Aalto-yliopisto
Aalto-universitetet
Aalto University

Master's Programme in Electronics and Nanotechnology

# Design of a SRAM memory controller and interface for in-memory computing applications

**Rafał Mazurkiewicz**

**Master's Thesis**
**2020**

Aalto-yliopisto
Aalto-universitetet
Aalto University

**Author** Rafał Mazurkiewicz

**Title of thesis** Design of a SRAM memory controller and interface for in-memory computing applications

**Programme** Electronics and Nanotechnology

**Major** Micro- and Nanoelectronic Circuit Design

**Thesis supervisor** Asst. Prof. Martin Andraud

**Thesis advisor(s)** M. Sc. Gaurav Singh

**Date** 05.08.2022 **Number of pages** 41 **Language** English

**Abstract**

Recently, neural networks have gained much attention, due to their high effectiveness. Their operation principle is based on massively parallel calculations, which posses a challenge for classical computing architectures, based on the Von Neumann principle, which uses separate memory and computing units. Due to low throughput of interconnections between these two systems (the so called Von-Neumann bottleneck) neural networks cannot be effectively computed by these classical architectures. Therefore many in-memory-computing architectures, where many computation are performed inside memory, have been recently proposed to solve this issue. In-memory-computing system provides efficient implementation of massively parallel computation. However, providing necessary weights of neural networks into the computing units poses challenges, as memory is typically too small to fit all weights and perform all computations at once. Yet, finding efficient ways of loading weights into this memory has not been extensively researched. For that reason, this thesis focuses on design of memory controller, that is used in in-memory-computing architecture for transferring weights into the underlying memory. Specifically, several controller topologies are compared and one selected design is simulated in the context of an in-memory computing matrix. In addition this thesis provides an extensive theory background of IMC system, namely its variations, basic building blocks, advantages and disadvantages.

**Keywords** in-memory-processing, in-memory-computing, computing-in-memory, neural networks, memory controller, SRAM.

# Contents

# Preface

I want to thank Professor Martin Andraud and my advisor Gaurav Singh for their guidance.

Otaniemi, 5 August 2022
Rafał Mazurkiewicz

# Symbols and abbreviations

## Symbols

| | |
|---|---|
| $b_i$ | i[th] bit |
| $E_{offset}$ | Offset error |
| $I_{out}$ | Output current |
| $I_{LSB}$ | Output current for least significant bit |
| R | Resistance |
| $V_{ref}$ | Reference voltage |
| $V_{out}$ | Output voltage |

## Abbreviations

| | |
|---|---|
| ADC | Analog to digital converter |
| ANN | Artificial Neural Networks |
| BL | Bit line |
| BLB | Bit line bar |
| CIM | Computing-in-memory |
| CMOS | Complementary Metal-Oxide-Semiconductor |
| CPU | Central processing unit |
| DAC | Digital to analog converter |
| DIMA | deep in-memory architecture |
| DRAM | Dynamic random access memory |
| GPU | Graphical processing unit |
| IC | Integrated circuits |
| INL | Integral nonlinearity |
| IMC | In-memory-computing |
| IMCU | Multibit in-memory compute unit |
| IMP | In-memory-processing |
| FS | Full scale |
| LSB | Least significant bit |
| MAC | Multiply and accumulate |
| MDAC | Multiplying digital to analog converter |
| MSB | Most significant bit |
| MOM | metal-oxide-metal |
| MOS | metal–oxide–semiconductor |
| MVM | Matrix-vector-multiplication |
| NN | Neural network |
| PCM | Phase-change memory |
| ReLU | Rectified Linear Unit |
| RRAM | Resistive random access memory |
| SRAM | Static random access memory |
| TIA | Transimpedance amplifier |
| WL | Word line |
| WLDAC | Word line digital to analog converter |

# 1 Introduction

In recent years, we have witnessed the development of applications relying on Artificial Intelligence (AI), in particular for embedded devices such as smartphones, smartwatches and other devices from the Internet-of Things. At the heart of these applications, artificial Neural Networks (ANNs) have received much attention due to their wide area of application and high effectivity. ANNs have proven to be powerful and successful in many disciplines, such as image recognition and classification, illness detection, as well as a development of game-winning strategies [1].

The operating principle of a neural network (NN) is largely based on relatively simple matrix-vector-multiplication (MVM) in which an input vector is multiplied by a matrix of network weights with the resulting vector being the output of the network. However, modern computers, based on the well-known Von-Neumann architecture, are inherently inefficient for this particular mathematical operation [2]. Figure 1 graphically presents differences between IMC and classical approach. Due to the central processing unit (CPU) being separated from the main memory, the cost of transferring data between those two subsystems causes a major bottleneck since this memory-CPU connection has low bandwidth. Typical operations in a system include fetching data from the memory, calculating, and returning the result by storing it in the memory. Thus, for every calculation, the system transfers data to/from the memory twice which limits the processing of large data as transferring data to/from the memory consume significantly more energy than performing the computation [3], [4]. To alleviate this problem, many researchers have proposed moving the computation part directly into the memory. This concept is known as in-memory-computing (IMC), in-memory-processing (IMP) or computing-in-memory (CIM).

IMC benefits from the underlying memory architecture structure, which is a matrix of single bit cells, which is reminiscence of a mathematical matrix. The operating principle of IMC, presented in Figure 2, is as follows: the Input vector is inserted into the memory grid horizontally. It is afterward multiplied by the neural network weights, which are stored inside the matrix. The result vector is retrieved by measuring voltage on each of the matrix column. IMC can be used to calculate simple binary matrices, although real improvement arises due to the possibility of performing calculations in the analog domain which significantly improves the overall speed of the system. Presently, memory is one of the costliest parts in modern integrated circuits (IC); therefore, it is highly optimized. Unfortunately, in many cases and for proper operation, IMC requires an alteration of currently used memory designs. Moreover, those new architectures need modified control units which insert weights into the memory array. However, there is a visible lack of proper research in this area, which has motivated this work.

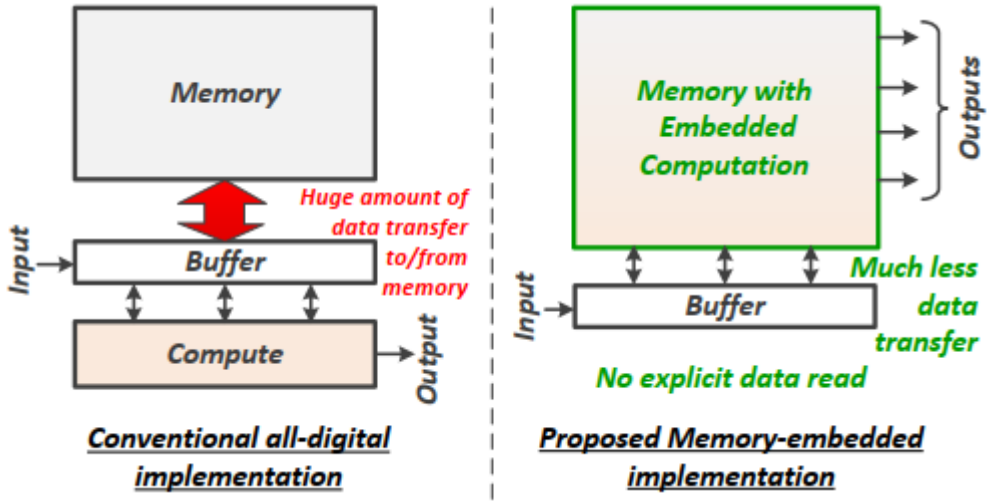


Figure 1 Difference between classical approach and in-memory-computing architecture. Red arrow indicates von-Neumann bottleneck. Figure from [5].

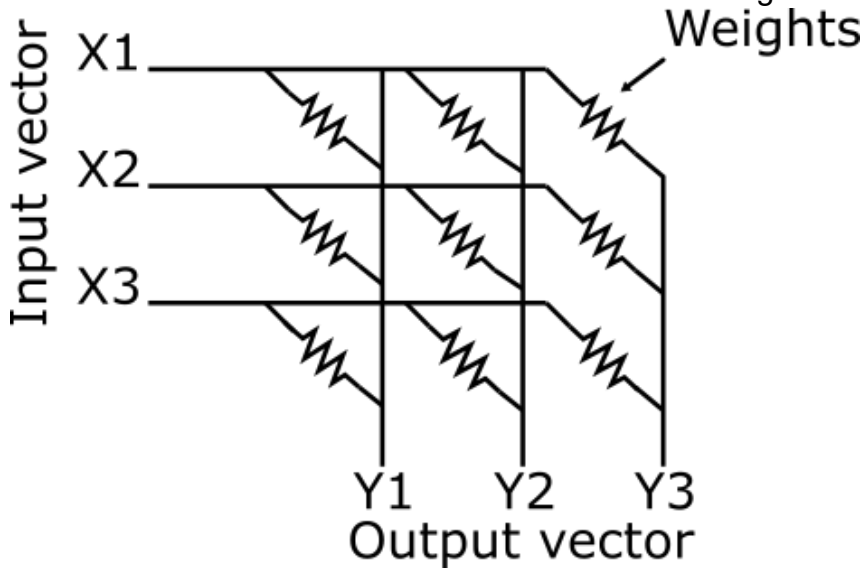


Figure 2 Structure of the in-memory-computing architecture.

Thus, the aim of this thesis is to design a memory controller capable of effectively transferring weights into memory by simulating and comparing different architecture designs.

The structure of thesis is organized as follows. Chapter 2 provides theoretical background. Chapter 3 briefly introduces controller architectures, provides design and simulation results. Chapter 4 is devoted to the summary of the work.

# 2 Theory background

Artificial neural network is a broad set of computing systems, that are based on their biological counterparts. Although NN can be divided into many subtypes, the basic idea is depicted in Figure 3. It consists of input layer, hidden layers (optional) and output layer. Essentially, each neuron takes its inputs and multiply them by a constants called weights, afterward summing these operations. The overall operation is called multiply-and-accumulate (MAC). MAC operations are considered to be the most critical operations for the neural network, and must be optimized for energy efficiency. Subsequently transferring this result to input of activation function (not shown in figure), which is directly connected to output [6]. NN is massively parallel by its nature, as every neuron of the same layer can be computed in parallel. Moving into hardware realization of the neuron, each inputs and weights should be fetched from memory, before performing the computation. Taking the example of Von-Neumann architecture (Figure 1), the computation happens in the Arithmetic and Logical Unit. After computation, the result is stored again in memory. As a result, data movement between memory and computational block accounts for a very large part of the overall power consumption of the processor. One way to amortize these memory transfers is to perform the computations of all neurons in parallel. This will reduce memory transfers as all neurons share the same inputs for instance. Further, this computation can happen directly where the weights are located, as these weights are fixed and do not change after training. This principle is referred to as "in-memory computing", which can be seen similar to computing capabilities of the human brain [7]. Neural networks based on such architecture consists of many in-memory-processing cores. Each of the cores corresponds to one layer in NN and output of one of the cores is connected to input of proceeding core [4]. Example core was already presented in Figure 2 in a very simplistic manner.

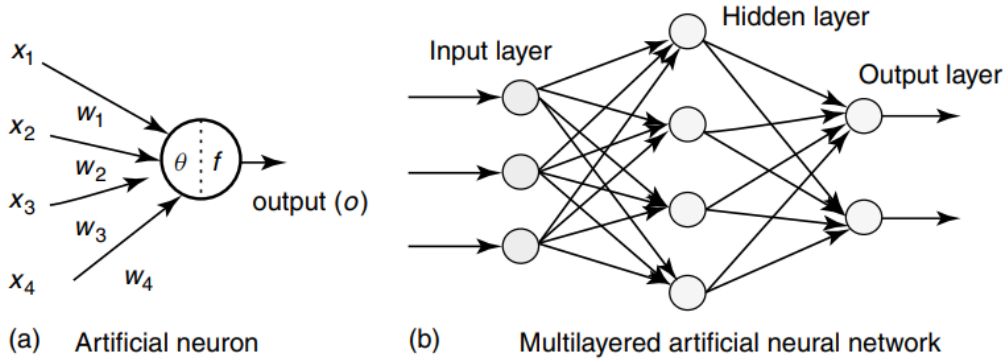


Figure 3 Artificial neural network. Each node in part b) is a single neuron from part a). Figure from [6].

## 2.1 In-memory-computing basics, applications, architectures and memory types

In-memory-computing is a wide concept, it covers many different applications and operations, such as: addition, search operations (finding where exactly the data is stored) , multiplication, etc. [8], [9]. Many researchers focus on binary XNOR, AND as well as other operations [2]. Nevertheless for artificial neural network the most important is matrix vector multiplication (MVM), also interchangeably called multiply and accumulate (MAC). Regarding IMC architectures, once that the computation part is brought inside the memory, the efficiency of the MAC operation can be considered as well. In particular, analog computing is considered to be a credible candidate, as additions in the analog domain only consist of connecting several nodes together, summing up the current. As shown in Figure 2, output e.g. Y1 is a sum of all input vectors (X1,X2,X3), multiplied by its corresponding weights. Multiplication of voltage (X1) by conductance (weight) results in current, which is merged in every node vertically (following Kirchhoff's law), leading to an output current Y1 [10].

