Evaluation of Design Metrics for DFT Implemented Using In-Memory Computing

Sofia Tatidis & Philip Nielsen
so2107ta-s@student.lu.se, philipnielsen64@gmail.com

Department of Electrical and Information Technology
Lund University & Ericsson

Supervisors: Liang Liu & Mojtaba Mahdavi

Examiner: Erik Larsson

March 18, 2026

© 2026
Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

The discrete Fourier transform (DFT) is one of the most important algorithms commonly used in baseband processing. While traditional hardware implementations like the Fast Fourier Transform (FFT) are highly optimized, they heavily suffer from the von Neumann bottleneck. To overcome this, performing the computations directly in memory using memristive crossbar arrays is a promising solution.

This thesis evaluates different simulators for analog in-memory-computing and found NeuroSim to be the most suitable for simulating power, performance, area and accuracy. Modifications to the simulator are proposed to support the assessment of three different crossbar-based designs for computing the DFT. To investigate whether the DFT can be reliably implemented in the analog domain without severe accuracy degradation, a Ferroelectric tunnel junction memristor was chosen due to its high resistance and inherent robustness against non-idealities like IR drop. We demonstrate that a symmetry design, which stacks the twiddle coefficients into one crossbar and leverages the conjugate symmetry of the DFT, is the most optimal for real-valued inputs. This design effectively reduces the hardware cost of the peripherals, reducing the crossbar area by 50% compared to a naive implementation.

For a small 64-point DFT, our evaluations show that the system can achieve a mean square error of magnitude 10^{-3}. However, scaling to a large 1024-point DFT in a single crossbar introduces significant IR drop, resulting in an increased accuracy degradation. To mitigate this, a tiled architecture is adopted. As tiling significantly increases energy and area due to the overhead of multiple analog-to-digital converters, a tradeoff analysis is performed. Square tiles of size $T = 1024$ are found to be the most optimal, effectively reducing the error margin to 2×10^{-2} while maintaining low energy consumption and latency. Finally, the results show that a single crossbar implementation consumes approximately 8.65 nJ, about half the energy of highly optimized CMOS FFTs, whereas the tiled architecture requires approximately 15.6 nJ

Popular Science Summary

In an increasingly connected world, the demand for fast wireless communication, like 5G and 6G, is rapidly growing. To process these wireless signals, our devices heavily rely on a crucial mathematical algorithm called the discrete Fourier transform (DFT). However, traditional computers have an architectural limit. They suffer from the *von Neumann bottleneck*, which means they waste massive amounts of time and energy simply moving data back and forth between the memory unit and the processing unit.

A solution to overcome this bottleneck is to perform the calculations directly inside the memory itself. This can be achieved with the emerging devices called *memristors*. Arranging these analog components in a grid-like structure called a crossbar array, we can utilize the current flowing through it, to do math directly inside the arrays. Using this can drastically reduce time complexity and save power.

In this project, our main purpose was to simulate and evaluate the best way to build a DFT calculator using these memristor crossbars. We proposed and tested three different hardware designs:

- Baseline: A standard, straightforward implementation of the DFT using differential pairs.
- Merged: A design where we stacked the calculation matrices together to significantly reduce the hardware cost of expensive peripherals, like analog-to-Digital Converters (ADCs).
- Symmetry: A highly optimized design that leverages the natural mathematical symmetry of the DFT to cut the required crossbar area completely in half for real-valued inputs.

Using a modified NeuroSim simulator, we found our Symmetry design consumes only 8.65 nJ of energy. This is roughly half the energy required by highly optimized traditional hardware setups. However, scaling up to larger computations (like a 1024-point DFT) introduces a major challenge known as *IR drop*. This phenomenon is linked to the natural resistance of longer interconnecting wires, causing a voltage loss that heavily degrades calculation accuracy.

To mitigate this IR drop, we introduce *tiling* to break the massive crossbar into smaller, interconnected grids. Furthermore, because adding tiles requires expensive extra peripherals (like the previously mentioned ADCs) that increase

energy and area costs, we performed a tradeoff analysis. We found that a square tile size of 1024 offers the most optimal balance between energy efficiency and accuracy for a 1024-point DFT.

Acknowledgments

This thesis has been the ultimate test of perseverance and dedication, the culmination of all the hard work put into our engineering degrees. It has opened our eyes to a whole new field of engineering, pushed and inspired us to go beyond what we initially thought was possible.

Throughout this project we have had the never-ending support of our main supervisor Mojtaba Mahdavi at Ericsson. Thank you for your patience with us and for taking a chance on two Computer Science students with no prior knowledge about memristors! It has been such a privilege to work with you and have your guidance. We would also like to thank our academic supervisor Liang Liu for your invaluable guidance and insights for this project. Thank you for making the introduction to Mojtaba and for always being available for questions and a second opinion.

We are also very grateful to the entire DPR department at Ericsson Lund for hosting us. A special thanks to Alexander Hunt, our manager, and to Martin Ek and Ziyin Peng for making the office a fun place to be! To all of the classmates who helped and motivated us along the way; thank you for taking us to the finish line! Without you we would still be stuck doing multicore labs and taking the functional programming re-exam.

Finally a huge thanks to our friends and family for the emotional support during this project. For always believing in us, for listening to our rants when the project proved challenging and for celebrating all the small victories along the way.

Thank you all for helping us reach the end of this academic journey!

Sofia Tatidis & Philip Nielsen
Winter 2026

Contribution Statement

Part	Sofia/Philip	Comments
Litterature study	50/50	Simulator investigation & previous work
Device research	100/00	Research of a suitable device with sufficient data characterized for simulation
Behavioral simulator	100/00	Extend the chosen simulator to support custom design
Hardware analyzer	00/100	Extend the chosen simulator to support custom design
Writing		
Abstract & Pop. Sci. Sum.	5/95	Philip main author, Sofia provided peer review
Introduction	95/5	Sofia main author, Philip found some of the previous work
Background	95/5	Sofia main author, Philip provided one subsection
Simulator Investigation	40/60	Both summarized simulator functionalities, Philip wrote conclusion
Method	50/50	Philip wrote about HA and IMC designs, Sofia wrote about BS, tiling and device, system & peripherals
Results	50/50	Sofia compiled accuracy results while Philip prepared the PPA results
Discussion	50/50	Sofia dicussed accuracy, while Philip discussed PPA
Conclusions	95/5	Sofia main author, Philip supplemented with simulator choice summary
Further work	50/50	Sofia wrote about accuracy improvement research, while Philip proposed extensions to the HA

Acronyms

1T1R One-Transistor–One-Resistor

ACIM Analog Computing-in-Memory
ADC Analog-to-Digital Converter

BER Bit Error Rate
BS Behavioral Simulator

CA Crossbar array
CIM Computing-in-Memory
CMOS Complementary Metal-Oxide-Semiconductor

DCT Discrete Cosine Transform
DFT Discrete Fourier Transform
DR Dynamic Range

eNVM Emerging Non-Volatile Memory

FeFET Ferroelectric Field-Effect Transistor
FeRAM Ferroelectric Random Access Memory
FFT Fast Fourier Transform
FTJ Ferroelectric Tunnel Junction

GPU Graphics Processing Unit

HA Hardware Analyzer
HRS High Resistance State

IDFT Inverse Discrete Fourier Transform

LRS Low Resistance State

MAC Multiply–Accumulate

MIR Memristive Image Reconstructor

ML Machine Learning

MRAM Magnetoresistive Random Access Memory

MS/s Mega Samples per Second

MSE Mean Square Error

NMC Neuromorphic Computing

NS NeuroSim

PCM Phase Change Memory

PIM Processing-in-Memory

PPA Power, Performance, and Area

RRAM Resistive Random Access Memory

SAR-ADC Successive Approximation Register Analog-to-Digital Converter

SNDR Signal-to-Noise-and-Distortion Ratio

SNR Signal-to-Noise Ratio

VMM Vector–Matrix Multiplication

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Research Context

In an increasingly connected world, the demand for fast wireless communication is rapidly growing. 5G has raised the bar for processing latency in digital baseband ASICs and 6G will raise it further in the coming years. Ericsson is a world leader in telecommunications who designs processors for baseband processing in the mobile communication domain.

The discrete Fourier transform (DFT) is commonly used in baseband processing, for instance it is used in orthogonal frequency division multiplexing [1], for uplink and downlink and filtering etc. A faster implementation of DFT is the Fast Fourier transform (FFT), which decreases the time complexity from $O(N^2)$ to $O(N \log N)$.

According to 5G standards, modern applications require extremely high data rates (up to 20 Gbps) and low latency (under 4 ms) [2]. To achieve this, 5G systems currently utilize OFDM with FFT sizes scaling up to 4096 points [3], and future 6G networks are expected to push this requirement to 8192 points or more [4]. Implementing FFT in traditional hardware is very memory intensive and has a high hardware cost and complexity that scales heavily with the size of the input [1]. While modern hardware can compute individual transforms quickly, the massive parallelism required in 5G/6G (Massive MIMO) leads to an unsustainable energy consumption. This is primarily due to the von Neumann bottleneck, where the constant movement of these massive data arrays between memory and processing units heavily dominates the power budget [5].

Performing in-memory computation (IMC) is potentially a way to overcome this bottleneck, as it eliminates the need to constantly move data. Doing the processing in arrays of memristive devices can instead reduce the time complexity to $O(1)$.

1.2 Thesis Scope

The project aims to simulate small and large DFTs implemented in memristor crossbar arrays to evaluate the performance metrics of the system. The implementation is assessed with respect to accuracy and performance, power/energy consumption and area (PPA).

Different simulators with the capability to simulate memristors and crossbar arrays are investigated and compared to determine whether the design can be implemented and the desired performance metrics can be evaluated. Three designs for IMC based DFT with different levels of optimization are proposed. Device parameters and non-idealities along with system-level noise are researched and injected to the simulations to find the impact on accuracy.

A small DFT (64 point) is implemented and simulated to find the performance metrics of a small system and evaluate which design is better. Additionally a larger DFT (1024 point) is implemented in the most optimal design for real input. To mitigate the static IR drop in large crossbar arrays tiling is introduced. The challenges that arise with increasing the size are evaluated and design tradeoffs are suggested. The PPA is compared to traditional hardware DFT implementations.

The thesis is limited to the IMC aspect of the DFT accelerator. It only briefly touches on interfacing the DFT design with a larger system such as traditional computing and I/O. The input is assumed to be analog and the output is converted to the digital domain for the final steps of the processing.

1.3 Previous Work

Analog in-memory computation (AIMC) implementations of DFT with memristor crossbar arrays have been evaluated in [1, 6–10]. Some of the designs are evaluated by simulating with hardware effects [6, 7, 10], while others are only evaluated theoretically [1] or without disclosing which hardware parameters are chosen [8, 9].

The design in [1] uses a novel coefficient mapping method to minimize the number of columns in the coefficient matrix. Furthermore, the entire output signal is created in the analog domain and only converted to the digital domain at the last step. In contrast, [6] performs ADC directly at the output of the crossbars moving more of the computation to the digital domain. The paper compares three DFT designs in terms of bit-error-rate (BER), energy and area as the result of coefficient bitwidth. However, the BER compares the original bit stream with the output after performing DFT and IDFT in the AIMC designs. The device used is a 2-bit RRAM device.

A memristive image reconstructor (MIR) for medical purposes is implemented in [7] using RRAM in the 1T1R-configuration with $0.13\mu m$ Si CMOS process. Both DFT and IDFT are implemented with memristor arrays as a part of the MIR. The MIR is experimentally shown to have significant speed-up and energy efficiency compared to using a conventional GPU for the same task.

The Discrete Cosine Transform (DCT) is closely related to the DFT but only uses the real part of the coefficients. In [10, 11], DCT is implemented with memristor crossbar arrays for image compression. For transform size (N) of 128 and 256, the result retains good accuracy compared with current methods and achieves constant time complexity [10] . Similar results were found in [11], but without evaluating system-level non-idealities.

To summarize, the feasibility of using memristor crossbar arrays to implement DFT have been previously evaluated and found to be promising thanks to low

power usage and high integration density. A number of designs have been proposed and evaluated. However, in simulations of AIMC systems, not all previous work incorporates device and system non-idealities as a part of the analysis which might prove unrepresentative of real life applications.

Chapter 2
Background

In order to implement DFT with in-memory computing, memristors are used. They are a class of non-volatile memory that have been found to have desirable properties such as small integration area and low power. By chaining them into crossbar arrays they can realize vector matrix multiplication. However there are a number of device and system non-idealities that arise from the analog nature of this type of IMC designs. In this chapter we present some background on these topics and techniques to minimize the effect of the system imperfections.

2.1 Discrete Fourier Transform

The DFT is an important transform used in wireless communication and other signal processing heavy fields. The DFT transforms a time series to the frequency domain and is defined as follows:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, 0 \leq k \leq N-1 \tag{2.1}$$

Where N is the length of the DFT, and can be expressed as vector matrix multiplication if input samples $x(n)$ and outputs $X(k)$ are expressed as column vectors $\boldsymbol{x}$ and $\boldsymbol{X}$. Then the DFT can be implemented as:

$$\mathbf{X} = \mathbf{W}_N \mathbf{x} \tag{2.2}$$

Where $\mathbf{W}_N$ is called the DFT matrix [12], or the transform matrix. The elements in the DFT matrix are called the twiddle factors and are defined as:

$$W_N^{nk} = e^{-j\frac{2\pi nk}{N}} \tag{2.3}$$

The elements in $\boldsymbol{x}$ are typically complex valued and represented as $a + bj$. So (2.1) can be expressed as

$$X(k) = \sum_{n=0}^{N-1} (a(n) + j \times b(n)) W_N^{nk}, 0 \leq k \leq N-1 \tag{2.4}$$

$$= \sum_{n=0}^{N-1} a(n) W_N^{nk} + j \times \sum_{n=0}^{N-1} b(n) W_N^{nk} \tag{2.5}$$

Using the vector notation introduced in (2.2), (2.5) can be expressed as

$$\mathbf{X} = \mathbf{W}_N a + j \times \mathbf{W}_N b \tag{2.6}$$

2.1.1 Traditional Hardware FFT Implementations

There are a number of approaches to implementing FFT in traditional hardware. In [1], a number of approaches have been compared. The computational complexity of the compared designs range from $O(N \log N)$ to $O(N \log_2 N)$. Traditional hardware implementations also suffer from the von Neumann bottleneck, have large memory requirements, high hardware cost and complexity and are limited in how much of the processing can be parallelized which in turn limits the throughput [1].

2.2 The Memristor

First proposed in 1971 by Leon Chua, the memristor was not discovered until 2008 at HP labs [13, 14]. Alongside the resistor, capacitor, and inductor, the memristor is the fourth fundamental electronic component, uniquely linking magnetic flux and electric charge [13].

Using memristors higher density and lower power can be achieved compared to CMOS technologies [8]. The memristor has been suggested as non-volatile memory and some papers, among them [15, 16], have proposed how to implement logic circuits with memristors. This could enable in-memory computing which would bridge the von Neumann bottleneck and increase the integration density even in a post-moore's law era.

For this thesis project, the memristor crossbar array configuration is explored as a way to implement in-memory computing.

2.2.1 Properties

The memristor behaves like a linear resistor with memory but with some nonlinear characteristics [17]. It can be current or voltage controlled depending on the device. The time-invariant behavior of a voltage controlled memristor is

$$\begin{aligned} \frac{dx}{dt} &= f(x, v) \\ i(t) &= G(x)v(t) \end{aligned} \tag{2.7}$$

where x is the state, $v(t)$ is the memristive device voltage, $i(t)$ is the current of the memristive device, $G(x)$ is the conductance of the device and t is the time.

The memristor conductance can be programmed with high voltage to change its value, and read with low voltage without disturbing its state [18]. High resistance in a memristor corresponds to low conductance, as such the high resistance state (HRS) is equivalent to the off state of the device. Using the same reasoning,

the low resistance state (LRS) corresponds to the on-state. The dynamic range (DR) of a memristor is expressed as the ratio

$$DR = \frac{G_{max}}{G_{min}} \tag{2.8}$$

between the LRS and HRS conductances.

2.2.2 Different Devices

Extensive investigations of Resistive Random Access Memory (RRAM) crossbar array enabled accelerators for neural networks have been made [19, 20].

Other non-volatile memory technologies, including RRAM, phase change memory (PCM), ferroelectric memory (FeRAM) and ferroelectric transistor (FeFET), and magneto-resistive random access memory (MRAM), have been explored for neuromorphic computing in the roadmap presented in [21]. The roadmap also maps out the device non-idealities that are commonly found when using these devices, such as nonlinearity, conductance drift, low resolution, device-to-device variability and read noise.

In [22], ferroelectric tunnel junctions (FTJ) are found to be a promising device for analog in-memory computing (AIMC) thanks to the low conductance level. The paper also notes that FTJ-based systems are sensitive to short-circuit failure of individual devices, and proposes that safeguards needs to be introduced to the system.

2.2.3 Non-Idealities of Memristors

While utilizing memristor crossbar arrays for in-memory computing is a promising in terms of power consumption, performance and integration density there are some drawbacks due to the non-ideal properties of memristor devices. Different classes of non-idealities are more common with certain devices. The following sections outlines common ones.

Conductance Variation

Real-world memristors will experience variations in their conductance levels. This class of non-ideality is called device-to-device variation. A number of devices all programmed to state X will not have the exact same conductance, but vary various amounts depending on the device.

The conductance variation is one of the limiting factors for how many states a device can support. A memristor with a conductance range of G0 to G1 can only support a number of states where there is some room to the next state so that the standard deviation is not too large and pushes that state to a state next to it. If the states are too close together, the conductance variation (and other noise) might cause the memristor to not reliably switch between states.

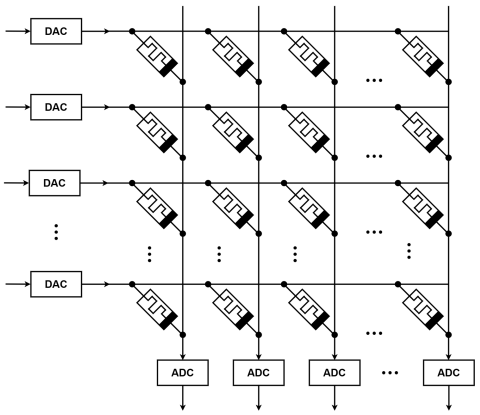


Figure 2.1: The memristor crossbar array with data converters

Read Noise

Read noise is the variation of the read values of the memristors. In [7], the read noise is not found to impact the accuracy of the MRI significantly thanks to low noise levels.

The authors of [6] propose a novel adaptive variability aware crossbar mapping scheme to mitigate the effects of read noise. In the end-to-end simulation framework, the BER is significantly improved with this novel mapping. From this work it is evident that the mapping scheme has significant impact on the noise sensitivity of the system.

Conductance Drift

The conductance of a memristor can start to drift with time. The speed of the drift determines the reliability of the system as well as how often a memristor needs to be reprogrammed to maintain accuracy.

Conductance drift is a significant issue with PCM memristors [23], but it also occurs for other types of memristors. The conductance drift of analog RRAM hinders its practical applications according to [24]. While FTJ memristors are found to have low, negligible, drift [22].

2.3 Crossbar Array

By placing memristors in a crossbar array, several parallel multiply-accumulate operations can be performed. The relationship between current, voltage and conductance of the memristor device, as defined in (2.7), correlates to Ohm's law. Consider also Kirchhoff's law to accumulate the current at the output node and we have a multiply-accumulate operation along one column in the crossbar array.

Figure 2.1 shows how the crossbar array is structured. The columns in the array are called bit lines, while the input rows are called word lines. The crossbar array can thus be used to implement a vector matrix multiplication (VMM) by

performing the dot product of the input vector and the values programmed on the crossbar columns as

$$\boldsymbol{I} = \boldsymbol{G}\boldsymbol{V} \tag{2.9}$$

where $\boldsymbol{V}$ is the input vector, $\boldsymbol{G}$ is the conductance matrix and $\boldsymbol{I}$ is the vector of current signals generated in the crossbar columns. $\boldsymbol{I}$ can be expressed as

$$I_j = \sum_i V_i G_{i,j} \tag{2.10}$$

Where I_j is the current at the output of bit line j, V_i is the input voltage of word line i and $G_{i,j}$ is the j-th entry of i-th row in matrix $\boldsymbol{G}$. Depending on the input and output domains of the crossbar array, DACs and ADCs might be placed at the input or output.

2.3.1 Differential Representation

The twiddle coefficients from (2.3) have values ranging from -1 to 1 since they are derived from the unit circle. The sign of the coefficients can be both positive and negative. However, memristors can only be programmed with positive conductance. In order to represent both positive and negative coefficients we employ differential pairs.

The differential pair consists of two memristors programmed with values so that the difference is the desired conductance. The relationship between the output current of one bit line as a function of the differential pairs of memristor conductances are described in (2.11) with $G^+_{i,j}$ and $G^-_{i,j}$ forms one differential pair. G^+ and G^- are the differential crossbar arrays.

$$I_{\text{out},j} = \sum_i [V_i G^+_{i,j} + (-V_i) G^-_{i,j}] = \sum_i V_i (G^+_{i,j} - G^-_{i,j}) \tag{2.11}$$

(2.11) shows the relationship between the differential pair and the subsequent output current derived from the differential pair. (2.11) is derived from [25].

Using differential pairs results in doubling the amount of devices to represent the coefficient matrix. When employing differential crossbars (two separate crossbars) the combined output is computed with adders at the outputs. The G^+ crossbar is programmed with all positive coefficients while G^- is programmed with the absolute value of the negative coefficients. The rest of the devices in the differential crossbars are programmed to 0 as to create the differential value. For instance if the desired conductance of $G_{i,j}$ is $-0.5S$, then program $G^+_{i,j} = 0S$ and $G^-_{i,j} = 0.5S$:

$$G_{i,j} = G^+_{i,j} - G^-_{i,j} = 0S - 0.5S = -0.5S$$

Inputs can also be represented in this manner. The implications are that the positive part of the input will be input to a pair of differential crossbar arrays, while the negative part is input to another pair. This approach requires four crossbar arrays to completely process the input. Another approach is to multiplex the input to the differential pair. Then only two crossbars are needed, but instead the operation will take two clock cycles per input. While there are other ways of

representing negative weights in the context of memristors, the differential representation has been found to be robust toward device and read variations [26] [27].

2.3.2 IR Drop in Crossbar Arrays

IR drop is the reduction of voltage that occurs due to the intrinsic resistance and current in metal wires. The IR drop increases with the length of the wire. In crossbar arrays, the IR drop is a severe bottleneck since the wire resistance increases quadratically with the number of rows and columns [28].

This is an important consideration when sizing crossbar arrays. Creating bit and word lines that are too long with many memristors attached can be problematic. In order to avoid inaccuracies from IR drop, the maximum size of the arrays must be considered. Furthermore, the use of FTJ memristors helps mitigate this issue due to their low conductance. This reduces the total current flowing through the wires, thus reducing the IR drop [28].

There exists two types of IR drop: static and dynamic. The static IR drop is the average voltage drop whereas dynamic IR drop depends on the switching activity of the design [29]. In this thesis we will primarily discuss static IR drop unless otherwise specified. While the impact of dynamic IR drop in crossbar arrays is important to evaluate, the simulation tools chosen for this thesis project lacked the capabilities to model this.

2.3.3 Tiling

To mitigate the IR drop effect the crossbar array can be divided into smaller arrays called tiles. The tiles can have any shape as long as the size is smaller than the original crossbar. In order to reconstruct the output of the crossbar first the outputs are added column by column. The full output is created by concatenating the output vectors.

Introducing more tiles in the vertical direction will increase the number of ADC needed since the columns are broken up and each partial result requires ADC. Additional peripheral circuitry will also be introduced to support the reconstruction of the full output.

2.3.4 Coefficient Slicing

When the dynamic range of a memristor is not large enough to support the full coefficient bitwidth the bits can be distributed across multiple memristors placed in adjacent columns. This technique is called coefficient slicing.

For instance, consider a 4-bit memristor and 8-bit weights. Two memristors are used for each weight, where the first four bits are stored in the first one, and the rest in the other as illustrated in Figure 2.2. To reconstruct the full output value some additional Shift-and-Add periphery is needed.

