FeFET-based On-chip learning for Convolutional Neural Networks

CESE5000: Thesis Project

Lars Hoogland

Delft University of Technology

TU Delft

FeFET-based On-chip learning for Convolutional Neural Networks

by

Lars Hoogland

Computer Engineering: Computer Architecture
Student Number: 4490363
Supervisor: Professor Said Hamdioui
Duration: June 2024 - August 2025

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended on Monday August 25, 2025 at 14:00 PM

Committee members:

Prof. dr. S. Hamdioui,	TU Delft, supervisor
Dr. A. Gebregiorgis,	TU Delft, supervisor
Dr. D. Muratore	TU Delft
T. Spyrou, Ph.D.	TU Delft
A. Mahmoud, Ph.D.	IMC

TU Delft

Preface

This M.Sc. thesis concludes my time as a Computer and Embedded Systems Engineering student as well as my full studies at the TU Delft. From the first moment I learned about Computation-In-Memory I knew I wanted to investigate and design a CIM system for my master thesis and throughout the process of this thesis my interest has only grown. I hope this work conveys my enthusiasm to you, the reader, and that you will enjoy learning about CIM as much as I do.

I would like to sincerely thank my supervisor Prof. Said Hamdioui for his feedback and advice during our few progress meetings. I would also like to thank my daily supervisors Anteneh Gebregiorgis, Theofilos Spyrou and Abdulqader Mahmoud for their continued support during our weekly meetings and for bringing me into contact with fellow students who could help me with detailed questions. Additionally, I want to thank Ph.D students Yash Biyani amd Emmanouil Arapidis for their assistance and answering my each and every question.

Furthermore, I want to thank my family and friends for supporting me throughout my research, providing me with the tools to keep working throughout the entire thesis process. Finally I would like to thank Cathelijne, for always being by my side.

Lars Hoogland
Delft, August 2025

Abstract

Modern Artificial Intelligence (AI) applications, such as Deep Neural Networks (DNNs), require substantial amounts of data in order to carry out the classification or recognition task, which must be retrieved from the memory, supplied to the processor, and finally the results stored back in the memory. In Von-Neumann architectures, this data movement incurs significant performance costs, leaving the CPU with many idle cycles while waiting for data to arrive. One way of addressing this issue is by investigating alternative computing paradigms, such as Computation in Memory (CIM). In CIM architectures, the processor and the memory are integrated into one physical location. As such, computations are performed in the memory core directly, without the need to be transferred to a central processor. A promising technology to efficiently implement CIM crossbar arrays is the emerging Ferroelectric Field Effect Transistor (FeFET), in which data can be stored in a non-volatile manner in the polarization state of a ferroelectric layer.

In existing literature, CIM crossbar arrays are optimized for the inference task, but do not perform the learning task locally. This means the neural network is trained externally, for example using cloud computing. Only once the training is finished, the weights are written to the physical crossbar array. For medical applications, such as ECG classification, sending sensitive medical data off to the cloud for training leads to privacy concerns. A solution to this problem is On-chip learning: training the network locally in the crossbar itself.

This thesis focuses on integrating the FeFET technology in a CIM architecture to design a crossbar array that supports On-Chip learning for Convolutional Neural Networks. The accelerator overcomes the memory-wall inherent to Von Neumann machines by embracing the CIM framework and uses FeFET devices to overcome the scaling walls associated with CMOS technology. The result is a novel accelerator which leverages the parallelism of Analog Crossbars to optimize the inference task and forward propagation, while leveraging the accuracy of Digital Crossbars to optimize the back propagation task.

Contents

List of Figures

List of Tables

1 Introduction

1.1. Motivation

Modern computing applications are becoming increasingly data-intensive. With the widespread integration of Artificial Intelligence (AI) workloads in fields such as healthcare, social media and security [1], big-data applications are pushing existing computing platforms to their limit. During the execution of these workloads, massive amounts of data have to be transferred back and forth between the processor and the memory.

Today's computing systems, such as the computers we have at home and the phones we carry around in our pockets, are based on the Von-Neumann architecture. These systems consist of CMOS-based digital hardware, including a processing unit and memory. Data is stored in the memory and whenever the processor requires a piece of data to execute a task, it fetches that piece of data from the memory, performs its computation, and writes the results back to the memory. Figure 1.1 displays a simplified illustration of Von-Neumann architecture-based computing systems.

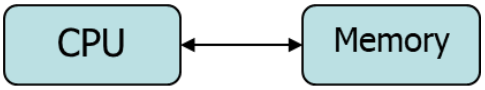

Figure 1.1: Illustration of Von-Neumann architecture

For neural network applications, the core computation being performed is the Multiply and Accumulate operation (MAC) between the inputs and weights of the network. When performed on Von-Neumann hardware, these vector-vector multiplications are performed by sequentially fetching each pair of vector elements from memory and multiplying them in the processing unit. Afterwards, an addition operation is performed to add the intermediate result to the previously stored result, requiring additional data retrieval and storage. The latency and energy costs associated with this data movement dominate the total latency and energy when compared to those of the processing element.

As such, for big data applications, these traditional Von-Neumann computing systems suffer from what is known as the *memory-wall* [2]. Due to the increasing gap between processor and memory speeds, the rate at which data is supplied to the processor is lower than its maximum throughput. This leads to the processor having many idle cycles, during which it is waiting for more data to arrive. This means the processor is not used in an efficient manner, having very low utilization. As such, the communications between processor and memory form a major bottleneck in terms of both power and latency.

Thus, alternative computer architectures are required that aim to overcome not only the memory wall that is inherent to the architecture, but also the three scaling walls that threaten CMOS-based systems. One of these alternative computer architectures is Computation-In-Memory (CIM). In a CIM-based architecture, the processor and memory are combined and integrated into one physical location, allowing arithmetic and logic operations to be performed in-memory. This means there is no need for communication between memory and a separate processor, thus overcoming the memory wall that is inherent

to Von-Neumann systems.

1.1.1. Demand: On-Chip Learning

In order to use a neural network for inference, it has to be trained beforehand. For existing CIM solutions, the Neural Network is trained off-chip [3] [4]. This means the training data for the network is sent off to a cloud computing service. There the network is trained in software to find the optimal weights. Then, only once the network is fully trained, the corresponding weights of the network are written to the physical CIM crossbar. From then on the weights are not updated again and the resulting NN accelerator is used for the inference task. Training externally is very fast and can be done at low cost, but has a major downside in terms of security. That is, if the data used to train the network is sensitive, sending this data off to a cloud computing service can pose a security risk. Firstly, a potential threat could extract the provided data, thus breaching the user's privacy. Secondly, a potential threat could manipulate input data by means of an injection attack to cause the model to malfunction. This can have major consequences in applications such as healthcare or autonomous driving [5].

An example of a sensitive data application that uses neural networks is Electrocardiogram (ECG) heartbeat analysis. Instead of manually detecting deviating heartbeats, a Neural Network can be used to efficiently classify individual heartbeats. This allows for a faster and more accurate diagnose of medical conditions such as cardiovascular disease. In such an application, the training data consists of sensitive medical data. If an external cloud computing service is used to train the network, the data has to leave the secure network of the hospital. As such, safety of the data is not guaranteed and bad actors could potentially access or steal the data, both in the cloud computing service and in transit. This means that off-chip learning is not suitable for sensitive data applications and a local method of training the network is required instead. On-Chip Learning refers to locally training the neural network on the same physical hardware that is used for the inference task. Existing CIM literature focuses exclusively on the inference task and disregards the learning task altogether. As such, there is a research gap for CIM crossbars that support On-Chip learning.

1.1.2. Opportunities

Computation-in-Memory (CIM)

A CIM crossbar is particularly suited for implementing the MAC operation. To implement a MAC operation on a CIM architecture, a crossbar of memory elements is used, in which the stored value in each memory element corresponds to one operand and the applied input voltages correspond to the other element. Through simple current laws, it can be shown that the output current generated in a column of the crossbar corresponds to a single Vector-Vector Multiplication. This is further explained in Section 2.2. A conventional CIM architecture is depicted in Figure 1.2.

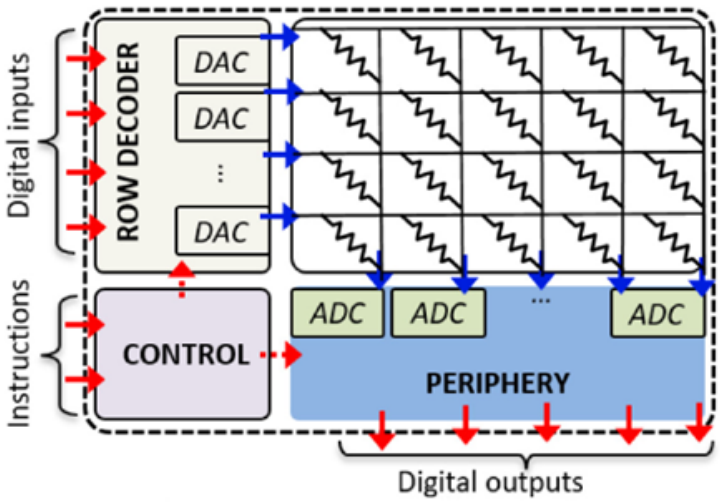

Figure 1.2: CIM Crossbar. Adapted from [6].

FeFET technology

Besides the challenges posed by the traditional Von-Neumann architectures, the CMOS technology on which modern computing systems are based has its own challenges. CMOS technology is running into several scaling walls of its own [7]. With the end of Moore's Law approaching or arguably already having been reached, static power consumption is becoming dominant at smaller technologies (*Leakage-wall*), technology scaling leads to reduced device lifetime (*Reliability-wall*) and the cost per device is plateauing (*Cost-wall*). As such, when designing an accelerator for big data applications, a lot can be gained from proper selection of both the computer architecture and memory technology.

A promising alternative for CMOS technology is the emerging FeFET device. A FeFET device is a three-terminal memory device which has a structure very similar to that of a MOSFET. The main difference is the insertion of a layer of ferroelectric material in the gate stack, between the gate metal and the dielectric. A representation of the FeFET device can be found in Figure 1.3.

A ferroelectric material is a material which, when exposed to an electric field, will retain its polarization, even after the electric field is removed [8]. By polarizing this ferroelectric layer in either direction, the threshold voltage that determines the conduction properties of the FeFET can be altered. This means that instead of storing data as charge, as is done for SRAM and DRAM, in a FeFET data can be stored as the polarization state of the ferroelectric layer. leading to a non-volatile memory that retains its value even if it loses power. The workings of a FeFET are further explained in Section 2.3.

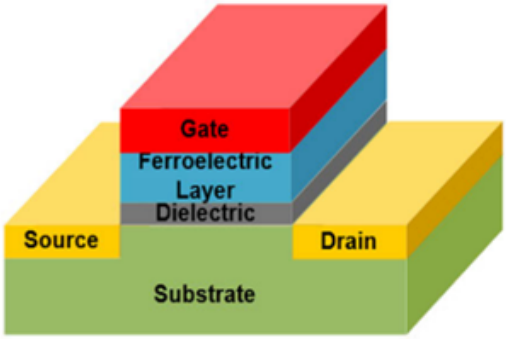

Figure 1.3: FeFET device. Adapted from [9].

1.2. Problem statement

Communication of neural network training data over the internet poses security risks for biomedical applications. Sensitive personal data could be accessed by potential threats, violating the privacy protection that biomedical circuits should offer. Additionally, bad actors can use injection attacks to cause the device to malfunction. With Artificial Intelligence becoming more widespread and being integrated in more medical processes, we are becoming more reliant on AI systems and we can not allow them to malfunction. As such, there is need for a method of training Neural Networks locally, to ensure proper functioning of the circuit and keeping the user's privacy intact.

Therefore, the aim of this project is to design a FeFET-based CIM architecture that supports local On-chip learning in order to address the safety concerns associated with communicating neural network training data over the internet. This accelerator should overcome the memory-wall inherent to Von-Neumann based computing systems by utilizing the CIM paradigm. Additionally, the design should overcome the scaling walls associated with CMOS technology by utilizing FeFET devices as memory devices.

1.3. Contribution

In this work, a FeFET-based CIM crossbar accelerator will be presented that supports On-chip learning. The architecture is designed to accelerate Convolutional Neural Networks (CNN) and allow for on-the-fly weight updates. The work will be benchmarked using the MIT-BIH datset for ECG signal classification.

The contributions of this work are the following.

- **Investigating On-chip Learning for CIM.** On-chip learning is a sparsely researched field, despite having many possible use cases for networks that are trained on sensitive data. Any network thats trains on sensitive data can benefit from training On-chip rather than sending their data off to the cloud.
- **Designing a CIM array using emerging FeFET devices.** The FeFET device is an emerging memory device with great potential for Computation-In-Memory architectures.
- **Increasing accuracy of a network using On-chip Learning.** By training on-chip, the network can be further trained whenever new data is available. This will increase the accuracy of the network.
- **Implementing On-chip Learning in a energy and area efficient manner.** For medical applications, the edge devices on which neural networks run are small wearable devices. These devices are small and have limited battery size. As such, are and energy are the most important performance indicators.

The main benchmarks for this work are the energy consumption and area. The reason for this is that for healthcare applications, data is generally collected using wearable devices that are worn by the patients. The most efficient way of implementing a local NN accelerator is to integrate it directly into these wearable devices. These devices should be small enough to not interfere with the patient wearing it and thus these devices have limited resources. This means these devices generally have both small batteries and small amounts of area available.

1.4. Outline

The remainder of this thesis is organized as follows:

- **Chapter 2:** In this chapter an overview of background information is given. Firstly, the principles of Convolutional Neural networks are explained and an overview of applications for which CNN is suitable are given. Next, an overview of Computation-In-Memory architecture is given. Finally, the FeFET device technology is explained.
- **Chapter 3:** In this chapter the design and implementation of the proposed CIM architecture is explained. It starts with an overview of the design for the accelerator and then explains the design of the bitcells and the crossbars used. Next, a novel method for scaling up crossbar netlists using python is presented. Finally, the periphery circuits used in the design are explained.
- **Chapter 4:** In this chapter the results of the proposed crossbar are given. It begins by verifying the behaviour of various bitcell types and then shows area and energy results. Next, a crossbar without periphery is verified and its area and energy results are presented. Finally, the behaviour of the On-Chip learning periphery is verified and area and energy results for the full system are presented.
- **Chapter 4.8:** This chapter provides an interpretation of the obtained results and compares the resulting accelerator with the contribution goals outlined in Section 1.3.
- **Chapter 5:** The final chapter provides a conclusion for the thesis and gives recommendations for future work.

2

Background Information

2.1. Convolutional Neural Networks

There is a wide range of tasks that are traditionally easy for humans to perform, but very difficult for computing systems to perform through explicit programming. For tasks such as image recognition, statistical analysis and data modeling, Artificial Neural Networks (ANNs) have become a widely employed tool, being used to solve complex problems in almost every domain, including image recognition, security and healthcare.

The design of ANNs is inspired by the way the human brain operates. The ANN consists of a series of layers, each of which consists of a set of neurons. Any neuron is fed by a set of one or more real-valued inputs through a series of weighted links. When a neuron is fed by an input, the input value is multiplied by the weight of the corresponding link. The neuron accumulates the value from all its weighted inputs and passes it to an activation function, which brings non-linearity to the resulting output. The basic structure of an ANN is displayed in Figure 2.1.

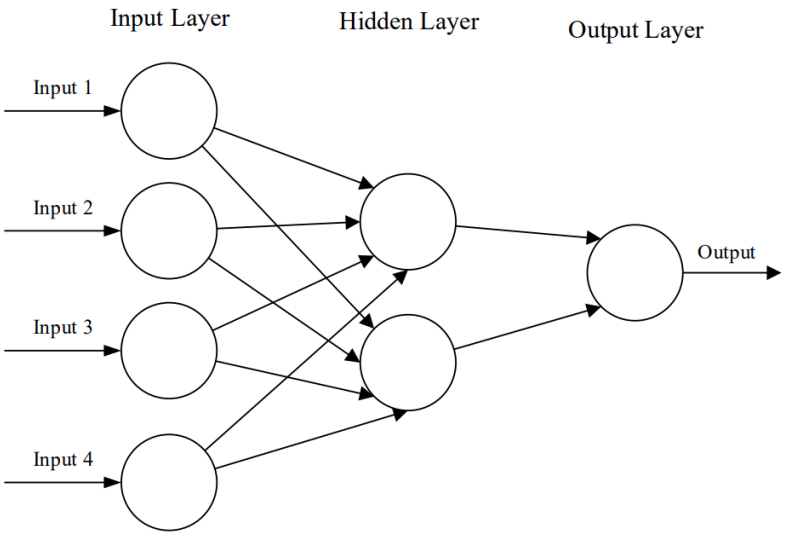

Figure 2.1: Basic structure of an ANN. Adapted from [10].

ANNs have two main tasks: The inference task and the learning task. The learning task refers to training the neural network to perform a given task. The inference task refers to utilizing the Neural Network to performing the given task for an input that was not part of the training data.

2.1.1. Learning Task

In order for an ANN to perform a given task, it needs to be trained. Training refers to updating the weighted connections between the neurons in order to optimize the correctness of the given output for a certain input. Training the network is done iteratively. At each iteration, small changes are made to weights to bring them closer to the desired values. Determining how to bring them closer to the desired values is done based on the selected learning paradigm.

