An Ultra-Low Power In-Memory Computing Cell for Binarized Neural Networks

Mr. Philip Chennakudy Jose

A thesis submitted for the degree of

Master of Research
at
Western Sydney University

August 2020

Supervisors:
Dr. Runchun Mark Wang
Dr. Upul Gunawardana

List of Contents

List of Figures

List of Tables

Abstract

Deep Neural Networks (DNN's) are widely used in many artificial intelligence applications such as image classification and image recognition. Data movement in DNN's results in increased power consumption. The primary reason behind the energy-expensive data movement in DNN's is due to the conventional Von Neuman architecture in which computing unit and memory are physically separated. To address the issue of energy-expensive data movement in DNN's in-memory computing schemes are proposed in the literature. The fundamental principle behind in-memory computing is to enable the vector computations closer to the memory. In-memory computing schemes based on CMOS technologies are of great importance nowadays due to the ease of massive production and commercialization. However, many of the proposed in-memory computing schemes suffer from power and performance degradation. Besides, some of them are capable of reducing power consumption only to a small extent and this requires sacrificing the overall signal to noise ratio (SNR).

This thesis discusses an efficient In-Memory Computing (IMC) cell for Binarized Neural Networks (BNNs). Moreover, IMC cell was modelled using the simplest current computing method. In this thesis, the developed IMC cell is a practical solution to the energy-expensive data movement within the BNNs. A 4-bit Digital to Analog Converter (DAC) is designed and simulated using 130nm CMOS process. Using the 4-bit DAC the functionality of IMC scheme for BNNs is demonstrated. The optimised 4-bit DAC shows that it is a powerful IMC method for BNNs. The results presented in this thesis show this approach of IMC is capable of accurately performing dot operation between the input activations and the weights. Furthermore, 4-bit DAC provides a 4-bit weight precision, which provides an effective means to improve the overall accuracy.

Acknowledgement

I would like to thank everyone who has been supportive throughout my Master of Research study at Western Sydney University. Firstly, I would like to thank my supervisor Dr Runchun Mark Wang for his limitless support throughout my studies. Without his support I would not have been able to complete my thesis. I also like to thank Dr Wang for introducing me to this wonderful area of research and for the training you have given to me to advance in this field.

Secondly, I would like to thank my co-supervisor Dr Upul Gunawardana for his constant support and encouragement. I also would like to thank Dr Andrew Nicholson from ICNS, for your support in Cadence setup and troubleshooting. I would also like to thank everyone from ICNS.

I would also like to thank my parents for the support and encouragement throughout my studies.

Finally, I would like to thank my wife Jency for her support and her tolerance throughout my studies.

Statement of Authentication

The work presented in this thesis is, to the best of my knowledge and belief, original except acknowledged in the text.

I hereby declare that I have not submitted this material either in full or in part, for a degree at this or any other institution.

Mr. Philip Chennakudy Jose August 24, 2020

Chapter 1 : Introduction

The invention of electricity has transformed countless industries such as transportation, manufacturing, health care, communication and more. Today, Artificial Intelligence (AI) is regarded as the new electricity and AI has already brought an equally big transformation. AI applications are increasing day by day, and the driving force behind the rapid growth of AI are mainly because of the recent developments that have happened in the field of Deep Neural Networks (DNNs) [1].

DNNs are regarded as the backbone for the vast majority of artificial intelligence (AI) applications that exist today. DNNs provide surprising accuracy for many of the AI applications such as computer vision tasks [2]–[4], self-driving cars [5], image recognition [4], speech recognition [6], very complex games [7], [8], and robotics [9]. In most of these applications DNNs are outperforming human accuracy. DNNs have a special ability to automatically extract the useful information that is needed for future predictions or to make a decision. This is different from Machine learning (ML) algorithms, which typically tries to define a set of rules in the data sets which are usually hand-engineered [10].

Applications involving DNNs can provide high accuracy, However, increased computational complexity is one of the major drawbacks of DNN. To classify a 50K pixel image we need 4-50 Giga operations if we employ image recognition using DNNs. There is a significant growth of computational complexity over the years and this has resulted in huge power consumption. The increased power consumption is mainly due to the conventional von-Neumann architecture in which the processing unit (PU) and the memory are physically separated. Whenever a computation is performed data will be taken from memory to PU creating more power consumption. Data movement is regarded as the primary cause for higher energy consumption in DNNs.

General-purpose compute engines such as Graphics Processing Units (GPUs), are widely used in DNNs for efficient processing. Over the past years, there is an increase in research interest providing more specialised accelerations of the DNN computations. Many of the recent works are focused on bringing the memory closer to the computation module or performing the computations in the memory itself. Another improvement happened is to perform computations inside the sensors where the primary data for processing are collected [11], [12].

One way to reduce the computational and storage complexity of DNNs is to reduce precision of weights and activations[13]. Binarized Neural Networks (BNNs) uses quantization to reduce the precision of weights and activations [14].

The main objective of this thesis is to demonstrate an in-memory computing (IMC) scheme for BNNs. The thesis aims to design an IMC cell for BNNs using current based computing strategy within the 6T-SRAM (Static Random Access Memory). To achieve this aim 4-bit DAC is developed using 130nm CMOS process. The working of the 4-bit DAC during the read phase of 6T-SRAM is investigated thoroughly. In addition, the performance of the 4-bit DAC is assessed using various simulations.

1.1 : Background of DNNs

DNNs are also referred to as Deep Learning and are incredibly a powerful tool. It is a part of AI. DNNs help as to create intelligent machines, which have the special ability to achieve goals or to complete tasks like humans do [10]. **Figure 1.1** depicts how AI and Deep Learning are inter-related.

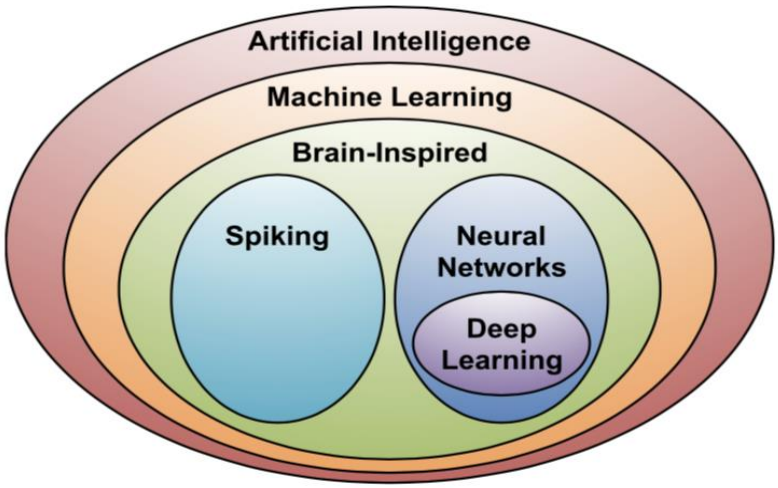


Figure 1.1. Relationship between Deep Learning and AI (Adopted from [13])

Intelligence is defined as the ability to make future decision's by processing the information. Furthermore, Artificial Intelligence (AI) are algorithms capable of making future decision's by processing the information's [10]. Machine Learning (ML) is a subset of AI that specifically focus on teaching an algorithm, how to process information without being programmed, to complete the task at hand. This means that once a programme is created, it will be able to perform intelligent activities, without being programmed again. ML algorithms try to define sets of rules and features in data, which are usually hand-engineered. These algorithms also overcome the laborious process of creating distinct programmes that could solve individual problems in a domain by creating a single ML algorithm, that needs to learn by a process called training, which could address each new problem [13].

Brain-inspired computation is an area within the field of Machine Learning. The brain is the best intelligent machine known to mankind for learning and solving problems. Brain-inspired computations are simply programme or algorithms that take the key aspects of how the brain works. It all depends upon how one understands how the brain works [13].

Although a lot of research is undergoing to explore how the brain works, neurons are considered to be the important computational element inside a brain. The neurons are connected to a number of input elements known as dendrites and connected to output elements known as axons. Axons are connected to the dendrites of other neurons and the connection between axon and dendrites is called a synapse [15]. **Figure 1.2** depicts the functionality of a single neuron.

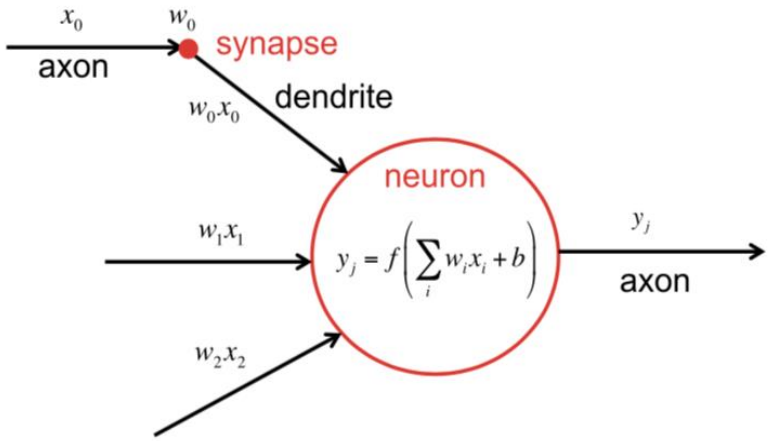


Figure 1.2. Functionality of a single neuron (adopted from [15])

Synapse has the ability to scale the signal (xi) while it crosses the junction. Weights (wi) are usually used to denote the scaling factor and is believed that, the brain learns from changes associated with these weights. Different weights could produce different outputs. In ML, learning refers to the adjustment of weights, in accordance with the stimulus, while the program (Brain) doesn't change. These are the key inspiration for the ML style computation.

Spiking computing is a subarea within brain-inspired computing. It is from the fact that, communication between axons and dendrites are through spike-like pulses, and the information that is exchanged does not only depends on spike amplitude, instead it depends on the time duration between the pulse arrivals. IBM true North [8], is an example of the spiking computation. Another subfield of the brain-inspired computation is the neural networks, which is the main focus of this thesis.

1.2 : Deep Neural Networks (DNNs)

The structural building block of DNNs is a single neuron, which is called a perceptron. The structure of a perceptron is depicted in **Figure 1.3**.

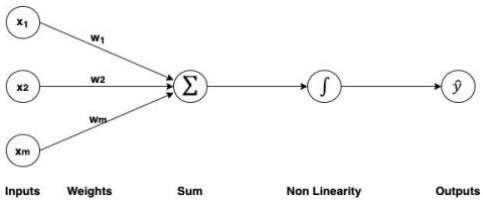


Figure 1.3. Structure of a perceptron (recreated from[10])

Inputs to the neurons $x1, x2, \ldots . xm$, each of these inputs have a corresponding weight $w1, w2, \ldots . . wm$. The primary computation within a neuron or the perceptron is the weighted sum of inputs and it resembles the scaling operation within a synapse. Furthermore, the result of the weighted sum is passed through a non-linear activation function. This will cause the perceptron to generate an output whenever inputs cross the defined threshold [10]. We will discuss some of these activation functions in **1.2.1**. The equation governing the operation of a perceptron is in Equation (1.1) (adopted from [10]).

$$\hat{y} = g(\sum_{i=1}^{m} x_i w_i) \qquad (1.1)$$

In equation 1.1 '$\hat{y}$' is the output of the perceptron, $'g'$ is the non-linear activation function, and '$x_i w_i'$ is the linear combination of inputs. Another term that we use with perceptron is the bias. As shown in **Figure 1.4**, the purpose of the bias term is to allow you to shift your activation function to the left or right regardless of your inputs. A computational neural network is depicted in **Figure 1.4**. The input layer of the neurons receives the input values and propagates them to the middle layer which is also known as 'hidden layer'. The final output of the network to the user will be the weighted sum from one or more hidden layers [10]. To match with the brain-inspired terminology with neurons, synapses are designated as weights and output of the neurons referred to as activations.

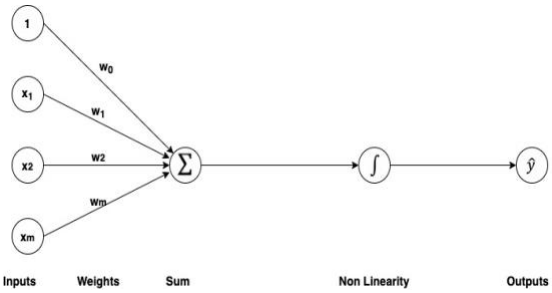


Figure 1.4. Computational neural network (Adopted from [10])

$$\hat{y} = g(\, w_0 + \sum_{i=1}^{m} x_i w_i) \qquad (1.2)$$

In the equation 1.2 (adopted from [10]) '$\hat{y}$' is the output of the perceptron, 'g' is the non-linear activation function, and '$x_i w_i'$ is the linear combination of inputs, which is same as equation 1.1. The extra parameter here is the 'w_0' which is the bias.

Deep learning is another area within neural networks, comprising more than three layers and are often termed as Deep Neural Networks (DNNs). In general, DNNs have more than a single hidden layer. Today, the number of network layers used in deep learning applications are more than a thousand. In this thesis DNNs are used to refer networks used in deep learning.

1.2.1 : Activation Functions

The sigmoid function defined in equation (1.3) (adopted from [10]) is very commonly used activation function. Furthermore, the function takes any real number as input on the x-axis and it transforms that real number into a scalar output between 0 and 1. More clearly the function has a bounded output between 0 and 1.

A good example for the application of sigmoid function includes computations involving probabilities. This is due to the fact that probability is always bounded between 0 and 1. Moreover, the sigmoid function is really useful when the constrain is to output a single number, so as to represent that number as a probability distribution.

$$g(z) = \sigma(z) = \frac{1}{1+e^{-z}} \qquad (1.3)$$

One problem associated with sigmoid function is the slow learning rate. Problems may occur when the slope of the function is nearly zero. The slow learning problem is addressed using a function called Rectified Linear Unit (ReLU). Equation 1.4 [10] represents a ReLU function.

$$g(z) = \max(0, z) \qquad (1.4)$$

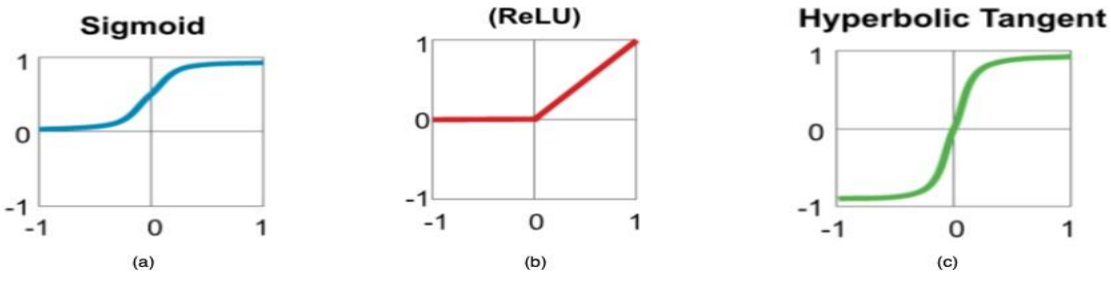


Figure 1.5. Commonly used non-linear activation function. (a) sigmoid function, (b) ReLU and (c) hyperbolic tangent [13]

Figure 1.5 depicts the commonly used non-linear activation function. **Figure 1.5.a** represents the plot of a sigmoid function characterized by equation (1.3). Whereas, **Figure 1.5.b** represents a ReLU function described by the equation (1.4). As shown in **Figure 1.5.b** the

gradient (slope) of ReLU is always '1' for all positive values, so that gradient is much less likely to shrink to zero. Furthermore, the slope of the line is zero in the left-hand side. Finally, **Figure 1.5.c** represents a hyperbolic tangent non-linear activation function characterized by equation (1.5) [10].

$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \qquad (1.5)$$

The objective of activation functions is to introduce non-linearities into the network or data. This objective is very important in real life because in real life almost all the data is non-linear. Furthermore, non-linearity allows as to approximate arbitrary complex functions by introducing non-linearities into decision boundaries. This makes neural network so powerful.

1.3 : Building Deep-Neural Networks (DNN's) from Perceptron's

Figure 1.6 depicts a simplified view of a computational neural network with bias and weight labels removed. 'z' represents the output of the dot product. i.e., element-wise multiplication of inputs with weights of the network. This operation is expressed in equation (1.6) (adopted from [10]). The output 'z' is feed into the non-linear activation function to produce the output of the network designated as 'y'.

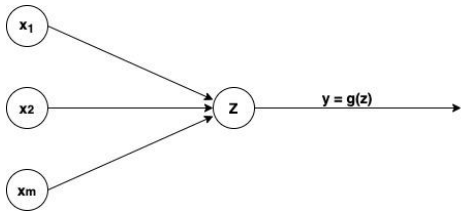


Figure 1.6. Simplified view of a computational Neural Network [10]

$$z = w_0 + \sum_{j=1}^{m} x_j w_j \qquad (1.6)$$

A multi-output perceptron is depicted in **Figure 1.7**. This is created by adding another more perceptron into the fundamental structure shown in **Figure 1.6**. The multi-output perceptron shown in **Figure 1.7** has two outputs $y_1$and y_2. One peculiarity of this structure is the presence of dense layers, since all the inputs are densely connected to all the outputs [10].

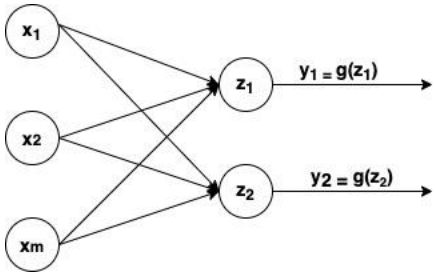


Figure 1.7. Multi output perceptron [10]

A single layer neural network is shown in **Figure 1.8**. The structure has a single hidden layer that feds into the single output layer. Unlike the inputs and outputs, the states of the hidden layers are not directly observable. One cannot directly enforce the states of hidden layers since those states are learned as opposed to the inputs which are provided by the user. This reflects the fact that the structure has two transformations. Firstly, a transformation between inputs and the hidden layers. Secondly, the transformation between hidden layers and output layers [10]. Furthermore, both transformations have a different weight matrix w^1 and w^2 as shown in **Figure 1.8**.

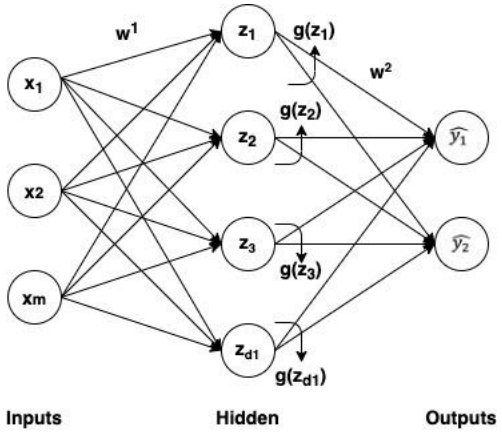


Figure 1.8. Single layer neural network [10]

Figure 1.9 represents a neural network with fully connected or dense layers between inputs and hidden layers or hidden layers and outputs. Here the symbol '**X**' represents the dense layers. Stacking of more layers like this will helps to create a deep-neural network as shown in **Figure 1.10**. The final output of the deep-neural network can be computed by going deep and deep into the network representation. Equation (1.7) (adopted from [10]) represents the output of the layer $z_{k,i}$ in the **Figure 1.10**.

$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{n_{k-1}} g(z_{k-1,j}) w_{j,i}^{(k)} \qquad (1.7)$$

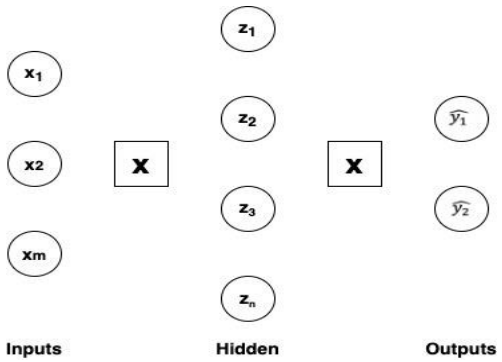


Figure 1.9. Fully connected or Dense layer neural network [10]

As depicted in **Figure 1.10** DNNs comprised of fully connected layers (multilayer perceptron's). Furthermore, output activations of a fully connected layer composed of the weighted sum of all input activations. This means all output activations have a connection to all inputs. Such a structure consumes more storage and it also requires more amount of computations. Depending upon the applications several methods such as weight sharing, sparsely connected layers have been used in literature to mitigate the issues of storage and computations.

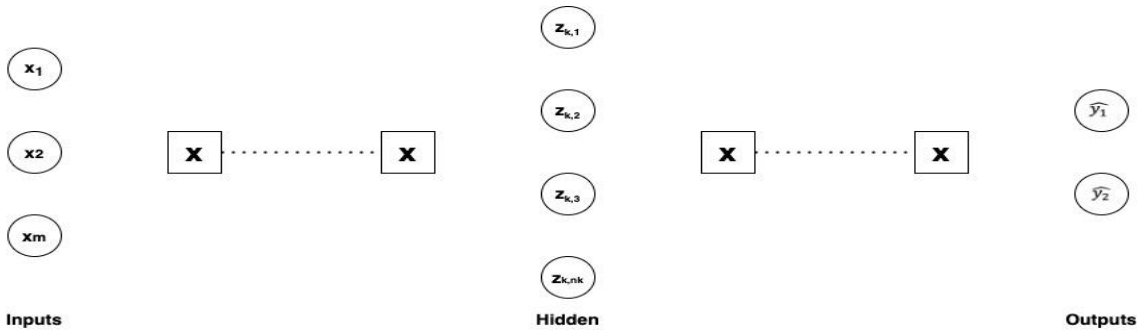


Figure 1.10. Representation of a deep-neural network [10]

1.3.1 : Computational Complexity of DNNs

Figure 1.11 shows convolutional neural networks (CNNs). CNNs are famous DNNs forms which are based on convolutional layers. Furthermore, the computations within a convolutional layer is basic convolution [13].

