signal IMC [71, 72]
Pros	Configurable Standard CMOS Scalable	High density	Standard CMOS	Standard CMOS Possible Core Efficiency: >4000 TOPS/W
Cons	High Speed Clock	Non Standard Technology High current or voltage needed for configuration	Writing issues (bit flip) large area overhead	Sensitive to PVT variations
Bitwidth limited by	Bitcell area	Technology	Circuit complexity	Latency

Table 3.1: Comparison of IMC architecture

Charge-based IMC uses standard CMOS and exploits the WL of an SRAM bitcell but suffers from bit flipping issues that can be resolved by adding transistors to the bitcell and therefore increasing the area overhead too. The current and time approach shows promising efficiency levels, although suffering from PVT can be mitigated through appropriate training of the NN. Table 3.1 summarizes this information.

Figure 3.5 presents an overview of the recent multibit NN accelerators (2018 to June 2022) and compares their efficiencies. The FPGA works are the most consuming ones and are nearly all under the 1 TOPS/W mark. Digital ASICs consume less power and are able to reach efficiencies between 1 and 10 TOPS/W with recent work getting close to 100 TOPS/W. Analog and Mixed ASICs are mainly located between the 10 and 100 TOPS/W lines. However, two projects go beyond the 100 TOPS/W. With our current and time approach, we take the opportunity to use all the available time between two consequent input data events on a neural network (i.e. 10 ms for KWS). This method allows reaching a low GOPS but ultra-low power system. Therefore, the target is an area never explored before for dedicated close to the application ASIC.

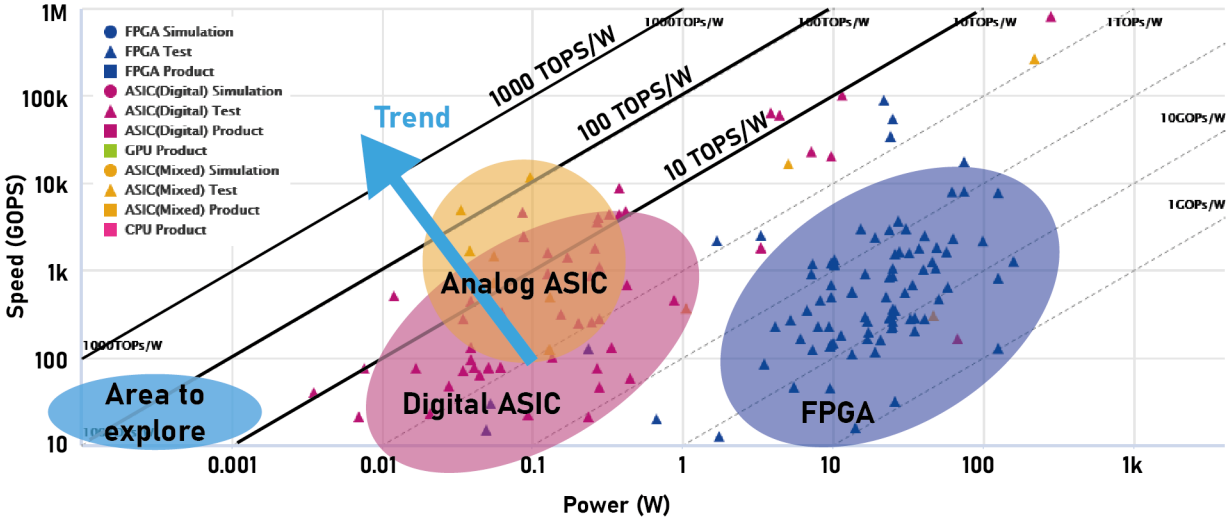


Figure 3.5: Comparison of multibit ASIC and FPGA from [73]

The two analogs ASIC over 100 TOPS/W are from [71] and [74]. Reference [71] exploits Time-domain Near-Memory computing architecture using foundry-provided SRAM, this solution split an 8-b input into 4 voltage references applied to an edge delay cell that will create different delay according to the weight bit value, and a time-to-digital converter is then used to convert this delay in a partial MAC value. By spanning across the weight bit and using additional adders and shifters, it is possible to compute the total MAC value. Reference [74] exploits charge-domain architecture, performing bitwise multiplication using XNOR and AND gates to drive a capacitor causing charges redistribution across a capacitance. Specific mapping and parallelization techniques allow for further optimization of the efficiency.

3.6 Conclusion

In this chapter, the In-Memory Computing principles were presented as well as multiple solutions to implement it, from digital to mixed-signal approaches. The main challenge with In-Memory Computing is to be able to decrease the energy/operation in a context where neural networks require lots of operation. TinyML offers the possibility to scale the performance and create hardware for specific applications that are matched together. Considering all the approaches, the current and time-based analog IMC seems to offer promising efficiency results, benefiting from the time between two input data. In this approach, it is possible to use a small current (thanks to recent CMOS technology nodes: 28 nm FDSOI in this work) during a long time to charge an accumulation line. This thesis work covers the design of an architecture exploiting the current and time approach and its implementation. The next chapters will present our approach, its implementation, and the test of a 28 nm FDSOI CMOS prototype.

Chapter 4

Time-Based Multiplication Concept

4.1 Introduction

The time and current-based analog compute in-memory take advantage of the available time between two incoming data of the neural network layer to perform multi-bit MAC operations. The principle of our method is to charge a capacitive line with a constant current during a time proportional to the product of an input and a weight. Figure 4.1 shows that a neuron is equivalent to multiple current sources controlled by a temporal signal charging an accumulation line. Each current source pair, noted processing elements, corresponds to a MAC

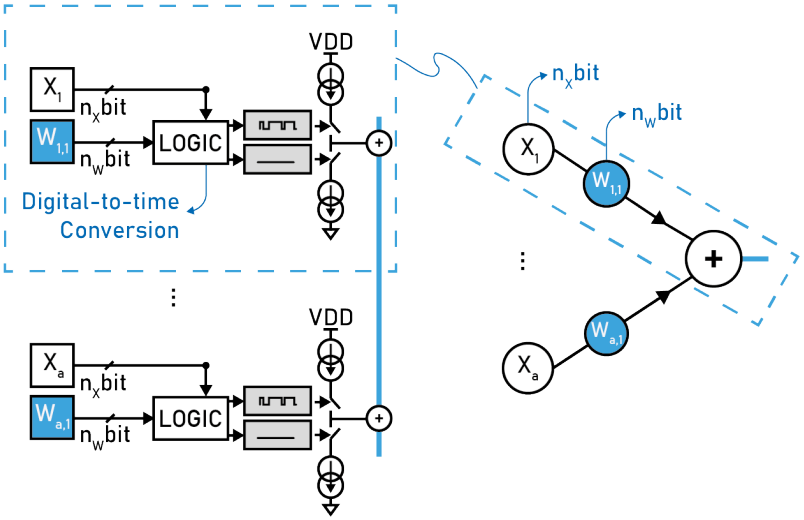


Figure 4.1: Left: Block diagram of a neuron implemented using time and current analog in-memory computing method. Right: Schematic of a neuron.

operation and requires a local command signal that allows the current to charge the accumulation line during a time proportional to the product of X and W, which are multi-bit.
In this chapter we will:

- present the product-to-time conversion principle,
- compare parallel and iterative architectures to identify the one that is best suited for low latency applications,
- present a global block diagram of the circuit.

4.2 Multiplication to Time Conversion

The goal of our macro is to achieve Matrix-Vector Multiplication by charging a capacitive line with a constant current during a time proportional to the vector-matrix product. The input vector X is composed of a elements (Equation 4.1) encoded on n_X bit (1-bit for the sign and $n_X - 1$ bit for the magnitude), the weight matrix W is of size $a \times n$ with n the number of accumulation line (Equation 4.2), the matrix elements are encoded on n_W bit, the same way as elements of vector X. The result output vector is O and is composed of n elements (Equation 4.3) encoded on N_O bit, the same way as the X vector.

$$X = \begin{bmatrix} X_1 & X_2 & \dots & X_a \end{bmatrix} \tag{4.1}$$

$$W = \begin{bmatrix} W_{1,1} & \dots & W_{1,n} \\ \vdots & \ddots & \vdots \\ W_{a,1} & \dots & W_{a,n} \end{bmatrix} \tag{4.2}$$

$$O = \begin{bmatrix} X_1 & X_2 & \dots & X_a \end{bmatrix} \times \begin{bmatrix} W_{1,1} & \dots & W_{1,n} \\ \vdots & \ddots & \vdots \\ W_{a,1} & \dots & W_{a,n} \end{bmatrix} = \begin{bmatrix} O_1 & O_2 & \dots & 0_n \end{bmatrix} \tag{4.3}$$

If we note $i \in \{1, n\}$ corresponding to the column index and $j \in \{1, a\}$ corresponding to the row index, then we can write.

$$O_i = \sum_{j=1}^{a} R_{j,i} \tag{4.4}$$

$$R_{j,i} = (\sum_{c=1}^{n_X - 1} 2^{c-1} X_j[c])(\sum_{d=1}^{n_W - 1} 2^{d-1} W_{j,i}[d]) \tag{4.5}$$

$R_{j,i}$ is a binary multiplication of two numbers of size n_X and n_W. Each bit is associated with a ponderation consisting of a power of two of its range order (ie:2^0 for the LSB). The resulting voltage across a capacitance charged by a constant current is described by Equation 4.6.

$$V_c = \frac{I \times T}{C} \tag{4.6}$$

The time needs to be proportional to the product result, the accumulation line final voltage V_{o_i} is equal to

$$V_{o_i} = \frac{I \times T \times O_i}{C} \tag{4.7}$$

Where T is a time reference calculated according to the maximum duration of an operation, the capacitance C, and the voltage range across C. To control the current charging the capacitance, a command signal used to control a switch is created and presented in Figure 4.2. With d the $W_{j,i}$ bit rank and c the X_j bit rank. This signal is then used to charge/discharge a capacitive line according to the sign of the multiplication.

4.3 Parallel and iterative architecture comparison

Using the previous technique, each MAC operation needs to have one command signal to perform MVM. In this section, parallel and iterative architectures will be analyzed and compared.

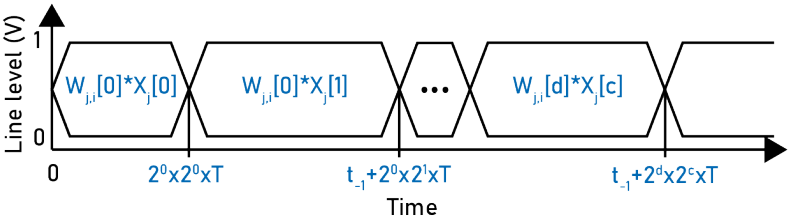


Figure 4.2: Command signal for a current source.

4.3.1 Parallel architecture

In this type of architecture, presented in Figure 4.3, there are two current sources per MAC placed along the accumulation line. To provide signed operations, there is a current source to charge and another to discharge the capacitive line. In this configuration, using a time reference $T = 20\,ns$, 100 5-bit MAC would take $4.5\mu s$ to be computed, and 100 8-bit MAC will be performed in $322\mu s$. Thus, in a parallel architecture, the number of bits limits the throughput.

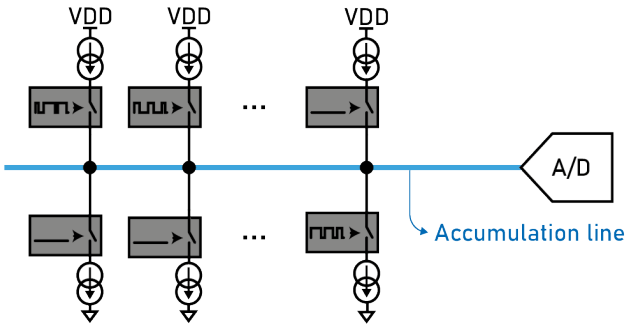


Figure 4.3: Current source parallel architecture.

4.3.2 Iterative architecture

In an iterative architecture, one current source is used for multiple MAC through time, as shown in Figure 4.4. In this configuration, the number of bits and the number of MACs will limit the throughput. Of course, there cannot be a single current source for an entire neural network, in our case study we state that there is one pair of current sources per accumulation line. In this configuration, using a time reference $T = 20\,ns$, 100 5-bit MAC takes $450\mu s$ to be computed, 100 8-bit MAC takes $32\,ms$.

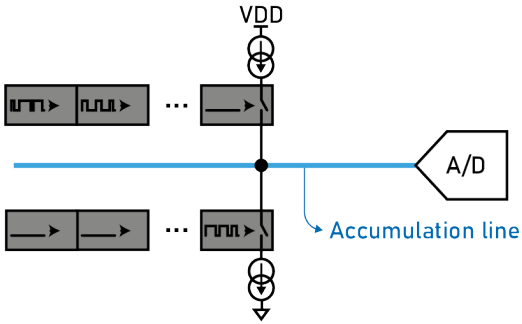


Figure 4.4: Current source iterative architecture.

4.3.3 Constraints defined by applications

Considering different sizes of arrays, Table 4.1 presents the resulting GOPS and the inference time for a parallel architecture running KWS applications using an LSTM neural network composed of 64 hidden units and a fully connected layer. The timing used for the elementwise

operations and additions is as if we use the array to perform them resulting in a conservative result in terms of inference time. Table 4.2 presents the same result for an iterative architecture running the same network.

Array Size	5bit GOPS	8bit GOPS	KWS Inference 5bits (ms)	KWS Inference 8bits (ms)
64x64	1.82	0.02	0.04	3.22
128x128	7.28	0.10	0.027	1.93
256x256	29.12	0.40	0.013	0.96

Table 4.1: GOPS and KWS Latency for different sizes parallel architecture running a 64 hidden units LSTM and 1 feed forward layer

Array Size	5bit GOPS	8bit GOPS	KWS Inference 5bits (ms)	KWS Inference 8bits (ms)
1x64	0.028	0.0004	2.88	206
1x128	0.056	0.0008	1.72	123
1x256	0.113	0.0016	0.86	61

Table 4.2: GOPS and KWS Latency for different sizes iterative architecture running a 64 hidden units LSTM and 1 feed forward layer

Iterative architecture running 8-bit MAC cannot be used for this type of application as the inference time is well over 10 ms, which is the minimum period of incoming inputs frame for KWS application, as shown in Chapter 2. Parallel architecture running 8-bit MAC can run KWS applications but their throughput might be limiting for more resources demanding tasks like image processing. 5-bit MAC offers suitable throughput for parallel architecture and can run KWS applications for the iterative architecture, however, in the same way, this type of architecture although occupying less space will be limiting for more demanding applications, like video applications that can require high frame rate for high-resolution images. The parallel architecture running 5-bit MAC offers promising throughput results, especially for an array of 128x128 and 256x256 with respectively 7.28 GOPS, and 29.12 GOPS allowing to implement audio applications and low latency video applications for preprocessing units.

4.4 Proposed high-level architecture

Figure 4.3 shows that a different command signal needs to be created for each current source pair. Each command signal is a representation of the multiplication of the input X_j times the weight value $W_{j,i}$. The weight needs to be placed close to the current sources. However, the input X is shared across multiple accumulation lines. We implemented a low consumption and configurable way to create a local command signal by using weighted time pulses that are broadcast across the array and gated by the X_j and $W_{j,i}$ values. A fully digital pattern generator creates these pulses. There is only one generator for the whole array, therefore amortizing its consumption across all the MACs. Figure 4.5 shows the different patterns created for a 5-bit architecture (Sign + 4bits).

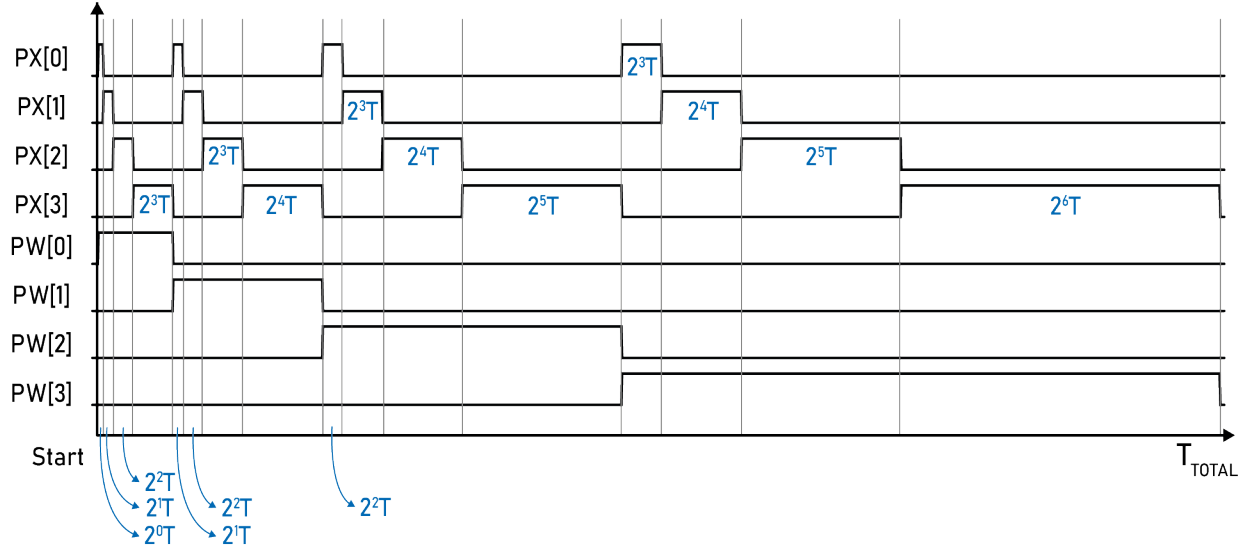


Figure 4.5: Time pulses created for the matrix array. PX[i] signals will be masked by the corresponding X[i] bit of the input and PW[j] signals will be masked by the corresponding W[j] bit of the weights.

The patterns are created without overlapping so that this masked signal can be used to command the switches of the current sources. Figure 4.6 represents what the overall architecture looks like. The PX patterns are first masked by the input cells. Since the patterns do not overlap, we can use an AND gate to send the masked signal into one X signal, which will be sent to the corresponding W rank thanks to the W patterns. This signal XW will then be masked by the stored W of each cell along the matrix row before controlling the switch. To perform signed multiplications, the accumulation line is pre-charged at $V_{DD}/2$, and an XOR on the sign is used to control the direction. The voltage across the AL will then evolve according to the current multiplication product value. At the end of the multiplication, an ADC per column converts the value.

4.5 Evaluation of non-idealities and mismatch for time and current-based Analog in-memory Computing

One of the main drawbacks of analog domain IMC is the sensitivity of the architecture to variations and non-linearity, introducing errors in the MAC operations and degrading the accuracy of the neural network inference. In this part, the impact of the mismatch on time and current-based in-memory computing is calculated using a Matlab® model. The analysis proceeds by studying the impact of other elements on the MAC operations result. From this results, the robustness of neural networks to output variation is measured by introducing an error at the output of the layer composing a CNN and a Feed Forward Neural Network performing MNIST classification.