There are two main learning paradigms. Supervised learning refers to learning by feeding labeled inputs to the network. An input is fed to the network and the Neural Network generates a corresponding output. The resulting output label is then compared against the label associated with the inputs. Based on the difference between these two labels, the weights of the Neural Network are updated. On the other hand, unsupervised learning does not use labeled data. Instead, the Neural Network tries to optimize a certain predefined cost function.

2.1.2. Inference Task

Once a network is fully trained using any of the above methods, the network will be able to provide an output for any set of input data, even for inputs that were not part of the training data. Letting the neural network generate an output for a set of inputs is known as the inference task. A schematic representation of inference for a single neuron is displayed in Figure 2.2

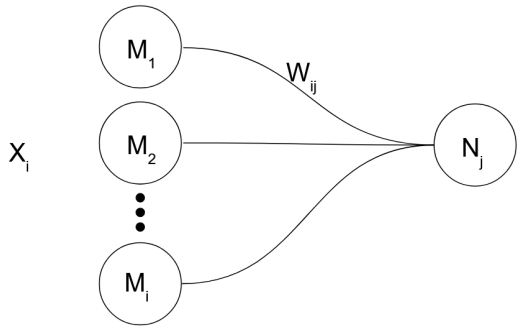

Figure 2.2: Representation of weights and inputs for a single neuron

When performing the inference task, any neuron N_j in a layer of the Neural Network is fed by a set of neurons M_i in the layer before it. Every connection has a corresponding weight W_{ij}. To calculate the value Y_j of neuron N_j, the value X_i of every input neuron M_i has to be multiplied by the corresponding weight w_{ij} and these intermediate results have to be summed. The result is then supplied to an activation function, which provides non-linearity to the network. This calculation follows Equation 2.1.

$$Y_j = f(\sum_{i=0}^{k} X_i W_{ij}) \tag{2.1}$$

where f is the activation function and k is the amount of nodes in the previous layer connected to node N_j.

This operation of multiplying two numbers (in this case a weight and an input) and adding the result to a running total is known as a Multiply and Accumulate (MAC) operation. Because the MAC operation is this inherent to the inference task, it is one of the most common and thus most important operations for any AI workload. Thus, the computer architecture on which the AI workload runs should be optimized for the MAC operation.

Convolutional Neural Networks (CNN) are a subset of Neural Networks that operate on data stored in a grid pattern. CNNs are designed to learn spatial hierarchies within the training data. Because images and video are inherently 2d representations of data, CNNs are generally used for the purposes of video and image processing. Convolutional Neural Networks have three types of layers: convolutional layers, pooling layers and fully-connected layers.

The input data for a convolutional network is stored in a grid. Convolutional layers consist of a kernel that traverse this input data. The kernel takes the form of a small array of numbers and performs an element-wise product between the stored numbers in the kernel, and the input values at the kernel's current position. The resulting products are summed up in order to obtain an output value, which is known as a feature. Thus, the Multiply and Accumulate operation is core to CNNs: A set of weights (in this case, the values stored in the kernel) and inputs are multiplied and the result is summed to generate a single feature.

Afterwards, the kernel is shifted over by a number of positions (this is known as stride) and applied to the next set of inputs. By repeating this process until the kernel has traversed all input data, a full feature map is generated. This process is depicted in Figure 2.3.

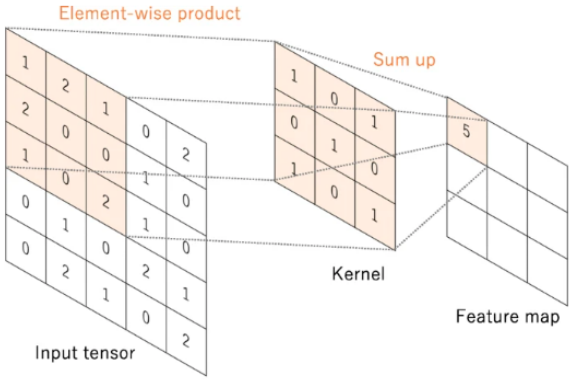

Figure 2.3: Convolutional layer. Adapted from [11]

The resulting feature map from the convolutional layer will be nearly as large as the original input tensor, or in the case of zero-padding, the same size as the original input tensor. This means the amount of learnable parameters is big and thus a small shift in any of these parameters can severely influence the CNN's performance. For this reason CNN's contain pooling layers. Pooling layers are used to reduce dimensionality of the feature map. This is done by having a pooling kernel traverse the feature map. As opposed to the convolutional kernel, the pooling kernel does not have any values stored inside of it and only looks at the data it traverses. The most commonly used pooling layer is max pooling, which extracts the maximum value of the subset of inputs that the kernel is currently covering and saves it to the output. By repeating this process until the kernel has traversed all input data, a full feature map is generated. This process is depicted in Figure 2.3.

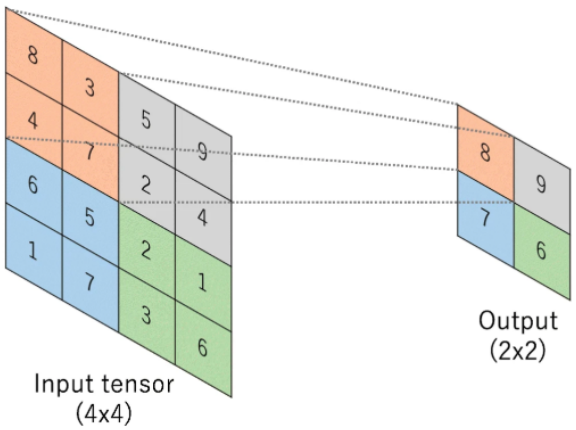

Figure 2.4: Pooling layer. Adapted from [11]

The combination of convolutional and pooling layers allows for the extraction of meaningful features while keeping dimensionality low and thus making the CNN robust and invariant to small shifts. How-

ever, this feature extraction does not allow for classification of the resulting features. Instead, classifying a set of extracted features requires a final fully-connected layer. Fully-connected layers are layers in which every neuron in one layer are connected to every neuron in the next layer. These are identical to the fully-connected layers described in section 2.1.2.

2.1.3. Applications for Convolutional Neural Networks

One application for Convolutional Neural Networks is Electrocardiogram (ECG) analysis. ECGs represent the electrical activity of the human heart and are used in the medical field to diagnose cardiovascular disease (CVD). Dectecting CVD in an early stage is challenging because of unobvious symptoms and very short duration of these symptoms. [12].

Generally, heartbeats are measured by on-body sensors, which in turn generate the ECG [13]. The ECG is then manually investigated and analyzed by a medical professional. An example heartbeat is displayed in Figure 2.5.

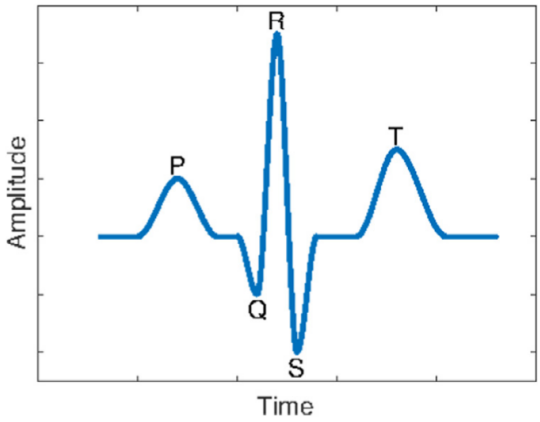

Figure 2.5: Regular heartbeat ECG. Adapted from [14].

Convolutional Networks are particularly suited for ECG analysis [14]. Ideally, the hardware on which the CNN runs should be embedded in the wearable on-body sensors that the patient is already equipped with. These devices should be small enough to not interfere with the patient wearing them and thus these devices have limited resources. This means these devices generally have both small batteries and small amounts of area available. This makes area and energy efficiency the main parameters the design should focus on. An efficient way of implementing Neural Networks in a small area is the use of Computation-In-Memory paradigm.

2.2. Computation-In-Memory

Computation-In-Memory (CIM) or In-Memory-Computing (IMC) is a computer architecture in which the Memory and Processor are integrated into one physical location. Instead of requiring communication between the two devices, computations can be performed directly in-memory. This architecture thus overcomes the communications bottleneck (Memory Wall) inherent to Von-Neumann architecture. CIM was originally invented by IBM in 1970 as a potential solution for embedded systems and big data applications, but the technology had fallen out of favor because of the limitations of DRAM technology. Recently, the architecture has seen a resurgence, because of its potential for AI workloads using emerging non-volatile memories.

Computation-In-Memory is facilitated by the usage of Non-charge-based memories, such as the memristor. In traditional charge-based memory technologies such as SRAM or DRAM, the value stored in the memory is encoded in the amount of charge stored in the circuit. In non-charge based memory devices however, the information to be saved is instead stored in the resistance state of the memory device, which can be altered through external polarization of the device. This means the memory is non-volatile: even when there is no power supplied to the crossbar, the memory cells retain their values.

CIM solutions for neural network applications focus on the MAC operation. An example of an individual CIM memory cell is illustrated in Figure 2.6.

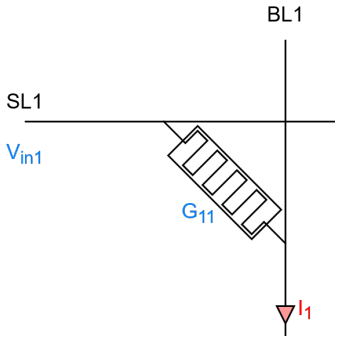

Figure 2.6: Individual memory cell (memristor)

A major advantage of encoding the stored value in the resistance state, is that the data can be processed in the analog domain by using basic circuit laws. As seen in Figure 2.6, the memristor is connected to a Source line (SL) which corresponds to the input of the crossbar, and to a Bit line (BL) which corresponds to the output of the crossbar. For an individual device, the output current corresponding to an input voltage can be calculated using Ohm's Law. By approaching the memristor as a variable conductance.

$$I_{device} = \frac{V_{in}}{R_{device}} = V_{in} \cdot G_{device} \tag{2.2}$$

Observe how the resulting current is proportional to a multiplication between two values: the input voltage of the device and the stored conductance value. This means that every memory device in the CIM crossbar can perform a single multiply operation. Thus, a memristor crossbar of size MxN can perform MxN multiply operations at the same time. Within a column of the crossbar, the total current can be calculated using Kirchhoff's Current Law. A column of the crossbar is illustrated in Figure 2.7.

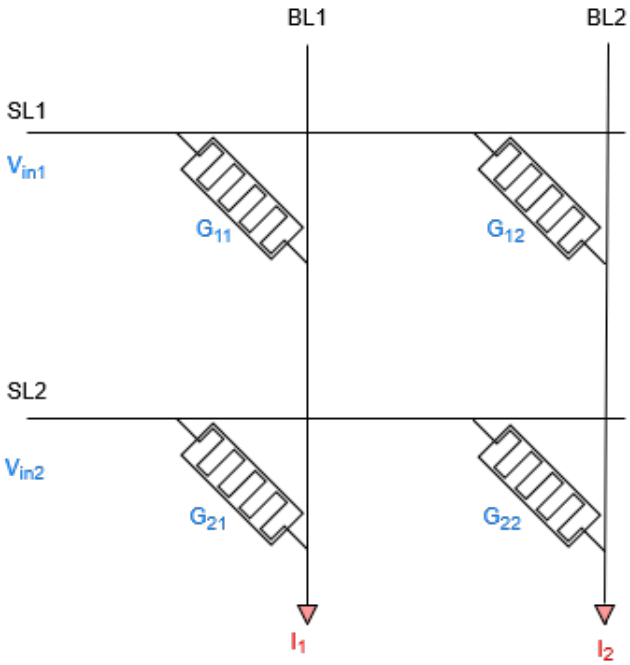

Figure 2.7: Crossbar Column

Using Kirchhoff's current Law (Equation 2.3, the total current flowing out of the column is equal to the sum of all currents flowing into the column.

$$I_{column} = \sum I_{device} \tag{2.3}$$

Observe how the total output current is a sum of the output current of the individual devices.A s such, the output current can be calculated as the summation of several multiplications: the MAC operation. For this reason, the CIM Crossbar architecture is very well suited for running AI workloads.

2.2.1. On-chip Learning

In existing CIM solutions, the network is trained off-chip. This means the neural network is trained externally, often using cloud computing services, and only once the network is fully trained, the final weights are obtained and written to the local CIM crossbar. For ECG analysis and other data-sensitive applications, sending training data off to the cloud poses a security risk. Instead, a method for training a neural network locally is required.

Within CIM architectures, it is possible to train the network on the same crossbar used for the inference task. This is known as On-chip Learning. The goal of training this way is to update the weights of the crossbar without requiring an external processor to calculate the weights. The most commonly used method for On-chip Learning is Stochastic Gradient Descent (SGD). SGD is a supervised learning method and as such requires input data that has been labeled. Training this way involves three stages: forward propagation, backward propagation and finally the weight update. An overview of these three stages can be found in Figure 2.8

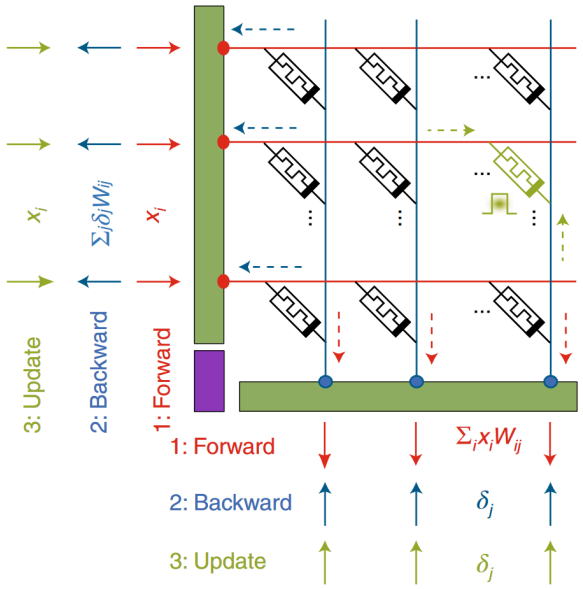

Figure 2.8: Three stages of On-chip learning. Adapted from [15]

The forward propagation stage is very similar to running the inference task on the crossbar: A set of input data is fed to the crossbar in the form of voltages, the crossbar performs the MAC operation between the inputs and the stored weights in the crossbar and the resulting sum is read out.

Once forward propagation is complete, the resulting outputs are compared against the labels of the training data. The differences are known as error gradients. In case of a layer with multiple networks, These error gradients can then be propagated in the reverse direction through the network, starting from the output layer. By applying these gradients as "reverse inputs" the error gradient per crossbar can be found on its "reverse output".

By then applying the input data in the forward direction while applying the error gradient in the reverse direction at the same time, any weights that were used to calculate the erroneous output in the forward

propagation are updated to compensate.

2.2.2. Periphery

The main challenge in On-chip Training is the selection of crossbar periphery. Most state-of-the-art solutions connect the outputs of the crossbar to Analog-to-Digital converters (ADCs). The ADC's allow multiple rows of the crossbar to be read out at the same time, thus increasing the parallelism and throughput of the crossbar. Converting between the analog and digital domain does have the downside of quantization errors: values that are in between two separate digital values will be quantized to either one of them, thus sacrificing some accuracy. For the inference task, this problem is small enough to be overlooked [16]. For the training task however, the loss of accuracy makes it impossible to accurately train the network.

In the State-of-the-art the most common approach is to have a single crossbar which performs the forward pass in the default direction, but can also perform the backwards pass in the opposite direction. One major challenge of this approach is the selection of the periphery to use: digital approaches have high accuracy but are slower, while analog approaches have higher parallelism but do not have the accuracy to efficiently perform the backward pass.

2.3. FeFET

The FeFET is a recently emerging memory technology which is well-suited for CIM. The structure of a FeFET is similar to that of a MOSFET transistor: both are three-terminal semiconductor devices with a Gate, Source and Drain terminal. The conduction between the source and drain can be controlled by the amplitude of the voltage applied to the gate. The difference between the FeFET and MOSFET is the addition of a layer of ferroelectric material within the gate stack. By altering the polarization of the ferroelectric material, the FeFET's threshold voltage can be altered between a high voltage threshold (HVT) corresponding to a logical '0' and a low voltage threshold corresponding to a logical '1'. Switching from the HVT to the LVT is known as writing, while switching from the LVT to HVT is known as erasing. A diagram of the composition of a FeFET device can be found in Figure 1.3 and a schematic of the FeFET device can be found in Figure 2.9.