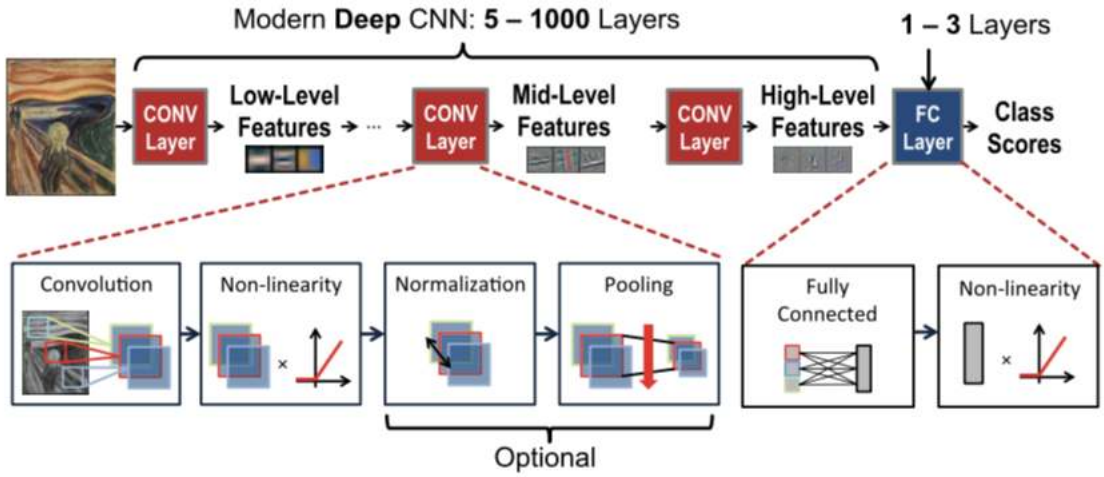


Figure 1.11. Convolutional Neural Network [13]

LeNet [16], AlexNet [17], Overfeat [18], VGG-16 [19], GoogLeNet [20], ResNet [21] are some of the famous DNN models available in literature. **Table 1** (adopted from [13]) shows a comparison between those models across matrices such as the number of filters, number of channels, weights and MACs. Forward pass through a DNN during inference requires an

enormous amount of MAC operations and memory access. Consider the case of AlexNet in **Table 1** It requires 724 million MAC operations and $3 * 10^9$ memory access. Furthermore, those operations are performed on 32-bit floating point operators.

From **Table 1** it is clear that energy cost associated with arithmetic and memory operations for DNN models during the inference phase is very high. This makes such models prohibitive for a number of applications. One way to minimise the computational and memory access cost is by quantizing the values to a smaller number of bits.

1.4 : Binarized Neural Networks (BNNs)

Binarization is a form of network quantization in which data can only have two values, either +1 or -1. Furthermore, those two values can be represented using binary variable such that a binary '1' represents +1 and a binary '0' represents the value -1. Binary Connect (BC) [22] is one form of network in which weights are quantized. Whereas in binarized neural networks (BNNs) both weight and activations are quantized [14].

A real-valued variable 'x' can be transformed into two values +1 and -1 using a deterministic binarization function [14]. The deterministic binarization function defined in equation 1.8 (adopted from [14]) .

$$x^b = Sign\ (x) = \begin{cases} +1\ if\ x \geq 0 \\ -1\ otherwise \end{cases} \qquad (1.8)$$

Table 1. Comparison of popular DNNs models (adopted from [13])

Metrices	LetNet-5 [16]	AlexNet [17]	Overfeat fast [18]	VGG-16 [19]	GoogLeNet-v1 [20]	ResNet-50 [21]
Input Size	28x28	227x227	231x231	224x224	224x224	224x224
CONV Layers	2	5	5	13	57	53
Filter Size	5	3,5,11	3,5,11	3	1,3,5,7	1,3,7
Channels	1,20	3 - 256	3 - 1024	3 - 512	3 - 832	3 - 2048
Weights	2.6k	2.3M	16M	14.7M	6.0M	23.5M
MACs	283K	666M	2.67G	15.3G	1.43G	3.86G
FC Layer	2	3	3	3	1	1
Filter Size	1,4	1,6	1,6,12	1,7	1	1
Channels	50, 500	256 - 4096	1024 - 4096	512 - 4096	1024	2048
Filters	10, 500	1000 - 4096	1000 - 4096	1000 - 4096	1000	1000
Weights	58K	58.6M	130M	124M	1M	2M
MACs	58K	58.6M	130M	124M	1M	2M
Total Weights	60K	61M	146M	138M	7M	25.5M
Total MACs	341K	724M	2.8G	15.5G	1.43G	3.9G

1.5 : Research Questions

A comprehensive literature survey comprising near-memory and in-memory computing is presented in **Chapter 3**. Several research questions raised in this thesis based on the literature review discussed in **Chapter 3** and are presented as follows:

i. Most of the DNN models explained earlier was designed to attain maximum accuracy. The hardware implementation complexity of these models is not properly considered [13]. One way to address this issue is by reducing the precision of operands as well as operations. BNNs falls into the category of fixed-point precision reduction method.

ii. In BNNs both activations and weights are 1-bit fixed-point. This has a significant impact in storage requirements. Furthermore, as the number of bits increases the energy and area associated with a fixed-point adder increases linearly. Whereas, the area and energy increase quadratically for a fixed-point multiplication [23].

iii. Memory energy consumption normalized with respect to multiply and accumulate energy consumption is 100 times compared to SRAM, 500 times compared to DRAM and 1000 times compared to Flash Memory [23]. In addition, most of the IMC scheme for BNNs available in the literature focused on utilising 8T-SRAM or 10T-SRAM. Compared to the conventional 6T-SRAM, 8T and 10T-SRAM consumes more power and area. Are there any possibilities to design an IMC scheme for BNNs targeting edge devices (L2, L3 caches)?

iv. Majority of the IMC scheme available in the literature do not fit for CNN applications due to degradation of accuracy. Furthermore, this is from the fact that most of the designs have 1-bit input or 1-bit output. Is it possible to design an IMC scheme for BNNs that is capable of improving the accuracy?

v. Normally, an IMC scheme needs ADC (Analog to Digital Converter) and DAC for its effectiveness. This also contributes to the total power consumption. Is it possible to design an IMC scheme for BNNs mitigating the power consumption of ADC and DAC?

vi. Investigate the possibility of an IMC scheme for BNNs employing the use of current computing (analog computing) which works well during the read operation of the 6T-SRAM bit cell.

1.6 : Objectives of the thesis:

The main objective of this thesis is to investigate the possibility of an IMC scheme for the BNNs targeting edge devices (L2 or L3 caches) and answer the research questions discussed earlier.

i. Investigate an IMC scheme for the BNN's where input activations and weights are quantised as +1 or -1 (encoded as binary '0' for -1 & binary '1' for +1). Furthermore, the work must focus on storing the input activations in 6T-SRAM cell rather than the weights.

ii. Determine the effectiveness of an IMC scheme for BNNs utilising the scope of current computing (analog computing) within the 6T-SRAM bit cell by designing a 4-bit DAC that can be attached to the output of a sense amplifier. Furthermore, check the functionality of the DAC circuit which performs a dot product operation between the input activations and the weights during the read phase of 6T-SRAM.

iii. Improve the accuracy of the system by enabling 4-bit precision to the weights ($W_3W_2W_1W_0$).

1.7 : Thesis Organisation

This thesis is organised as follows,

The thesis starts with the research topic and the motivations, behind the selection of this particular research area. A brief introduction to DNN's, it's terminologies, and drawbacks (complexity) are discussed in **Chapter 1**. Furthermore, **Chapter 1** also focused on discussing key conceptual ideas of BNN's including methods to reduce precision.

Chapter 2 has provided a detailed introduction to compute-in memory. The chapter starts with the importance of compute in-memory and its relevance for today's data-centric applications. The basic difference between near memory computing and in-memory computing was also highlighted in the chapter. Furthermore, the chapter also covered strategies for effective IMC design for DNN's.

Chapter 3 briefly reviewed the key advancements that happened in the field of compute in-memory. Furthermore, the chapter also focused in reviewing, how compute in-memory was used in memory technology such as SRAM.

The major objective of this thesis was to design and implement, an IMC scheme for BNN's. The detailed circuit and layout design, simulation and results are briefly discussed in **Chapter 4**. **Chapter 5** concludes the thesis with future direction for improving the proposed method and some of our future research plans.

Chapter 2 : Theoretical Background of Compute In-Memory

2.1 : Compute In-Memory

Memories are a very important part of many computing systems. It allows storing critical parameters and data, which needs to be accessed while processing. Faster and smaller transistors are produced as a result of CMOS scaling. Even though CMOS scaling helped in improving the modern CPU's speed, the slow access time of memories have a negative impact on the overall processing time. The increasing gap between memory access time and CPU clock speed is often regarded as Memory Wall [24]. Multicore processors are common nowadays. However, energy and bandwidth have become dominant issues while multicore processors are operated in parallel [25].

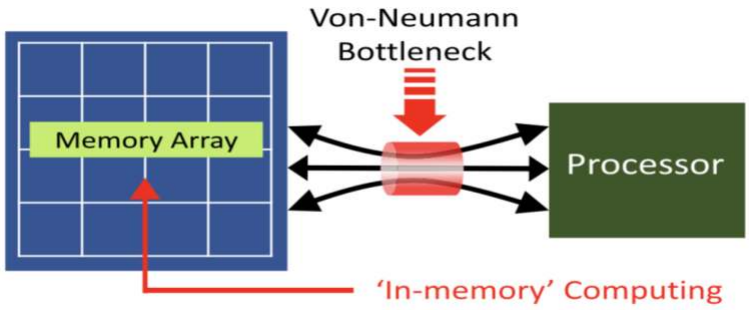


Figure 2.1. The basic idea of in-memory computing and the conventional Von-Neuman Architecture (adopted from [26])

Figure 2.1 depicts the conventional Von-Neuman architecture. The memory and processing units are separated in this architecture, with data flows between them. Von-Neuman architecture is regarded as the primary reason for memory becoming a bottleneck in computing systems. Today's computing systems are built based on Von-Neuman architecture, where data is shuttled between memory and processing units. Once the execution of programs is over, the results must have to be taken back to the memory from the processing unit. This process of shuttling back and forth of data between the memory and the processing unit incurs a significant loss of energy and latency. Data movement is very expensive in terms of energy and it is getting aggravated by a tremendous growth in data-centric applications such as AI. Moreover, the latency is another bottleneck associated with accessing data from memory unit is also increasing [27].

It is more expensive to access data from memory compared to multiplying two numbers [23]. Modern approaches such as Graphics Processing Units (GPU), where hundreds of processors

are kept in parallel [28], and Application Specific Processors [29], which are specifically designed for certain applications are not able to mitigate the problems associated data movements. Those problems have prompted researchers to think about another computing paradigm which is different from Von-Neuman architecture and is known as In-Memory Computing (IMC).

2.2 : What is Compute in-Memory?

Compute in Memory comprise of some typical solutions based on a single principle where additional compute modules are created within the memory rather than moving the data out of the main memory to the CPU. Such a method makes the system efficient and faster. Moreover, such strategies are important for today's data centric applications. Today AI systems are operated on large data set. Moreover, those systems are designed to collect and process a large amount of data. Due to data movement, the memory becomes a bottleneck in such systems. Compute in-memory is designed to adjust those bottlenecks. Moreover, compute in-memory is often treated as relative to any existing system comprising compute and memory [30]. Theoretically, one question arise which memory is referring to compute in-memory?

Figure 2.2 shows a relatively standard computing system for general purpose applications. The system has a microprocessor with CPU and increasing large levels of cache. DRAM chips and SSD (hard drive) are off-chip. During general operation, the system has a progression about how the memory is used. L1 and L2 caches are small as well as the fastest in the memory hierarchy and are placed nearer to the CPU.

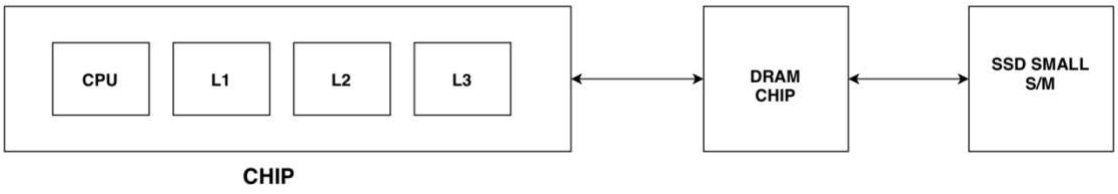


Figure 2.2. Relatively standard computing system for general purpose applications [30]

The memory size increases gradually whereas speed decreases quickly as we move horizontally from L1 cache to SSD. DRAM chips and SSD are an order of magnitude larger than the on-chip memory (L1, L2, L3 Cache). For a memory system as shown in **Figure 2.2** CPU works directly on L1 cache, whereas other aspects of the memory systems such as L2 and L3 caches, DRAM and SSD are accessed by the CPU at regular intervals. This means L1 cache is accessed frequently, whereas L2 cache will be an order of magnitude less frequent compared to L1

cache. Furthermore, L3 will be an order of magnitude less frequent compared to L2 cache. This is applicable to off-chip DRAM chips and SSD memory as well [30]

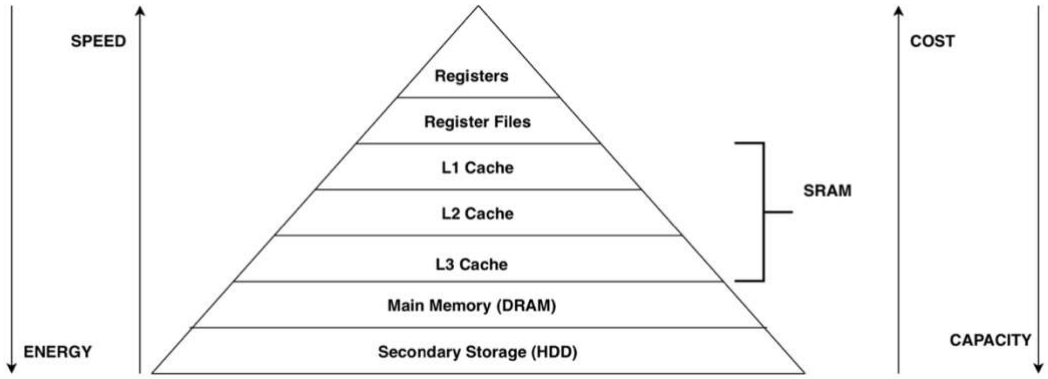


Figure 2.3. Typical Memory Hierarchy [31]

Figure 2.3 depicts the general memory hierarchy. As we move up the pyramid the energy decreases and speed increases. On the other hand, as we move down the pyramid the manufacturing cost decreases with an increase in the density of stored bits. Therefore, frequently accessed data are stored in memories that are nearer to the processing engines, whereas data's that are not accessed frequently are stored in distant memory units such as DRAM or HDD's.

Fundamentally Compute In-Memory is not applicable to L1 cache as it is used by CPU for computation [30]. Instead, Compute In-Memory would mean,

- Compute at L2 cache which transports data from L2 to L1
- Compute at L3 cache, same as L2 cache.
- Compute at DRAM chips: Instead of transferring gigabytes of data through DDR interface, compute directly at DRAM so to avoid data transfer.
- SSD(HDD): Instead of transferring terabytes of data, add compute in memory to SSD, to avoid data transfer.

The existing memory structures available on today's markets have some built-in assumptions.

1. "Temporal Locality: If any particular location of memory is accessed at a particular point of time, then it is likely that the same location will be accessed again in the near future" [30].
2. "Spatial Locality: If a particular storage location is referenced at a particular time, then it is likely that the same location will be referenced again in the near feature" [30].
3. "Probability not certain: It is not possible to predict which data is needed next" [30].

Most of the modern AI applications have data patterns that does not match well with the traditional memory hierarchy shown in **Figure 2.3** and moreover, it won't work well with the built-in assumptions. Compute In-Memory is specifically designed for capturing more difficult access patterns. The amount of memory needed by any application over a period of time is defined as the 'working set'[30]. Moreover, the working set helps to determine the frequently used/accessed memory locations.

Compute in-memory for an application is designed by two steps. Firstly, one should identify the working set of that application. Secondly, map the working set to the memory location that suits the application and finally, one should design a compute in-memory for the respective application [30].

Compute In-Memory is not always useful. Firstly, an application is always needed to take advantage of and hence not good for general purpose applications. Secondly, applications must be greater than 90% of the system power and time. Otherwise, it is hard to get significant improvements (10x). Finally, applications that suit into L1 cache are always hard to improve [30].

2.3 : In-Memory computing for DNN's

In conventional Von-Neuman architecture, the data is required to move from memory to CPU for performing certain computations and the results are sent back to the memory. This movement of data increases the overall energy consumption in DNN's. Furthermore, the distance between the memory and computational unit limits the overall bandwidth that supplies data to the computing unit. In addition, it also reduces the overall throughput of the system. This is referred to as the memory wall [24].

Several previous studies have tried to bring the memory and computing unit closer or to integrate memory and computing units together. One of such is IMC, in which the computational unit is integrated within the memory. This technique is applicable to various memory technologies such as SRAM, DRAM etc. The interesting feature about this mode of computation is that it mostly relies more on mixed-signal circuit design topologies that favours them to perform the computation in analog domain [32].

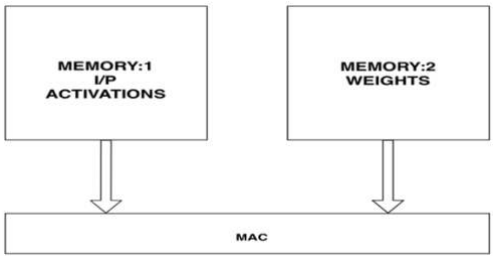


Figure 2.4. Conventional Architecture [32]

The conventional processing architectures used in DNN's consist of two memory units. One serves as the memory to store input activations and the other for storing weights. During the DNN processing phase, both the input activations and weight vectors are readout and are passed through a MAC as shown in **Figure 2.4**. The main issue with such a processing strategy is two-folded. Firstly, the memory interface put a limitation on the number of weights that are read at a time. Secondly, limited memory bandwidth possess a limitation in the number of parallel MAC operations, consequently, overall throughput of the system is reduced [32].

The important operation within a DNN is MAC. The efficient processing of DNN can be accomplished by matrix-vector multiplication. This matrix-vector multiplication can be performed by an IMC strategy. **Figure 2.5** depicts the architectural feature of IMC. The key feature of the architecture is to move and integrate the computing unit into the weight memory that serves to store the weight matrix. Moving of computation into the weight memory helps to improve the processing in three ways [32]. Firstly, a significant reduction in weight movement, which also reduces the reading of weights. Secondly, it helps to read out the final outputs from the memory. Finally, processing In-memory also helps in increasing the overall bandwidth, because of the number of weights that can be accessed in parallel increases. The memory interface is not a constraint here, as the entire weight memory AxB can be read out in parallel [32].

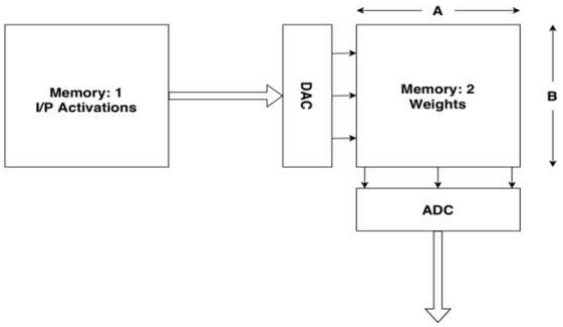


Figure 2.5. Processing In-Memory Architecture recreated from [32]

Figure 2.6 represents the conventional data flow architecture for IMC. In this architecture the data flow inside the weight storage memory is assumed to be stationary. The input activations stored in memory_1 are read and passed through a DAC and are applied to the storage elements of weight memory through the word lines. MAC array is implemented using storage elements that store the weights. At each storage element, a multiplication is performed. The computed output activations or the partial sum of a computation are read from the memory using bit lines (BL/BLB). For the implemented MAC array 'A' elements with 'B' rows can be accessed at the same time. Furthermore, AXB operations are performed in parallel per cycle.

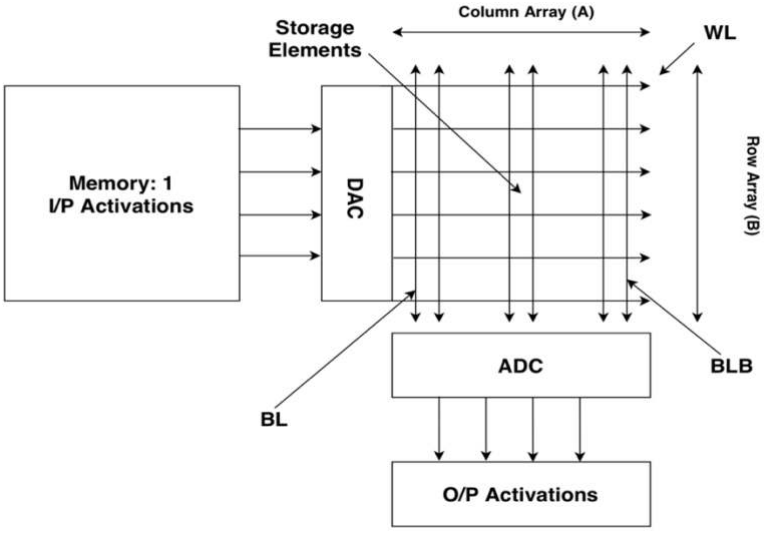


Figure 2.6. Conventional data flow architecture for In-Memory computing recreated from [32]

One way to reduce the number of input activations read is to reuse the input activations used in weight memory (memory_2) across the columns. DAC and ADC are required for converting bit line (BL/BLB) and word line (WL) values. The cost of DAC in this architecture depends on the precision of input activations whereas, the cost of ADC is decided by the precision of outputs [32].

Chapter 3 : Literature Survey

The significance of compute in-memory in today's data centric applications was clearly discussed in **Chapter 2**. Von-Neuman architecture is regarded as the primary reason for higher energy consumption in DNN's. To overcome the 'memory wall' compute-in-memory is proposed in the literature. This chapter reviews the main advancement that has happened in the field of compute in-memory. Furthermore, the chapter briefly reviews how compute-in-memory is used in memory technology such as Static Random-Access Memory (SRAM).