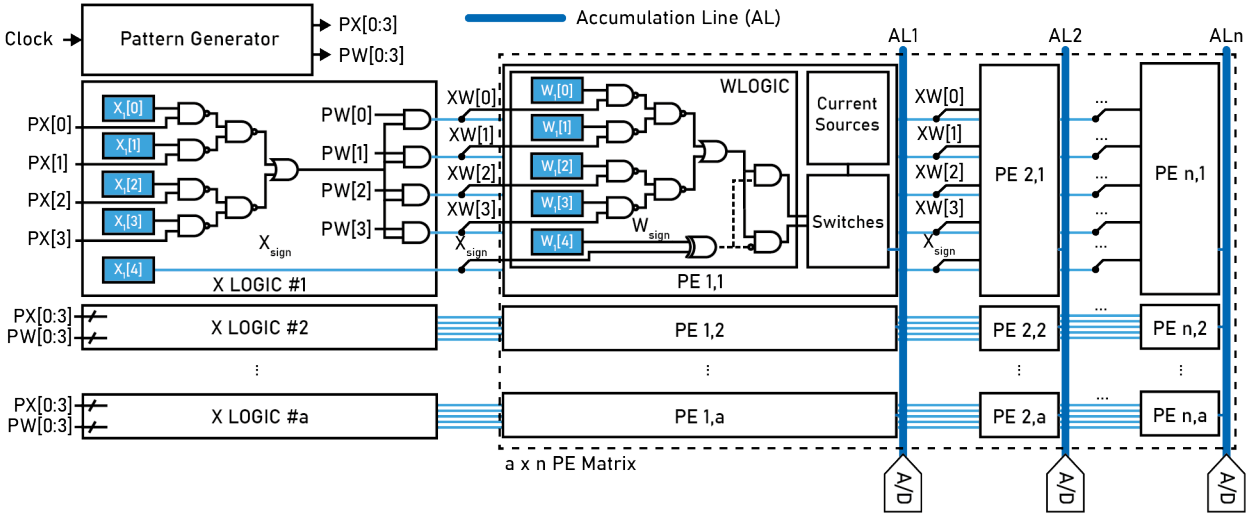


Figure 4.6: Pattern broadcast and time masking architecture.

4.5.1 Simulation of the time and current-based computation

A computation simulation was developed on Matlab® to evaluate the impact of mismatch, accumulation line capacitance variation, and noise on the output result. Each error might compensate for the other, they are evaluated separately first and together in the last part of this section. Inputs and weights are converted into an array storing their binary representation (Sign + 4b Magnitude). The error is expressed in number of full precision LSBs, which is equal to the voltage dynamic ($600mV - 200mV$) divided by the number of levels. $LSB = 400mV/45000 = 8.8\mu V$, the number of levels is equal to $(2^4 - 1)^2 * 100 * 2 = 45000$. At the beginning of each pattern pulse, the number of contributing current sources is calculated by summing the number of corresponding bits noted N_{ON} for the number of discharging current sources and P_{ON} for the number of charging current sources. P_{ON} and N_{ON} are updated at each time step corresponding to a new pulse. For higher precision, the simulation is performed discretely in time according to a time step T_s. For each time step, we calculate the output voltage:

$$V_{out}(T_s) = V_{out}(T_s - 1) - \frac{I_{ref} T_s N_{ON}}{C_{ref}} + \frac{I_{ref} T_s P_{ON}}{C_{ref}} \tag{4.8}$$

Here the current used is equal to a current reference I_{ref}, exported from Cadence® and used in Matlab thanks to interpolation methods allowing to choose the reference current according to the voltage level. The different error values are computed for 10,000 random inputs and weights drawn from a normal distribution. The mean error standard deviation due to the nonlinearity of the current source is equal to $1.36\mu V$ ($0.15LSB$).

Impact of mismatch

We model a mismatch by drawing current from the following normal distribution:

$$I_{mismatch} = \mathcal{N}(I_{ref}, \sigma = I_{ref} \times \epsilon) \tag{4.9}$$

with ϵ the mismatch value in percentage of μ. The $I_{mismatch}$ is drawn once for each element at the beginning of the computation and kept constant during the computation of the accumulation line. With an input mismatch of 10%, the mean final error standard deviation is $1.4mV$ $(160LSB)$.

Impact of accumulation line capacitance non-linearity

We model the variation of the capacitance by retrieving the capacitance value observed on the output of a processing element from Cadence® simulation. Figure 5.9 shows the capacitance value of an accumulation line composed of 100 PE depending on the state of the switch. A mean capacitance value is calculated, which corresponds to one of the two switches controlling the two current sources being on and the other off. By using interpolation we can use a capacitance value at each time step according to the accumulation line voltage. The error standard deviation due to capacitance non-linearity is equal to $21\mu V$ $(2.36LSB)$.

Summary of contributions

The final error contribution, simulated by integrating all the error sources during computation has a standard deviation equal to $1.4mV$ dominated by the current mismatch value. With an available voltage dynamic ranging from 0.2 V to 0.6 V the number of quantization level N_Q is equal to:

$$N_Q = \frac{0.6 - 0.2}{0.0014} = 286 \tag{4.10}$$

Current Mismatch	Standard Deviation σ	$\epsilon = \sigma/\mu$	Output Precision Effective bits
10%	160 LSB (1.4 mV)	0.35%	8.15-bit
20%	319 LSB (2.8 mV)	0.7%	7.15-bit

Table 4.3: Evolution of the output precision in function of current mismatch value, the number of LSB corresponds to a full precision LSB of $8.8\mu V$ (see computation details at the beginning of the section).

286 quantization levels correspond to 8.15-bit resolution. As the error is dominated by the current mismatch value, we performed simulations for different values and listed the corresponding output quantification in 4.3. The $\epsilon = \sigma/\mu$ value is calculated with a $\mu = 400\,mV$ and σ equal to the error standard deviation, it represents the output deviation of a 100-element accumulation line with respect to the current mismatch and random inputs and weights. In the worst-case scenario of a 20% current mismatch, the error rate on a 7-bit output would be 13%. This error would drop below 1% in the case of a 5-bit output, as shown in Figure 4.7, which validates the architecture choices.

4.5.2 NN robustness to deviation

This analysis focuses on finding the effect of an error on the output of a layer on the accuracy of a neural network. To be complete, the study of such effects would need to be performed across a wide range of neural network models. This analysis tries to get a sense of the accuracy drop associated with an added error on two models: a CNN and a Feed Forward Neural Network performing MNIST Classification.

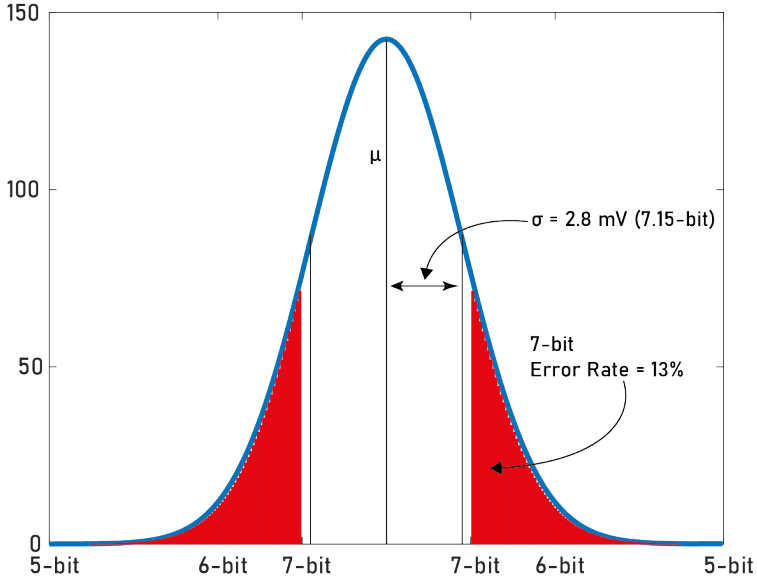


Figure 4.7: Distribution of the final output voltage of a computation with a 20% current mismatch, resulting in a standard deviation of 2.8 mV.

A special Error Layer was implemented on Matlab®, the vector X and the mismatch value ϵ are used as inputs, the output is:

$$Z = \mathcal{N}(X, X \times \epsilon) \tag{4.11}$$

Which draws a random number for each element X_i from a normal distribution $\mathcal{N}(X_i, X_i \times \epsilon)$. The ϵ values are derived from 4.3. This layer was included in neural network architectures performing MNIST classification. The first one is a Convolutional Neural Network (CNN) composed of 3 Convolutional Layers and their respective activations,

Batch Normalization layers, and the final Fully Connected and Softmax layers (see Figure 4.8). The second one is a simpler Feed Forward Neural Network composed of two Fully Connected Layer and their respective activations (see Figure 4.9).

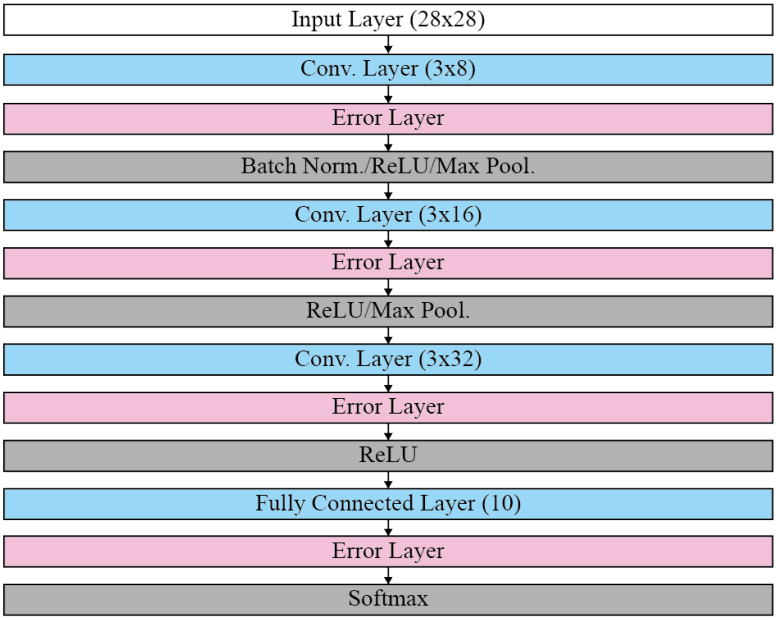


Figure 4.8: Detailed architecture of the Feed Forward NN.

Between each layer and their respective activations and normalization, the Mismatch Layer is added to simulate the impact of an error on the output. Table 4.4 shows the accuracy for different mismatch values used during training. A 20% current mismatch results in a less than 1% accuracy drop if trained properly and even some increase in accuracy in some cases, as the introduction of jitter during training can help to converge [75].

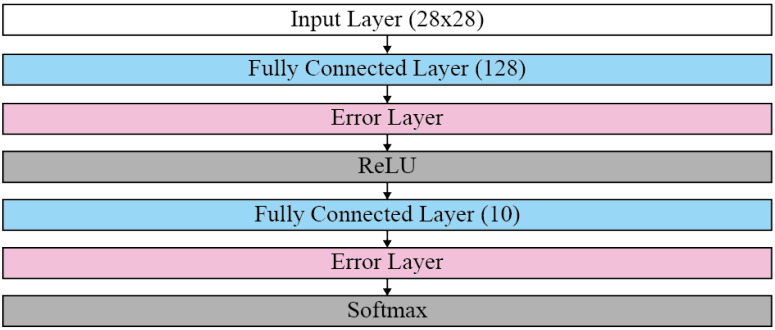


Figure 4.9: Detailed architecture of the Feed Forward NN.

Table 4.4: Accuracy VS Mismatch

Current Mismatch	Output Dev	CNN MNIST Accuracy	FFNN MNIST Accuracy
0%	0%	99.84%	91.8%
10%	0.35%	99.48%	93%
20%	0.7%	99.52%	92.16%

However, the output deviation resulting from the current mismatch value is dependent on the number of elements on the accumulation line and the value of the multiplication. Therefore, it is difficult to link a specific mismatch to an output deviation. Further studies, achieved by setting the output deviation to 10% show an accuracy drop of less than 1%, showing the robustness of neural networks to current mismatch.

4.6 Conclusion

We propose to use a parallel architecture coupled with a central pattern generator to provide an efficient way to perform MVM in a mixed-signal array. The energy consumption of the pattern generator is amortized across the MAC array, meaning that the efficiency increases with the number of MAC implemented. The number of bits used for each MAC is configurable on the pattern generator and the clock frequency can be changed to suit the capacitance and current ratio according to the application needs. The analysis of the precision of the time and current-based in-memory computing compared to the NN robustness shows that our solution is in line with the targeted applications. In the next chapter, the implementation of the circuit is presented and each block's energy consumption is simulated.

Chapter 5

Circuit Implementation in 28 nm FDSOI

5.1 Introduction

The proposed time- and current-based architecture was implemented in 28 nm FDSOI CMOS technology from STMicroelectronics. This technology node includes extended body biasing features compared to standard CMOS thanks to a buried oxide layer, which isolates the transistor from the substrate, allowing the body biasing voltage to reach +/- 3 V. This technology reduces the leakage and allows controlling transistor threshold voltage. Conventional well configuration is used for Regular Voltage Threshold (RVT) devices, it is used for leakage optimization. Low Voltage Threshold (LVT) devices use the flip-well architecture, which allows NMOS transistors to be fabricated on N-Well and PMOS transistors to be fabricated on P-Well, they are mainly used for speed optimization. Polybiased digital RVT and LVT IPs are provided, reducing even further their leakage.

The circuit contains 4 accumulation lines sharing 100 inputs. Its objective is to validate the product-to-time concept and to measure the consumption and accuracy of the 5-bit MAC operation. It is mounted in a JLCC68 package and the ADCs and 50 MHz clock are external for testing purposes. The circuit block diagram, presented in Figure 5.1, is composed of four main parts: the Processing Elements (PE) array, the Intput block, the Pattern Generator, and the Output block. Each PE includes the weights storage, the logic for the digital-to-time conversion, the current sources, and the switches. Each PE row is connected to an element of the Input Block. This block is connected to the Pattern Generator, a fully digital block that outputs the patterns presented in Figure 4.5 in the previous chapter. The output block is composed of the accumulation line and an Operational Amplifier to output the accumulation line voltage. In this chapter, each element will be detailed with simulation results and estimated power consumption. Additional elements will be added to solve issues and tackle design challenges.

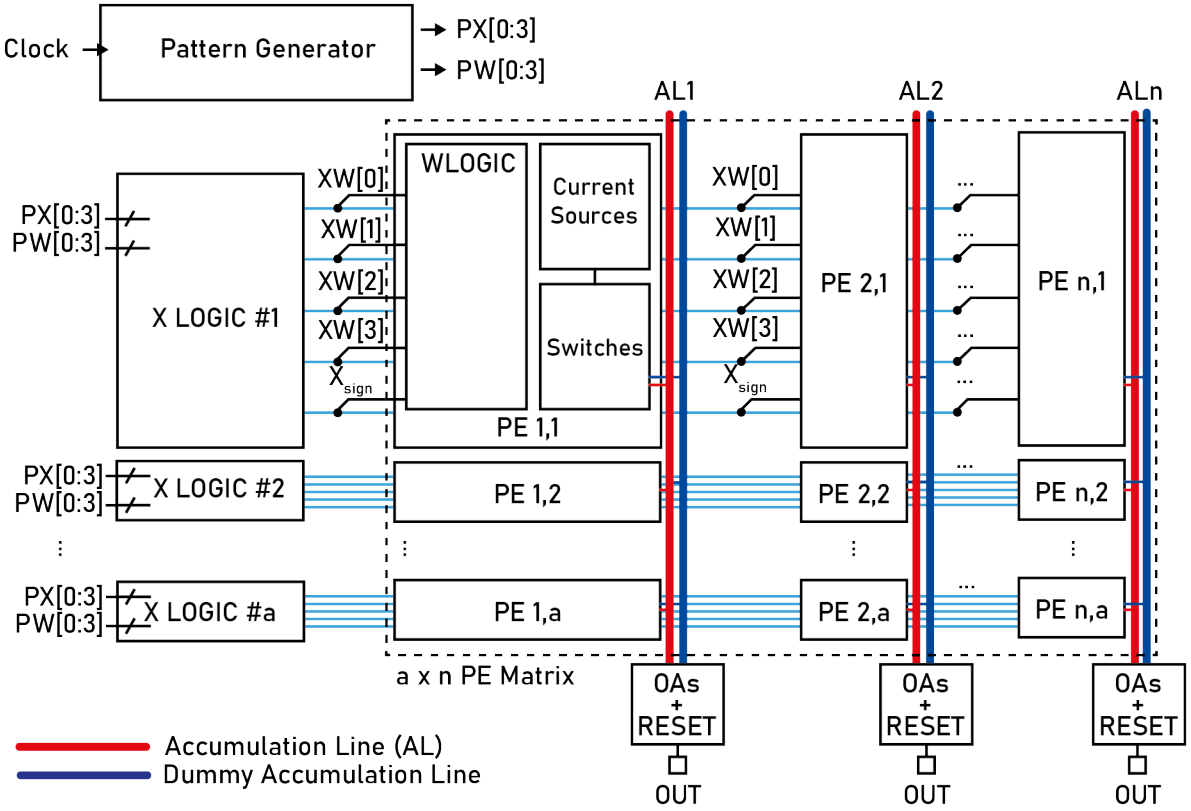


Figure 5.1: Global architecture. Where CS is Current Source, SW is Switches, RST is Reset, and OA for Operational Amplifier.

5.2 Current sources

5.2.1 Current mirror architecture

The number of current sources present on an accumulation line corresponds to the number of inputs for the input layer and to the size of the previous neuron layer for subsequent layers. Current mirrors are used to provide the reference current to each MAC PE, as shown in Figure 5.4. In our case, the input layer size is 100. One hundred gates connected together make the gate leakage quite significant. Leakage is around 320 fA per transistor for PMOS and 50 fA for NMOS transistors, reaching a 32 pA decrease on the output of the PMOS current sources and 5 pA on the NMOS current source for 100 input lines with a 100 pA current reference. This leakage problem is solved by using thick oxide transistors, lowering the total gate leakage to 5 aA on average for PMOS and NMOS transistors, for 100 gates in parallel, according to the DC simulation. Another effect of this current mirror architecture is that the accumulation line voltage may push the transistors in their linear region causing the output current to drop. This effect prevents the use of the full output voltage range. Figure 5.2 shows that for a 0.8 V supply voltage the current is constant between 0.2 V and 0.6 V for cascoded current sources and a current reference of 100 pA. This phenomenon can be advantageously used as an activation function like Sigmoid and Tanh as the extremities of the available range are squashed. Furthermore, training NN using Sigmoid, Tanh, and Batch Normalization techniques gives weights that output Gaussian shapes activation [76], centered in the linear part of our current sources. Finally, including this effect during training can help to reduce its effect.

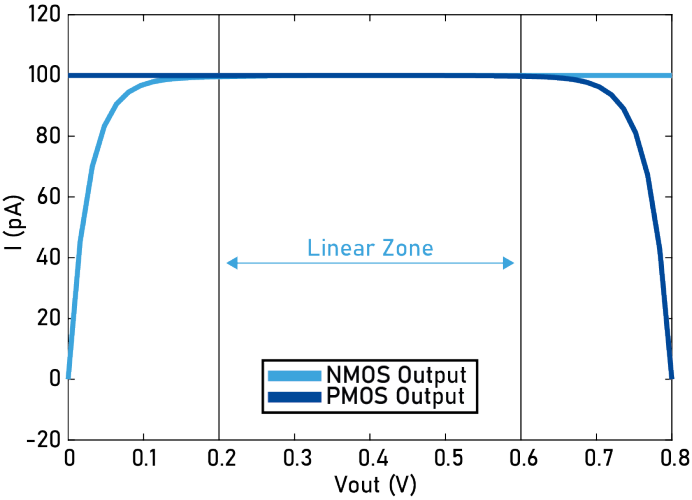


Figure 5.2: Current output for PMOS and NMOS current sources with cascoded architecture for 800n by 800n transistors.