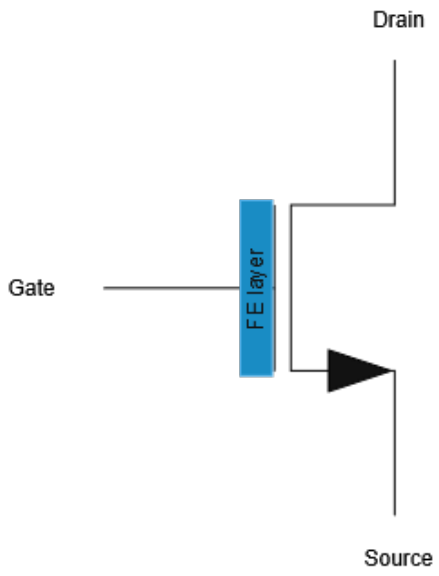

Figure 2.9: Schematic of FeFET Device

2.3.1. Ferroelectric materials

Materials can be polarized by applying an electric field to them. Ferroelectric materials are materials which retain their induced polarization after being polarized, even when the electric field is removed. The induced polarization is nonlinear and non-zero and follows a loop pattern. An example of a hys-

teresis loop for the induced polarization can be found in Figure 2.10.

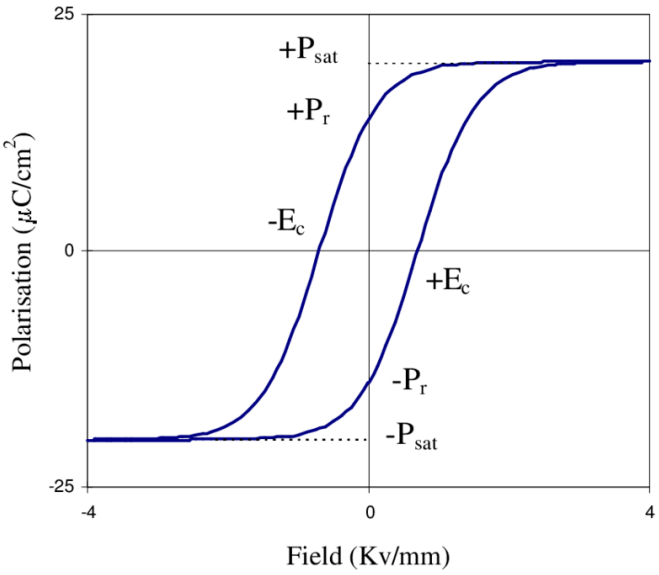

Figure 2.10: Hysteresis loop for ferroelectric materials. Adapted from [8]

Starting from the origin of the graph, Figure 2.10 shows that if an electric field is applied to a non-polarized material, the polarization increases in a non-linear fashion up to a saturation polarization. If the electric field is then reduced to 0, the polarization does not decay to zero, but instead remains at a certain value P_r. This value is known as the remanent polarization.

If a negative electric field is now applied, the polarization drops to a negative value, following the green line to the bottom-left of the figure. When the negative electric field is removed, the polarization now decays in the positive direction, following the red line until the negative remanent polarization is reached. Plotting the polarization against the electric field for a non-ferroelectric material would show a linear relationship between the electric field and polarization, equaling zero when the electric field is equal to zero.

In order to change the polarization of the ferroelectric layer, ”write” and ”erase” pulses are used. By appling a positive voltage pulse to the gate of the FeFET while keeping the source and drain terminals at 0 V, the ferroelectric layer will be polarized towards the channel, thus storing a logical ’1’ or ”writing” the FeFET. On the other hand, by applying a negative voltage accross the gate compared to the source and drain terminals, the ferroelectric layer will be polarized in the other direction, thus storing a logical ’0’ or erasing the FeFET.

2.3.2. FeFET as memory device

By polarizing the internal ferrotelectric layer, two distinct values can be stored in a FeFET: a logical ’1’ corresponding to a positive polarization of the ferroelectric layer and a logical ’0’ corresponding to a negative polarization of the ferroelectric layer. The polarization of the ferroelectric layer will influence the threshold voltage of the FeFET. When the ferroelectric layer is polarized towards the channel, it will be easier to form a channel, and thus the threshold voltage of the FeFET will be reduced. On the other hand, if the ferroelectric layer is polarized away from the channel, it will be harder to form a channel and thus the threshold voltage will increase. This means that, in addition to controlling the conductance of the channel by controlling the gate voltage, the threshold voltage of the FeFET can be controlled through the polarization state. The combination of the polarization state and the gate voltage will thus dictate whether or not the FeFET conducts.

The influence of both the gate voltage and the state of the ferroelectric layer is investigated by considering the circuit displayed in Figure 2.11.

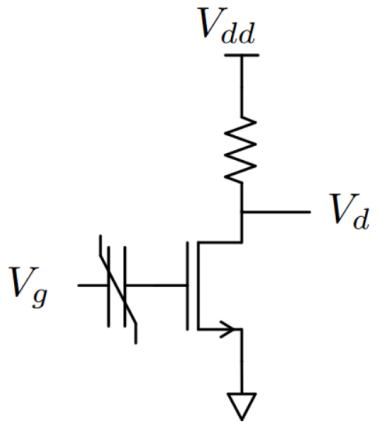

Figure 2.11: Sample FeFET circuit for NAND/AND logical operation. Adapted from [17]

In this circuit, the FeFET (displayed here as a separate FeCAP and transistor) is used as a pull-down network and a resistor is used as a pull-up network. The drain voltage can then be plotted against the gate voltage for both the HVT and the LVT. This plot can be found in Figure 2.12.

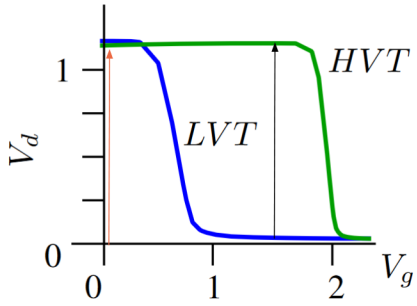

Figure 2.12: Drain voltage plotted against gate voltage for both HVT and LVT states for FeFET. Adapted from [17]

Figure 2.12 displays the conductivity of the FeFET device based on its threshold state and the applied gate voltage. When the gate voltage is 0, the FeFET does not conduct. Thus, the the resulting drain voltage V_d will be high because no current can flow to the ground terminal. If the gate voltage is increased, the device will eventually begin conducting, allowing current to flow through the transistor, into the ground terminal. Once the device conducts, the drain voltage drops to zero. The required voltage for the FeFET to start conducting is based on its threshold state. When the device is in LVT, it will start conducting at a low voltage (in the Figure this is the blue line). If the device is in HVT instead, a much higher gate voltage is required to make the device conduct. (in the Figure, this is the green line).

By selecting a gate voltage in the range where an LVT FeFET will start conducting but an HVT FeFET will not start conducting, a logical NAND operation can be implemented [17]. This means a voltage has to be selected in between the blue and green lines in Figure 2.12, An example suitable gate voltage is represented by the black arrow.

This behaviour corresponds to that of a NAND cell, where the gate voltage and the stored bit are its inputs and the drain voltage is its output. This behaviour is displayed in Table 2.1.

If instead, the current flowing into the ground terminal is taken as an output, this same circuit now realizes a logical AND operation: When the device is conducting, the current is high and when the device is not conducting the current goes to zero. This behaviour is displayed in Table 2.2

Table 2.1: FeFET NAND behaviour

Input A (Stored bit)	Threshold state	Input B (Gate voltage)	Output (Drain Voltage)
0	HVT	0	1
0	HVT	1	1
1	LVT	0	1
1	LVT	1	0

Table 2.2: FeFET AND behaviour

Input A (Stored bit)	Threshold state	Input B (Gate voltage)	Output (Source current)
0	HVT	0	0
0	HVT	1	0
1	LVT	0	0
1	LVT	1	1

As described before, the MAC operation is the most important operation in any NN workload. A multiplication of two bits that can be either '0' or '1' is equivalent to performing an AND operation on these two bits. This is displayed in Table 2.3. Thus, by storing a single bit in a FeFET cell, the inherent AND operation is equivalent to performing the multiplication part of the MAC operation. This property makes the FeFET a promising candidate for CIM.

Table 2.3: Comparison of AND operation and bit multiplication results

Input A	Input B	AND result	Multiplication result
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Research has also been done into utilizing intermediate polarization states and thus storing multiple bits within a single FeFET device. In doing so, instead of using only the remanent polarization values, any amount of intermediate polarization states will also correspond to a certain combination of stored bits. These intermediate polarization states can be reached through applying write/erase pulses of a smaller amplitude or shorter duration.

2.3.3. Comparison of Memory Technologies

In well-known SRAM and DRAM memory technologies, the data is stored as an amount of charge. In FeFETs however, the data is stored in the polarization of the ferroelectric layer. This means the FeFET can function as a non-charge based memory. This means that even when there is no power applied to the FeFET, the stored data is retained, whereas SRAM and DRAM will lose their values if power is disconnected. This property is known as non-volatility, which the FeFET shares with standard two-terminal memristors.

However, traditional memristors, such as PCM and ReRAM have their own downsides. Firstly, they use current-based writing schemes. This means a high write current flows during the write operation, leading to a high write energy consumption. In addition, the read sensing margin is very small because of a low I_{on}/I_{off} ratio [18]. This means the difference between different outputs is very small in terms of output voltage/current. FeEFETs have large current swings, comparable to standard FETs [9] and are thus well suited for CIM applications.

In Table 2.4 the FeFET memory technology is compared to both traditional CMOS-based memory technologies as well as memristor technologies.

2.3.4. FeFET for CIM

In addition to its advantages in terms of electrical properties, the FeFET also has other very useful properties. One main advantage of the FeFET is the separation of read and write paths. In traditional

Table 2.4: Comparison of memory technologies. Adapted from [2].

Metric	SRAM (6T)	DRAM (1T1C)	Flash (1T)	RRAM (1T1R)	MRAM (1T1R)	PCRAM (1T1R)	FeFET (1C1T)
Size	120-150	10-30	10-30	10-30	10-30	10-30	10-30
Volatility	Yes	Yes	No	No	No	No	No
Write energy	fJ	10fJ	100pJ	1pJ	1pJ	10pJ	fJ
Write latency	1ns	10ns	0.1-1ms	10ns	5ns	10ns	20ns
Read latency	1ns	3ns	100ns	10ns	5ns	10ns	0.28ns
Endurance	10^16	10^16	10^4-10^6	10^7	10^15	10^12	10^9

memristors, both reading and writing happen through the same path. This means that whenever a memristor is read, the memristor's stored value will be influenced. In the three-terminal FeFET however, writing does not require a current to flow through the channel of the transistor, preventing this disturbance.

Another advantage of FeFET technology is the similarity in structures to MOSFET transistors. Because of their similarity, the addition of the ferroelectric layer is a minimally invasive procedure, allowing FeFETs to be incorporated directly into the CMOS production process [19] [20].

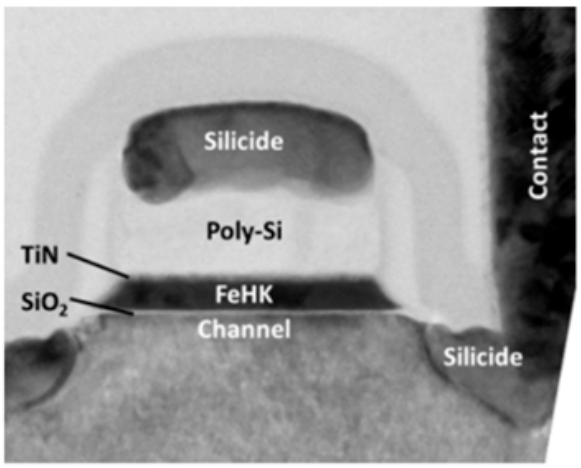

Figure 2.13: Integration of ferroelectric layer in MOSFET production process. Adapted from [19]

On the other hand, FeFETs also pose a unique challenge. FeFETs are three-terminal devices, unlike the two-terminal memristors. This means that they can not be inserted directly into existing CIM layouts, but require significant array-level changes to the architecture.

3

FeFET-based CIM Accelerator Design for On-Chip Learning

In this chapter a novel design for a Neural Network Accelerator that supports online Learning on FeFET devices is proposed. Firstly, an overview of the design is given, after which individual components are explained in detail.

3.1. Design overview

The design for the Neural Network Accelerator is displayed in Figure 3.1. The accelerator consists of two separate crossbar types: An analog crossbar which is used for the inference task as well as the forward propagation, and a secondary digital crossbar which is used for backpropagation. This hybrid approach is necessary, because the quantization errors that stem from analog crossbars make them unfit for the backpropagation step. Quantization errors made in the ADC during the backpropagation step would lead to incorrect error gradients, which in turn means the convergence of the training could not be guaranteed. As such, a digital crossbar is required for the backpropagation step because of its high accuracy.

Both crossbars consist of two components: The FeFET crossbar itself, and the corresponding periphery. The analog crossbar uses a novel TDC approach where a capacitor is precharged and then connected to the bitline inputs of the crossbar. The speed at which the capacitor discharges is then measured using a TDC and the duration of the capacitor discharge is converted to an output value using a counter. The digital crossbar uses a buffer connected to the senseline outputs of the crossbar to read the crossbar out row by row. The counter is then used to add the digital pulses to obtain the final result.

The hybrid approach has several advantages:

- The high parallelism of the analog crossbar is used for as many tasks as possible: the inference, forward prapagation and weight update are all performed in the analog crossbar. This leads to a high throughput.
- Using the digital crossbar for the backward propagation leads to high accuracy, but reading row by row sacrifices parallelism. This design combines the accuracy of the digital crossbar with the throughput of the analog crossbar.

3.2. FeFET Bitcell Design

As part of the Ferro4EdgeAI project, the Ferroelectric Material Company (FMC) has provided a model of a Ferroelectric Capacitor (FeCAP). This component functions as a capacitor with a ferroelectric material, but is not a FeFET itself. In order to model FeFET behaviour, this FeCAP model is combined with a 40nm TSMC NMOS transistor. A schematic representing the FeFET circuit can be found in Figure 3.2.