There are two approaches to coefficient slicing, the least significant bit (LSB) method and the most significant bit (MSB) method. For the LSB method, the least significant bits are mapped into devices first. While for the MSB method

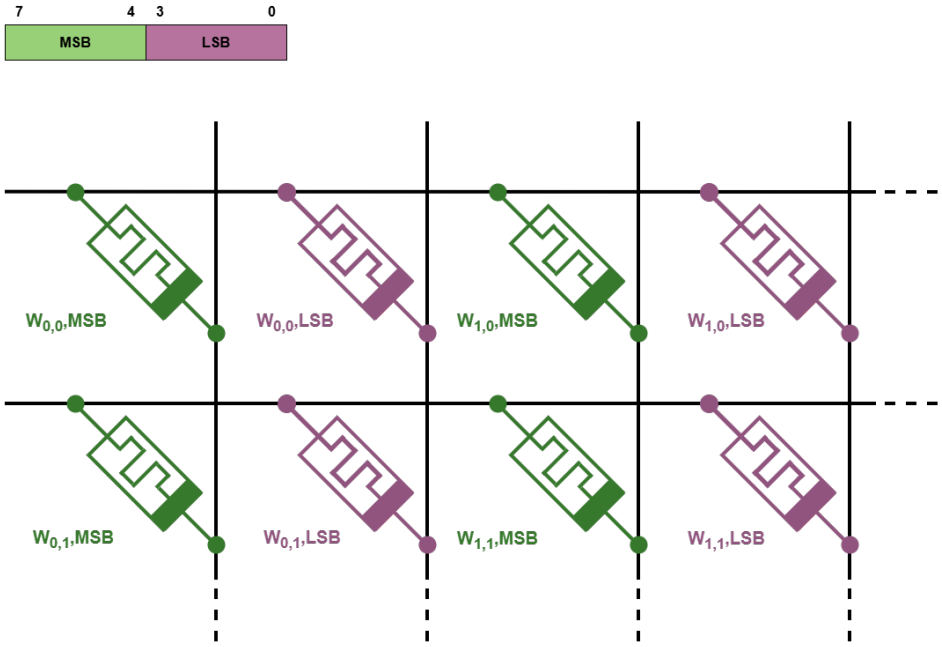


Figure 2.2: Coefficient slicing

the most significant bits are mapped into devices first. While these two methods appear to be the virtually the same (and in terms of hardware operations they are in fact the same) the implications are notable. When the coefficient bitwidth is a multiple of the bits per device this means that the range of the memristor is fully taken advantage of. However, when there is a mismatch the LSB mapping only utilize the full range of the first few memristors while the last memristor will not be fully programmed with the weight.

If each device stores 6 bits and the coefficient bitwidth is 8 bits, then the first memristor will be programmed with the 6 LSB while the 2 MSB are put in a second memristor. This means that the conductance levels in the second device are always relatively low. But the conductance and read variations are still in the same magnitude. Which leads to the second device being more susceptible to the device non-idealities.

2.3.5 Bit-Serial Input

When interfacing the crossbar arrays with the digital domain, DACs are required to convert the digital signals into analog voltages for the crossbar[1]. Multi-bit DACs can be used to generate precise analog voltage levels but are costly while introducing noise in the conversion. To solve this, the bit-serial scheme is often used. With this technique, the multi-bit digital input is time-multiplexed by processing a single bit at a time using simple 1-bit DACs. This drastically reduces noise sensitivity, as the crossbar only needs to differentiate between simple high

and low voltages instead of fine-grained analog levels.

In order to obtain the full output, the partial results are added together after being shifted according to the bit significance. While this requires extra digital components like shift registers and adders, it eliminates the need for massive, high-precision multi-bit DACs. And also ensures low ADC resolution which in turn could lower the total cost of peripherals. In summation, bit-serial input leads to higher latency in exchange for achieving higher accuracy, better noise immunity, and reducing the total peripheral cost.

2.4 Analog to Digital Conversion

The main computation in the IMC design is performed in the analog domain. In order to finalize the output in the digital domain analog to digital conversion (ADC) is needed. The number of bits supported by the ADC component should be chosen carefully in order to balance accuracy in the conversion with the power consumption that it adds to the system.

ADCs are placed at the output of the crossbar arrays. Different schemes can be selected to optimize the number of ADC devices per crossbar array. The naive solution is to place one ADC at each bitline. But due to the size of an ADC compared to the memristor, this might prove impractical [30]. To optimize the floorplanning of the IMC accelerator, it might be more feasible to share one ADC between several bitlines.

The conversion to the digital domain takes place before the digital shift-add operation that constructs one element in the output vector. A general approach to the ADC resolution is given by the number of rows in the crossbar, the bitwidth of the input and the bitwidth of the coefficients in the crossbar as proposed by [31].

$$B_{\text{adc}} = \begin{cases} B_{coeff} + B_{\text{in}} + \log_2 M, & \text{if } B_{coeff} > 1,\ B_{\text{in}} > 1 \\ B_{coeff} + B_{\text{in}} + \log_2 M - 1, & \text{otherwise} \end{cases} \tag{2.12}$$

Where B_{adc} is the ADC resolution, M is the number of crossbar rows, B_{in} is the bitwidth of the input and B_{coeff} is the bitwidth of the coefficients. This equation states the minimal ADC resolution to ensure no clipping. Quantization noise still occurs despite choosing an optimal sized ADC resolution.

[30] found that Flash-ADC and successive-approximation ADC (SAR-ADC) have suitable properties for IMC.

2.5 System Parameters

The system level performance and accuracy in a crossbar array are affected by a number of parameters and how they are combined.

- *Number of rows (word lines)*: The number of rows in a crossbar corresponds to the number of inputs. More rows causes higher IR drop which in turn can distort the accuracy of the computation. The number of rows is used in calculating the ADC resolution.

- *Number of columns (bitlines)*: The number of columns in a crossbar array corresponds to the size of the output. More columns also leads to higher IR drop. The number of bitlines is proportional to the number of ADCs in the system.
- *Bits per device*: The number of bits per device is the limiting factor for how many states that can be represented by a memristor and is a device intrinsic parameter. Bits per device is used in calculating the ADC resolution.
- *Input bitwidth*: The number of input bits that can be processed simultaneously. Input bitwidth is used in calculating the ADC resolution.
- *Coefficient bitwidth*: The coefficient bitwidth is used for quantization of the weights. If the desired coefficient bitwidth is larger than the bits per device, coefficient slicing can be used to still represent the weight using the desired device. Larger coefficient bitwidth means smaller quantization error.
- *Input bitwidth*: The number of bits per input is used for quantization of the input. If the desired input bitwidth is not supported, bit-serial input can be used to still utilize the full input bitwidth. Larger input bitwidth means smaller quantization error.
- *ADC resolution*: The number of bits that the analog to digital conversion supports. Increasing the ADC resolution generates larger energy consumption. The ADC resolution is derived from the number of word lines in the crossbar, the input bitwidth and bits per device. A too small ADC resolution risks clipping large input values, while too large ADC can lead to redundant power consumption.

Chapter 3
Investigation of IMC Simulators

In order to perform our measurements, the design needs to be implemented in a simulator. Finding a simulator which supports measuring accuracy, power and performance proved challenging. Memristor crossbar arrays are an emerging technology in the field of neuromorphic computing. As such a majority of the simulators available to simulate complex memristor crossbar array designs are created for machine learning applications.

There exist different classes of simulators. There are the Python interfaced simulators that integrate common Python ML libraries with some sort of memristor hardware calculation. Then there are the SPICE-based analog circuit simulators that create linear equations for each device and solve to find the behavior of the circuit.

In this section we evaluate simulators that we have considered and the features and limitations of each of them. We will compare them and present the reasoning for our selection.

3.1 CiMLoop

The CiMLoop simulator is a flexible, accurate, and fast compute-in-memory modeling tool, developed at MIT by Tanner Andrulis, Vivienne Sze, and Joel S. Emer [32]. Its main purpose is to allow researchers to easily and rapidly compare the PPA (Power, Performance and Area) metrics of different architectural designs using memristors and crossbar arrays. CiMLoop is based on TimeLoop and Accelergy, extending their mapping and estimation engines to support analog components and compute-in-memory. Furthermore, to accurately model circuit-level components such as ADCs, memory cells, and row/column drivers, CiMLoop utilizes a built-in NeuroSim plug-in to extract realistic energy and area values.

3.1.1 PPA Calculations in CiMLoop

One of the ways that CiMLoop makes it possible to rapidly deploy and evaluate different designs is through its statistical energy model. Instead of relying on slow, cycle-by-cycle simulations of every specific data value to determine energy consumption, CiMLoop calculates the average energy per action based on the probability distribution of the data values. This allows CiMLoop to evaluate

designs orders of magnitude faster than a traditional cycle-by-cycle approach, while still maintaining high circuit-level accuracy.

3.2 CrossSim

CrossSim [33] is a GPU-accelerated and Python based simulator developed by Sandia Labs for simulating crossbar arrays and model analog in-memory computing. The intended use of CrossSim is to simulate Neural Networks for machine learning on crossbar array and to determine the accuracy of the calculations. CrossSim supports multiple design choices that will impact accuracy, such as if the input is bit sliced or not.

The accuracy is simulated with different parameters and properties of memristors. One of those is a parasitic voltage that may occur and that will impact performance. The calculations are not done in IMC, however they are altered as if they were calculated in a crossbar. This is why CrossSim is able to calculate the accuracy of a crossbar array.

CrossSim is not intended for doing PPA calculations for the crossbar. The only intended usage is accuracy. This is due to the simulator being developed for machine learning, as such it is focused on accuracy. However, Sandia labs suggests that if you wanted PPA calculations, then you should use their other simulator, Xyce, for such calculations, in addition to using CrossSim.

3.3 AiHWKit

AiHWKit is a crossbar simulator developed by IBM. It is also developed with the same purpose as Sandia's CrossSim, primarily focused on simulating analog crossbar arrays for neural network training and inference. While it offers extensive functionality for behavioral simulation, it does not support measurements of PPA. Consequently, both AiHWKit and CrossSim were ruled out as they lacked the necessary features for our specific goals.

3.4 Xyce

The Xyce simulator, developed by Sandia National Laboratories, is an open source SPICE based simulator. In the most basic form, the simulator takes a netlist which is a list of devices and their properties and interconnects [34]. It creates a system of differential equations to represent the circuit and solves this to produce simulation results.

Although Xyce lacks native support for crossbar arrays, it provides all the necessary components to simulate them, including memristor devices [34]. The outputs from the simulations are the node currents and voltages in the shape of DC and transient analyses. The simulation granularity of the Xyce simulator is quite high. The current and voltage for each circuit element can be calculated for every time interval.

In conclusion, Xyce is suitable for detailed analysis of circuits. But in order to do system-level evaluation of different designs a lot of effort is needed to build the entire system for each design from the ground up.

3.5 DNN+NeuroSim V1.5

The DNN+NeuroSim V1.5 (NeuroSim) is an open source framework for benchmarking IMC accelerators for deep neural networks [35][36][37]. The simulator is suited for evaluating design space explorations of neuro-inspired AIMC designs. The framework can provide inference accuracy and PPA estimations for AIMC, as a result of design choices on both circuit and device level.

NeuroSim consists of two parts; a behavioral simulator (BS) and a hardware analyzer (HA). The BS is a Python wrapper that enables inference with PyTorch [35]. It also leverages TensorRT for post-training quantization [35]. The BS creates traces that are used as input to the HA to estimate PPA. The HA is a circuit level model for benchmarking neuro-inspired designs. It is written in C++ and allows for hierarchical organization from the device level to the circuit level [38]. Taking the network topology and traces from the BS as input, the HA uses a novel floorplanning and mapping algorithm to create a chip design [38]. The HA provides PPA metrics for this design hierarchically.

3.6 Badcrossbar

Badcrossbar is an open-source tool for computing currents and voltages in non-ideal crossbar arrays to find the effect of IR drop on the crossbar [39]. The tool is developed as a Python package which makes it easily interfaced with other Python frameworks. The tool takes the input voltages, the crossbar resistances and the wire resistance. Using these parameters the non-ideal output is calculated considering IR drop. The tool is limited to IR drop analysis and does not consider other device non-idealities. While this limits the suitability of using Badcrossbar as independent simulator, it is still useful for combining with other Python based simulators that do not consider IR drop.

It is important to note that Badcrossbar only models static IR drop and does not consider the switching activity of the design. Another drawback of Badcrossbar is the scalability of input. As the size of crossbar arrays grow, the computation time and resources significantly increase. This presents a challenge for simulating large arrays where IR drop analysis is most prudent.

3.7 Summary of Investigations

As summarized in Table 3.1, finding a single simulator that fulfills all thesis requirements proved challenging. Tools like CrossSim and AiHWKit provide robust accuracy simulations but completely lack hardware-level PPA metrics, even if modified. Conversely, Xyce offers detailed SPICE-level analysis but lacks built-in crossbar support, making system-level PPA evaluation highly impractical.

Table 3.1: Comparison of evaluated IMC simulators based on key features required for DFT evaluation.

Simulator	Simulation Basis	Accuracy Support	PPA Support	Native Crossbar Support
CiMLoop	Statistical	No	Yes	Yes
CrossSim	GPU/Python	Yes	No	Yes
AiHWKit	Python	Yes	No	Yes
Xyce	SPICE	No	Yes (Impractical)	No
NeuroSim	SPICE/Circuit-level	Yes	Yes	Yes
Badcrossbar	Python	Yes (IR drop only)	No	Yes

In this evaluation, "Accuracy Support" means that the simulator can extract accuracy metrics entirely in software, either through built-in features or by mathematically modeling the hardware's behavior.

Both CiMLoop and NeuroSim emerged as comprehensive tools capable of handling PPA and evaluations for crossbar architectures. While CiMLoop accelerates simulations using statistical models, NeuroSim offers deterministic, cycle-by-cycle circuit-level simulations. However, because NeuroSim also supported accuracy, it became the better choice.

Therefore, NeuroSim was retained as the primary simulator. To address its limitations regarding static IR drop analysis, it was supplemented with the Badcrossbar tool, creating a robust environment for evaluating the proposed designs. To the best of our knowledge, we did not identify any other lightweight, open-source IR drop simulation tools outside of SPICE-based frameworks. Although non-SPICE approaches may exist, they appear to be either proprietary or not publicly available. So while Badcrossbar lacks dynamic IR drop modeling and is time and memory intensive for large transform sizes it will still contribute another layer to the investigation and highlight the effect of static IR drop on large crossbar arrays.

Finally, since these tools primarily target neural network inference, a custom framework was implemented on top of them to calculate the required algorithmic accuracy (MSE) for the DFT. Detailed modifications will be discussed in the methodology section.

3.8 In-Depth Analysis of NeuroSim

3.8.1 IMC Simulation (Behavioral Simulation)

NeuroSim leverages a Python wrapper to enable the popular Python machine learning library PyTorch for inference. NeuroSim supports pooling, linear and convolution layers by extending the existing PyTorch implementations of these layers with hardware simulation of the multiply-accumulate operation [35] with quantization performed with TensorRT.

The inputs are normalized to be nonzero and padded with dummy rows and columns. In increments configurable from one to several bits at a time the inputs are multiplied with one or several bits of the weights. And finally ADC sensing is emulated. The outputs for each column are shift-added to compute the entire result of the VMM.

NeuroSim uses this behavioral simulation in its own implementation of the different layers listed above. Finally NeuroSim will present the inference accuracy of the model.

Simulate Non-Idealities

In the BS of NeuroSim, there are two mutually exclusive ways to add hardware non-idealities and simulate how accuracy is affected: device expert mode and circuit expert mode [40]. Circuit expert mode is used when data of the aggregated non-idealities from detailed simulations or measurements from a fabricated IMC macro is available.

Device expert mode is suitable when only the device characteristics are available [40]. The non-idealities supported by device expert mode includes SAF, conductance drift, read noise and device-to-device variation, non-linear states and standard deviation of each state.

3.8.2 Memristor Devices in NeuroSim

The device states can be set with a file specifying the conductance levels of the simulator. The simulator supports a number of devices including SRAM, FeFET and memristors (eNVM, emerging non-volatile memory). However even when using analog devices, the input bitwidth is always 1-bit. When using more bits to represent the input, NeuroSim will bit-slice the input. Multiple bit precision is supported for the weights.

3.8.3 Floor Planning in NeuroSim

In the hardware analyzer core of NeuroSim [40], the hardware design is constructed using a single chip with hierarchies of components. This hierarchy is divided up by a chip that consists of pooling units, accumulation units, activation units, a global buffer and tiles. The tiles in turn consist of a tile buffer, an accumulation unit, an output buffer and processing units. The processing units are made up of a buffer, an accumulation unit, an output buffer and synaptic arrays. These synaptic arrays then have switch matrices, muxes, ADCs, Shift-and-Add components, and the crossbar consisting of memristors. The complete overview of the chip design can be seen in figure 3.1.

The synaptic array, or SubArray as it is called in NeuroSim, is the primary module of interest for our analysis. This is because it contains the actual memristive crossbar array, where the DFT twiddle factors are mapped as conductances. Furthermore, the SubArray houses all the necessary peripheral circuitry, such as wordline drivers, ADCs, and Shift-and-Add circuits, all required to perform the VMM in the analog domain.

The measurements of computing a full DFT, is done entirely within this module. Therefore, analyzing the SubArray provides the exact PPA metrics for our In-Memory Computing implementations. The diagram for the SubArray module can be seen in Figure 3.2.

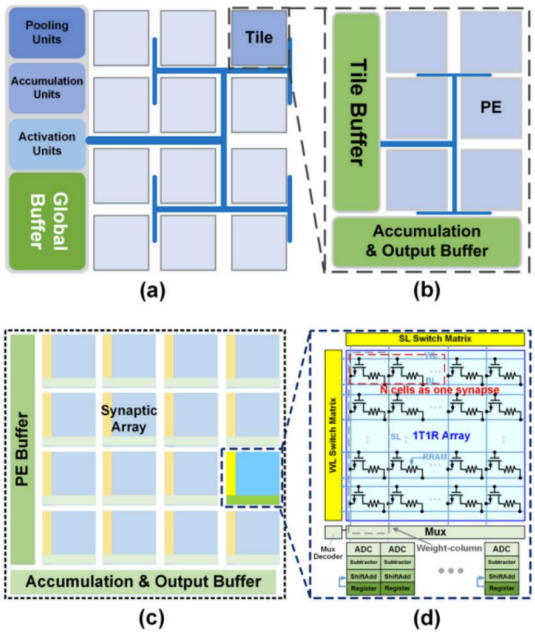


Figure 3.1: Diagram of NeuroSim's chip design [40]. (c) represents one processing unit (PE).

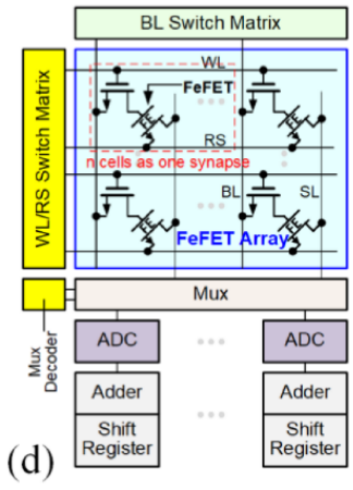


Figure 3.2: Diagram of the SubArray module in NeuroSim [40].

3.8.4 PPA in NeuroSim

Estimating the PPA in NeuroSim, is done by the HA. The HA calculates these metrics individually for each hardware component. Furthermore, the total cost is determined by a bottom-up summation. This starts by accumulating the components within a single SubArray, followed by the SubArrays within a tile, and finally summing all elements to acquire the total chip cost. Calculating the energy for a crossbar array, is done with the standard dynamic energy equation:

$$E_{dyn} \propto C \cdot V^2 \tag{3.1}$$

Writing the dynamic energy equation with a proportional sign, is done to show the standard scaling relationship, where the energy scales linearly with capacitance and quadratically with voltage. This is because we are not using exact constants in this general model. In simulation environments like NeuroSim [35], this is modeled explicitly by estimating the capacitance of the wordlines (C_{WL}) and bitlines (C_{BL}) to compute the total array energy:

$$E_{array} \approx C_{WL} \cdot V_{dd}^2 \cdot N_{WL} + C_{BL} \cdot V_{read}^2 \cdot N_{BL} \tag{3.2}$$

where N_{WL} and N_{BL} represent the number of selected wordlines and bitlines, respectively.

For the performance metric, the critical path of a crossbar array is defined by the analog read operation [35]. Calculating the total delay ($t_{critical}$) is done by summing the distinct stages in this path:

$$t_{critical} = \max(t_{WL}, t_{Mux}) + t_{precharge} + t_{ADC} \tag{3.3}$$

It must be noted that this latency estimation, along with the energy calculations for the wordlines and bitlines, forms the core of our PPA evaluation when scaling the DFT designs. Furthermore, non-idealities such as bitline delay and the energy cost of resistive losses are not included in NeuroSim 1.5. However, it is possible to integrate models for these, which we have done in this thesis.

3.8.5 Bit-Serial Input in NeuroSim

In the Hardware Analyzer core of NeuroSim, the input bitwidth is strictly limited to 1 bit per cycle. When higher input bitwidth is desired, the simulator automatically bit-slices the input and processes it sequentially. This is done to get rid of Digital-to-Analog Converters (DACs) at the crossbar inputs, as the nonlinearity in I-V curve of eNVMs will introduce distortion in parallel read-out [40]. This architectural choice currently has the biggest impact on the simulated latency and throughput.

3.8.6 NeuroSim's usage of Digital Values

One of the design choices in NeuroSim[35], is the usage of digital values between crossbars. In the intended usage of NeuroSim, that follows a traditional floorplan, then multiple crossbars are connected in the digital domain. However, as we are

doing the DFT algorithm, we have no need for this behavior. Furthermore, it makes us unable to implement certain designs, where for example different crossbars are connected to each other in the analog domain, such as Mojtaba's FFT implementation [1].

3.8.7 ADC

There are two available ADCs to choose from in NeuroSim: a multilevel sense amplifier or a SAR ADC. The SAR ADC is modeled in NeuroSim by calculating the energy per column. It takes the technology-dependent column power (P_{col}) and multiplies it by the time it takes to complete the read. The energy per column is modeled as:

$$E_{col} = P_{col} \times (\log_2(levelOutput) + 1) \times 10^{-9} \tag{3.4}$$

where levelOutput is the number of quantization levels. Because the term $\log_2(levelOutput)$ corresponds to the ADC resolution in bits, the conversion takes one cycle per bit plus one additional cycle for setup. The column power, P_{col}, is an empirically derived value based on the chosen technology node, device roadmap, and the equivalent column resistance. While the energy equation evaluated per column might appear linear at first glance, the total energy actually scales quadratically with the ADC resolution. In the NeuroSim model, the column power (P_{col}) itself is defined as a linear function of the ADC resolution.

The latency for the ADCs reflects this cycle count as well. Since NeuroSim models each cycle as taking 1 ns (10^{-9} seconds) for the SAR ADC, the total latency for the ADC across a given number of reads (numRead) is modeled as:

$$t_{ADC} = (\log_2(levelOutput) + 1) \times 10^{-9} \times numRead \tag{3.5}$$

Chapter 4
Methodology

To improve the energy efficiency and accuracy of IMC DFT we propose three designs that utilize the properties of DFT or the crossbar array to optimize some aspect of the hardware. In order to evaluate the proposed designs, simulations are run with the NeuroSim simulator described in Section 3.5. Power/energy, latency and accuracy as a result of different design choices are modeled and measured.

4.1 IMC Designs

Three designs for how to compute DFT with crossbar arrays are presented. The Baseline design is a naive implementation of the DFT that uses differential representation for both inputs and weights. The Merged design stacks the weight matrices into one to minimize the use of costly ADC operations. The Symmetry design further optimizes the Merged design by leveraging the symmetry of real DFT to half the number of columns needed for the DFT operation. Additionally, the device, peripherals and system parameters and non-idealities used in the simulations are outlined.