5.2.2 Transistor Mismatch

We reach a $\frac{\sigma}{\mu}$ of 18% mismatch using 800 nm by 800 nm PMOS and 6% mismatch for NMOS transistors of the same size. This allows for a 5-bit output according to our simulation from Chapter 4. Taking into consideration that other errors will add up when implementing other parts of the circuit. The values are found from a 200 points Montecarlo analysis with a constant W of 800 nm and a current reference of 100 pA.

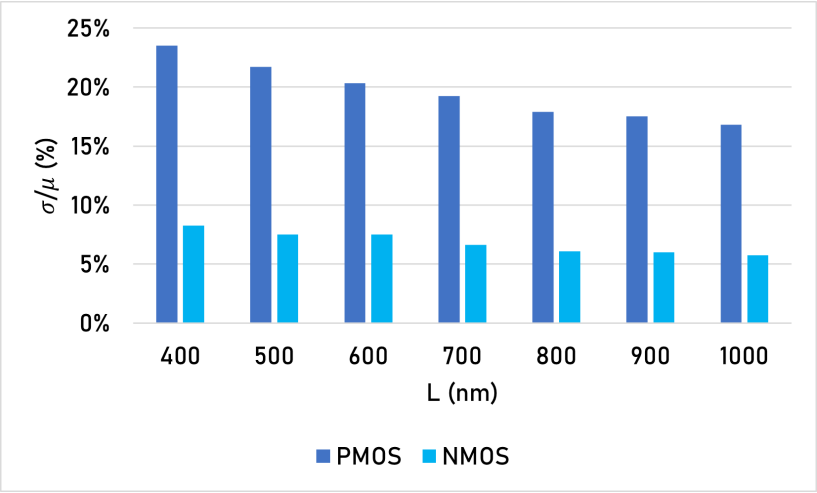


Figure 5.3: Simulated mismatch value from 200 points Montecarlo analysis with a W = 800nm for different L.

5.2.3 Output impedance

The output impedance of the current mirror introduces an error in the product and accumulation result. To reduce this error by increasing the output impedance of the current mirror, we chose a cascode architecture. The output impedance goes from $1.30 \times 10^{11}\Omega$ with a simple current mirror to $5.45 \times 10^{12}\Omega$ with the cascode one for the NMOS and from $1.6 \times 10^{11}\Omega$ with a simple current mirror to $3 \times 10^{13}\Omega$ with the cascode one for the PMOS. The maximum output error is reduced from 9.31 mV (2% of the available voltage range) with the simple current mirror to 0.05 mV with the cascode one, meaning that the mismatch error is more dominant.

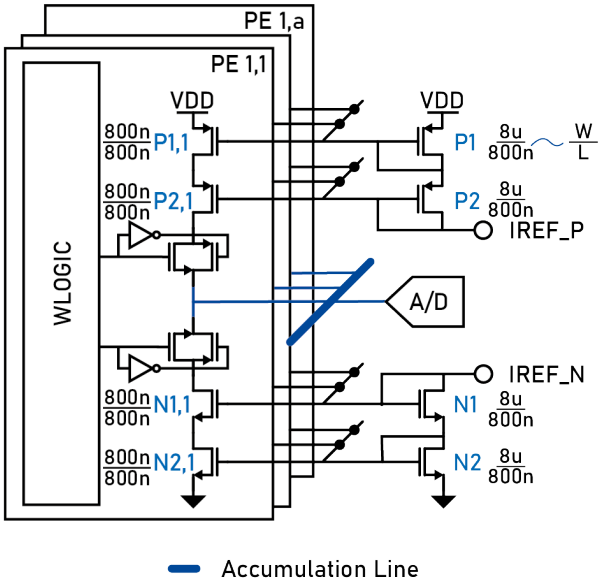


Figure 5.4: Architecture of the current sources.

5.2.4 Current Sources Consumption

The current sources are driven through a bias tree, with an input reference current of 100 nA and the final current sources drawing 100 pA. The simulated total power consumption, including the reference current mirrors, under 0.8V is 182.4 nW for the 4-by-100 array. For a 5-bit MAC operation, lasting $4.5\mu s$ (time reference $T = 20$ns), this adds to an energy of 2.05 fJ/MAC.

Power	182.4 nW
Energy/MAC	2.05 fJ/MAC

Table 5.1: Consumption and energy of the current source block.

5.3 Switches

5.3.1 Settling time

Depending on the place of the switch, the settling time varies. According to our simulations, if the switches are placed on the gate of the current mirror transistors or on their output, the settling time is greater than 700 ns. This value is too high compared to the reference time T fixed at 20 ns. To solve this issue, we can steer the current in a dummy line. As the current is small ($< 100\,pA$ according to simulated accumulation line capacitance) its consumption won't impact the efficiency of the system. The switches need to be at the output of the current source to implement the current steering methods. The settling time is then around 100 ps. As the accumulation line value can range from 0 V to V_{DD} we use pass gate switches. The power consumption of the switch mainly comes from the leakage and the dynamic power dissipation when switching. Considering a gate capacitance value of $C_n = 70\,aF$ for the NMOS and $C_p = 70\,aF$ for the PMOS, we have an equivalent capacitance $C_{eq} = 140\,aF$. If we take S_{max} the maximum number of toggles per MAC, which is equal to 8 for 5-bit inputs and weights, and $V_{dd} = 0.8V$ the switching energy per MAC is defined by equation 5.1:

$$E_{switch}/MAC = S_{max} C_{eq} V_{dd}^2 \tag{5.1}$$

$$E_{switch}/MAC = 0.71\,fJ \tag{5.2}$$

If the N and P current sources are matched, the current leakage doesn't impact the computation precision. The leakage is around 0.5 pA for 100 nm by 100 nm transistors. Increasing their size to reduce the leakage will not reduce the total consumption as the current is steered in the dummy line, therefore the size is preferably optimized.

5.3.2 Charge injection

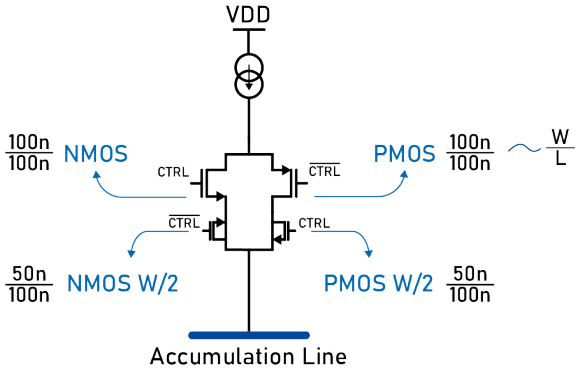


Figure 5.5: Passgate and dummy switches.

By placing the switches on the output of the current sources, charge injection caused by the switches modifies the accumulation line level value and introduces an error on the product and accumulation result. To mitigate this effect, dummy transistors are placed on the output of the switch as shown in Figure 5.5. The final switch size is 100 nm by 100 nm with dummy transistors' width divided by two. The effect of charge injection mitigation is shown in Figure 5.6 where a current source output is switched off and on.

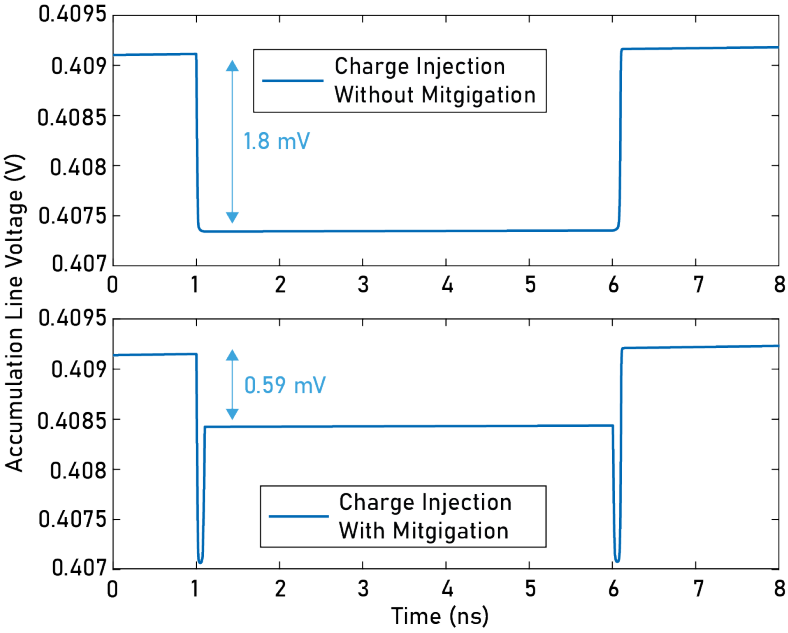


Figure 5.6: Simulation of two switches on an accumulation line with and without mitigation with dummy transistor.

5.3.3 Switch Consumption

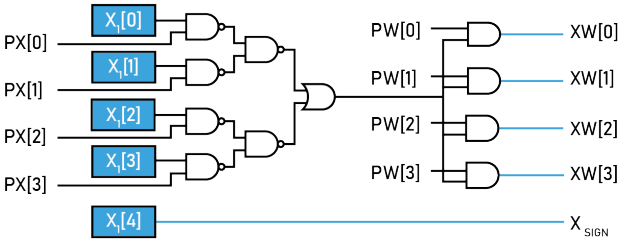
The total energy consumption of the switches is only due to dynamic power dissipation. A maximum energy of 6.38 zJ is reached for a 5-bit architecture as shown in Table 5.2 since there are two switches per PE.

Max Energy/MAC	6.38 zJ

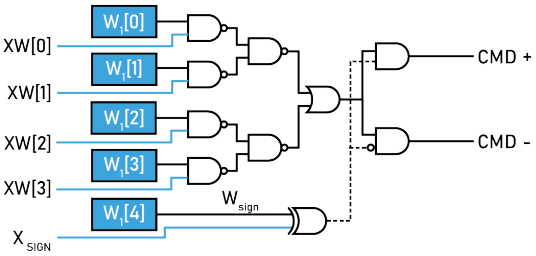
Table 5.2: Consumption and energy of one switch block.

5.4 X LOGIC and W LOGIC

5.4.1 Block architecture



(a) XLogic block diagram.



(b) WLogic block

Figure 5.7: Logic blocks X and W.

The XLOGIC block receives the pattern $PX[0]$ to $PX[n_x - 2]$ that will be gated by the X_j value, this signal is then sent to be gated to the corresponding $W_{j,i}$ line thanks to the $PW[0]$ to $PW[n_x - 2]$, we note this signal XW. In the same way XLOGIC is gating the pattern,

WLOGIC gates the XW signals and sends it to the switch controlling the current source. The blocks have been designed using digital gates according to Figures 5.7a and 5.7b, the storage elements are flip-flop registers.

5.4.2 Logic Blocks Consumption

The power consumption of these blocks is extracted from a transient simulation, performing the maximum number of switches on the CMD+/- lines. By taking the average current drawn from the supply of the logic block during $4.5\mu s$, which is the time of a 5-bit MAC. The total consumption of one WLOGIC cell is simulated at 2.22 nW under 0.8V with the maximum number of switches to perform. The XLOGIC block power is simulated at 23 nW with the maximum number of switches. As the result of XLOGIC being shared across all the array rows, the energy is calculated for a 100x100 array.

WLOGIC Power	2.22 nW
XLOGIC Power	23 nW
WLOGIC Energy	10 fJ
XLOGIC Energy	1 fJ

Table 5.3: Consumption and energy of the logic blocks.

5.5 Accumulation lines

5.5.1 Accumulation line capacitance

The accumulation line capacitance is created by the parasitic capacitance of the switches and by the metal line of the accumulation line, as seen in Figure 5.8. The capacitance evaluated in the simulation was around 70fF for the parasitic capacitance and 9 fF for the metal line.

5.5.2 Accumulation line non-linear capacitance

Depending on the state of the switches and the capacitance-voltage level, the capacitance evolves. Figure 5.9 shows the evolution of the capacitance without taking into account the metal line. There is a difference of 50 fF on average between the capacitance with open switches and closed switches. As the number of opened/closed switches depends

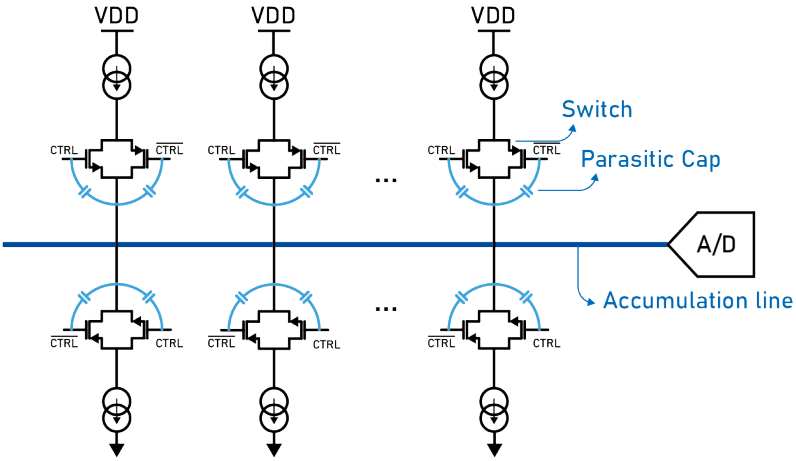


Figure 5.8: Accumulation line capacitance created by switches parasitic caps.

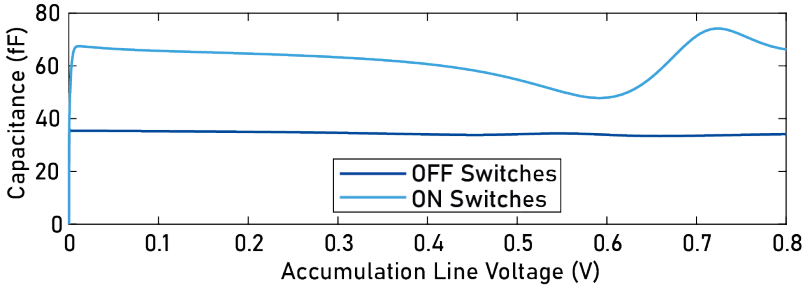


Figure 5.9: Evolution of the capacitive line value depending on the position of all the switches.

on the inputs and weights, we introduce the capacitance variation of the accumulation line on our Matlab model. At each time step, as the new output voltage is calculated, we interpolate the new value of the capacitance depending on the number of open and closed switches. The output error is around 11% to 67% LSB which is in line with our previous simulation on MNIST where the error can be 1.5 LSB with an impact on the accuracy inferior to 5%.

5.5.3 Charge sharing effect

Due to the settling time being too long compared to the reference time, we introduce the use of current steering techniques to have a reliable fast settling time. Figure 5.10 shows the added dummy line where the current is steered. As the two opposed switches are not perfectly synchronized, there are a few nanoseconds where the accumulation line and the dummy accumulation line are connected. Charge sharing occurs during that time and the two lines try to equilibrate to the same voltage level causing them to lose the product value on the accumulation line as shown in Figure 5.11.

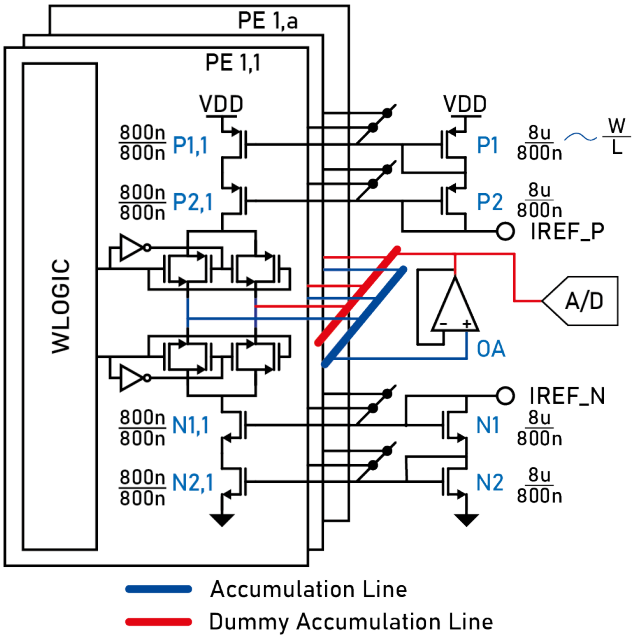


Figure 5.10: PE block diagram with the dummy line.

5.5.4 Charge-sharing mitigation

To mitigate the charge-sharing effect, we use one Operational Amplifier (OA) between the accumulation line and the dummy accumulation line as a voltage follower so that the lines are at the same level. In our chosen configuration, the dummy line capacitance has the same value as the main accumulation line capacitance. However, it is possible to increase the main accumulation capacitance so that the charge sharing is reduced, at the expense of increased area and current. The accumulation line level is output from the dummy accumulation line through a second voltage follower, noted OA B, that works the same way as the one between the two accumulation lines, noted OA A.

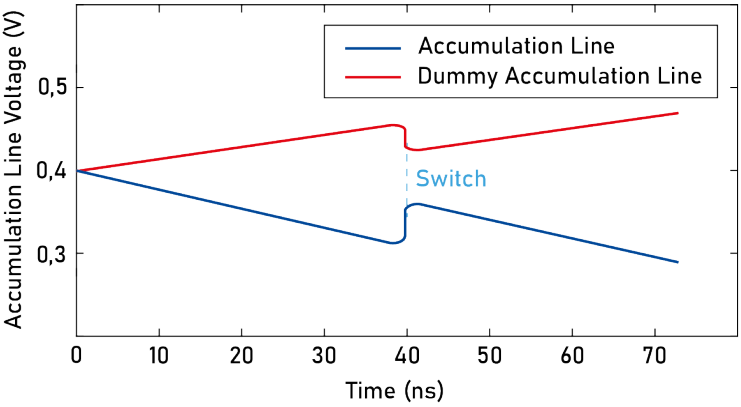


Figure 5.11: Charge sharing effect.