IMC can be divided into three categories: analog IMC, digital IMC and hybrid analog-digital IMC, whereas the latter is only a proposal [11]. This thesis mainly focuses on multibit analog computation of MVM, and other uses or architectures will be only briefly introduced if any.

### 2.1.1 Ditigal IMC arrays

IMC based on digital circuits have several disadvantages mainly dealing with nonidealities due to digital memory, which decrease accuracy; therefore high amount of extra circuit is required in order to maintain system requirements [12]. Some proposed NN accelerators uses fully digital matrix-vector-multiplication and additional near-memory computing circuits [9], which do not use all the potential of IMC. However digital IMC systems perform good in logic operations (AND, XOR, etc.), and require only minor modifications of memory structure [4] .

### 2.1.2 Analog IMC arrays

Analog IMP hardware is implemented using various topologies and usually in 4-8 bit precision [12] (higher resolution systems are more prone to analog nonidealities [10]). IMC is composed of computational units, analog to digital and digital to analog converters [12]. Since IMC employs Kirchhoff's law for mathematical summing (current summing), the output is in form of current, which needs to be converted back to a digital form. It is usually done by the means of transimpedance amplifier followed by ADC. DAC and ADC are one of the biggest limiting factors in MVM IMC systems, as they consume large amount of power and require large chip area [10] [13]. There exists designs where ADC and DAC is implemented as an intermediate step in CMOS

logic [13], but they have higher circuit complexity. Theirs advantage is that they need less chip area. Some of the designs uses single bit converters with shifter to reduce overall chip area, with a drawback of being slow [14]. Other designs use pulse width modulation reducing need for conventional DAC, though still require ADC [15] (these techniques can be called timing based techniques). Those design do not need to implement additional MDACs, and thus standard memory layout can be used. Its major drawback is that computation time is exponentially proportional to the number of bits [16]. In case analog sensors act as inputs for the IMC system, then DAC can be excluded from design. Similarly for system which drives analog actuators, ADC is not required [12].

Other set of problems, which are present in analog domain comes from process variations. IMC is more sensitive to process variation, temperature variation, voltage variations etc., which introduces possibility of errors not present in its digital counterparts. To alleviate this issue some researchers propose on-chip training of every manufactured unit to include individual process variations [17]. However it is a comber stone and inefficient implementation as it require online training of every chip. Another issue is the parasitic resistance and capacitance along vertical and horizontal connections within memory (often referred to as a crossbar array). As the memory becomes larger, the problem is more severe [10]. IMC, especially analog, requires highly specialized hardware that is different from currently used commercial memory chips.

### 2.1.3 Memory types

Computational units can be based on charge based memory or resistive memory. Charge based memories rely on stored charge within cell. For example in FLASH memory it is a charge in a floating gate. Resistive memories use varying resistance as a mean to store data. They are often referred to as memristors, since their underlying physical structure can be modified, which leads to change in resistance. The structure is retained after modification, thereby possessing memory-like effect. e.g. applying heat to phase change memory (PCM) changes structure from amorphous to crystalline (or vice versa), thereby changing its resistance. Memristors are non-linear devices and can also store non-binary values.

For charge based memory SRAM (static random access memory), DRAM (Dynamic random access memory) and FLASH can be included, while for resistive based memory PCM (phase change memory), RRAM (resistive random-access memory) and similar can be included [8]. FLASH memory can also be viewed as a memristor, since it can work in partly ON state where floating gate is partly charged. Memristors have several advantages - they are non-volatile, they store data in non-binary (continuous) form and provide high resolution [10]. However they often require high power and are limited by write/read lifetime cycles [8]. Many of the memristors technologies are

fairly still not mature and as such have low commercial availability. To add to this, memristors cannot be produced with standard CMOS technology [18], [2]. Major drawback of DRAM based design is that the read operation is destructive; therefore weights needs to be stored elsewhere, which impose additional overhead [9]. Currently, FLASH and SRAM based IMC are widely researched, because of availability of processes and well-know behaviour. Flash memories have read/write cycle limits, require high voltage circuit for writing data and are power demanding when compared to SRAM [19]. The fastest type of memory is SRAM [10]. Therefore this thesis will focus mainly on SRAM, even though there exist successful commercial FLASH based designs [20].

### 2.1.4 Memory structures for crossbar arrays

Although simple designs use 1 bit weight matrix it is possible to implement this as multibit, by using e.g. MDACs (multiplying digital to analog converter) and bit cells as control bits [12]. Design from [21] uses MOM (metal-oxide-metal) capacitors for multiplication and summing, but has only 1 bit precision. Similar design described in [22] uses 8T SRAM, MOM capacitors but contrary to previous, adds digital bit shifting and summing circuit to impellent a 4-bit weights. Drawback of this design is the need for additional digital circuit and 4 times more analog to digital converters. [16] Is a combination of previous designs. It uses switched capacitor technique for MDAC and summation node. Another way is proposed by [23] The input vector is placed on WL (word-line) of SRAM as analog voltage and then precharge circuit is enabled with varying pulse width in order to capture bit significance of bits placed inside SRAM cells; thus for 4 bit precision the widths pulses of precharge circuit would be 8:4:2:1 (MSB to LSB). In addition the WL is enabled only for short period of time to reduce non-linearity caused by both bit lines discharge rate (when bit lines reach 100 mV, discharge rate is drastically reduced). Discharge time is always constant.

A somewhat similar approach (called deep in-memory architecture – DIMA) is presented in [15] and in Figure 4. Contrary to standard SRAM data words are placed on column-wise cells instead of row-wise. Analogous as in previous paragraph WLs are enabled with different pulse width to capture bit significance. Precharge circuit is the same as in conventional memory. Together with stored bits inside memory discharges bit lines rates are proportional to input vector and NN weights. There are also techniques with pulse height modulation and varying number of pulses, but they are not well researched [24].

In [5] the MVM is computed by using 256×64 cell wide memory array, that store weights (1 bit wide), together with local analog averaging circuit. Each row's average is then directed to ADC. To prevent write disturbances across bit line columns the 10T SRAM is utilized.

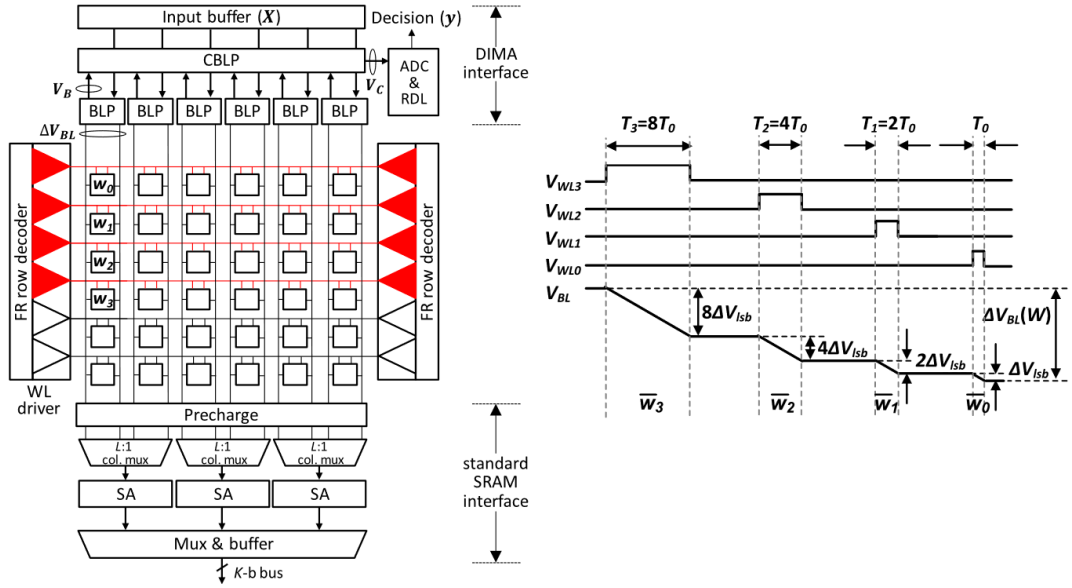


Figure 4 DIMA architecture is shown on the left. Weights are stored column-wise ($w_0$-$w_3$), and their significance is obtained by enabling word lines with specific period (right part of the figure). Impulses can overlap. Figure adopted from [15].

Another approach is taken by [16] and [12], where multiplying is empowered by a DAC. In [12] it is resistive MDAC and the output is a sum of current, while in [16] it is based on switching capacitors and the result is in terms of stored charge in capacitors. [16] is also presented in Figure 5. This works fall into two categories of MAC: current based computing based on resistive circuits and charge domain computing [25].

Lastly, design from [26] uses weighted SRAM cells. It is composed of multiply of 4 columns. Transistors in each column cell are sized in ratio 8:4:2:1, to capture significance of bits, similarly as in previously described designs. It consists of 8T SRAM cell in which 6T is a common base cell, that is connected to separate read transistors of varying width.

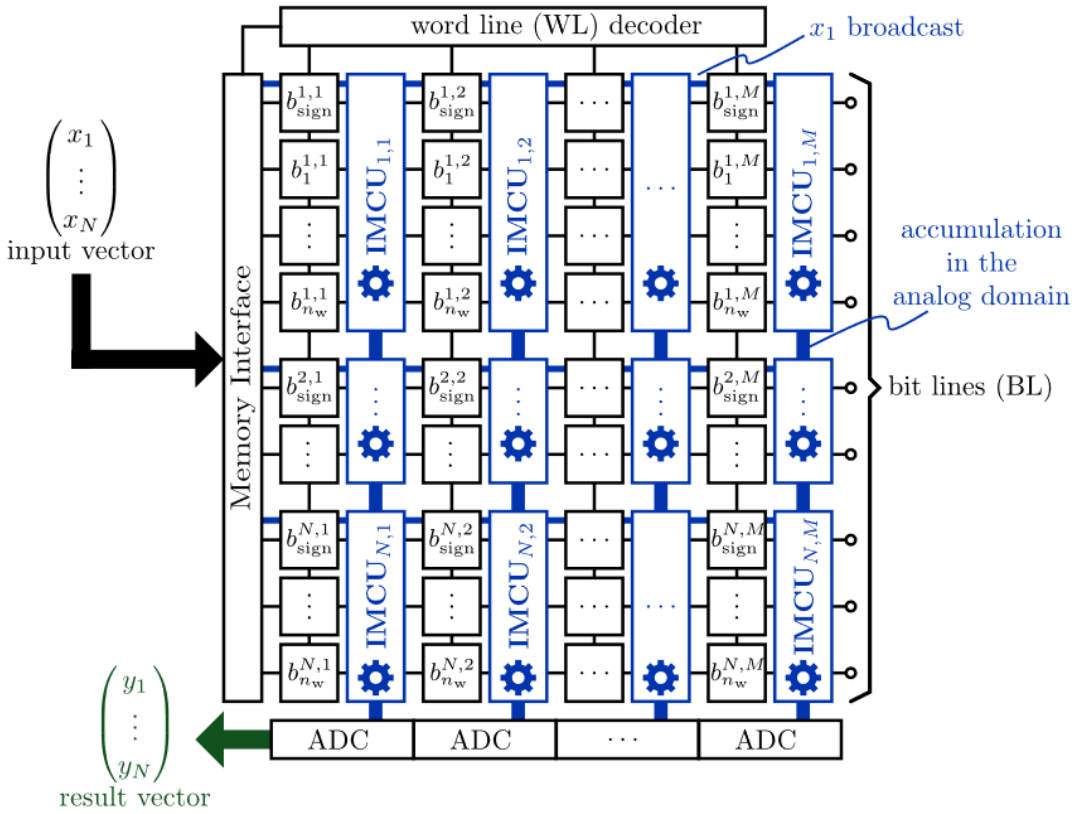


Figure 5 In this IMC all word lines are simultaneously asserted high, and the bit significance is captured by the mean of multiplying charge based digital to analog converter called in-memory compute unit (IMCU). Figure adopted from [16].

## 2.2 Digital to analog converter

Inside in-memory-computing system typically standard DAC would be used as a word line DAC (WLDAC) and MDAC (which is a typical DAC in current mode configuration as described in [27]) for multiplying input vector by NN weights (the latter used only in case system is not timing based nor made of memristors).

### 2.2.1 Types of DAC

There exist various different architectures of digital to analog converters; among others: current scaling, voltage scaling and charge scaling [28].