The first section of the chapter is devoted to describing the efforts to bring computing nearer to the off-chip memory (eg: DRAM, SSD etc.). This approach is referred to as near memory computing. In such a strategy processing is performed very nearer or at close proximity to the memory array, and hence the name near memory/near-data-processing. In recent years near memory computing has focused on advanced memory technologies such as embedded-DRAM (e-DRAM) and 3-dimensional stacked memories (3-d stacked DRAM's). The second section of this chapter focuses on In-Memory Computing (IMC). IMC tries to do the computations within the confine of the computational memory unit to reduce the data movement significantly. Many recent works are focused on IMC within the well-known SRAM, DRAM and NVM.

3.1 : Near Memory Computing

Process technology for processors is different from high-density memories. As a result, high-density memories are fabricated as individual chips. Separate Fabrication of memory chips creates problems with accessing these off-chip memories (Bandwidth and energy). These limitations can be overcome by creating compute near to the high-density memory. This strategy helps in reducing the energy required for accessing the off-chip. Moreover, it improves the overall memory bandwidth.

Advanced memory technologies such as e-DRAM and stacked DRAM's (3-D) have enabled processing near the memory. This section describes how these technologies are used to process DNN's.

3.1.1 : High-Density Memories

Off-chip access during DNN processing can be effectively reduced with the help of high-density memories such as embedded DRAM (e-DRAM)[33]. Higher storage density is one of the typical features of e-DRAM and hence is used to store megabytes of weights and activations. One such design is DaDiaNo [34]. **Figure 3.1** shows the fundamental architecture

of DiaNao accelerator proposed by Chen.et. al [35]. DiaNao is famous for its speed and low energy execution of DNN's.

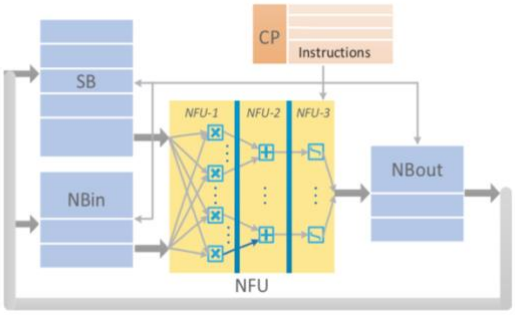


Figure 3.1. Fundamental Architecture of DiaNao [35]

The important part of a DiaNao architecture is the Neuronal Functional Unit (NFU), where computations of a neuronal outputs are evaluated (Pipelined). The same DiaNao architecture is modified to address the fundamental issue of memory storage (for reuse) and bandwidth (for fetching) [34]. The key features of the architecture are three folded. Firstly, neurons and synapses are placed close to each other. This helps in minimising the data movement. Moreover, placing neurons and synapses close to each other saves energy and time. Secondly, neuron values are transferred compared to the synapse's values. This helps in reducing the external bandwidth since neuron values are an order of magnitude less compared to the synapse's values. Finally, the local storage is broken into smaller tiles to achieve higher internal bandwidth. The NFU proposed in[34] is far more complex than the one in [35]. Higher latching compared to SRAM, destructive reads and periodic refresh are the main drawbacks of using e-DRAM.

3.1.2 : 3-D Memory (Stacked Memory)

3-D memory is the latest memory technology in which DRAM's are stacked on top of one another using a technology known as Through-Silicon-Vias (TSVs). Hybrid Memory Cube (HMC)[36] and Hybrid Bandwidth Memory (HBM)[37] are commercial forms of 3-D memory. Compared to the conventional 2-D memory, 3-D memories offer higher memory bandwidth (order of magnitude) and at least 5-times reduction in access energy. Moreover, TSVs offer less capacitance compared to the interconnects used in conventional 2-D memory. Performance of multicore processors is often limited by the memory system bandwidth. To address this issue, HMC [36] and HBM [37] are proposed in the literature. HMC is a three-dimensional DRAM architecture, which improves overall system Bandwidth, latency, power and density. **Figure 3.2** shows an HMC device with TSVs technology and fine pitch copper

pillar interconnect. Normally, HMC contains a stack of heterogeneous dies. Moreover, HMC has a greater number of connections, so as to reduce the distance travelled by the signals.

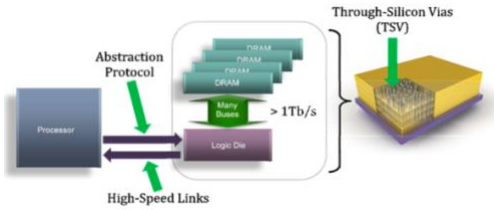


Figure 3.2. HMC device with TSVs technology and fine pitch copper pillar interconnect [36]

However, HBM is a standard DRAM design capable of achieving a bandwidth greater than 256 Gbps with a reduction in overall power consumption. **Figure 3.3** depicts the fundamental structure of an HBM device. The core Die inside a heterogenous HBM structure composed of a conventional DRAM architecture with TSVs interface. Compared to the conventional DRAM's, HBM offers improved capacity, bandwidth and power efficiency.

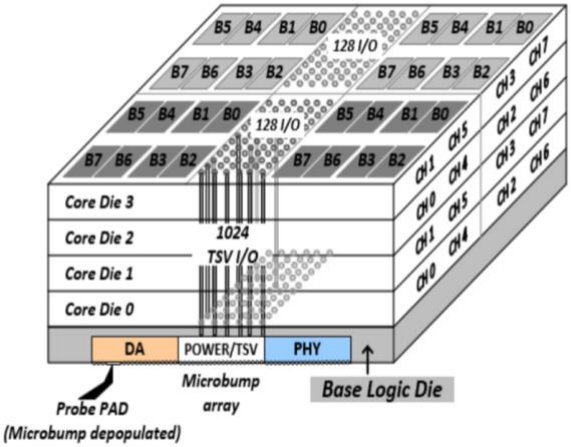


Figure 3.3. Fundamental structure of an HBM device [37]

Recent works on efficient DNN processing have explored the use of HMC in a number of ways. Firstly, Neurocube [38] has brought the memory and computations close to each other with the help of HMC technology. **Figure 3.4** shows the key components of Neurocube. The architecture composes of a global controller, processing elements (PE's), routers and programmable neuro sequence generator (PNG). Out of these key components, PE's are the main computing units, with several multiply and accumulate (MAC) units. Clusters of PE's are connected by a 2-D mesh network for processing. Moreover, PE's are capable of accessing memory channels in parallel. Furthermore, with the help of high-speed TSVs technology, multiple PE's are in communication with the DRAM vaults.

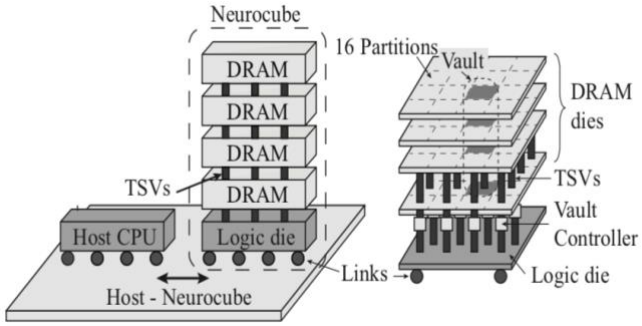


Figure 3.4. Key components of Neurocube [38]

Another DNN processing work that has exploited the use of HMC is the Tetris[39]. Tetris is a neural network accelerator exploiting near data processing. **Figure 3.5.a** depicts the hardware architecture of Tetris. HMC is used as the substrate of Tetris and is divided vertically into 16, 32-bit wide vaults. DRAM dies are connected to the base logic using vault channel buses utilizing TSVs technology. Each DRAM composed of either two banks or vaults. Moreover, access to different vaults or banks can overlap. **Figure 3.5.b** shows the detailed structure of the Neural Network (NN) engine within each vault. A 2-D network array is created by connecting hundreds of PE's together. Moreover, a global buffer is shared across all PE's for storing and accessing data from the memory.

Long DRAM latency remains problematic for the design aspects discussed earlier. One way to improve the performance is by stacking SRAM's instead of DRAM's known as QUEST[40]. This design strategy helps in reducing memory access. **Figure 3.6** depicts the overall architecture of QUEST. An SRAM capacity of 96MB in total is achievable by stacking 8-SRAM dies. The SRAM dies are connected to the chip using inductive coupling interface known as Thru Chip Interface (TCI). TCI has a lower integration cost compared to TSVs. Furthermore, with wireless communication (TCI) both high memory bandwidth and low access latency are achieved.

So far 3-D memory design discussed involves TSVs or TCI for connecting memory and logic dies, which are designed and fabricated separately. Monolithic 3-D integration is nanotechnology, which helps to fabricate thin layers of logic and memory on the top of each other. N3XT[41], uses a non-volatile memory using a monolithic 3-D integration system.

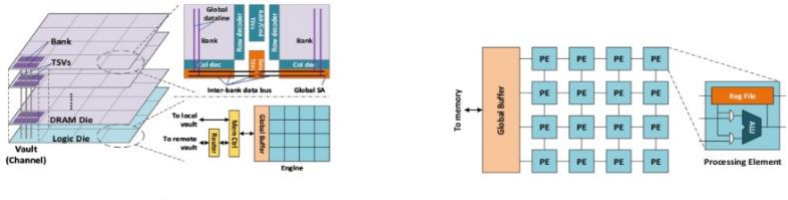


Figure 3.5. (a) Hardware architecture of Tetris, (b) detailed structure of an NN [39]

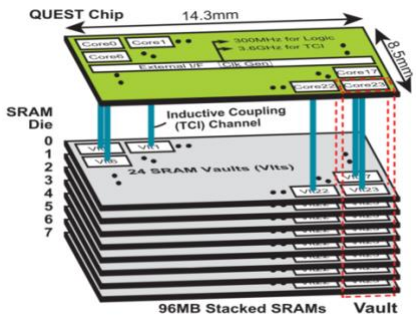


Figure 3.6. Overall architecture of QUEST [40]

3.2 : In-Memory Computing (Processing In-Memory)

Compute in-memory based on CMOS technologies are of great importance nowadays due to the presence of field effect transistors. Furthermore, the ease of massive production and commercialisation makes CMOS technologies very trending. Emerging non-volatile memories based on memristors are shown to be very efficient in performing dot product operation. Analog dot product operations based on memristor technologies are very fast and efficient, when compared to the Boolean operations. However, rapid commercialisation of memristor technologies are hindered due to the challenges related to large scale manufacturing.

This section discusses in-memory computing, which enables computations within the memory. The difference between conventional memory architecture and in-memory architecture was highlighted in Chapter 2. Furthermore, the benefits of using compute in-memory were discussed very briefly. This section primarily focuses on discussing well known compute in-memory designs and how processing in memory is performed using memory technology such as SRAM.

3.2.1 : Static Random-Access Memory (SRAM)

In recent years, a number of efficient DNN processing design's enabling SRAM bit cell for performing In-Memory and bit-cell computations have been proposed. In general, SRAM bit-cell computations can be classified as current-based and charge-based design. Current-based designs perform dot operations utilizing current-voltage (IV) characteristics of a bit-cell. A well-known current-based bit-cell design is machine learning classifier implemented in 6T-SRAM[42]. Since computations are performed inside the 6T-SRAM cell, excessive memory access is reduced significantly.

Figure 3.7 depicts the overall architecture of the machine learning classifier. The whole system operates in two modes. SRAM and Classify mode. During SRAM mode, normal read, write is performed and during Classify mode the system implements a weak classifier. During the classify mode, bit line pairs (BL & BLB) are pre-charged first. This is followed by driving the

word lines (WL) with an analog voltage corresponding to the input activations. Word line Digital to Analog Converters (WLDAC) are used at the periphery of each WL's, enabling the bit-cell to pull a current (IBC) modulated by its own WL voltage drop (VBL) from the bit lines (BL or BLB) according to the stored data. This operation is equivalent to the dot product of the stored data by +1/-1. All the IBC across columns add together as an inner product.

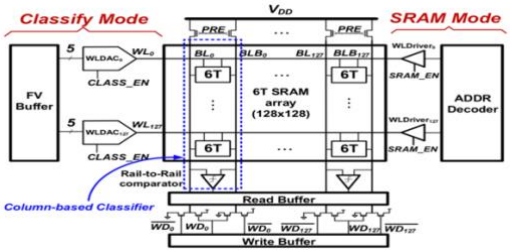


Figure 3.7. Overall architecture of machine learning classifier [42]

Figure 3.8.a shows the cell current accumulation and **Figure 3.8.b** depicts a WLDAC's circuit diagram. The binary-weighted PMOS current sources are selected using a 5-bit digital features. Moreover, PMOS current sources helps in enhancing the linearity of charge drawn by bit-cells. Furthermore, the driver transistor (M_{DR}) and a diode-connected access transistor (M_{AR}) are biased using the current from the PMOS current source and helps in driving the WL's.

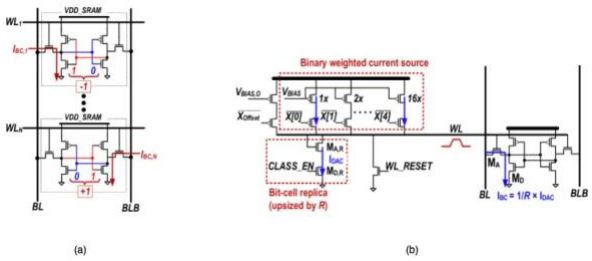


Figure 3.8. (a) Cell current accumulation in machine learning classifier, (b) WLDAC circuit diagram [42]

Embedded machine learning classifier with 1-bit outputs or smaller number of output classes are found to be insufficient for efficient implementation of CNN's. This issue is addressed by a current based IMC design with 7-bit inputs and outputs known as CONV-SRAM[43]. **Figure 3.9** shows the overall architecture of a 256 x 64 CONV-SRAM array. The area of ADC's and analog multiply-and-average (MAVa) circuits are optimised by dividing the whole architecture into 16 local arrays, with each array further divided into 16 rows.

CONV-SRAM uses a 10-T bit-cell instead of a 6-T, for storing binary weights across the local array for individual 3-D filter in a CONV-layer. The dedicated ADC across each local array computes the convolution outputs. The input feature maps (Xin) are converted to an analog voltage with the help of column-wise DAC's (GBL_DAC). **Figure 3.10** shows the schematic of the GBL_DAC circuit, which is shared across the local arrays. The schematic consists of

cascaded constant PMOS current sources. Global read lines (GRBL) are charged with current directly proportional to Xin for a duration of t_{on}. Small PMOS stack to charge GRBL, for all input codes provides better linearity and mismatch compared to 6T-SRAM, ML-Classifier. Furthermore, the pulse width of timing signals has fewer variations compared to PMOS V_t variations.

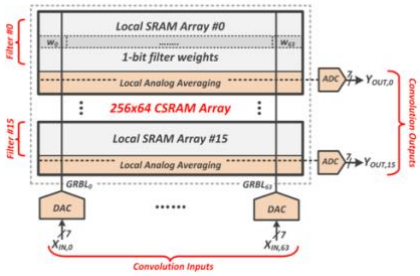


Figure 3.9. Overall architecture of CONV-SRAM [43]

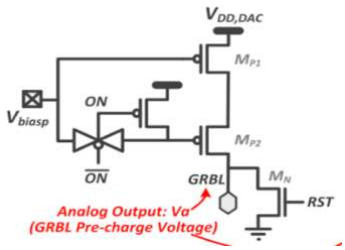


Figure 3.10. Circuit diagram of GBL_DAC [43]

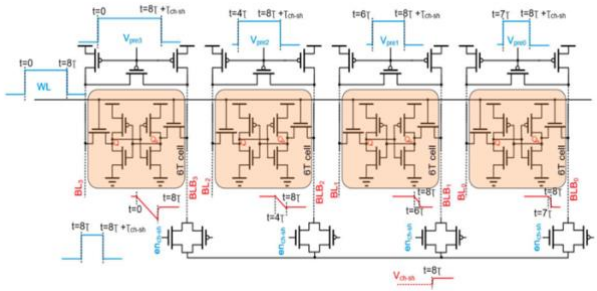


Figure 3.11. Pre-charge enabling In-memory multiplication [44]

A novel In-Memory multiplication and accumulation inside 6T-SRAM is presented in[44]. The work makes use of the pre-charge circuit where they encoded the bit significance of pre-charge pulse, to perform analog multiplication inside the SRAM. **Figure 3.11** shows the In-memory multiplication. The operand 'w' is stored in the memory cells as shown in **Figure 3.11**. Furthermore, on the word line, other operand is encoded as an analog voltage. Bit significance pre-charge is performed by enabling the pre-charge circuit at different instance and hence the analog multiplication in SRAM.

Apart from the 6T and 10T-SRAM bit cell current based computing designs, standard 8T-SRAM has also been enabled for analog like In-Memory multibit dot product operations[45]. The fundamental idea here is to apply analog voltages to the read port of the 8T-SRAM bit-

cell and concurrently sensing the output current. Without modifying or altering the standard structure of the basic 8T-SRAM cell, two configurations of dot product computations are performed**. Figure 3.12** shows the basic structure of an 8T-SRAM. However, 8T-SRAM consist of a basic 6T-SRAM bit-cell with two additional transistors which constitute the decoupled read port.

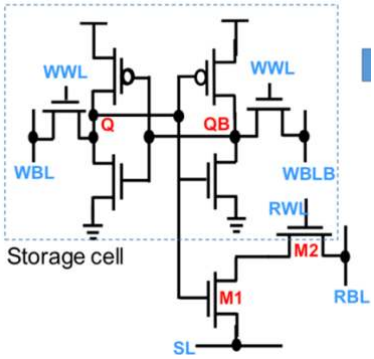


Figure 3.12. 8T-SRAM [45]

Figure 3.13.a shows the first configuration enabling dot-product operation in 8T-SRAM. As shown in the figure, whenever 'SL' is enabled by connecting it to the input voltage 'vi' and 'RWL' get turned ON [45]. Current 'IRBL' through 'RBL' is sensed and is directly proportional to the dot product 'vi.gi', where 'gi' is the 'ON/OFF' conductance of transistor M1 and M2 [45]. Another way to implement the dot product is depicted in **Figure 3.13.b**. Here the analog voltage is applied to the 'RWL', where 'SL' is supplied with constant 'Vbias'. The current produced in this case is sensed in the same way discussed earlier

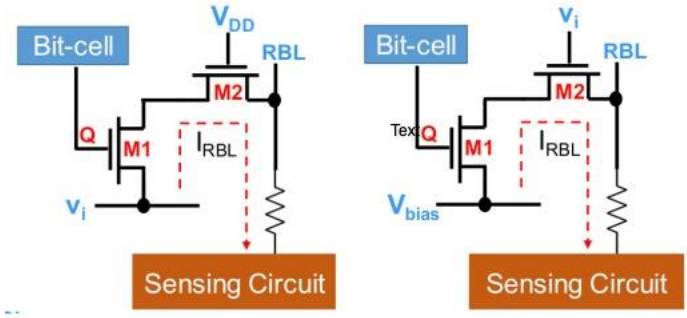


Figure 3.13. (a) configuration A, (b) configuration B [45]

Apart from performing dot operations using current based IMC design in SRAM bit-cells, it is also possible to perform Boolean operations in conventional SRAM bit-cells using IMC. Such a design is the X-SRAM [26]. The design explores in-memory vector operation within the standard CMOS 8T and 8^{+}T differential SRAM cells, with minimum modification of the periphery circuit. The design explores six different configurations for in-memory vector

operations. 75% of the total memory access have been saved by read-compute-store mechanism, which enables the storing of computed Boolean directly on to the memory without latching the data and write operations [26].

However, X-SRAM suffers a number of drawbacks. Firstly, NAND operation suffers low sense margin and it requires additional timing control. Secondly, Boolean operation in 8T-SRAM with voltage divider strategy requires additional voltage boosting. Finally, the differential cell 8^{+}T- SRAM, requires two skewed sense amplifiers [26].

The IMC designs discussed so far focused on current based computing strategy in the SRAM bit cells. Modern approaches for IMC designs have also utilised charge-based computing strategy in SRAM bit cells too. One such design in the BNN implementation in which charge based computations are enabled for storage and multiplications [46]. However, the bit cell size is higher than that off conventional 6T-SRAM. Significant reduction in data movement, as well as input and output activations, are the benefits of this design. **Figure 3.14** shows a hidden layer circuit and a neuron array structure of an M-BC with 8-transistors.

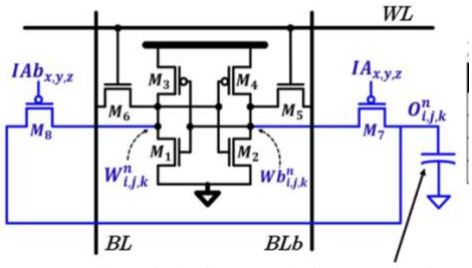


Figure 3.14. structure of an M-BC with 8-transistors [46]

Usually in BNN's Input activations (IA) or weights (W) are 1-bit, so as to perform XNOR operation. Conventionally, the charge sampled on the cap is taken as the output XNOR value, whereas accumulation for the pre-activation are computed by shorting all the caps in the neurons. Multiplying Bit-Cell (M-BC) is the key for eliminating or minimizing IA or W's movements. Moreover, pre-activation movement is eliminated by accumulation, which happens as a part of charge shorting. Introducing two additional PMOS in standard 6T-SRAM bit-cell, M-BC enables storage and XNOR operation at the cost of additional area.

Binary convolutions within the SRAM cell are accelerated using sectioned SRAM [47]. A typical binary convolution operation is performed by XNOR operation followed by a population count (pop-count). The work presents two techniques to accelerate the binary convolution operation. Firstly, the parasitic capacitance present in 10T-SRAM is enabled as charge sharing, so as to perform vector XNOR. Furthermore, this strategy helps in approximating the pop-count. Secondly, the design has utilised bitwise XNOR, in which digital

bit-tree adder performs the pop-count operations. Apart from the two-design strategy, the design has enabled sectioned SRAM that allows performing multirow convolutions in parallel. So far, the section discussed IMC using current and charge based designs. Recent works on the efficient processing of DNN's have also utilised the scope of mixed-signal circuit designs too. Deep-In-Memory-Architectures (DIMA) or processors for inference applications using 6T-SRAM array is a well-known mixed-signal circuit design enabling IMC with multifunctional capabilities [48].