4.1.1 Baseline Design

The baseline DFT design presented in [6] is a naive implementation utilizing the differential representation detailed in Section 2.3.1. The input vector is represented in the differential fashion and is the input to each of the differential crossbar arrays. Considering only real valued input (2.6) can be expressed as $\boldsymbol{X} = \boldsymbol{x}\,\boldsymbol{W}_N$. From this, and (2.11), the DFT can be rewritten as

$$\boldsymbol{X} = \boldsymbol{x}_+\,\boldsymbol{W}_{N+} + \boldsymbol{x}_-\,\boldsymbol{W}_{N-} - (\boldsymbol{x}_+\,\boldsymbol{W}_{N-} + \boldsymbol{x}_-\,\boldsymbol{W}_{N+}) \tag{4.1}$$

where each of the $\boldsymbol{W}_N$ matrices is one crossbar array of dimensions $N \times 2N$ with the real and imaginary parts of the DFT matrix and the inputs $\boldsymbol{x}_+$ and $\boldsymbol{x}_-$ is a row vector of size $1 \times N$.

$$\boldsymbol{W}_{N\pm} = \underbrace{\begin{bmatrix}\Re\{\boldsymbol{W}_{N\pm}\} & \Im\{\boldsymbol{W}_{N\pm}\}\end{bmatrix}}_{N \times 2N} \tag{4.2}$$

For simplicity, assuming that the transform size is constant, $\Re\{\boldsymbol{W}_{N\pm}\}$ will be expressed as $\boldsymbol{W}^{\pm}_{\mathrm{Re}}$ and $\Im\{\boldsymbol{W}_{N\pm}\}$ will be expressed as $\boldsymbol{W}^{\pm}_{\mathrm{Im}}$. The outputs from each crossbar are added together to form the output vectors $\boldsymbol{X}_{Re}$ and $\boldsymbol{X}_{Im}$. Thus for the real input case ($\boldsymbol{x}_{Re}$), the real and complex part of the DFT is reconstructed by:

$$\begin{aligned}\boldsymbol{X}_{Re} &= (\boldsymbol{W}^{+}_{Re}\boldsymbol{x}^{+}_{Re} + \boldsymbol{W}^{-}_{Re}\boldsymbol{x}^{-}_{Re}) - (\boldsymbol{W}^{+}_{Re}\boldsymbol{x}^{-}_{Re} + \boldsymbol{W}^{-}_{Re}\boldsymbol{x}^{+}_{Re}) \\ \boldsymbol{X}_{Im} &= (\boldsymbol{W}^{+}_{Im}\boldsymbol{x}^{+}_{Re} + \boldsymbol{W}^{-}_{Im}\boldsymbol{x}^{-}_{Re}) - (\boldsymbol{W}^{+}_{Im}\boldsymbol{x}^{-}_{Re} + \boldsymbol{W}^{-}_{Im}\boldsymbol{x}^{+}_{Re})\end{aligned} \tag{4.3}$$

For complex input ($\boldsymbol{x}_{Re}$ & $\boldsymbol{x}_{Im}$), the design is extended with duplicated hardware to process the imaginary input vector in the same fashion as the real input. The real and imaginary output vectors are added digitally to form the complete complex output.

$$\boldsymbol{X} = \underbrace{(\boldsymbol{X}_{ReRe} - \boldsymbol{X}_{ImIm})}_{\text{Real Part}} + j\underbrace{(\boldsymbol{X}_{ReIm} + \boldsymbol{X}_{ImRe})}_{\text{Imaginary Part}} \tag{4.4}$$

This operation can be generalized for any combination of real and imaginary components as:

$$\boldsymbol{X}_{AB} = (\boldsymbol{W}^{+}_{A}\boldsymbol{x}^{+}_{B} + \boldsymbol{W}^{-}_{A}\boldsymbol{x}^{-}_{B}) - (\boldsymbol{W}^{+}_{A}\boldsymbol{x}^{-}_{B} + \boldsymbol{W}^{-}_{A}\boldsymbol{x}^{+}_{B}) \quad \text{for } A, B \in \{\mathrm{Re}, \mathrm{Im}\} \tag{4.5}$$

Because of the differential representation, this physical hardware mapping perfectly reconstructs the mathematical product $(\boldsymbol{W}^{+}_{A} - \boldsymbol{W}^{-}_{A})(\boldsymbol{x}^{+}_{B} - \boldsymbol{x}^{-}_{B})$. One of the crossbars used in the baseline design is shown in Figure 4.1. As stated earlier, either four or eight crossbars will be used in total, for the real and complex case respectively.

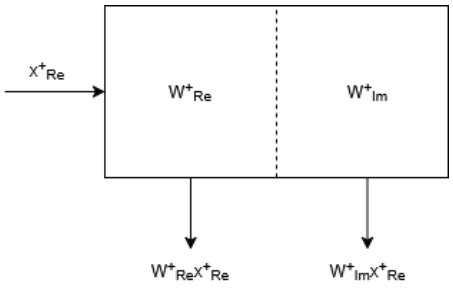


Figure 4.1: Diagram of a baseline crossbar.

4.1.2 Merged Design

Computing a DFT with multiple crossbar arrays, as is the case for the Baseline design outlined in 4.1.1, will lead to a costly use of peripherals. In the paper by Zhao et al. [7], a more optimized version of a DFT is proposed by stacking the arrays, reducing the hardware cost of the peripherals by half, like ADCs. We have taken inspiration from this design and have concentrated the DFT even further,

to one single crossbar. This approach reduces the hardware cost of the peripherals to one fourth of the naive implementation.

To clarify this architecture, we will first detail the matrix configuration for real-valued inputs, and then extend the design for the complex case. Thus, given a real-valued differential input vector $\boldsymbol{x} = \boldsymbol{x}^+ - \boldsymbol{x}^-$, the full system equation to obtain the differential Real and Imaginary outputs is:

$$\underbrace{\begin{bmatrix} \boldsymbol{X}^+_{\mathrm{Re}} \\ \boldsymbol{X}^-_{\mathrm{Re}} \\ \boldsymbol{X}^+_{\mathrm{Im}} \\ \boldsymbol{X}^-_{\mathrm{Im}} \end{bmatrix}^T}_{\text{Complex Output}(1\times 4N)} = \underbrace{\begin{bmatrix} \boldsymbol{x}^+_{\mathrm{Re}} & \boldsymbol{x}^-_{\mathrm{Re}} \end{bmatrix}}_{\substack{\text{Real Input} \\ (1\times 2N)}} \cdot \underbrace{\begin{bmatrix} \boldsymbol{W}^+_{\mathrm{Re}} & \boldsymbol{W}^-_{\mathrm{Re}} & \boldsymbol{W}^+_{\mathrm{Im}} & \boldsymbol{W}^-_{\mathrm{Im}} \\ \boldsymbol{W}^-_{\mathrm{Re}} & \boldsymbol{W}^+_{\mathrm{Re}} & \boldsymbol{W}^-_{\mathrm{Im}} & \boldsymbol{W}^+_{\mathrm{Im}} \end{bmatrix}}_{\substack{\text{Merged Differential Matrix} \\ (2N\times 4N)}} \tag{4.6}$$

For complex input we can extend the merged matrix to combine the calculations of the real and imaginary inputs. Essentially the merged matrix is extended into a $4N \times 4N$ matrix. This computes the full output for the DFT, with differential representation. The full output can be fully reconstructed either in the analog or digital domain. This is done with (4.4).

$$\underbrace{\begin{bmatrix} \boldsymbol{X}^+_{\mathrm{Re}} \\ \boldsymbol{X}^-_{\mathrm{Re}} \\ \boldsymbol{X}^+_{\mathrm{Im}} \\ \boldsymbol{X}^-_{\mathrm{Im}} \end{bmatrix}^T}_{\substack{\text{Full Differential Output} \\ (1\times 4N)}} = \underbrace{\begin{bmatrix} \boldsymbol{x}^+_{\mathrm{Re}} & \boldsymbol{x}^-_{\mathrm{Re}} & \boldsymbol{x}^+_{\mathrm{Im}} & \boldsymbol{x}^-_{\mathrm{Im}} \end{bmatrix}}_{\substack{\text{Complex Input} \\ (1\times 4N)}} \cdot \underbrace{\begin{bmatrix} \boldsymbol{W}^+_{\mathrm{Re}} & \boldsymbol{W}^-_{\mathrm{Re}} & \boldsymbol{W}^+_{\mathrm{Im}} & \boldsymbol{W}^-_{\mathrm{Im}} \\ \boldsymbol{W}^-_{\mathrm{Re}} & \boldsymbol{W}^+_{\mathrm{Re}} & \boldsymbol{W}^-_{\mathrm{Im}} & \boldsymbol{W}^+_{\mathrm{Im}} \\ \boldsymbol{W}^-_{\mathrm{Im}} & \boldsymbol{W}^+_{\mathrm{Im}} & \boldsymbol{W}^+_{\mathrm{Re}} & \boldsymbol{W}^-_{\mathrm{Re}} \\ \boldsymbol{W}^+_{\mathrm{Im}} & \boldsymbol{W}^-_{\mathrm{Im}} & \boldsymbol{W}^-_{\mathrm{Re}} & \boldsymbol{W}^+_{\mathrm{Re}} \end{bmatrix}}_{\substack{\text{Merged Differential Matrix for Complex Case} \\ (4N\times 4N)}} \tag{4.7}$$

In Figure 4.2, the diagram of the Merged design is shown, as to make it clearer how the crossbars have been concatenated. To simplify, differential representation is not shown in the diagram.

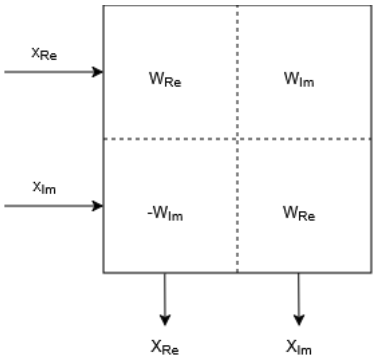


Figure 4.2: Diagram of the merged design

4.1.3 Symmetric Design

To further optimize the DFT for hardware, we have utilized the conjugate Symmetry of the DFT [41]. This states that for real valued inputs for a DFT, you can compute the full DFT, with only the first $N/2+1$ elements. This is because $\boldsymbol{X}[K] = \boldsymbol{X}^*[N-K]$. However, as $\boldsymbol{X}[0]$ and $\boldsymbol{X}[N/2]$ is unique, we need to compute them, along with $\boldsymbol{X}[1]$ to $\boldsymbol{X}[N/2-1]$. Furthermore, as $\boldsymbol{X}[0]$ and $\boldsymbol{X}[N/2]$ are purely real values, i.e. they are real numbers, thus we can reduce the number of columns by 4 for the crossbar array.

With this optimization, we can reduce the hardware by half, as we decrease the columns from $4N$ to $2N$. Thus reducing the costly ADCs that otherwise would be needed to convert the analog signals to digital. The equation would be the same as (4.6), with the difference that the Merged Matrix will have the size $2N \times 2N$.

$$\underbrace{\begin{bmatrix} \boldsymbol{X}[0:N/2]^{+}_{\mathrm{Re}} \\ \boldsymbol{X}[0:N/2]^{-}_{\mathrm{Re}} \\ \boldsymbol{X}[1:N/2-1]^{+}_{\mathrm{Im}} \\ \boldsymbol{X}[1:N/2-1]^{-}_{\mathrm{Im}} \end{bmatrix}^{T}}_{\text{Unique Output Values}(1\times 2N)} = \underbrace{\begin{bmatrix} \boldsymbol{x}^{+}_{\mathrm{Re}} & \boldsymbol{x}^{-}_{\mathrm{Re}} \end{bmatrix}}_{\substack{\text{Real Input} \\ (1\times 2N)}} \cdot \underbrace{\begin{bmatrix} \boldsymbol{W}^{*+}_{\mathrm{Re}} & \boldsymbol{W}^{*-}_{\mathrm{Re}} & \boldsymbol{W}^{*+}_{\mathrm{Im}} & \boldsymbol{W}^{*-}_{\mathrm{Im}} \\ \boldsymbol{W}^{*-}_{\mathrm{Re}} & \boldsymbol{W}^{*+}_{\mathrm{Re}} & \boldsymbol{W}^{*-}_{\mathrm{Im}} & \boldsymbol{W}^{*+}_{\mathrm{Im}} \end{bmatrix}}_{\substack{\text{Merged Symmetry Matrix} \\ (2N\times 2N)}} \tag{4.8}$$

Where $\boldsymbol{W}^{*}_{\mathrm{Re}}$ is the real twiddle matrix of the first $N/2$ values, while $\boldsymbol{W}^{*}_{\mathrm{Im}}$ is the imaginary twiddle matrix for the indices 1 to N/2-1. Thus the size of $\boldsymbol{W}^{*}_{\mathrm{Re}}$ is $N \times N/2+1$ and $\boldsymbol{W}^{*}_{\mathrm{Im}}$ is $N \times N/2-1$. Which makes the total size of the merged matrix, $2N \times (2(N/2+1) + 2(N/2-1)) = 2N \times 2N$,

$$\boldsymbol{W}^{*}_{Re} = \begin{bmatrix} \cos(\frac{2\pi\cdot 0\cdot 0}{N}) & \cdots & \cos(\frac{2\pi\cdot 0\cdot k}{N}) & \cdots & \cos(\frac{2\pi\cdot 0\cdot N/2}{N}) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \cos(\frac{2\pi\cdot n\cdot 0}{N}) & \cdots & \cos(\frac{2\pi\cdot n\cdot k}{N}) & \cdots & \cos(\frac{2\pi\cdot n\cdot N/2}{N}) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \cos(\frac{2\pi(N-1)0}{N}) & \cdots & \cos(\frac{2\pi(N-1)k}{N}) & \cdots & \cos(\frac{2\pi(N-1)N/2}{N}) \end{bmatrix}_{N\times(N/2+1)} \tag{4.9}$$

$$\boldsymbol{W}^{*}_{Im} = \begin{bmatrix} \sin(\frac{2\pi\cdot 0\cdot 1}{N}) & \cdots & \sin(\frac{2\pi\cdot 0\cdot k}{N}) & \cdots & \sin(\frac{2\pi\cdot 0\cdot (N/2-1)}{N}) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \sin(\frac{2\pi\cdot n\cdot 1}{N}) & \cdots & \sin(\frac{2\pi\cdot n\cdot k}{N}) & \cdots & \sin(\frac{2\pi\cdot n\cdot (N/2-1)}{N}) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \sin(\frac{2\pi(N-1)1}{N}) & \cdots & \sin(\frac{2\pi(N-1)k}{N}) & \cdots & \sin(\frac{2\pi(N-1)(N/2-1)}{N}) \end{bmatrix}_{N\times(N/2-1)} \tag{4.10}$$

Reconstructing the Output for Symmetry

For a real-valued input vector $\boldsymbol{x}$ (where $\boldsymbol{x} \in \mathbb{R}^N$), the crossbar computes the partial output consisting of the real and imaginary parts for the first half of the

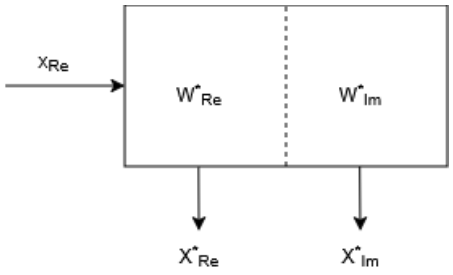


Figure 4.3: Diagram of the Symmetry crossbar

spectrum.

The full output vector $\boldsymbol{X}[k]$ (for $k = 0, \ldots, N-1$) is reconstructed as follows:

$$\boldsymbol{X}[k] = \begin{cases} \boldsymbol{X}_{Re}[k] & \text{for } k = 0 \\ \boldsymbol{X}_{Re}[k] + j \cdot \boldsymbol{X}_{Im}[k] & \text{for } 1 \leq k \leq \frac{N}{2} - 1 \\ \boldsymbol{X}_{Re}[k] & \text{for } k = \frac{N}{2} \\ \boldsymbol{X}_{Re}[N-k] - j \cdot \boldsymbol{X}_{Im}[N-k] & \text{for } \frac{N}{2} < k \leq N-1 \end{cases} \tag{4.11}$$

Complex-Valued Input

For a complex input $\boldsymbol{x} = \boldsymbol{a} + j\boldsymbol{b}$ (where $\boldsymbol{a}$ and $\boldsymbol{b}$ are real vectors), the linearity of the DFT is utilized. The computation is split into two passes (or two hardware instances) involving the real components $\boldsymbol{a}$ and $\boldsymbol{b}$. The complete complex output $\boldsymbol{X}_{total}$ is calculated by combining the transforms of the real and imaginary parts:

$$\boldsymbol{X}_{total} = \mathrm{DFT}(\boldsymbol{a}) + j \cdot \mathrm{DFT}(\boldsymbol{b})$$

Expanding this into real ($\Re$) and imaginary ($\Im$) components for the final digital addition:

$$\boldsymbol{X}_{total}[k] = \underbrace{(\Re\{\boldsymbol{A}[k]\} - \Im\{\boldsymbol{B}[k]\})}_{\text{Final Real Part}} + j \cdot \underbrace{(\Im\{\boldsymbol{A}[k]\} + \Re\{\boldsymbol{B}[k]\})}_{\text{Final Imaginary Part}} \tag{4.12}$$

where:

- $\boldsymbol{A}[k]$ is the full DFT of the input's real vector component $\boldsymbol{a}$, reconstructed using (4.11).
- $\boldsymbol{B}[k]$ is the full DFT of the input's imaginary vector component $\boldsymbol{b}$, reconstructed using (4.11).

Because of the conjugate properties of the imaginary part, for $k > N/2$:

$$\boldsymbol{X}_{total}[k] = \underbrace{(\Re\{\boldsymbol{A}[N-k]\} + \Im\{\boldsymbol{B}[N-k]\})}_{\text{Final Real Part}} + j \cdot \underbrace{(\Re\{\boldsymbol{B}[N-k]\} - \Im\{\boldsymbol{A}[N-k]\})}_{\text{Final Imaginary Part}} \tag{4.13}$$

For clarity, the first half of the spectrum ($0 \leq k \leq N/2$) can be written as:

$$\begin{aligned} \boldsymbol{X}_{Re}[k] &= \Re\{\boldsymbol{A}[k]\} - \Im\{\boldsymbol{B}[k]\} \\ \boldsymbol{X}_{Im}[k] &= \Im\{\boldsymbol{A}[k]\} + \Re\{\boldsymbol{B}[k]\} \end{aligned} \tag{4.14}$$

For the second half of the spectrum ($k > N/2$), utilizing index $m = N - k$:

$$\begin{aligned} \boldsymbol{X}_{Re}[k] &= \Re\{\boldsymbol{A}[m]\} + \Im\{\boldsymbol{B}[m]\} \\ \boldsymbol{X}_{Im}[k] &= \Re\{\boldsymbol{B}[m]\} - \Im\{\boldsymbol{A}[m]\} \end{aligned} \tag{4.15}$$

The full design of the Symmetry in the complex case, can be seen in Figure 4.4. As stated before, two crossbars are needed and because of the conjugate, the operators are switched when computing the second half of the spectrum, as seen in (4.15).

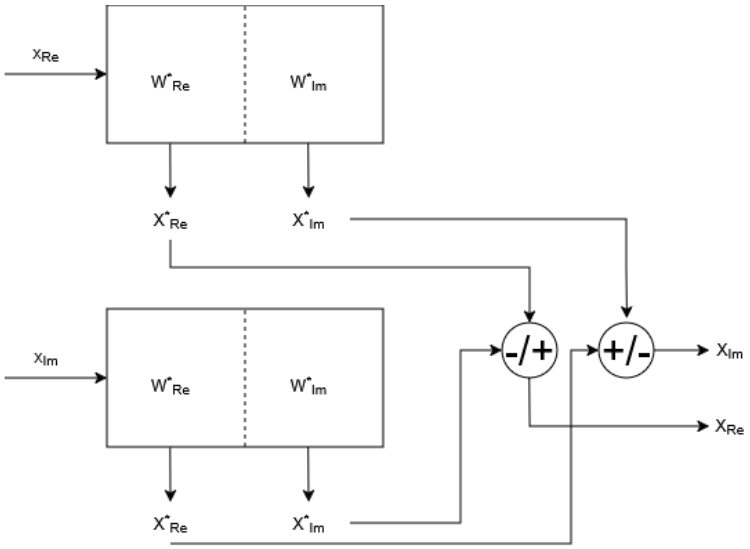


Figure 4.4: Diagram of the Symmetry design in the complex case

4.1.4 Design Comparison

The dimensions and area of the crossbars employed in the three designs are key to predict the system properties. Larger area increase the effect of IR drop and power consumption. The number of columns decide how many ADC components are required. To easily compare the dimensions and area of the designs they have been compiled in Table 4.1.

4.2 Tiling

For large crossbar arrays the IR drop might degrade the accuracy and render the system useless. To reduce this issue the crossbar can be divided into smaller crossbars called tiles. To enable large-scale DFT processing, two tiling schemes are investigated: square and rectangular.

Table 4.1: Crossbar dimensions and area comparison for real and complex inputs

Real Input			
Design	Crossbar Dimensions	Total Area	% of Baseline Area
Baseline	$4(N \times 2N)$	$8N^2$	-
Merged	$2N \times 4N$	$8N^2$	100
Symmetry	$2N \times 2N$	$4N^2$	50
Complex Input			
Design	Crossbar Dimensions	Total Area	% of Baseline Area
Baseline	$8(N \times 2N)$	$16N^2$	-
Merged	$4N \times 4N$	$16N^2$	100
Symmetry	$2(2N \times 2N)$	$8N^2$	50

4.2.1 Square Tiles

Square tiles are tiles where the number of devices in the word and bit lines (the height and width of the tile) are equal. T denotes the number of devices along one side in the square tile. In Table 4.2 we have outlined which tile sizes T we are investigating.

Table 4.2: Number of tiles per tile size T for real input Symmetry design $N = 1024$.

T	Number of tiles	Number of rows
64	1024	32
128	256	16
256	64	8
512	16	4
1024	4	2

4.2.2 Rectangular Tiles

Rectangular tiles are introduced to investigate if oblong tiles, which require fewer ADCs, are feasible. The idea is to minimize the number of ADCs by having fewer vertical tiles than the square tiling scheme and find out how wide the crossbars can be to enable good accuracy and while avoiding large IR drop.

The dimensions of rectangular tiles are written as T = (number of word lines)× (number of bitlines) = (tile height)× (tile width). For example T = 1024×512 for a large DFT ($N = 1024$) Symmetry design implementation without coefficient slicing would mean that there are 8 rectangular tiles, 4 in the top row and 4 in the bottom row. Figure 4.5 visualizes this example also considering two coefficient slicing scenarios (2 and 3 devices/weight).

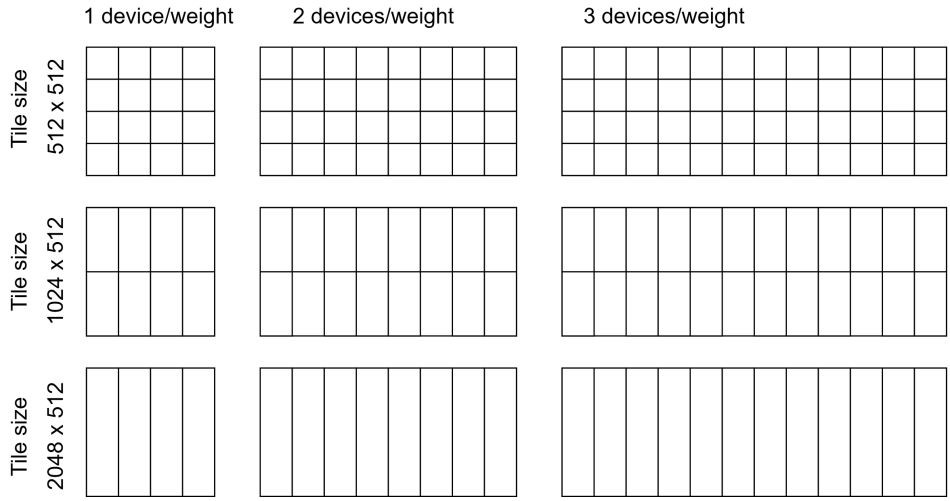


Figure 4.5: An example of how the square and rectangular tiling schemes are implemented for real input Symmetry design.

4.3 Device, System and Peripherals

4.3.1 FTJ Device

In order to simulate a memristor device in NeuroSim there are a number of device parameters that needs to be characterized. A FTJ memristor device is sufficiently characterized in [28][22]. In fact the paper authors used NeuroSim to investigate the feasibility of the FTJ device for in-memory computing in [28] with promising results thanks to good linearity, wide dynamic range, low conductance and little device-to-device variation and read noise.

Up to 60 distinct states have been achieved, with a dynamic range ($DR = \frac{G_{max}}{G_{min}}$) of up to 10 and $G_{max} = 1.2nS$ [22, 28]. The read noise is on average 3.5% and the device-to-device variation is an average of 0.8%. The device exhibit near-linear programming behavior, and the low conductance radically reduces the IR drop.