Voltage scaling converters typically use resistors connected in series between reference voltage and ground [28]. They require high number of resistors, e.g. 64 for a 6-bit converter. Thus they consume large area on chip, but are guaranteed to be monotonic.

Current scaling converters typically transforms reference voltage into a set of current sources. Current scaling DAC can employ e.g. binary weighted resistors to produce weighted current sources. One of the disadvantages of this configuration is that it require large resistors value spread. This means that in case of 6 bit resolution, MSB resistor is 32 times larger than LSB resistor [29]. This issue is alleviated with R-2R DAC (presented in the upper part of Figure 6), which is usually constructed of a set of 3 resistors of the same value – two of them are connected in series [28]. One of the biggest disadvantages of R-2R ladder DAC is its nonmonotonicity due to nonidealities. As an advantages they are fast, require fewer elements, and have low area usage. Output of such converter can be in terms of current or voltage, depending on used topology. Current mode R-2R DACs output can be easily transformed into voltage by using an amplifier connected to the output [12].

DACs based on transistors are another alternative to resistor based. They can be made using w-2w or binary weighted. However using binary current sources as described in [30] has a disadvantage of additional multiplying factor, that limits usefulness in IMC as MVM is depended on said factor. While some researchers report multiplying DACs based purely on transistors [31] their usage is limited to narrow operating region, due to nonlinearity with larger gate-source voltages. Therefore transistor based converters are suitable only for word line DAC.

Charge scaling converters have higher accuracy compared to R-2R based, however they require large value spread, additional clock source and buffer, which reduce usefulness [28]. They have the best accuracy.

Some researchers use other topologies, but they are less popular, e.g. used by [5] digital to time and time to analog converters as WLDAC (in [5] named as a global DAC) architecture is reported to perform better than e.g. PMOS binary weighted DAC, but it needs initial calibration.

### 2.2.2 R-2R MDAC

For R-2R MDAC the output current is described by following equation [28]:

$$I_{out} = \frac{V_{ref}}{R} \ . \qquad 1$$

Where $V_{ref}$ – input voltage,
$I_{out}$ – output current and
R – resistance of binary ladder.

Addition of transimpedance amplifier (TIA) transforms current mode DAC into standard DAC and its output voltage is given by (assuming amplifier's feedback resistor equals R):

$$V_{out} = V_{ref} \sum_{i=1}^{N} \frac{b_i}{2^i}. \qquad 2$$

Where $b_i$ is $i^{th}$ bit and N is the total number of bits. Another metrics is the so called full scale (FS) value. Since DACs resolution is finite, output voltage is not equal to reference voltage. The FS can be defined as [28]:

$$FS = V_{ref} - LSB = V_{ref}\left(1 - \frac{1}{2^N}\right) \quad 3$$

Current can be described similarly, for example 6bit DAC, made of 1 kΩ resistors yields:

$$I_{out,max} = \frac{V_{ref}}{R} - LSB = \frac{V_{ref}}{R} - \frac{V_{ref}}{64R} = 1.2\text{mA} - 18.75\text{uA} = 1.18125\text{mA} \quad (4)$$

$$where\ LSB = \frac{V_{ref}}{64R} = 18.75\text{uA} \quad (5)$$

Multiplying digital to analog converter have to be monotonic in order to perform correct MVM computations. DAC is inherently monotonic if integral nonlinearity error (INL) is less than 1 LSB [29]. INL describes how much DAC output diverges from ideal straight line and is maximal at MSB transition (during transition of binary code from 011…1 to 100…0) [32]. It is measured without taking into consideration offset error and gain error. Offset error is described as a deviation of actual output from ideal output when input code is zero, while gain error is defined for full scale. This is also described by following equations [29]:

$$E_{offset} = \left.\frac{I_{out}}{I_{LSB}}\right|_{0\ldots0} \quad (6)$$

$$E_{offset} = \left.\frac{I_{out}}{I_{LSB}}\right|_{1\ldots1} - E_{offset} - (2^N - 1) \quad (7)$$

Another way to describe converter nonideality is to use differential nonlinearity error (DNL), which is described as a maximum amount of 1 LSB deviation from its ideal 1 LSB step size [33]. If DNL is less than 0.5 LSB then it is guaranteed that DAC is monotonic. Glitches in MDACs are not important, because output S&H can be sampling at specific (correct) intervals. It is crucial that resistors value spread is as low as possible. For a 6-bit converter MSB resistor should have inaccuracy smaller than $\frac{R}{64} \approx 1.6\%$.

Switching elements of R-2R MDAC have to be sized carefully as it is copied many times in the crossbar array, so they should as small as possible. For this purpose NMOS transistor are more suitable as they have smaller area usage. In addition, as suggested by [34], using differential configuration of switches helps mitigating current surges. As a drawback it uses twice as many transistors.

As an example, in Figure 6 is depicted R-2R MDAC and unit cell, that when connected in parallel with five more adjacent cells forms 6 bit MDAC

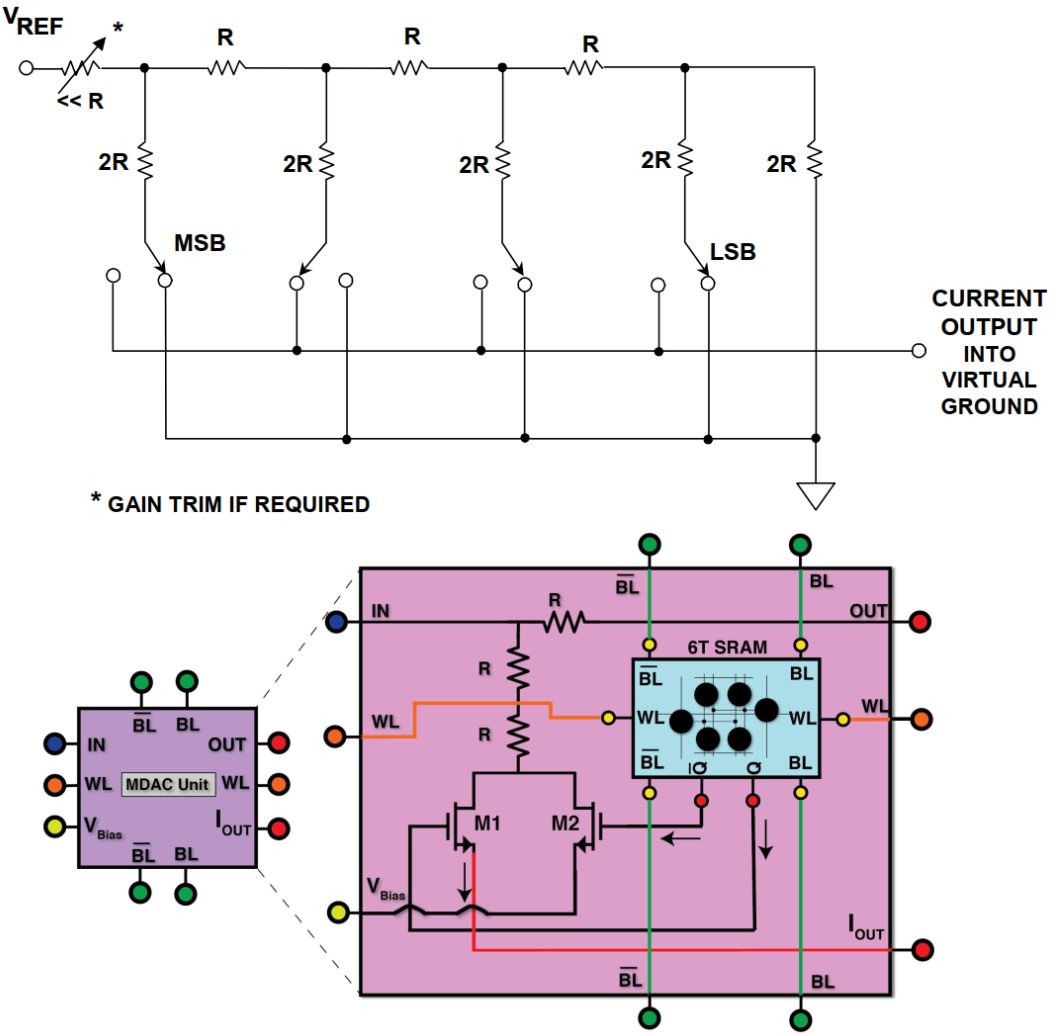


Figure 6 Typical R-2R MDAC (top) and unit cell of MDAC proposed by [12]. It uses SRAM cells as an input code. N connected cells will yield N bit converter. In one MDAC, cell area of switches and resistors is considerably larger than area of SRAM cell (9 μm$^2$, 7.8 μm$^2$, 5.72 μm$^2$ respectively). Figures combined from [27] and [12].

## 2.3 Transimpedance amplifier

In order to read a result of in memory MVM the transimpedance amplifier (TIA), which converts current to voltage, is usually employed [35]. TIA consists of amplifier in closed loop configuration and a feedback resistor $R_F$ [29]. Resistor is conventionally sized to match R-2R DAC resistors [36], however in IMC system, where many DACs are connected to one line impedance will vary in greater extend (impedance of DAC is dependent on digital input code). For example if the crossbar array size is 8x8, then the actual impedance of the set can be 8 times smaller than that of a single converter. Therefore feedback resistor should have smaller value than normally. In addition TIA should be small, as it will be multiplied across all columns. If IMC is

supposed to provide analog signals as outputs, TIA should preferable have full range rail-to-rail voltage output ( in case of TSMC 65nm process it is in 0-1.2V range). On the other hand if TIA stage is followed by ADC, the load capacitance is minimal. It is important to note that TIA uses inverting configuration. There is a visible lack of scientific papers, that consider this peripheral in IMC systems. As an example [12] uses a two stage miller compensated differential amplifier.

## 2.4 Activation function

Activation function is an important part of NN, because without it, only linear problems are solvable [37]. There are many types of activation function, among others: Sigmoid, hyperbolic tangent (tanh) and Rectified linear function (ReLU) [38]. The most popular are Sigmoid, which is used for output layer and ReLU, which is used for hidden layers [39]. Nevertheless ReLU is the simplest and it is close to what neuroscience says about brain (less neurons are saturated simultaneously)[37]. Activation function can be implemented in a digital fashion after analog to digital conversion [9], or in analog domain as presented in [12], where ReLU is realized using voltage-mode MAX circuit.

## 2.5 Analog to digital converter

As has been mentioned earlier ADC is another bottleneck in terms of power and chip area usage. ADC have to be duplicated across all memory columns in order to achieve massive parallelism. Flash ADC, successive approximation register ADC (SAR ADC), Integrating ADC and Ramp ADC are some of the possible architectures. Bit resolution is not a priority in this design [40], thus high accuracy sigma-delta converters are not required. Another important point is that, assuming digital input vector or NN weights are not changed during ADC conversion phase, sample and hold circuit is not required for ADC (it might however be necessary for IMC input if it is directly taken from analog sensors. Normally, to distinguish all input level ADC would need to have considerably large resolution e.g. crossbar memory array composed of 8 rows with 6 bit wide input and weights yields $6 \times 6 \times 8 = 288$ individual levels, which means converter of at least 9 bit resolution [10]. However ignoring the surplus and converting in lower resolution gives good enough results [40] (as it has been shown by [41], weights of the neural network are usually centred around zero).

In addition to aforementioned converters types there exist designs of current-mode ADC [42], but they are not widely researched, therefore will not be further elaborated.

### 2.5.1 SAR ADC

SAR ADC is successively approximating input value and requires several cycles (number of cycles equals number of converter's bits) to perform conversion. It consists of comparator, DAC, and control logic [28]. SAR ADC brings only minor overhead to chip area as resolution increases [40]. It is also a good compromise between speed and resolution [43]. As an example, IMC described in [12] uses 6 bit SAR ADC.

### 2.5.2 Flash ADC

6 bit Flash ADC is considerable larger in terms of area usage and energy than e.g. SAR ADC [40] (about three times larger). Considering that it has to be duplicated across many columns it is not suitable choice for the IMC system. However it provides good accuracy and it is the fastest solution [28]. As an example [40] uses low resolution 4 bit Flash ADC, which is only slightly larger than SAR ADC.

### 2.5.3 Ramp and integrating ADC

The biggest advantage of Ramp and integrating converters comes from easy parallelization [44],[45]. Ramp column parallel ADC needs only one comparator, and latch per one crossbar column. All columns share single ramp generator and digital counter. Therefore it has much smaller die area than SAR ADC [44]. However it comes at the expense of long conversion time. In case of 6 bit, Ramp converter takes roughly $\frac{2^N}{N} = \frac{2^6}{6} \approx 11$ times more clock periods than SAR based [44]. To alleviate this obstacle new designs that combine both architectures are introduced [44], [46], however they are more relevant to higher resolution converters. Ramp type ADC can use same ramp generator layout as WLDAC, so the linearity, process variation etc. are similar. Typical dual-slope integrating converter has two times longer conversion time than single slope ramp ADC, but they have higher accuracy [28].