5.5.5 Operational Amplifiers

The OA A is used as a voltage follower between the AL and DAL. It is working between 0.2 V to 0.6 V which corresponds to the maximum voltage reachable by the accumulation line (at $V_{DD} = 0.8\,V$). A rail-to-rail OA architecture shown in Figure 5.12 is used with the two differential pairs composed of transistors P1/P2 and N1/N2. Depending on the input level, one of the differential pairs will amplify the signal. Input transistor mismatch in Operational Amplifier A introduces an offset between the two accumulation lines. Body biasing is used to reduce this offset. Figure 5.13 shows the impact of body biasing on transistors P2 and N2. The same bias is added to the nominal body biasing voltage, 0V for PMOS and V_{dd} for NMOS. Figure 5.15 shows the end of the accumulation line. A second operational amplifier, OA B, is placed as a follower between the dummy accumulation line and the output pin of the circuit to drive the off-chip capacitance load. OA B works the same way as OA A with modified sizing and bias. However, there is no body-biasing to cancel the offset. The offset

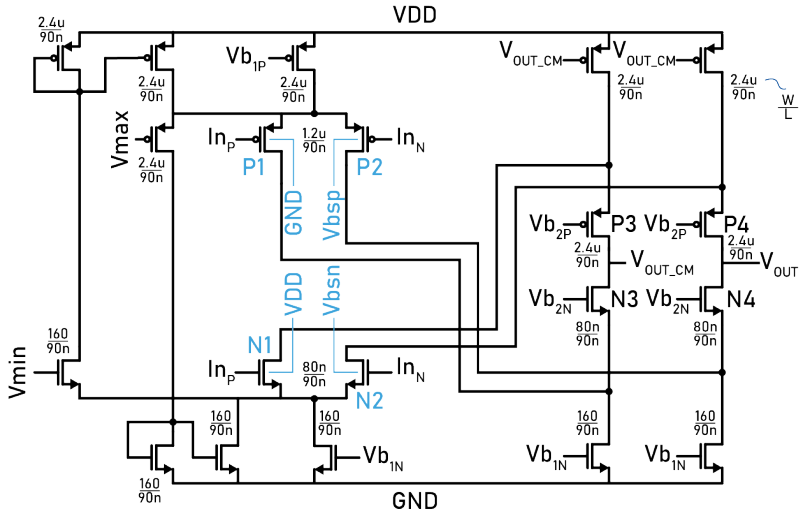


Figure 5.12: Operational amplifier schematic.

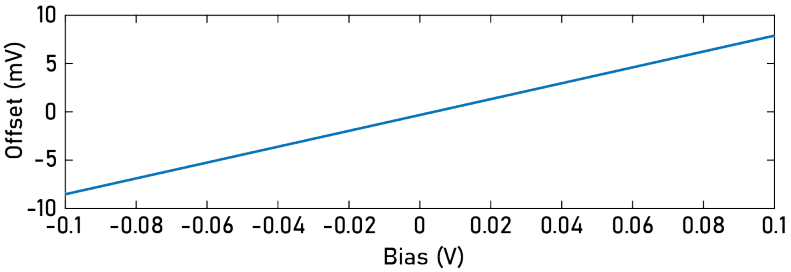


Figure 5.13: OA A Offset vs. Body Biasing Bias on V_{bsn} and V_{bsp} voltage.

is measured once by applying a voltage on VREF_DLN and measuring the output on the V_{out} pin and then subtracted from all the measurements. Each accumulation line has a reset switch to charge them to their initial value. The switch is a passgate sized to limit the leakage.

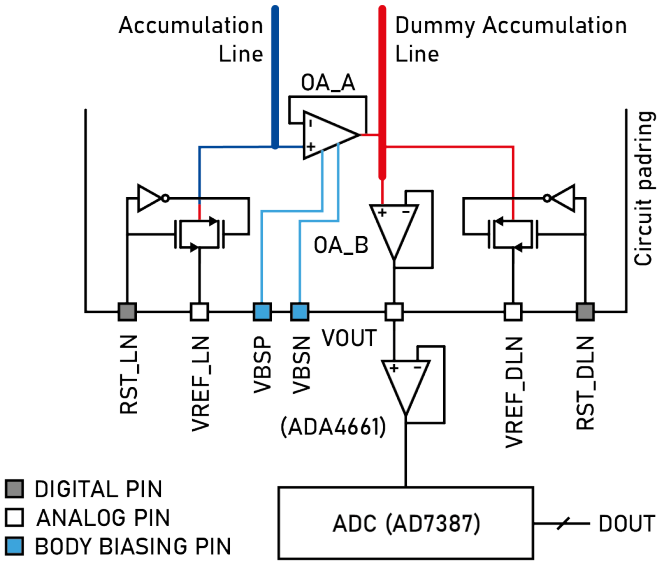


Figure 5.14: Accumulation line readout circuit.

5.5.6 Operational Amplifiers Consumption

The consumption of OA_A is simulated around 66 nW and therefore gives an efficiency of 2.97 fJ/MAC . With 11.78 μW power consumption OA_B reaches 530 fJ/MAC, but is not included in the final consumption as it is not compulsory for the macro to work, but useful for testing purposes.

5.6 Consumption Summary

The final simulated consumption is given in Table 5.5 and allows reaching 97.8 TOPS/W. It is to note that the pattern generator efficiency and the X Logic efficiency are given for a 100 by 100 array. As the signal is broadcast across all the MAC, we can see that the power con-

OA_A Power	66 nW
OA_A Energy/MAC	2.97 fJ/MAC
OA_B Power	11.78 μW
OA_B Energy/MAC	530 fJ/MAC

Table 5.4: Consumption and energy of the Operational Amplifiers.

Block	Energy Consumption
Current Source	2.05 fJ/MAC
Logic X and W	11 fJ/MAC
OA_A	2.97fJ/MAC
Pattern Generator	4.41 fJ/MAC
Total Energy/MAC	20.43 fJ/MAC
Global Efficiency	**97.8 TOPS/W**

Table 5.5: Summary of the consumption.

sumption is dominated by the Logic blocks, XLOGIC and WLOGIC, followed by the Operational Amplifier and the current sources. It is to note that the logic X and W block consumption is high mainly due to the registers used to store the inputs and weights which represent 68% of the consumption of the WLOGIC block.

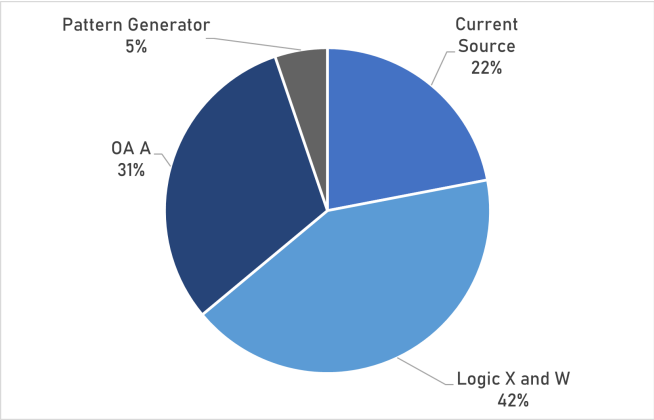


Figure 5.15: Power consumption repartition.

5.7 Corner Analysis

The behavior of one, 100 input, accumulation line in different process corners with random inputs and weights, was simulated in Cadence and is shown in Figure 5.16. The maximum final deviation with respect to the typical (TT) curve is $3\,mV$ (less than 1 LSB for a 5-bit output) for the SS corner. However, as the accumulation line is calibrated for the right current/time/capacitance ratio at start-up, it has no impact on the result value after calibration. More frequent calibration of the line can be made to mitigate also voltage and temperature dependant variations.

5.8 Conclusion

In this chapter we described the architecture of an In-Memory MVM array implementing a time- and current-based analog method. This solution exploits the available time during a computation as well as the possibility to amortize the most consuming part across the whole

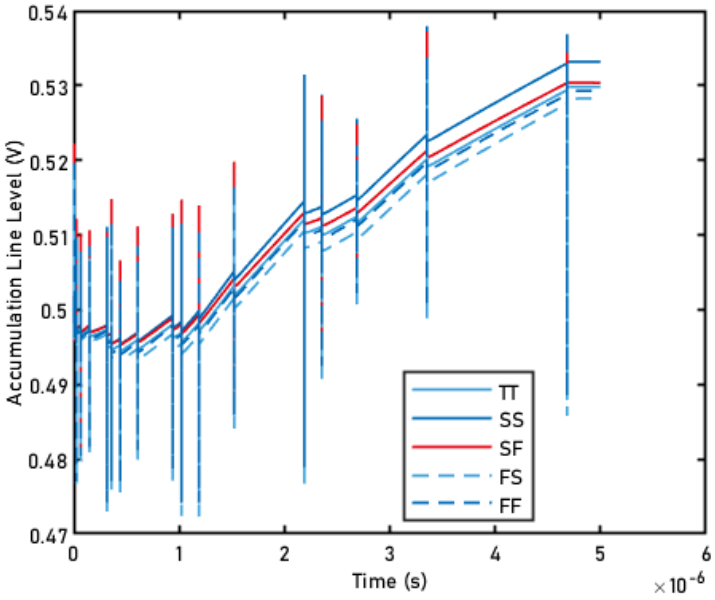


Figure 5.16: Accumulation Line Behavior with corner analysis.

array. Our solution was designed to be able to perform MVM with an error that will have a low impact on a neural network accuracy ($< 5\%$) reaching 97.8 TOPS/W. In the next chapter, the circuit will be tested. The behavior of the accumulation line will be shown, and precision and consumption is measured and compared with the state-of-the-art.

Chapter 6

IC measurement results

6.1 Introduction

In this chapter, the results of the measurement performed on the circuit are presented. The test bench and the software designed for the tests are featured in the first section. The chip characterization follows with detailed measurements of the operational amplifiers, the current sources, and the accumulation line capacitance. Furthermore, the transfer function of the system as well as the evaluation of the error is presented. The chapter ends with a summary of the metrics of the system, including power consumption and efficiency, to compare it with state-of-the-art devices. This comparison shows promising results for solutions targeting low throughput and high efficiency like preprocessing units for smart wake-up.

6.2 Test environment

The test bench, presented in Figure 6.1, is composed of three Printed Circuit Boards (PCB), an FPGA development board, and a computer running Matlab. The first one called the "daughter board" is the one that will receive the circuit. It includes 4 voltage followers (ADA4661) that isolate the accumulation line outputs from the 4 inputs multiplexed to 2 ADCs in a chip (AD7387) controlled by SPI. Finally, a connector allows the forwarding of all the signals to the second board. Called the "Motherboard", this PCB can receive 8 "Daughterboards" to parallelize computation, although this feature is not currently in use in this work. The motherboard includes all the power inputs and an SPI-controlled shift register to command the reset signals of the circuit. The body bias and reference voltages are provided by an external 10µV precision power unit for all the daughterboards connected to the motherboard. If different voltages are needed for each chip, the motherboard includes resistive trimmers that can be tuned to set voltages independently. All

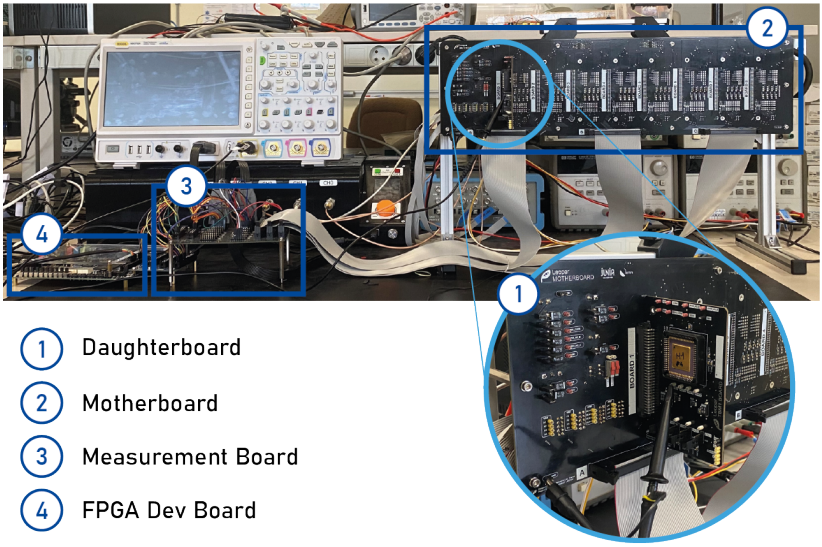


Figure 6.1: Photograph of the test bench.

the signals are then connected to the FPGA through a measurement board that allows observing signals with an oscilloscope. The FPGA features SPI to command the ADC, Shift Registers, and custom IPs to load data into the chip register. Finally, the FPGA is controlled via UART with Matlab code. All the different tools including the oscilloscope, multimeter, power units, and clock generator are controlled via Matlab as well.

6.3 Chip characterization

6.3.1 Chip presentation

Figure 6.2 shows a picture of the die, it measures 1.3mm by 1.3mm ($1.69mm^2$) but the surface of the macro is only $0.48mm^2$. To be able to measure the consumption of each block as presented in Chapter 5, we separated their supply voltage as shown in Figure 6.3.

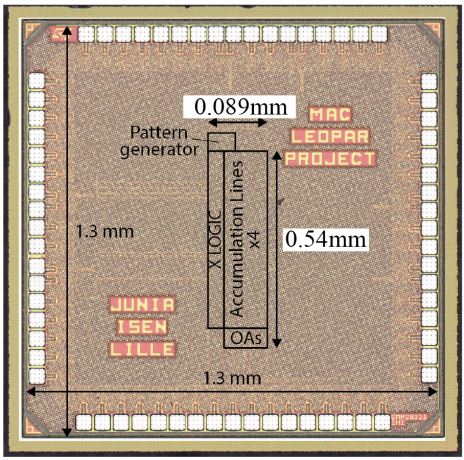


Figure 6.2: Photo of the die.

6.3.2 OA Characterization

The offset of the OA B is measured by first applying a reference voltage on the dummy accumulation line and then measuring the output on the circuit output pin with a multimeter as shown in Figure 6.4. The offset of the OA A is measured by applying a reference voltage on the accumulation line and then measuring the output on the circuit pin

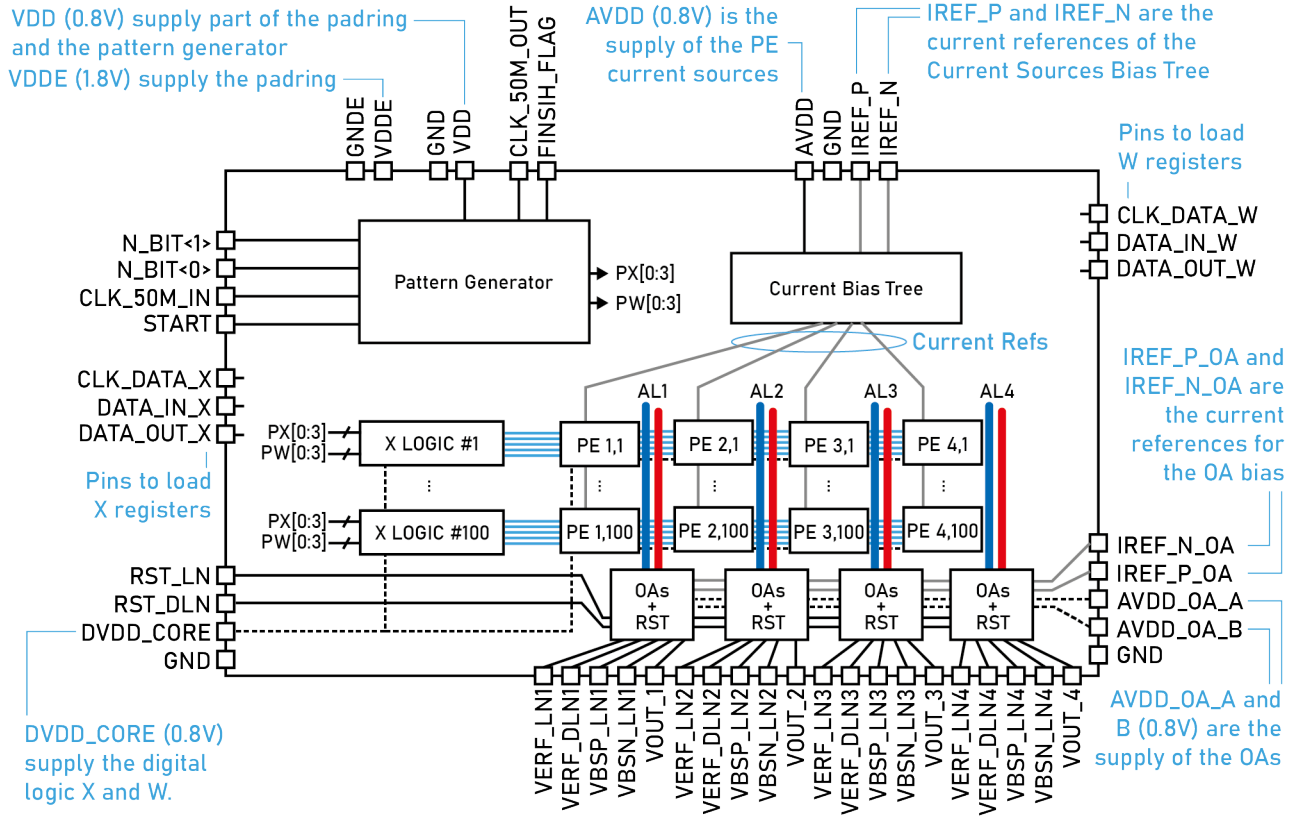


Figure 6.3: Block diagram of the circuit and its I/Os.

with a multimeter subtracting the value of OA B offset, as shown in Figure 6.5. Unfortunately, the body biasing pins for PMOS and NMOS are swapped. The body biasing applied to the PMOS is 0V but can't go lower than -0.3V which is close to the standard value. However, the NMOS body biasing is around 0V but can't go higher than 0.3V. The standard value being 1V, this is creating an offset. However, we can see in Figure 6.6 the impact of the body biasing on the NMOS differential pair of OA A. The offset measured across OA A and OA B is decreasing when VBSN is increased. Figure 6.7 shows that a body biasing value of -0.1V on the PMOS differential pair also decreases the value of the offset. The measurements with the Vbsn at 0.3 V and Vbsp at -0.1 V are presented on Figures 6.8 and 6.9 Table 6.2 shows the measured consumption for the OA A and OA B under 0.8V for a 100nA reference on the OA biasing.

OA_A Power	91.5 nW
OA_A Energy/MAC	3.29 fJ/MAC
OA_B Power	11.8 μW
OA_B Energy/MAC	531 fJ/MAC

Table 6.1: Measured consumption and energy of the Operational Amplifiers.

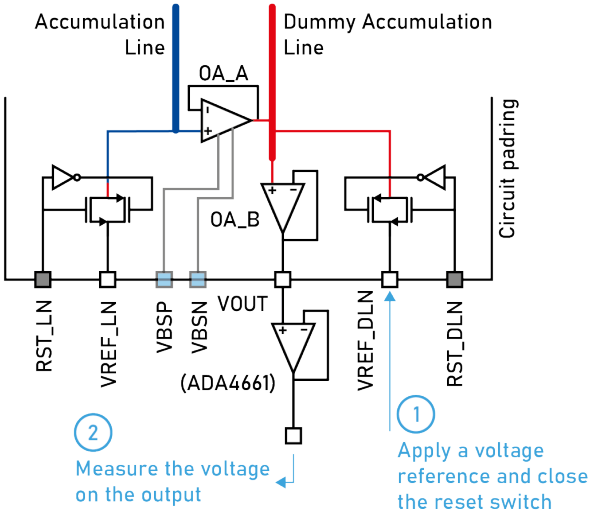


Figure 6.4: Measurement steps of OA B.