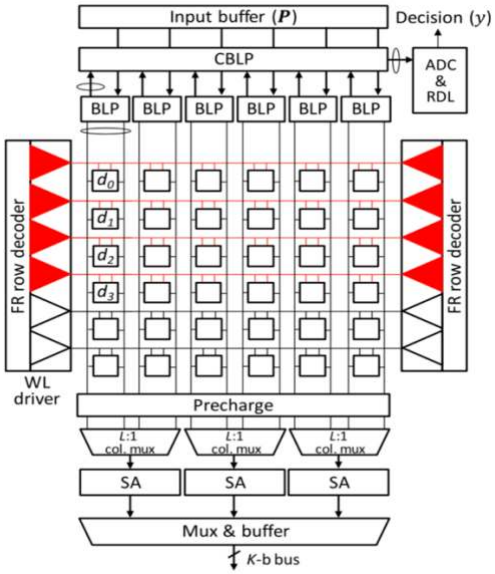


Figure 3.15. DIMA Architecture [48]

Figure 3.15 shows a conventional DIMA architecture. The DIMA architecture composes of four stages. Firstly, Multi-Row Functional Read (MR-FR), that can access multiple rows in a single pre-charge. Pulse Width Modulated- Word-line (PWM-WL) signals generate BL voltages equivalent to a weighted sum of multiple bits. Secondly, 'BL' Processing (BLP) helps in performing computations such as multiply, absolute value or comparison of the BL voltages. Moreover, these computations are performed in massive-column-parallel fashion. Cross BL Processing (CBLP)is the third vital part of DIMA, capable of aggregating the BLP voltage to a scalar value. Furthermore, the same scalar value is sliced for the final decision. The final stage of DIMA is the ADC and Slicing.

Figure 3.16 depicts the chip architecture of a DIMA inference processor. The chip architecture is composed of a core, digital controller (CTRL) and an input register. The function of input register is to stream the operands. However, the normal read and write circuitry is placed at the bottom part of the chip, whereas In-Memory processing blocks are placed on the upper part of the chip. Such a placement helps in physical separation as well as functional maintenance. The

architecture is capable of processing 128, 8-b words per cycle. This means two consecutive cycles can process 256 vectors, with 8-b elements.

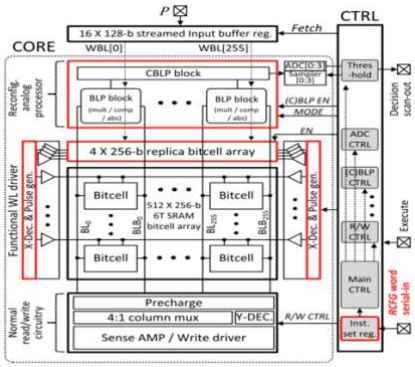


Figure 3.16. Chip architecture of DIMA inference processor [48]

DIMA-CNN [49], is a variant of DIMA architecture implemented with CNN. The architecture and circuit topology used in this work mainly focuses on reducing energy and delay associated with data movement in CNN. Furthermore, DIMA is embedded with mixed-signal computations that are low swing and energy-efficient at the periphery of SRAM bit-cells. The key design aspect of the work is a mixed-signal multiplier enabled for data reuse in convolutions. The work has also tried to address some of the implementation issues. Firstly, frequent data access, processing and convolutions in parallel are addressed through DIMA's energy efficiency, intrinsic parallelism, multiple bit-cell array banks and optimised kernel data storage. Secondly, the sliding window for convolution is optimised by using sliding window FM register. Finally, charge-recycling mixed-signal multiplier somewhat addresses data reuse in analog computations.

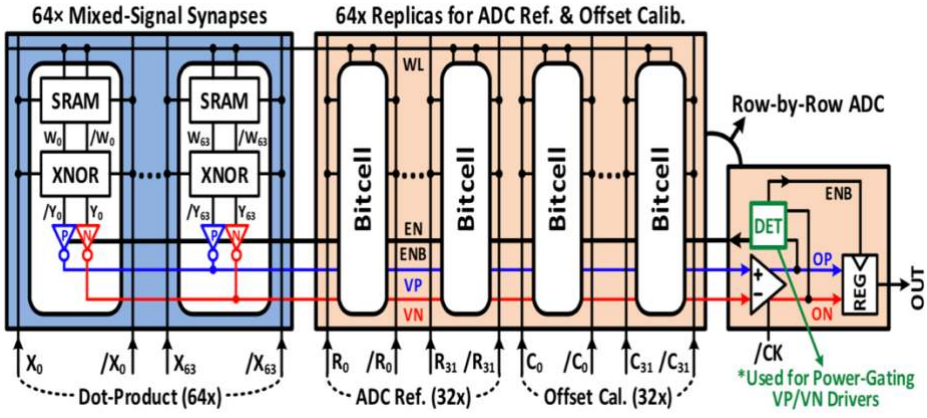


Figure 3.17. IMC Unit [50]

Another variant of the mixed-signal IMC based on standard SRAM bit cell is presented in [50]. In this work an SRAM bit-cell composed of the standard 6T-SRAM cell, a binary multiplier (XNOR based) and a pseudo-differential voltage-mode driver. **Figure 3.17** shows the fundamental IMC unit. The unit consists of 128-bit cells implementing mixed-signal dot

product between 64 inputs and synapses. The first 64-bit-cells contains the synapse weights which are multiplied with the inputs using XNOR gates, which are embedded within the bit-cells. A pair of inverters (PIN) is used to accumulate the multiplication results. Furthermore, the PIN's are capable of driving the bit-cells to positive (v_p) and negative (v_n) voltage levels. However, out of the 128-bit-cells only 64 are used for dot product operation. Whereas, out of the remaining 64-bit-cells, 32 are used for ADC reference and 32 for offset cancellations. In the final stage, pseudo-differential accumulators, output voltage levels and is converted to a digital output by an ADC at each row.

Chapter 4 : Proposed In-Memory Computing Cell

As discussed in **Chapter 1** computational complexity of DNN's are increasing as the number of hidden layers increases. This in-turn requires an enormous amount of MAC operations and memory access, needed for a forward pass within a DNN. One way to reduce the computational cost and the memory access is to quantise the values of operands to a smaller number of bits. The most extreme form of quantisation is known as 'Binarization' in which only two values for the quantities +1 and -1 is used. These two variables can be represented using a binary variable, where binary'1' represents the value '+1' and binary '0' represents the value '-1'. In this chapter, an In-Memory computing scheme for 'Binarized Neural Network' (BNN) is proposed. Quantisation is applied to both weights and activations within the BNN's (i.e., inputs to the next subsequent layers).

4.1 : The proposed In-Memory architecture for BNN's

Figure 4.1 shows the proposed 4x4 SRAM array employing In-Memory architecture for BNN's. The input activations corresponding to the layers of BNNs are stored in the memory. This means that the input activations are stored in the individual 6T-SRAM bit cells which is organised as 4 rows and 4 columns as in **Figure 4.1.** During the read operation, the sense amplifiers sense the input activations stored in each bit cells along the column as analog voltage and convert it to a digital data. The digital output of the sense amplifier is passed through the proposed 4-bit Digital to Analog Converter (DAC), which enables a multiplication with the input activations and the 4-bit weights ($W_3W_2W_1W_0$) applied as binary bits '1 or 0' to produce an output voltage (V_{Out}) in each individual column. The key aspects of the proposed IMC design are as follows.

- An IMC scheme is proposed for the BNN's where input activations and weights are quantised as +1 or -1 (*encoded as binary '0' for -1 & binary '1' for +1. In general, the software implementation of BNNs using +1 and -1 is easy. But it is complicated in hardware implementation. In this work, the designed IMC circuit is not capable of handling '-1'. Instead, -1 is treated as '0'. This is due to the limitation of time allocated for the research. However, this will be addressed in the next part of my research.*). Furthermore, the work focuses on storing the input activations corresponding to the layers of BNNs in 6T-SRAM cell rather than the weights.

- The proposed IMC scheme has utilised the scope of current computing (analog computing) within the 6T-SRAM bit cell by designing a 4-bit DAC that can be attached to the output of a sense amplifier.
- The proposed 4-bit DAC circuit enables a dot product operation between the input activations and the weights ($W_3W_2W_1W_0$).
- Normally in BNN's, XNOR operations on binary encoding is equivalent to a dot operation, which requires the accumulation of all products. Furthermore, the accumulation requires the summation of results. Usually, population count (pop-count) is used to estimate the total number of 1 bit in a group. The proposed 4-bit DAC circuit eliminates the need of pop-count. Accumulation (summation) of current happens within the 4-bit DAC at an accumulation net. The 4-bit DAC act as a current adder where the accumulated current is converted to an equivalent voltage with the help of the sensing element.
- The important feature of the proposed 4-bit DAC is that it has 4-bit weight ($W_3W_2W_1W_0$) precision which improves the overall throughput efficiency. This can be further increased by the proper design of the circuit.

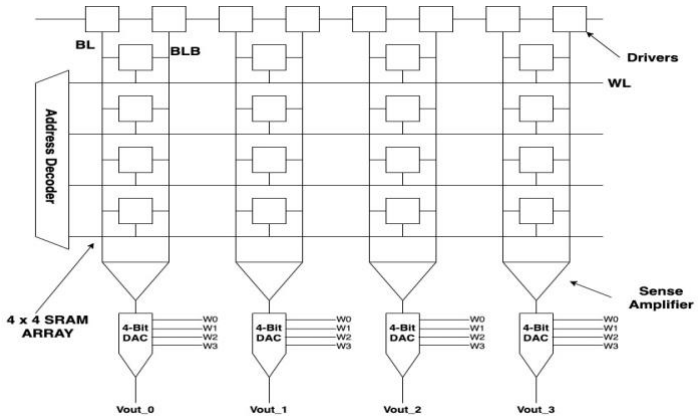


Figure 4.1. Proposed IMC architecture for the BNN's

4.2 : Overview of 6T-SRAM Cell

Memories are characterised according to the area, speed and power. Smaller the area of a single bit cell, more the memory that could be accommodated on a single chip. In other words, the cost per bit can be reduced to a greater extent. In normal practice SRAM are arranged as an array of memory locations. Writing and reading of a single bit from this array of memory is regarded as the memory access. **Figure 4.2** shows the component layout of 4x4 SRAM array. In this array individual bit cells are organized as 4 rows by 4 columns [31].

A common word line (WL) is shared across each 4 rows of bit lines (BL/BLB), whereas the bit line pairs BL and BLB, shares multiple bit lines in a column. For an SRAM, with $'n'$number of bit line pairs and $'m'$ is the bit width of a single word, the ratio $(^{n}/_{m})$ is defined as the column ratio [31].

Read and write operations are performed by the help of circuitry at the periphery of SRAM arrays. The periphery circuits decode the address locations in the form of bits, which uniquely points towards the SRAM array locations that need to access. During the upcoming access, the address decoder sets one among the 4-word line high to enable that particular row. At this moment all the other word lines are set to low, which in-turn disable the SRAM cells they control. The activated word line enables all the SRAM bit cells on the selected row, connecting to a pair of bit lines.

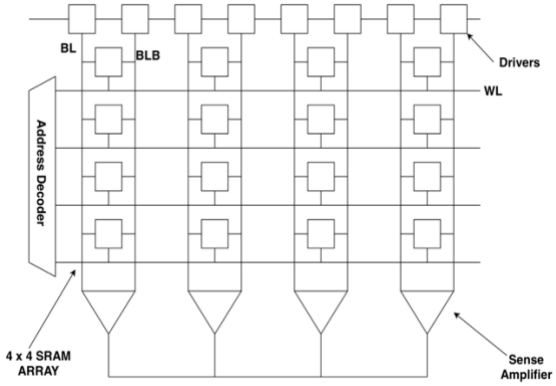


Figure 4.2. Component layout of a 4x4 SRAM array [31]

As stated earlier, the access operations are read and write. During read operations, the bit lines will carry the analog signal from the selected bit lines to the sense amplifier that converts the analog signal to digital data. In contrast, during the write operations, the incoming data is driven into the selected SRAM cell through the bit lines.

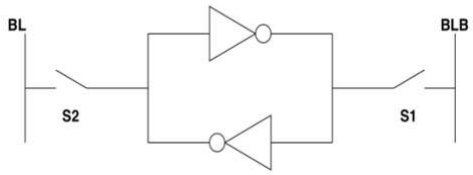


Figure 4.3. Back to back connections of inverters in a typical 6T-SRAM cell [31]

The backbone of an SRAM array is the 6T-SRAM bit cell. The main characteristics of a 6T-SRAM bit cell are low power dissipation, high switching speed and good noise margin. A 6T-SRAM bit cell is realized by the back to back connection of two CMOS inverters as shown in **Figure 4.3**. S1 and S2 are the two switches (access transistors) in the bit cell. Read and write operations can be performed with the aid of these two switches. The CMOS inverters are

configured in a cross-coupled fashion so that the output of one inverter is the input to the other inverter. Such wiring results in a positive feedback loop to create a bi-stable storage element [31].

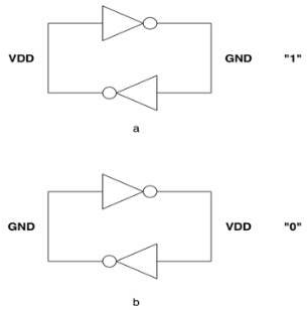


Figure 4.4. Bi-stable operation of a cross-coupled inverter [31]

Figure 4.4 represents the bi-stable operation of the cross-coupled inverters. The configuration in **Figure 4.4.a** shows the cell storing a '1' bit, whereas **Figure 4.4.b** depicts the configuration storing '0' bit. The cells capability to provide a stable state is from the fact that as long as the inverters are powered, the noise immunity of the inverters will ensure that the logic states are not altered even in the presence of electrical noise on either inverter inputs.

Figure 4.5 shows a conventional 6T-SRAM. The main feature of 6T-SRAM is the back to back connected inverters. The inverter pairs (T1, T3) and (T2, T4) act as the CMOS inverter pairs. The output of (T1, T3) is connected to the input of (T2, T4) and output of (T2, T4) is connected to the input of (T1, T3), as shown in **Figure 4.5**. The positive feedback makes sure that the inverters are capable of storing the desired states '0' and '1' at the nodes N1 and N2, as long as both the access transistors T5 and T6 are off and the SRAM cell is powered up. Access transistors T5 and T6 are turned on during read and write operations, so that the internal nodes N1, N2 will get connected to the bit lines BL and BLB [31].

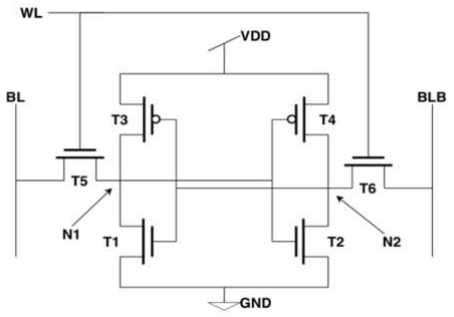


Figure 4.5. Conventional 6T-SRAM [31]

4.2.1 : 6T-SRAM bit cell write operation

The first step in writing into an SRAM cell is to drive the bit lines BL and BLB to the data values to be written. By asserting the word line (WL), both access transistors T5 and T6 get

turned on. Assume that bit '1' is available at BL and '0' at BLB. So, when T5 and T6 turned on we have '1' at N1 and '0' at N2. This will make transistors T2 on and T4 off. Furthermore, '0' at N2, makes T1 off and T3 on [31]. **Figure 4.6** shows the write operation within a 6-T SRAM cell.

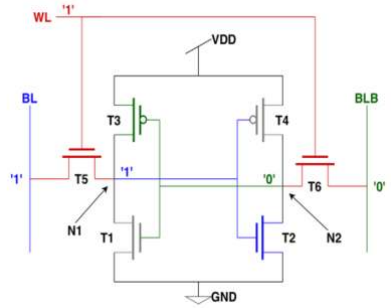


Figure 4.6. 6T-SRAM write operation [31]

If data to be written is opposite to the previously stored state, the potentials of high internal node is lowered. This action purely depends on the drive strengths of pullup (PMOS) devices and the access (NMOS) devices. An important parameter in the 6T-SRAM design is the 'γ'-ratio. The ratio of drive strength of access transistors (T5, T6) and pull up transistors (T3, T4) is known as the 'γ'-ratio. Write failure will occur if the γ-ratio is very low. Transistors needs to be properly sized so that γ-ratio is high enough to lower the potentials of high internal node (N1 or N2) below the $vtrip$ of the inverter [31].

4.2.2 : 6T-SRAM cell read operation

The first step in read operation is to pre-charge the bit lines BL and BLB to a known voltage 'VDD'. The word lines are activated and bit lines are kept floating. If the bit line columns are large, the bit line capacitances are also very large. Depending on what is stored in bit lines pairs BL and BLB, one of them starts to discharge through the access transistors (T5 or T6) and pull-down NMOS transistors (T1 or T2) that are connected in series [31].

Changes in bit lines are very slow because of large capacitances. Bit line differential voltage of the order of small millivolts can be sensed by a sense amplifier to output the stored data. The discharge current produced during a read operation normally flows from bit line to the cell ground. This normally happens to the side of the bit cell storing a logic '0'. This operation is depicted in **Figure 4.7**.

As shown in **Figure 4.7**, the discharge current will increase the potential voltage at N2 and the amount of disturbance on the drive strength of access transistor T6 and the pull-down NMOS transistor (T2). If the increased potential voltage is above the $vtrip$ of the connected inverter

formed by the transistor pair (T1, T3) then the stored data gets flipped. This phenomenon is known as read disturb [31].

The best way to prevent read disturb is to make the pull down NMOS transistors (T1 and T2), stronger than the NMOS access transistors (T5 and T6). The ratio between the strength of pull down NMOS transistor (T1, T2) to NMOS access transistors (T5, T6), known as the 'β'-ratio and is another important parameter in SRAM design. A successful read operation can only be performed by carrying out a careful sizing of NMOS transistors. This helps to achieve the desired 'β'-ratio and thus eliminating read disturb [31].

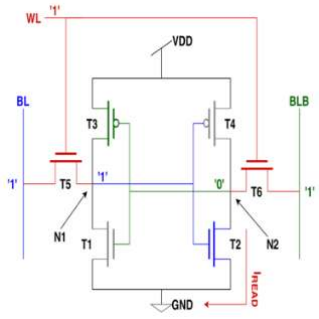


Figure 4.7. 6T-SRAM write operation [31]

4.3 : Digital to Analog Converter (DAC) Design

This section presents a current based SRAM bit cell computation scheme. The current based bit cell computation technique is used at the reading phase of SRAM operation. The basic idea is to create binary weighted current sources using MOS devices namely PMOS and NMOS.

4.3.1 : MOSFET as a current source

An ideal current source is one whose output has a constant current, regardless of the voltage applied between its inputs. MOSFET can operate as a voltage-controlled current source and produces an amplification. When a MOS device is in saturation, its output current is relatively constant[51].

Figure 4.8 shows an NMOS biased to be in saturation. Equation 4.1 (adopted from [51]), represents the drain current that flows through the NMOS device. The drain current 'I_D' in equation 4.1 is a function of the drain voltage (V_{DS}) and gate source voltage (V_{GS}), around the terminals. 'μ_n' represents the mobility of electrons in NMOS device, 'C_{OX}' is the gate oxide thickness, ($\frac{W}{L}$) represents the aspect ratio of the MOS device where 'W' is the width and 'L' is the length of the device. Finally, ($V_{GS} - V_{DS}$) represents the overdrive voltage. Plotting of equation 4.1, gives a parabola reaching its peak at overdrive voltage, reaching the maximum

current 'I_{Dmax}'. Equation 4.2 (taken from [51]) shows the maximum drain current at the peak of the parabola [51].

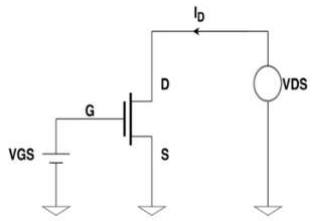


Figure 4.8. NMOS biased with gate and drain voltage [51]

$$I_D = \mu_n C_{ox} \frac{W}{L} \left[(V_{GS} - V_{TH})^2 V_{DS} - \frac{1}{2} {V_{DS}}^2 \right] \qquad (4.1)$$

$$I_{Dmax} = \frac{1}{2} \mu_n C_{ox} [(V_{GS} - V_{TH})^2] \qquad (4.2)$$

Whenever $V_{DS} < (V_{GS} - V_{TH})$, the MOS device starts to move towards the triode region. At the parabolic maximum, drain current starts to deviate from the parabolic path, and becomes relatively constant, as V_{DS} exceeds the overdrive voltage. At this point of operation, the device is said to be in saturation [51].

$$I_D = -\mu_p C_{ox} \frac{W}{L} \left[(V_{GS} - V_{TH})^2 V_{DS} - \frac{1}{2} {V_{DS}}^2 \right] \qquad (4.3)$$

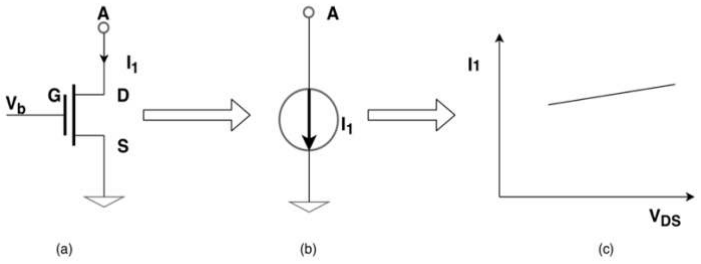


Figure 4.9. (a) NMOS in saturation operating as a current source, (b) current flow direction, (c) I_1 vs V_{DS} plot of saturated NMOS device [51].

MOSFET in saturation can be used as a current source. **Figure 4.9.a** and **Figure 4.10.a** shows an NMOS and PMOS current source. When an NMOS act as a current source, the current flows from any arbitrary node 'A' in the circuit to the ground as shown in **Figure 4.9.b**. Whereas, in the case of PMOS current source current always flows from 'VDD' to any arbitrary node 'A' as shown in **Figure 4.10.b**. Even in saturation, the current is not flat because of the channel length modulation [51]. This phenomenon is represented in **Figure 4.9.c** and **Figure 4.10.c**. In the proposed IMC work channel length modulation [51] is neglected.