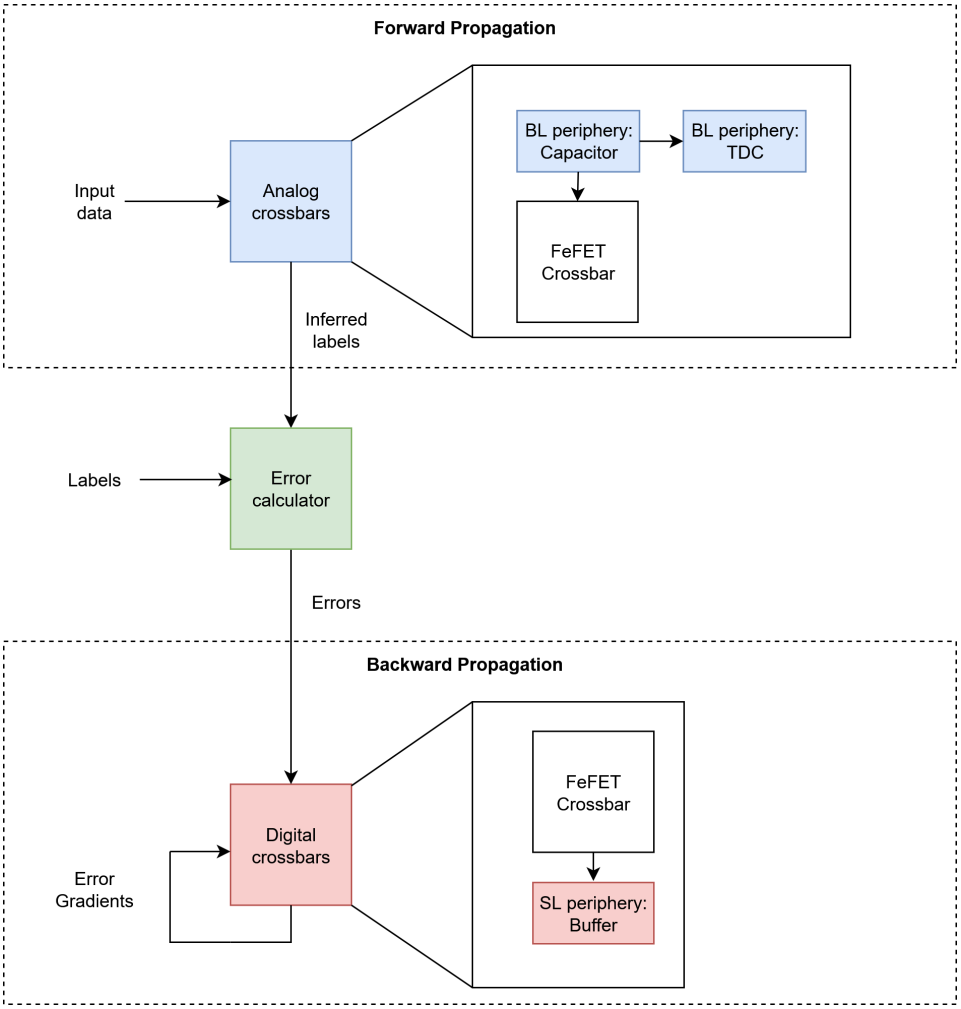

Figure 3.1: System Overview for the Accelerator

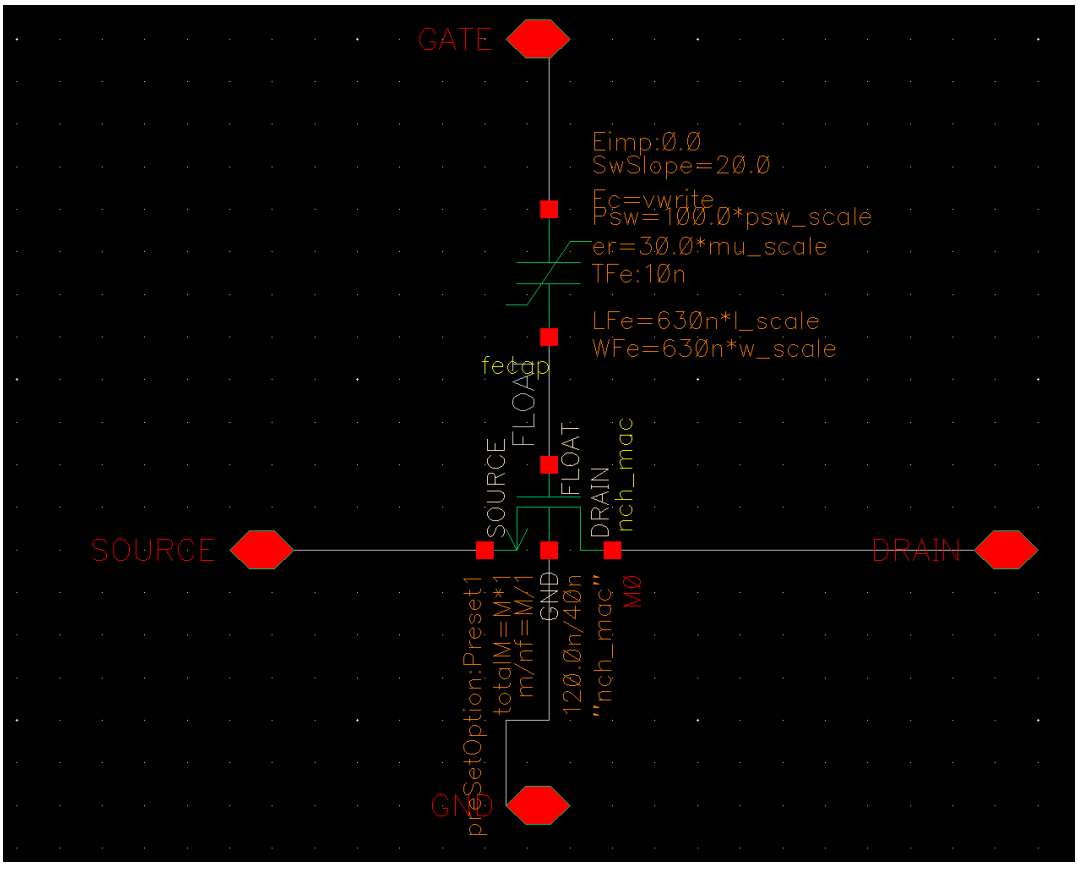

Figure 3.2: Schematic of the modelled FeFET device

There are some important differences in using this FeCAP-MOSFET model compared to the FeFET model in literature. Firstly, in literature a positive pulse applied to the gate of the FeFET polarizes the device to a LVT-state, which corresponds to storing a logical '1' and thus a WRITE operation. The reason for this is that the polarization direction points towards the channel, which causes electrons in the substrate to form a channel more easily, in turn reducing the V_T. In the combined FeCAP-NMOS model, a positive 'write'-pulse instead polarizes the device to a HVT-state, which corresponds to storing a logical '0' or ERASE operation instead. The reason for this difference is that the FeCAP is not embedded in the gate stack, but instead added on top of the gate stack. As such, the voltage drop across the capacitor does not assist in creating a channel, but instead provides a potential barrier for the voltage that is applied to the gate. This means a higher voltage is required to overcome this potential barrier and reach the actual gate of the FeFET.

Secondly, the FeCAP is now a separate device rather than being integrated. This means the design will take up additional area as opposed to having the ferroelectric layer integrated within the FeFET itself. When the ferroelectric layer would be integrated in the FeFET, the ferroelectric material takes up no extra area at all. To compensate for this, all area taken up by the FeCAP devices will be ignored.

Thirdly, the model itself ignores the time dependency of the polarization that is described in Section 2.3. This means the polarization does not increase to a saturation and then drop back down to a remanent polarization once the electric field is removed. Instead, the polarization goes up to a maximum polarization and stays there without decaying.

3.2.1. Determining control parameters

Writing to the FeFET is done by applying a positive voltage pulse on the Source and Drain of the FeFET, while keeping the gate voltage at 0. In doing so, the ferroelectric capacitor is polarized in the direction away from the channel, reducing the gate voltage needed to allow for conduction through the channel. Erasing the FeFET can be done by applying a positive voltage pulse on the Gate of the FeFET, keeping the Source and Drain voltages at 0. Now, the ferroelectric capactior is polarized towards the channel, forming a potential barrier that has to be crossed in order to allow conduction through the channel. Reading the FeFET is done by applying a smaller read voltage V_{read} at the gate and applying a Drain-source voltage V_{top} at the source of the FeFET. Now, based on the stored memory state of the FeFET, the FeFET will either be turned on or off, generating a output current. An overview of these control signals can be found in Table 3.3.

Table 3.1: Control signal information for 1C1T FeFET device

Operation	Write	Erase	Read
Gate	0	V_{write}	V_{read}
Source	V_{write}	0	V_{top}
Drain	V_{write}	0	Float

The values for these parameters should be set in such a way to optimize the I_{on}/I_{off} ratio. That is, the difference in amplitude between the output current for a logical '1' output and the output current for a logical '0' output should be as large as possible. To do so, the FeFET device should be (operating in cut-off mode i.e. turned off) when in HVT and operating in saturation mode when in LVT. This means that in HVT the threshold voltage should exceed the gate-source voltage:

$$V_t > V_{gs} \tag{3.1}$$

And for LVT, the gate-soure voltage should exceed the threshold voltage while the drain-source voltage exceeds the gate-source voltage minus the threshold voltage.

$$V_{gs} > V_t \tag{3.2}$$

$$V_{ds} > V_{gs} - Vt \tag{3.3}$$

As the gate-source and drain-source voltage can be controlled manually, their values should be based upon the threshold voltage of the FeFET. In order to determine the threshold voltage, a simulation is

performed. In this simulation, either a 'write' or 'erase' pulse of varying length is applied to initialize the FeFET to a certain polarization state. Afterwards, the gate-source voltage V_{gs} is sweeped. The value of V_{gs} at the moment the transistor turns on then corresponds to the device's threshold voltage. In order this simulation, both the amplitude and duration of the pulses is varied. The results of this simulation for various pulse lengths can be found in Figure 3.3 and 3.4.

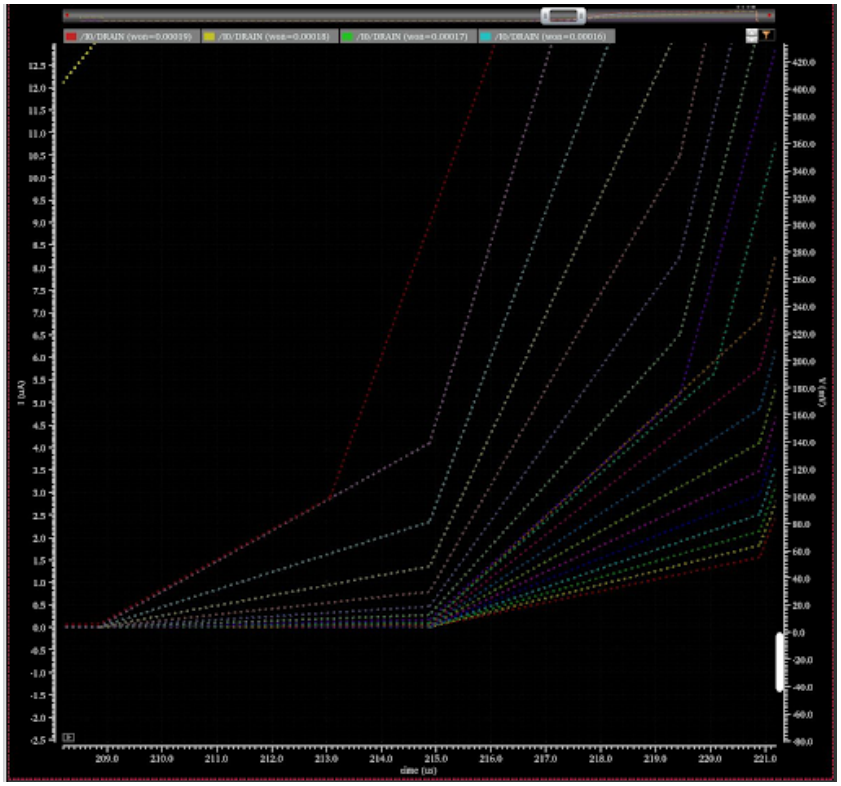

Figure 3.3: Threshold voltages for varying erase pulses

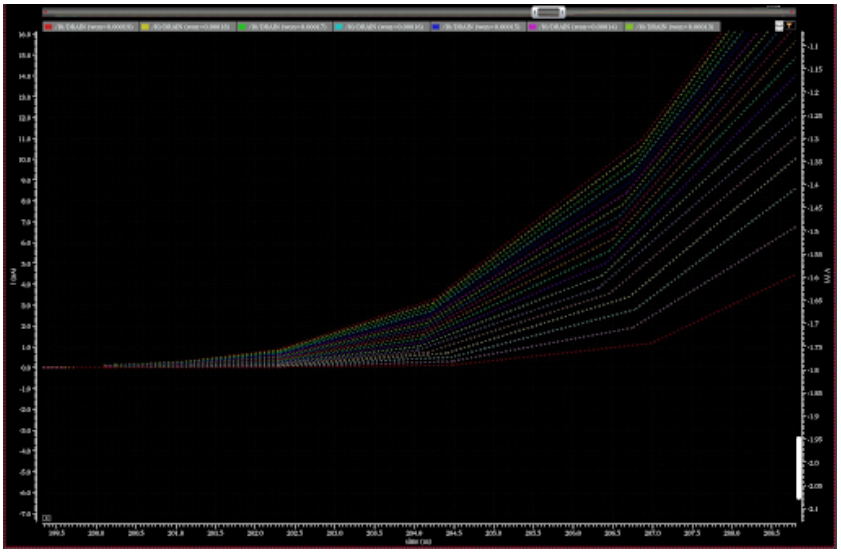

Figure 3.4: Threshold voltages for varying write pulses

In Figure 3.3, the drain current I_d is plotted against time for different erase states. In the first 200us of the simulation, the FeFET is initialized by erasing it for varying lengths at V_{write} = 1.8. Starting from t = 200us, the gate voltage is swept. At t=215us, the transistor starts conducting for even the longest

erase pulses. At this point, the threshold voltage V_{gs} is equal to 0.75V. Equation 3.1 tells us that the gate-source voltage should thus be below 0.75V.

On the other hand, in Figure 3.4, the drain current I_d is plotted against time for different write states. In the first 200us of the simulation, the FeFET is initialized by writing it for varying lengths at V_{write} = 1.8. Starting from t = 200us, the gate voltage is swept. At t=209us, the transistor starts conducting for even the longest erase pulses. At this point, the threshold voltage V_{gs} is equal to 0.45V. Equation 3.2 tells us that the gate-source voltage should thus be above 0.45V. Additionally, the drain-source voltage should be bigger than the difference between the gate-source and the threshold voltage.

Combining these two results, the gate-source voltage V_{read} should be in between 0.45V and 0.75V to satisfy both constraints. Selecting V_{read} = 0.5V, The drain-source voltage V_{top} should exceed the difference between the read voltage 0.5V and the threshold voltage 0.45V. Thus, only a minimal gate-source voltage V_{top} > 0.05V is required. The final control signal values for the FeFET can be found in Table 3.2.

The proper functioning of these parameters will be verified in Section 4.

Table 3.2: Control signal voltages for 1C1T FeFET device

Operation	Write (V)	Erase (V)	Read (V)
Gate	0	1.8	0.5
Source	1.8	0	0.1
Drain	1.8	0	Float

3.3. Crossbar Design

The simplest FeFET bitcell consists of just a single FeFET. This is referred to as a 1C1T or 1F bitcell. A sample layout for a 2x2 crossbar consisting of 1C1T bitcells can be found in Figure 3.5 The gate of each bitcell is connected to a Wordline (WL), the drain is connected to a Senseline (SL) and the source is connected to a Bitline (BL).

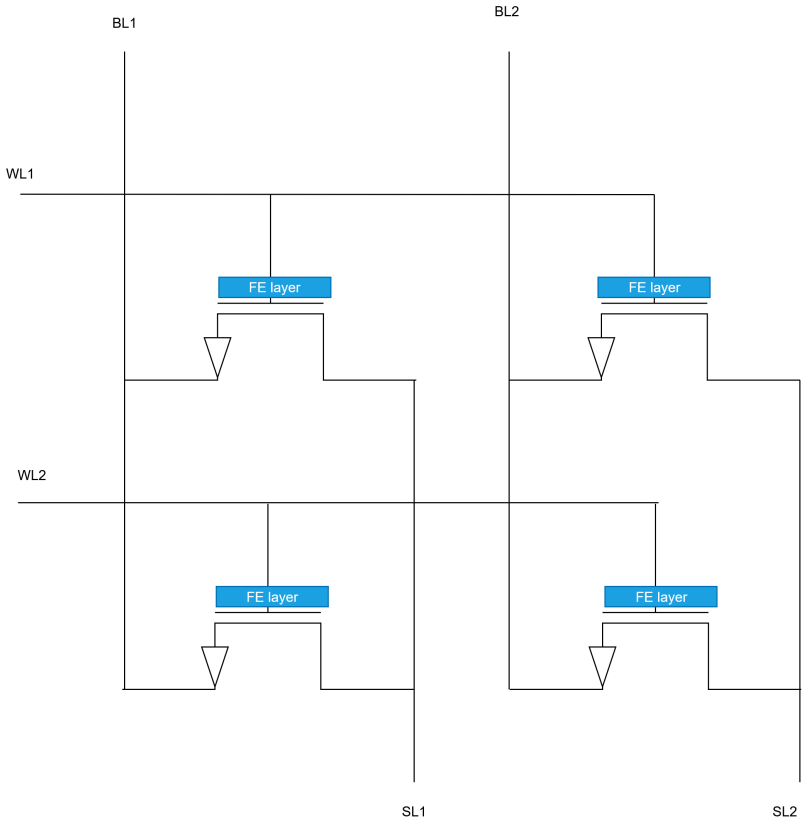

Figure 3.5: 1C1T Crossbar

The main advantages of the 1C1T bitcell are its simplicity and very high density. However, when integrating the bitcell into a crossbar it suffers from write disturbance. In order to write to $FeFET_{11}$ in the top left corner, bitline BL1 and senseline SL1 have to be set to V_{write} while wordline WL1 is set to 0. If all other lines are also set to 0, then $FeFET_{21}$, located in the same column as the targeted FeFET will experience the same voltages on its drain and source, thus also writing to $FeFET_{21}$. This means that whenever a FeFET is written to, every FeFET in the same column will also be written to.

The same is true for erasing: in order to erase $FeFET_{11}$ in the top left corner, its wordline WL1 has to be set to V_{write} while all other lines are kept at 0. In doing so, $FeFET_{12}$ will experience the same voltage on its gate, leading to the entire row being erased.

In both cases, this behaviour is undesired and is known as write/erase disturbance. In order to alleviate this, an inhibition scheme is required. An inhibition scheme refers to inhibtiting unselected FeFET cells during write operations on the other FeFET cells by appling various voltages. The most common are the $V_{write}/2$ and $V_{write}/3$ IB schemes, in which voltages equal to either half or one third of V_{write} are applied to non-active lines. These inhibition schemes prevent write disturbance, but have the major downside of high power consumption [9]. Generating non-standard control signals for all inactive Bitlines, Senselines and Wordlines is very expensive in terms of power consumption.

The writing scheme for a 1C1T crossbar involves two write steps on every row: One for writing to LVT and another for erasing to HVT. The first step writes '1' to the selected cells in the row by setting the row's WL to 0 and setting BL's and SL's to V_{write} where needed and inhibiting all other WL's and SL's. The second step writes '0' to the other cells by setting the row's WL to V_{write} and the respective BL's and SL's to 0.

3.3.1. 1C2T

The 1C2T or 1F1T bitcell aims to overcome the problem of write disturbance by connecting an access transistor to the gate of the FeFET. A 2x2 crossbar of 1C2T bitcells is displayed in Figure 3.6.