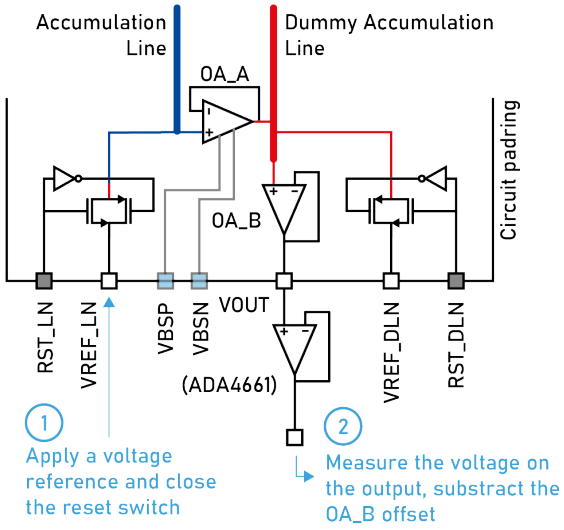


Figure 6.5: Measurement steps of OA A.

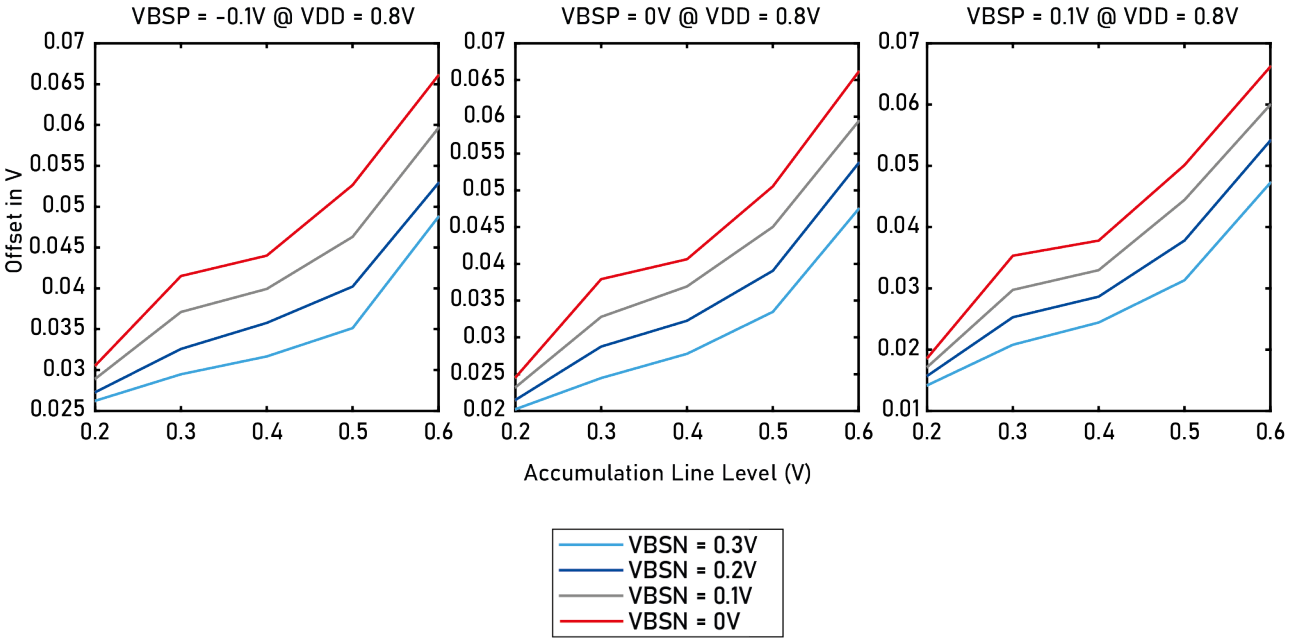


Figure 6.6: Impact of body biasing on the offset of OA A and B, body biasing only impacting OA A.

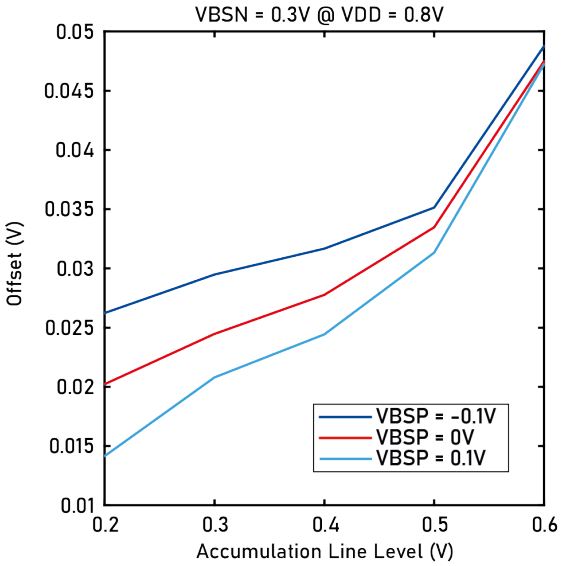


Figure 6.7: OA A and B offset for different VBSP.

6.3.3 Current source Characterization

The current sources are first characterized by measuring the current drawn on the input AVDD powering the bias tree, presented in Figure 6.10. According to the schematics, the three current mirrors are supposed to draw a current given by Equation 6.1:

$$I_{TOT} = I_{ref} + \frac{I_{ref}}{10} \times 8 + \frac{I_{ref}}{100} \times 8 + \frac{I_{ref}}{1000} \times 400 \qquad (6.1)$$

For I_{ref} =100 nA I_{TOT} is nominally 228 nA. The measure gives us 233 nA. The consumption of all the current sources is measured at 184 nW under 0.8 V.

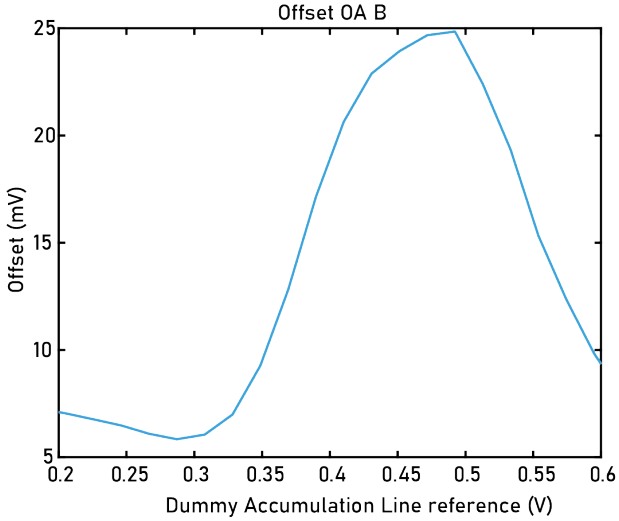


Figure 6.8: Offset of OA B.

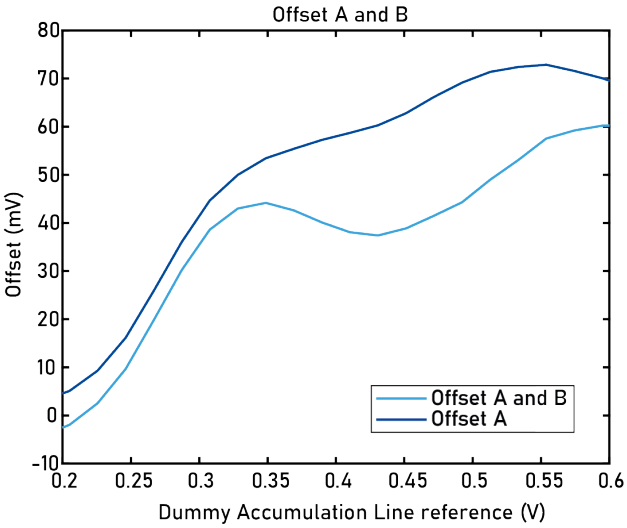


Figure 6.9: Offset of OA A.

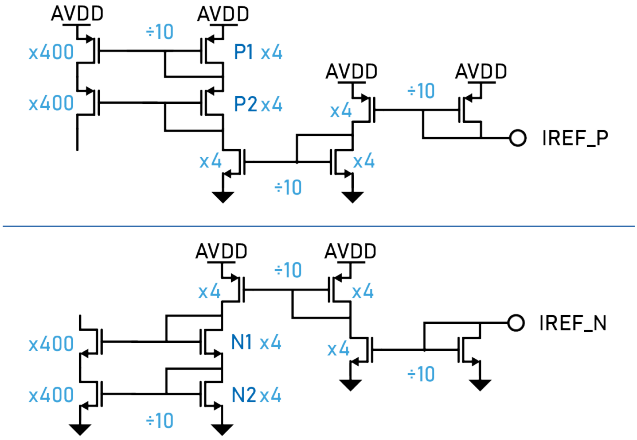


Figure 6.10: Current source bias tree.

Current Sources	184 nW
Current Sources Energy/MAC	2,07 fJ/MAC

Table 6.2: Measured consumption and energy of the Current Sources.

6.3.4 Capacitive line evaluation

The capacitive line evaluation is done by setting the clock frequency to 1MHz with a current reference reduced to 40 nA to ease the observation of the charge of the capacitive line with an oscilloscope. Figure 6.11 shows the measured output, the capacitance value that was simulated at 90 fF. At a constant current, we can compute the capacitance value with the following equation:

$$C = \frac{I \times \delta T}{\delta V} = \frac{40.10^{-9} \times 10.10^{-6}}{0.1} = 400 fF$$

The measured capacitance is approximately 400 fF. The difference between the measurement and the simulated value can come from the simulation with extracted parasitic that was performed on one PE only and then scaled to a complete accumulation line. In addition, the OA A is increasing the capacitance value. Therefore, we need to scale the current to the actual capacitance value.

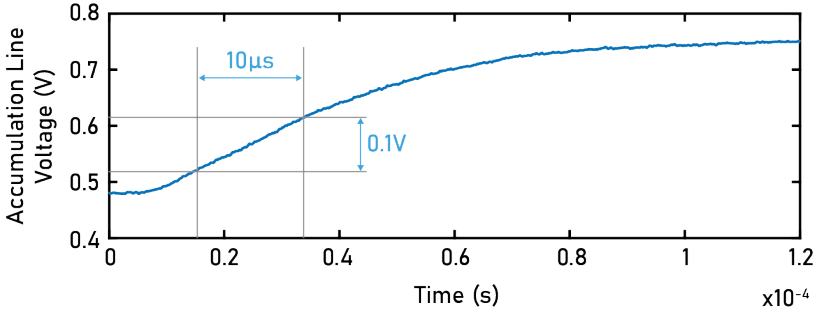


Figure 6.11: Measurement of the capacitance value for a constant current of 40 nA charging the capacitive line.

6.4 Accumulation Line behavior

To validate the functioning of the pattern generator, we needed to be able to display the different steps of the computation and compare it with the theory. We included a clock divider in the circuit to be able to check that the 50MHz external clock was correctly entering the circuit and the measurement indicates it was the case. In addition, when the computation is done, a Finish flag is raised and we can observe that the computation is taking $4.5\mu s$ as expected with a 50 MHz clock. However, it was difficult to display properly each step of the computation. Upon further investigations, it happens that the OAs are limited in slewrate and bandwidth.

6.4.1 Bandwidth limitation

Figure 6.12 shows the transfer function of the accumulation line for different clock values and a current reference of 300nA (300 pA for each current source). The scales have been adjusted for the 25MHz and 10MHz clocks to match the 50MHz product results. We can see that at 50MHz, the curve doesn't follow the other one. This can be explained by bandwidth and slew rate limitation from OA A and OA B. As the measurement is taken on the dummy accumulation line, OA A might be unable to follow the voltage of the main line fast enough. In addition, another effect happens when the current is scaled according to the clock and the capacitance value. Figure 6.13 shows that for the same frequency, with a lower current we don't reach the saturation value. This effect can be related to the current mirrors entering their triode region, as the current decreases, it prevents reaching saturation value.

6.4.2 System functioning

To be able to observe the different computation steps, we decreased the clock frequency to 1MHz and scaled the current accordingly around 10 pA. Figures 6.14, 6.15 and 6.16 show different computation compared to theoretical behaviour. We can see the effect of current mirrors entering the triode region when reaching 0.7V on the accumulation line. This measurement is obtained with the oscilloscope and transferring the acquired data in Matlab. To improve the comparison, the signal was filtered with a moving average with a window of 10 points except for Figure 6.16 where the number of points was increased. We can see that the starting point of the computation is around 0.470 V instead of 0.4 V. This is because the charge can't be kept at the desired value and goes back to the reset voltage of the accumulation line which is around 0.470 V.

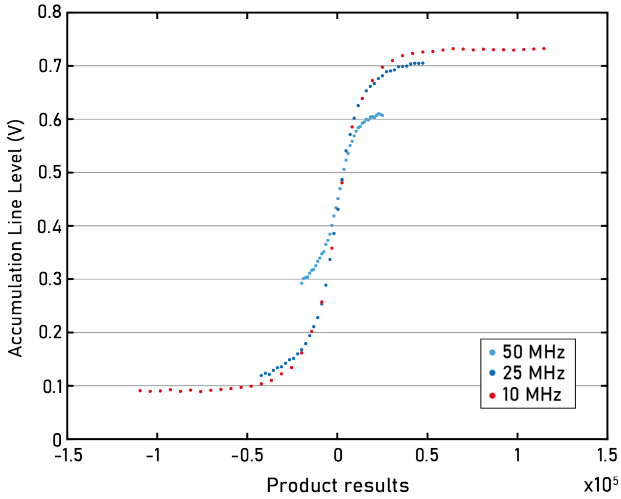


Figure 6.12: Observation of bandwidth limitations with a constant current of 300 pA for different clock frequencies.

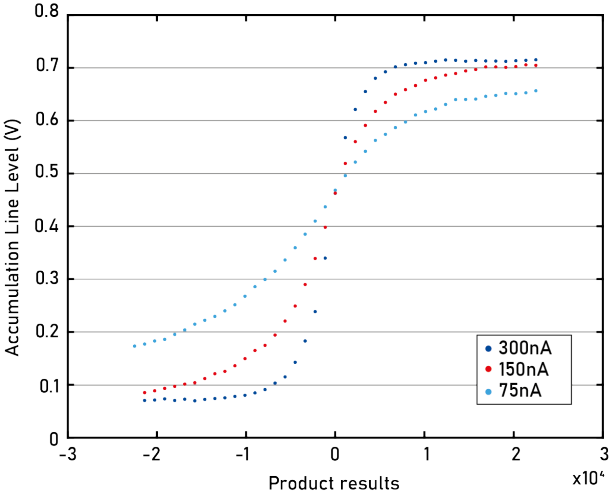


Figure 6.13: Observation of current limitations with a constant frequency of 10MHz and different reference current values.

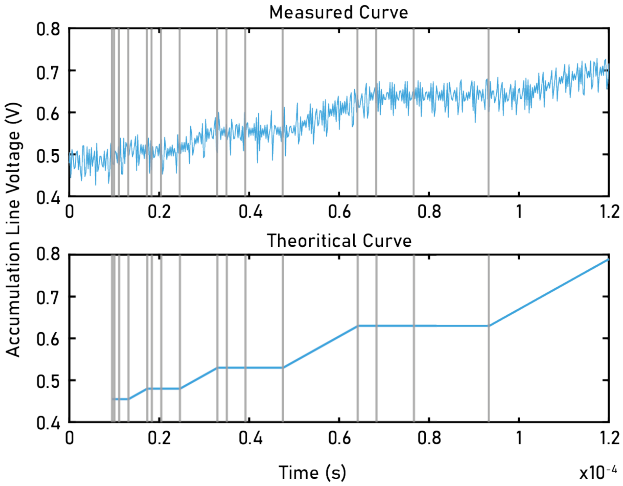


Figure 6.14: Example of computation with all the X equal to 8 and W equal to 15.

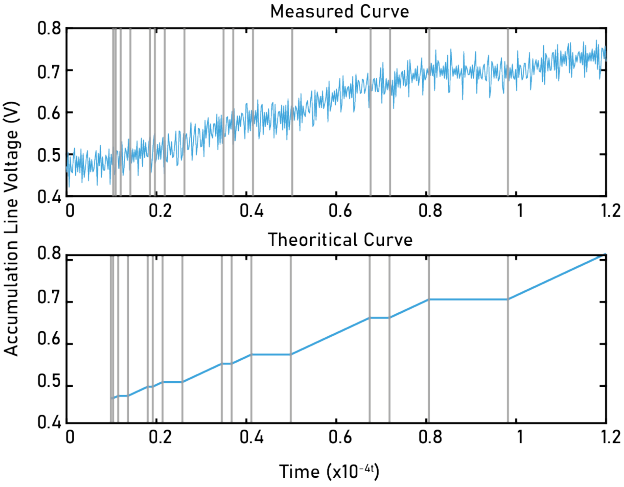


Figure 6.15: Example of computation with all the X equal to 10 and W equal to 15.

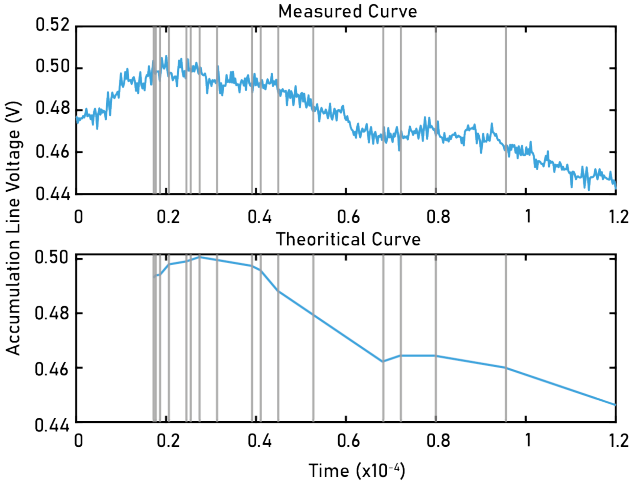


Figure 6.16: Example of computation with random values for X and W.

6.5 Error Evaluation

To be able to measure the transistor mismatch value, we will perform the same computation on each line with all the weight bits set to one and the first 10 input X values with all the bits set to one. Then we shift the X value to the next 10 inputs and so on until we cover the 100 inputs. The mismatch measured is around 0.55 % for a current reference of 300 nA (on IREF_P and IREF_N pins) which offers a better result than the simulation. It is noted the simulation was performed with a 100 nA reference.

To measure the error, we performed 200 computations with uniform distribution of random values of X and W and measured the difference between the computed and the theoretical values. By selecting uniformly distributed values for W and X, the resulting distribution is

following a Gaussian shape with a mean of 0, which corresponds to the reset voltage of the accumulation line. The error was computed for

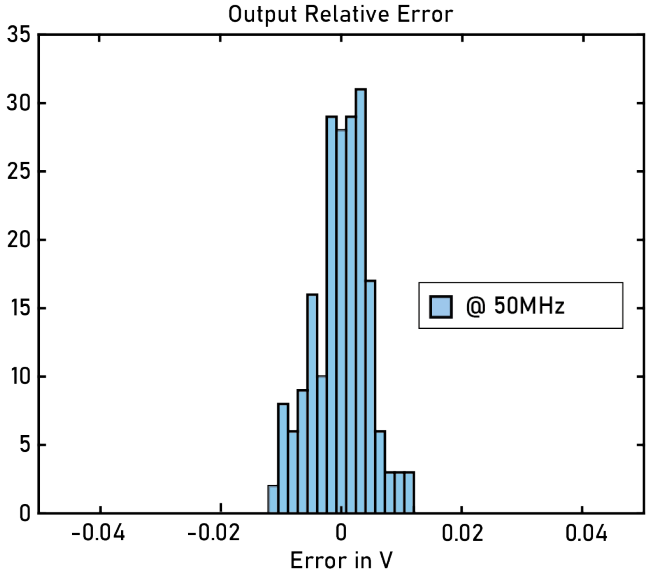


Figure 6.17: Error value.

a 50MHz clock and 300 pA current and gives a standard deviation of 4.5 mV and therefore 88 quantization levels which corresponds to a 7-bit quantized output. The values are acquired by the oscilloscope and the data are filtered with a window of 60 points. Figure 6.17 shows a histogram of the error. It is to note that filtering might improve the result. By comparison, the error at 50MHz measured with the ADC gives a standard deviation of 7.2 mV and a 6-bit precision. Furthermore, as the result of a uniformly distributed input and weight form a Gaussian

shape output distribution around 0 and therefore around 0.4 V on the accumulation line, the error due to the current mirror transistors entering their triode region doesn't have an impact. These results allow to confidently use 5-bit outputs to match the input bitwidth.