In Ramp ADC the only analog part are comparators and generator. Since generator structure can be reused from WLDAC, comparators are the main concern. Latch based comparator has low voltage resolution [47], therefore it is not suitable. Two stage open-loop comparator has resolution as high as 1 mV, unfortunately response time is slow [47].

## 2.6 Memory controller

Memory array has to work in at least two modes: computational and standard memory access [17],[48]. The latter is used for inputting NN weights, while the former is used when performing actual MVM operation. Memory controller can be same as in standard SRAM design [17], nevertheless many systems use custom crossbar architecture, therefore controllers have to be modified accordingly. Depending on the chip design, if the NN weights are constant, or number of NN layers is low enough, the controller will be used only

once during boot-up. However, if weights are variable, it should be strongly optimized.

As reported by [40] most IMC architectures assume one-time programming of weights. Therefore Many researchers skip entirely the controller design e.g. [5],[49],[23],[48] to mention few. Some authors focus only on subparts e.g. [15] mainly describes column multiplexing problem. On-chip training of the SRAM based systems requires peripheral circuit that will calculate values of new weights based on previous results [40]. If SRAM is made of 8T or 10T cells, which allow asymmetric reading and writing, the speed constraint is not important. Nevertheless updating data in 6T cell prevents reading, so the NN will not work during this time. In addition, it is frequently assumed that chip is large enough to store all weights, which is not always valid. As stated by [40] 7.9 MB model would require 22 $mm^2$ of chip area in 7nm process (SRAM); therefore larger neural networks are prone to performance bottleneck due to frequent weights reload from external memory, which is similar to mentioned in introduction von-Neumann bottleneck. Chip area constraint can reduce throughput even 4 times when compared to full size, which lefts CIM benefits questionable [50]. Architecture presented in [51] consumes 50% of overall energy consumption for off-chip DRAM access, because it has not enough space in internal buffers. Another reason why memory controller should be optimized is time of memory calibration. Commercially available FLASH based design is recommended to have calibration on a daily basis, which requires one minute [20]. It is worth noting that weights update increases energy consumption considerably [17].

# 3 Design

## 3.1 Overview of memory architectures

A straightforward way to implement the SRAM controller would be to design a layout where memory peripheral controls are on two sides of memory array. This arrangement require small area and is suitable for small memory sizes. It is presented in Figure 7 (together with analog IMC peripherals). It is sometimes referred to as “quad butterfly architecture”.

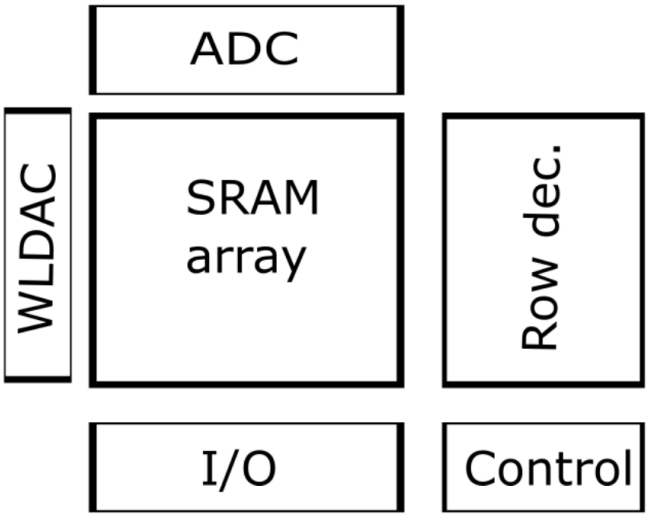


Figure 7 Quad butterfly layout of memory controller.

### 3.1.1 Full butterfly architecture

However, more efficient way is to split memory array to four banks as presented in Figure 8 leading to the so called “full butterfly architecture”. The memory controller is in the middle, thus line parasitic capacitances and resistance are less severe. Therefore signal time propagation will be two times smaller. Additionally only one bank can be selected at the time, in order to decrease power consumption. Main drawback of this architecture is increased chip area usage.

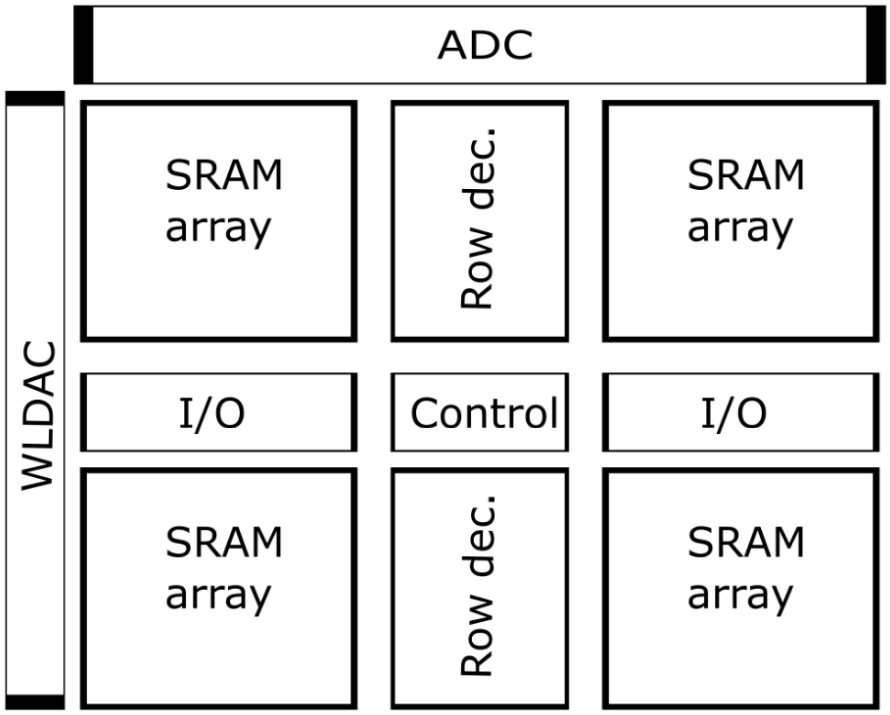

Figure 8 Full butterfly architecture. Control block being in the centre of memory has optimal path to all SRAM cells.

By definition such a design has similar dimensions of SRAM array on both bit lines and word lines. Depending on required memory size it might be more optimal to use different aspect ratios, e.g. stack only two banks out of four, leading to half butterfly architecture. It is to be decided based on the application, nevertheless full butterfly scheme has the best performances.

Even so, it might be necessary to use larger number of banks, meaning that full butterfly architecture will not be so effective anymore. While large numbers of smaller banks improves signal to noise ratio (SNR) of analog computation result, it increases wiring cost, as memory banks are farther from each other. In case of many banks, it is often organized in such a way that each memory bank presented in Figure 8 is itself a nested, independent unit with individual controller. For better understanding an example layout is shown in Figure 9. It is highly hierarchical design; therefore posses a risk of high read/write latency.

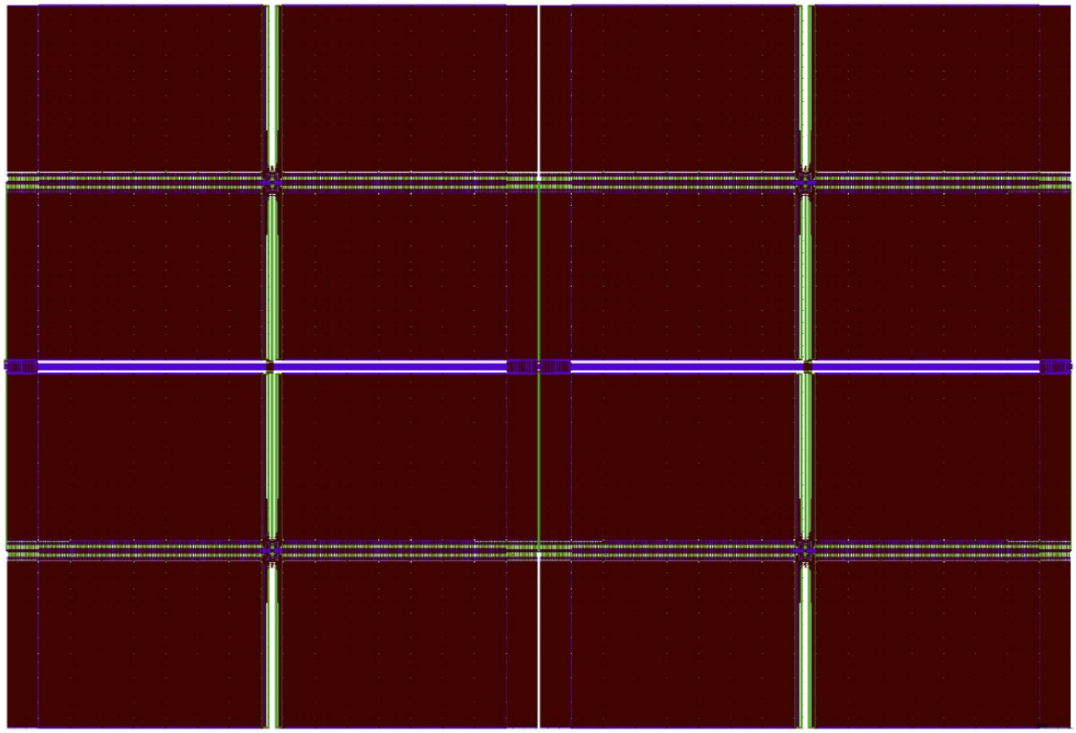

Figure 9 A 16 banks memory layout. It consists of 4 low level and 1 top level full butterfly arrangement, which results in total of 5 memory controllers. Figure taken from [52].

Depending on level of system complexity controller can be designed with aid of memory compilers such as OpenRAM [53], but even then, the layout have to accommodate additional analog circuits. Memory compiler called OpenRAM supports maximum of 4 banks [53], which are arranged as in Figure 8. Another project called “Asynchronous Memory Compiler” (AMC) can provide up to 16 banks [52]. In that case memory consists of 4 blocks, as described above. AMC is able to produce stretched layout with different aspect ratio, but due to larger distances it requires pipelining.

## 3.2 Specific constraints related to IMC

As has been mentioned earlier the layout of memory cells in IMC system might differ from the common one. Timing based techniques use custom word drivers and/or precharge circuit, which require additional clock generators or sources. Depending on design, if same bit line is used both for programming and interference, then there is a large current surge [9]. To prevent this, interconnections should be larger, but it leads to larger line capacitance. Maximal memory size of memristor array is reported to be 128x64 [54], [55]. Thus for higher size of input/output vector multi-staging is required. Crossbar array have to be splitted into smaller subarrays, so that low resolution ADC is able to distinguish all voltage levels, which might posses additional problems for controller design. Simulations from [26] have shown

that output error can reach up to 14% due to parasitic resistances (The design is made of weighted SRAM cells and is relying on transistor conductance). Large number of subarrays increase share of wiring in response time of the memory. Experiments from [56] have shown that wiring cost in fully digital IMC may account for more that half of the overall read access time (although the authors focus on digital implementation, the numbers will be most likely smaller in analog IMC).

Additionally depending on implemented scheme of multiplication operation, if MDAC is used, it might be beneficial to switch from the most popular thin-cell layout to others such as wider ultra-thin-cell [57], so that space is used evenly. Such wider cells would increase load on WL, but at the same time column circuits could have more spare room.

IMC contrary to standard memory does not require reading data from subsequent cells, meaning that memory controller can be simpler than normally.

Presented in [4] design focuses on convolutional neural networks, rather than simple feedforward NN, thus the controller is substantially complicated. Also size of the area usage of the mentioned controller and input SRAM buffer is rather high. In 14nm process it is 0.0132 $mm^2$ and 0.0092 $mm^2$ respectively (under assumption that chip will cooperate with $256 \times 256$ PCM crossbar array). Unfortunately authors have not compared throughput with classical GPU approach.

There are many IMC designs, that use the simplest approach, examples are shown in Figure 10 thru Figure 12.

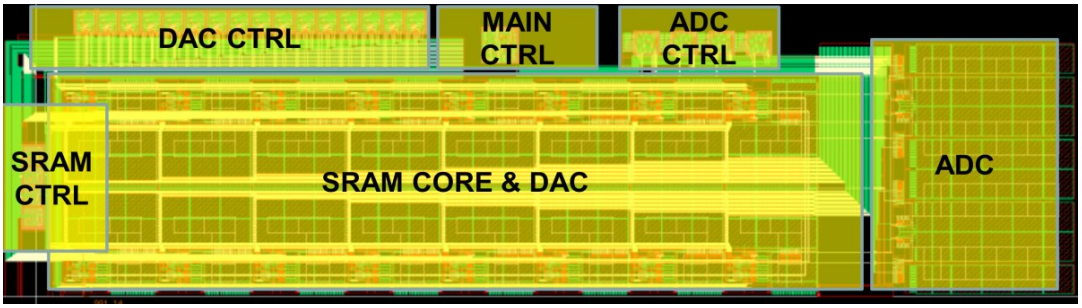


Figure 10 Layout of IMP presented in [58]. The schematic is also presented in Figure 11 for clarity. This system consist of 16x4 SRAM cells 16 DACs and 4 ADCs.