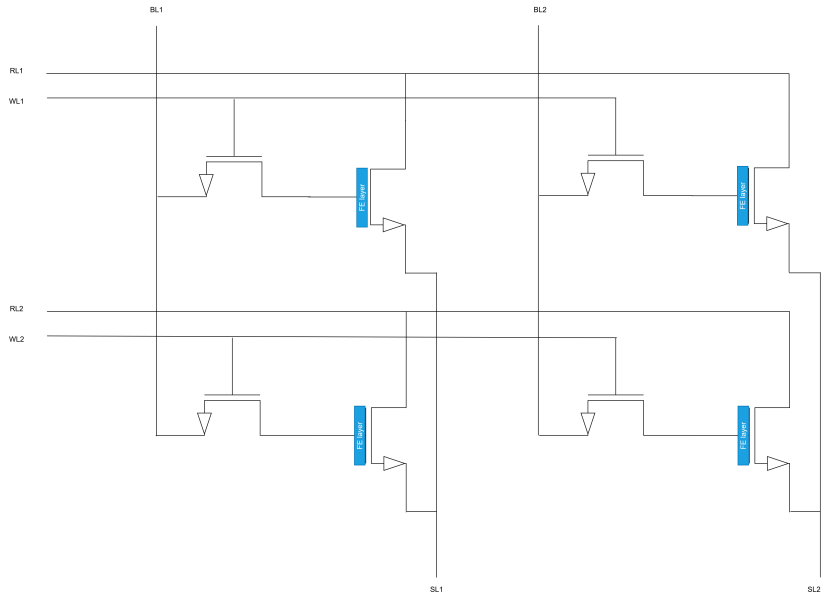

Figure 3.6: 1C2T Crossbar

By adding an access transistor to the gate of the FeFET, inactive FeFETS (i.e. FeFETs other than the FeFET currently selected to be written to) will no longer suffer from write disturbance. The downside of this approach is that every bitcell now requires more area.

For this work, both the 1C1T and 1C2T crossbars are investigated and compared to reach an optimal design.

Table 3.3: Control signal information for 1C2T FeFET device

Operation	Write	Erase	Read
Access Line	V_{write}	V_{write}	V_{dd}
Word Line	0	V_{write}	V_{read}
Bit Line	V_{write}	0	V_{top}
Sense Line	0	0	Float

The writing scheme for the 1C2T step is similar to that of the 1C1T crossbar. In both cases, there are two writing steps: one for writing to LVT and one for erasing to HVT. The main difference lies in how the 1C2T cell writes '1' to all cells in the row during the 'write' step and then overwrites the necessary cells with '0' during the 'erase' step. The 1C1T cell mentioned before instead only writes '1' to the selected cells and doesn't touch the cells where '0' has to be written to. This writing scheme is based on [9].

3.4. Periphery for On-chip training

The crossbar itself can perform the MAC-operation, but periphery circuitry is required to measure the output of the crossbar in a manner that other devices can interact with it. In this design, two approaches for periphery are used:

- **Analog periphery**, where Analog-to-Digital converters are used to read out the entire crossbar in one go. This approach is used for the forward pass, because of its high parallelism.
- **Digital periphery**, where the crossbar is read out row-by-row. This approach is used exclusively for the backwards propagation, because the backwards propagation step requires high accuracy. The quantization errors in analog periphery would prevent the training from converging.

In this section, the circuits and design approach for both digital and analog periphery are examined and their applications for On-chip training are considered.

3.4.1. Digital periphery

The digital periphery consists of a buffer connected to every senseline, which is in turn connected to a counter. When a cell is read, the output pulse or lack thereof on the senseline is fed to the buffer, which restores it to a pulse of magnitude V_{DD}. This output pulse is then fed to a 4-bit counter, which counts the amount of pulses it receives. The total amount of received pulses after reading the entire crossbar row by row, corresponds to the output value for every column. A major advantage of this approach is its simplicity in terms of design. In addition, it prevents the need for ADC circuits, which are both slow and energy inefficient. On the other hand, this digital approach means that only one row can be read out at a time, compared to the analog crossbar in which the entire crossbar can be read in one go. As such, parallelism will be lower compared to the Analog approach.

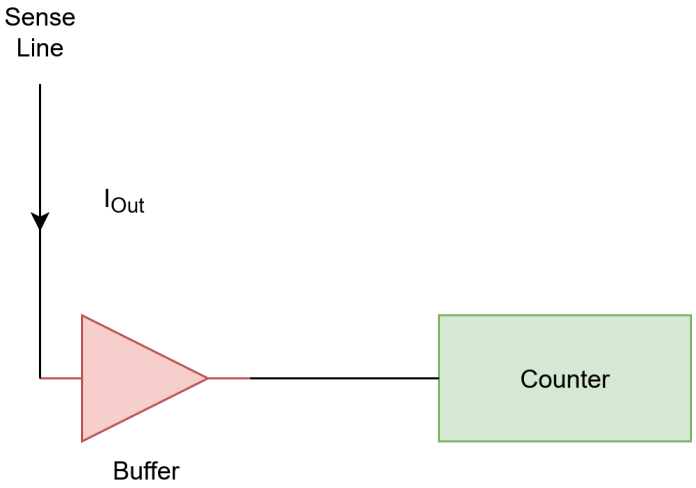

Figure 3.7: Overview of Senseline output with digital periphery

Counter

The counter that is used is a simple 4-bit counter consisting of 4 D-flipflops. Once the first pulse arrives, the counter sets all outputs to '0' and then increments by one for every pulse that arrives. The left-most counter corresponds to the LSB. A circuit schematic of the counter can be found in Figure 3.8

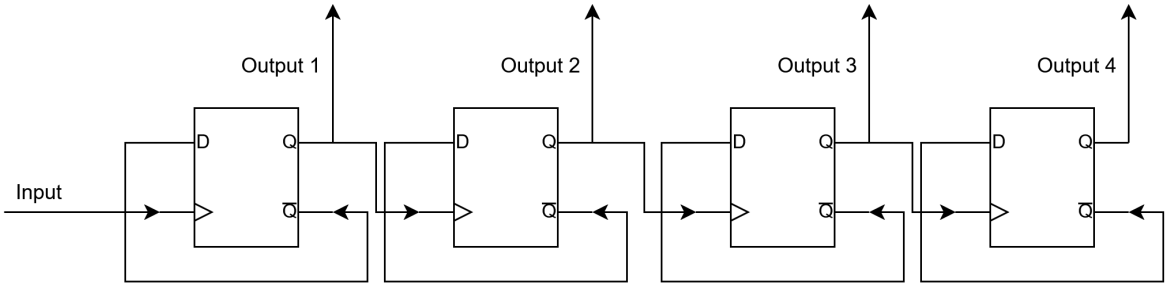

Figure 3.8: Circuit schematic of 4-bit counter

3.4.2. Analog periphery

The analog periphery uses a TDC-based approach. In order to read the crossbar, a set of capacitors is precharged. Then, each capacitor is connected to one of the bitlines, which will cause it to discharge. The rate of discharge is then measured by using a TDC circuit. The TDC circuit consists of a comparator, which will output a pulse as long as the voltage across the capacitor is above a threshold voltage. This output pulse is fed into an AND gate alongside a clock signal, which results in a set of pulses corresponding to the duration it takes the capacitor to discharge to the specified threshold voltage.

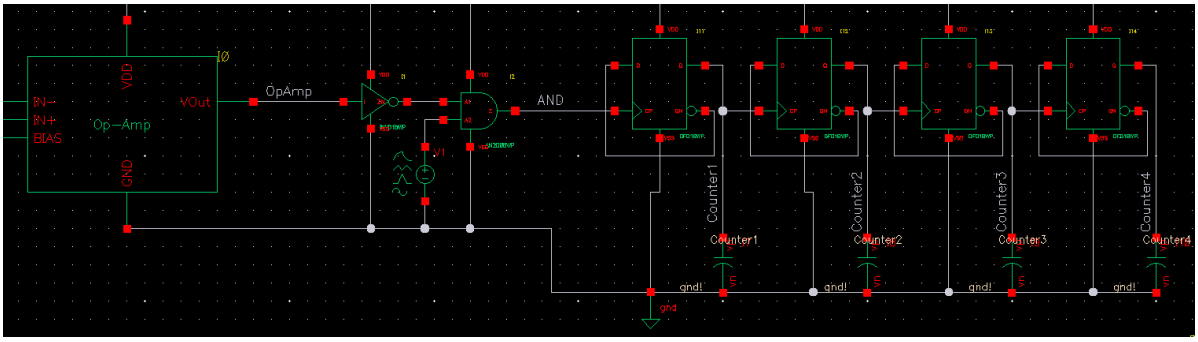

Figure 3.9: Circuit schematic of analog periphery

Comparator

The comparator was designed using 5 transistors. The transistors widths were determined using a bottom-up approach, starting from the bias transistor. A bias of 0.5V and a source voltage of 0.1V were applied to the transistor, after which the current was measured for different transistor widths. For a width of 490nm, the output current is equal to 1.2 μA.

Dividing this current over the two input transistors means the input transistors each need to have an output current of 0.6μA. In order to achieve the desired 0.1V on the source of the bias transistor, a gate voltage of 0.5V and a source voltage of 0.6V are required. By applying these voltages, the desired width of the transistors was found to be 630nm.

In a similar way, the widths for the top transistors can be calculated. The output voltage should be 0.6V, Which means that for a drain voltage $V_{DD} = 1.1V$, a gate voltage of 0.6V is required. In order to have an output current of 0.6μA for these voltages, the required width was found to be 120nm.

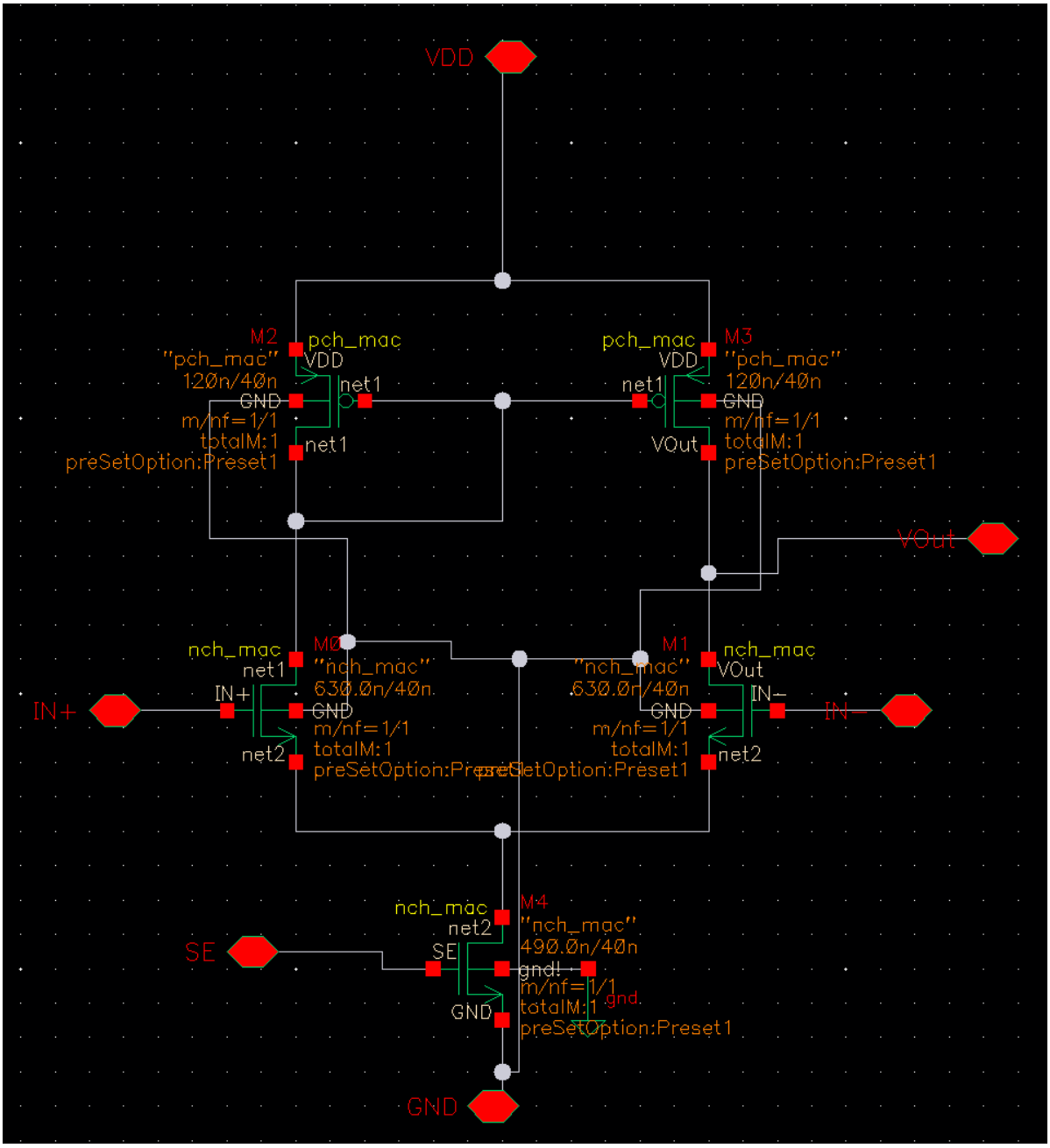

Figure 3.10: Circuit schematic of the comparator

3.5. Crossbar Upscaler

In literature, CIM crossbars are generally large, having around 2^{10} devices per row/column. Due to hardware limitations, generating a netlist of this size using circuit-level simulation tools such as cadence is unfeasible. To compensate for this, cadence virtuoso is used to generate a 2x2 crossbar. Afterwards, the resulting netlist file is fed to a python script, which expands the 2x2 crossbar to a larger MxN crossbar. The python script takes four inputs: the crossbar type (1c1t/1c2t), the desired periphery (analog/digital), the input values for the network and the weight values for the network. When the program is called, it outputs a netlist.scs file which can then be simulated using Spectre's CLI. An overview of the Crossbar Upscaler can be found in Figure 3.11.

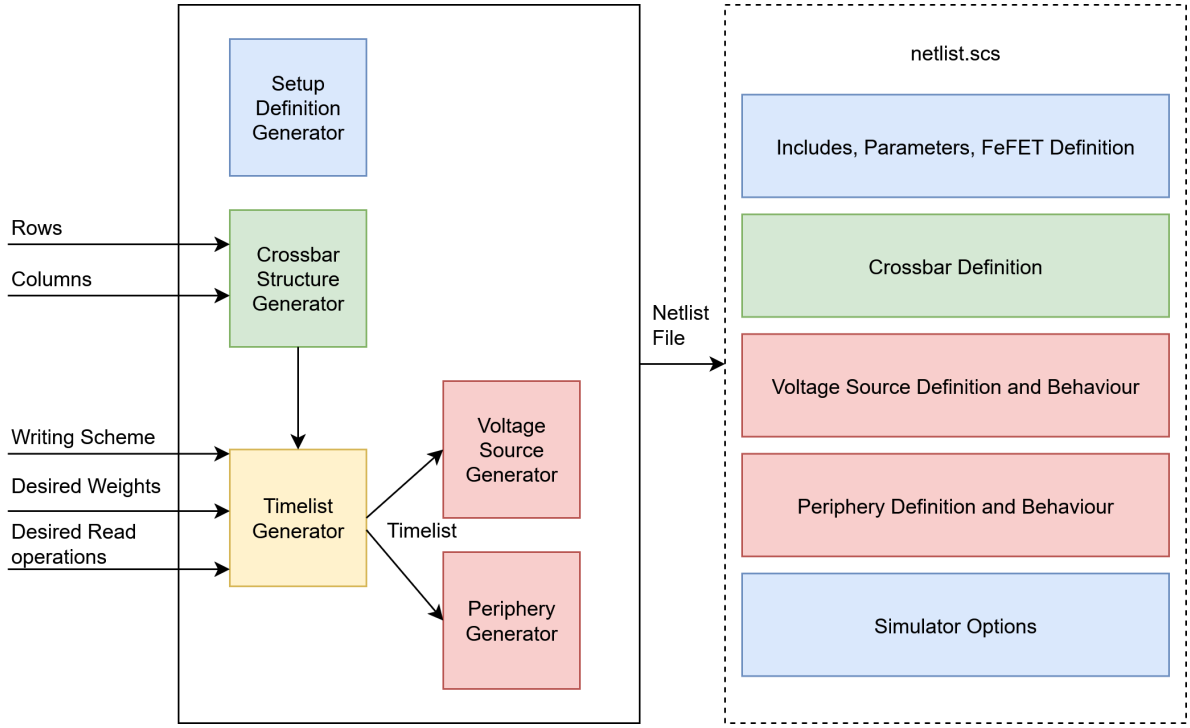

Figure 3.11: Overview of Crossbar Upscaler and Generated Netlist

3.5.1. Definition and Parameter Generator

The program starts by writing the required definitions for the basic FeFET device and any supplied parameters into the netlist.scs file. These are the same for all simulations and as such do not require any specific calculations. Additionally, the program saves the defined options for the spectre simulator and writes them to the file once the rest of the netlist is complete.

3.5.2. Crossbar Structure Generator

The program then generates the structural netlist code to connect every bitcell to its correct connection lines. This is done by iterating over the rows and columns based on the bitcell type and connecting the devices to the respective connection lines for the bitcell type.

3.5.3. Timelist Generator

Next, the program generates a timelist: a list of all the time-voltage pairs at which one of the signals needs to be enabled/disabled based on the write/read steps required. For now, the voltages are kept at zero and only the timestamps are calculated. In order to do this, the script looks at the amount of rows/columns present in the crossbar and calculates the total amount of write and read steps required based on the writing scheme. For both the 1C1T and 1C2T crossbars, the writing scheme consists of two write steps per row: one corresponding to an LVT write and the other two an HVT erase. The details of these writing schemes can be found in Section 4.2.