6.6 Consumption summary

The consumption of each block was measured and is summarized in Table 6.3. The circuit is powered under 0.8 V and performs 5-bit MAC operations with a time reference of 20 ns resulting in a total computation time of 4.5μs for a 100 elements accumulation line. The pat-

Block	**Energy Consumption**	
Iref	100nA	300nA
Current Source	2.07 fJ/MAC	6.2 fJ/MAC
Logic X and W	1.11 fJ/MAC	
OA_A	3.29fJ/MAC	8.46 fJ/MAC
Pattern Generator	4.41 fJ/MAC	
Total Energy/MAC	10.88 fJ/MAC	20.2 fJ/MAC
Global Efficiency	183.82 TOPS/W	99.2 TOPS/W

Table 6.3: Summary of the measured consumption.

tern generator consumption is estimated from the simulation since the block shares its voltage supply with the padring and we can't evaluate the contribution properly. We can also observe that the logic X and W blocks consume 10 times less than the simulation. The measurements are the same if performed with a source meter or a multi-

meter. One cause might be that the signal activity is lower than in the simulation, in which we are deriving the worst-case scenario. Under a 0.8 V supply, the power consumption of a single accumulation line is measured at 351 nW, including 100 cell units, the OA, and all biases. The OA accounts for 188 nW, which is half the power consumption. While performing 100 MAC operations in 4.5 µs, this core (not including the pattern generator) has an efficiency of 63.4 TOPS/W. The pattern generator consumes 9.76 µW. The total efficiency drops at 15.8 TOPS/W for a 4-by-100-accumulation line and reaches a theoretical value of 99.2 TOPS/W for 100 and 100 accumulation lines, as the pattern generator consumption is amortized on a bigger array.

6.7 Comparison with State-of-the-art

The measurement results are compared with references using different architectures for multi-bit MVM in Table 6.4. The works are taken from recent papers in ISSCC, and publications in JSSC. They target applications with input and weight precisions similar to our work (around 5-bit). This table shows the caracteristics and performances of the macro which performs MVM, except for [77]. Indeed, the RRAM array in [77] represents 89% of the total power consumption.

The works are compared according to their throughputs (GOPS), their efficiency (TOPS/W), and their input, weight, and output precision. Additional metrics are derived for comparison purposes, like the TOPS-1b/W which is found by multiplying the TOPS/W metrics by the number of bits used for inputs and weights. The GOPS/mm^2 is showing the density of the studied macro.

Compared to other works, our solution presents the highest efficiency at 2,480 TOPS-1b/W with the consumption scaled to a 10,000 MAC array. However, this result doesn't account for the ADC consumption needed for a fair comparison with the other works. In the

Reference	[ISSCC'22] [77]	[JSSC'22] [78]	[JSSC'19] [67]	[ISSCC'21] [74]	[JSSC'19][39]	[ISSCC'22] [71]	Our Work		
Technology	40nm	65nm	65nm	16nm	65nm	28nm	28nm		
MAC Operation	Digital RRAM	Digital CIM	Charge Domain	Charge Domain	Time Domain	Time Domain	Time and Current		
Supply (V)	0.9	0.9 - 1.5	0.68 - 0.94 - 1.2	0.8	0.8	0.65 - 0.9	0.8		
Macro Area (mm^2)	-	-	12.6	25	-	-	0.048		
PE Area (μm^2)	43,547	-	-	-	-	12.14	-		
# Input Channels	-	64	-	1152	64	64	100		
# Output Channels	-	64	-	256	16	256	4	100	100
Input Precision (bit)	4	2/4/6/8	1	1-8	6	4/8	5	5	5
Weight Precision (bit)	4	4/8	1	1-8	1	4/8	5	5	5
Output Precision (bit)	32	5	-	8	-	14/22	5	5	5
GOPS	94.75	2,000	18,876	11,800 (4b)	4	4,256(4b)	0.088	2.22	2.22
TOPS/W	3.79	158.7 (2b I,4b W)	866	121 (4b)	51.3	84.45(4b)	15.8	99.2[a]	49.72[a b]
GOPS/mm^2	-	-	1,498	2,670	-	-	18.34	-	-
TOPS-1b/W	60.64	1,269.6	866	1,936	307.8	1,351.2	395	2,480[a]	1,244[a b]

[a]Simulated. [b] with theoretical 8-bit ADC consumption of 10fJ/MAC

Table 6.4: Comparison with prior work

last column of Table 6.4, we added a theoretical 10pJ/conversion 8-bit ADC that adds a 10fJ/MAC energy consumption if added to our 100 input channel accumulation line. That corresponds to a theoretical 8-bit, 3.9fJ/conversion-step ADC with a sampling frequency at 222 kHz (the computation time is equal to $4.5\mu s$ with a 20 ns reference time), which is in line with state-of-the-art ADCs efficiencies [79]. We reach 49.72 TOPS/W and 1,244 TOPS-1b/W efficiency. This shows that our work still compares well with other references. Additionally, we uses flip-flop registers to store inputs and weights that can be optimized with SRAM storage that consumes less static energy and reduces the area.

For the sake of normalized comparison, we chose the TOPS-1b/W metrics but it does not reflect the actual consumption of the works for different resolutions. As we use power of two ponderations with time in our implementation, reducing the number of bits scales the computation time and efficiency logarithmically. Table 6.5 shows the theoretical efficiency of a 10,000 MAC array for different inputs and weights precision. The length of the computation is equal to $T_{tot} = (2^{n-1} - 1)^2 * T_{ref}$ with n the number of bits. Our system is even more efficient as the number of bits decreases. Due to the presence of a sign bit, the "2-bit" case corresponds actually to 3 distinct states, which is noted as "1.5 bits" in the table.

Table 6.4 shows the wide variety of architectures that can be used to reach high-efficiency MAC operations. Our work, fully optimized for larger arrays of MAC to amortize the most consuming part of the circuits, shows promising results compared to the recent state of the art and offers room for greater optimization.

For example, works [71, 74, 78] show higher efficiency. However, they exploit sparsity (setting some weights to zero) that increases the efficiency of a given application with proper training. Results presented in [71] use 50% sparsity of inputs and weights, to reach 1,351.2 TOPS/W.

Number of bits	T_{tot}	Efficiency
5	4.5μs	99.2 TOPS/W
4	0.98μs	455 TOPS/W
3	0.18μs	2,477 TOPS/W
1.5	0.02μs	22,295 TOPS/W

Table 6.5: Evolution of the efficiency compared to the number of bit.

Reference [74] exploits specific mapping and parallelization techniques for further optimization of the efficiency. [78] presents a digital CIM that exploits block-wise sparsity of activation and weights for higher efficiency. The efficiency presented in Table 6.4 corresponds to a network performing 0.63 GOPS for MNIST classification with 60.3% of weights set to zero. Our implementation can't exploit sparsity because the current steering method we use makes the current flow in the dummy line even if the weights are set to zero. The efficiency we show is therefore not application dependent. However, this is future optimization that can be made to our system to further increase efficiency.

6.8 Conclusion

In this chapter, we demonstrate the principle of time- and current-based analog IMC implemented on-chip using 28 nm FDSOI CMOS technology. The macro offers good results in terms of efficiency and accuracy although an error was introduced by an offset created by a wrong body

biasing on the NMOS pair of the OA. We can conclude by saying that this time- and current-based principle offers very promising results for low to medium-resolution AI applications. In the next chapter, we will conclude this work and present some perspectives on future work.

Conclusion and Future Work

This work shows the implementation of a time- and current-based analog In-Memory Computing macro able to reach 99.2 TOPS/W for 5-bit matrix-vector multiplications. This architecture is suitable for low-to-medium resolution embedded AI applications.

In **Chapter 1** we presented how neural networks work and show the perceptron neurons that composed layers, which are Matrix-Vector Multiplication composed of MAC operations. This operation is therefore found in lots of neural networks such as Feed Forward neural networks, LSTM and CNN. We highlighted how these networks are able to *learn* their parameters to complete a task thanks to backpropagation and gradient descent algorithms. This leads us to the different metrics and datasets used to benchmark neural networks.

We then dive into AI audio applications to present the different levels of AI computing in **Chapter 2**. The *cloud* environment allows computing complex applications with high accuracy but consumes lots of energy. *Edge Computing* computes neural networks on embedded devices but still relies on their connectivity and the cloud to compute more demanding tasks but consume a medium amount of energy. The TinyML environment represents dedicated chips for dedicated applications with ultra-low energy consumption but high accuracy. We've seen that particular architectures including smart progressive wake-up allow for reducing the embedded system energy consumption. In audio applications, the VAD algorithm allows waking up the KWS algorithm only when a voice is heard by the system. The KWS will then wake up more consuming elements when a specific keyword is found on the signal. We saw that on this specific application, Neural Networks offer great accuracy, however classical digital architectures like "Von Neumann" prevent reaching high efficiency due to the memory access cost.

In **Chapter 3**, we present In-Memory computing techniques, using digital and mixed-signal architectures. Digital architectures, although suitable for binary weights, suffer from an area overhead when working

with multibit MAC. RRAM approaches offer the advantage of using the same elements for storage and computation but don't use the standard CMOS cells, need high current, and are limited to 4 bits per cell. Charge-based IMC uses standard CMOS and exploits the WL of an SRAM bit cell but suffers from bit flipped issues that can be resolved by adding transistors to the bit cell and therefore increasing the area overhead too. Finally, we present the current and time approach that shows promising efficiency levels, although suffering from PVT that can be mitigated by training the NN.

Chapter 4 details the principle of time- and current-based IMC. We show that we can leverage the available time during a computation to charge a capacitive line with a low current (<400pA). Using parallel architecture, we can reach sufficient throughput for a wide range of applications. In **Chapter 5**, the implementation of this principle on-chip using 28 nm FDSOI CMOS technology is presented. Finally, the circuit is tested in **Chapter 6**, where we were able to show the computation principle and measure the energy consumption. The measured energy allows reaching 99 TOPS/W efficiency for a 100x100 array. The most energy-consuming part of the system relies on the size of the array to reduce its energy consumption.

Future work

The next step of this work is to include an ADC at the end of each accumulation line. Level Crossing ADCs seem particularly suited to the time- and current-based IMC. In addition, we saw that the usage of OA between the two accumulation lines increases the energy consumption per MAC as the block is amortized only across one line. The offset between the two lines doesn't seem to have that big of an impact on the accuracy of the systems. Therefore, we need to explore a configuration where we use no OAs and a level crossing ADC is resetting the

accumulation line to its resting level at each LSB crossing. The dummy accumulation line can be clamped to the resting voltage. The precision is now shifted on the charge injection and the mismatch of the current sources. However, as the accumulation line range of possible levels is now a fixed voltage plus or minus an LSB, we might not need a cascode architecture and therefore increase the size of the current mirror transistors to reduce the mismatch.

The storage of the inputs and weights also needs to be improved as the register dominates the consumption of the logic block. Using SRAM with only a writing feature on the driver will allow reducing the energy consumption. Custom SRAM blocks need to be designed as we want to place them close to the current sources. The footprint would also be reduced, allowing it to fit more weights next to a current source. Therefore, we can implement multiple layers per PE by including a multiplexer to select the weights.

Increasing the array size to 256x256 or 512x512 will allow running neural networks-on-chip and adding configurable power to the current source, which would allow using a custom array size without consuming energy.

Finally, including mapping techniques and modifying the architecture to use inputs, and weights sparsity would help to increase the efficiency for specifically trained applications.

Bibliography

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". en. In: *Communications of the ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3065386. URL: https://dl.acm.org/doi/10.1145/3065386 (visited on 01/26/2022).

[2] Paul Palomero Bernardo et al. "UltraTrail: A Configurable Ultralow-Power TC-ResNet AI Accelerator for Efficient Keyword Spotting". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39.11 (Nov. 2020), pp. 4240–4251. ISSN: 0278-0070, 1937-4151. DOI: 10.1109/TCAD.2020.3012320. URL: https://ieeexplore.ieee.org/document/9216480/ (visited on 05/04/2021).

[3] Diego Ardila et al. "End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography". en. In: *Nature Medicine* 25.6 (June 2019), pp. 954–961. ISSN: 1078-8956, 1546-170X. DOI: 10.1038/s41591-019-0447-x. URL: http://www.nature.com/articles/s41591-019-0447-x (visited on 01/26/2022).

[4] F. Rosenblatt. “The perceptron: A probabilistic model for information storage and organization in the brain.” en. In: *Psychological Review* 65.6 (1958), pp. 386–408. ISSN: 1939-1471, 0033-295X. DOI: 10.1037/h0042519. URL: http://doi.apa.org/getdoi.cfm?doi=10.1037/h0042519 (visited on 06/11/2019).

[5] Y. LeCun et al. “Backpropagation Applied to Handwritten Zip Code Recognition”. en. In: *Neural Computation* 1.4 (Dec. 1989), pp. 541–551. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1989.1.4.541. URL: http://www.mitpressjournals.org/doi/10.1162/neco.1989.1.4.541 (visited on 06/11/2019).

[6] Barry J. Wythoff. “Backpropagation neural networks: A tutorial”. en. In: *Chemometrics and Intelligent Laboratory Systems* 18.2 (Feb. 1993), pp. 115–155. ISSN: 0169-7439. DOI: 10.1016/0169-7439(93)80052-J. URL: https://www.sciencedirect.com/science/article/pii/016974399380052J (visited on 03/02/2023).

[7] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. en. In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1997.9.8.1735. URL: http://www.mitpressjournals.org/doi/10.1162/neco.1997.9.8.1735 (visited on 05/20/2019).

[8] Pete Warden. “Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition”. In: *arXiv:1804.03209 [cs]* (Apr. 2018). arXiv: 1804.03209. URL: http://arxiv.org/abs/1804.03209 (visited on 05/05/2021).

[9] *TIMIT: acoustic-phonetic continuous speech corpus.* English. OCLC: 53222255. Philadelphia, Pa., 1993.

[10] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (Nov. 1998), pp. 2278–2324. ISSN: 00189219. DOI: 10.1109/5.726791. URL: http://ieeexplore.ieee.org/document/726791/ (visited on 05/09/2022).

[11] *The Smart Audio Report*. en. URL: https://www.nationalpublicmedia.com/insights/reports/smart-audio-report/ (visited on 01/06/2022).

[12] Assaf Hurwitz Michaely et al. "Keyword spotting for Google assistant using contextual speech recognition". In: *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. Okinawa: IEEE, Dec. 2017, pp. 272–278. ISBN: 978-1-5090-4788-8. DOI: 10.1109/ASRU.2017.8268946. URL: http://ieeexplore.ieee.org/document/8268946/ (visited on 03/02/2023).

[13] Michael Price, James Glass, and Anantha P. Chandrakasan. "A Low-Power Speech Recognizer and Voice Activity Detector Using Deep Neural Networks". In: *IEEE Journal of Solid-State Circuits* 53.1 (Jan. 2018), pp. 66–75. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2017.2752838. URL: http://ieeexplore.ieee.org/document/8082747/ (visited on 12/22/2021).

[14] Arijit Raychowdhury et al. "A 2.3 nJ/Frame Voice Activity Detector-Based Audio Front-End for Context-Aware System-On-Chip Applications in 32-nm CMOS". In: *IEEE Journal of Solid-State Circuits* 48.8 (Aug. 2013), pp. 1963–1969. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2013.2258827. URL: http://ieeexplore.ieee.org/document/6519946/ (visited on 12/22/2021).

[15] Badami. "A 90 nm CMOS, Power-Proportional Acoustic Sensing Frontend for Voice Activity Detection". In: *IEEE Journal of Solid-State Circuits* 51.1 (Jan. 2016), pp. 291–302. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2015.2487276. URL: http://ieeexplore.ieee.org/document/7315025/ (visited on 12/04/2018).

[16] Marco Croce et al. “A 760 nW, 180 nm CMOS Analog Voice Activity Detection System”. In: *2020 IEEE Custom Integrated Circuits Conference (CICC)*. Boston, MA, USA: IEEE, Mar. 2020, pp. 1–4. ISBN: 978-1-72816-031-3. DOI: 10.1109/CICC48029.2020.9075954. URL: https://ieeexplore.ieee.org/document/9075954/ (visited on 12/15/2021).

[17] Minhao Yang et al. “Design of an Always-On Deep Neural Network-Based 1-μ W Voice Activity Detector Aided With a Customized Software Model for Analog Feature Extraction”. In: *IEEE Journal of Solid-State Circuits* 54.6 (June 2019), pp. 1764–1777. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2019.2894360. URL: https://ieeexplore.ieee.org/document/8693834/ (visited on 10/29/2019).

[18] Minhao Yang et al. “Nanowatt Acoustic Inference Sensing Exploiting Nonlinear Analog Feature Extraction”. In: *IEEE Journal of Solid-State Circuits* 56.10 (Oct. 2021), pp. 3123–3133. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2021.3076344. URL: https://ieeexplore.ieee.org/document/9429864/ (visited on 12/22/2021).

[19] Sechang Oh et al. “An Acoustic Signal Processing Chip With 142-nW Voice Activity Detection Using Mixer-Based Sequential Frequency Scanning and Neural Network Classification”. In: *IEEE Journal of Solid-State Circuits* 54.11 (Nov. 2019), pp. 3005–3016. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2019.2936756. URL: https://ieeexplore.ieee.org/document/8834813/ (visited on 12/22/2021).

[20] Juan Sebastian P. Giraldo et al. “Vocell: A 65-nm Speech-Triggered Wake-Up SoC for 10-μ W Keyword Spotting and Speaker Verification”. In: *IEEE Journal of Solid-State Circuits* 55.4 (Apr. 2020), pp. 868–878. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2020.2968800. URL: https://ieeexplore.ieee.org/document/8978574/ (visited on 05/04/2021).

[21] Weiwei Shan et al. “A 510-nW Wake-Up Keyword-Spotting Chip Using Serial-FFT-Based MFCC and Binarized Depthwise Separable CNN in 28-nm CMOS”. In: *IEEE Journal of Solid-State Circuits* 56.1 (Jan. 2021), pp. 151–164. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2020.3029097. URL: https://ieeexplore.ieee.org/document/9233931/ (visited on 05/04/2021).

[22] Clemens JS Schaefer et al. “LSTMs for Keyword Spotting with ReRAM-based Compute-In-Memory Architectures”. In: *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. Daegu, Korea (South): IEEE, May 2021, pp. 1–5. ISBN: 978-1-72819-201-7. DOI: 10.1109/ISCAS51556.2021.9401295. URL: https://ieeexplore.ieee.org/document/9401295/ (visited on 05/03/2021).