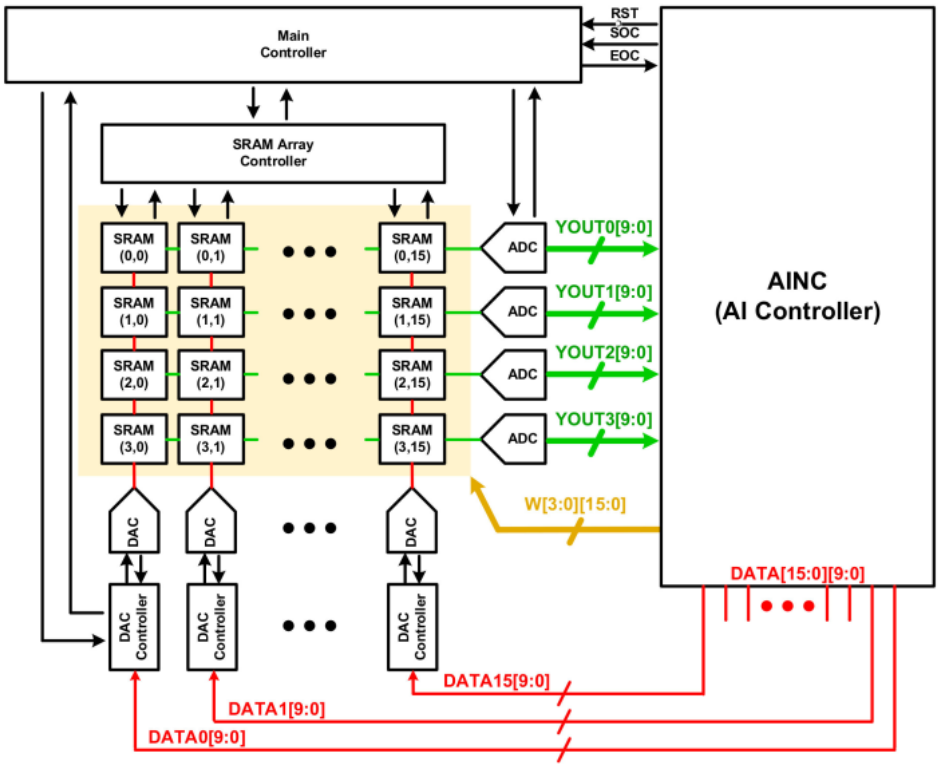


Figure 11 IMP system proposed by [58]. Its layout is presented in Figure 10.

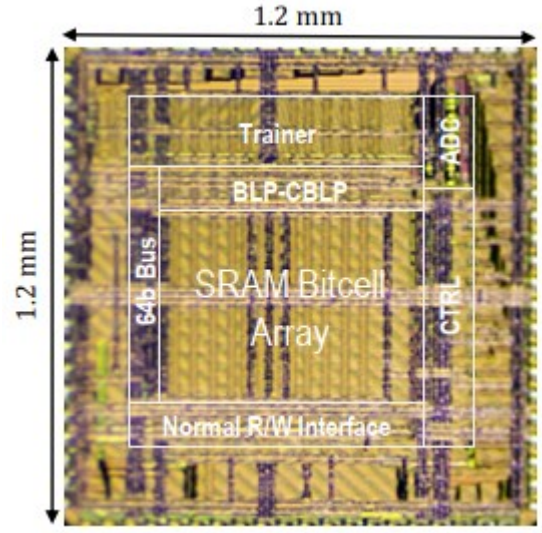


Figure 12 Layout of DIMA architecture described in [17].

## 3.3 Design assumptions

Hereafter, the memory controller designed in this particular work is presented. It is assumed that IMC is using an R-2R MDACs and WLDACs. Therefore no changes have to be made when compared to standard memory interface. Design is made using TSMC 65nm process.

All tests have been performed with SRAM test cell. Its size is $2.9\ \mu m^2 \times 1.9\ \mu m^2$ and the post-layout extracted parasitic capacitances are

0.2 fF for bit lines and 0.51 fF for word line. Transistors sizes are minimal (60 nm per 120 nm), the cell ratio equal to 2 and pull-up ratio equal to 1.

Memory size is 1 kB. When using architecture from Figure 8, each bank has size of 256 bits and by using 2:1 column multiplexing divided into 2 sub banks of size 16×8 cells. Thus one SRAM array block has width across word line equal to 16 cells and height across bit line equal to 16 cells. When using architecture from Figure 7, for the sake of simplicity a similar arrangement has been made, that is SRAM array block is composed of 8 subarrays via 8:1 column multiplexing yielding dimensions 64×16 (width per height). This arrangement is arbitrary, since full column layout has not been made. Additionally analog part of IMC system has not been implemented. Therefore it is unknown whenever multiplexing is required, but according to [59], at least 2:1 would be necessary, if the cell spacing is the same as in common non-IMC layout.

## 3.4 Initial designs and results

First of all two architectures were designed for comparison purposes, that is full butterfly and half butterfly architecture. The peripheral circuits are based on [60] and are presented in Figure 13 to Figure 16. Controller's schematic, the timing generation part, is shown in Figure 17. Part of design enabling different memory banks is presented in Figure 18. Although reading weights from memory is not necessary, a simple sense amplifier has been implemented. Column multiplexers have been made of N type pass gate transistors. Since memory size is not big, row decoders are made of AND gates, without multi-staging. All transistors have minimal length (60 nm). Transistors width in precharge, and write driver is 200nm and 3 μm respectively. Controller has bidirectional data bus (8 bits wide), which is controlled by additional Read/Write pin. Therefore it needs latches for proper operation, which are shown in Figure 19 (for visibility purposes, only a part) Address bus is 7 bits wide. In this design Reset function is implemented as simple pull-down transistors.

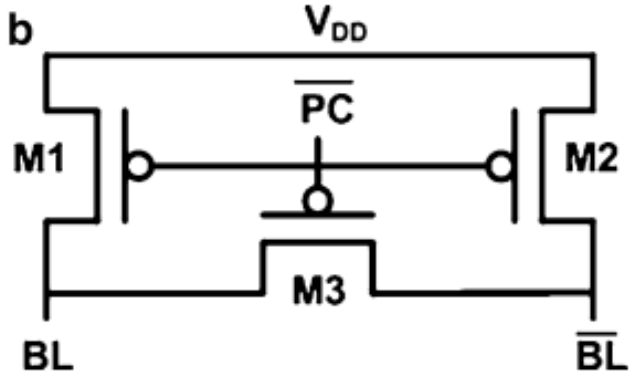


Figure 13 Precharge circuit. Picture from [60].

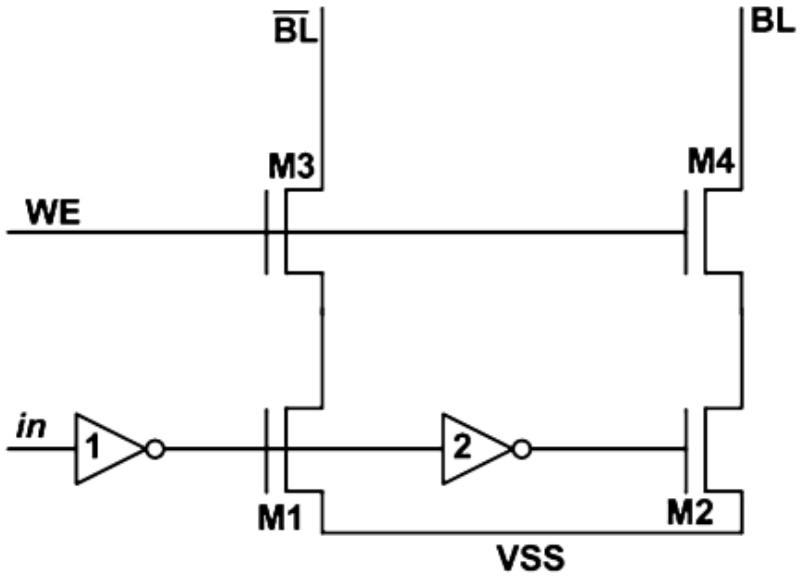


Figure 14 Write driver. Figure from [60].

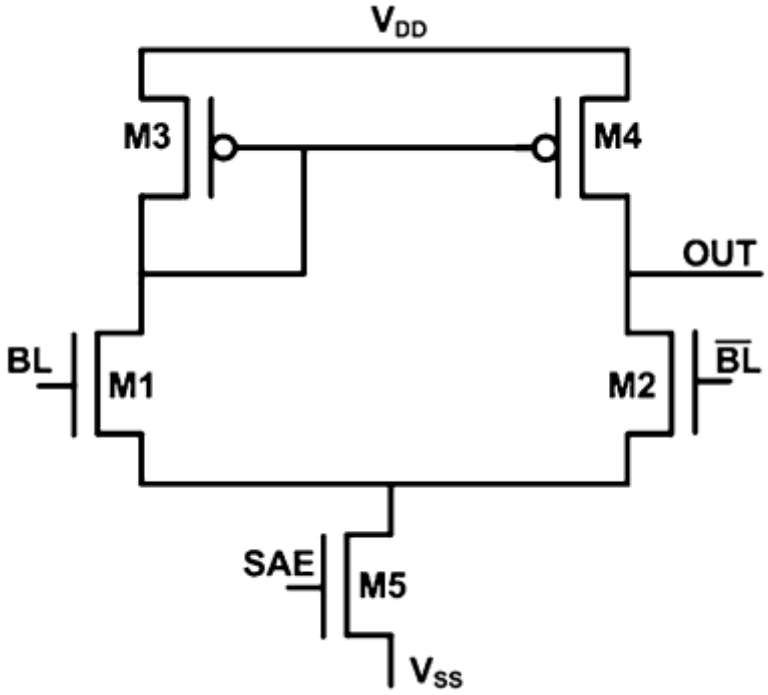


Figure 15 Sense amplifier. Figure from [60].

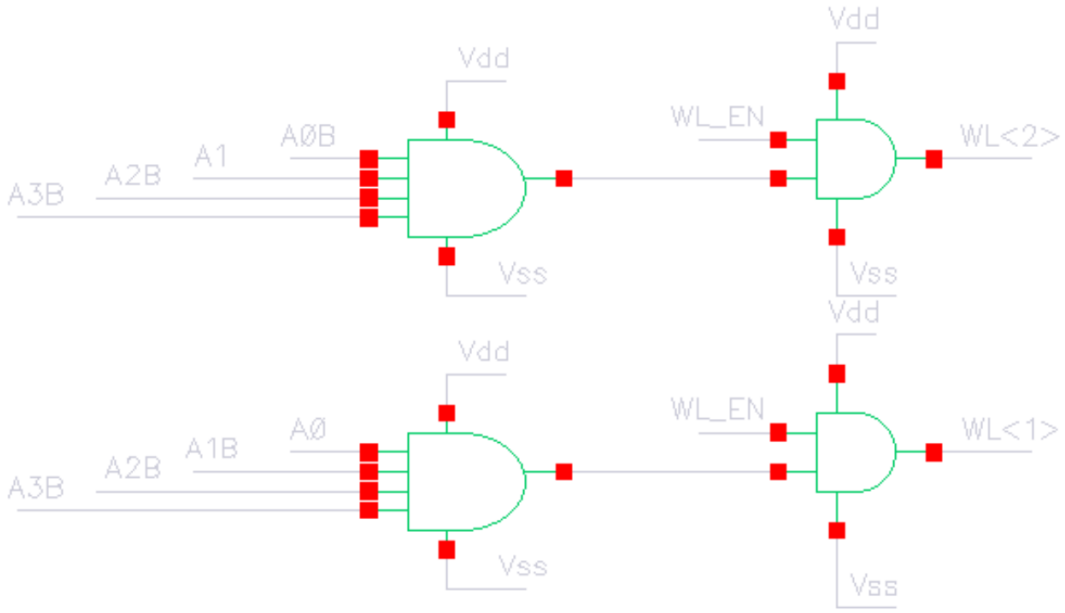


Figure 16 Part of schematic for row decoders implementation. Only 2 row drivers are shown.