In NeuroSim the device was modelled by assuming linear state distribution. All the parameters used in the simulation are listed in Table 4.3. The parameters are taken from [28][22].

4.3.2 Bits Per Device

Since the ADC resolution is determined in part by the bits per device and the choice of device, these parameters will have a system level impact. As discussed in the next chapter and in the background, increasing the ADC resolution reduces the impact of noise but also leads to higher energy consumption. In the case that the coefficient bitwidth is larger than the bits per device, coefficient slicing will be employed which in turn increases the energy and area footprint. Fewer bits

Table 4.3: Device parameters for FTJ memristor [28][22]

Parameter	Value	Unit
Technology	20	nm
Read pulse width	5	ns
Bits per device	4, 6	bits
Read voltage	0.3	V
Dynamic range	10	
G_{max}	1.2	ns
Device-to-device variation	0.8	%
Read noise	3.5	%
Drift coefficient	2×10^{-5}	
Drift time	1	s

per device might also allow for better separability between the states and might decrease the effect of device-to-device variation on the accuracy.

So while the FTJ device supports up to 6 bits/device there might be benefits to not employ all of them. Fewer states per device could decrease the ADC resolution and might potentially decrease the energy consumption. However the need for coefficient slicing might emerge for devices that supports too few states. As such we will explore using the FTJ memristor with both 4 and 6 bits per device to highlight the effect of device bitwidth on the system.

4.3.3 ADC

In order to maintain good accuracy in the system while keeping the energy consumption and area low we want to find the minimal ADC resolution. Using (2.12), is a good approach for finding the minimal ADC resolution for general purpose application where the weights have unknown values. However since the exact values of the weights are known, the minimal ADC resolution can be defined by the specific values in the coefficients.

To decide the minimal ADC resolution, the sum at each column is calculated to find out what the largest value calculated at each column can be, assuming that the bit-serial inputs are all ones. This calculation corresponds to a vector multiplication between input of all ones and one column in the crossbar array. Note that the bits per device affects this calculation.

Due to the repeating nature of the weights of the three designs the minimal ADC resolution is the same for the designs despite having widely different number of word lines. For instance in (4.6) and (4.8) the differential pair of weight submatrices are stacked on top of each other. The largest sum along the columns will always be the first column in the real coefficient matrix, since it is all ones. While the number of word lines in the Merged and Symmetry designs is higher than in the Baseline design, the largest column-wise sum is still the same.

The formula for the minimal ADC resolution when no tiling is involved:

$$B_{\text{opt,ADC}} = \log_2 N + B_{device} \tag{4.16}$$

where B_{device} is the bits per device.

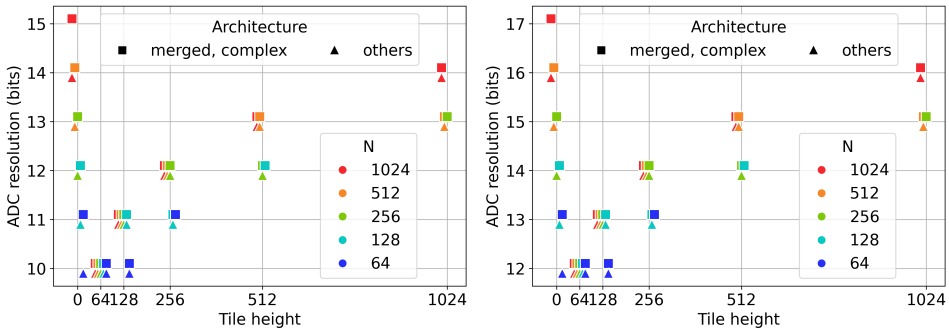
There is one exception however. For the Merged design and complex input the weight coefficients are repeated along the column axis. As a result one more bit is needed for the ADC resolution to assure no loss.

$$B_{\text{merged,complex,ADC}} = \log_2 N + B_{device} + 1 \tag{4.17}$$

Note that N is the size of the input (i.e. the same as in the context N-point DFT). Not to be confused with M in (2.12) which denoted the number of word lines in the crossbar array. When tiling, the ADC resolution is defined by the tile height T or N, depending on which is the smallest.

$$B_{\text{tile,ADC}} = \log_2(min(T, N)) + B_{device} \tag{4.18}$$

Where the tile height T is always less than or equal to N for the Symmetry design. Figure 4.6 illustrates the increased ADC resolution as a result of design, input, bits per device and tile size.



(a) 4-bit device **(b)** 6-bit device

Figure 4.6: ADC resolutions as defined in (4.16)–(4.18). A tile height of 0 indicates that no tiling is employed.

4.3.4 IR Drop

To assess the impact of IR drop in the system we use the same method as proposed in [28]. IR drop is a result of intrinsic wire resistance as well as the resistance of the memristors. The wire resistance for a FTJ memristor crossbar array is estimated to be 10Ω in [6]. The IR drop analysis in [28] show the voltage error as a function of the crossbar size for FTJ memristor crossbar arrays as calculated using the Badcrossbar simulator [39].

4.4 Accuracy Simulation

The accuracy simulation is performed using the behavioral simulator (BS) of NeuroSim along with Badcrossbar for IR drop simulation.

NeuroSim is the main tool for accuracy simulations and is used as the base. The primary objective of the BS in NeuroSim is to estimate the inference accuracy. For this thesis, however, we isolate the BS to find the accuracy of the designs presented in chapter 4.1. To support this effort, some changes are made to the BS to simplify the existing code and streamline the simulation process. The effect of IR drop is added to the existing BS in NeuroSim using Badcrossbar.

Extensions and Modifications to NeuroSim Behavioral Simulator

To support different representation modes and encapsulate all functionality to compare accuracy of a crossbar array, the `CrossbarArray` class is created. `CrossbarArray` inherits the NeuroSim `macro.IMC` and `_utils.QuantMixin` classes that contains the hardware simulation. A `CrossbarArray` object represents a crossbar array with user defined arguments and size N. The object quantizes its weights at initialization.

The `CrossbarArray` class is further extended by the classes `MergedCrossbarArray`, `BaselineCrossbarArray`, `SymmetryCrossbarArray`. Each of the children classes of `CrossbarArray` has their own implementation of the hardware simulation in the `simulate()` method. The `simulate()` method takes one (for real input) or two (for complex input) vectors, quantizes according to the user defined arguments and returns the de-quantized hardware simulated VMM result.

In order to support changing the parameters for simulation, the arguments of the NeuroSim simulator are extended. All the arguments that can be set in the BS remain with the additions listed in Table 4.4.

Parameter	Description
`N`	The input size
`arch`	Which design to simulate
`device`	Which memristor device to use
`input`	Specify real or complex input
`noise`	Simulate with or without noise
`ir_drop`	Simulate with or without IR drop
`error`	Error metric
`save_table`	Save simulation data to csv
`tile_height`	Tile height
`tile_width`	Tile width
`iterations`	How many random inputs to generate

Table 4.4: Simulation arguments

Furthermore, some changes are made to the hardware simulating functions defined in `macro.IMC`:

- Dummy columns are removed due to incompatibility with 1D input.
- Assume linear distribution between states rather than use the gap between first and second state when calculating `analog_step`.
- Ignore automatic quantization and mapping in favor for quantizing weights in the `CrossbarArray` class.
- Remove usage of TensorRT (a software development kit for optimizing and accelerating deep learning inference on NVIDIA GPUs) to make framework compatible with non-NVIDIA hardware.
- Update the coefficient slicing by putting devices in adjacent columns rather than separate crossbars
- Implement MSB coefficient slicing to minimize the impact of device non-idealities
- Enable tiling before crossbar VMM simulation

In order to also take the effects of static IR drop into account the Badcrossbar simulator, described in section 3.6, is used. In NeuroSim all the device non-idealities of device expert mode are injected by modifying the values of the matrix that represents the crossbar array. Each element in the matrix has a conductance value that is modified with the noise arguments, then the output of the crossbar array is computed by performing VMM with the input and non-ideal conductance matrix.

To expand the simulation to include IR drop, the non-ideal conductance matrix, the input voltages and wire resistance are used in a Badcrossbar simulation. The output of the simulation is the non-ideal currents as the result of the VMM considering IR drop. This output is passed along to the ADC simulation.

The default NeuroSim BS implementation of coefficient slicing effectively introduced separate crossbars to store the MSB and LSB devices. This is contradiction to the hardware analyzer and user manual [40] that states that the devices of a single weight are stored in adjacent columns. So in order to mimick this behavior the coefficient slicing of the BS was modified to centralize all devices into one crossbar with devices for one weight in adjacent columns.

Tiling is enabled for tile dimensions that evenly divide the crossbar. The tiling is placed right before the crossbar VMM operation so as to preserve all the peripheral simulation along with the crossbar. Digital output reconstruction is implemented for the tiles. To support flexible tiling investigation, the tile height and width are set with CLI arguments.

Note that IR drop simulation is very time consuming for larger transform sizes. For the Baseline design a single crossbar array has the dimensions $N \times 2N$. Simulating Symmetry and Merged designs and/or larger degree of coefficient slicing will also increase the simulation time due to larger crossbars.

Simulation Setup

Behavioral simulations are run in a separate Python program that creates `CrossbarArray` objects with the user specified parameters. The simulator generates 10 random

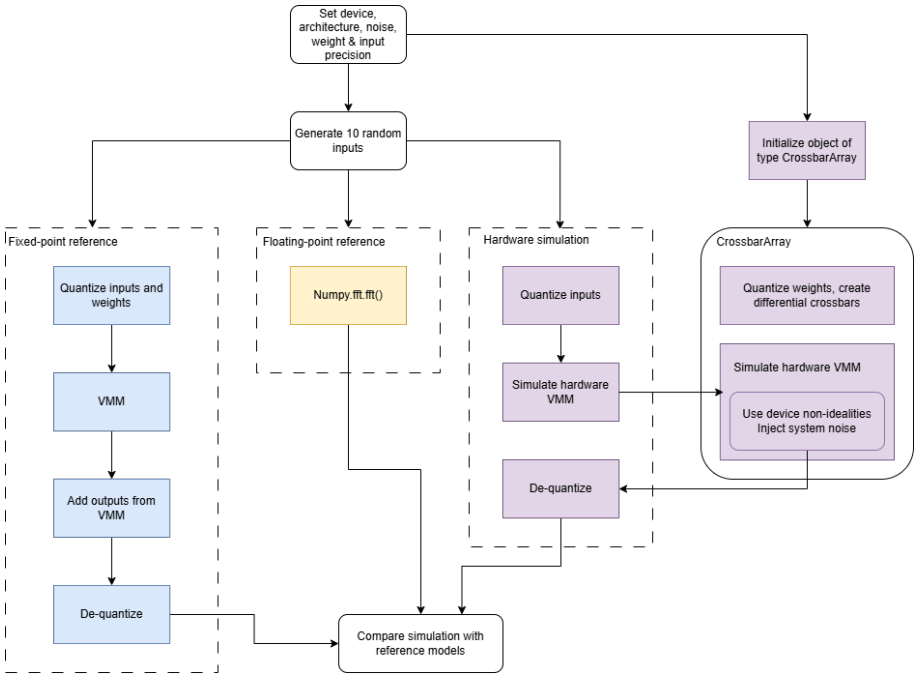


Figure 4.7: Schematic of the behavioral simulation

inputs of size N. Comparing the accuracy shows the average mean square error (MSE) for these 10 inputs to account for some of the randomness brought on by the irregular device non-idealities. The randomized inputs are floating points in the range -1 to 1.

The simulation program takes the arguments from Table 4.4 and sets the relevant parameters for creating the `CrossbarArray` according to the desired simulation setup. Additionally, the ADC resolution can be set manually or automatically (default) by setting the `adc_precision` argument to -1. The automatic ADC resolution is derived from (2.12).

Accuracy Evaluation

Two reference models are used, one fixed-point and one floating point, to compare with the simulated values. The floating point reference model takes the floating point random input as input and uses the `fft.fft()` function from the Numpy Python library to calculate the output. The fixed-point reference model uses the same arguments as for the IMC calculation to quantize inputs and weights.

The accuracy is compared three ways:

- *Quantization error*: The fixed-point reference model is compared to the floating-point reference model to show the quantization error without any hardware effects. In this case, the fixed-point reference model is considered the reference value.

- *Hardware-induced accuracy loss*: The hardware simulation result is compared to the fixed-point reference model to find only the hardware effects.
- *Total accuracy loss*: Finally the hardware simulation result is compared to the floating-point reference model to find only the hardware effects identify the full accuracy loss as a result of both hardware non-idealities and quantization.

The accuracy loss is expressed in terms of MSE defined as:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|^2 \tag{4.19}$$

Where $\boldsymbol{y}$ is the output of the reference model, and is the output of the simulation (or fixed-point reference model in the context of quantization error).

Bear in mind that since the $\boldsymbol{y}$ vectors are complex valued $|y_i - \hat{y}_i|$ represents the Euclidean distance between the two elements in the complex plane. The MSE therefore measures the average squared Euclidean distance between the simulated and reference outputs. Intuitively MSE quantifies how much on average the simulated values deviate element-wise from the reference model. In order to compare the MSE of different input sizes the MSE has been normalized with the average values in Table 4.5. Unless otherwise specified the MSE used in this report is the normalized mean square error.

Table 4.5: Average output values of real and complex DFTs for varying transform lengths N, computed over 100 random inputs.

N	Real Case (Avg)	Complex Case (Avg)
64	4.05	5.81
128	5.79	8.20
256	8.12	11.57
512	11.61	16.35
1024	16.39	23.16

Conductance Drift Investigation

The accuracy as a result of conductance drift is measured separately. The setup is almost identical to the rest of the measurements, but the drift time is increased from one second to one year. The simulations are run on 100 random inputs, rather than 10, to further decrease the effect of randomness in the result.

The default drift simulation mode in NeuroSim [40] is used. The mode assumes random drift. Additionally, each state is assumed to drift at the same speed.

IR Drop Investigation

IR drop degrades the accuracy of the DFT computation due to the device and wire resistances. As such the size of the crossbar employed will have an impact

on the system. For the same size input the crossbars for the different designs vary significantly. For instance the crossbar dimensions of the Merged design for complex input is $4N \times 4N$ while the Baseline design for complex input is $N \times 2N$. Although the total area is the same for the two designs, the Merged design is at risk to be more affected by IR drop.

Furthermore, by introducing coefficient slicing additional devices are placed in adjacent columns and will increase the size of the crossbar which exacerbate the voltage drop in the wires. To investigate whether the effects of IR drop are noticeable for small DFT in the different designs separate measurements are carried out. One configuration of parameters each for real and complex input, are simulated with and without IR drop to ascertain how the accuracy is affected by the larger crossbar array dimensions. Ten random input sequences are simulated.

Another set of simulations are carried out to compare the effect of IR drop on increased DFT size for the Symmetry design with real only input. The accuracy is compared for different coefficient bitwidths and bits per device to investigate the impact of coefficient slicing.

Tiling Investigation

To analyze the impact of tile size on IR drop, simulations of a large DFT ($N = 1024$) are performed for different tile sizes. Only one input set is simulated for each configuration due to the increased processing time when considering IR drop. Square tiles and rectangular tiles are described in Section 4.2. The rectangular tiles are simulated with tile heights 1024 and 2048, and tile widths 32, 64, 128, 256, 512.

4.4.1 Quantization and De-Quantization

Both weights and inputs are quantized in `CrossbarArray`, as well as the fixed-point reference model, based on the input arguments. For instance, if the `coefficient_bitwidth` parameter is 8, that means 8 bits are used to represent the weight value. This parameter is used to quantize the weights to the range $[0, 255]$, ($2^8 = 256$). The same procedure is applied to the inputs. Note that the input and coefficient bitwidths do not need to be the same.

After the VMM operation has been performed, the result is in the quantized domain. But in order to compare the accuracy of the simulation with the reference model, the output needs to be de-quantized. The scaling factor S is defined as:

$$S = 2^{B_{in}} \times 2^{B_{coeff}} \tag{4.20}$$

where B_{in} and B_{coeff} are the input and coefficient bitwidths.

4.5 Hardware Analyzer

Simulating the PPA of the hardware for our designs, is done with the Hardware Analyzer of NeuroSim. The HA offers PPA for a whole chip design used for inference, where multiple crossbars are used, that are connected with wires and

buffers. This has been modified by us to only focus on a single large crossbar and its peripherals.

4.5.1 Testbench

We have developed a testbench to automate the simulation of various DFT sizes and configurations. The testbench for the HA consists of a Python script to generate the correct input traces (network and weights), and a bash script that compiles the C++ Hardware Analyzer and runs it iteratively with different parameters. Running the bench requires two arguments; which design to analyze and what size of N (64 or 1024). Then a full sweep test of input and coefficient bitwidths is done. It runs with the same inputs as the BS.

4.5.2 Custom Floorplan and Netlist

From the user manual [40] it specifies how to define the network for NeuroSim. This network is used for creating the floorplan, and how it should create the different layers. We have created our own netlist, that tells NeuroSim to create one single layer, with one big crossbar, with the correct size for input and output.

Table 4.6: Netlist configuration parameters for the single-layer NeuroSim implementation.

Parameter	Value	Description
IFM_H	1	Input Height (1D Vector)
IFM_W	1	Input Width (1D Vector)
IFM_D	`rows`	Input Depth (Crossbar Rows)
Kernel_H	1	Kernel Height
Kernel_W	1	Kernel Width
Kernel_D	`cols`	Output Depth (Crossbar Columns)
Pooling	0	No Pooling needed
Stride	1	Standard Stride

This makes sure that the simulated metrics used are for the physical implementation of the FFT, where only one big crossbar is used. By setting `IFM_H`, `IFM_W`, `Kernel_H`, and `Kernel_W` to 1, NeuroSim is forced to run a Matrix-Vector Multiplication (MVM) rather than a spatial convolution.

The **stride** is set to 1. In Convolutional Neural Networks, stride defines the step size of the kernel window. In NeuroSim, the output dimension is calculated as $N_{out} = \lfloor (H - K)/S + 1 \rfloor$ [40]. Thus, setting **stride** to 1 will give us 1 output dimension, ensuring a one-to-one mapping between the input vector and the crossbar output without any spatial reduction.

4.5.3 Parameters for Hardware Analyzer

All relevant parameters for Hardware Analyzer, derived from the papers by Athle et al. [28] and Borg et al. [22] are listed in table 4.7.

Since the 20 nm technology node is not natively supported by the Hardware Analyzer, we have used a hybrid modeling approach. We have explicitly defined the physical parameters for the FTJ memristor based on the data from Athle et al. [28]. For the peripheral circuitry, we have utilized the existing parameters to calculate custom parameters for 20 nm by interpolating between the available technology nodes. This is done to provide a realistic estimation of the area and energy scaling for the digital peripherals. The interconnecting wires has also been changed to tungsten, based on the data from Athle et al. [28].

Table 4.7: Device parameters for HA

Parameter	**Value**	**Unit**
Technology	20	nm
Bits per device	6 & 4	bits
Read voltage	0.3	V
Write voltage	2.4	V
Max Conductance (G_{max})	1.2	ns
On-Resistance (R_{on})	0.84	GΩ
Off-Resistance (R_{off})	8.4	GΩ

4.5.4 Isolating SubArray Metrics

By default, the standard Hardware Analyzer reports the total PPA for the entire chip hierarchy. To focus on the relevant components, the modularity of NeuroSim was utilized. We modified the `SubArray` class to isolate the crossbar metrics and all its peripherals (ADC, Mux and Shift-and-Add). This enables us to extract the relevant metrics and to exclude the global chip buffers and routing that are not relevant to our specific component-level analysis.

4.5.5 Tiling

Square tiling is natively supported in the HA of NeuroSim. This can be simulated by simply reducing the physical crossbar limit parameter, allowing NeuroSim's floorplanning algorithm to handle the spatial distribution automatically. However, this floorplanning algorithm is also the reason why rectangular tiles cannot be simulated, as the tool incorrectly instantiates and routes non-square tile dimensions.

Resistive Energy Modeling

The standard NeuroSim energy model (3.2) primarily captures capacitive dynamic energy through the CV^2 term, while resistive losses due to wire currents (I^2R) are not included in the energy calculations. For high-resistance devices such as FTJs, these losses are typically small due to the low operating currents [28]. However, as crossbar dimensions increase, the resistive dissipation in the interconnects can

become increasingly significant. To account for this effect, the `SubArray` class is extended to model ohmic losses in the wordlines and bitlines and take it into addition of the energy consumption.

The resistive energy is computed by modeling the wordlines (WLs) and bitlines (BLs) as distributed resistive interconnects, following the interconnect modeling approach described by Weste and Harris [42]. For a single line, assuming a uniform spatial distribution of active cells, the current decreases linearly with distance from the row driver (or increases toward the column sink). The power loss for a single interconnect carrying a specific line current I_{line} is obtained by integrating the square of the position-dependent current over the line resistance:

$$P_{\text{line}} = \int_0^L I(x)^2 \, dR(x) = \frac{1}{3} I_{\text{line}}^2 R_{\text{wire}} \tag{4.21}$$

To determine the total resistive energy for the array, the power loss must be summed across all parallel wordlines and bitlines. Unlike capacitive charging, where a global current is often assumed, resistive loss occurs independently in each active line. Therefore, the total dissipation is the sum of the losses in N active wordlines and M active bitlines.

The total resistive energy added to the simulation during a read operation is formally expressed as:

$$E_{\text{resistive}} = \frac{t_{\text{pulse}}}{3} \left[\sum_{i=1}^{N} \left(I_{\text{WL},i}^2 R_{\text{WL}} \right) + \sum_{j=1}^{M} \left(I_{\text{BL},j}^2 R_{\text{BL}} \right) \right] \tag{4.22}$$

If we assume a uniform current distribution where every active wordline carries an average current $I_{\text{WL,avg}}$ and every active bitline carries an average current $I_{\text{BL,avg}}$, the expression simplifies to:

$$E_{\text{resistive}} = \frac{t_{\text{pulse}}}{3} \left(N \cdot I_{\text{WL,avg}}^2 R_{\text{WL}} + M \cdot I_{\text{BL,avg}}^2 R_{\text{BL}} \right) \tag{4.23}$$

where t_{pulse} denotes the read pulse width, R_{WL} and R_{BL} represent the resistance of a single wordline or bitline, and N and M represent the number of active rows and columns, respectively. It must be noted that this is a first order model, based on strong assumptions. If more accurate values for IR drop are needed, then a SPICE simulation is required to get a more accurate model.

4.5.6 Elmore Delay Model

The RC delay that accumulates in the bitlines and wordlines as the size scales is not included in the standard latency calculations of NeuroSim. Thus, we have added a first-order Elmore Delay model to the simulation, based on theory on delay [42]. This comprises the distributed RC delay of the bitline wire and the lumped RC discharge time of the memory cells.

$$\tau_{\text{wire}} = \frac{1}{2} R_{\text{BL}} C_{\text{BL}} \tag{4.24}$$

where R_{BL} is the total bitline resistance and C_{BL} is the total bitline capacitance. The memory cells are modelled as a lumped resistive driver charging the bitline capacitance, with a time constant defined as:

$$\tau_{\text{cells}} = R_{\text{eff}} C_{\text{BL}} \tag{4.25}$$

where R_{eff} represents the effective parallel resistance of the active memory cells. The total bitline latency is derived by summing these terms. Furthermore, we apply a scaling factor of 0.2 to the cell time constant. This is derived from NeuroSim[35], to account for the small-signal sensing margin (voltage swing $\approx 18\%$ of V_{DD}):

$$t_{\text{BL}} = 0.2 \cdot \tau_{\text{cells}} + \tau_{\text{wire}} \tag{4.26}$$

4.5.7 Digital Adders

Because we use differential representation, the output needs to be reconstructed. Furthermore, when using the Baseline and Symmetry designs, additional adders are required. This has been implemented in NeuroSim, where we simulate:

$$M_{adders} = \begin{cases} \begin{cases} 6N & \text{for Baseline} \\ N & \text{for Symmetry} \\ 2N & \text{for Merged} \end{cases} & \text{if using Real Input} \\ \begin{cases} 14N & \text{for Baseline} \\ 4N & \text{for Symmetry} \\ 2N & \text{for Merged} \end{cases} & \text{if using Complex Input} \end{cases} \tag{4.27}$$

where:

- N is the size of the DFT.
- M is the total amount of digital adders.