For a write step of duration w_{on}, four timestamps are logged in the timelist:

- $t_{current} - t_{rise}/2$, corresponding to the moment the write signal starts being applied, taking rise time into account.
- $t_{current} + t_{rise}/2$, corresponding to the moment the full write signal is applied.
- $t_{current} + w_{on} - t_{fall}/2$, corresponding to the last moment the full write signal is applied.
- $t_{current} + w_{on} + t_{fall}/2$, corresponding to the moment where the write signal drops back down to zero.

For every write step, the $T_{current}$ value is then incremented by the total writing time w_{total} and afterwards

the timestamps for the next write step are calculated.

Once every write step has its respective timestamps logged in the timelist, the script moves on to the read steps. For read steps, the process is very similar. For a read step of duration r_{on}, four timestamps are logged in the timelist:

- $t_{current} - t_{rise}/2$, corresponding to the moment the read signal starts being applied, taking rise time into account.
- $t_{current} + t_{rise}/2$, corresponding to the moment the full read signal is applied.
- $t_{current} + r_{on} - t_{fall}/2$, corresponding to the last moment the full read signal is applied.
- $t_{current} + r_{on} + t_{fall}/2$, corresponding to the moment where the read signal drops back down to zero.

For every read step, the $T_{current}$ value is then incremented by the total reading time r_{total} and afterwards the timestamps for the next read step are calculated.

3.5.4. Voltage Source Generator

Finally, the program generates the netlist code for the voltage sources that control the crossbar. For every bitline, senseline, wordline and accessline, a single voltage source instance is created and connected to that line. The voltage-time pairs are calculated by taking the originally generated timelist and setting voltages based on the writing scheme and instructions.

4

Results and Discussion

This chapter will outline the results of the NN accelerator design as well as those of its individual components. The full simulation results as well as the files used to generate them can be found on the github repository [21].

4.1. Simulation setup

For every setup, functional results are obtained by simulating the respective spectre netlist using the spectre CLI. These results differ from performing simulations in cadence directly. The reason for this is described in Chapter 4.8. Area results are obtained by measuring the device using the ruler function in cadence Layout XL. Energy results are obtained by integrating the average power of every instruction (read/write/erase) over the duration of the resepctive operation.

4.2. Bitcell Design Comparison

To verify the behaviour of both the 1C1T and 1C2T bitcell, a simulation was run using the control signals calculated in Section 3.3. For both simulations, an individual cell is written to, then read, then erased, then read once again, corresponding to the writing schemes explained in Section 3.3 The parameter values are summarized in Table 4.1.

Table 4.1: Parameters for bitcell simulations

Parameter	Value
Write voltage 1C1T	1.8 V
Read voltage (gate)	0.75 V
Read voltage (source)	0.5 V
Write time	200 us
Read time	1 us
Operation order	Write, Read, Erase, Read

The results for both the 1C1T and 1C2T bitcell simulation are displayed in Table 4.2.

Table 4.2: Simulation Results for 1C1T cell simulation

Result	Value (1C1T)	Value (1C2T)
Output current LVT	5e-5 A	5e-5 A
Output current HVT	1.8e-10 A	2e-8 A
Write energy	1.81e-13 J	2.57e-13 J
LVT read energy	2.43e-11 J	2.06e-11 J
Erase energy	5.57e-14 J	1.12e-11 J
HVT read energy	8.54e-17 J	1.86e-14 J

The results show that for both the 1C1T and 1C2T cell the output current flowing through the FeFET in LVT has a magnitude of 5e-5, while the HVT output current is higher for the 1C2T bitcell. The I_{on}/I_{off} ratio for the 1C1T cell is 2.5e5, while the I_{on}/I_{off} ratio for the 1C2T cell is only 2.5e3. The performance of both bitcells is similar in terms of energy, with the 1C2T bitcell outperforming the 1C1T cell mainly in erasing the bitcell and reading an empty cell.

Afterwards, the FeFET cells' areas were measured by investigating the FeCAP and transistor model using Virtuoso Layout XL. The FeCAP model does not have a built-in layout and as such can not be measured directly. In an actual FeFET device, the Ferroelectric layer would be integrated within the transistor, at no cost to the area as described in Section 3.2. This means the area overhead of the FeCAP device can thus be ignored and only the transistor area needs to be measured. An overview of the transistor layout can be found in Figure 4.1. The results for the area measurements are displayed in Table 4.14.

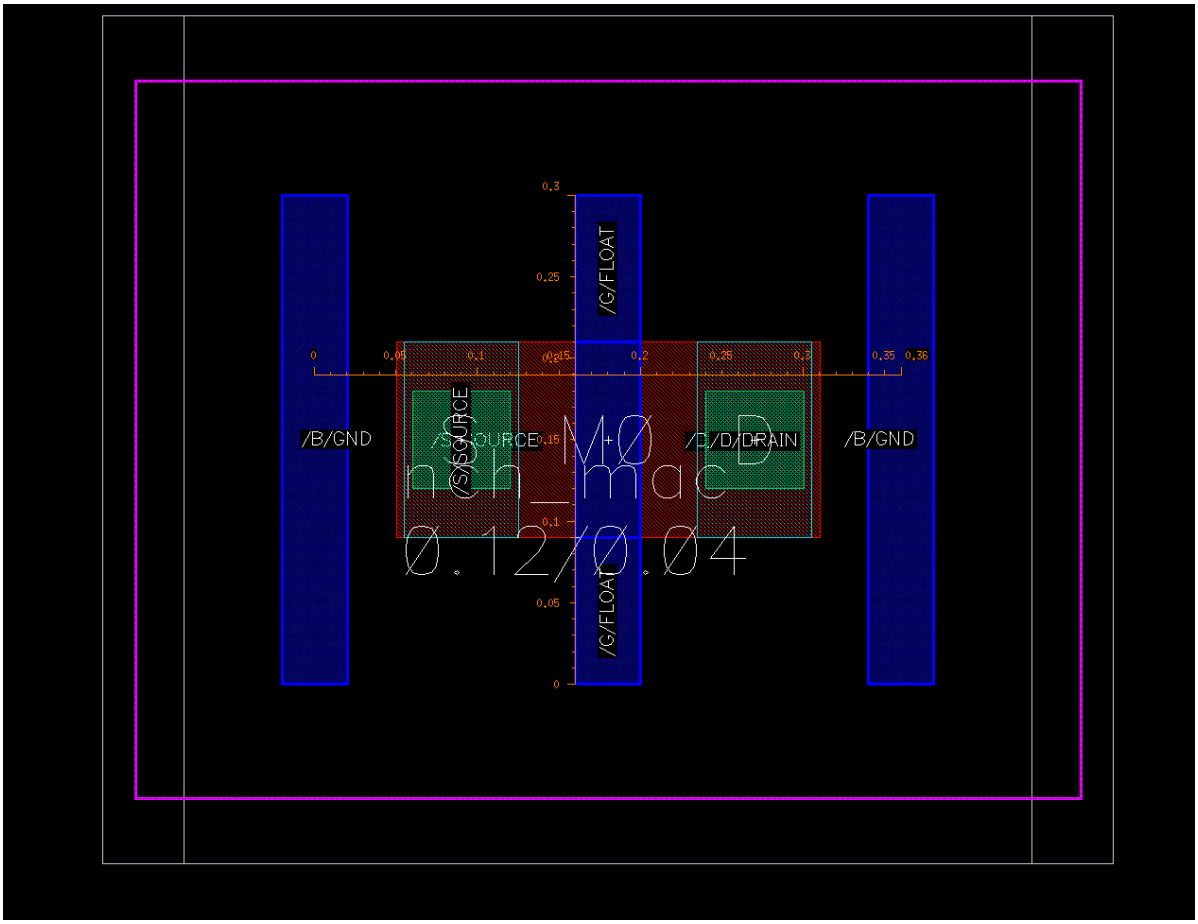

Figure 4.1: Area measurement for transistor used in FeFET

Table 4.3: Area Results for 1C1T cell

Total width	$0.3um$
Total length	$0.36um$
Total area	$0.108um^2$

For the 1C2T bitcell, the bitcell now consists of two of the exact same transistors and a singular FeCAP of which we can once again ignore the area. Thus, the 1C2T bitcell has an area that is double that of the 1C1T bitcell area to $0.216um^2$.

These results show that both bitcell designs are functional, and an individual 1C1T FeFET bitcell outperforms an individual 1C2T bitcell in terms of area, while matching in terms of energy dissipation. However, when multiple 1C1T bitcells are combined to form a FeFET crossbar, many inhibit signals are required to prevent the write disturbance that is inherent to the 1C1T cell which will consume additional power the larger the crossbar is. As such, the 1C2T bitcell was selected for the final design.

4.3. Verification: 2x2 1C2T Crossbar

In order to verify functionality of the crossbar, a 2x2 crossbar will be tested. The 1C2T crossbar uses a writing scheme in which every cell is written to LVT, before desired HVT cells are erased to HVT. In terms of power, this approach inherently favors applications in which the majority of the cells have to be set to LVT. For this reason, two results will be reported: A worst-case result where every cell is set to LVT and then overwritten to HVT, and a best-case result in which every cell is written to LVT immediately. The parameters for the simulation of the crossbar can be found in Table 4.10.

Table 4.4: Parameters for bitcell simulations

Parameter	Value
Write voltage	1.8 V
Read voltage (gate)	0.75 V
Read voltage (source)	0.5 V
Write time	200 us
Read time	1 us
Best-case data to be written	All 1
Worst-case data to be written	All 0

Table 4.5: Simulation Results for 1C2T 2x2 crossbar simulation

Parameter	Value
Output current individual cell LVT	3.8e-5 A
Output current individual cell HVT	4.0e-8 A
Energy best case: Write all '1'	1.28e-11 J
Energy worst case: Write all '0'	8.23e-11 J
Read all '1'	4.17e-11 J
Read all '0'	1.86e-13 J

The results show that the output current flowing through a single FeFET in LVT has a magnitude of 3.8e-5, while the HVT current has a magnitude of 1e-10. This means the I_{on}/I_{off} ratio is around 950. Writing all '0's requires 13% more energy than writing all '1's.

4.4. Digital Crossbar

In order to simulate the digital periphery, the connection between the senseline and ground terminals has to be disconnected. Otherwise, any output current from a read operation would flow directly into the ground, disregarding the output periphery. In the simulation setup, this is done by adding an ideal

switch which disconnects the ground terminal from the senseline once the write operation is finished. The full schematic can be found in Figure 4.2.

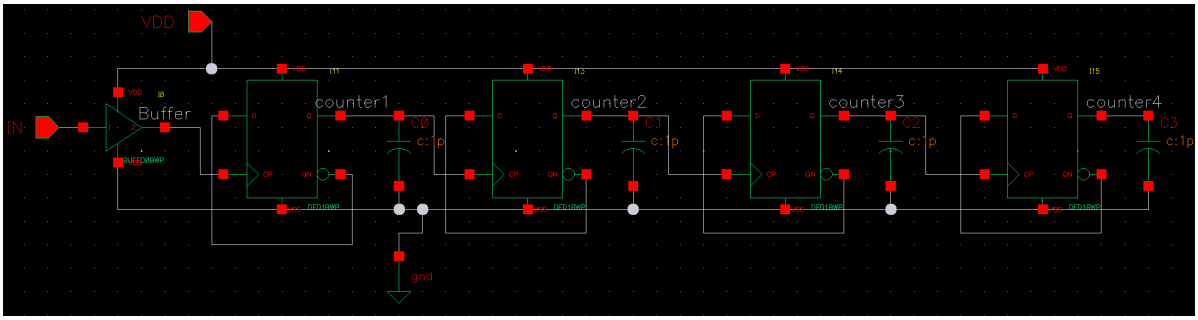

Figure 4.2: Circuit schematic of 2-cell column crossbar with digital periphery

4.4.1. Verification: Digital Periphery

There are many methods of designing a counter using four D-flipflops. Figure 4.3 displays the counter output for a clock input with period 10ns. Once the first pulse arrives, the counter resets to '0000'. Afterwards, for every incoming pulse the counter increments by one, where Counter 1 corresponds to the Least Significant Bit.

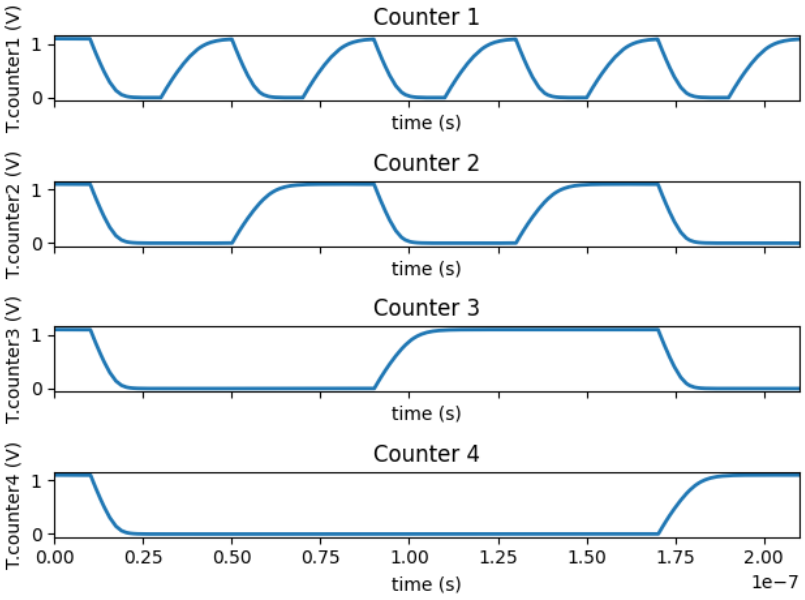

Figure 4.3: Counter output for clock input T=10ns

The results show the counter functions properly at period T=10ns. The buffer that is used is a component directly added from the TSMC library. As such it does not require individual verification.

4.4.2. Verification: Digital Crossbar

Next, the periphery can be connected to a 2x2 crossbar in order to verify the behaviour of the full system. The parameters for this simulation are displayed in Table 4.10.

Table 4.6: Parameters for digital crossbar simulations

Parameter	Value
Stored data Column 1	’1’ ’1’
Stored data Column 2	’1’ ’0’
Write voltage (gate)	1.8 V
Read voltage (gate)	0.75 V
Read voltage (source)	1.6 V
Access voltage (write)	1.8 V
Access voltage (read)	2.1 V
Write time	200 us
Read time	1 us

The resulting voltages at the output of the Buffer and the Counter are displayed in Figures 4.9 and 4.10

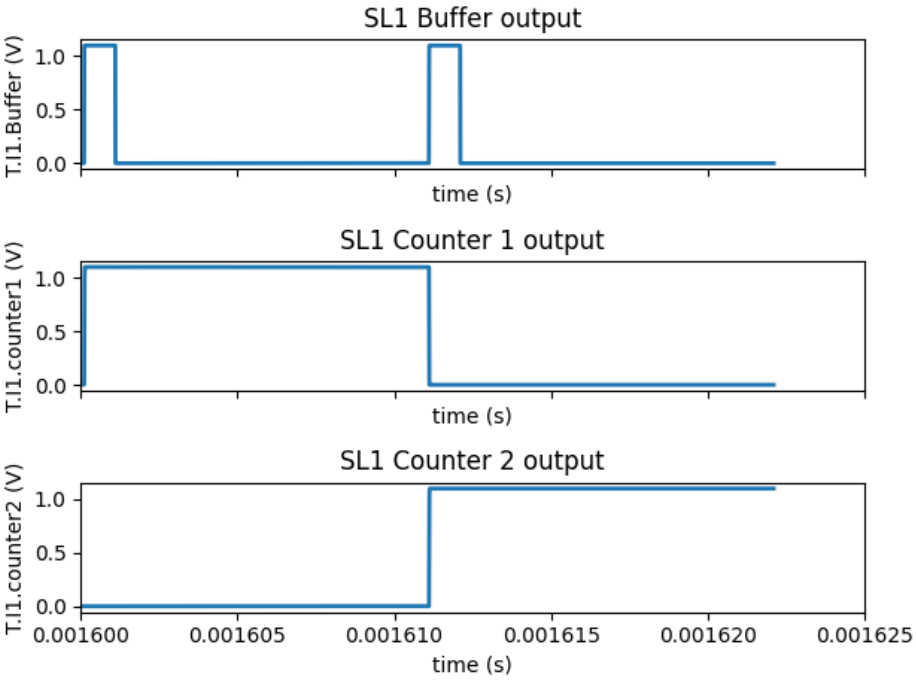

Figure 4.4: Buffer and counter output for column containing ’1’ ’1’

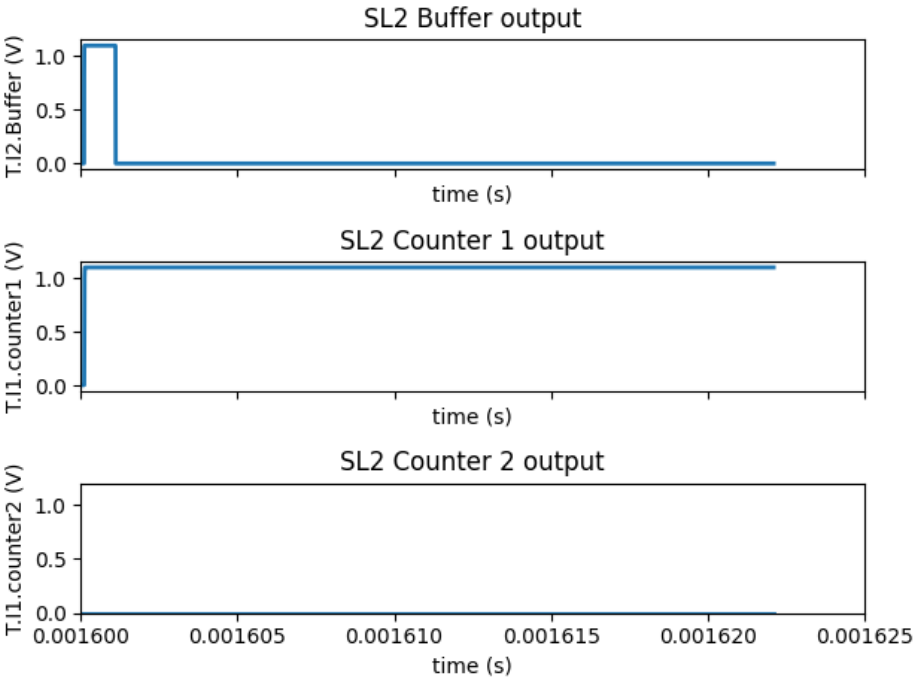

Figure 4.5: Buffer and counter output for column containing '1' '0'

The results show that the counter's output corresponds to the amount of pulses received and thus the circuit is functional.