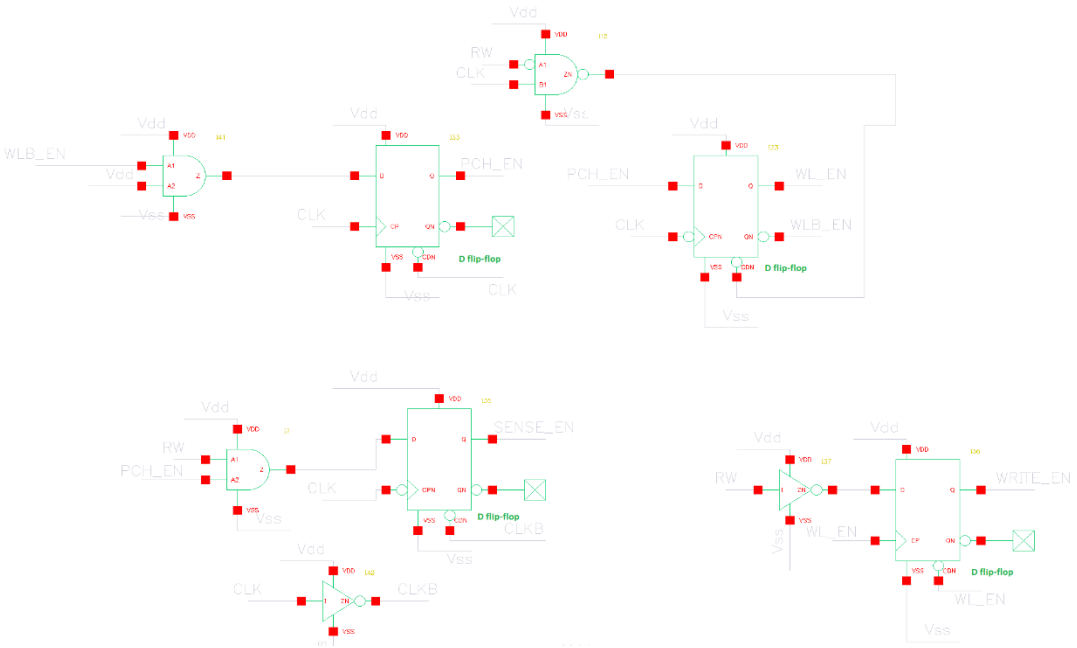


Figure 17 Main part of the memory controller.

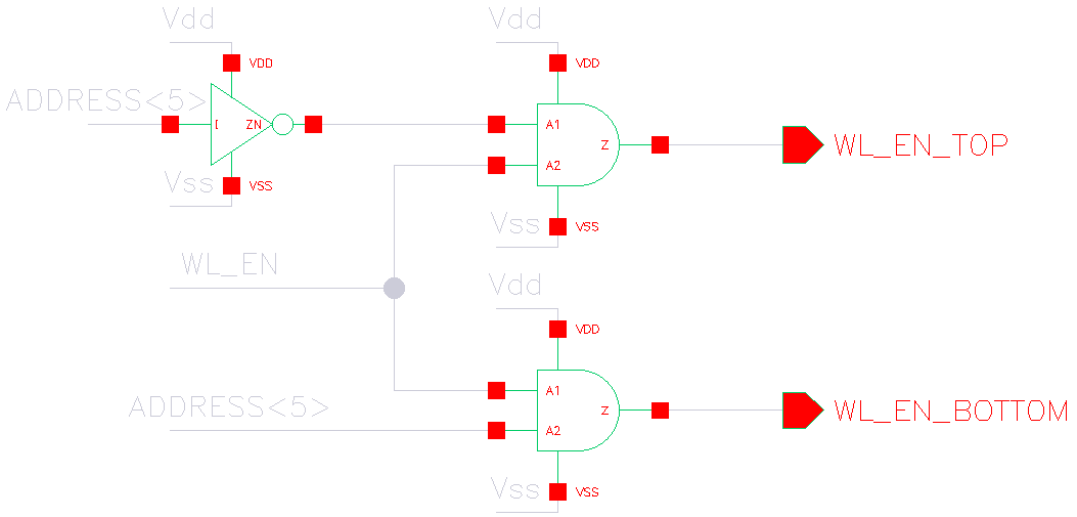


Figure 18 Scheme of enabling different memory banks. This schematic is present only in full butterfly architecture. Output (e.g. WL_EN_TOP) serves as input to row decoders shown in Figure 16.

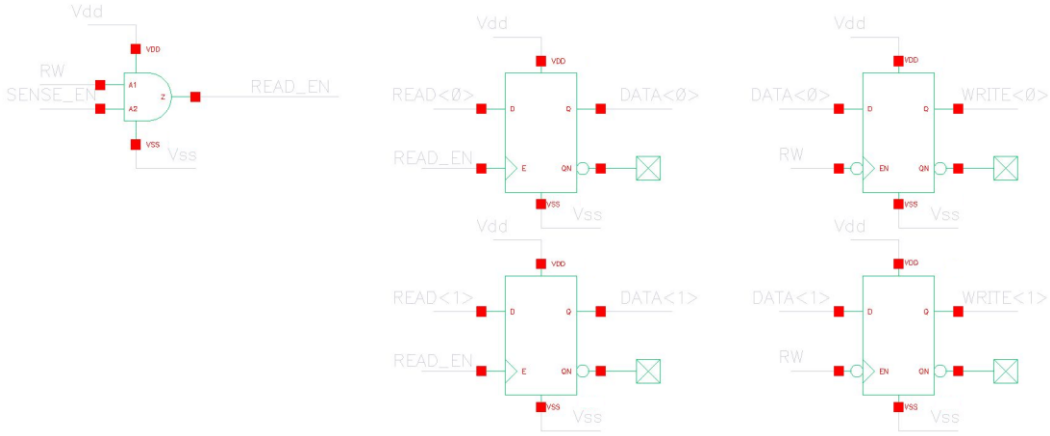


Figure 19 Design of latches on data bus.

### 3.4.1 Results

Results are presented in Figure 20 and Figure 21. Both architectures have same operating scheme. Clock period is 1 ns. The operation is as follow: on the first clock tick the controller is being initialized (time span: 0-1 ns), then on next two clock periods the data is being write to the farthest SRAM cell, on both word line and bit line (time span: 1-3ns).

The control block is synchronous and is using arbitrary timings, without any feedback from SRAM. It needs 2 clock cycles both for read and write operation. Controller consists of few latches and flip flops. The difference between quad butterfly and full butterfly implementation is that the latter uses additional AND gates to select only one memory array, thus write enable signal (WE in Figure 14 and Write driver trace in Figure 21) is stronger, because AND gates have stronger driving capabilities. Schematic of AND gates is

shown in Figure 18. Therefore one can see in the resulting plots, that write enable signal has lower rise time in quad butterfly.

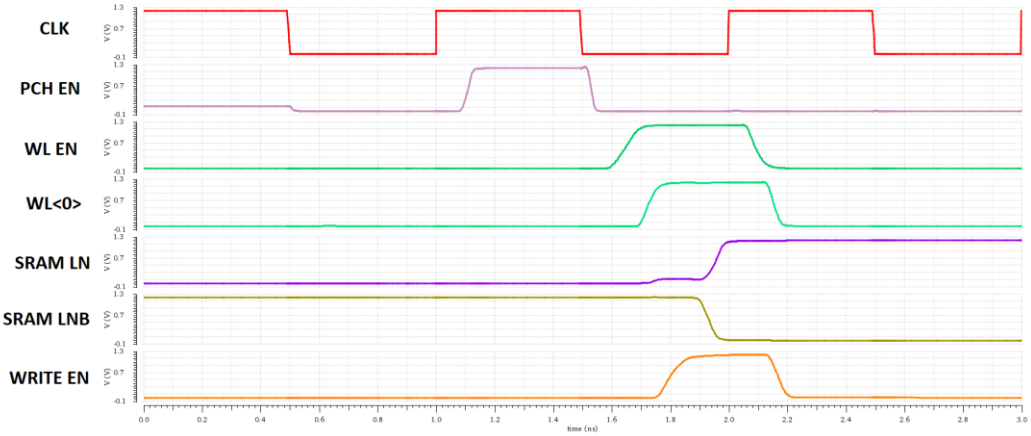


Figure 20 Plot results of write operation in quad butterfly architecture.

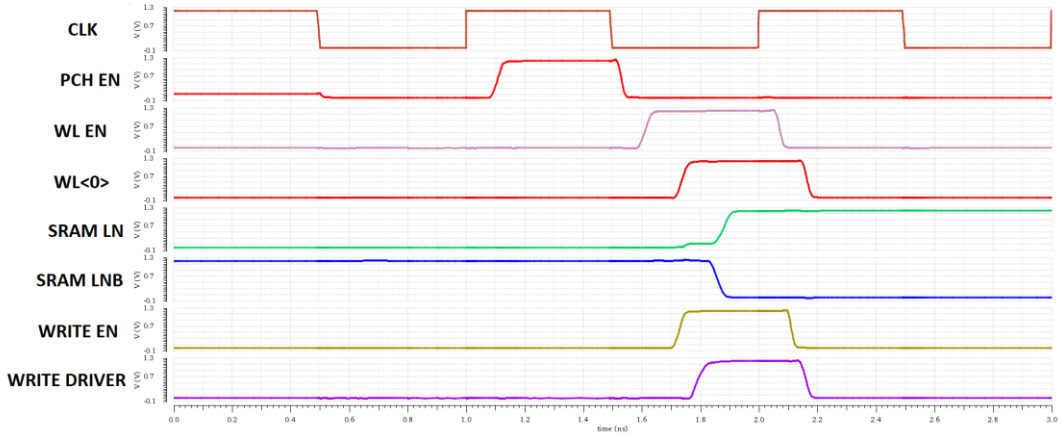


Figure 21 Plot results of write operation in full butterfly architecture.

With this design, it takes roughly 1 ns to update weights into one SRAM word. Additionally, from turning ON write driver to flipping state of memory cell, about 200 ps are required. Word line is 4 times longer in quad butterfly architecture, than it is in full butterfly architecture. Thus the former has slower rise time – 61 ps and 38 ps respectively. Rise time is not important unless IMC system by design has long word lines. Worth noting is that in full butterfly arrangement each column circuit and row circuit drives 2 memory banks.

Propagation delays are highly visible. In the plots, clock signal is driven by ideal voltage source. Since timing is based arbitrary on clock input the pre-charge stage is longer than necessary. Waveforms named SRAM LN and SRAM LNB are internals nodes of memory cell. As initial tests have shown

such logic circuit, in general, is very inefficient implementation of memory controller.

## 3.5 Improved Control circuit

The controller works correctly in basic implementation described above, nevertheless it has some room for improvements. New design, the logic control part is presented in Figure 22. In the next iteration of design, writing procedure has been reduced to one clock cycle. Precharge stage is now shorter, and its length is controlled by delay chain composed of four minimum size inverters. Write enable signal is now driven by logic gates instead of slower flip-flop. Width of precharge transistors is increased to 2 micrometres. Apart from that there was no more changes. Controller has been redesigned so that its always active (it is visible in Figure 23), namely word line is almost always enabled (it is turned off only during precharge stage). Write access time has been reduced to one clock period; therefore timing constraints are now more demanding. Worth noting is the fact that WRITE EN signal is internal to memory controller, the one that is actually present in driver circuit is delayed.

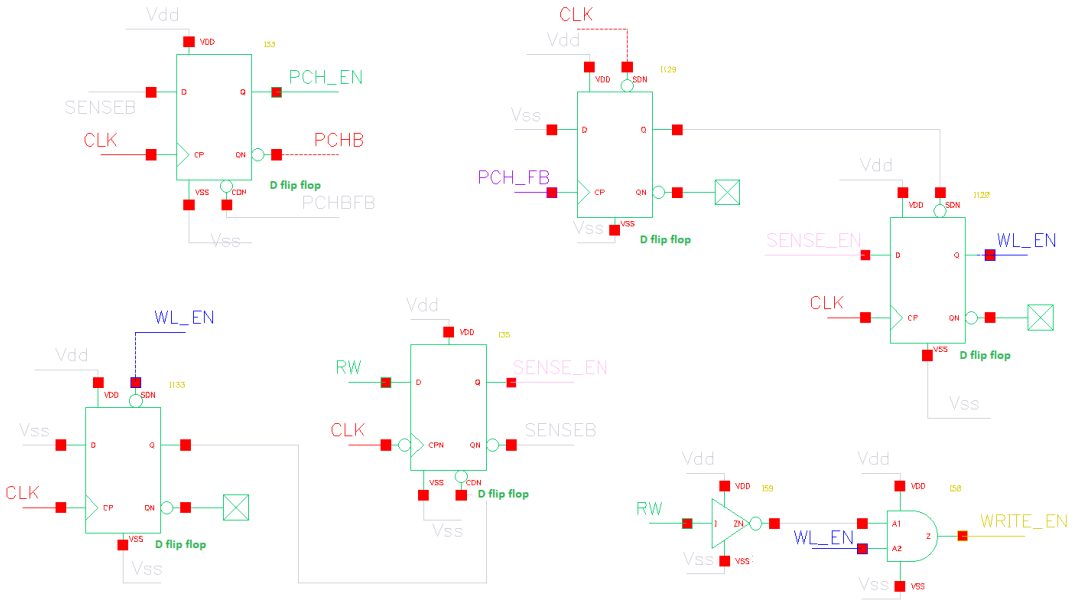


Figure 22 Part of improved controller schematic .