For the real case, the differential outputs need to be reconstructed; thus, the amount required is the same size as the output vector. Therefore, the Symmetry design can suffice with only N adders, as only half the output is needed. However, for the complex case, the number of adders is doubled for both Baseline and Symmetry, as the hardware is doubled. Then, a further step is needed to reconstruct the output, as seen in (4.12). This requires $2N$ more adders and also needs to be computed afterwards, thus doubling the latency for the adders specifically. It should be noted that the differential reconstruction can also be done in the analog domain, before the ADCs. However, because this is not possible in NeuroSim due to limitations with digital values, we have chosen to do it in the digital domain.

For the Symmetry design, the conjugate properties are used to reconstruct the full output. However, the imaginary conjugate is negative value of the calculated value. Performing this subtraction requires inverting the input bits and setting the carry-in bit high to achieve two's complement negation. While this requires

additional inverters, the hardware overhead is negligible compared to the multi-bit adder itself. Therefore, the Adder module in NeuroSim is used to model both addition and subtraction operations.

4.5.8 Latency

To be clear on our definitions, latency in this report is the total delay between the first input cycle and the last output cycle. Specifically, it represents the total end-to-end temporal delay of the hardware array, spanning from the exact start of the first input at the peripheral circuits until the complete output is generated and fully read out, which effectively constitutes the critical path for a complete DFT operation.

4.5.9 ADC

There are two available ADCs to choose from in NeuroSim: a multilevel sense amplifier or a SAR ADC. However, because we changed which device we use, we opted for the SAR ADC. Otherwise we would have needed to explicitly model the multilevel sense amplifier in a SPICE simulator, to then change the model used in NeuroSim. Whereas the SAR ADC is scaled automatically, depending on the device parameters.

4.5.10 Muxing in NeuroSim

NeuroSim supports the usage of Muxes, placed before the ADCs in their SubArray module. The Mux is used to significantly reduce the area cost of ADCs. In the user manual of NeuroSim [40], it is stated that the Mux can be used to share the read periphery circuits among synaptic array columns. This is to make it more area efficient. However, this will increase the latency, as time multiplexing is needed, which will be controlled by the Mux Decoder. In our testbench, we have chosen to simulate without a mux, to minimize the latency, at the cost of area usage. This was done as for 5G and 6G technology, the latency is the more important metric. To show the impact of muxing, we have performed an additional simulation with 16 columns per ADC.

4.5.11 Assumptions made for HA Modifications

To enable the evaluation of the proposed designs within the simulation framework, the following assumptions have been made regarding the device models and physics.

- **Distributed Interconnects:** The wordlines and bitlines are modelled as distributed resistive interconnects, assuming a uniform spatial distribution of active cells to calculate resistive losses.
- **Uniform Current Distribution:** For the simplified resistive energy model, it is assumed that every active wordline and bitline carries an average current, rather than modeling the exact current gradient for every unique input pattern.

- **First-Order Delay:** The interconnect latency is approximated using a first-order Elmore Delay model. In this configuration, memory cells are modelled as lumped resistive drivers charging the total bitline capacitance.
- **Hybrid Technology Scaling:** As the 20 nm technology node is not natively supported by the Hardware Analyzer, a hybrid modeling approach is assumed. Physical parameters for the FTJ device are explicitly defined, while the area and energy for peripheral circuitry are interpolated between available technology nodes.
- **Adder/Subtractor Equivalence:** The hardware overhead required for two's complement inversion is assumed to be negligible compared to the multi-bit adder itself. Therefore, the standard Adder module is utilized to model both addition and subtraction operations.

Chapter 5
Result & Discussion

The focus of this chapter is to present the simulation results and evaluate the design parameters. First, a small DFT is simulated to figure out the impact of parameters on a smaller crossbar. In the next section a large DFT is evaluated for real input. The investigation includes the impact of static IR drop on the larger crossbars. This section also highlights how the energy consumption and latency increases with larger inputs.

Finally tiling is explored as a potential solution to mitigate the IR drop noise on the large DFT. The ADC quantization error, latency and energy consumptions are considered together to find a tile size with balanced performance in these three aspects.

5.1 Small Size DFT

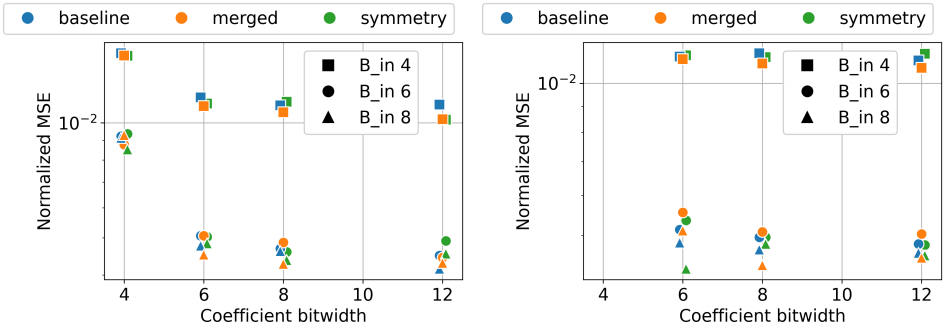
The small 64-point DFT is assessed for real and complex input. All three designs are compared to find the best one in terms of accuracy and PPA. Tradeoffs are investigated by exploring different combinations for device., coefficient and input bitwidth. In addition, the impact of static IR drop and conductance drift is analyzed.

5.1.1 Accuracy

The total accuracy loss as shown in Figure 5.1 consists of two parts; the quantization error and hardware-induced accuracy loss.

Quantization

Figure 5.2 shows that quantization error is independent of design. It only depends on the input and coefficient bitwidth, which also follows from how the quantization is performed as described in Section 4.4.1. The trend that can be found in Figure 5.2 is that larger coefficient and input bitwidth creates smaller quantization error. While the accuracy loss is larger for complex input compared to real input, the trend remains. The quantization error for complex input is larger since both the real and complex input vectors are subject to quantization noise.



(a) Real input, 4-bit device

(b) Real input, 6-bit device

(c) Complex input, 4-bit device

(d) Complex input, 6-bit device

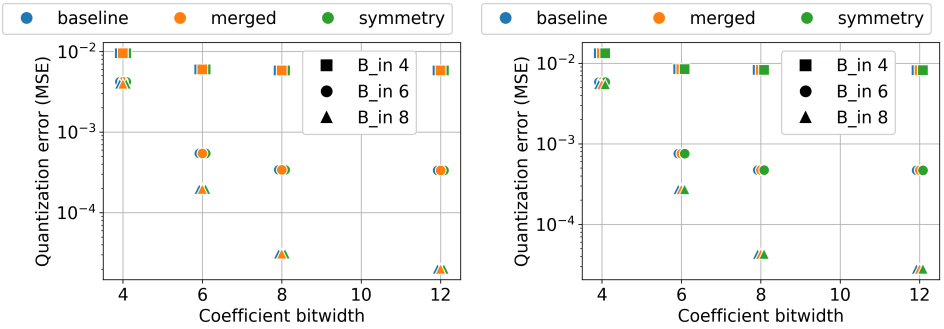
Figure 5.1: Normalized MSE of each design illustrating the impact of coefficient and input bitwidth.

From Figure 5.2 we can also see that increasing the coefficient bitwidth beyond 8 bits leads to minimal improvement. Increasing the input bitwidth however has large impact, even when increasing the coefficient bitwidth no longer improves the accuracy

Hardware Non-Idealities

The hardware-induced accuracy loss refers to the effect of device and system non-idealities and does not include the quantization error. In Figure 5.3 it can be seen that the impact of input bitwidth on the accuracy does not appear to follow any trend. It can also be observed that the bits per device and design parameters play a larger role in affecting the hardware-induced accuracy loss. In general more states per device experience better accuracy.

Coefficient bitwidth has a small impact on the accuracy. In the case 4 bits per weight and a 4-bit device, Figures 5.3a and 5.3c, the accuracy is slightly worse across the board compared to higher coefficient bitwidths. While for a 6-bit device, Figures 5.3b and 5.3d, the impact of coefficient bitwidth is smaller. So for a 4-bit device, introducing coefficient slicing has larger robustness to accuracy degradation as a result of device non-idealities. And for a 6-bit device, coefficient slicing has



(a) Real input **(b)** Complex input

Figure 5.2: Quantization error as a function of input and coefficient bitwidth, expressed as normalized MSE.

little impact on hardware-induced accuracy loss.

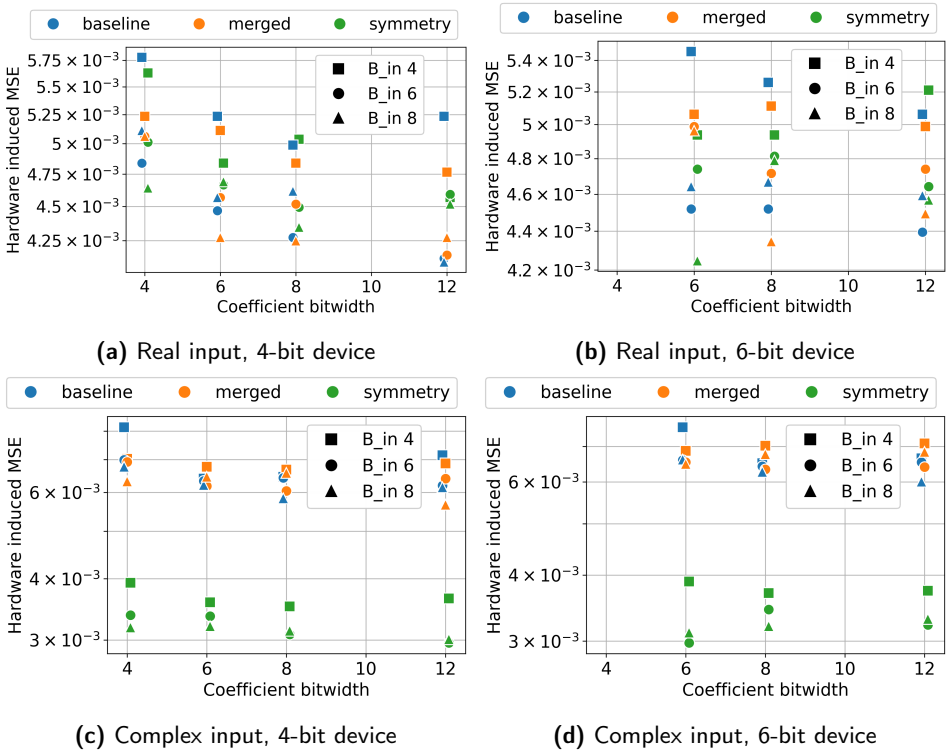
Coefficient slicing has in fact been found to slightly improve the inference accuracy of neural networks due to device variations cancelling each other out when reconstructing the output [27]. Which might explain the slight improvement of accuracy in the small DFT when coefficient slicing is used. However the general impact of coefficient bitwidth on the accuracy is very small.

The choice of design also affects the hardware-induced accuracy loss. The Merged and Baseline designs have slightly worse accuracy performance compared to Symmetry. This is expected since the Symmetry design employs far less hardware as evident by Table 4.1. This trend is especially apparent for complex input, where once again the number of memristors are far less in the Symmetry design compared to the other designs. For real input the difference between designs is less distinct. So while the crossbar size of the Baseline design is the smallest, it is the total amount of hardware that mostly impact the hardware-induced accuracy loss for small DFT. This is since IR drop effects on accuracy are hardly visible for small DFT no matter the design.

For a 6-bit device the Merged design performs slightly worse than the Baseline design despite using the same number of devices, while for a 4-bit device the performance of these two designs are on par. This might stem from the Baseline design requiring more ADC compared to the stacked architectures, which makes the design potentially more susceptible to ADC quantization noise.

Combined Impact of Quantization and Hardware Non-Idealities

The total accuracy loss, presented in Figure 5.1, is how much the simulated DFT are affected by both hardware and quantization effects combined. The total accuracy loss is roughly the sum of the quantization and hardware-induced accuracy losss. However there can be instances where, on chance, the quantization and hardware-induced accuracy losss compensate for each other and minimally decrease the total accuracy loss. For that reason, the total accuracy loss presented in Figure 5.1 is the simulated DFT compared with the floating point reference



(a) Real input, 4-bit device

(b) Real input, 6-bit device

(c) Complex input, 4-bit device

(d) Complex input, 6-bit device

Figure 5.3: Hardware-induced accuracy loss as a function of input and coefficient bitwidth, expressed as normalized MSE.

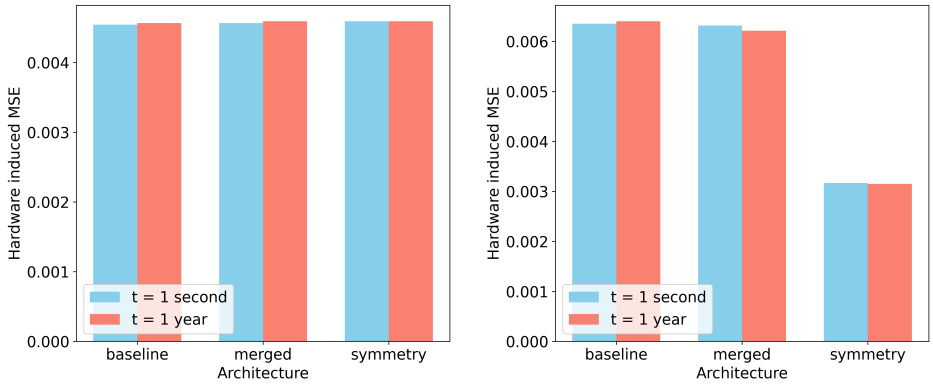
model rather than the sum of the other two accuracy loss metrics.

In general all three designs perform similarly in terms of accuracy. Which indicates that the quantization error is the largest part of the total accuracy loss since quantization effects are only affected by weight and input bitwidth. In fact when comparing the magnitude of quantization error, Figure 5.2, with that of the accuracy loss from hardware non-idealities, Figure 5.3, the hardware-induced accuracy loss is of a magnitude of $\sim 10^{-3}$ while the quantization error is of a magnitude of $[10^{-2}, 10^{-4}]$. From this follows that configurations where the quantization error is low is more susceptible to hardware non-idealities. This is also evident from the data in Figure 5.3 where the combinations with larger input and coefficient bitwidths experience larger variation between designs.

The impact of increasing coefficient bitwidth to improve accuracy saturates at 6 bits. The reason for this is that the twiddle factors used for 64-point DFT can be sufficiently represented by 6 bits. To improve the accuracy further, the input bitwidth can be tweaked for larger impact. For real input the lowest accuracy loss simulated is of 4.25×10^{-3} MSE and for complex input this is roughly 3×10^{-3} MSE.

Conductance Drift

The conductance of the memristor will slowly change its value with time. The results for the conductance drift investigation is presented in Figure 5.4. Only the result for a configuration with 4-bit device, 8-bit weights and 6-bit input is presented in the figure, but the same behavior is observed for any combination of parameters. In Figure 5.4 it can be seen that the hardware-induced accuracy loss as a result of the conductance drift after one year is minimal compared with after one second. This is also supported by the findings in [22].



(a) Real input **(b)** Complex input

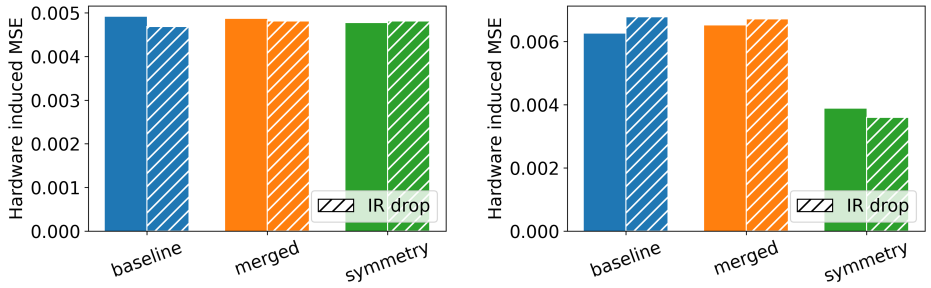
Figure 5.4: Hardware-induced accuracy loss as a result of conductance drift expressed as normalized MSE.

For complex input the Symmetry design appears to have a larger MSE at $t = 1s$ compared to after one year of drift as seen in Figure 5.5b. Since there is a certain randomness to how the drift occur it might mean that the drift can improve the accuracy by chance though compensating for the conductance variation. However the impact of drift on the accuracy is less than 10^{-3} MSE across the board. The implications of this is that frequent re-programming of the devices is not needed.

IR Drop

In Figure 5.5 the hardware-induced accuracy loss for a 4-bit device with 8 bits per weight and 6-bit input configurations are shown. The system-level impact of IR drop are shown compared to ideal crossbars with no IR drop. Other system non-idealities are also still simulated.

From Figure 5.5 the IR drop appears to have a small negative impact on the accuracy for real input across all designs and slightly more impact on Merged and Symmetry design for complex input. The IR drop effects on Baseline design appears to have positive impact on the system accuracy. Note however that the impact of IR drop very small and could be attributed to the random device-to-device- and read variation. As such the impact of IR drop on small size DFT (even for the Merged design) is found to be negligible.



(a) Real input **(b)** Complex input

Figure 5.5: Hardware-induced accuracy loss with and without IR drop, expressed as normalized MSE.

5.1.2 PPA

Energy

In Figure 5.6, the comparison between the designs and using different parameters can be seen. There we can see how the energy scales from using different coefficient bitwidth for each device bitwidth, showing how the size of the crossbar is extended if coefficient bitwidth is larger than bits per device due to coefficient slicing. We can see that it is more energy efficient, regardless of design, to map 8 bits to two 4-bit devices, rather than two 6-bit devices. This is because the ADC resolution will increase with the use of a device with higher device bitwidth according to (2.12).

Increasing the input bitwidth inherently leads to higher total energy consumption. This is due to the bit-serial nature of the input processing in the simulated architecture, which requires the hardware to evaluate the data sequentially. Consequently, for every additional bit of input precision, the hardware must remain active for an additional cycles to complete the multiply-accumulate operations. Because the active peripheral circuitry continuously consumes dynamic power during each of these operation cycles, the total dynamic energy consumption of the system increases linearly with the input bitwidth.

In Figure 5.7, the breakdown of the energy metrics can be seen, using a chosen set of parameters. Looking at these, it is clearly seen that the ADC is the component that uses the most energy. The ADC approximately consumes 80% of the total energy, followed by the Shift-and-Add components that consumes approximately 15%. That is over 95% of the total energy consumed by peripherals. Thus, the merged and Symmetry design consumes less energy than the baseline design, due to the reduced number of ADCs. Furthermore, in the real case, the Symmetry is even more energy efficient as it reduces the ADCs by half, compared to the merged design. However, in the complex case, both the merged and Symmetry uses the same amount of ADCs, but since the merged design employs twice the number of rows, it has a higher ADC resolution. This leads to a higher energy consumption of $\approx 10\%$ for the merged design compared to symmetry in the

complex case.

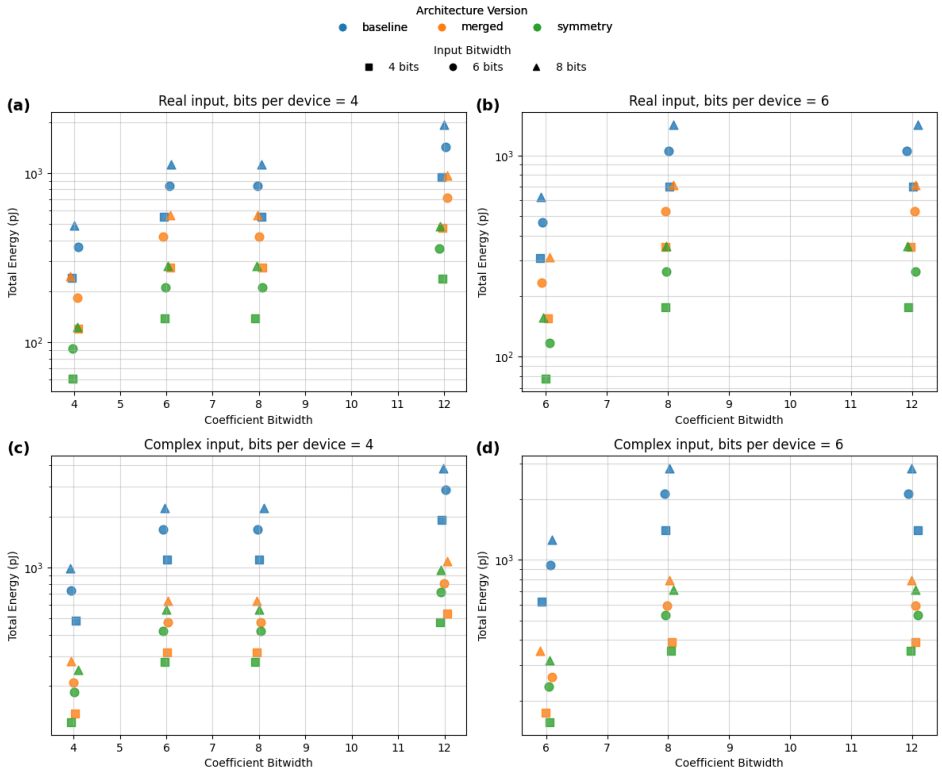


Figure 5.6: Energy for small DFT, as a result of bits per device, input and coefficient bitwidth. (a) Real input with 4-bit device, (b) Real input with 6-bit device, (c) Complex input with 4-bit device, and (d) Complex input with 6-bit device.

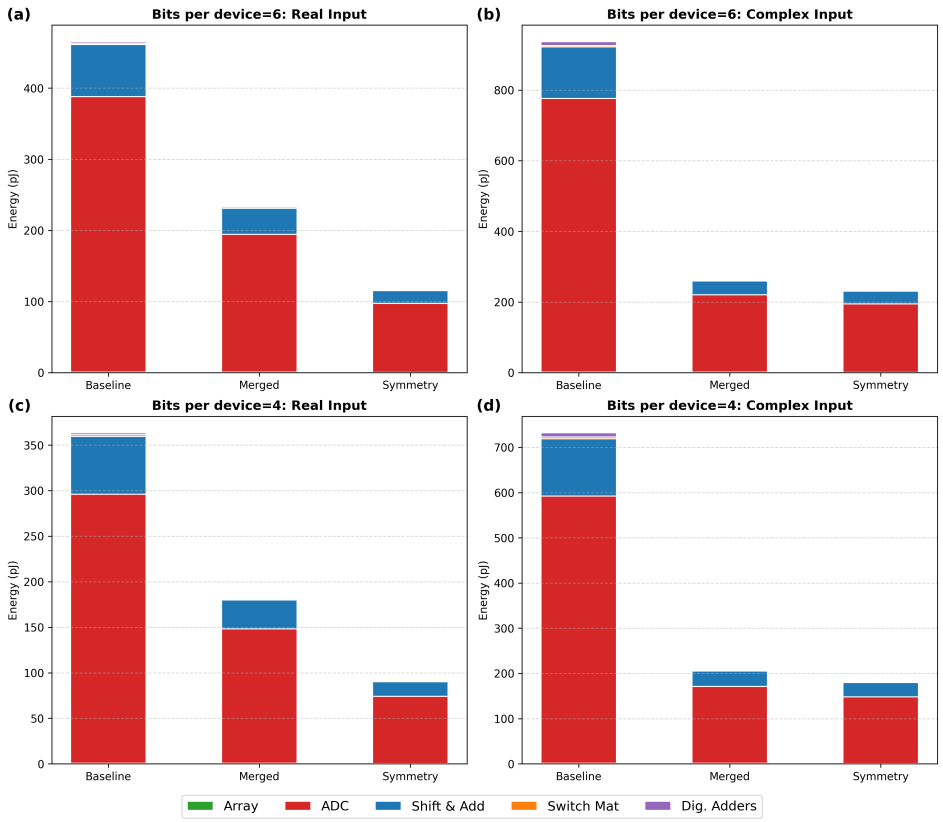


Figure 5.7: Energy breakdown for small DFT. (a) Real input with 6-bit device, (b) Complex input with 6-bit device, (c) Real input with 4-bit device, and (d) Complex input with 4-bit device.

Area

Figure 5.8 compares the designs' area usage, for different coefficient bitwidth, input bitwidth, bits per device and for both the real and complex case. From this it can been seen that input bitwidth does not impact the area. This is in fact due to how NeuroSim handles input slicing, by multiplexing the input to the same hardware. In theory it would be possible to extend the hardware and compute all bits at the same time, thus reducing the latency but increasing the area costs.