4.4.3. Digital Crossbar Energy results

Next, an energy simulation was run for a 2x2 and 10x10 crossbar with the digital periphery connected using the same parameters displayed in Table 4.10. The results for these simulations can be found in Table 4.12 and 4.13 respectively.

Table 4.7: Simulation Results for 1C2T 2x2 crossbar simulation with digital periphery

Parameter	Value
Output current individual cell LVT	3.8e-5 A
Output current individual cell HVT	4.0e-8 A
Energy best case: Write all '1'	7.30e-13 J
Energy worst case: Write all '0'	7.70e-13 J
Read all '1'	8.55e-12 J
Read all '0'	1.42e-13 J

Table 4.8: Simulation Results for 1C2T 10x10 crossbar simulation with digital periphery

Parameter	Value
Output current individual cell LVT	3.8e-5 A
Output current individual cell HVT	4.0e-8 A
Energy best case: Write all '1'	2.19e-11 J
Energy worst case: Write all '0'	3.38e-11 J
Read all '1'	1.79e-11 J
Read all '0'	2.73e-11 J

4.4.4. Area Measurements for Digital Periphery

The area for the buffer and D-flipflop used in the digital periphery as measured using Cadence Layout XL. The layouts can be found in Figure 4.6 and 4.7 respectively.

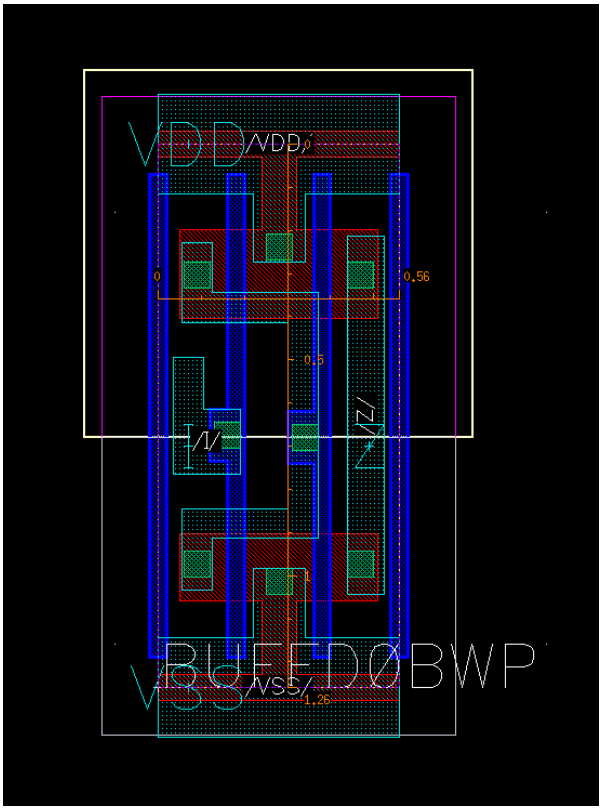

Figure 4.6: Area measurement for Buffer

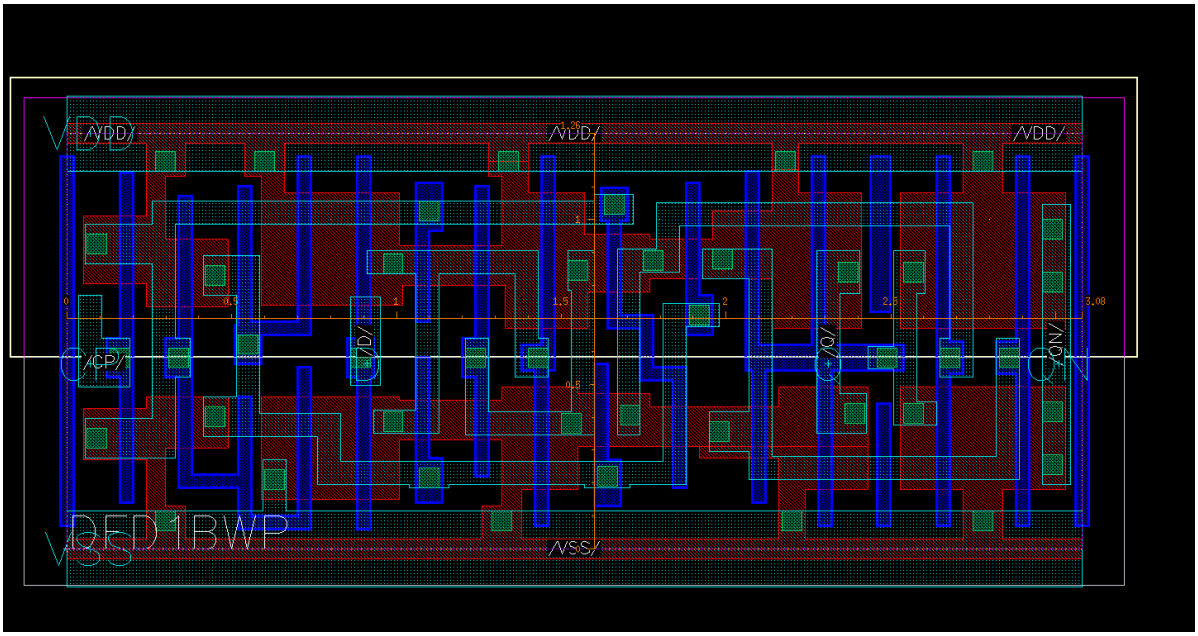

Figure 4.7: Area measurement for D flipflop

The total area of these components can now be calculated by multiplying the respective lengths and

widths. The results of these calculations are displayed in Table 4.14

Table 4.9: Area Results for Digital periphery

Buffer area	$0.7056um^2$
Flipflop area	$3.8808um^2$

The total area of the periphery can be calculated using the following equation:

$$A_{buffer} + 4 \cdot A_{flipflop} = 16.2288um^2 \tag{4.1}$$

per column of the crossbar. The area of the capacitors used in the counter is ignored, because the counter uses ideal capacitors which do not have a layout associated with them.

4.5. Analog Crossbar

Similarly to the digital periphery, the analog periphery also requires usage of a switch. In this case, the switch is used to disconnect the bitline voltage source and to connect the precharged capacitor instead. The full circuit is displayed in Figure 4.8.

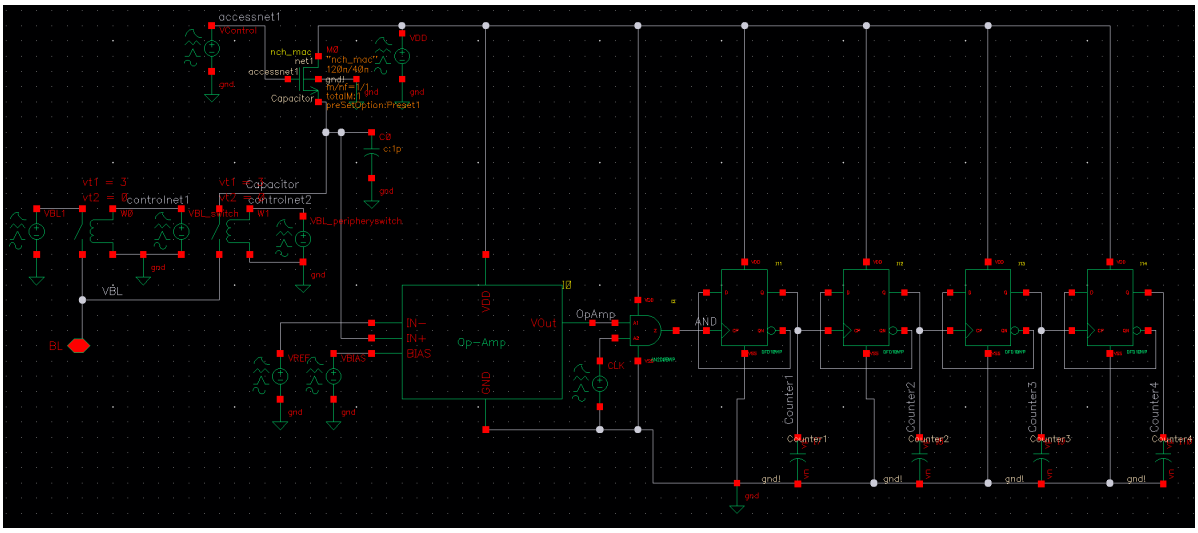

Figure 4.8: Circuit schematic of 2-cell column of a crossbar with analog periphery

4.5.1. Verification: Analog Crossbar

Next, the periphery can be connected to a 2x2 crossbar in order to verify the behaviour of the full system. The parameters for this simulation are displayed in Table 4.10.

Table 4.10: Parameters for Analog crossbar simulations

Parameter	Value
Stored data Column 1	'1' '1'
Stored data Column 2	'1' '0'
Write voltage (gate)	1.8 V
Read voltage (gate)	0.75 V
Read voltage (source)	0.5 V
Access voltage (write)	1.8 V
Access voltage (read)	1.6 V
Write time	200 us
Read time	1 us

The resulting voltages at the output of the Buffer and the Counter are displayed in Figures 4.9 and 4.10

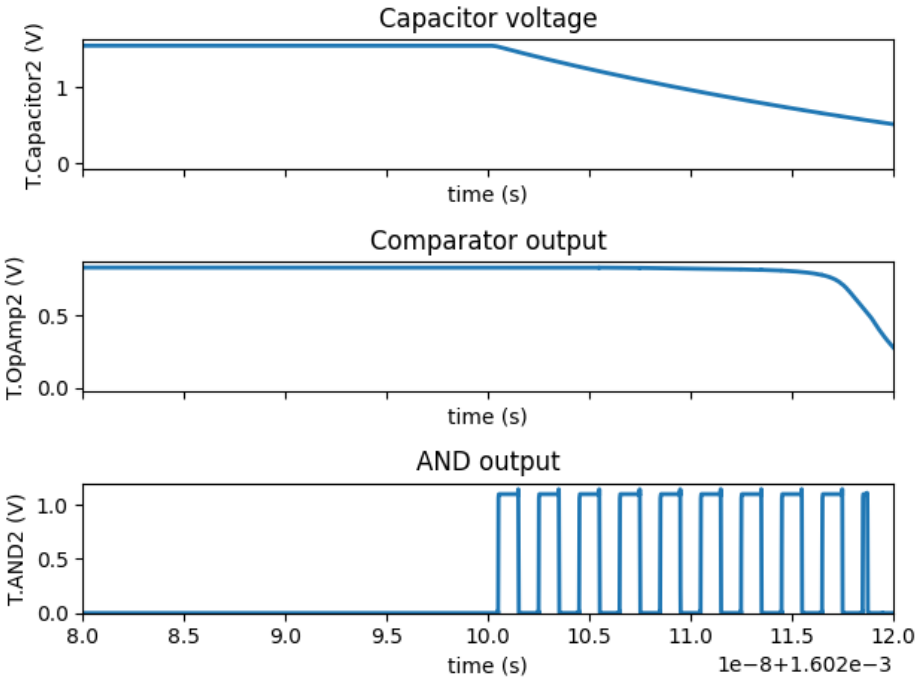

Figure 4.9: Capacitor, Comparator and AND-gate output for column containing '1' '1'

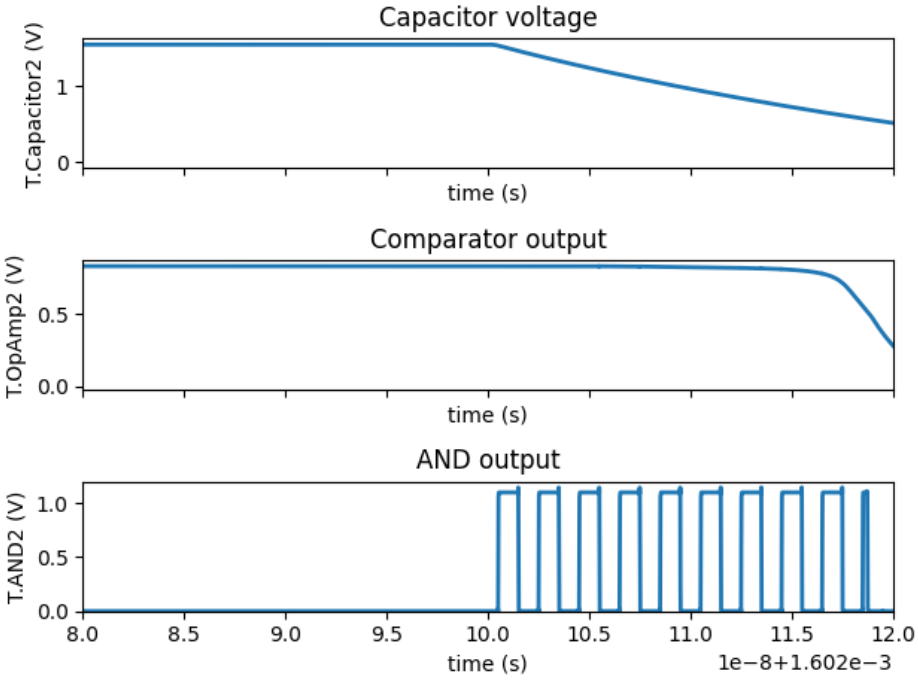

Figure 4.10: Capacitor, Comparator and AND-gate output for column containing '1' '0'

The results show that the amount of pulses sent to the counter is inversely proportional to the amount of devices that are enabled. As such, the circuit is functional.

Table 4.11: Parameters for analog crossbar simulations

Parameter	Value
Write voltage (gate)	1.8 V
Read voltage (gate)	0.75 V
Read voltage (source)	0.5 V
Access voltage (write)	1.8 V
Access voltage (read)	1.6 V
Write time	200 us
Read time	1 us

4.5.2. Analog Crossbar Energy results

Table 4.12: Simulation Results for 1C2T 2x2 crossbar simulation with analog periphery

Parameter	Value
Output current individual cell LVT	3.8e-5 A
Output current individual cell HVT	4.0e-8 A
Energy best case: Write all ’1’	8.69e-13 J
Energy worst case: Write all ’0’	5.47e-13 J
Read all ’1’	8.55e-12 J
Read all ’0’	1.42e-13 J

Table 4.13: Simulation Results for 1C2T 10x10 crossbar simulation with analog periphery

Parameter	Value
Output current individual cell LVT	3.8e-5 A
Output current individual cell HVT	4.0e-8 A
Energy best case: Write all ’1’	2.18e-11 J
Energy worst case: Write all ’0’	3.39e-11 J
Read all ’1’	1.76e-11 J
Read all ’0’	2.73e-11 J

4.6. Area Measurements for Analog Periphery

The area for the comparator and AND-gate used in the analog periphery was measured using Cadence Layout XL. As the digital and analog peripheries utilize the same counter, the area for the counter is the same as the area measured in Section 4.4.1.