[23] Zhixuan Wang et al. “12.1 A 148nW General-Purpose Event-Driven Intelligent Wake-Up Chip for AIoT Devices Using Asynchronous Spike-Based Feature Extractor and Convolutional Neural Network”. In: *2021 IEEE International Solid- State Circuits Conference (ISSCC)*. San Francisco, CA, USA: IEEE, Feb. 2021, pp. 436–438. ISBN: 978-1-72819-549-0. DOI: 10.1109/ISSCC42613.2021.9365816. URL: https://ieeexplore.ieee.org/document/9365816/ (visited on 05/04/2021).

[24] Bo Liu et al. “A 22nm, 10.8 W/15.1 W Dual Computing Modes High Power-Performance-Area Efficiency Domained Background Noise Aware Keyword- Spotting Processor”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 67.12 (Dec. 2020), pp. 4733–4746. ISSN: 1549-8328, 1558-0806. DOI: 10.1109/TCSI.2020.2997913. URL: https://ieeexplore.ieee.org/document/9106775/ (visited on 05/04/2021).

[25] Boris Murmann. “Mixed-Signal Processing Opportunieties for AI”. In: *ESSCIRC 2020* ().

[26] Kevin Herisse et al. “Keyword Spotting System using Low-complexity Feature Extraction and Quantized LSTM”. In: *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. Dubai, United Arab Emirates: IEEE, Nov. 2021, pp. 1–4. ISBN: 978-1-72818-281-0. DOI: 10.1109/ICECS53924.2021.9665486. URL: https://ieeexplore.ieee.org/document/9665486/ (visited on 02/20/2022).

[27] Davis Blalock et al. *What is the State of Neural Network Pruning?* arXiv:2003.03033 [cs, stat]. Mar. 2020. DOI: 10.48550/arXiv.2003.03033. URL: http://arxiv.org/abs/2003.03033 (visited on 03/13/2023).

[28] Song Han, Huizi Mao, and William J. Dally. *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. arXiv:1510.00149 [cs]. Feb. 2016. DOI: 10.48550/arXiv.1510.00149. URL: http://arxiv.org/abs/1510.00149 (visited on 03/13/2023).

[29] Markus Nagel et al. *A White Paper on Neural Network Quantization*. en. arXiv:2106.08295 [cs]. June 2021. URL: http://arxiv.org/abs/2106.08295 (visited on 03/13/2023).

[30] Eunhyeok Park, Sungjoo Yoo, and Peter Vajda. “Value-Aware Quantization for Training and Inference of Neural Networks”. en. In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari et al. Vol. 11208. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 608–624. ISBN: 978-3-030-01224-3 978-3-030-01225-0. DOI: 10.1007/978-3-030-01225-0_36. URL: https://link.springer.com/10.1007/978-3-030-01225-0_36 (visited on 03/13/2023).

[31] Kaiming He et al. *Deep Residual Learning for Image Recognition*. arXiv:1512.03385 [cs]. Dec. 2015. DOI: 10.48550/arXiv.1512.03385. URL: http://arxiv.org/abs/1512.03385 (visited on 03/13/2023).

[32] Markus Nagel et al. “Overcoming Oscillations in Quantization-Aware Training”. en. In: ().

[33] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. arXiv:1704.04861 [cs]. Apr. 2017. DOI: 10.48550/arXiv.1704.04861. URL: http://arxiv.org/abs/1704.04861 (visited on 03/13/2023).

[34] Ron Banner et al. “ACIQ: Analytical Clipping for Integer Quantization of neural networks”. en. In: (Feb. 2022). URL: https://openreview.net/forum?id=B1x33sC9KQ (visited on 03/13/2023).

[35] Itay Hubara et al. “Accurate Post Training Quantization With Small Calibration Sets”. en. In: ().

[36] Markus Nagel and Rana Ali Amjad. “Up or Down? Adaptive Rounding for Post-Training Quantization”. en. In: ().

[37] Chunyu Yuan and Sos S. Agaian. *A comprehensive review of Binary Neural Network*. arXiv:2110.06804 [cs]. Feb. 2022. DOI: 10.48550/arXiv.2110.06804. URL: http://arxiv.org/abs/2110.06804 (visited on 03/13/2023).

[38] Wm. A. Wulf and Sally A. McKee. “Hitting the memory wall: implications of the obvious”. en. In: *ACM SIGARCH Computer Architecture News* 23.1 (Mar. 1995), pp. 20–24. ISSN: 0163-5964. DOI: 10.1145/216585.216588. URL: https://dl.acm.org/doi/10.1145/216585.216588 (visited on 03/02/2023).

[39] Avishek Biswas and Anantha P. Chandrakasan. “CONV-SRAM: An Energy-Efficient SRAM With In-Memory Dot-Product Computation for Low-Power Convolutional Neural Networks”. In: *IEEE Journal of Solid-State Circuits* 54.1 (Jan. 2019), pp. 217–230. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2018.2880918. URL: https://ieeexplore.ieee.org/document/8579538/ (visited on 06/08/2022).

[40] Caiwen Ding et al. “CirCNN: Accelerating and Compressing Deep Neural Networks Using Block-CirculantWeight Matrices”. en. In: *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. arXiv:1708.08917 [cs, stat]. Oct. 2017, pp. 395–408. DOI: 10.1145/3123939.3124552. URL: http://arxiv.org/abs/1708.08917 (visited on 03/02/2023).

[41] Changchun Zhou et al. “An Energy-Efficient Low-Latency 3D-CNN Accelerator Leveraging Temporal Locality, Full Zero-Skipping, and Hierarchical Load Balance”. In: *2021 58th ACM/IEEE Design Automation Conference (DAC)*. ISSN: 0738-100X. Dec. 2021, pp. 241–246. DOI: 10.1109/DAC18074.2021.9586299.

[42] Jun-Seok Park et al. “A Multi-Mode 8k-MAC HW-Utilization-Aware Neural Processing Unit With a Unified Multi-Precision Datapath in 4-nm Flagship Mobile SoC”. In: *IEEE Journal of Solid-State Circuits* 58.1 (Jan. 2023). Conference Name: IEEE Journal of Solid-State Circuits, pp. 189–202. ISSN: 1558-173X. DOI: 10.1109/JSSC.2022.3205713.

[43] Brandon Reagen et al. “Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators”. In: *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. Seoul: IEEE, June 2016, pp. 267–278. ISBN: 978-1-4673-8947-1. DOI: 10.1109/ISCA.2016.32. URL: https://ieeexplore.ieee.org/document/7551399/ (visited on 06/08/2022).

[44] Hidehiro Fujiwara et al. “A 5-nm 254-TOPS/W 221-TOPS/mm ^{2} Fully-Digital Computing-in-Memory Macro Supporting Wide-Range Dynamic-Voltage-Frequency Scaling and Simultaneous MAC and Write Operations”. In: *2022 IEEE International Solid- State Circuits Conference (ISSCC)*. San Francisco, CA, USA: IEEE, Feb. 2022, pp. 1–3. ISBN: 978-1-66542-800-2. DOI: 10.1109/ISSCC42614.2022.9731754. URL: https://ieeexplore.ieee.org/document/9731754/ (visited on 03/31/2022).

[45] Hyunjoon Kim et al. “Colonnade: A Reconfigurable SRAM-Based Digital Bit-Serial Compute-In-Memory Macro for Processing Neural Networks”. In: *IEEE Journal of Solid-State Circuits* 56.7 (July 2021), pp. 2221–2233. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2021.3061508. URL: https://ieeexplore.ieee.org/document/9373949/ (visited on 06/07/2022).

[46] An Chen. “A review of emerging non-volatile memory (NVM) technologies and applications”. en. In: *Solid-State Electronics*. Extended papers selected from ESSDERC 2015 125 (Nov. 2016), pp. 25–38. ISSN: 0038-1101. DOI: 10.1016/j.sse.2016.07.006. URL: https://www.sciencedirect.com/science/article/pii/S0038110116300867 (visited on 03/02/2023).

[47] Geoffrey W. Burr et al. “Phase change memory technology”. en. In: *Journal of Vacuum Science & Technology B, Nanotechnology and Microelectronics: Materials, Processing, Measurement, and Phenomena* 28.2 (Mar. 2010). arXiv:1001.1164 [cond-mat], pp. 223–262. ISSN: 2166-2746, 2166-2754. DOI: 10.1116/1.3301579. URL: http://arxiv.org/abs/1001.1164 (visited on 03/02/2023).

[48] J. M. Slaughter et al. “High density ST-MRAM technology (Invited)”. en. In: *2012 International Electron Devices Meeting*. San Francisco, CA, USA: IEEE, Dec. 2012, pp. 29.3.1–29.3.4. ISBN: 978-1-4673-4871-3 978-1-4673-4872-0 978-1-4673-4870-6. DOI: 10.1109/IEDM.2012.6479128. URL: http://ieeexplore.ieee.org/document/6479128/ (visited on 03/02/2023).

[49] I.G. Baek et al. “Highly scalable nonvolatile resistive memory using simple binary oxide driven by asymmetric unipolar voltage pulses”. In: *IEDM Technical Digest. IEEE International Electron Devices Meeting, 2004.* Dec. 2004, pp. 587–590. DOI: 10.1109/IEDM.2004.1419228.

[50] J. Müller et al. “Ferroelectric Hafnium Oxide Based Materials and Devices: Assessment of Current Status and Future Prospects”. en. In: *ECS Journal of Solid State Science and Technology* 4.5 (2015), N30–N35. ISSN: 2162-8769, 2162-8777. DOI: 10.1149/2.0081505jss. URL: https://iopscience.iop.org/article/10.1149/2.0081505jss (visited on 03/02/2023).

[51] Ali Keshavarzi et al. “FerroElectronics for Edge Intelligence”. en. In: *IEEE Micro* 40.6 (Nov. 2020), pp. 33–48. ISSN: 0272-1732, 1937-4143. DOI: 10.1109/MM.2020.3026667. URL: https://ieeexplore.ieee.org/document/9207822/ (visited on 02/25/2023).

[52] R. Khaddam-Aljameh et al. “HERMES Core – A 14nm CMOS and PCM-based In-Memory Compute Core using an array of 300ps/LSB Linearized CCO-based ADCs and local digital processing”. In: *2021 Symposium on VLSI Circuits*. ISSN: 2158-5636. June 2021, pp. 1–2. DOI: 10.23919/VLSICircuits52068.2021.9492362.

[53] Giacomo Pedretti et al. “A Spiking Recurrent Neural Network With Phase-Change Memory Neurons and Synapses for the Accelerated Solution of Constraint Satisfaction Problems”. In: *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* 6.1 (June 2020). Conference Name: IEEE Journal on Exploratory Solid-State Computational Devices and Circuits, pp. 89–97. ISSN: 2329-9231. DOI: 10.1109/JXCDC.2020.2992691.

[54] Kun Zhang et al. “Rectified Tunnel Magnetoresistance Device With High On/Off Ratio for In-Memory Computing”. In: *IEEE Electron Device Letters* 41.6 (June 2020). Conference Name: IEEE Electron Device Letters, pp. 928–931. ISSN: 1558-0563. DOI: 10.1109/LED.2020.2987211.

[55] Hao Cai et al. “Proposal of Analog In-Memory Computing With Magnified Tunnel Magnetoresistance Ratio and Universal STT-MRAM Cell”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 69.4 (Apr. 2022). Conference Name: IEEE Transactions on Circuits and Systems I: Regular Papers, pp. 1519–1531. ISSN: 1558-0806. DOI: 10.1109/TCSI.2022.3140769.

[56] Peter Deaville et al. “A Maximally Row-Parallel MRAM In-Memory-Computing Macro Addressing Readout Circuit Sensitivity and Area”. In: *ESSCIRC 2021 - IEEE 47th European Solid State Circuits Conference (ESSCIRC)*. Sept. 2021, pp. 75–78. DOI: 10.1109/ESSCIRC53450.2021.9567807.

[57] Van-Tinh Nguyen et al. “STT-BSNN: An In-Memory Deep Binary Spiking Neural Network Based on STT-MRAM”. In: *IEEE Access* 9 (2021). Conference Name: IEEE Access, pp. 151373–151385. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3125685.

[58] S. Cosemans et al. “Towards 10000TOPS/W DNN Inference with Analog in-Memory Computing – A Circuit Blueprint, Device Options and Requirements”. In: *2019 IEEE International Electron Devices Meeting (IEDM)*. San Francisco, CA, USA: IEEE, Dec. 2019, pp. 22.2.1–22.2.4. ISBN: 978-1-72814-032-2. DOI: 10.1109/IEDM19573.2019.8993599. URL: https://ieeexplore.ieee.org/document/8993599/ (visited on 05/05/2021).

[59] E. R. Hsieh et al. “Four-Bits-Per-Memory One-Transistor-and-Eight-Resistive-Random-Access-Memory (1T8R) Array”. In: *IEEE Electron Device Letters* 42.3 (Mar. 2021), pp. 335–338. ISSN: 0741-3106, 1558-0563. DOI: 10.1109/LED.2021.3055017. URL: https://ieeexplore.ieee.org/document/9336666/ (visited on 05/17/2021).

[60] Nguyen Cong Dao and Dirk Koch. “Memristor-based Reconfigurable Circuits: Challenges in Implementation”. In: *2020 International Conference on Electronics, Information, and Communication (ICEIC)*. Barcelona, Spain: IEEE, Jan. 2020, pp. 1–6. ISBN: 978-1-72816-289-8. DOI: 10.1109/ICEIC49074.2020.9051174. URL: https://ieeexplore.ieee.org/document/9051174/ (visited on 05/13/2021).

[61] Boris Murmann. “Mixed-Signal Computing for Deep Neural Network Inference”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29.1 (Jan. 2021), pp. 3–13. ISSN: 1063-8210, 1557-9999. DOI: 10.1109/TVLSI.2020.3020286. URL: https://ieeexplore.ieee.org/document/9197673/ (visited on 05/08/2021).

[62] David Lehninger et al. “Enabling Ferroelectric Memories in BEoL - towards advanced neuromorphic computing architectures”. In: *2021 IEEE International Interconnect Technology Conference (IITC)*. ISSN: 2380-6338. July 2021, pp. 1–4. DOI: 10.1109/IITC51362.2021.9537346.

[63] Cedric Marchand et al. “A FeFET-Based Hybrid Memory Accessible by Content and by Address”. In: *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* 8.1 (June 2022), pp. 19–26. ISSN: 2329-9231. DOI: 10.1109/JXCDC.2022.3168057. URL: https://ieeexplore.ieee.org/document/9758734/ (visited on 03/03/2023).

[64] Sourav De et al. “Roadmap for Ferroelectric Memory: Challenges and Opportunities for IMC Applications”. In: *2022 19th International SoC Design Conference (ISOCC)*. Gangneung-si, Korea, Republic of: IEEE, Oct. 2022, pp. 167–168. ISBN: 978-1-66545-971-6. DOI: 10.1109/ISOCC56007.2022.10031437. URL: https://ieeexplore.ieee.org/document/10031437/ (visited on 03/06/2023).

[65] Phil C. Knag et al. “A 617-TOPS/W All-Digital Binary Neural Network Accelerator in 10-nm FinFET CMOS”. In: *IEEE Journal of Solid-State Circuits* 56.4 (Apr. 2021), pp. 1082–1092. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2020.3038616. URL: https://ieeexplore.ieee.org/document/9280331/ (visited on 05/04/2021).

[66] Daniel Bankman and Boris Murmann. "An 8-bit, 16 input, 3.2 pJ/op switched-capacitor dot product circuit in 28-nm FDSOI CMOS". In: *2016 IEEE Asian Solid-State Circuits Conference (A-SSCC)*. Toyama, Japan: IEEE, Nov. 2016, pp. 21–24. ISBN: 978-1-5090-3699-8 978-1-5090-3700-1. DOI: 10.1109/ASSCC.2016.7844125. URL: http://ieeexplore.ieee.org/document/7844125/ (visited on 09/06/2020).

[67] Hossein Valavi et al. "A 64-Tile 2.4-Mb In-Memory-Computing CNN Accelerator Employing Charge-Domain Compute". In: *IEEE Journal of Solid-State Circuits* 54.6 (June 2019), pp. 1789–1799. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2019.2899730. URL: https://ieeexplore.ieee.org/document/8660469/ (visited on 06/08/2022).

[68] Shihui Yin et al. "XNOR-SRAM: In-Memory Computing SRAM Macro for Binary/Ternary Deep Neural Networks". In: *IEEE Journal of Solid-State Circuits* (2020), pp. 1–11. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2019.2963616. URL: https://ieeexplore.ieee.org/document/8959407/ (visited on 06/08/2022).

[69] Jintao Zhang, Zhuo Wang, and Naveen Verma. "In-Memory Computation of a Machine-Learning Classifier in a Standard 6T SRAM Array". In: *IEEE Journal of Solid-State Circuits* 52.4 (Apr. 2017), pp. 915–924. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2016.2642198. URL: http://ieeexplore.ieee.org/document/7875410/ (visited on 06/09/2022).

[70] Mingu Kang et al. "A Multi-Functional In-Memory Inference Processor Using a Standard 6T SRAM Array". In: *IEEE Journal of Solid-State Circuits* 53.2 (Feb. 2018), pp. 642–655. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2017.2782087. URL: http://ieeexplore.ieee.org/document/8246704/ (visited on 06/09/2022).

[71] Ping-Chun Wu et al. “A 28nm 1Mb Time-Domain Computing-in-Memory 6T-SRAM Macro with a 6.6ns Latency, 1241GOPS and 37.01TOPS/W for 8b-MAC Operations for Edge-AI Devices”. In: *2022 IEEE International Solid- State Circuits Conference (ISSCC)*. Vol. 65. ISSN: 2376-8606. Feb. 2022, pp. 1–3. DOI: 10.1109/ISSCC42614.2022.9731681.

[72] Sujan K. Gonugondla, Mingu Kang, and Naresh R. Shanbhag. “A Variation-Tolerant In-Memory Machine Learning Classifier via On-Chip Training”. In: *IEEE Journal of Solid-State Circuits* 53.11 (Nov. 2018), pp. 3163–3173. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2018.2867275. URL: https://ieeexplore.ieee.org/document/8463601/ (visited on 07/21/2022).

[73] *NN Accelerator | NICS EFC Lab*. URL: https://nicsefc.ee.tsinghua.edu.cn/network.html (visited on 10/24/2022).

[74] Hongyang Jia et al. “15.1 A Programmable Neural-Network Inference Accelerator Based on Scalable In-Memory Computing”. In: *2021 IEEE International Solid- State Circuits Conference (ISSCC)*. San Francisco, CA, USA: IEEE, Feb. 2021, pp. 236–238. ISBN: 978-1-72819-549-0. DOI: 10.1109/ISSCC42613.2021.9365788. URL: https://ieeexplore.ieee.org/document/9365788/ (visited on 02/13/2023).

[75] R. M Zur, Y Jiang, and C. E Metz. “Comparison of two methods of adding jitter to artificial neural network training”. en. In: *International Congress Series*. CARS 2004 - Computer Assisted Radiology and Surgery. Proceedings of the 18th International Congress and Exhibition 1268 (June 2004), pp. 886–889. ISSN: 0531-5131. DOI: 10.1016/j.ics.2004.03.238. URL: https://www.sciencedirect.com/science/article/pii/S0531513104006697 (visited on 03/14/2023).