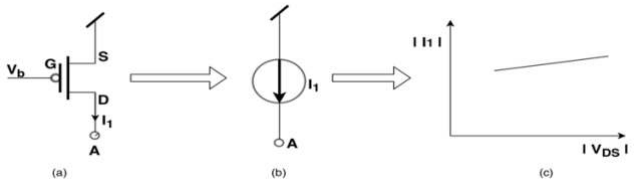


Figure 4.10. (a) Saturated PMOS in saturation operating as a current source, (b) current flow direction, (c) I_1 vs V_{DS} plot of saturated PMOS device [51]

Multiple copies of the current source (NMOS & PMOS) can be created by a technique known as a current mirror. A conventional NMOS current mirror is shown in **Figure 4.11**. 'I_{ref}' is the ideal current source or the global current source that needs to be copied multiple times within a circuit. 'I_{ref}' can be scaled to different values by sizing the dimensions of MOS devices within the current mirror circuit [51].

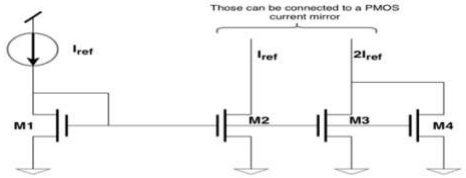


Figure 4.11. Typical NMOS current mirror configuration [51]

In **Figure 4.11**, the transistor M1is known as the diode-connected device. In a diode-connected device, both drain (D) and gate (G) are tied together and it act as a two-terminal device, with one terminal at the drain and other at the source as shown in **Figure 4.12**.

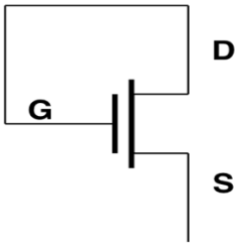


Figure 4.12. An NMOS diode connected device [51]

The reference current (I_{ref}), is normally connected to the shorted terminal between gate and drain. Shorting the gate and drain together ensure that the potential at both the terminals are the same, and hence M1 in saturation. Whenever a reference current (I_{ref}) is injected into the diode connected device, it generates a voltage proportional to the injected 'I_{ref}'. The voltage from the diode connected device is injected into the gate (G) of M2, to make it conduct. The output voltage produced by the diode connected device is used as V_{GS} for M2, to produce an exact copy of I_{ref} at the drain terminal [51].

$$I_{out} = \frac{\left(\frac{W}{L}\right)_2}{\left(\frac{W}{L}\right)_1} I_{ref} \qquad (4.4)$$

Equation 4.4 (adopted from [51] represents the output drain current from M2. Moreover, the equation ensures that I_{ref} is properly copied, so as to generate I_{out}. However, the relationship between I_{ref} and I_{out} is just a constant number. Constant number is just the ratio of $\left(\frac{W}{L}\right)$'s of the MOS devices. Moreover, this ratio ensures that it is not temperature and supply independent. However, one should ensure that the golden current source (I_{ref}) is a constant one, so that I_{out} will be.

4.3.2 : 4-Bit DAC Circuit Design

The fundamental neuron operation is governed by the equation 4.5 (adopted from [10]). The 4-bit DAC circuit deals with summation ($\sum$) and product operation within the brackets of the equation 4.5. Circuit diagram of the proposed 4-bit DAC circuit is shown in **Figure 4.13.a**. During the read operation of 6T-SRAM bit cell, the differential bit-line voltages are sensed by the sense amplifier and amplifies the changes in the bit-lines. The sense amplifier produces a digital output corresponding to the sensed differential voltage from bit-lines BL and BLB. This digital voltage is converted to a current by passing through a MOS device (NMOS).

$$y_j = f\left(\sum_i \left(w_{ij} x_i\right)\right) \qquad (4.5)$$

In **Figure 4.13.a**, the current source $I_{read-sen}$, represents the equivalent current to the digital voltage from the output of sense amplifier. For simulation purpose, a unit cell current of $1\mu A$ is used. For to design a 4-bit DAC that operates exactly as equation 4.5, firstly we need multiple copies of $I_{read-sen}$, whenever a read operation is performed inside 6T-SRAM. A PMOS current mirror is used to create identical copies of $I_{read-sen}$. Transistor M_0 is a diode connected device with drain and gate terminals shorted together. Furthermore, this will ensure M_0 is in saturation. Transistors M_1, M_2, M_3, and M_4 act as exact copies of the current source $I_{read-sen}$.The current $I_{read-sen}$ is converted to an equivalent voltage of $V_{read-sen}$ by the transistor M_0 (diode-connected device).

The voltage $V_{read-sen}$ is used as the V_{GS} for the transistors M_1, M_2, M_3, and M_4. As the voltage $V_{read-sen}$ is injected into the gate of the transistors M_1, M_2, M_3, and M_4, devices satisfying the bias condition starts to move from cut-off region to saturation. This makes transistors M_1, M_2, M_3, and M_4 acts as a binarized current sources (PMOS), and act as exact current copies of $I_{read-sen}$.

The 4-bit DAC circuit also consist of 4-binary weighted NMOS switches constituted by the transistors M_5, M_6, M_7, and M_8. The weight vectors W_0, W_1, W_2, and W_3, are injected into the gates of the transistors M_5, M_6, M_7, and M_8 as shown in **Figure 4.13**. The moment at which

the transistors M_5, M_6, M_7, and M_8, meets its biasing conditions, the corresponding NMOS transistor moves towards linear region and act as a switch. The switch opens and closes according to the weight vectors W_0, W_1, W_2, and W_3 at the gates.

The 4-bit RDAC can have 16-binary weighted combinations, starting from W [3:0] = 4'b0000 to W [3:0] = 4'b1111. In **Figure 4.13.a**, the sizing of the transistors is also mentioned. PMOS transistor M_1 is eight times the width (w) of transistor M_4. Whereas, the width of PMOS transistor M_2 is four times that of M_4. Furthermore, PMOS transistor M_3 is twice the width of M_4. PMOS transistor M_4 is kept as a unit current source. Such a sizing of the transistor ensures that the proposed 4-bit DAC works exactly as a digital to analog converter with sixteen levels of distinct voltage levels at its output.

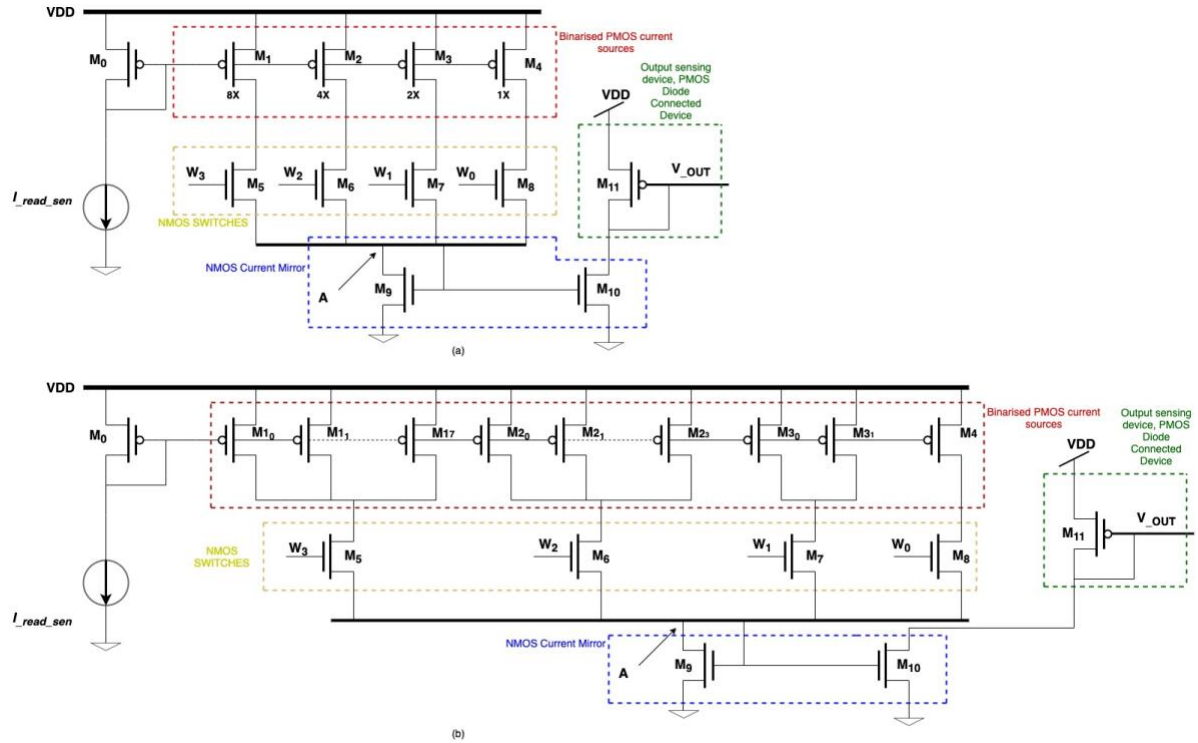


Figure 4.13. (a) 4-bit DAC circuit, (b) actual implementation of 4-bit DAC

Instead of using a transistor sizing as mentioned, the real implementation of 4-bit DAC in cadence tool is shown in **Figure 4.13.b**. The basic idea is to create weighted currents. During the normal working of the circuit, PMOS transistor M_1 should produce $8* I_{read-sen}$, when switch M_5 is on. The same current can be produced by placing eight-unit size PMOS transistors $M_{1_0}$to M_{1_7}in parallel and shorting its drain. Such a configuration produces a current of $8* I_{read-sen}$, whenever the switch M_5 is turned on. PMOS transistor M_2 should produce $4* I_{read-sen}$ and can be achieved by placing $M_{2_0}$to M_{2_3}in parallel as shown in **Figure 4.13.b**. Furthermore, PMOS transistor M_3 should produce $2* I_{read-sen}$. A current of two times $I_{read-sen}$ is achieved by placing the transistors $M_{3_0}$to M_{3_1}in parallel and shorting their drains

as shown in **Figure 4.13.b**. However, transistor M_4 is kept as a unit current source, so that a current of $I_{read-sen}$ flows through it whenever the switch M_8 is turned on.

The current $I_{read-sen}$ after passing through M_0, can flow through mainly four paths. The four possible paths, path_1, path_2, path_3, path_4 are formed according to the weight vectors W_0, W_1, W_2, W_3, and injected into the gates of the respective NMOS transistors.

Path_1: $I_{read-sen}$ current flows from the PMOS binary weighted current source M_4 through NMOS binary weighted switch M_8 carrying a current of $I_{read-sen}$ as shown in **Figure 4.14**.

Path_2: $I_{read-sen}$ current flows from the PMOS binary weighted current source M_3 through NMOS binary weighted current source M_7 carrying a current of $2 * I_{read-sen}$ as shown in **Figure 4.15**.

Path_3: $I_{read-sen}$ current flows from the PMOS binary weighted current source M_2 through NMOS binary weighted current source M_6 carrying a current of $4 * I_{read-sen}$ as shown in **Figure 4.16**.

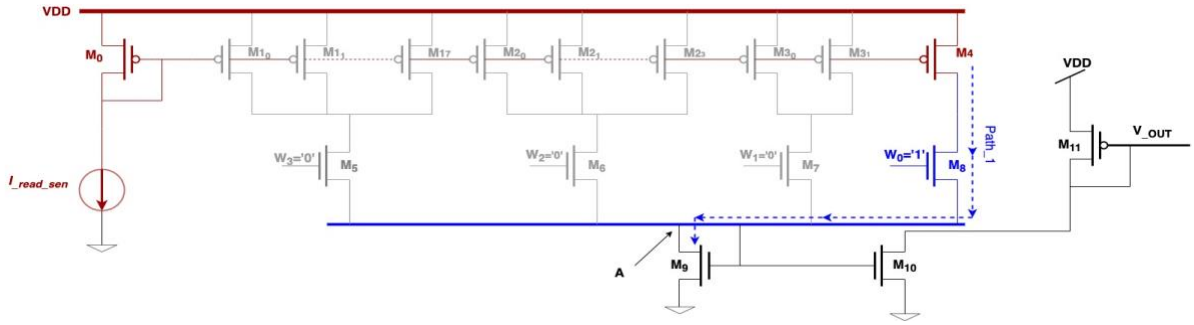


Figure 4.14. Current flow (path_1) in 4-bit DAC circuit when W_0 = '1'

Path_4: $I_{read-sen}$ current flows from the PMOS binary weighted current source M_1 through NMOS binary weighted current source M_5 carrying a current of $8 * I_{read-sen}$ as shown in **Figure 4.17**.

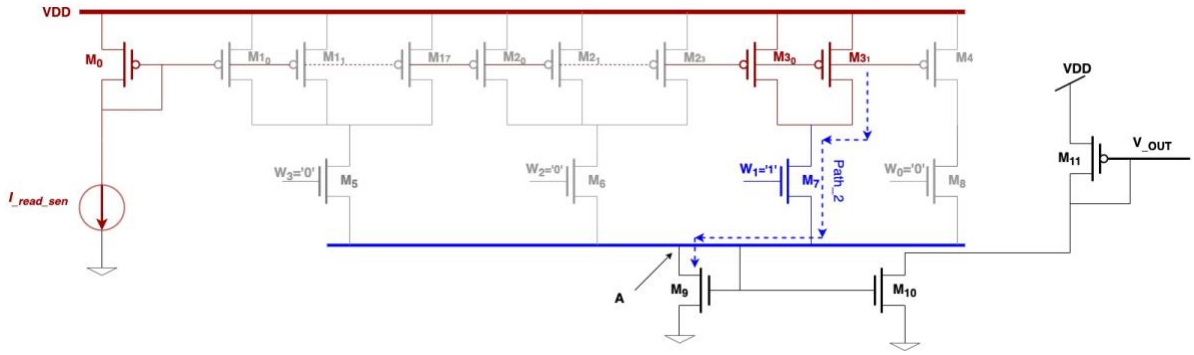


Figure 4.15. Current flow (Path_2) in 4-bit DAC when W_1 = '1'

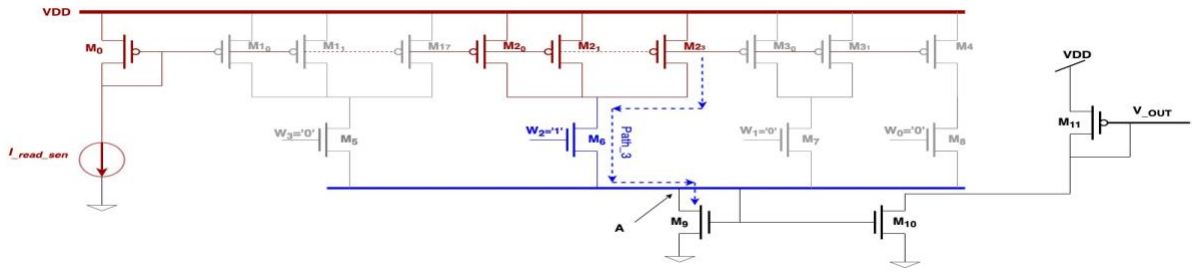


Figure 4.16. Current flow (Path_3) in 4-bit DAC circuit when W_2 = '1'

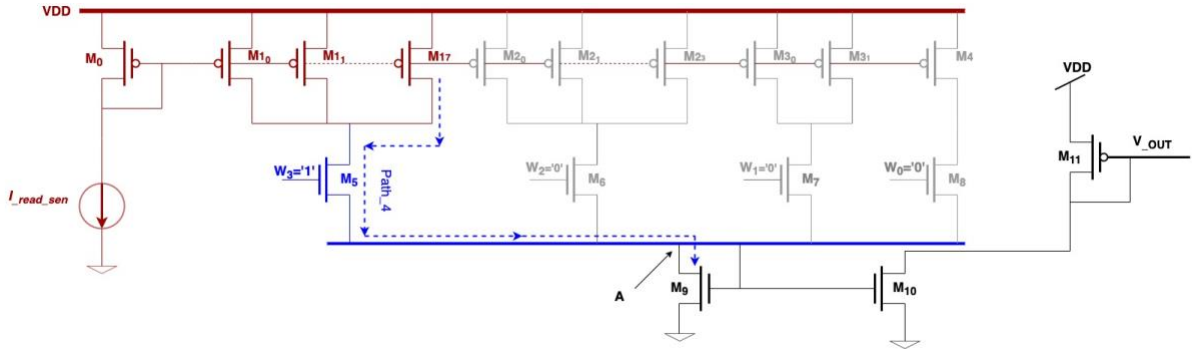


Figure 4.17. Current flow (Path_4) in 4-bit DAC when W_3 = '1'

At node 'A' a current accumulation $(\sum)$ can be identified. The individual currents through the four paths are available at node 'A'. According to the Kirchhoff's current law, the current at node 'A', I_A is equivalent to the sum of four currents arrived at node 'A' from the distinct four paths. Equation 4.6 represents the total current at node 'A' when the weights applied are $W_3W_2W_1W_0 = 1111$.

$$I_A = 8 * I_{read\_sen} + 4 * I_{read\_sen} + 2 * I_{read\_sen} + I_{read\_sen} \quad (4.6)$$

Equation 4.6 is similar to the fundamental neuron equation, that is represented in equation 4.5. Current at node 'A' is comparatively large compared to the individual currents that flows through the distinct four paths. Transistor M_9 is a globally biased NMOS diode connected device. The dimensions of M_9 is adjusted to hold the large current I_A accumulated at node 'A'. This globally biased device produces a voltage V_A, and is injected into the gate of NMOS device M_{10}. The gate voltage V_A and the biasing condition ensures that M_{10} is in saturation. M_{10} now act as a current source, producing an equivalent current I_A, that is produced at node 'A'.

Output of 4-bit DAC circuit is a voltage, sensed by PMOS diode connected device M_{11}. M_{11} is biased from 'vdd'. M_{11} produces an output voltage equivalent to the accumulated current I_A at node 'A'. The output voltage from the diode connected device M_{11} is a word-line (WL) voltage, which could be written back to the output activation memory. In this work, we haven't implemented the write back circuit.

4.4 : Results and Analysis

The I-V characteristics of an NMOS transistor (130nm) is depicted in **Figure 4.18**. **Figure 4.18.a** shows ID vs V_{GS} of an NMOS transistor with width W=280nm and length L=130nm (minimum W & L of the technology used for the experiment). V_{DS} is fixed at 1.2 V and V_{GS} varied from 0 V to 1.2 V. Furthermore, **Figure 4.18.b** shows the plot of parametric analysis for I_D vs V_{GS} of NMOS where V_{DS} is varied from 0 V to 1.2 V in four steps. It is evident from the sweep that the threshold voltage (V_T) of an NMOS transistor (130nm) is 355 mV. At this threshold voltage the drain current (ID) is increasing abruptly. ID vs V_{DS} characteristics for varying V_{GS} for the same dimensions of an NMOS device is shown in **Figure 4.18.c**. V_{GS} is fixed at 1.2 V whereas, V_{DS} is varied from 0 V to 1.2 V. **Figure 4.18.d** shows the parametric analysis for ID vs V_{DS} for the same NMOS device with V_{GS} fixed at 1.2 V and V_{GS} varied from 0 V to 1.2 V in 4 steps. It is also evident from the parametric analysis that drain current saturates at an overdrive voltage of $V_{GS} - V_{TH}$.

I-V characteristics for a PMOS device (130nm) of dimensions (W=280nm, L=130nm) is depicted in **Figure 4.19**. I_D vs V_{GS} of the PMOS device where V_{DS} is fixed at 1.2 V and V_{GS} is varied from 0 V to 1.2 V is shown in **Figure 4.19.a**. **Figure 4.19.b** shows a parametric analysis conducted in four steps for the I_D vs V_{GS} of a PMOS device where V_{DS} is varied from 0 V to 1.2 V, keeping V_{GS} fixed at 1.2 V. From this experiment the threshold voltage of the PMOS device (130nm) is identified as 300 mV. Both the plots for I_D vs V_{GS} elucidates that the drain current is getting rapid increase at 300 mV. I_D vs V_{DS} for different values of V_{GS} (0 V to 1.2 V), where V_{DS} is changed from 0 V to 1.2 V is plotted in **Figure 4.19.c**. A parametric analysis for the same I_D vs V_{DS} of the PMOS device with varying V_{GS} from 0 V to 1.2 V in four steps is shown in **Figure 4.19.d**. The drain current of the PMOS device saturates at an overdrive voltage of $V_{GS} - V_{TH}$.

The operation of the proposed 4-bit DAC circuit configuration for performing dot operation was simulated using Cadence Spectre on 130nm CMOS process. V_{DD} of 1.2 V is used for the entire circuit analysis. The main components of the 4-bit DAC implementation are Read-Sense current ($I_{read-sen}$), of $1\mu A$, equivalent to the current drawn from sense amplifier of the 6-T SRAM, weight vectors applied to gates of NMOS transistors (input voltages) and the conductance of the transistors according to the various states of read sense current and input voltages (weights).

Consider I_D vs V_{DS} plot of NMOS and PMOS in **Figure 4.18.c** and **Figure 4.19.c**. In general, we expect a parabolic path for I_D curve. But as depicted in **Figure 4.18.c** and **Figure 4.19.c** it did not followed the parabolic path. The characteristic started as a parabola, the current attains the

maximum and stayed as a constant. Furthermore, on close observation the current I_D is not actually a constant. In reality the curve goes up by a small amount (red colour) as we increase V_{DS} and is almost a straight line. The same issue is depicted in **Figure 4.9.c** and **Figure 4.10.c**. In **Figure 4.9.a**, the source of NMOS transistor is grounded gate and drain has some amount of voltage so that the device is in saturation. Device in saturation has a pinch-off point. Furthermore, the voltage at the drain end is higher and pinch-off point has to move back a little from its initial point. In other words, as V_{DS} increases the channel length (L) decreases by a small amount. This enables the drain current I_D to increase by a small amount. This behaviour is known as channel length modulation [51].