In the Figure 5.9, the area breakdown of a small DFT can be seen for the different designs. From this figure it is clear that the peripherals use the highest amount of area, with the ADCs being the largest, followed by the switch matrix and then the Shift-and-Add components. These three components take $\approx 93\%$ of the total area. For the real case, the Symmetry design uses the least amount of area, due to the reduced peripherals. However, for the complex case, the merged design uses the smallest area, as it has the least amount of peripherals. The Symmetry design is slightly larger, since it requires the use two switch matrices. In terms of area, the merged design performs best because it uses the fewest peripherals.

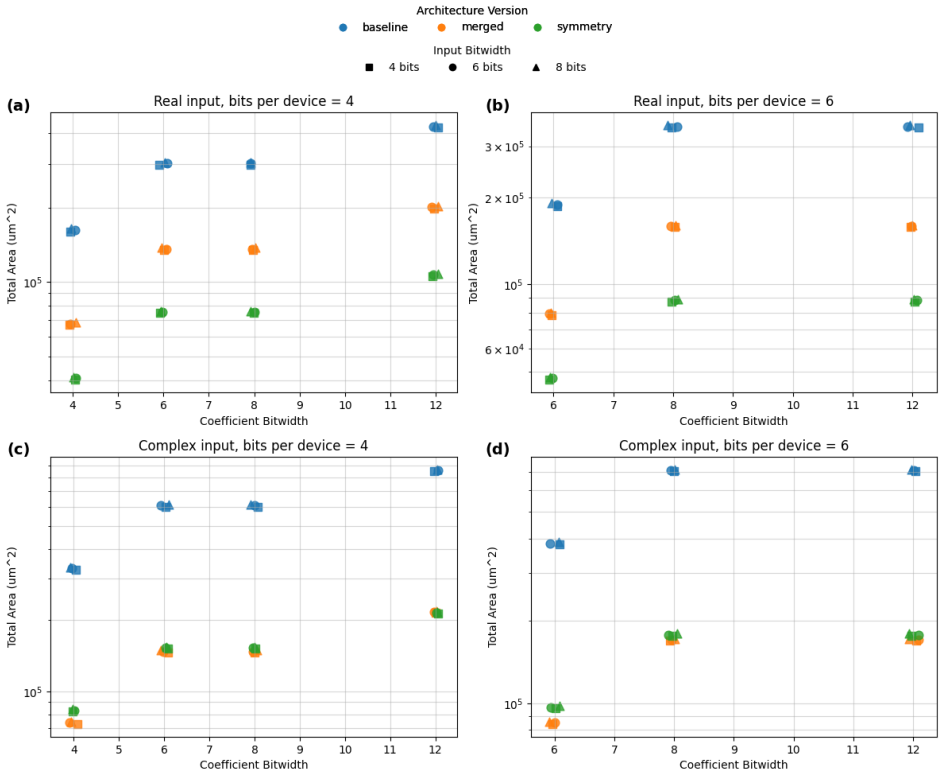


Figure 5.8: Area for small DFT, as a result of bits per device, input and coefficient bitwidth. (a) Real input with 4-bit device, (b) Real input with 6-bit device, (c) Complex input with 4-bit device, and (d) Complex input with 6-bit device.

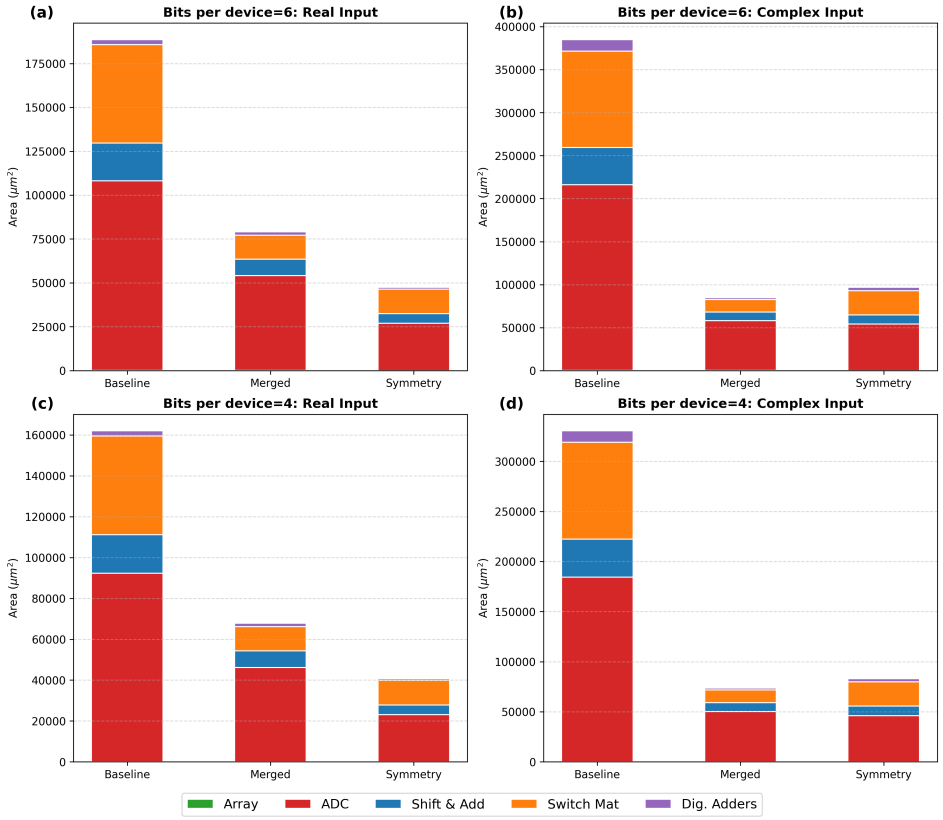


Figure 5.9: Plot of area breakdown for small DFT. (a) Real input with 6-bit device, (b) Complex input with 6-bit device, (c) Real input with 4-bit device, and (d) Complex input with 4-bit device.

Latency & Throughput

Looking at Tables 5.2 and 5.1, they show us the critical path of the DFT. The critical path is the summation of the latency for the Switch Matrix, ADC, and digital adders. It becomes very clear that the big bottleneck in terms of latency is the ADCs. It must be noted that due to the input being done in bit-serial, the Shift-and-Add components after the ADCs get pipelined. Therefore, they are not included in the critical path.

The latency of the ADC is directly tied to its resolution. A higher resolution requires more time to convert the analog value to digital. For the real case, both Symmetry and merged design have the same latency and throughput, because their ADC resolution is the same. However, in the complex case, the Symmetry design is slightly faster. This is due to the rows being $4N$ for the merged, and $2N$ for the Symmetry crossbars, which requires the merged design to utilize a slightly higher ADC resolution.

The comparison of the latency and throughput for a small DFT can be seen in

Figure 5.10. From this figure it is clear how the input bitwidth affects the latency. Due to NeuroSim being unable to do multi bit input, it instead does bit-serial. Therefore, requiring more cycles to process each bit of input. This is currently the biggest impact to latency and throughput.

However, the coefficient bitwidth does not affect the latency, even though coefficient slicing is needed for larger precisions. While slicing expands the physical size of the crossbar, everything is still run in parallel. Increasing the array size does introduce a slight RC delay to the wires, but this impact on the array latency is practically negligible compared to the ADC bottleneck.

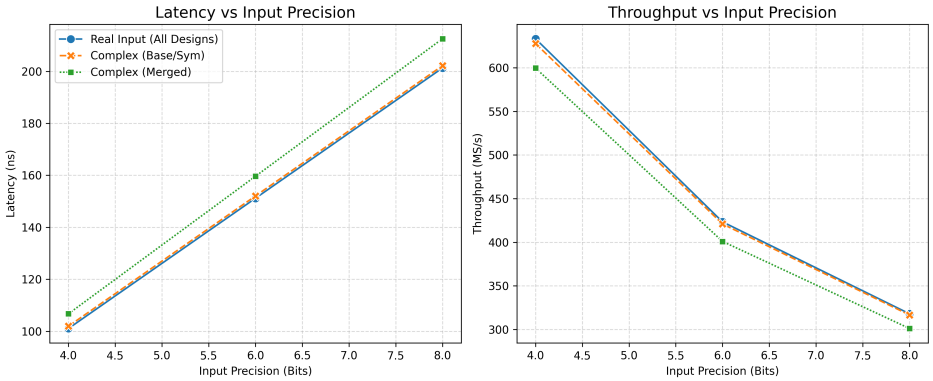


Figure 5.10: Latency and Throughput for small DFT using a 6-bit device, as a result of input bitwidth

Table 5.1: Latency breakdown (ns) for small DFT (6-bit device, Weight=6, Input=6).

Design	ADC	Array	Sw. Mat.	D. Adders	Total
		Real Input			
baseline	120.91	29.34	0.02	0.89	151.16
merged	120.91	29.34	0.02	0.89	151.16
Symmetry	120.91	29.34	0.02	0.89	151.16
		Complex Input			
baseline	120.91	29.34	0.02	1.77	152.05
merged	129.30	29.35	0.02	0.95	159.62
Symmetry	120.91	29.34	0.02	1.77	152.05

Muxing

The area overhead of peripheral circuitry can be heavily mitigated by introducing multiplexers (muxes) at the crossbar outputs. Table 5.3 illustrates this effect, showing that configuring 16 bitlines to share a single ADC reduces the total system area by 78%. This spatial efficiency, however, dictates a severe performance trade-off. Because the shared ADC must process the column outputs sequentially rather than in parallel, the overall latency increases significantly, resulting in a proportionally lower throughput.

5.2 Single Crossbar Implementation of Large Size DFT

In this section we increase the size of the input to 1024 and investigate real input DFT. In the previous section the symmetry design was found to be on par with

Table 5.2: Latency breakdown (ns) for small DFT (4-bit device, Weight=4, Input=6).

Design	ADC	Array	Sw. Mat.	D. Adders	Total
		Real Input			
baseline	104.11	29.34	0.02	0.76	134.24
merged	104.11	29.34	0.02	0.76	134.24
Symmetry	104.11	29.34	0.02	0.76	134.24
		Complex Input			
baseline	104.11	29.34	0.02	1.53	135.01
merged	112.50	29.35	0.02	0.83	142.70
Symmetry	104.11	29.34	0.02	1.53	135.01

Table 5.3: Comparison of metrics with and without Multiplexing for the Symmetry design (Complex Input, $N = 64$, 6-bit device, wp=6, ip=6, 16 columns per ADC).

Metric	Without Mux	With Mux	Change Factor
ADC Area (μm^2)	54041	3377	$\approx -\mathbf{94}\%$
Total Area (μm^2)	96732	21290	$\approx -\mathbf{78}\%$
Total Latency (ns)	150	2401	$\approx +16\times$
Total Energy (pJ)	236	269	$+14\%$

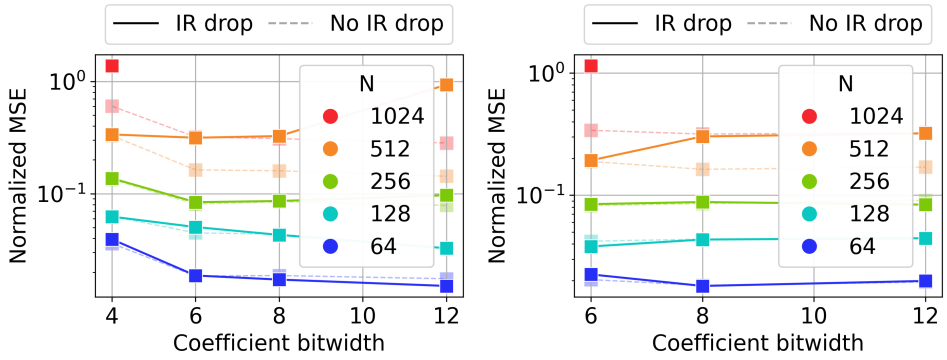
the others (or slightly better) in terms of accuracy and latency and with significant energy and area savings for real input. As a result this section only considers the symmetry architecture.

5.2.1 Accuracy

When increasing the size of the DFT beyond 64, there are a number of factors that degrade the accuracy of the calculations. The key issues are quantization and IR drop. In Figure 5.11 we can see the total accuracy as a result of increased transform size. And in Figures 5.12 and 5.13 we can see the impact of hardware non-idealities and quantization effects respectively. For clarity, Figures 5.11 and 5.12 are restricted to 8-bit inputs. Nevertheless, the observed behavior is consistent across all considered input bitwidths.

Due to memory limitations on the workstations used for simulation, DFT length of 1024 could not be simulated for coefficient bitwidths that require coefficient slicing. From the trend shown in Figure 5.11 it is evident that for $N = 512$ the IR drop significantly impacts the accuracy of the DFT. For even larger transform sizes the IR drop exacerbates the accuracy degradation, which is evident from the data points that display $N = 1024$ without coefficient slicing in Figure 5.11.

So while there is no data to support it, large DFT with coefficient slicing is



(a) 4-bit device **(b)** 6-bit device

Figure 5.11: Normalized MSE with and without IR drop simulation for different DFT lengths and coefficient bitwidths.

expected to have even worse accuracy with the larger crossbar utilized on account of more devices per weight.

Quantization

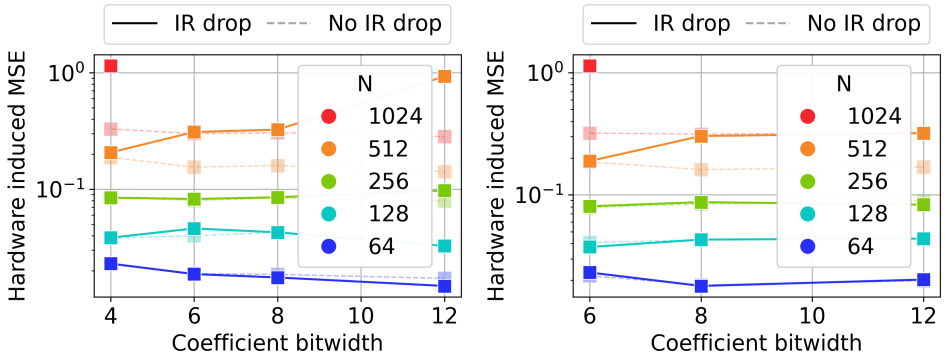
Increasing the DFT size to 1024 increases the number of twiddle factors involved, which in turn raises the precision required to keep the coefficients separable and prevent them from mapping to the same conductance values. Additionally the inputs also need to be sufficiently represented to ensure differentiation between values.

From Figure 5.13 we can see that in order to achieve similarly low quantization error for $N = 1024$ to that of $N = 64$ a large input and coefficient bitwidth both are needed. Increasing both the input and coefficient bitwidths to 8 pushes the quantization error beyond 10^{-2} and establishes low base error and sets up more tolerance for hardware non-idealities.

Hardware Non-Idealities

The effect of hardware non-idealities is far more pronounced for larger transform sizes. In Figure 5.12 the impact of hardware non-idealities have been isolated. Simulations with and without IR drop are plotted to highlight the effect of IR drop. Take into account that the crossbar size of the Symmetry design is $2N \times 2N$. So for $N = 256$ the crossbar dimensions are in fact 512×512 (without any coefficient slicing).

In previous analyses (Section 5.1.1) it was found that larger crossbar designs experience some inaccuracies due to the accumulated accuracy loss from device non-idealities. However for the Symmetry design and larger transform size it appears that this effect is not very noticeable. IR drop on the other hand is a significantly larger portion of the hardware-induced accuracy loss for $N \geq 256$. For



(a) 4-bit device **(b)** 6-bit device

Figure 5.12: Hardware-induced accuracy loss with and without IR drop simulation for different DFT lengths and coefficient bitwidths expressed as normalized MSE.

coefficient bitwidths larger than the bits per device, where coefficient slicing occurs, the IR drop is particularly large. Note that due to the increased computation time when simulating IR drop, only one set of inputs was simulated. This results in some randomness in the plotted values. This is likely why the values in Figure 5.12 for $N = 64$ do not correspond exactly to those in Figure 5.3.

Combined Impact of Quantization and Hardware Non-Idealities

The total impact of the quantization and hardware non-idealities combined are found in Figure 5.11. Simulations conducted with and without IR drop are presented to demonstrate that IR drop is the primary cause of the observed accuracy degradation. Figure 5.11 illustrates that increased coefficient bitwidth will not necessarily improve the accuracy. This stems from the magnitudes of the two error components; quantization error might be small for larger coefficient bitwidths, but the hardware-induced accuracy loss is several times larger and dominates the total error due to IR drop in larger crossbars.

For 1024-point DFT the MSE is right around ~ 1 for the smaller coefficient bitwidth where no coefficient slicing is used. Larger input bitwidths where coefficient slicing is used is likely subject to even more IR drop and will experience larger accuracy loss. And as a result, unless this magnitude of error is acceptable, large DFT implemented in a single crossbar is not viable with regard to accuracy.

5.2.2 PPA

Energy

Scaling the DFT size significantly impacts energy consumption. The trend, i.e. the linear scaling of the energy can be seen in Figure 5.17. This behavior is expected, as the power-hungry peripherals of both the ADCs and Shift-and-Add

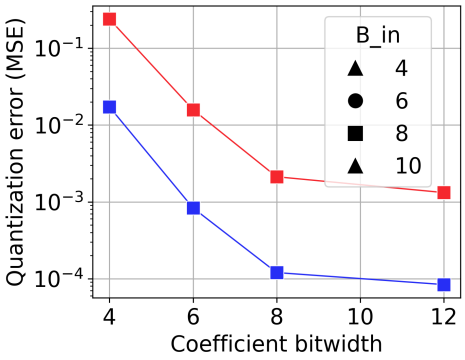


Figure 5.13: Quantization error comparing $N = 64$ and $N = 1024$, input and coefficient bitwidth.

components increase in both quantity and required resolution. Furthermore, the energy contribution of the memristors becomes more pronounced at larger sizes, as illustrated in Figure 5.15. This trend in array energy is attributed to capacitive switching, which follows a quadratic scaling law due to the number of memristors increasing quadratically. Notably, however, the IR drop remains negligible even at $N = 1024$. This effective mitigation of IR drop, is due to the specific FTJ device characteristics [28], where the extremely high device resistance results in low currents, thereby minimizing the energy cost associated with IR drop. In Table 5.5, this can been as reducing the tile size, does not reduce the energy cost of the array. A 2048×4096 crossbar, consumes 211.3 pJ, while multiple smaller 64×64 tiles together consume 211.2 pJ.

In Figure 5.17 it can be seen that, using two 4-bit devices is more optimal, in terms of energy and area, than two 6-bit devices, when 8-bit is used as coefficient bitwidth. This is true for all sizes $N \in \{64, 128, \ldots, 1024\}$.

Area

The area scales as expected as seen in Figure 5.17, with an increase in DFT size leading to larger peripheral circuitry. Double the size of the DFT, double approximately the cost of area. Although the memristor array grows quadratically with N, the total footprint remains dominated by the peripherals. As N increases, the resolution of the required ADCs increases logarithmically (2.12). Higher resolution requires physically larger ADCs, ensuring they remain the primary contributor to the total area, as seen in Figure 5.16.

Latency & Throughput

Latency increases with the DFT size primarily due to the increased ADC resolution requirements. A larger transform size results in a larger accumulation of current, necessitating higher resolution ADCs to avoid clipping and quantization error, which in turn take longer to convert. However, the throughput (MS/s) remains

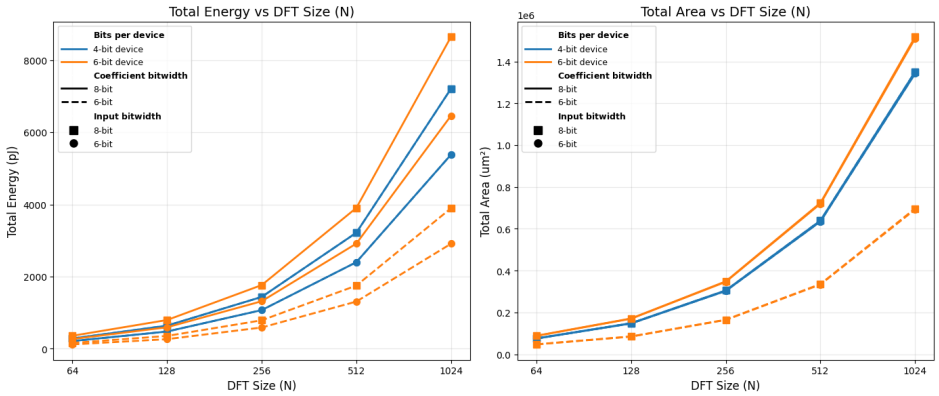


Figure 5.14: Energy consumption and area for different N, input and coefficient bitwidth. The design is Symmetry for real only input. *(Note: The lines for the 4-bit device overlap perfectly.)*

relatively high because the crossbar processes the larger vectors in parallel as can be seen in Figure 5.18. Increasing the size of the DFT increases the amount of samples computed, and therefore, the throughput increases for large N, even though latency increases.

The IR drop does not significantly affect the latency, as the delay is dominated by the ADC conversion time rather than the RC delay of the wires. In Figure 5.19, a full breakdown of each latency is shown, and how they are affected by larger size. There it is clearly seen the low impact of RC delay on the Array Latency, which barely gets higher. This behavior is explained by the properties of our chosen FTJ device, specifically its low conductance [28].

5.3 Employing Tiling for Large Size DFT

By analyzing the large DFT implemented in a single crossbar it is evident that the Symmetry design cannot accurately support 1024 input size due to IR drop. Even without coefficient slicing the dimensions of the crossbar are subject to large distortions due to IR drop. Increasing the bit width of the weights would double or even triple the number of bit-lines and render the output unusable.

To address this problem we investigate splitting the crossbar into smaller tiles which are subject to IR drop to a far lesser extent. First we explore square tiles to map out the general considerations for tiling. Then, using the insights from square tiles we find a more optimal tile shape and size for improving accuracy. The observed behavior is consistent across all considered input bitwidths. Unless otherwise specified, Figures in this chapter are limited to 8-bit inputs for brevity.

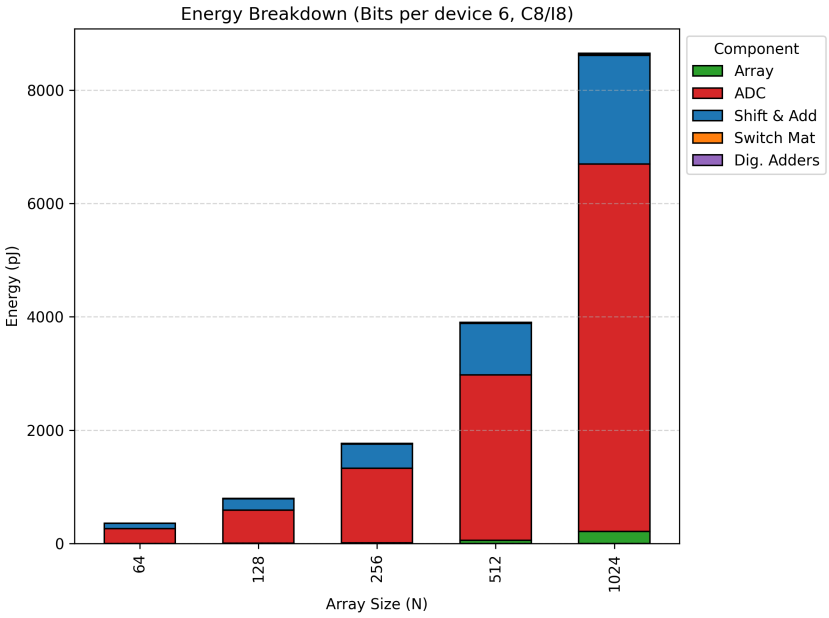


Figure 5.15: Energy breakdown for different DFT size (N), for 8 bits coefficient bid-width, 8-bit input bid-width and a 6-bit device. The design is Symmetry for real only input.