4.6.1. Area Measurements for Comparator

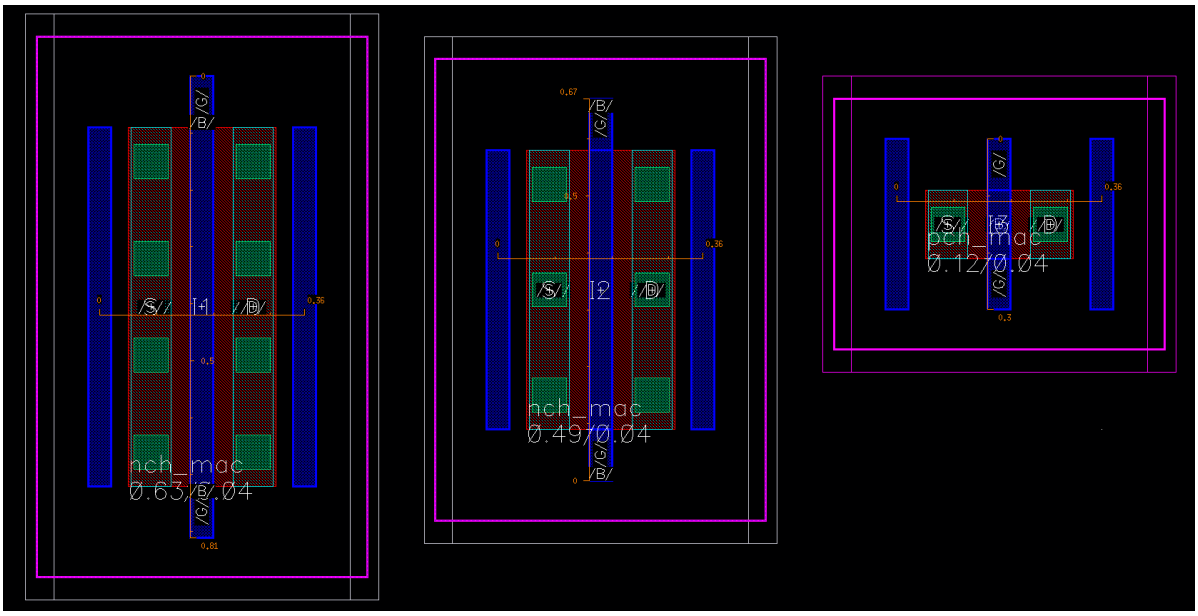

Figure 4.11: Area measurement for transistors of different widths used in the Comparator

Table 4.14: Area Results for Comparator

Bias transistor area	$0.2412um^2$
Input transistor area	$0.2916um^2$
Pmos transistor area	$0.108um^2$

The total area of the comparator can be calculated using

$$A_{total} = A_{bias} + 2 \cdot A_{input} + 2 \cdot A_{pmos} = 1.0404um^2 \tag{4.2}$$

per column of the crossbar.

4.6.2. Area Measurements for AND gate

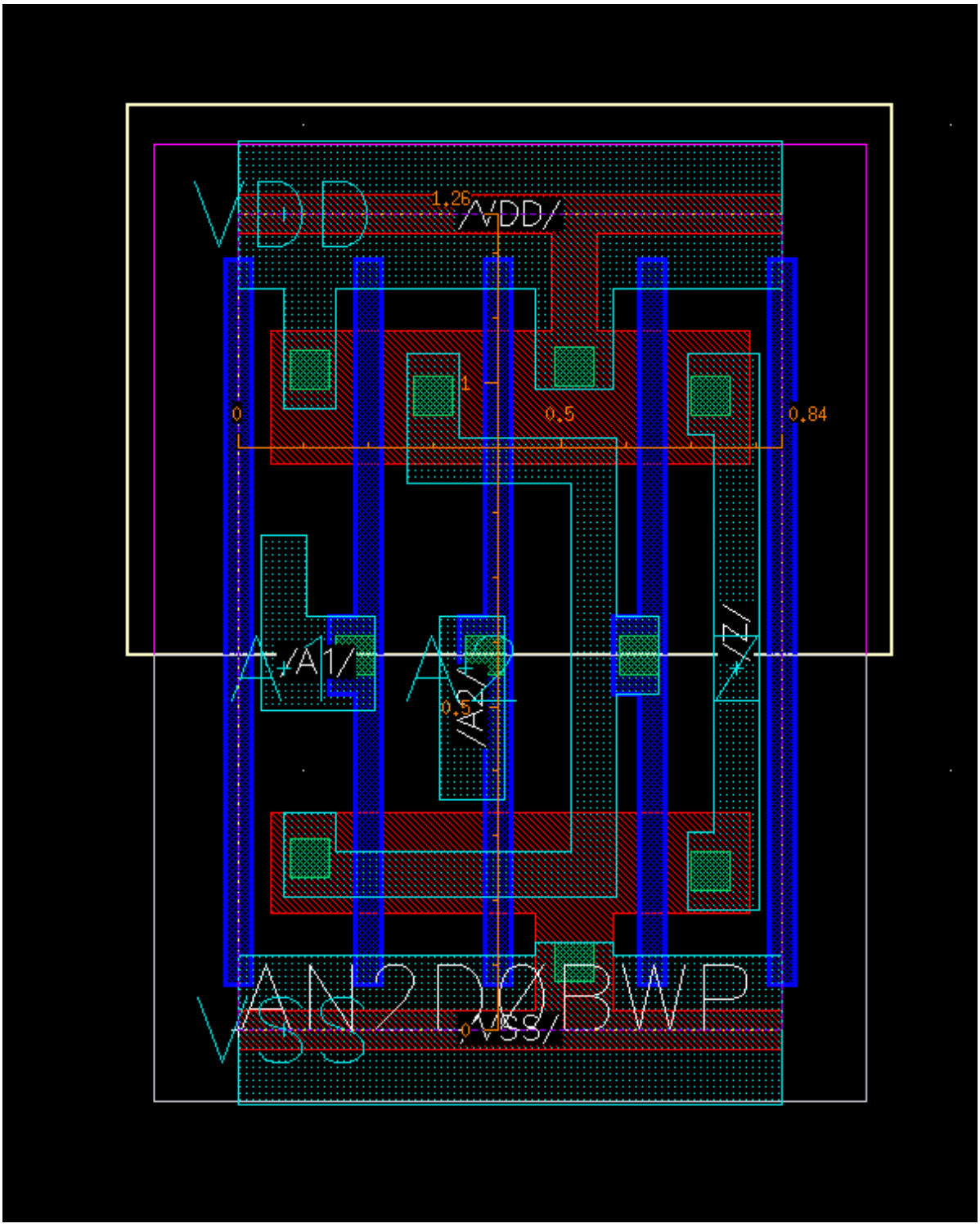

Figure 4.12: Area measurement for inverter and AND gate

By multiplying the width and length, the area of the AND gate was measured to be A = $1.0584um^2$

Finally, the are of the counter will be identical to that of the counter used in the digital periphery, as they use the same counter design. This means the full analog periphery area per column can be calculated

as follows:

$$A_{analogperiphery} = A_{comparator} + A_{buffer} + A_{AND} + A_{counter} \tag{4.3}$$

Filling in the values gives the following calculation:

$$A_{analogperiphery} = 1.0404um^2 + 0.7056um^2 + 1.0584um^2 + 15.5232um^2 = 18.3276um^2 \tag{4.4}$$

As such, the analog periphery requires an additional area of $18.3276um^2$ per column.

4.7. On-Chip Learning simulation

Because the FeCAP model can not be initialized at an arbitrary value and a series of write pulses is required instead, the complexity and thus duration of the simulations is very high. Combining the crossbar with the periphery increases the complexity even further and as such simulations were kept at a 10x10 size for the simulations.

A single iteration of the learning task consists of a forwards pass performed in the analog crossbar, a backwards pass performed in the digital crossbar and finally a weight update performed on both crossbars at the same time.

4.7.1. Energy consumption

The energy consumed by a full step of the learning task can be calculated using:

$$E_{tot} = E_{ForwardPass} + E_{BackwardPass} + E_{WeightUpdate} \tag{4.5}$$

For the worst case of having to write all 0's to both crossbars and having to read all 0's, that means:

$$E_{tot} = E_{readall0analog} + E_{readall0digital} + E_{writeall0analog} + E_{writeall0digital} \tag{4.6}$$

Filling in the values:

$$E_{tot} = 2.73e-11 + 2.73e-11 + 3.38e-11 + 7.70e-13 = 8.91e-11J \tag{4.7}$$

This means a full step of the learning task requires 89.1 pJ of energy.

4.7.2. Area

In terms of area, this approach requires a 10x10 digital crossbar and a 10x10 analog crossbar. The area for the analog crossbar can then be calculated using:

$$A_{analog} = (n_{columns} \cdot A_{analogperiphery}) + (n_{cells} \cdot A_{1c2tcell}) \tag{4.8}$$

Filling in the values calculated previously gives

$$A_{analog} = (10 \cdot 18.32um^2) + (100 \cdot 0.216um^2) = 204.8um^2 \tag{4.9}$$

Note that this area is dominated by the periphery. The area for the digital crossbar can then be calculated using:

$$A_{digital} = (n_{columns} \cdot A_{digitalperiphery}) + (n_{cells} \cdot A_{1c2tcell}) \tag{4.10}$$

Filling in the values calculated previously gives

$$A_{digital} = (10 \cdot 16.22um^2) + (100 \cdot 0.216um^2) = 183.8um^2 \tag{4.11}$$

The total area can then be calculated by summing the analog and digital areas:

$$A_{total} = A_{analog} + A_{digital} = 183.8um^2 + 204.8um^2 = 388.2um^2 \tag{4.12}$$

This means the full area taken up by the two 10x10 crossbars and their associated periphery equals 388.2 um^2

4.8. Discussion

4.8.1. 1C1T vs 1C2T

For an individual cell, the 1C1T bitcell has both lower area and lower energy consumption than the 1C2T bitcell. Both of these differences can be attributed to the access transistor. However, in larger crossbars the 1C2T bitcell will start to outperform the 1C1T bitcell in terms of energy. For every additional row and column, the 1C1T device requires additional inhibition signals to be generated, each of which consume power. These signals are present for every single write and read pulse, in addition to the extra power needed to write/read the extra rows and column. On the other hand, a 1C2T crossbar read pulse does not require extra inhibition signals to be generated, and as such will scale more easily.

4.8.2. Analog vs Digital Crossbar

In terms of area, the digital crossbar is smaller than the analog crossbar. However, in terms of energy the two crossbars perform very similarly. This suggests that the digital crossbar is more suitable for wearable devices. However, in a neural network application the inference task will generally be performed a lot more often than the training task. As such, it makes sense to optimize for the inference task, in which high parallelism is required. As such, the Analog crossbar is used for the forward pass, while the Digital crossbar is used for the backwards pass.

4.8.3. Comparison to Literature

Because of the limitations of the FeFET model used, it is not possible to simulate a crossbar of a significant size. Similar works using different models and physical FeFET devices are able to generate crossbars with up to 108 Gb worth of FeFET-chips [22] which are not possible to simulate using the available tools. Because of the non-linear scaling of the crossbar energy consumption, it is non-trivial to compare the solutions in a meaningful way.

5

Conclusion

5.1. Conclusion

In this work, a FeFET-based CIM architecture was proposed that supports On-chip learning. The architecture was designed using a bottom-up approach, starting from the FeFET model and working up through the bitcell, 2x2 crossbar, larger MxN crossbar, and finally adding periphery. The design utilizes analog crossbars for forward propagation and the inference task, while using digital crossbars for the backward propagation step. In doing so a design was presented that is optimized for both the inference and the training task.

The presented design is a secure accelerator which can train the network On-Chip and as such, it can be used for data-sensitive applications such as ECG analysis and autonomous driving as opposed to most CIM literature, which focuses exclusively on the inference task and trains off-chip. The accelerator overcomes the memory-wall inherent to Von Neumann machines by embracing the CIM framework and uses FeFET devices to overcome the scaling walls associated with CMOS technology.

5.2. Recommendations for Future Work

Some recommendations for future work are proposed below.

- Investigating multi-bit FeFET cells, which support several intermediate polarization states
- Running larger-scale simulations by utilizing a FeFET model that allows initalization of the polarization at any desired value. The simulation duration was a major bottleneck for this work and using a different FeFET model will allow the crossbar to be scaled up even further.
- Using an integrated FeFET model as opposed to separate FeCAP and transistor models.
- Investigating 2C2T FeFET cells for complementary sensing

Bibliography

[1] Said Hamdioui et al. “Memristor Based Computation-in-Memory Architecture for Data-Intensive Applications”. In: *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Mar. 2015, pp. 1718–1725. (Visited on 03/20/2024).

[2] Anteneh Gebregiorgis et al. “A Survey on Memory-centric Computer Architectures”. In: *ACM Journal on Emerging Technologies in Computing Systems* 18.4 (Oct. 2022), 79:1–79:50. DOI: 10.1145/3544974. (Visited on 03/20/2024).

[3] Ping Chi et al. “PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-based Main Memory”. In: *SIGARCH Comput. Archit. News* 44.3 (June 2016), pp. 27–39. DOI: 10.1145/3007787.3001140. (Visited on 11/22/2024).

[4] Ali Shafiee et al. “ISAAC: A Convolutional Neural Network Accelerator with in-Situ Analog Arithmetic in Crossbars”. In: *ACM SIGARCH Computer Architecture News* 44.3 (Oct. 2016), pp. 14–26. DOI: 10.1145/3007787.3001139. (Visited on 10/11/2024).

[5] Ziyu Wang et al. “Safe, Secure and Trustworthy Compute-in-Memory Accelerators”. In: *Nature Electronics* 7.12 (Dec. 2024), pp. 1086–1097. DOI: 10.1038/s41928-024-01312-y. (Visited on 08/01/2025).

[6] Anteneh Gebregiorgis et al. “Tutorial on Memristor-Based Computing for Smart Edge Applications”. In: *Memories - Materials, Devices, Circuits and Systems* 4 (July 2023), p. 100025. DOI: 10.1016/j.memori.2023.100025. (Visited on 03/20/2024).

[7] Hoang Anh Du Nguyen et al. “A Classification of Memory-Centric Computing”. In: *ACM Journal on Emerging Technologies in Computing Systems* 16.2 (Apr. 2020), pp. 1–26. DOI: 10.1145/3365837. (Visited on 03/20/2024).

[8] M Stewart et al. “Ferroelectric hysteresis measurement and analysis”. In: *Report CMMT (A)* 152 (Jan. 1999).

[9] Dong Han Ko et al. “Comparative Analysis and Energy-Efficient Write Scheme of Ferroelectric FET-Based Memory Cells”. In: *IEEE Access* 9 (2021), pp. 127895–127905. DOI: 10.1109/ACCESS.2021.3111913. (Visited on 06/20/2024).

[10] Keiron O’Shea et al. *An Introduction to Convolutional Neural Networks*. Dec. 2015. DOI: 10.48550/arXiv.1511.08458. arXiv: 1511.08458 [cs]. (Visited on 03/20/2024).

[11] Rikiya Yamashita et al. “Convolutional Neural Networks: An Overview and Application in Radiology”. In: *Insights into Imaging* 9.4 (Aug. 2018), pp. 611–629. DOI: 10.1007/s13244-018-0639-9. (Visited on 07/24/2025).

[12] Hanshi Sun et al. *Arrhythmia Classifier Using Convolutional Neural Network with Adaptive Loss-aware Multi-bit Networks Quantization*. Feb. 2022. DOI: 10.48550/arXiv.2202.12943. arXiv: 2202.12943 [eess]. (Visited on 05/23/2025).

[13] Ellina Zhang et al. “Manhattan Rule for Robust In-Situ Training of Memristive Deep Neural Network Accelerators”. In: *2024 IEEE 67th International Midwest Symposium on Circuits and Systems (MWSCAS)*. Aug. 2024, pp. 1324–1328. DOI: 10.1109/MWSCAS60917.2024.10658688. (Visited on 05/09/2025).

[14] Ruggero Donida Labati et al. “Deep-ECG: Convolutional Neural Networks for ECG Biometric Recognition”. In: *Pattern Recognition Letters*. Robustness, Security and Regulation Aspects in Current Biometric Systems 126 (Sept. 2019), pp. 78–85. DOI: 10.1016/j.patrec.2018.03.028. (Visited on 07/25/2025).

[15] Abu Sebastian et al. “Memory Devices and Applications for In-Memory Computing”. In: *Nature Nanotechnology* 15.7 (July 2020), pp. 529–544. DOI: 10.1038/s41565-020-0655-z. (Visited on 11/22/2024).

[16] Hai Jin et al. *ReHy: A ReRAM-Based Digital/Analog Hybrid PIM Architecture for Accelerating CNN Training*. 2022. DOI: 10.1109/TPDS.2021.3138087. (Visited on 11/22/2024).

[17] Cédric Marchand et al. “FeFET Based Logic-in-Memory: An Overview”. In: *2021 16th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*. June 2021, pp. 1–6. DOI: 10.1109/DTIS53253.2021.9505078. (Visited on 06/26/2024).

[18] Shyh-Shyuan Sheu et al. “Fast-Write Resistive RAM (RRAM) for Embedded Applications”. In: *IEEE Design & Test of Computers* 28.1 (Jan. 2011), pp. 64–71. DOI: 10.1109/MDT.2010.96. (Visited on 07/24/2025).

[19] Sven Beyer et al. “FeFET: A Versatile CMOS Compatible Device with Game-Changing Potential”. In: *2020 IEEE International Memory Workshop (IMW)*. May 2020, pp. 1–4. DOI: 10.1109/IMW48823.2020.9108150. (Visited on 05/30/2024).

[20] J. Ajayan et al. “Ferroelectric Field Effect Transistors (FeFETs): Advancements, Challenges and Exciting Prospects for next Generation Non-Volatile Memory (NVM) Applications”. In: *Materials Today Communications* 35 (June 2023), p. 105591. DOI: 10.1016/j.mtcomm.2023.105591. (Visited on 06/13/2024).

[21] Lars Hoogland. *FeFET-based On-chip Learning for CNN repository*. URL: https://github.com/Larshoog/Thesis.

[22] Wonbo Shim et al. “Ferroelectric Field-Effect Transistor-Based 3-D NAND Architecture for Energy-Efficient on-Chip Training Accelerator”. In: *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* 7.1 (June 2021), pp. 1–9. DOI: 10.1109/JXCDC.2021.3057856. (Visited on 08/02/2025).