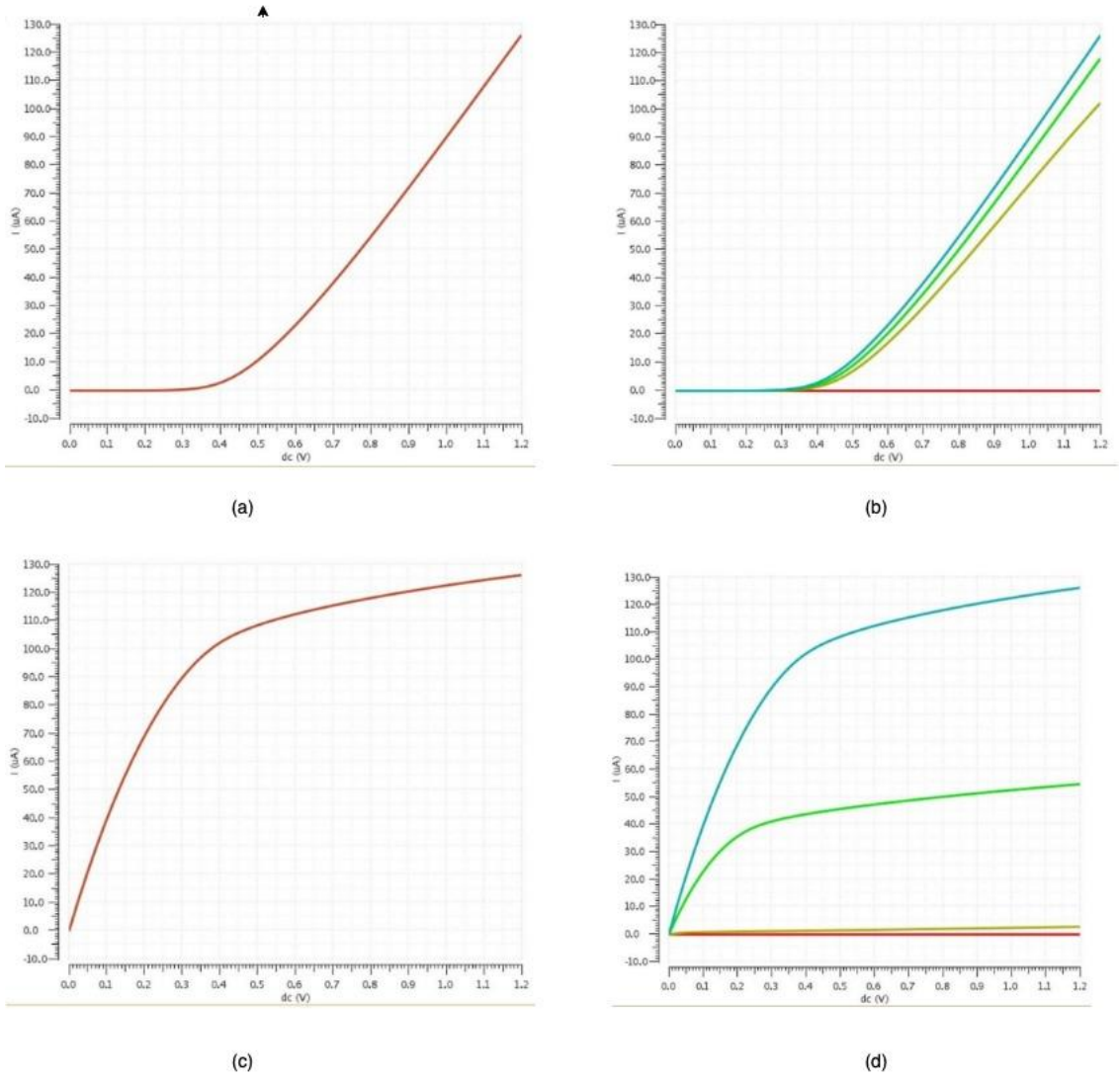


Figure 4.18. (a) I_D vs V_{GS} of an NMOS transistor with width W=280nm and length L=130nm, V_{DS} is fixed at 1.2 V and V_{GS} varied from 0 V to 1.2 V. (b) Parametric analysis for I_D vs V_{GS} of NMOS where V_{DS} is varied from 0 V to 1.2 V in four steps. (c) I_D vs V_{DS} characteristics for different values of V_{GS} for the same dimensions of an NMOS device, V_{GS} is fixed at 1.2 V whereas, V_{DS} is varied from 0 to 1.2 V. (d) Parametric analysis for I_D vs V_{DS} for the same NMOS device with V_{GS} fixed at 1.2 V and V_{GS} varied from 0 V to 1.2 V in 4 steps.

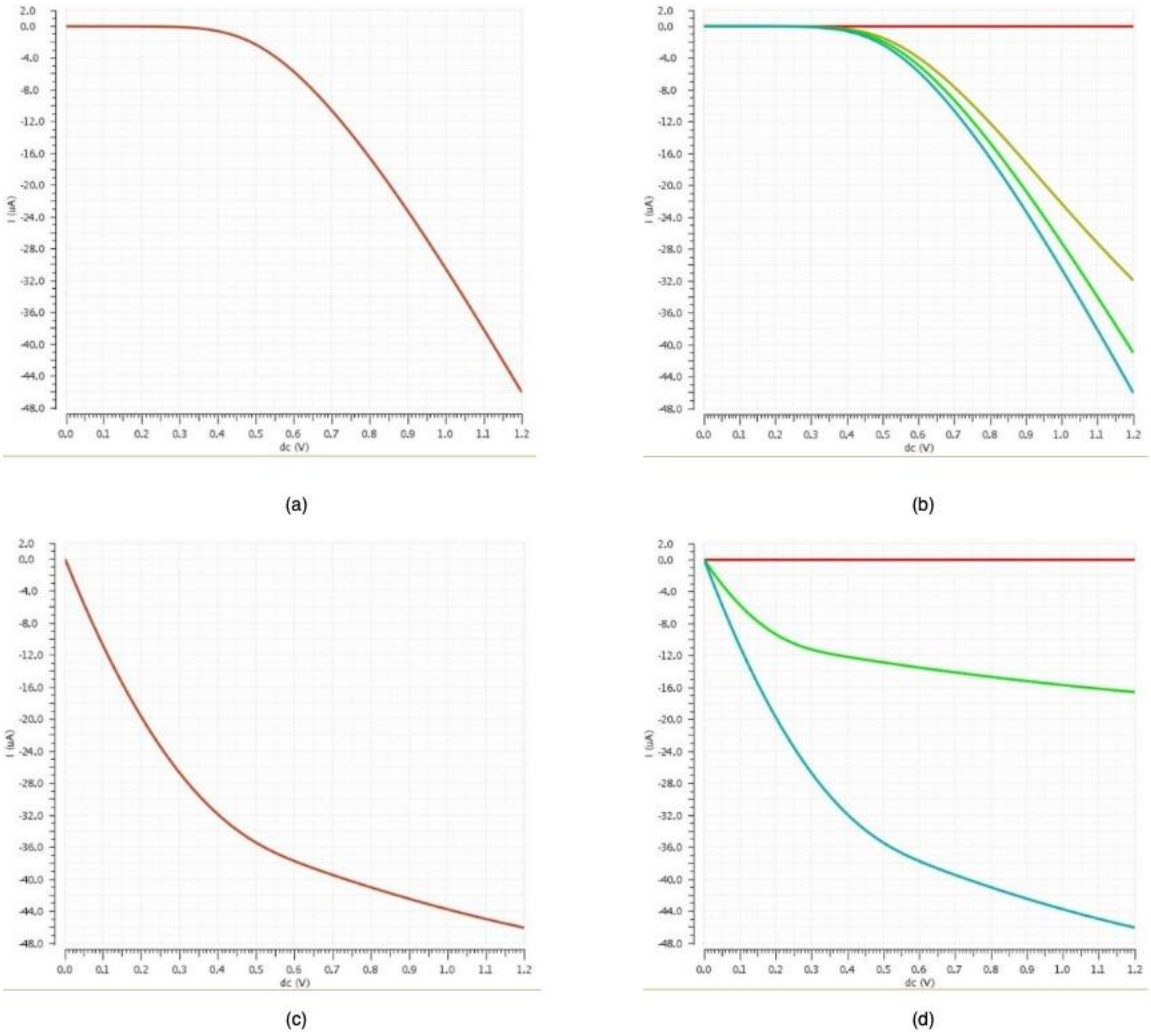


Figure 4.19. (a) I_D vs V_{GS} of an PMOS transistor with width W=280nm and length L=130nm, V_{DS} is fixed at 1.2 V and V_{GS} varied from 0 V to 1.2 V. (b) Parametric analysis for I_D vs V_{GS} of NMOS where V_{DS} is varied from 0 V to 1.2 V in four steps. (c) I_D vs V_{DS} characteristics for different values of V_{GS} for the same dimensions of an NMOS device, V_{GS} is fixed at 1.2 V whereas, V_{DS} is varied from 0 V to 1.2 V. (d) Parametric analysis for I_D vs V_{DS} for the same NMOS device with V_{GS} fixed at 1.2 V and V_{GS} varied from 0 V to 1.2 V in 4 steps.

Table 2 shows the 4-bit DAC device dimensions used in the design. The length of all devices is kept constant. As mentioned earlier, unit cell PMOS current sources are created using a device dimension of $W = 560nm$ and $L = 130nm$. The NMOS switch M_5 needs to be wider compared to the other switches so as to operate in linear region. The globally biased NMOS diode connected device has a device dimension of $W = 4.2\ \mu m$ and $L = 130nm$. The output capacitance seen at the output of the circuit shown in **Figure 4.13.b** is negligible. So, the width of both the device M_{10} & M_{11} are selected to be $W/15^{th}$ of the actual needed width. If the output capacitance is large enough, it is also possible to increase the width of these devices so that the device can source or sink more current.

The working of the 4-bit DAC is verified mainly using four worst case 4-bit weight configurations. Using the weight vectors $(W_3W_2W_1W_0) = 0000, 0100, 1001, 1111$ the operation

of the circuit is analysed. Firstly, **Figure 4.20** shows the experimental behaviour of 4-bit DAC circuit, when the binary weights are assigned as $W_3W_2W_1W_0$ = 0000 and $I_{read-sen} = 1\mu A$. **Figure 4.20.a** shows the plot of measured drain current from diode connected device M_{11} (ID_{M11}) when the weights $W_3W_2W_1W_0$ = 0000 is varied from 0 V to 0.2 V. .2 V is not enough to make the NMOS transistors $M_{5,}M_{6,}\ M_{7,}\ M_8$ to turn on. The maximum drain current ID_{M11} measured is -667.5 pA. **Figure 4.20.b** shows plot of output voltage (V_{OUT}) sweep across the weights $W_3W_2W_1W_0$ = 0000 from 0 V to .2 V. A maximum voltage of 1.089 V is measured across the sensing device M_{11} when the weights are ($W_3W_2W_1W_0$) = 0000.

Table 2. 4-bit DAC device dimensions used in the design

Device	Width (W)	Length (L)	No. of Fingers
M_0	280nm	130nm	2
M_{1_0}	280nm	130nm	2
M_{1_1}	280nm	130nm	2
M_{1_2}	280nm	130nm	2
M_{1_3}	280nm	130nm	2
M_{1_4}	280nm	130nm	2
M_{1_5}	280nm	130nm	2
M_{1_6}	280nm	130nm	2
M_{1_7}	280nm	130nm	2
M_{2_0}	280nm	130nm	2
M_{2_1}	280nm	130nm	2
M_{2_2}	280nm	130nm	2
M_{2_3}	280nm	130nm	2
M_{3_0}	280nm	130nm	2
M_{3_1}	280nm	130nm	2
M_4	280nm	130nm	2
M_5	280nm	130nm	1
M_6	280nm	130nm	1
M_7	280nm	130nm	1
M_8	280nm	130nm	3
M_9	280nm	130nm	15
M_{10}	280nm	130nm	1
M_{11}	280nm	130nm	2

Secondly, the response of the 4-bit DAC circuit for a weight combination ($W_3W_2W_1W_0$) = 0100 was analysed. **Figure 4.21.a** is the plot of the drain current (ID_{M11}) vs the weights ($W_3W_2W_1W_0$) = 0100. The gate voltages corresponding to the weights $W_3W_2W_0$ are varied from 0 V to .2 V in four steps for the binary weights '0', so that the NMOS transistors

M_5, M_6, M_8 turns off. Furthermore, the binary weight '1' is applied to the gate of M_7 as a voltage, which is varied from 0 V to 1.2 V in four steps for the analysis. The maximum drain current ID_{M11} flowing through the drain of sensing device M_{11} while the weights applied are 0100 is -868.8 nA. **Figure 4.21.b** shows the plot of output voltage (V_{OUT}) vs the weights ($W_3W_2W_1W_0$) = 0100. The same experimental setup used for measuring ID_{M11} is used for measuring (V_{OUT}) as well. The maximum V_{OUT} measured during the parametric sweep while the weights are increased in four steps is 791.3 mV.

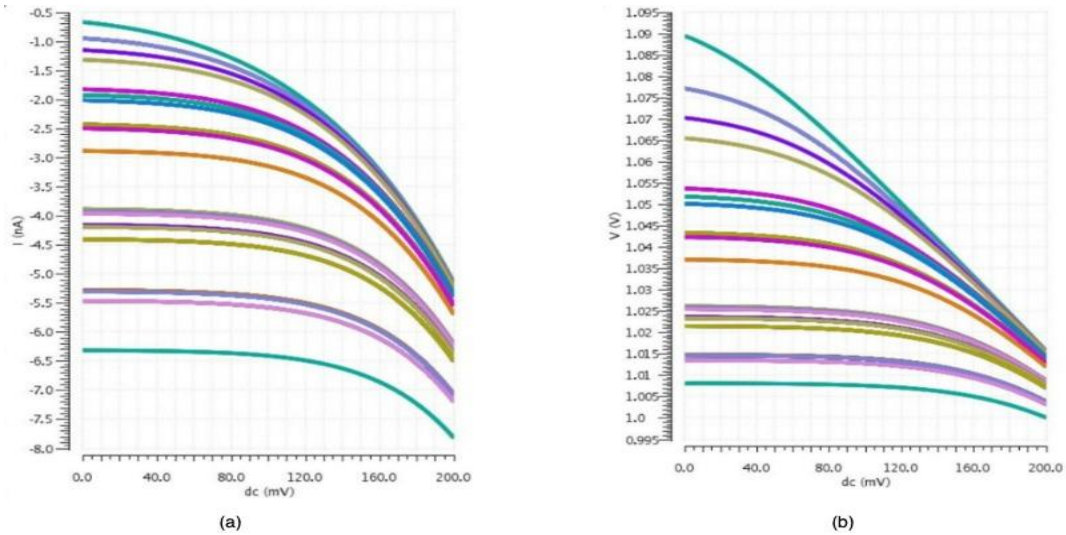


Figure 4.20. (a) plot of measured drain current from diode connected device M_{11} (ID_{M11}) when the weights $W_3W_2W_1W_0$ = 0000 is varied from 0 V to 0.2 V. (b) plot of output voltage (V_{OUT}) sweep across the weights $W_3W_2W_1W_0$ = 0000 from 0 V to .2 V

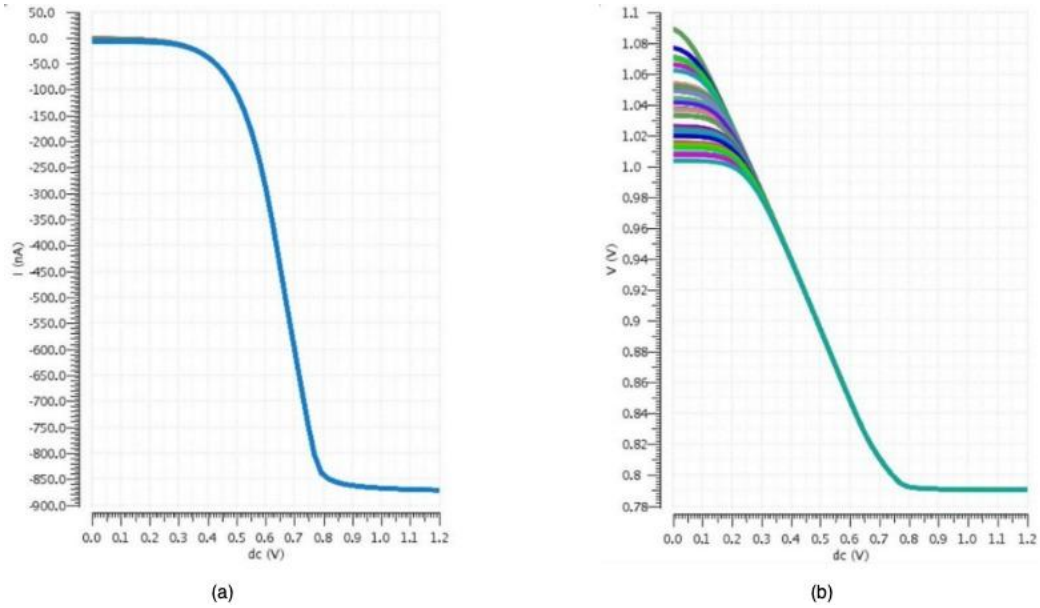


Figure 4.21. (a) Plot of the drain current (ID_{M11}) vs the weights ($W_3W_2W_1W_0$) = 0100. Gate voltages corresponding to the weights $W_3W_2W_0$ are varied from 0 V to .2 V in four steps for the binary weights '0' and for binary weight '1' gate voltage is varied from 0 V to 1.2 V in four steps. (b) Plot of the output voltage V_{Out} vs the weights ($W_3W_2W_1W_0$) = 0100. Gate voltages corresponding to the weights $W_3W_2W_0$ are varied from 0 V to .2 V

in four steps for the binary weights '0' and for binary weight '1' gate voltage is varied from 0 V to 1.2 V in four steps

The third case examined during the experimental phase is the behaviour of 4-bit DAC circuit for the weight combination $(W_3W_2W_1W_0)$ = 1001. Injecting those weights to the gates of the NMOS transistors $M_{5,}M_{6,}$ $M_{7,}$ M_8 makes $M_{5,}M_8$ on while, $M_{6,}M_7$ off. **Figure 4.22.a** shows the plot of drain current ID_{M11}measured when the gates of transistors $M_{6,}M_7$ varied in four steps from 0 V to .2 V, which is equivalent to a binary weight '0'. Furthermore, the gate voltages of transistors $M_{5,}M_8$ is varied from 0 V to 1.2 V in four steps which is equivalent to binary weight '1'. The maximum drain current (ID_{M11}) measured is -1.649 μA. The output voltage (V_{OUT}) measured across M_{11} is shown in **Figure 4.22.b**, where the maximum measured voltage is 753 mV.

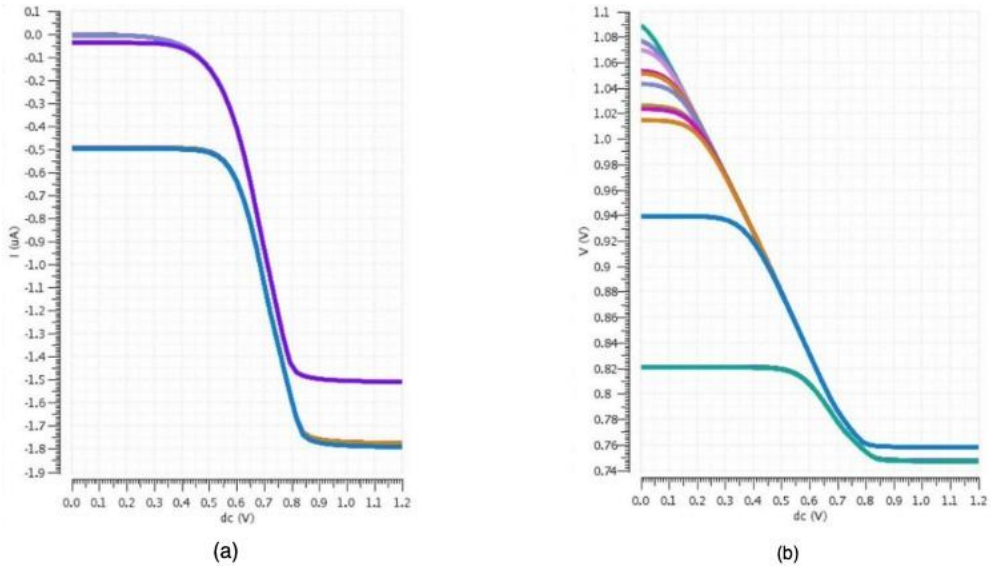
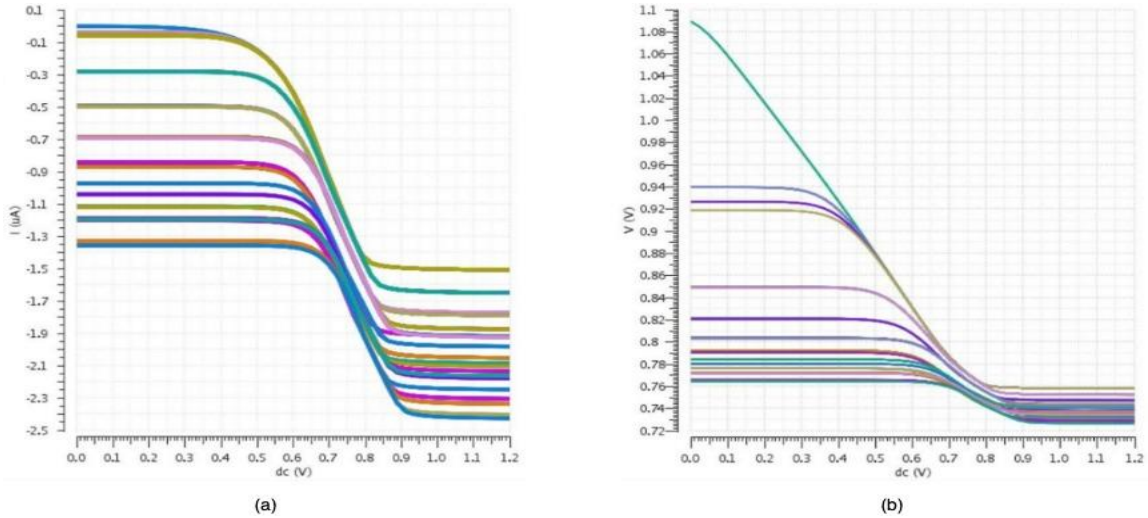


Figure 4.22. (a) Plot of the drain current (ID_{M11}) vs the weights $(W_3W_2W_1W_0)$ = 1001. Gate voltages corresponding to the weights W_2W_1 are varied from 0 V to .2 V in four steps for the binary weights '0' and for binary weight '1'(W_3W_0) gate voltage is varied from 0 V to 1.2 V in four steps. (b) Plot of the output voltage V_{Out} vs the weights $(W_3W_2W_1W_0)$ = 1001. Gate voltages corresponding to the weights W_2W_1 are varied from 0 V to .2 V in four steps for the binary weights '0' and for binary weight '1'(W_3W_0) gate voltage is varied from 0 V to 1.2 V in four steps

The final case is the worst-case weight combination $(W_3W_2W_1W_0)$ = 1111. Those weights make all the NMOS transistors $M_{5,}M_{6,}$ $M_{7,}$ M_8 turned on. So, the total drain current measured is the sum of the individual current flowing through the four individual paths. **Figure 4.23.a** shows the plot of the drain current ID_{M11}, while the gates of the NMOS transistors $M_{5,}M_{6,}$ $M_{7,}$ M_8 varied from 0 V to 1.2 V in four steps, which is equivalent to a binary '1'. During this phase of experiment V_{DD} is kept constant at 1.2 V. The maximum drain current ID_{M11} measured is -2.421 μA.