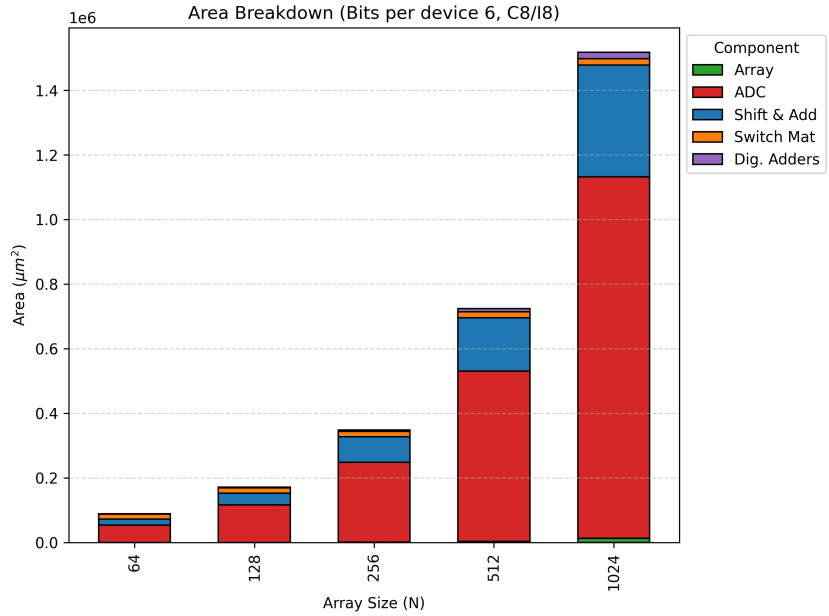


Figure 5.16: Area breakdown for different DFT size (N), for 8 bits per weight, 8-bit input and a 6-bit device. The design is Symmetry for real only input.

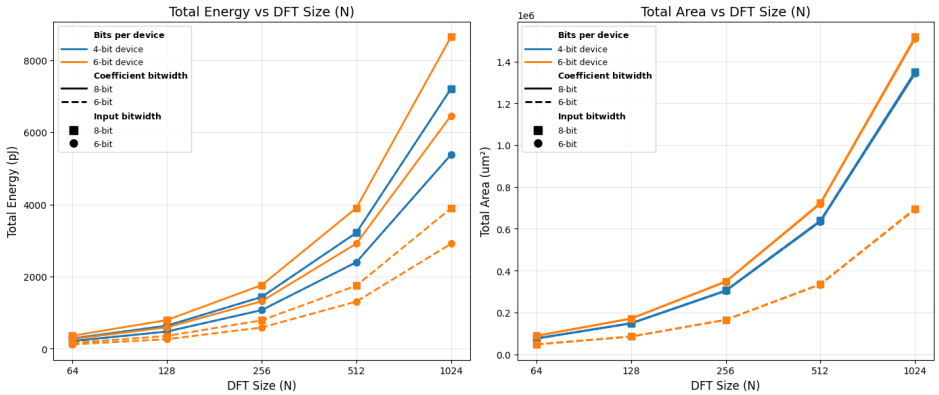


Figure 5.17: Energy consumption and area for different N, input and coefficient bitwidth. The design is Symmetry for real only input. *(Note: The lines for the 4-bit device overlap perfectly.)*

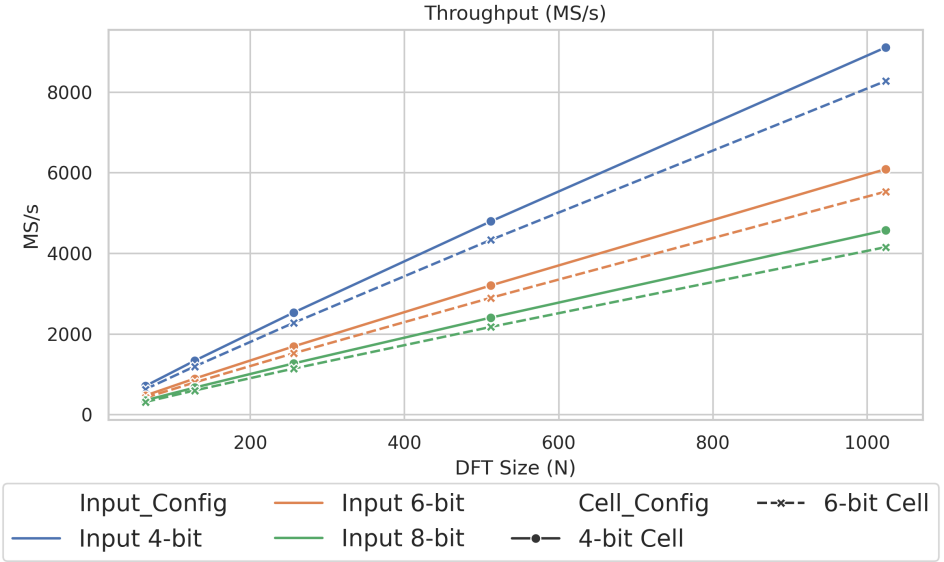


Figure 5.18: Throughput (mega samples/second) for different N and input bitwidth. Results are grouped by input bitwidth and Cell Technology. The design is Symmetry for real only input.

5.3.1 Square tiles

Accuracy

In Figure 5.20 the total accuracy loss of 1024-point DFT with different tile size T are compared. Contrary to intuition the smallest tiles do not result in the least accuracy loss. In theory, the accuracy in larger tiles should deteriorate due to IR drop. However, in Figure 5.20 IR drop only impacts the accuracy loss when coefficient slicing is employed, which leads us to consider other factors that might impact the accuracy.

The usage of tiles introduces a multitude of ADCs compared to using a single crossbar. The output of each ADC is subject to quantization error. Smaller tile sizes require more ADCs which contribute to larger accuracy degradation. In Table 4.2 the number of tiles used for large real input Symmetry design as a result of tile size T is shown. The number of ADC operations performed on a single output element depends on how many tiles are placed in the vertical direction. As shown in the table, doubling the tile size results in half the amount of ADCs.

Bear in mind that the quantization error as a result of the weight and input bitwidths remains unchanged when tiling. This observation clarifies how device non-idealities, together with the additional quantization error introduced by extra ADC conversions during tiling, shape the overall accuracy loss distribution shown in Figure 5.20. Consequently, the best accuracy is found for the tile size $T = 512$. For a 4-bit device using 8-bit input and coefficient bitwidth the accuracy loss is less than 2×10^{-2} MSE. Which is a significant improvement compared to the accuracy loss of the single crossbar.

PPA

Tiling with square tiles has an interesting effect on the PPA values, and this is mainly due to how NeuroSim handles multiple crossbars. As can be seen in Table 5.4, both area and energy massively increase with reduced tile size. This is unfortunately due to NeuroSim placing ADCs after each crossbar; thus, the more crossbars required for smaller tiles, the more ADCs are placed. These peripherals have a massive energy cost, as seen in Table 5.5. This occurs despite the ADCs having reduced resolution, as the resolution is decided by the size of the tile. Without this limitation, it would have been possible to connect all tiles in the analog domain, thus greatly reducing the need for and costs of peripherals.

There is a strong negative correlation between tile size and resource consumption. As the size of the tiles decreases, the total number of tiles increases quadratically to support the full $N = 1024$ matrix. Thus, we get this massive overhead of energy and area costs with decreased tile size, and the breakdowns in Figure 5.22b and Figure 5.22a show us exactly how much each component costs. Interestingly, for tile size $T = 64$, the Switch Matrix is the dominant cost in terms of area, while having negligible energy cost compared to the ADCs and Shift-and-Add components.

Conversely, there is a positive correlation between tile size and latency, meaning smaller tiles yield faster execution. As shown in Table 5.4, the total latency decreases as the tile size shrinks. This is primarily due to the reduced requirements

on ADC resolution. According to (2.12), a smaller tile size requires fewer quantization levels, which allows ADCs to complete the conversion faster. The breakdown in Figure 5.23 confirms that the ADC latency (red bar) decreases significantly for smaller tiles ($T = 64$) compared to the large tile size of ($T = 1024$).

Table 5.4: Impact of Tile Size (T) on PPA Metrics for Large DFT ($N = 1024$). For Weight=8, Input=8 and a 6-bit device

Tile Size (T)	**Total Area** (μm^2)	**Total Energy** (pJ)	**Latency** (ns)	**Throughput** (MS/s)
64	66.4×10^6	162,869	190.0	5,390
128	27.4×10^6	91,279	201.3	5,088
256	12.1×10^6	50,871	212.5	4,818
512	5.8×10^6	28,236	223.8	4,576
1024	3.0×10^6	15,639	235.0	4,357
Single CA (2048)	1.5×10^6	8,649	246.3	4,158

Table 5.5: Breakdown of Energy Components vs. Tile Size ($N = 1024$, Weight=8, Input=8, 6-bit device).

Tile Size (T)	**Array Energy** (pJ)	**ADC Energy** (pJ)	**Total Energy** (pJ)
64	211.2	116,058	162,869
128	211.2	66,150	91,279
256	211.2	37,389	50,871
512	211.2	20,978	28,236
1024	211.3	11,695	15,639
Single CA (2048)	211.3	6,483	8,649

5.3.2 Rectangular Tiles

In the previous sections it was found that employing square tiles makes the calculation subject to large ADC quantization noise. To address this the number of vertical tiles should be minimized. Due to this we investigate tile shapes that make use of only one or two vertical tiles and vary the number of horizontal tiles to find out whether there is a shape that improves accuracy and power consumption.

To compare the accuracy of rectangular tiling to square tiling the MSE as a result of the tile width is plotted in Figure 5.24. Once again the plot only . Two rectangular tile heights are investigated, 1024 and 2048, which corresponds to two and one ADC per output element respectively. From this plot it is further evident that the ADC quantization is the main issue for square tiling.

For smaller tile widths the rectangular approach by far outperforms the square tiles. And since the effect of IR drop is consistently low for all rectangular tiles there is hardly any gain in terms of accuracy for changing the width of the tiles. In fact the square tile approach converges with the higher rectangular tile approach for tile width 512.

There is also a stark difference between using one or two vertical tiles. The shorter rectangular tiles experience less IR drop and has a persistently higher

accuracy despite employing more ADC. In conclusion, the accuracy loss for tiled crossbars are mainly decided by the number of ADC conversions unless the tile size induces large amount of IR drop. The accuracy loss for rectangular tiles with $T = 1024 \times 512$, with input = coefficient bitwidth = 8 and a 4-bit device, is around 1.6×10^{-2} MSE. Which is a marginal improvement compared to the 2×10^{-2} MSE for square tiles. As previously stated, rectangular tiling is not supported by the Hardware Analyzer. So while we leave more in depth evaluation for further research, we will discuss what potential energy savings can be obtained from rectangular tiling.

As established in Figures 5.22a and 5.22b, peripheral circuitry, specifically the ADCs, dominates area and energy consumption. A tile height of 1024 will require the same number of ADCs and the same ADC resolution as for square tiles of height and width 1024. Consequently, there is no energy gain to be found for this shape of tile. However, taller tiles of height 2048 require half the amount of ADCs compared to square and rectangular tiles of height 1024. This should translate to roughly half the ADC energy of tile size 1024 as illustrated by Table 5.5. In terms of accuracy the total accuracy loss is at 1.9×10^{-2} MSE. Note that this rudimentary analysis does not consider how the need for other peripherals change with the shape and size of the tiles. More in depth research will have to investigate the viability of differently shaped tiles.

5.4 Comparison to Conventional Hardware Architecture

There are many optimized implementations of the FFT. To contextualize our results, we compare our proposed Compute-In-Memory (CIM) architecture against the ultra-low voltage FFT core utilizing super-pipelining developed by Seok et al. [43]. Their highly optimized 1024-point complex FFT, fabricated in a 65 nm CMOS process and operating at an aggressive 0.27 V to minimize power, reported an energy consumption of 17.7 nJ per transform.

In comparison, our single-crossbar Symmetry implementation consumes approximately 8.65 nJ (as seen in Table 5.4), which is roughly half the energy. It must be noted that this is not a direct one-to-one comparison. Our design utilizes a smaller 20 nm technology node, and conventional digital hardware does not suffer from the analog accuracy degradation inherent to memristor implementations. However, the architectural advantages of the CIM design remain clear. Even when compared to a conventional hardware specifically optimized for ultra-low energy at the cost of speed, our approach avoids the von Neumann bottleneck entirely. This in-memory parallelism allows the Symmetry design to maintain competitive energy efficiency while delivering a massive improvement in performance, exceeding the conventional hardware's throughput by approximately 1,700%.

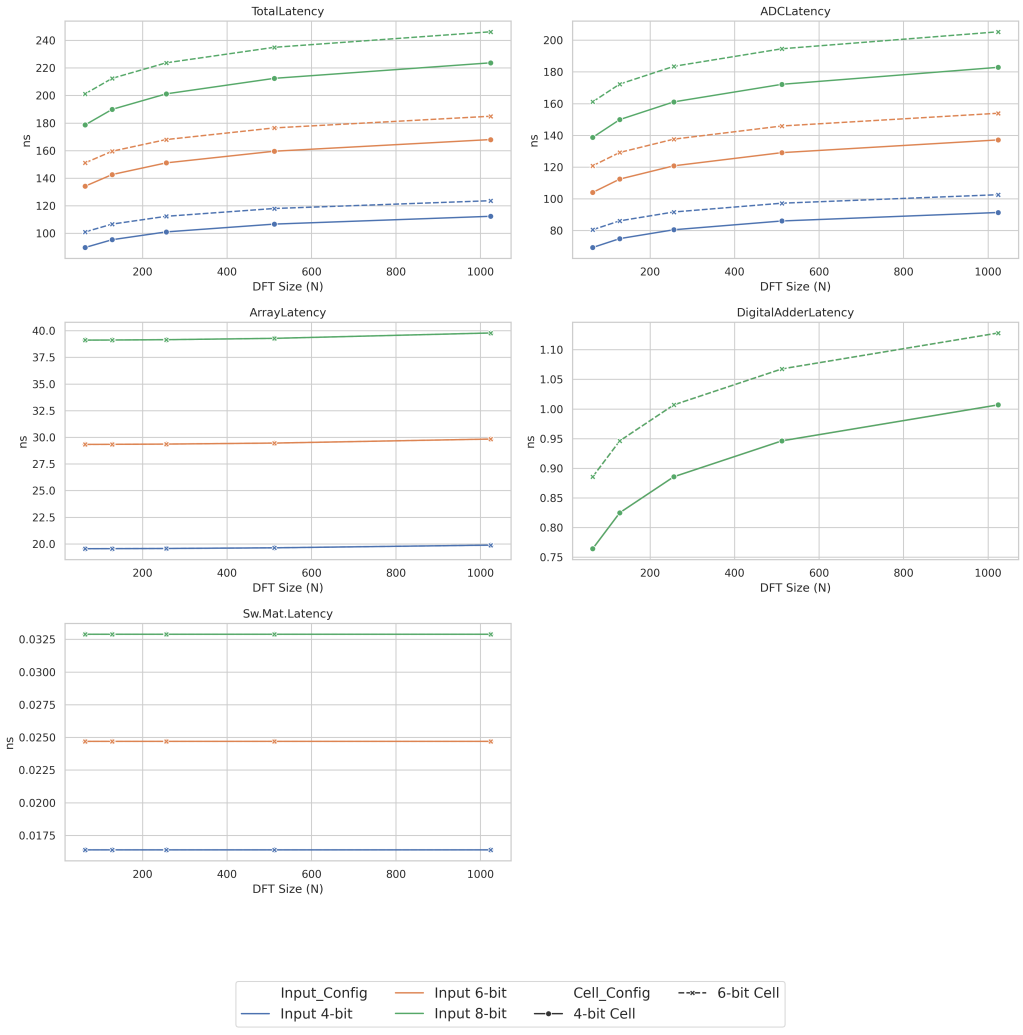
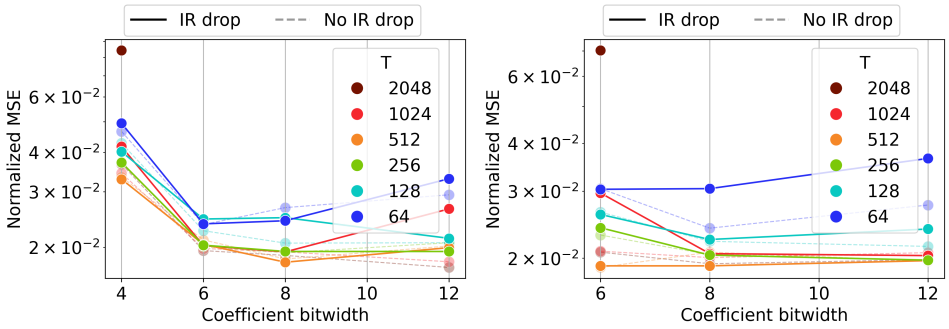


Figure 5.19: Latency for different N and input bitwidth. Results are grouped by input bitwidth and Cell Technology. The design is Symmetry for real only input.



(a) 4-bit device **(b)** 6-bit device

Figure 5.20: Normalized MSE with and without IR drop simulation for different square tile sizes and coefficient bitwidths. $T = 2048$ employs a single tile.

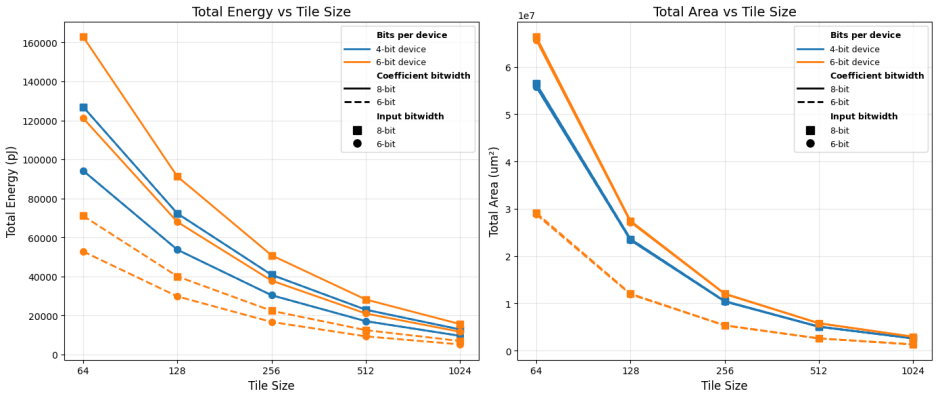
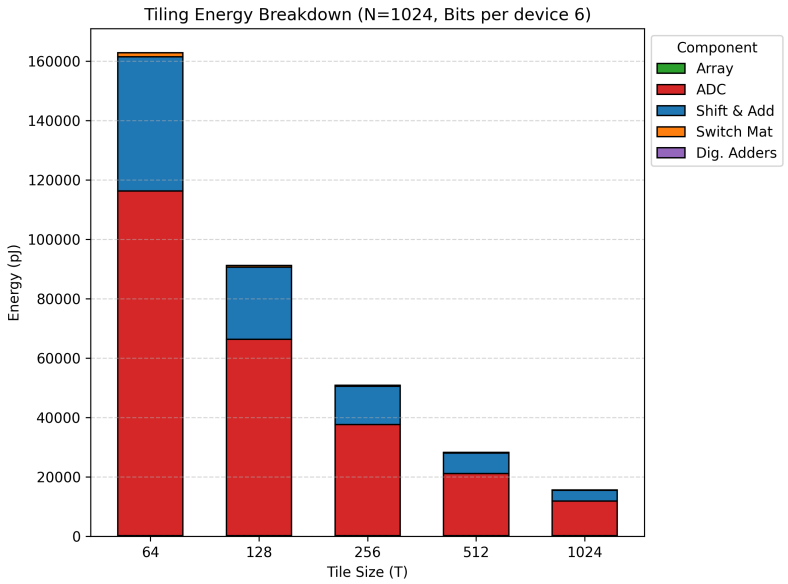
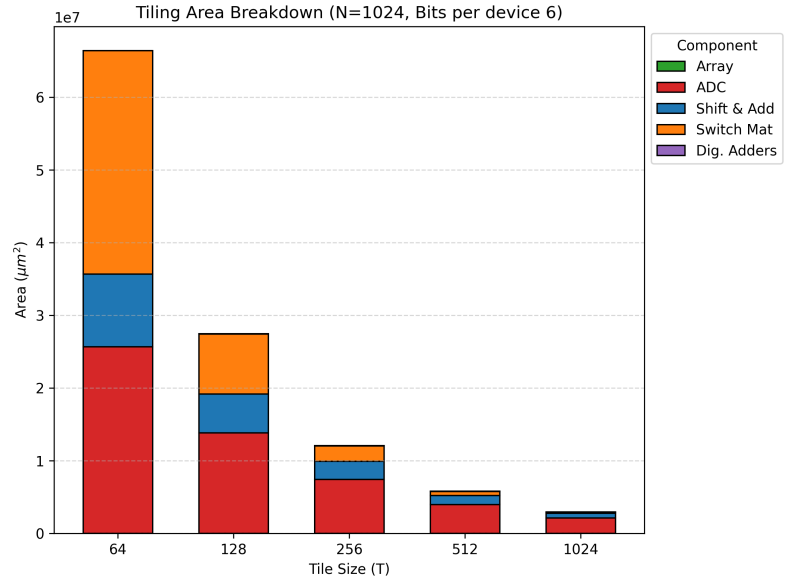


Figure 5.21: Energy consumption and area for different tile size, input and coefficient bitwidth. The design is Symmetry for real only input.



(a) Energy breakdown



(b) Area breakdown

Figure 5.22: Energy and area breakdown for different tile sizes. Evaluated using a 6-bit device, with input and coefficient bitwidth set to 8. The design is Symmetry for real only input.

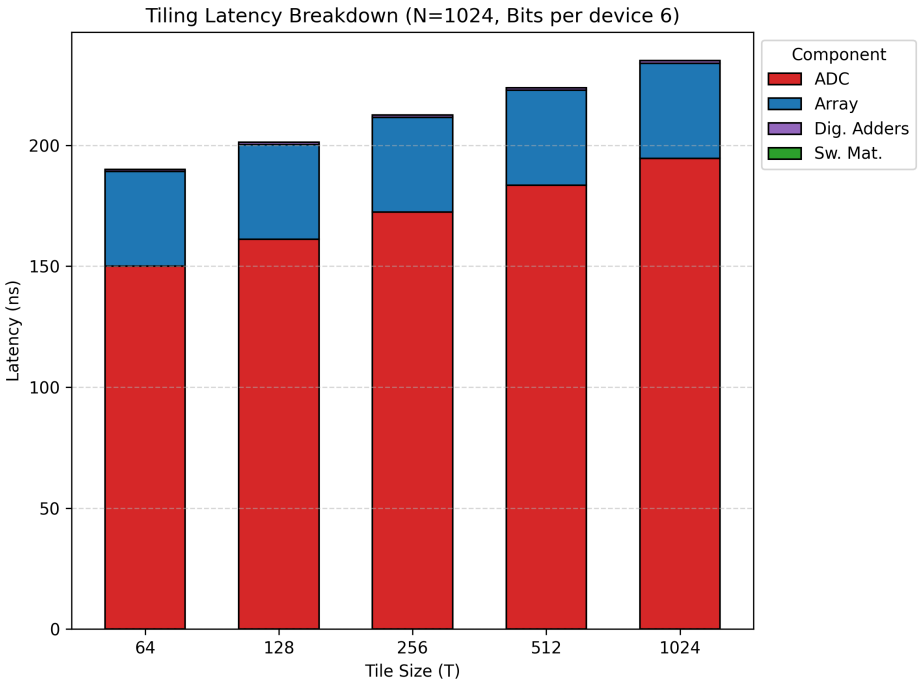


Figure 5.23: Latency breakdown for different tile size, 6-bit device, input and coefficient bitwidth set to 8. The design is Symmetry for real only input.

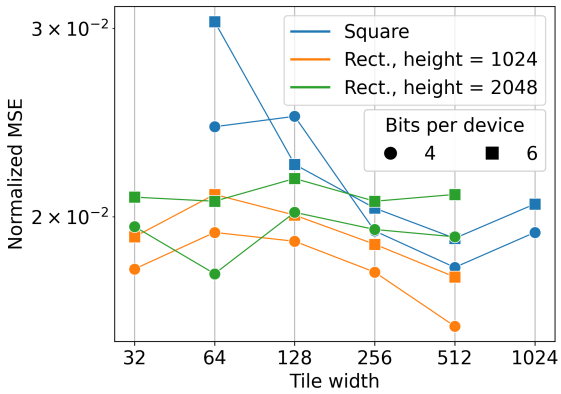


Figure 5.24: Accuracy loss as a result of bits per device, tile shape and size expressed in normalized MSE.

Chapter 6
Conclusions

In this chapter the findings of this report is presented along with a number of tradeoffs for balanced performance in terms of accuracy and PPA for small and large DFT. Additionally, the simulator choice is motivated.

6.1 Simulator Choice

The choice of using NeuroSim as a simulator was decided for the following reasons:

- Due to its suitability for configuring a DFT implementation.
- Being able to precisely configure what memristor device to use.
- Getting all relevant metrics, such as accuracy and PPA.