It can be observed, that memory cell has flipped its state before even precharge stage. This happens due to activated word line. Actual moment of valid write time is visible as a small disturbance in cell's internal nodes in a timestamp around 1.4 ns. Since writing time has been reduced, from 2 ns onward, next operation is performed.

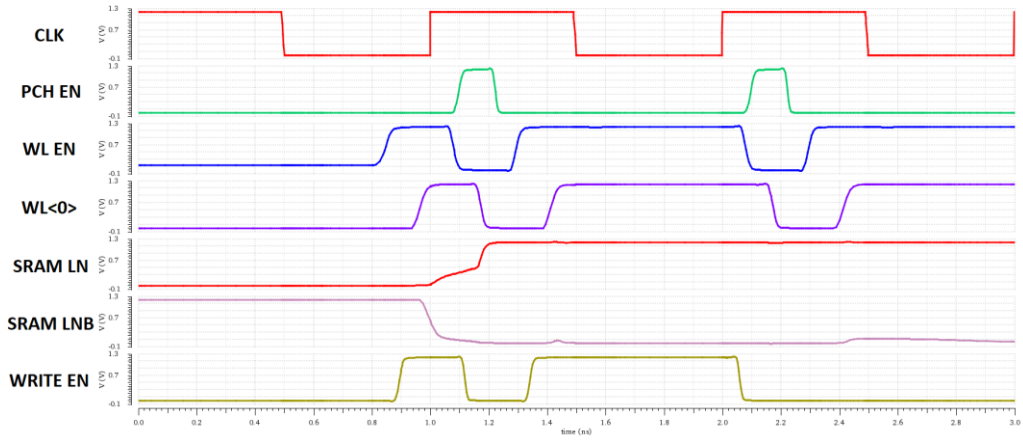


Figure 23 Plot results of the improved controller design in full butterfly arrangement.

In Figure 24 a slightly more advanced test is presented. The operation is as follow. After initialization stage, 0x01 is written to address 0x00 (0.5 – 1 ns). Then it is immediately read to confirm correctness of write operation (1 – 2 ns). Afterwards, 0x04 is written to address 0x02 (2 – 2.5 ns) and 0x06 to address 0x04 (2.5 – 3ns). After that, reading in the same order is implemented to confirm validity (3 – 4 ns, and 4 – 5 ns). Reading process takes two clock cycles. During reading, data is valid only after first clock tick, before that moment, the data line is being left floating; therefore it has random values. Even though writing procedure is not entirely correct, the data is successfully transferred to and from memory, and the results are correct.

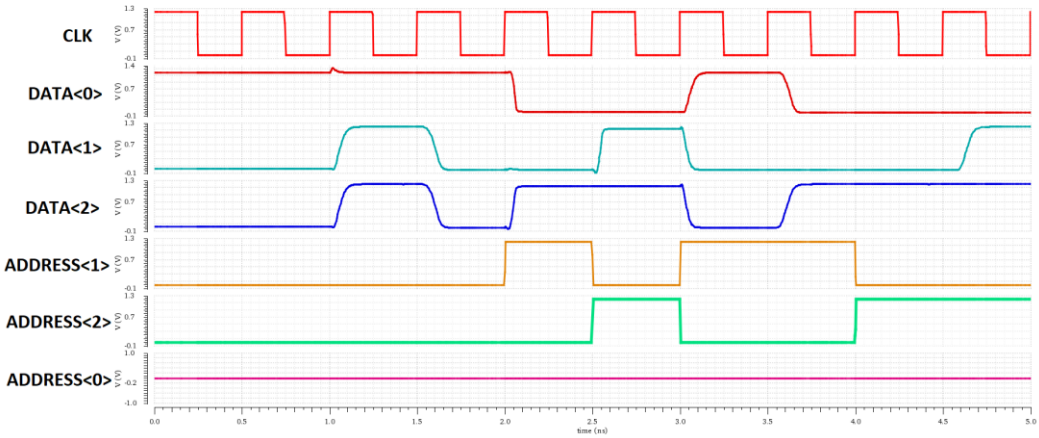


Figure 24 Writing and reading data from different addresses. Data input/output is driven by tri-state buffers, while address and clock inputs are driven by ideal voltage source.

One of the drawbacks of this controller is that it has no latches on address inputs, meaning controller relies on assumption, that external circuit provides same (valid) address during whole operation. This is not the best

assumption, as time window to change address is very narrow (in the plots, address input is driven by ideal source). Also Controller has no working implementation of power-down mode.

In Figure 25 ideal and actual signals are presented. Voltages on bit lines are presented as well. It is clearly visible how SRAM responds to writing and reading data. Neither of the signals overlap in harmful manner. For sake of visibility, precharge enable signal was shown on all previous plots, but real circuit requires inverted one. In the plot inverted precharge enable signal is marked as PCHB.

In Figure 26 simulation of different process corners and temperatures is presented. For the fastest case writing time is less than 350 ps, while for the slowest 600 ps. In the worst case operating frequency is $\frac{1}{600\ ps} \approx 1.67\ GHz$.

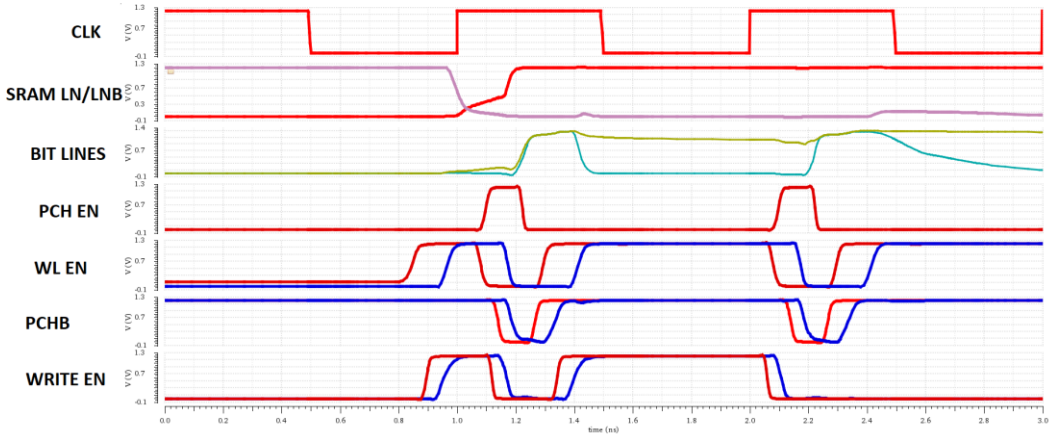


Figure 25 Internal signals of controller (red) and signals which are actually present closest to SRAM cell (blue). Figure also shows both bit lines (light blue colour is negative node).

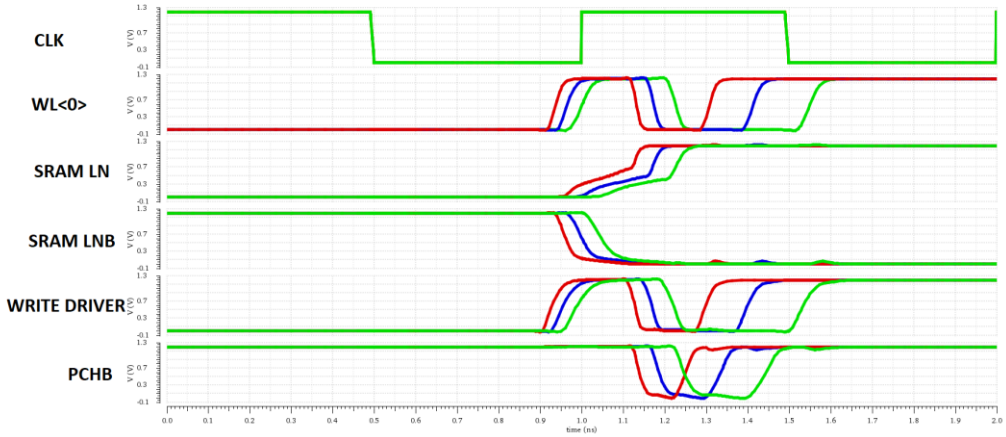


Figure 26 Different process corners and temperatures. Red colour is the fastest, green the slowest and blue is nominal.

## 3.6 Design summary

Table 1, shows a summary of results for the different designs. Write access time is also the smallest possible clock period the controller can accept. The difference between quad and full butterfly architecture is relatively small ($986\ ns - 908ns = 78\ ns$). Since in second version of design word line is enabled too soon, exact write time is hard to distinguish and approximate value is provided. Also in second version WL_EN signal, present in schematic in Figure 22, has more logic gates to drive; therefore rise time of the whole word line has worsened.

Table 1 Comparision of performance.

| | Quad Butterfly | Full butterfly | Full butterfly version 2 |
|---|---|---|---|
| Required number of clock periods for writing | 2 | 2 | 1 |
| Write access time | 986 ps | 908 ps | ≈440 ps |
| Rise time of world line | 61 ps | 38 ps | 42 ps |

# 4 Conclusions

In-memory-computing is a prominent future architecture, which will speed-up neural networks operation. Modern computer architectures are inherently bad at parallel computing due to von-Neumann bottleneck. To fully exploit advantages of IMC, technical challenges have to be overcame. One of such challenges is effective data update, stored inside IMC, which requires fast memory controller. If memory is large enough such a controller is of minor interest, but that is rarely a case, as neural networks are extensive and resource demanding.

Basic structure of IMC system, its different architectures, operation principles, challenges and implementations were discussed in this thesis. IMC can be designed in a variety of ways, that can be very different from each other. This results in many possibilities of memory layout and organization. Therefore memory controller is not universal and has to be tailored for specific system.

In this thesis, memory controller has been designed in two variants of memory arrangement and the faster one further optimized. As has been shown full butterfly architecture has better response time. Two consecutive design iterations have been described.

The first version, for the sake of comparison, bears major similarities to the controller used in quad butterfly architecture. The difference in access time is minor, only 78 ns. The main improvement of the second version is remodelled control logic, which requires only one clock period for writing operation instead of two. The precharge stage is now controlled by delay chain rather than relying on input clock (most of the timings in the first implementation are generated based on input clock). In all versions two clock periods are required for reading. In the final version the operating frequency can reach 1.67 GHz.

After all, current design has plenty of room for further optimization. Since this project has been carried blindly, without actual IMC architecture, the banks arrangement, as well as memory size cannot be exactly determined. As a further work, the design can be integrated with one of existing IMC architectures. All in all the presented design is working correctly.

# 5 References


[1] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, 'Recent advances in convolutional neural network acceleration', *Neurocomputing*, vol. 323, pp. 37–51, Jan. 2019, doi: 10.1016/j.neucom.2018.09.038.

[2] 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. Regul. Pap.*, vol. 65, no. 12, pp. 4219–4232, Dec. 2018, doi: 10.1109/TCSI.2018.2848999.

[3] N. Verma *et al.*, 'In-Memory Computing: Advances and Prospects', *IEEE Solid-State Circuits Mag.*, vol. 11, no. 3, pp. 43–55, 2019, doi: 10.1109/MSSC.2019.2922889.

[4] M. Dazzi, 'Accelerating Inference of Convolutional Neural Networks Using In-memory Computing', *Front. Comput. Neurosci.*, vol. 15, p. 19, 2021.

[5] 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, Jan. 2019, doi: 10.1109/JSSC.2018.2880918.

[6] K. Shiruru, 'An Introduction To Artificial Neural Network', *Int. J. Adv. Res. Innov. Ideas Educ.*, vol. 1, pp. 27–30, Sep. 2016.

[7] D. Ielmini and G. Pedretti, 'Device and Circuit Architectures for In-Memory Computing', *Adv. Intell. Syst.*, vol. 2, no. 7, p. 2000040, Jul. 2020, doi: 10.1002/aisy.202000040.

[8] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, 'Memory devices and applications for in-memory computing', *Nat. Nanotechnol.*, vol. 15, no. 7, pp. 529–544, Jul. 2020, doi: 10.1038/s41565-020-0655-z.

[9] S. Mittal, G. Verma, B. Kaushik, and F. A. Khanday, 'A survey of SRAM-based in-memory computing techniques and applications', *J. Syst. Archit.*, vol. 119, p. 102276, Oct. 2021, doi: 10.1016/j.sysarc.2021.102276.