Figure 4.23.b shows the plot of measured output voltage (V_{OUT}) across M_{11}, while V_{DD} is kept constant at 1.2 V and gate voltages of transistors M_5, M_6, M_7, M_8 varied from 0 V to 1.2 V in four steps. The maximum measured output voltage is 727.3 mV.

The experimental analysis of the 4-bit DAC circuit shows that the drain current ID_{M11} is linear according to the applied gate voltages (weights) of the NMOS transistors. **Figure 4.24** shows a plot of the drain current $|ID_{M11}|$ for the 2^4 =16 weight combinations starting from 0000 to 1111. The least drain current measured is .0067 μA for the weight combination 0000.



(a) (b)

Figure 4.23. (a) plot of measured drain current from diode connected device M_{11} (ID_{M11}) when the weights $W_3W_2W_1W_0$ = 1111 is varied from 0 V to 1.2 V. (b) plot of output voltage (V_{OUT}) sweep across the weights $W_3W_2W_1W_0$ = 1111 from 0 V to 1.2 V

Even though the read sense current ($I_{read-sen}$) applied is 1μA, the binary weighted current sources M_1, M_2, M_3, M_4 deviates from 1μA and produce a higher current of 1.36 μA. This deviation is mainly due to the channel length modulation. The same issue is occurring at the NMOS current mirror used to copy the accumulated current. Replacing current mirrors with cascode current mirror can eliminate this issue to an extent. However, a drain current equivalent to zero is expected for the weight combination 0000. Even though the NMOS transistors M_5, M_6, M_7, M_8 are turned off while weight 0000 are injected to their gates, a small leakage current flows through the circuit.

Figure 4.24 can be interpreted in four ways. Firstly, for the weights ($W_3W_2W_1W_0$) = 0001, 0010, 0100, 1000 any one of the transistor M_5, M_6, M_7, M_8 are on at the same time, letting the current to flow through one of the active path and hence an accumulation of current. The maximum current will get accumulated for the weight vector ($W_3W_2W_1W_0$) = 1000. Furthermore, this weight vector will turn on the NMOS switch M_5. Whenever, M_5 is turned on a current of $8 * I_{read-sen}$ flows through the circuit.

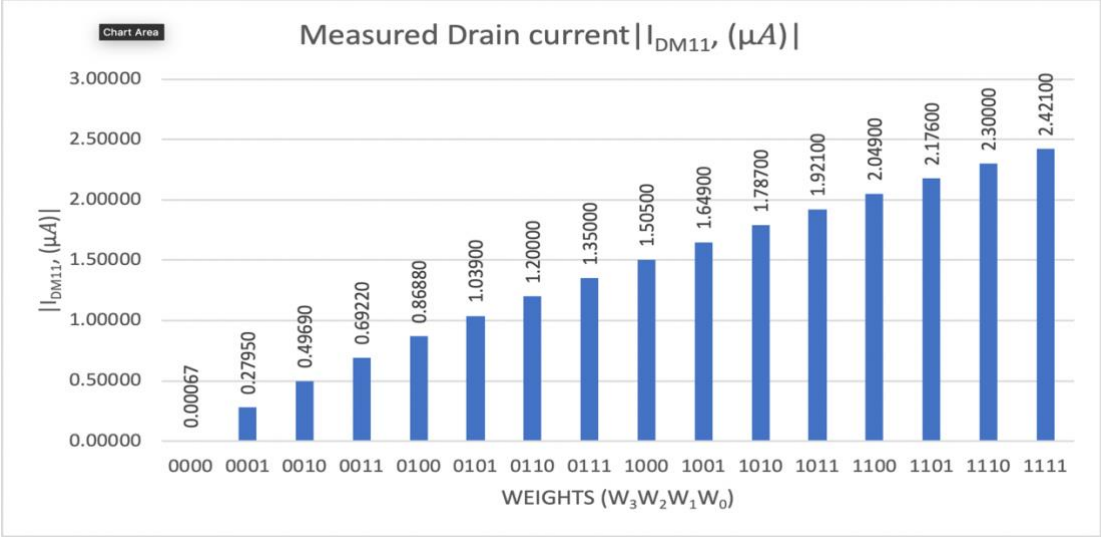


Figure 4.24. shows a plot of the drain current $|ID_{M11}|$ for the 2^4 =16 weight combinations starting from 0000 to 1111.

Secondly, for the weight combination $(W_3W_2W_1W_0)$ = 0011, 0101, 0110,1001,1010,1100 six distinct current get accumulates for all the six-weight combination. This phenomenon is due to the fact that, any two NMOS switches out of $M_{5,}M_{6,}$ $M_{7,}$ M_8 are turned on at the same time. Maximum current flows for the weight vector $(W_3W_2W_1W_0)$ = 1100 whereas, least current flows for the weight combination $(W_3W_2W_1W_0)$ = 0011. This is due to the fact that, for the weight vector $(W_3W_2W_1W_0)$ = 1100, a current of $12 * I_{read-sen}$ get accumulated. However, only $3 * I_{read-sen}$ flows through the circuit when weight vector $(W_3W_2W_1W_0)$ = 0011 is applied.

Third interpretation is associated with the weight combination $(W_3W_2W_1W_0)$ = 0111, 1011, 1101, 1110. In this case any three NMOS switches among $M_{5,}M_{6,}$ $M_{7,}$ M_8 are turned on at the same time according to the weight vector. A maximum current of $14 * I_{read-sen}$ flows through the circuit when the weight vector is $(W_3W_2W_1W_0)$ = 1110. Finally, for the weight combination $(W_3W_2W_1W_0)$ = 1111, all the NMOS switches $M_{5,}M_{6,}$ $M_{7,}$ M_8 are turned on at the same time and hence a current of $15 * I_{read-sen}$ conducts through the circuit leading to a maximum current of $|ID_{M11}|$ = 2.42100µA.

The output voltage (V_{OUT}) across the 4-bit DAC circuit is measured and plotted for the 2^4 =16 weight combinations as shown in **Figure 4.25.** Sixteen distinct voltage levels are measured for the 16 weight combinations. The maximum voltage (V_{OUT}) of 1.089 V is measured for the weight combination $(W_3W_2W_1W_0)$ = 0000, as all the NMOS transistors are turned off. Hence the whole V_{DD} is dropped across the sensing device M_{11}. Minimum voltage (V_{OUT}) of 727.3 mV is measured for the weight combination $(W_3W_2W_1W_0)$ = 1111. This is due to the fact that

all the NMOS transistors $M_{5,}M_{6,}$ $M_{7,}$ M_8 are turned on. It is also evident that the output voltage primarily depends upon the weights applied to the gates of the NMOS transistors $M_{5,}M_{6,}$ $M_{7,}$ M_8. For example, the weight combination $(W_3W_2W_1W_0)$ = 0001, 0010, 0100, 1000 produce four different output voltages (V_{OUT}). In this case any one of the NMOS transistor $M_{5,}M_{6,}$ $M_{7,}$ M_8 is turned on at the same time. Whereas for the weights $(W_3W_2W_1W_0)$ = 0011, 0101, 0110, 1001, 1010, 1100 any two NMOS switches among $M_{5,}M_{6,}$ $M_{7,}$ M_8 is turned on at the same time and hence produce six distinct output voltages. Maximum voltage is produced for the weight combination $(W_3W_2W_1W_0)$ = 1100 when a current of $12 * I_{read-sen}$. Furthermore, for the weight combination $(W_3W_2W_1W_0)$ = 0111, 1011, 1101, 1110 three NMOS transistors among $M_{5,}M_{6,}$ $M_{7,}$ M_8 are turned on at the same time producing four different output voltages. A maximum voltage of 730.9 mV is produced for the weight vector combination $(W_3W_2W_1W_0)$ = 1110.

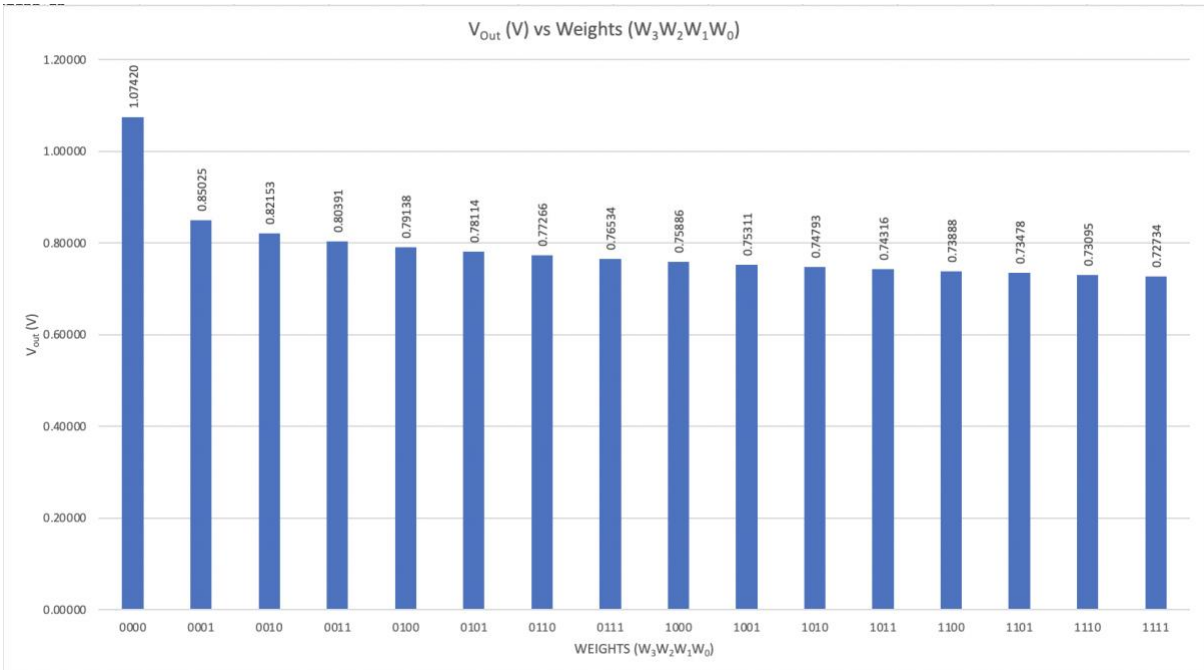


Figure 4.25. shows a plot of the output voltage V_{Out} for the 2^4 =16 weight combinations starting from 0000 to 1111

Transient analysis for the proposed 4-bit DAC is performed for 4000ns. The V_{DD} is kept constant at 1.2 V. A read sense current $(I_{read-sen})$ = 1μA is used. All the weights $(W_3W_2W_1W_0)$ is applied as a pulse with upper voltage swing of 1.2 V and lower voltage swing of 0 V. The period of the weight vector W_0 is 400ns with a pulse width of 200ns. For the weight vector W_1 the period is defined as 800ns with a pulse width of 400ns. Weight vector W_2 is applied for a period of 1600ns with a pulse width of 800ns. Finally, the weight vector W_3 is defined for a

period of 3200ns with a pulse width of 1600ns. Such a combination helps to create all the weight vectors starting from ($W_3W_2W_1W_0$) = 0000 to ($W_3W_2W_1W_0$) = 1111.

Figure 4.26.a shows the transient analysis for the proposed 4-bit DAC for 4000ns with a rise time and fall time of the weight vectors ($W_3W_2W_1W_0$) defined as 5% (20ns) of the period of the weight pulse W_0. Whereas, **Figure 4.26.b** represents a transient analysis of the 4-bit DAC with rise time and fall time defined as 1ps for all the weight vectors ($W_3W_2W_1W_0$). It is evident from the transient analysis that the designed 4-bit DAC is working as expected with current accumulation and voltage spike at each instant when the weight pulse makes a transition. It is evident from the **Figure 4.26.a** and **Figure 4.26.b**, that sixteen different voltage steps are happening throughout the transient analysis. This confirms the working of 4-bit DAC as a perfect digital to analog converter.

The power analysis for the proposed 4-bit DAC is performed to measure the static power and total power consumption. The static power for the proposed circuit is measured by applying the weights ($W_3W_2W_1W_0$) as a static (DC) input, so that no switching activity will take place in the circuit. The static power consumption of the 4-bit DAC is measured for two weight combination. Firstly, when the weights ($W_3W_2W_1W_0$)= 1111, where all the NMOS switches M_5, M_6, M_7, M_8 are conducting and hence the maximum current accumulation happens. Furthermore, to measure the static power in this case, the gate voltages of NMOS switches M_5, M_6, M_7, M_8 are maintained at 1.2 V. The current drawn by the 4-bit DAC circuit for this weight combination is measured as i(A)= $2.2778755E^{-05}$. Therefore, a static power of $2.6559511E^{-05}$ ($P_{Sat} = VI$) is consumed by the 4-bit DAC circuit for the weight combination ($W_3W_2W_1W_0$)= 1111.

Secondly, the static power drawn by the 4-bit DAC circuit for the weight combination ($W_3W_2W_1W_0$)= 0000 is measured. In this case the gate voltages of the NMOS switches M_5, M_6, M_7, M_8 are maintained at 0 V, since the applied weights are 0000. This makes all the NMOS switches turned off and hence no current conduction and accumulation. The current drawn by the 4-bit DAC circuit from the supply voltage (V_{DD}) of 1.2 V is, measured as i(A) = $1.00051E^{-06}$. A static power (P_{sat}) of $4.2562073E^{-07}$ is consumed by the circuit for the weight combination ($W_3W_2W_1W_0$)= 0000.

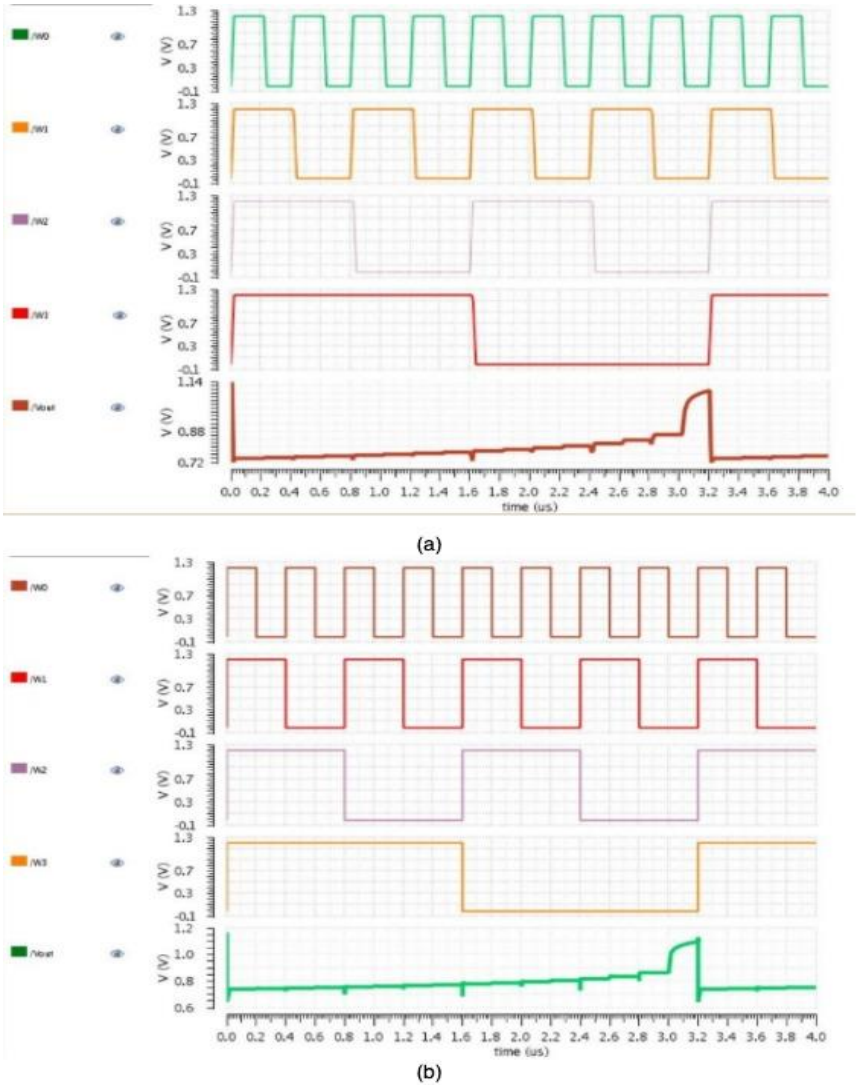


Figure 4.26. (a) transient analysis for the proposed 4-bit DAC for 4000ns with a rise time and fall time of the weight vectors ($W_3W_2W_1W_0$) defined as 5% (20ns) of the period of weight pulses W_0. (b) transient analysis of the 4-bit RDAC with rise time and fall time defined as 1ps for all the weight vectors ($W_3W_2W_1W_0$)

Finally, the total power consumption of the 4-bit DAC is also measured by running a transient analysis for 4000ns. The weights are applied as pulse mentioned earlier. The supply voltage (V_{DD}) is kept constant at 1.2 V. In order to calculate the total power consumption, the output voltage V_{out} of the circuit is measured after running the transient analysis. The measured output voltage is plotted as V_{out} (*/vout* (green signal)) and is shown in **Figure 4.27**. The transient power for the corresponding V_{out} is measured and plotted as shown in **Figure 4.27** (*/pwr* (red signal)). Furthermore, *'pwr'* signal is clipped for a duration of 0ns to 3200ns and is averaged (yellow signal) to calculate the total power consumption and is equal to $13.96E^{-6}$.

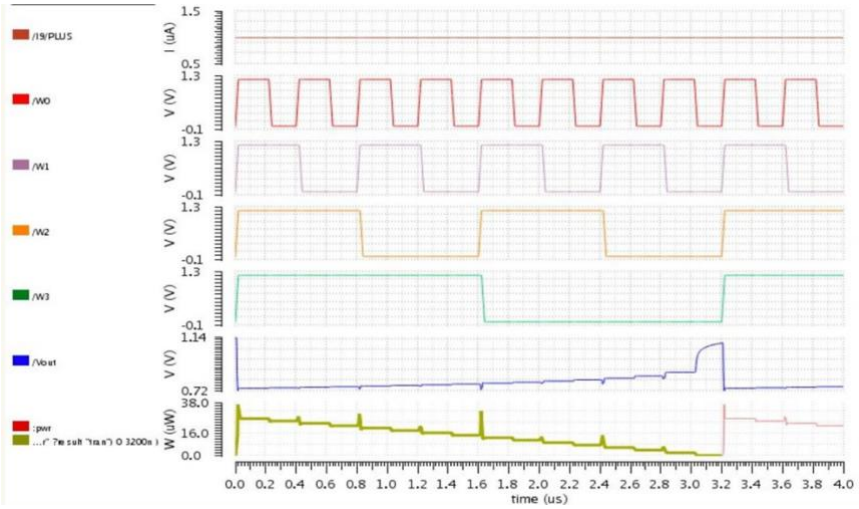


Figure 4.27. Plot of measured power for a transient analysis of 4000ns. V_{out} signal (blue colour) represents the output voltage for the proposed 4-bit DAC. $'pwr'$ signal (yellow colour) represent the average power

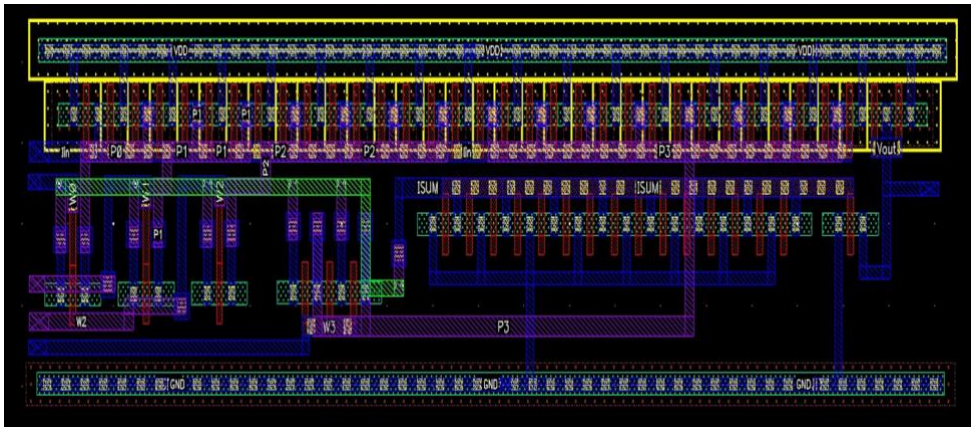


Figure 4.28. Layout of the designed 4-bit DAC

Layout of the designed 4-bit DAC circuit is shown in **Figure 4.28**. A compact layout design consuming a size of $20.34\mu m * 4.73\mu m$. The binary weighted PMOS current sources M_1, M_2, M_3, M_4 and the diode connected device M_0 are placed in same horizontal line. The NMOS switches M_5, M_6, M_7, M_8 are also placed in the same horizontal line maintaining minimum DRC rules. Guard rings are created for V_{DD} and G_{ND}. For the entire 4-bit DAC circuit layout only three metal layers are used.

Figure 4.29 shows the layout of the circuit with M_1 metal routing. M_1 routing is only done in horizontal direction whereas M_2metal routing is performed in the vertical direction as shown in **Figure 4.30**. M_3 metal routing which is used only for the accumulation net in 4-bit DAC and is depicted in **Figure 4.31**.