6.2 Trade-Offs

Depending on the application, energy consumption might be more important than accuracy, or perhaps latency could be the bottleneck in the system. In the discussion these performance metrics are discussed separately. But in reality tradeoffs between accuracy, energy consumption and latency are required for a feasible system. In this section we aim to find a combination of parameters which provides balanced performance in these three aspects. First the designs and system parameters are evaluated for small DFT. Then the best design is assessed with tilling for large DFT.

6.2.1 Small DFT and Design Evaluation

For small DFT, the three designs are found to in general have very similar accuracy with the Symmetry design outperforming the other two slightly for complex input as found in Section 5.2.1. In terms of PPA the Symmetry design is also found to be slightly less power hungry, and to have the smallest area. The latency is essentially the same across the designs with the exception of complex input for Merged design which is marginally less. In conclusion, the Symmetry design is more optimal or on par with the other designs with regards to accuracy and PPA.

As such the conclusions and tradeoff suggestions for the other design metrics are made with the Symmetry design in mind, unless otherwise specified.

From the analysis in Section 5.1.1 it is evident that increasing the coefficient bitwidth for small DFT beyond 6 bits has insignificant impact on the accuracy. And while increasing the coefficient bitwidth does not in itself increase the energy consumption or latency, coefficient bitwidths larger than the bits per device will introduce more devices in the crossbar due to coefficient slicing. Meaning that there is a benefit to choosing the larger device bitwidth which can represent the entire 6 bits of weights in a single device. However, more programmable bits per device will increase the need for ADC as evident by (4.16).

To compare the tradeoff between increasing bits per device (and in turn the ADC resolution) versus increasing number of devices (due to coefficient slicing) with constant coefficient bitwidth in terms of energy we compare the configuration A with a 6-bit device, and 6 bits per weight with configuration B with a 4-bit device, and 6 bits per weight. Despite the increased energy in the ADC due to larger device bitwidth, configuration A slightly outperforms B as seen in Figure 5.6.

Now that the optimal bits per device and coefficient bitwidths have been decided we move to investigate the optimal input bitwidth. Due to the bit-serial nature of the input, increasing the input bitwidth will increase the energy consumption linearly. So while the input bitwidth will seriously improve the accuracy, the additional energy consumption might not be justified. In Figure 6.1 the energy consumption and accuracy for the Symmetry design with configuration A is compared to find a tradeoff for input bitwidth. We can see that the accuracy improves less when increasing from 6 to 8-bit input, while the energy is significantly increased. Depending on the system requirements more or less bits can be used in the input to optimize for either power consumption or accuracy.

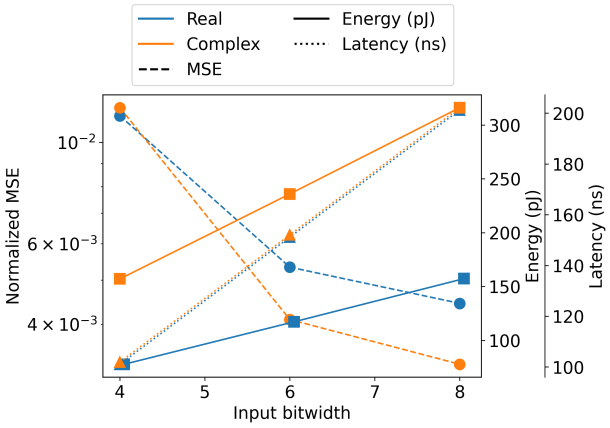


Figure 6.1: Impact of input bitwidth on accuracy, energy consumption, and latency for the Symmetry architecture with a 6-bit device and coefficient bitwidth.

In conclusion, the Symmetry design overall shows the best properties compared

to the designs for both real and complex input. For small DFT 6 bits is sufficient to represent the twiddle coefficients. And for 6 bits per weight, configurations with 6-bit devices outperform 4-bit devices. Finally, the accuracy is minimally improved by increasing the input bitwidth beyond 6 bits, where we find a good tradeoff between smaller accuracy loss and large energy consumption. Table 6.1 summarizes the performance metrics of this configuration of parameters.

Table 6.1: Performance summary for the Symmetry design using a 6-bit device and 6-bit input and coefficient bitwidth.

Metric	Real Input	Complex Input
MSE	4×10^{-3}	8×10^{-3}
Energy (pJ)	131	264
Area (μm^2)	54 126	110 312
Latency (ns)	160	160

Lastly, IR drop was found to be negligible for the crossbar sizes employed for small DFT. And the conductance drift found to not impact the accuracy one year after programming the devices. The peripheral circuitry, especially the ADC, are identified as largest contributors to area, latency and energy consumption.

6.2.2 Large DFT and Tiling Evaluation

For the large DFT, around ~ 1 MSE is attributed to large IR drop as seen in Figure 5.11. If higher accuracy is desired, the 1024 point DFT cannot be viably implemented with a single crossbar using the Symmetry design due to the IR drop. While the IR drop does not noticeably affect the energy consumption, it does have a pronounced impact on the accuracy. Tiling can mitigate this issue. Using square tiles the best accuracy can be obtained with 4-bit deviced and input and coefficient bitwidth $= 8$ and $T = 1024$, as presented in Figure 5.20. However, comparable accuracy can be achieved with $T = 512$ and a 6-bit device.

By consulting Figure 5.21 we find that 4-bit devices and $T = 1024$ slightly outperform any other combinations with respect to both area and energy. In terms of latency the trend is reversed. Larger tile sizes require higher resolution ADC which lead to larger latency. Even so, this increase is logarithmic to the size of the tile. Figure 6.2 highlights this relationship. So while the latency is slightly increased with large tile sizes, the energy and area is significantly smaller. In conclusion, unless the latency is the most critical aspect of the system, Square tiles with $T = 1024$ is the best option in terms of both PPA and accuracy. And in applications where latency is critical, such as in wireless communication, Figure 5.21 can be consulted to find the parameters to suit the system requirements.

The two bottlenecks for managing performance in tiled crossbar arrays are IR drop and ADC quantization. Rectangular tiles can potentially mitigate the ADC quantization error by requiring fewer tiles in the horizontal direction. In Figure 5.24 this speculation is confirmed. The accuracy showcased by rectangular tiles with height 1024 consistently surpass square tiles thanks to fewer ADC conversions. Yet further increasing the tile height to 2048, thereby requiring only a single

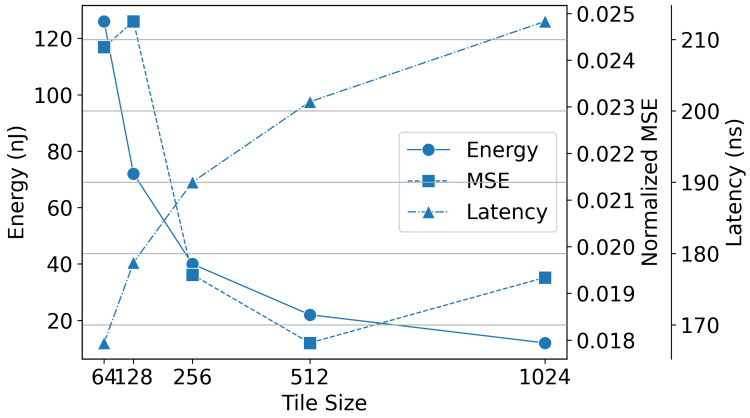


Figure 6.2: Impact of square tile size on accuracy, energy consumption, and latency for the Symmetry architecture with a 4-bit device and 8-bit coefficient and input bitwidth.

ADC conversion, does not lead to additional improvement. Instead, accuracy degrades as a result of the increased IR drop within the crossbar. However, since tiles of height 2048 require half the amount of ADCs there is a large potential energy saving to be found for the taller tiles. But since the HA does not support rectangular tiles there might be other considerations that complicates the usage of the taller tiles. The accuracy is slightly worse than the other tile configuration at 1.9×10^{-2}. So if an accuracy degradation of this magnitude is acceptable, further investigations for tiles of size 2048 might prove useful.

Table 6.2: Real input large DFT with and without tiling compared to small DFT.

	MSE	**Energy (pJ)**	**Area (mm^2)**	**Latency (ns)**
Small DFT	4×10^{-3}	131	0.05	160
Large DFT	1	8 649	1.52	246
Square tiles	2×10^{-2}	15 638	2.95	235
Rectangular tiles	1.2×10^{-2}			

6.3 Key Metrics Evaluation

The previous section outlines a number of design choices that creates a good balance between accuracy and energy consumption, area and latency. By tweaking some key metrics, the performance in the system can be tailored to the application specific requirements. This section summarizes our findings for how key metrics affect the performance of the DFT.

- **Input bitwidth** - due to the bit-serial nature of the input in the proposed design the input bitwidth has a proportional impact on the energy consumption and latency. Decreasing the input bit width will degrade the accuracy but have potential to lower latency and create energy savings.
- **Coefficient bitwidth** - the required coefficient bitwidth is constrained by the size of the DFT, as larger transforms require higher precision to distinguish between closely spaced twiddle factors. For $N = 64$ and $N = 1024$, 6 and 8 bits were found to be sufficient to represent the twiddle coefficients.
- **ADC resolution** - larger ADCs consume more energy. If the inputs are biased, the ADC resolution could potentially be lowered with maintained accuracy.
- **Crossbar size** - crossbars of width and height = 1024 or larger experience non-negligible degradation of accuracy due to IR drop. Larger crossbars typically require higher ADC resolution.
- **Tile shape** - few large tiles require less peripheral circuitry which has a positive effect on accuracy and energy consumption. On the other hand, latency is increased with larger arrays.
- **Device** - memristor devices with high resistance ensures lower IR drop. Low conductance variability and drift, high separability between states and near-linear behavior are other properties that enable accurate performance.
- **Bits per device** - the number of programmable states used in a memristor is an important factor in deciding the ADC resolution. Furthermore, the device bitwidth determines the number of devices used per weight which increases the size of the crossbar.

Chapter 7
Further Work

In this chapter some potential further investigations are suggested. We propose potential architectural techniques for minimizing inaccuracies due to device non-idealities. Then a number of future additions to address the limitations of the chosen simulator are suggested along with some areas for improved fidelity in the simulations.

7.1 Encoding Schemes and Mapping of Inputs and Weights

In this work, the mapping of coefficients to devices is optimized from the LSB to MSB method, as outlined in Section 2.3.4, to improve robustness towards device non-idealities. Other more involved mapping techniques could potentially be used to further minimize the impact of device imperfections. In [6], the proposed mapping scheme effectively eliminates the bit error rate. Investigating this or similar mapping schemes could potentially enhance the accuracy of the DFT.

The mapping of input to voltages could also be further explored. If the input distribution is skewed, densely populated input regions could be assigned a wider voltage range to improve separability and resolution. Differential representation was selected for encoding in this project in part because of the inherent robustness to device non-idealities. This design choice requires larger crossbars. Other methods of encoding could be explored to find out if similar accuracy can be achieved but with less energy consumption. In [6], the offset representation was implemented with 18% less area and 57% less energy consumption compared to the differential representation. Moreover, the usage of dummy columns to deal with non-ideal LRS was removed from the BS due to incompatibility with one-dimensional input. Another interesting investigation would be to compare implementations with and without the dummy column, especially in combination with the different encoding schemes.

7.2 Simulator Extensions

7.2.1 Analog Components

In the current version of NeuroSim, there are no analog components capable of connecting crossbars in the analog domain. An example of such a component would

be a current-to-voltage converter, like a transimpedance amplifier (TIA). Future work could implement these components to improve PPA metrics and enable fully analog DFT architectures, such as the one proposed by Mahdavi [1].

A critical extension for future versions of NeuroSim is supporting analog-domain differential subtraction. Currently, the differential subtraction of (I^+ & I^-) must be reconstructed using digital adders. If NeuroSim is expanded to allow connecting bitlines before the ADC, this computation could be performed in the analog domain. This upgrade would effectively halve the required number of ADCs and completely eliminate the need for digital adders in the merged design, as well as the symmetry design for real inputs. Implementing this feature is essential for future research to simulate fully optimized custom circuits, thereby bypassing the pessimistic area and energy overheads caused by digital summation.

7.2.2 Rectangular Tiling Support

As noted in our PPA evaluation, the Hardware Analyzer natively supports only square tiling. When simulating rectangular tiles, the floorplanning algorithm incorrectly instantiates and routes the non-square dimensions. Updating NeuroSim to support rectangular tiling would allow for direct hardware simulations of topologies that minimize vertical crossbar fragmentation, which would significantly reduce the ADC area overhead.

7.2.3 Multi-Bit Input Support

Currently, the Hardware Analyzer of NeuroSim is strictly limited to 1-bit per cycle input bitwidth. Due to this bit-serial processing, increasing input bitwidth leads to a linear increase in latency and dynamic energy. Adding support for multi-bit input processing, utilizing multi-bit DACs, would drastically reduce latency. For example, processing 2 or 4 bits per cycle would allow us to perform a tradeoff analysis to determine exactly how many bits can be processed concurrently before the noise causes unacceptable accuracy degradation. There are also some drawbacks to this method since multi-bit DACs are also quite costly in terms of PPA. Additionally, multi-bit inputs will increase the required ADC resolution. In order to thoroughly explore this design choice, the ADC and DAC energies must be properly modeled.

7.3 Improve Simulation Fidelity

Despite the extensions introduced in this work, the NeuroSim simulator exhibits inherent limitations in modeling accuracy. To improve simulation fidelity, several areas for further development have been identified.

- **ADC energy modeling**: The energy model for ADC in NeuroSim is simplified and assumes linear increase in energy consumption as the resolutions grows. This is a simplification as larger resolutions, beyond 8 bits, can have a larger than linear increase in energy consumption. Improved ADC modeling is essential for informed tradeoff analysis and may support the use of tiling to lower ADC resolution requirements.

- **ADC error**: NeuroSim currently only support quantization and truncation in the device expert mode. Incorporating noise data from simulations or real life measurements could promote more realistic simulation results.
- **Dynamic IR drop**: The chosen simulator setup only supports simulation of static IR drop. Extending it to include dynamic IR drop simulation is a natural next step toward higher simulation fidelity.
- **Resistive loss and delay modeling**: Our current models for calculating resistive loss and the Elmore delay, are based on strong assumptions. To improve the accuracy of these values, SPICE simulations would be needed to create more accurate models for our designs.

Bibliography

[1] M. Mahdavi, "Fully parallel and reconfigurable realization of dft/idft using in-memory computing," in *2023 10th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 2023, pp. 1–8. DOI: 10.1109/WINCOM59760.2023.10322956.

[2] ITU-R, "Recommendation ITU-R M.2083-0: IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond," International Telecommunication Union (ITU), Tech. Rep., Sep. 2015. [Online]. Available: https://www.itu.int/rec/R-REC-M.2083/en.

[3] M. Shafi et al., "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1201–1221, 2017.

[4] Y. Yi, B. Kim, H. Seo, H. Ko, and K. Kim, *Method and apparatus for handling DC subcarrier in NR carrier in wireless communication system*, WO Patent 2018021676 A1, Feb. 2018.

[5] A. Bourenane, E. Paolini, N. Andriolli, and L. Valcarenghi, "Reducing the energy footprint of 5G: A gNB on MPSoC with low-power FFT acceleration," *IEEE Open Journal of the Communications Society*, vol. 7, pp. 1889–1900, 2026.

[6] J. Wen et al., "Towards reliable and energy-efficient rram based discrete fourier transform accelerator," in *2024 Design, Automation & Test in Europe Conference Exhibition (DATE)*, 2024, pp. 1–6. DOI: 10.23919/DATE58400.2024.10546709.

[7] H. Zhao et al., "Energy-efficient high-fidelity image reconstruction with memristor arrays for medical diagnosis," *Nature Communications*, vol. 14, no. 1, p. 2276, 2023. DOI: 10.1038/s41467-023-38021-7.

[8] G. Yuan et al., "Memristor crossbar-based ultra-efficient next-generation baseband processors," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 1121–1124. DOI: 10.1109/MWSCAS.2017.8053125.

[9] R. Cai, A. Ren, Y. Wang, and B. Yuan, "Memristor-based discrete fourier transform for improving performance and energy efficiency," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2016, pp. 643–648. DOI: 10.1109/ISVLSI.2016.124.

[10] M. Hu and J. P. Strachan, "Accelerating discrete fourier transforms with dot-product engine," in *2016 IEEE International Conference on Rebooting Computing (ICRC)*, 2016, pp. 1–5. DOI: 10.1109/ICRC.2016.7738682.

[11] Q. Hong, B. He, Z. Zhang, P. Xiao, S. Du, and J. Zhang, "Circuit design and application of discrete cosine transform based on memristor," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 13, no. 2, pp. 502–513, 2023. DOI: 10.1109/JETCAS.2023.3243569.

[12] V. Ingle and J. Proakis, *Digital Signal Processing Using Matlab*. Brooks/Cole Publishing, 2000.

[13] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971. DOI: 10.1109/TCT.1971.1083337.

[14] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008, ISSN: 1476-4687. DOI: 10.1038/nature06932. [Online]. Available: https://doi.org/10.1038/nature06932.

[15] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based material implication (imply) logic: Design principles and methodologies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 10, pp. 2054–2066, 2014. DOI: 10.1109/TVLSI.2013.2282132.

[16] S. Kvatinsky et al., "Magic—memristor-aided logic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 11, pp. 895–899, 2014. DOI: 10.1109/TCSII.2014.2357292.

[17] L. Chua and S. M. Kang, "Memristive devices and systems," *Proceedings of the IEEE*, vol. 64, no. 2, pp. 209–223, 1976. DOI: 10.1109/PROC.1976.10092.

[18] Y. V. Pershin and M. Di Ventra, "Practical approach to programmable analog circuits with memristors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 8, pp. 1857–1864, 2010. DOI: 10.1109/TCSI.2009.2038539.

[19] C. Li et al., "Efficient and self-adaptive in-situ learning in multilayer memristor neural networks," *Nature Communications*, vol. 9, no. 1, p. 2385, 2018. DOI: 10.1038/s41467-018-04484-2. [Online]. Available: https://doi.org/10.1038/s41467-018-04484-2.

[20] H. Veluri, U. Chand, C.-K. Chen, and A. V.-Y. Thean, "A low-latency dnn accelerator enabled by dft-based convolution execution within crossbar arrays," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 1, pp. 1015–1028, 2025. DOI: 10.1109/TNNLS.2023.3327122.

[21] A. Mehonic et al., "Roadmap to neuromorphic computing with emerging technologies," *APL Materials*, vol. 12, no. 10, p. 109 201, Oct. 2024, ISSN: 2166-532X. DOI: 10.1063/5.0179424. eprint: https://pubs.aip.org/aip/apm/article-pdf/doi/10.1063/5.0179424/20213999/109201_1_5.0179424.pdf. [Online]. Available: https://doi.org/10.1063/5.0179424.

[22] M. Borg, C. Papadopoulos, A. Guerin, R. Athle, and S. Bastani, "Prospects of analog in-memory computing using ferroelectric tunnel junctions," *Neuromorphic Computing and Engineering*, vol. 5, no. 2, p. 024 006, May 2025, Focus issue on Devices, Circuits and Systems for Unconventional Computing. DOI: 10.1088/2634-4386/add0d9. [Online]. Available: https://doi.org/10.1088/2634-4386/add0d9.

[23] R. S. Khan, A. Hasan Talukder, F. Dirisaglik, A. Gokirmak, and H. Silva, "Stopping resistance drift in phase change memory cells," in *2020 Device Research Conference (DRC)*, 2020, pp. 1–2. DOI: 10.1109/DRC50226.2020.9135147.

[24] R. Dittmann, S. Menzel, and R. Waser, "Nanoionic memristive phenomena in metal oxides: The valence change mechanism," *Advances in Physics*, vol. 70, no. 2, pp. 155–349, 2021. DOI: 10.1080/00018732.2022.2084006. eprint: https://doi.org/10.1080/00018732.2022.2084006. [Online]. Available: https://doi.org/10.1080/00018732.2022.2084006.

[25] C. Li et al., "Large memristor crossbars for analog computing," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–4. DOI: 10.1109/ISCAS.2018.8351877.

[26] J. Wen, A. Baroni, E. Perez, M. Ulbricht, C. Wenger, and M. Krstic, "Evaluating read disturb effect on rram based ai accelerator with multilevel states and input voltages," in *2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2022, pp. 1–6. DOI: 10.1109/DFT56152.2022.9962345.

[27] T. P. Xiao et al., "On the accuracy of analog neural network inference accelerators," *IEEE Circuits and Systems Magazine*, vol. 22, no. 4, pp. 26–48, 2022. DOI: 10.1109/MCAS.2022.3214409.

[28] R. Athle and M. Borg, "Ferroelectric tunnel junction memristors for in-memory computing accelerators," *Advanced Intelligent Systems*, vol. 6, no. 3, p. 2 300 554, 2024. DOI: https://doi.org/10.1002/aisy.202300554. eprint: https://advanced.onlinelibrary.wiley.com/doi/pdf/10.1002/aisy.202300554. [Online]. Available: https://advanced.onlinelibrary.wiley.com/doi/abs/10.1002/aisy.202300554.

[29] S. K. Nithin, G. Shanmugam, and S. Chandrasekar, "Dynamic voltage (ir) drop analysis and design closure: Issues and challenges," in *2010 11th International Symposium on Quality Electronic Design (ISQED)*, 2010, pp. 611–617. DOI: 10.1109/ISQED.2010.5450515.

[30] S. Yu, H. Jiang, S. Huang, X. Peng, and A. Lu, "Compute-in-memory chips for deep learning: Recent trends and prospects," *IEEE Circuits and Systems Magazine*, vol. 21, no. 3, pp. 31–56, 2021. DOI: 10.1109/MCAS.2021.3092533.

[31] A. Shafiee et al., "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 14–26. DOI: 10.1109/ISCA.2016.12.

[32] T. Andrulis, J. S. Emer, and V. Sze, "CiMLoop: A flexible, accurate, and fast compute-in-memory modeling tool," in *2024 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2024, pp. 10–23. DOI: 10.1109/ISPASS61541.2024.00012.

[33] B. Feinberg, T. P. Xiao, C. J. Brinker, C. H. Bennett, M. J. Marinella, and S. Agarwal, "Crosssim: Accuracy simulation of analog in-memory computing," [Online]. Available: https://github.com/sandialabs/cross-sim.

[34] E. R. Keiter et al., *Xyce™ Parallel Electronic Simulator Users' Guide*, Version 7.10, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, USA, May 2025. [Online]. Available: https://xyce.sandia.gov/.

[35] J. Read, M.-Y. Lee, W.-H. Huang, Y.-C. Luo, A. Lu, and S. Yu, "Neurosim v1.5: Improved software backbone for benchmarking compute-in-memory accelerators with device and circuit-level non-idealities," May 2025. DOI: 10.48550/arXiv.2505.02314.

[36] J. Lee, A. Lu, W. Li, and S. Yu, "Neurosim v1.4: Extending technology support for digital compute-in-memory toward 1nm node," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 71, no. 4, pp. 1733–1744, 2024. DOI: 10.1109/TCSI.2024.3362822.

[37] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "Dnn+neurosim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," in *2019 IEEE International Electron Devices Meeting (IEDM)*, 2019, pp. 32.5.1–32.5.4. DOI: 10.1109/IEDM19573.2019.8993491.

[38] P.-Y. Chen, X. Peng, and S. Yu, "Neurosim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3067–3080, 2018. DOI: 10.1109/TCAD.2018.2789723.

[39] D. Joksas and A. Mehonic, "Badcrossbar: A python tool for computing and plotting currents and voltages in passive crossbar arrays," *SoftwareX*, vol. 12, p. 100 617, 2020, ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2020.100617. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711020303307.

[40] X. Peng et al., *User manual of dnn+neurosim framework v1.5*, Developers: Xiaochen Peng, Shanshi Huang, Anni Lu, Junmo Lee, James Read, and Ming-Yen Lee. PI: Prof. Shimeng Yu, Georgia Institute of Technology, May 2025.

[41] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd. Pearson, 2010.

[42] N. H. E. Weste and D. M. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th. Boston: Addison-Wesley, 2011, ch. 6, Sections 6.2–6.4.

[43] M. Seok, D. Jeon, C. Chakrabarti, D. Blaauw, and D. Sylvester, "A 0.27V 30MHz 17.7nJ/transform 1024-pt complex FFT core with super-pipelining," in *2011 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, Feb. 2011, pp. 342–344.