[10] A. Amirsoleimani *et al.*, 'In-Memory Vector-Matrix Multiplication in Monolithic Complementary Metal–Oxide–Semiconductor-Memristor Integrated Circuits: Design Choices, Challenges, and Perspectives', *Adv. Intell. Syst.*, vol. 2, no. 11, p. 2000115, Nov. 2020, doi: 10.1002/aisy.202000115.

[11] M. R. Haq Rashed, S. K. Jha, and R. Ewetz, 'Hybrid Analog-Digital In-Memory Computing', in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, Munich, Germany, Nov. 2021, pp. 1–9. doi: 10.1109/ICCAD51958.2021.9643526.

[12] O. Numan, 'Integrated Circuit Blocks for In-Memory-Computing', Master thesis, Aalto university, 2020.

[13] Q. Zheng *et al.*, 'Lattice: An ADC/DAC-less ReRAM-based Processing-In-Memory Architecture for Accelerating Deep Convolution Neural Networks', in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, Jul. 2020, pp. 1–6. doi: 10.1109/DAC18072.2020.9218590.

[14] R. Xiao, K. Huang, Y. Zhang, and H. Shen, 'A Low Power In-Memory Multiplication andAccumulation Array with Modified Radix-4 Inputand Canonical Signed Digit Weights'. arXiv, Jan. 07, 2021. Accessed: Jun. 09, 2022. [Online]. Available: http://arxiv.org/abs/2101.02419
[15] 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, Sep. 2018, doi: 10.1109/JETCAS.2018.2829522.
[16] R. Khaddam-Aljameh, P.-A. Francese, L. Benini, and E. Eleftheriou, 'An SRAM-Based Multibit In-Memory Matrix-Vector Multiplier With a Precision That Scales Linearly in Area, Time, and Power', *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 29, no. 2, pp. 372–385, Feb. 2021, doi: 10.1109/TVLSI.2020.3037871.
[17] S. K. Gonugondla, M. Kang, and N. R. Shanbhag, 'A Variation-Tolerant In-Memory Machine Learning Classifier via On-Chip Training', *IEEE J. Solid-State Circuits*, vol. 53, no. 11, pp. 3163–3173, Nov. 2018, doi: 10.1109/JSSC.2018.2867275.
[18] N. Yadav, Y. Kim, S. Li, and K. K. Choi, 'Stable, Low Power and Bit-Interleaving Aware SRAM Memory for Multi-Core Processing Elements', *Electronics*, vol. 10, no. 21, p. 2724, Nov. 2021, doi: 10.3390/electronics10212724.
[19] M. A. Zidan, H. A. H. Fahmy, M. M. Hussain, and K. N. Salama, 'Memristor-based memory: The sneak paths problem and solutions', *Microelectron. J.*, vol. 44, no. 2, pp. 176–183, Feb. 2013, doi: 10.1016/j.mejo.2012.10.001.
[20] M. Demler, 'Mythic Multiplies In A Flash', *Microprocess. Rep.*, p. 3, Aug. 2018.
[21] 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. 54, no. 6, pp. 1789–1799, Jun. 2019, doi: 10.1109/JSSC.2019.2899730.
[22] H. Jia, H. Valavi, Y. Tang, J. Zhang, and N. Verma, 'A Programmable Heterogeneous Microprocessor Based on Bit-Scalable In-Memory Computing', *IEEE J. Solid-State Circuits*, vol. 55, no. 9, pp. 2609–2621, Sep. 2020, doi: 10.1109/JSSC.2020.2987714.
[23] 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. Regul. Pap.*, vol. 67, no. 8, pp. 2521–2531, Aug. 2020, doi: 10.1109/TCSI.2020.2981901.
[24] Z. Lin *et al.*, 'A review on SRAM-based computing in-memory: Circuits, functions, and applications', *J. Semicond.*, vol. 43, no. 3, p. 031401, Mar. 2022, doi: 10.1088/1674-4926/43/3/031401.
[25] Z. Jiang, S. Yin, J.-S. Seo, and M. Seok, 'C3SRAM: An In-Memory-Computing SRAM Macro Based on Robust Capacitive Coupling Computing Mechanism', *IEEE J. Solid-State Circuits*, vol. 55, no. 7, pp. 1888–1897, Jul. 2020, doi: 10.1109/JSSC.2020.2992886.

[26] 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. VLSI Syst.*, vol. 27, no. 11, pp. 2556–2567, Nov. 2019, doi: 10.1109/TVLSI.2019.2929245.
[27] W. Kester and Analog Devices, inc, Eds., *Data conversion handbook*. Amsterdam ; Boston: Elsevier ; Newnes, 2005.
[28] P. E. Allen and D. R. Holberg, *CMOS analog circuit design*, 3rd ed. New York ; Oxford: Oxford University Press, USA, 2012.
[29] T. C. Carusone, D. Johns, K. W. Martin, and D. Johns, *Analog integrated circuit design*, 2nd ed. Hoboken, NJ: John Wiley & Sons, 2012.
[30] J. D. Norris and G. Von Dolteren, 'Understanding Current Output Digital to Analog Converters', INTERSIL, Application Note AN9845, Jun. 1999.
[31] I. Sperotto, H. Klimach, and S. Bampi, 'MOS-only M-2M DAC for ultra-low voltage applications', in *2015 IEEE 6th Latin American Symposium on Circuits & Systems (LASCAS)*, Montevideo, Uruguay, Feb. 2015, pp. 1–4. doi: 10.1109/LASCAS.2015.7250471.
[32] K. Vleugels, 'EE315B VLSI Data Conversion Circuits', Stanford University, Autumn 2011.
[33] H. Zumbahlenas and Analog Devices, inc, Eds., *Linear circuit design handbook*. Amsterdam ; Boston: Elsevier/Newnes Press, 2008.
[34] J. J. Patel, 'Comparative Study Of Current Steering Dac Based On Implementation Using Various Types Of Switches', p. 9.
[35] Z. Sun and R. Huang, 'Time Complexity of In-Memory Matrix-Vector Multiplication', *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 68, no. 8, pp. 2785–2789, Aug. 2021, doi: 10.1109/TCSII.2021.3068764.
[36] E. Mejia, K. Duke, and N. Kommaraju, 'TI Precision Designs: Verified Design ±10V 4-Quadrant Multiplying DAC', p. 23, Oct. 2013.
[37] B. Ding, H. Qian, and J. Zhou, 'Activation functions and their characteristics in deep neural networks', in *2018 Chinese Control And Decision Conference (CCDC)*, Shenyang, Jun. 2018, pp. 1836–1841. doi: 10.1109/CCDC.2018.8407425.
[38] S. Sharma, S. Sharma, and A. Athaiya, 'Activation Functions In Neural Networks', *Int. J. Eng. Appl. Sci. Technol.*, vol. 04, no. 12, pp. 310–316, May 2020, doi: 10.33564/IJEAST.2020.v04i12.054.
[39] J. Feng and S. Lu, 'Performance Analysis of Various Activation Functions in Artificial Neural Networks', *J. Phys. Conf. Ser.*, vol. 1237, no. 2, p. 022030, Jun. 2019, doi: 10.1088/1742-6596/1237/2/022030.
[40] S. Yu, H. Jiang, S. Huang, X. Peng, and A. Lu, 'Compute-in-Memory Chips for Deep Learning: Recent Trends and Prospects', *IEEE Circuits Syst. Mag.*, vol. 21, no. 3, pp. 31–56, 2021, doi: 10.1109/MCAS.2021.3092533.
[41] A. Heydari and S. N. Balakrishnan, 'Finite-Horizon Control-Constrained Nonlinear Optimal Control Using Single Network Adaptive Critics', *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 1, pp. 145–157, Jan. 2013, doi: 10.1109/TNNLS.2012.2227339.

[42] N. Nambiar, B. J. Blalock, and M. Nance Ericson, 'A novel current-mode multi-channel integrating ADC', *Analog Integr. Circuits Signal Process.*, vol. 63, no. 2, pp. 283–291, May 2010, doi: 10.1007/s10470-009-9393-8.
[43] J. Fredenburg and M. P. Flynn, 'ADC trends and impact on SAR ADC architecture and analysis', in *2015 IEEE Custom Integrated Circuits Conference (CICC)*, San Jose, CA, USA, Sep. 2015, pp. 1–8. doi: 10.1109/CICC.2015.7338380.
[44] M. F. Snoeij, A. J. P. Theuwissen, K. A. A. Makinwa, and J. H. Huijsing, 'Multiple-Ramp Column-Parallel ADC Architectures for CMOS Image Sensors', *IEEE J. Solid-State Circuits*, vol. 42, no. 12, pp. 2968–2977, Dec. 2007, doi: 10.1109/JSSC.2007.908720.
[45] M. R. Elmezayen, B. Wu, and S. U. Ay, 'Single-Slope Look-Ahead Ramp ADC for CMOS Image Sensors', *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 67, no. 12, pp. 4484–4493, Dec. 2020, doi: 10.1109/TCSI.2020.3007882.
[46] F. Z. Nelson, M. N. Alam, and S. U. Ay, 'A Single-Slope Look-Ahead Ramp (SSLAR) ADC for Column Parallel CMOS Image Sensors', in *2009 IEEE Workshop on Microelectronics and Electron Devices*, Boise, Idaho, USA, Apr. 2009, pp. 1–4. doi: 10.1109/WMED.2009.4816152.
[47] M. Madhavilatha, G. L. Madhumati, and K. R. K. Rao, 'Design of CMOS Comparators for FLASH ADC', *Int. J. Electron. Eng.*, no. 1, pp. 53–57, 2009.
[48] 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, Apr. 2017, doi: 10.1109/JSSC.2016.2642198.
[49] M. Yamaguchi, G. Iwamoto, Y. Nishimura, H. Tamukoh, and T. Morie, 'An Energy-Efficient Time-Domain Analog CMOS BinaryConnect Neural Network Processor Based on a Pulse-Width Modulation Approach', *IEEE Access*, vol. 9, pp. 2644–2654, 2021, doi: 10.1109/ACCESS.2020.3047619.
[50] A. Lu, X. Peng, Y. Luo, and S. Yu, 'Benchmark of the Compute-in-Memory-Based DNN Accelerator With Area Constraint', *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 28, no. 9, pp. 1945–1952, Sep. 2020, doi: 10.1109/TVLSI.2020.3001526.
[51] H. Jiang *et al.*, 'A Two-way SRAM Array based Accelerator for Deep Neural Network On-chip Training', in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, Jul. 2020, pp. 1–6. doi: 10.1109/DAC18072.2020.9218524.
[52] S. Ataei and R. Manohar, 'AMC: An Asynchronous Memory Compiler', in *2019 25th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, Hirosaki, Japan, May 2019, pp. 1–8. doi: 10.1109/ASYNC.2019.00009.
[53] M. R. Guthaus, J. E. Stine, S. Ataei, B. Chen, B. Wu, and M. Sarwar, 'OpenRAM: an open-source memory compiler', in *Proceedings of the*

*35th International Conference on Computer-Aided Design*, Austin Texas, Nov. 2016, pp. 1–6. doi: 10.1145/2966986.2980098.
[54] C. Li *et al.*, 'Analogue signal and image processing with large memristor crossbars', 2018, doi: 10.1038/S41928-017-0002-Z.
[55] B. Feinberg, U. K. R. Vengalam, N. Whitehair, S. Wang, and E. Ipek, 'Enabling Scientific Computing on Memristive Accelerators', in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, Los Angeles, CA, Jun. 2018, pp. 367–382. doi: 10.1109/ISCA.2018.00039.
[56] R. Gauchi *et al.*, 'Memory Sizing of a Scalable SRAM In-Memory Computing Tile Based Architecture', in *2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC)*, Cuzco, Peru, Oct. 2019, pp. 166–171. doi: 10.1109/VLSI-SoC.2019.8920373.
[57] D. Balobas and N. Konofaos, 'Design and evaluation of 6T SRAM layout designs at modern nanoscale CMOS processes', p. 5, 2015.
[58] S.-J. Byun *et al.*, 'A Low-Power Analog Processor-in-Memory-Based Convolutional Neural Network for Biosensor Applications', *Sensors*, vol. 22, no. 12, p. 4555, Jun. 2022, doi: 10.3390/s22124555.
[59] K. Zhang *et al.*, 'SRAM design on 65-nm CMOS technology with dynamic sleep transistor for leakage reduction', *IEEE J. Solid-State Circuits*, vol. 40, no. 4, pp. 895–901, Apr. 2005, doi: 10.1109/JSSC.2004.842846.
[60] J. Singh, S. P. Mohanty, and D. K. Pradhan, *Robust SRAM Designs and Analysis*. New York, NY: Springer New York, 2013. doi: 10.1007/978-1-4614-0818-5.