The NMOS current mirror comprising globally biased diode-connected device M_9 and globally biased NMOS M_{10} are placed adjacent to each other without violating the DRC rules. Finally, the sensing device M_{11} is also placed adjacent to the binary-weighted PMOS current sources

as shown in **Figure 4.28**. The routing of weights ($W_3W_2W_1W_0$) and read sense current ($I_{read-sen}$), are performed in fashion such that they can be applied to the left side of the 4-bit DAC. Furthermore, the output of the circuit is routed to the right side of the layout.

However, the layout of the designed 4-bit DAC shown in **Figure 4.28** will have matching issues if taped out (binary-weighted current sources won't match well). This issue of layout matching can be improved by the common centroid method.

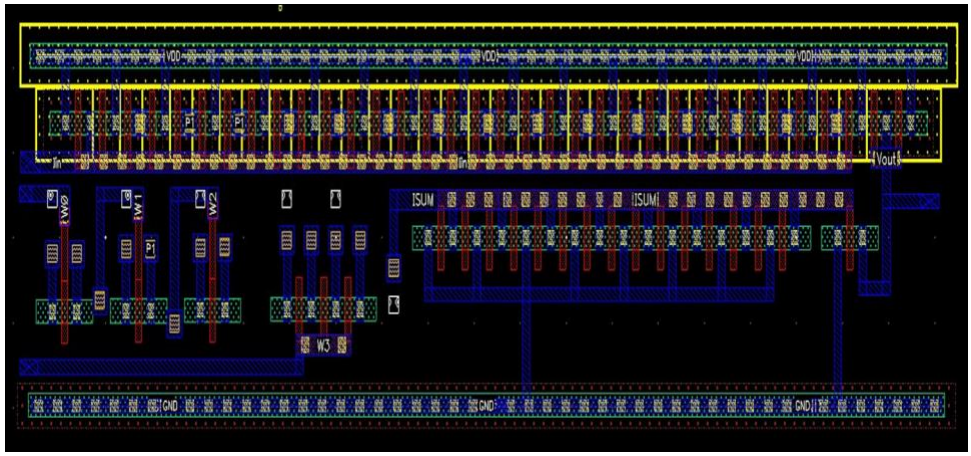


Figure 4.29. Layout of the 4-bit DAC circuit with M_1 metal routing

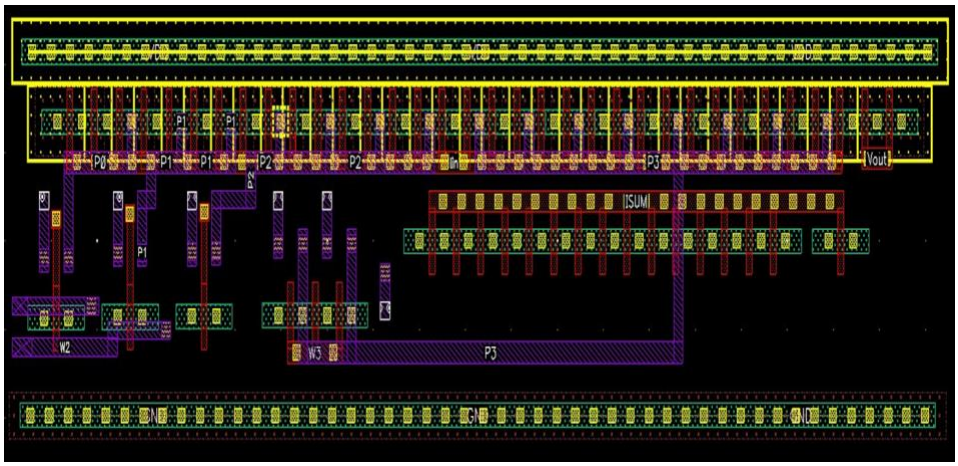


Figure 4.30. Layout of the 4-bit DAC circuit with M_2 metal routing

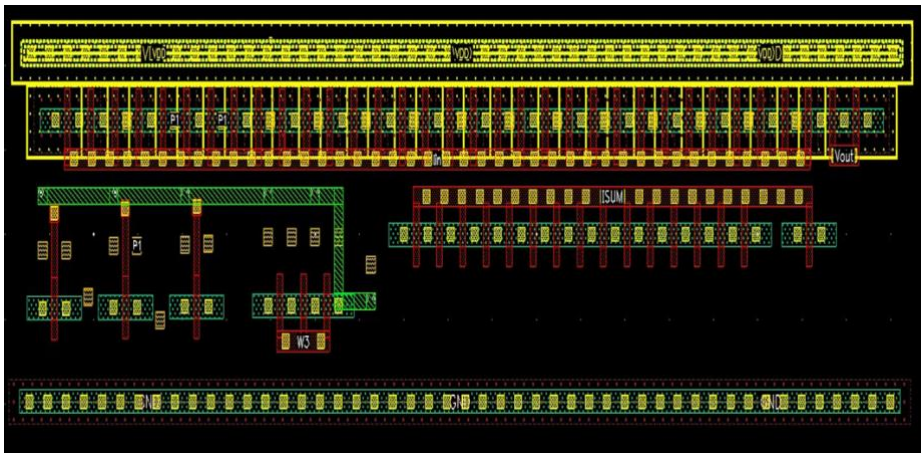
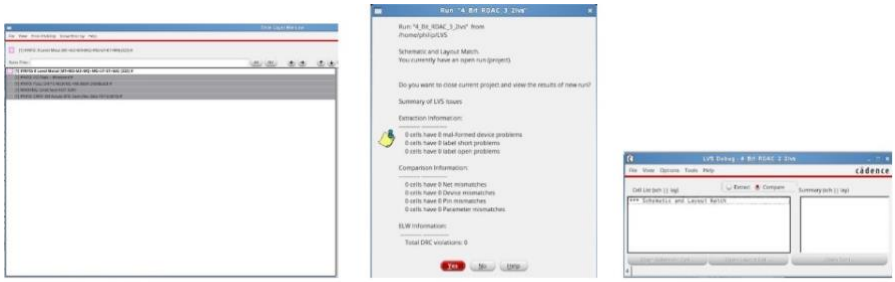


Figure 4.31. Layout of the 4-bit DAC circuit with M_3 metal routing

Figure 4.32.a shows the DRC check results and **Figure 4.32.b** depicts the summary of the LVS check. Furthermore, LVS debug check is depicted in **Figure 4.32.c**. **Figure 4.33** shows the basic test bench circuit used for verifying the functionality of the proposed circuit during post-layout simulation.



(a) (b) (c)

Figure 4.32. DRC check results for 4-bit DAC, (b) summary of the LVS check, (c) LVS debug check

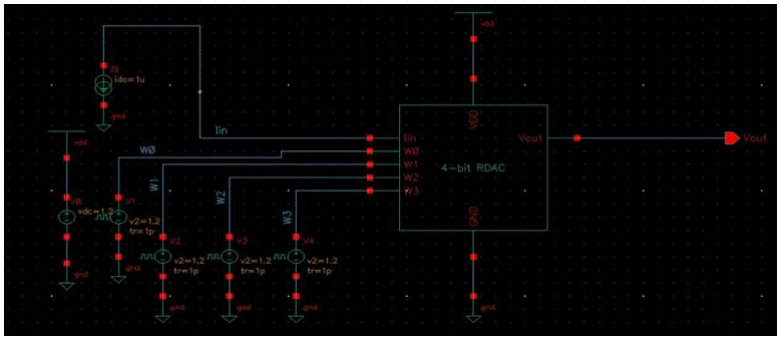
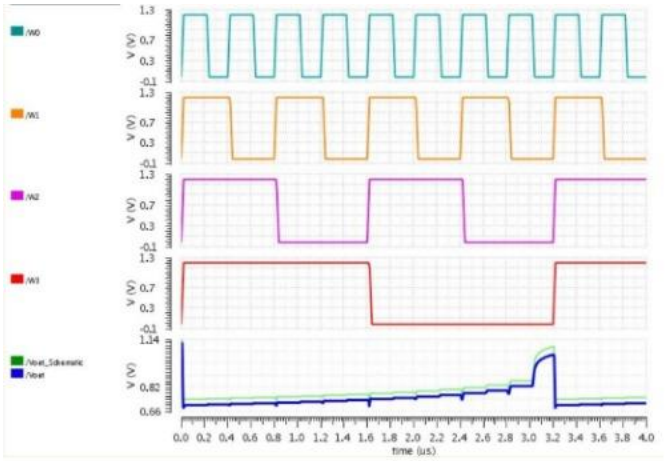


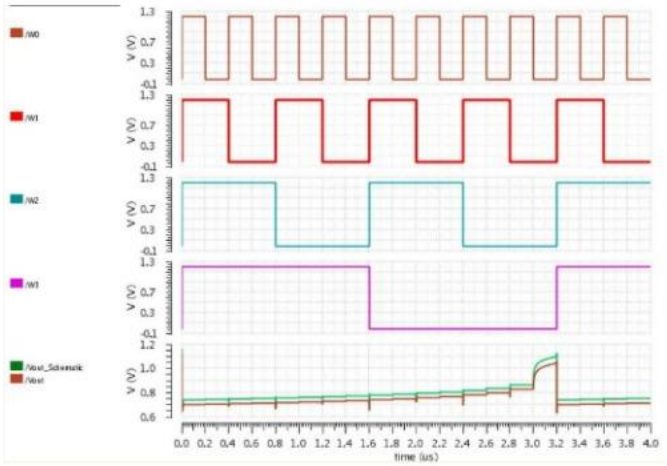
Figure 4.33. Test bench circuit setup used for the post layout simulation

Post layout simulation is also carried out for the designed 4-bit DAC circuit in two steps. As mentioned earlier the four weight vectors are applied in the form of pulses. An upper voltage swing of 1.2 V and lower voltage swing of 0 V is used for the four weight vectors. The pulse width and period of the weight vector ($W_3W_2W_1W_0$) are maintained the same as what was used in the previous section. **Figure 4.34.(a)** shows the transient analysis performed on the test bench circuit for 4000ns where, the weights ($W_3W_2W_1W_0$) used have rise and fall time fixed at 5% (20ns) of the period of the pulse W_0. $V_{out-Schematic}$ in **Figure 4.34.(a)** represents the output voltage of the 4-Bit DAC circuit for the transient analysis whereas, V_{out} shows the behaviour of av_extracted view (post layout simulation) for the corresponding transient analysis.

Figure 4.34.b represents post-layout simulation for the 4-bit DAC circuit when the weight vectors ($W_3W_2W_1W_0$) have a rise-time and fall-time of 1ps. In both the cases, the av_extracted view of 4-bit DAC has a delay in responding.



(a)



(b)

Figure 4.34. (a) Post layout simulation of 4-bit DAC when the weights ($W_3W_2W_1W_0$) have 5% (20ns) rise and fall time of the period of W_0. (b) Post layout simulation of 4-bit DAC when the weights ($W_3W_2W_1W_0$) have 1ps rise and fall-time

Chapter 5 : Conclusion and Future Works

5.1 : Conclusion

This thesis presents the research conducted by the author on in-memory computing cell for binarized neural networks. Through various investigations, the effectiveness of IMC cell for BNNs was demonstrated. The current computing method was introduced with the intention of modelling the IMC cell with binary-weighted current sources. The designed IMC cell was successfully validated using various simulations (transient, DC and post-layout). This thesis directly addresses the challenge of energy-expensive data movement and accuracy problems associated with hardware implementation of BNNs.

The key conclusions drawn from the designed IMC cell for BNN's are as follows.

- An IMC cell is designed for the BNN's where input activations and weights are quantised as +1 or -1 (*encoded as binary '0' for -1 & binary '1' for +1. The IMC cell was targeted for BNNs including -1. But the task could not be finished in the limited research time.*). Furthermore, the work focused on storing the input activations in 6T-SRAM cell rather than the weights. The designed IMC cell can be effectively used in edge devices (L2 or L3 caches).
- The designed IMC cell has utilised the scope of current computing (analog computing) within the 6T-SRAM bit cell by designing a 4-bit DAC that can be attached to the output of a sense amplifier. Furthermore, the designed 4-bit DAC circuit accurately perform the dot product operation between the input activations and the weights ($W_3W_2W_1W_0$).
- The IMC cell demonstrated in the thesis has effectively eliminated the need of a population count. Furthermore, the designed 4-bit DAC act as a current adder.
- The designed 4-bit DAC functionality was verified using five worst-case test weight scenarios ($W_3W_2W_1W_0 = 0000, 0100, 1010, 1110, 1111$). In all the five cases, the accumulated current and output voltage is measured. It is noted that in all the cases the accumulated current is linear to the weights injected into the gates of NMOS switches (M_5, M_6, M_7, M_8).
- The designed 4-bit DAC has an average power consumption of $13.76E^{-6}$. The static power associated with the circuit is also measured. A minimum static power of $4.2562073E^{-07}$ is measured for the designed 4-bit DAC by injecting the weights ($W_3W_2W_1W_0 = 0000$) to the gates of NMOS switches (M_5, M_6, M_7, M_8), that makes

them turned off. Whereas, a maximum static power of $2.6559511E^{-05}$ measured is for the weight combination ($W_3W_2W_1W_0 = 1111$), where all the NMOS switches ($M_{5,}M_{6,}\ M_{7,}\ M_8$) are turned on.

- The important feature of the proposed 4-bit DAC is that it has 4-bit weight ($W_3W_2W_1W_0$) precision which improves the overall accuracy. This can further be increased by proper design of circuit.
- The layout of the designed 4-bit DAC has an area of $20.34\mu m$ x 4.73µm.

5.2 : Future Works

Some possible directions for future research for IMC in BNN's are as follows.

- 130nm is an old technology but the same design can be ported to state-of-the-art process, e.g., 16nm. This will significantly improve the results presented in the thesis.
- Only a single bit cell performing IMC was implemented in this thesis. In future, a convolutional neural network using the designed 4-bit DAC can be designed. This includes designing mixed-signal circuits that are capable of writing back the output voltage of 4-bit DAC to the 6T-SRAM cells storing input activations.
- The designed 4-bit DAC uses a reference current of $1\mu Amps$. Due to channel length modulation, the copied unit cell current sources produce a current of $1.36\mu A$. The same phenomenon is affecting the NMOS current mirror used to copy the accumulated current. One way to overcome this problem is to use cascode current mirrors. This is proposed to be implemented in the next stage of research.
- The layout of the designed 4-bit DAC has matching issues if taped out (binary weighted current sources do not match). One way to improve the layout matching is to do the layout using common the centroid method. This is also proposed to be implemented in the next stage of research.
- The designed IMC scheme for BNN's using 6T-SRAM only supports the usage of 1-bit binary weights (1/0). In future works, it is proposed to make the current design to support 2-bits ternary weights (+1/0/-1). Also, the options of upgrading the designed IMC scheme capable of doing more computations such as Boolean, arithmetic into the same memory can be investigated.
- As discussed earlier the current work can be used for edge devices (at L2 or L3 cache). As we have more data, more memory will be needed for operation. Normally, this need moving from L2, L3 caches to the main memory using DRAM. The memory energy consumption normalized with respect to multiply and accumulate energy consumption

is 100 times compared to SRAM, 500 times compared to DRAM and 1000 times compared to Flash memory [23]. This is an important research area that can be studied in-depth in future works.

References:

[1] L. Y., B. Y., and H. G., "NatureDeepReview," *Nature*, 2015.

[2] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.*, 2015.

[3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015.

[4] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, 2014.

[5] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving," *arXiv preprint arXiv:1610.03295v1*. 2016.

[6] G. Hinton *et al.*, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," *IEEE Signal Process. Mag.*, 2012.

[7] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, 2016.

[8] M. R. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, "Playing Atari with Deep Reinforcement Learning," *IJCAI International Joint Conference on Artificial Intelligence*. 2016.

[9] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017.

[10] A. A. and A. Soleimany, "6S191_MIT_DeepLearning_L1.pdf." .

[11] J. Zhang, Z. Wang, and N. Verma, "A matrix-multiplying ADC implementing a machine-learning classifier directly with data conversion," *Dig. Tech. Pap. - IEEE Int. Solid-State Circuits Conf.*, vol. 58, pp. 332–333, 2015.

[12] E. H. Lee and S. S. Wong, "A 2.5GHz 7.7TOPS/W switched-capacitor matrix multiplier with co-designed local memory in 40nm," *Dig. Tech. Pap. - IEEE Int. Solid-State Circuits Conf.*, vol. 59, pp. 418–419, 2016.

[13] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proceedings of the IEEE*. 2017.

[14] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems*, 2016.

[15] A. Karpathy, "CS231n Convolutional Neural Networks for Visual Recognition," *Stanford University*. 2016.

[16] Y. Le Cun *et al.*, "Handwritten Digit Recognition: Applications of Neural Network Chips and Automatic Learning," *IEEE Commun. Mag.*, 1989.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, 2017.

[18] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 141–142, 2015.

[20] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and*

Pattern Recognition, 2016.
[22] M. Courbariaux, Y. Bengio, and J. P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015.
[23] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," *Dig. Tech. Pap. - IEEE Int. Solid-State Circuits Conf.*, vol. 57, pp. 10–14, 2014.
[24] W. A. Wulf and S. A. McKee, "Hitting the memory wall," *ACM SIGARCH Comput. Archit. News*, vol. 23, no. 1, pp. 20–24, 1995.
[25] A. Biswas, "Energy-efficient smart embedded memory design for IoT and AI," 2018.
[26] A. Agrawal, A. Jaiswal, C. Lee, and K. Roy, "X-SRAM: Enabling in-memory boolean computations in CMOS static random access memories," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 65, no. 12, pp. 4219–4232, 2018.
[27] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nat. Nanotechnol.*, 2020.
[28] S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, D. Glasco, and NVIDIA, "Gpus and the Future of Parallel Computing Is Investigating an Architecture for a Heterogeneous High - Performance," pp. 7–17, 2011.
[29] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," *Proc. - Int. Symp. Comput. Archit.*, vol. Part F1286, pp. 1–12, 2017.
[30] L. Fick and D. Fick, "Introduction to Compute-in-Memory," *Proc. Cust. Integr. Circuits Conf.*, vol. 2019-April, pp. 1–65, 2019.
[31] K. Zhang, *Embedded Memories for Nano-Scale VLSIs*. 2009.
[32] V. Sze, "How to Evaluate Efficient Deep Neural Network Approaches Book on Efficient Processing of DNNs." [Online]. Available: https://www.rle.mit.edu/eems/wp-content/uploads/2020/06/2020_CVPR_evaluate_dnn.pdf.
[33] D. Keitel-Schulz and N. Wehn, "Embedded DRAM development: Technology, physical design, and application issues," *IEEE Des. Test Comput.*, vol. 18, no. 3, pp. 7–15, 2001.
[34] Y. Chen *et al.*, "DaDianNao: A Machine-Learning Supercomputer," *Proc. Annu. Int. Symp. Microarchitecture, MICRO*, vol. 2015-Janua, no. January, pp. 609–622, 2015.
[35] T. Chen, J. Wang, Y. Chen, and O. Temam, "Proceedings of the 1996 7th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS-VII," *Comput. Archit. News*, vol. 24, no. Special Issu, pp. 269–283, 1996.
[36] J. Jeddeloh and B. Keeth, "Hybrid memory cube new DRAM architecture increases density and performance," *Dig. Tech. Pap. - Symp. VLSI Technol.*, pp. 87–88, 2012.
[37] H. Jun *et al.*, "HBM (High bandwidth memory) DRAM technology and architecture," *2017 IEEE 9th Int. Mem. Work. IMW 2017*, pp. 2–5, 2017.
[38] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, "Neurocube: A Programmable Digital Neuromorphic Architecture with High-Density 3D Memory," *Proc. - 2016 43rd Int. Symp. Comput. Archit. ISCA 2016*, pp. 380–392, 2016.
[39] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. Kozyrakis, "TETRIS: Scalable and Efficient Neural Network Acceleration with 3D Memory," *ASPLOS '17 Proc. Twenty-Second Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, pp. 751–764, 2017.
[40] K. Ueyoshi *et al.*, "QUEST: Multi-purpose log-quantized DNN inference engine stacked on 96-MB 3-D SRAM using inductive coupling technology in 40-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 186–196, 2019.
[41] M. M. Sabry Aly *et al.*, "The N3XT Approach to Energy-Efficient Abundant-Data Computing," *Proc. IEEE*, vol. 107, no. 1, pp. 19–48, 2019.
[42] J. Zhang, Z. Wang, and N. Verma, "In-Memory Computation of a Machine-Learning

Classifier in a Standard 6T SRAM Array," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, 2017.
[43] A. Biswas and A. P. Chandrakasan, "CONV-SRAM: An energy-efficient SRAM with in-memory dot-product computation for low-power convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, 2019.
[44] M. Ali, A. Jaiswal, S. Kodge, A. Agrawal, I. Chakraborty, and K. Roy, "IMAC: In-Memory Multi-Bit Multiplication and ACcumulation in 6T SRAM Array," *IEEE Trans. Circuits Syst. I Regul. Pap.*, pp. 1–11, 2020.
[45] A. Jaiswal, I. Chakraborty, A. Agrawal, and K. Roy, "8T SRAM Cell as a Multibit Dot-Product Engine for beyond Von Neumann Computing," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 27, no. 11, pp. 2556–2567, 2019.
[46] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-Tile 2.4-Mb In-Memory-Computing CNN Accelerator Employing Charge-Domain Compute," *IEEE J. Solid-State Circuits*, vol. PP, pp. 1–11, 2019.
[47] A. Agrawal *et al.*, "Xcel-RAM: Accelerating Binary Neural Networks in High-Throughput SRAM Compute Arrays," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. PP, pp. 1–13, 2018.
[48] M. Kang, S. K. Gonugondla, A. Patil, and N. R. Shanbhag, "A Multi-Functional In-Memory Inference Processor Using a Standard 6T SRAM Array," *IEEE J. Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, 2018.
[49] M. Kang, S. Lim, S. Gonugondla, and N. R. Shanbhag, "An In-Memory VLSI Architecture for Convolutional Neural Networks," *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 8, no. 3, pp. 494–505, 2018.
[50] H. Kim, Q. Chen, and B. Kim, "A 16K SRAM-Based Mixed-Signal In-Memory Computing Macro Featuring Voltage-Mode Accumulator and Row-by-Row ADC," *Proc. - 2019 IEEE Asian Solid-State Circuits Conf. A-SSCC 2019*, pp. 35–36, 2019.
[51] B. Razavi, *Fundamentals of Microelectronics*. 2008.