[76] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[77] Muya Chang et al. “A 40nm 60.64TOPS/W ECC-Capable Compute-in-Memory/Digital 2.25MB/768KB RRAM/SRAM System with Embedded Cortex M3 Microprocessor for Edge Recommendation Systems”. In: *2022 IEEE International Solid- State Circuits Conference (ISSCC)*. San Francisco, CA, USA: IEEE, Feb. 2022, pp. 1–3. ISBN: 978-1-66542-800-2. DOI: 10.1109/ISSCC42614.2022.9731679. URL: https://ieeexplore.ieee.org/document/9731679/ (visited on 02/13/2023).

[78] Jinshan Yue et al. “STICKER-IM: A 65 nm Computing-in-Memory NN Processor Using Block-Wise Sparsity Optimization and Inter/Intra-Macro Data Reuse”. In: *IEEE Journal of Solid-State Circuits* 57.8 (Aug. 2022), pp. 2560–2573. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2022.3148273. URL: https://ieeexplore.ieee.org/document/9714739/ (visited on 02/13/2023).

[79] Boris Murmann. *ADC Performance Survey 1997-2021*. 1997. URL: http://web.stanford.edu/~murmann/adcsurvey.html..

Appendix A

Keyword Spotting System using Low-complexity Feature Extraction and Quantized LSTM

Kévin Hérissé, Benoit Larras, Antoine Frappé, Andreas Kaiser

2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)

Keyword Spotting System using Low-complexity Feature Extraction and Quantized LSTM

Kévin Hérissé, Benoit Larras, Antoine Frappé, Andreas Kaiser
Univ. Lille, CNRS, Centrale Lille, Junia, Univ. Polytechnique Hauts-de-France, UMR 8520 – IEMN
Lille, France
{name.surname}@junia.com

***Abstract*—Long Short-Term Memory (LSTM) neural networks offer state-of-the-art results to compute sequential data and address applications like keyword spotting. Mel Frequency Cepstral Coefficients (MFCC) are the most common features used to train this neural network model. However, the complexity of MFCC coupled with highly optimized machine learning neural networks usually makes the MFCC feature extraction the most power-consuming block of the system. This paper presents a low complexity feature extraction method using a filter bank composed of 16 channels with a quality factor of 1.3 to compute a spectrogram. It shows that we can achieve an 89.45% accuracy on 12 classes of the Google Speech Command Dataset using an LSTM network of 64 hidden units with weights and activation quantized to 9 bits and inputs quantized to 8 bits.**

Keywords—Keyword Spotting, Machine Learning, Long Short-Term Memory, MFCC

I. Introduction

The latest developments in consumer electronics made voice-activated devices used every day. The need to embed Keyword Spotting (KWS) solutions at the edge led to the development of always-on low-power preprocessing units to avoid the computation of the audio signal by power-hungry elements. Figure 1 shows a typical architecture, in which a preprocessing unit triggers the main processor only if the analyzed audio signal is a relevant keyword. The unit is composed of a feature extractor that will divide the audio signal into multiple frequency bands to compute the per band energy. A classification neural network uses these features as inputs to detect if the audio signal corresponds to one of the predefined classes learned by the classifier. Long Short-Term Memory (LSTM) [1] neural networks are well-suited classifiers to manage sequential data. However, LSTMs require lots of data and computational power and therefore need to be optimized for integration at the edge. This is achieved, for example, by training the network with a small number of hidden units (56 in [2]) or by quantizing weights (5-bit in [3]). These optimizations are made possible thanks to the use of input features such as Mel Frequency Cepstral Coefficients (MFCC). However, MFCC extraction requires Fast Fourier Transforms (FFT) and Discrete Cosine Transforms (DCT). For this reason, the feature extraction (FE) block usually consumes most of the energy of the system. To tackle this challenge, this paper presents the following contributions:

- A low-complexity feature extraction technique with a 16 channels filter bank with a quality factor of 1.3 was used to compute the power spectral density.

This work was supported in part by the French National Research Agency under Grant ANR-18-CE24-0006-01 LEOPAR.

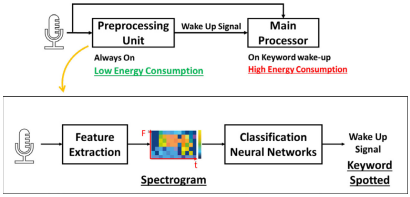


Fig. 1. Typical architecture of a preprocessing unit

- An associated optimized LSTM model with 64 hidden units post-quantized on 8 bits for inputs and 9 bits for weight/activation achieving 89.45% accuracy on recognition of 12 classes of the Google Speech Command Dataset (GSCD) [4].

The remainder of this article is structured as follows. Section II reviews different feature extraction methods and introduces the proposed filter bank together with simulations using Matlab®. Section III presents the LSTM neural network and the method to quantize it. Section IV explores the results in comparison with the state-of-the-art circuits before section V concludes this paper.

II. Feature Extraction

The GSCD is used as a reference for keyword spotting applications. It is composed of 60,000 audio files of approximately 1-second length with recordings of 31 different keywords. A common test case for comparison is to train networks using 12 classes (10 selected keywords + unknown words + silence).

A. Impact of feature extraction on the global consumption

To extract features from this dataset, state-of-the-art solutions [2], [3], [5], [6] use MFCC features. The MFCC is computed in this order: (i) FFT of an audio sample window,

TABLE I. Contribution of FE in state-of-the-art circuits

Reference	[2]	[3]	[5]	[6]
Embedded FE	FE on software	FE on software	Real FFT - MFCC	Serial FFT - MFCC
Global Consumption (µW)	5	0.5[a]	16.1	0.51
Percentage of global consumption due to FE	- (Soft.)	- (Soft.)	50%	66%

[a] Estimation from available metrics

(ii) Mel filtering using a digital high-order filter bank, (iii) computation of the log of the power for each filter output, and (iv) DCT of each computed value. The MFCCs are extracted as the amplitudes of the output spectrum. We can analyze from the literature (Table I) that the contribution of the feature extraction is more than half of the global consumption of the classification system. In [5], the authors report that the computational power is dominated by the FFT, which accounts for 72% of the total number of sums and multiplications of the FE block. To reduce the computationally expensive MFCC extraction, [7] presents a 32-channel analog filter bank employing a passive N-path filter topology consuming 800nW, while [8] introduces an event-driven approach, in which the system simulations show up to a 4000x lower consumption compared to a conventional discrete-time system.

TABLE II. FILTER BANK CONFIGURATION

Number of bands	16
Filter order	3
Frequency range	50 Hz to 5 kHz
Q	1.3

B. Proposed feature extraction architecture

The proposed FE architecture is composed of a 16-channel filter bank, with center frequencies spread from 50 Hz to 5 kHz. The quality factor Q is only 1.3, making this solution easily realizable in analog or mixed-signal domains. Table II presents the configuration of our filter bank and Figure 2 presents the frequency response of the filter bank according to Mel, Bark, and Logarithmic scales. Mel and Bark scales are perceptual scales and are created from how humans hear. Classification results using these scales are compared in section V.

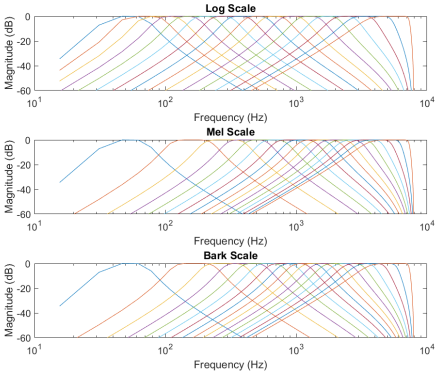


Fig. 2. 16-channel filter bank frequency response according to different scales. Mel and Bark scales are perceptual scales, used to mimic human hearing. There is less frequency bands under 100 Hz with these scales.

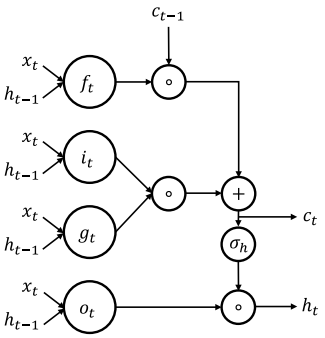


Fig. 3. Schematics of a Long Short-Term Memory neural network

When the spectral signal is divided into 16 bands, the energy in each band is calculated as:

$$E = \sum_{t=t_0}^{t_0+dt} |y(t)^2| \quad (1)$$

with $y(t)$ the filtered audio signal and dt the frame duration (set to 25 ms with an overlap of 12.5 ms). The filter bank is simulated in Matlab using Butterworth filters. There is no logarithmic scaling on the output data, meaning that when the energy is extracted from each band, it can directly be converted using an ADC and sent to the classifier.

III. CLASSIFICATION NEURAL NETWORK

A. Long Short-Term Memory

LSTM networks are a type of recurrent neural network composed of 4 intermediate sets of neurons called gates (input i_t , forget f_t, output o_t, candidate gate g_t) (2)-(5) that will compute a state vector c_t (6) that in turn is used to compute the hidden vector h_t (7). This vector will be used in the next inference together with the next input vector x_t.

$$f_t = \sigma_s(W_f x_t + R_f h_{t-1} + b_f) \quad (2)$$

$$i_t = \sigma_s(W_i x_t + R_i h_{t-1} + b_i) \quad (3)$$

$$o_t = \sigma_s(W_o x_t + R_o h_{t-1} + b_f) \quad (4)$$

$$g_t = \sigma_h(W_g x_t + R_g h_{t-1} + b_f) \quad (5)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ g_t \quad (6)$$

$$h_t = o_t \circ \sigma_h(c_t) \quad (7)$$

where W_* and R_* are weight matrices for each gate and b_* biasing values that are obtained by training the neural networks. Figure 3 shows a schematic of the LSTM architecture. The length of the states and hidden unit vectors allows storing information in time at each loop, meaning that the LSTM has the faculty to remember what just happened and therefore modify its outputs knowing this information, making this type of network a good choice for sequential tasks.

LSTM models can be stacked and are followed by one or more fully connected layers:

$$z_t = W_{fc} h_t \quad (8)$$

where W_{fc} is a weight matrix for the fully connected layer. A softmax layer is added at the end to perform the prediction as can be seen in Figure 4. This network is trained using Stochastic Gradient Descent algorithms.

B. Post-Quantization

Post-quantization techniques are introduced to perform an n-bit quantization of the LSTM model. The custom LSTM layers are described in Matlab® and are initialized with the weights obtained from the full-precision training, to accelerate the convergence. At each forward propagation, the results of equations (2)-(7) are quantized using equation (9).

$$quant(x) = round\left(\frac{x}{\max(x)} \times 2^{n-1} - 1\right) \times \frac{\max(x)}{2^{n-1} - 1} \quad (9)$$

where n is the number of quantization bits. Figure 5 shows that the weights and activation vectors follow a normal distribution. Therefore, to improve the internal representation of the system, clipping can be introduced with little impact. The introduced method consists of performing iterations with decreasing clipping values until the accuracy drops by a given amount. The clipping value associated with the maximum accuracy is eventually selected. This simple and effective method is suited to a low-complexity network that allows exploring several parameters over multiple iterations. However, for larger and computationally-intensive networks, more efficient in-training quantization methods such as PACT [9] have been developed.

IV. Simulation Results

To compose the dataset, 10 keywords are chosen from the GSCD: {"zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine"} (around 1800 samples of each word) plus 20% of other keywords from the dataset labeled "unknown" and 4,000 samples of background noise. The selected dataset is shared between training, validation, and testing datasets following this repartition: 70%, 15%, and 15%. The simulations are made with an LSTM composed of 64 hidden units. Using the computed spectrograms from the FE block as described in section II, multiple simulations are run to observe the impact of the number of bands and the input bit width on the accuracy. The per-band energy values are quantized on n bits and are then trained with the basic *LstmLayer* model from Matlab® with weights and activation coded on 32 bits. Figure 6 recapitulates those simulations and shows that there is no particular scale that stands out and would give better results than others. However, filter banks with more than 12 frequency bands offer better results than the 8 band case. An input bit width below 8 bits significantly decreases the accuracy. The best accuracy obtained for the test dataset is 90.02% for a 16-channel filter bank using a logarithmic scale with input data quantized on 8 bits. This setup is taken as a reference point for the development of the quantized LSTM model.

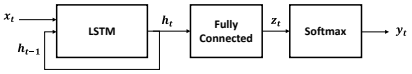


Fig. 4. Schematics of the neural network used in this paper.

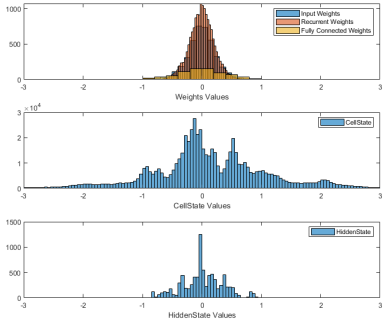


Fig. 5. Histograms showing the weights and activation functions distributions.

TABLE III. Comparison with SoA

Reference	[2]	[3]	[5]	Our Work
Feature Extraction	MFCC	MFCC	MFCC	Power Spectrum
FFT	Yes (soft.)	Yes (soft.)	DFT	No FFT
DCT	Yes (soft.)	Yes (soft.)	Yes	No DCT
Number of channels	39	40	13	16
Quantization Method	Post Quantization	In-training	Post Quantization	Post Quantization
Hidden Units	56	128	64	64
Inputs Bit width	8	5	8	8
Weights Bit width	8	5	8	9
Activation Bit width	32	8	8	9
Number of classes	4	12	12	12
Dataset	TIMIT [b]: 4 KW	GSCD: 10KW + unknown + silence	GSCD: 10KW + unknown + silence	GSCD: 10KW + unknown + silence
Accuracy	91.7%	90%	90.87%	89,45%

[b] Texas Instrument Massachusetts Institute of Technology dataset [10]

A custom LSTM model has been developed to explore the impact of quantization. Simulating our custom LSTM model with the previous setup gives a similar reference accuracy value. The code of our model and the simulation methods are available on GitHub [1]. The model allows quantizing the network to n bits using the proposed post-quantization method. The simulated accuracy is shown in Figure 7. The

[1] https://github.com/kevinherisse/leo_lstm

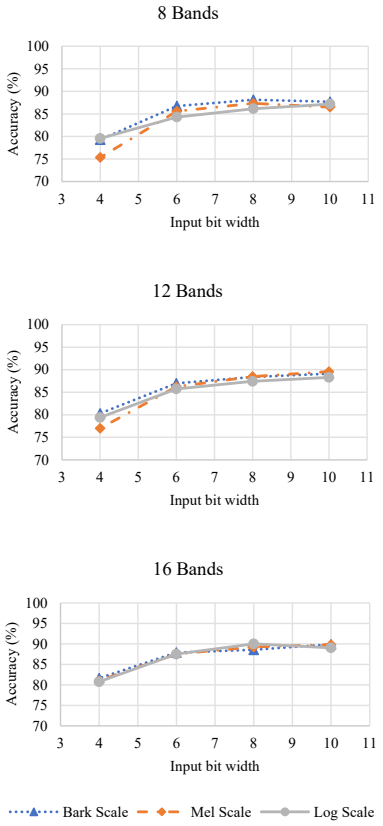


Fig. 6. Accuracy of the full-resolution network as a function of the input bit width according to different number of bands. The values are extracted as the best accuracy found over multiple trainings. The best value obtained during training is 90.02% with 16-channel and 8-bit input quantization.

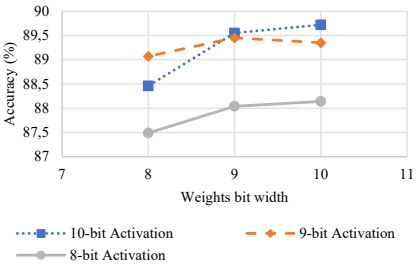


Fig. 7. Accuracy versus quantization weight bit width for multiple activation quantization

quantization only implies an accuracy drop of 0.55% for 8-bit input and 9-bit activation/weight.

Table III shows a comparison with state-of-the-art approaches. Similar accuracy is obtained while the feature extraction method is much less complex than the MFCC computation. The presented method uses a 16-channel filter bank, with third-order filters and a quality factor of 1.3 that could be implemented with low-consumption techniques such as [7] or [8].

V. CONCLUSION

This paper shows that it is possible to extract relevant audio features with a simple FE block composed of a 16-channel third order filter bank. Using a quantized 64-hidden unit LSTM model, an accuracy of 89.45% on 12 classes of the GSCD is demonstrated. These results open significant perspectives on reducing the hardware complexity of the FE function. Future work will concern the implementation of the complete processing chain and measurement of the impact on energy consumption.

REFERENCES

[1] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[2] J. S. P. Giraldo and M. Verhelst, “Laika: A 5uW Programmable LSTM Accelerator for Always-on Keyword Spotting in 65nm CMOS,” in *ESSCIRC 2018 - IEEE 44th European Solid State Circuits Conference (ESSCIRC)*, Dresden, Sep. 2018, pp. 166–169. doi: 10.1109/ESSCIRC.2018.8494342.

[3] C. J. Schaefer, M. Horeni, P. Taheri, and S. Joshi, “LSTMs for Keyword Spotting with ReRAM-based Compute-In-Memory Architectures,” in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, Daegu, Korea (South), May 2021, pp. 1–5. doi: 10.1109/ISCAS51556.2021.9401295.

[4] P. Warden, “Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition,” *ArXiv180403209 Cs*, Apr. 2018, Accessed: May 05, 2021. [Online]. Available: http://arxiv.org/abs/1804.03209

[5] J. S. P. Giraldo, S. Lauwereins, K. Badami, and M. Verhelst, “Vocell: A 65-nm Speech-Triggered Wake-Up SoC for 10-μ W Keyword Spotting and Speaker Verification,” *IEEE J. Solid-State Circuits*, vol. 55, no. 4, pp. 868–878, Apr. 2020, doi: 10.1109/JSSC.2020.2968800.

[6] W. Shan *et al.*, “A 510-nW Wake-Up Keyword-Spotting Chip Using Serial-FFT-Based MFCC and Binarized Depthwise Separable CNN in 28-nm CMOS,” *IEEE J. Solid-State Circuits*, vol. 56, no. 1, pp. 151–164, Jan. 2021, doi: 10.1109/JSSC.2020.3029097.

[7] D. A. Villamizar, D. G. Muratore, J. B. Wieser, and B. Murmann, “An 800 nW Switched-Capacitor Feature Extraction Filterbank for Sound Classification,” *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 68, no. 4, pp. 1578–1588, Apr. 2021, doi: 10.1109/TCSI.2020.3047035.

[8] S. Mourrane, B. Larras, A. Cathelin, and A. Frappe, “Event-Driven Continuous-Time Feature Extraction for Ultra Low-Power Audio Keyword Spotting,” in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, Washington DC, DC, USA, Jun. 2021, pp. 1–4. doi: 10.1109/AICAS51828.2021.9458425.

[9] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, “PACT: Parameterized Clipping Activation for Quantized Neural Networks,” *ArXiv180506085 Cs*, Jul. 2018, Accessed: Jul. 29, 2021. [Online]. Available: http://arxiv.org/abs/1805.06085

[10] *TIMIT: acoustic-phonetic continuous speech corpus.* Philadelphia, Pa.: Linguistic Data Consortium, 1993.