A Multiply and Accumulate Scheme for SRAM-based In-Memory Computing

Karagiannidis Christos

Thesis

Supervisor: Yiorgos Tsiatouhas

Ioannina, March 2024

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Περίληψη

Στην παρούσα εργασία ασχολούμαστε με μια XNOR-SRAM τοπολογία, η οποία είναι μια μεικτού σήματος δομή για υπολογισμό εντός της μνήμης, αποτελούμενη από ένα κελί και κατάλληλη αρχιτεκτονική SRAM ώστε να επιτελεστεί η XNOR-and-accumulate (XAC) πράξη σε δυαδικά βαθιά νευρωνικά δίκτυα, χωρίς να απαιτείται η πρόσβαση στα δεδομένα γραμμή προς γραμμή. Κάθε κελί υπολογίζει τον πολλαπλασιασμό ανάμεσα στην τιμή που είναι αποθηκευμένη στο υποκελί μνήμης (βάρος) και της παρεχόμενης εισόδου. Τα αποτελέσματα από κάθε κελί της ίδιας στήλης συσσωρεύονται σε μια κοινή γραμμή (Read Bit Line – RBL) μέσω της ταυτόχρονης ενεργοποίησης και όλων των γραμμών της μνήμης, δημιουργώντας κατά αυτόν τον τρόπο έναν διαιρέτη τάσης. Η αναλογική τάση που δημιουργείται ψηφιοποιείται με έναν Analog to Digital (ADC) μετατροπέα εξωτερικά από το πλέγμα των κελιών. Το προτεινόμενο σχήμα σχεδιάστηκε στην τεχνολογία 90-nm της UMC και επιτυγχάνει τον υπολογισμό της τιμής XAC χρησιμοποιώντας δύο γραμμές δεδομένων ανά σειρά και μια γραμμή ελέγχου ανά στήλη, με βάση ένα κελί 11 τρανζίστορ (11T), αλλά με κόστος την απαίτηση μιας πρόσθετης τροφοδοσίας. Εναλλακτικά, προτείνουμε μια δεύτερη σχεδίαση με ένα πρόσθετο τρανζίστορ PMOS, δηλ. συνολικά 12 τρανζίστορ (12T), και μια δεύτερη γραμμή ελέγχου ανά στήλη, επομένως απαιτείται μια σχετική αύξηση της επιφάνειας πυριτίου, αλλά χωρίς την απαίτηση για μια πρόσθετη τροφοδοσία.

Λέξεις Κλειδιά: Δυαδικά βαθιά νευρωνικά δίκτυα, υπολογισμός εντός της μνήμης, SRAM, πολλαπλασιασμός και συσσώρευση, XNOR και συσσώρευση.

Abstract

In this thesis we deal with an XNOR-SRAM topology, a mixed-signal In Memory Computing (IMC) SRAM cell and architecture that performs XNOR-and-accumulate operations for binary deep neural networks without row-by-row data access. Each cell calculates the multiplication between the value stored in its memory subcell (weight) and the input provided. The results from each cell in the same column are accumulated on a Read Bit Line (RBL) via a concurrent activation of all rows, forming a resistive voltage divider. The analog voltage is digitized with an Analog to Digital converter (ADC) outside of the cell array. The proposed scheme designed in the 90-nm technology of UMC, and it achieves to calculate the XAC value utilizing two data lines per row and a single control line per column using 11 transistors (11T), but at the cost of an extra power supply. Alternatively, we propose another design with one additional PMOS transistor for a total of 12 (12T) and a second control line per column, therefore a moderate increase in silicon area but no extra power supply is required.

Keywords: Binary deep neural networks, in memory computing, SRAM, Multiply and Accumulate (MAC), XNOR and Accumulate (XAC)

Contents

Chapter 1. Introduction

1.1 Objectives

As deep neural networks (DNNs) continue to demonstrate improvements, deep learning is shifting toward edge computing. This development has motivated works on low-resource machine learning algorithms and their accelerating hardware. The most common DNN operations are Multiply-and-Accumulate (MAC) which dominate in power and delay while also having high parallelism capabilities. However, the amount of memory access limits the energy efficiency of such accelerators. As a result, in-memory computing (IMC) has become increasingly appealing because it attempts to make calculations within the memory array itself. One of the memory types commonly used for IMC applications is SRAM. Recent SRAM IMC works show significant energy efficiency and throughput advantages over conventional architectures. In this thesis we attempt to provide a viable solution to XNOR SRAM IMC, a bit cell design and architecture that calculates MAC operations on a common RBL. Our first proposed design is an 11 transistor (11T), featuring a 6T memory subcell, two full pass-gates and one NMOS pass-gate while requiring only two datalines and one control line for the computational subcell. We also propose an alternative 12T design that changes the NMOS pass-gate into another full pass-gate hence requiring another control signal to drive the additional PMOS transistor.

1.2 Organization

This thesis is structured as follows. Chapter 2 introduces the In-memory computing concept as well as the different memory types that have been considered as possible candidates. Related work to SRAM-Based In-memory computing in particular for bitwise XNOR operations is presented in Chapter 3. The proposed 11T and 12T SRAM bitcells along with related operation and macro architecture are described in Chapter 4. Chapter

5 presents the simulation results for the proposed designs comparing it to a similar well known XNOR-SRAM work in the literature. Finally the conclusions are given in Chapter 6.

Chapter 2. In-Memory Computing

2.1 Introduction to In Memory Computing (IMC)

A series of Boolean logic operations carried out in silicon by complementary metal-oxide-semiconductor (CMOS) technology typically handles data processing in digital computers. For the past 40 years, the CMOS transistor has been scaling steadily according to Moore's law, which states that as transistor size decreases, less area is consumed, which lowers fabrication costs. A decrease in power consumption and an increase in operation frequency coincided with the downscaling of transistor sizes, which improved circuit performance over the years [M65]. Regrettably, decreasing the device area also results in a rise in power density, which has slowed the CMOS scaling trend for the past ten years [H14].

Emerging applications of signal processing and machine learning are driving platforms to their limits in terms of throughput and energy efficiency due to their computing demands. Digital accelerators are able to achieve 10–100× higher energy efficiency and speed than general-purpose processors, demonstrating the need of hardware specialization. These benefits, however, are mostly related to computation rather than memory access or data movement in general. Applications that involve big data sets or high-dimensionality data structures are becoming more and more datacentric. This causes the focus to shift to data mobility, which regrettably places restrictions on the potential benefits of traditional digital acceleration.

One example of this is the well-known "memory wall"[SCC+19]. Though it is frequently related to off-chip memory access, the issue is more basic and arises in any design that divides memory and computation. The amount of space needed to store data in physical bit cells increases with the volume of data to be stored. Moving data outside of memory, from the site of storage to the point of calculation, results in a proportional communication cost.

Any architecture built around separate memory and computing module is commonly referred to as “Von Neumann” architecture. Historically there have been two types of computers, Fixed Program Computers whose function is very specific and they can’t be reprogrammed such as calculators, and Stored Program Computers. The stored-program concept was introduced by John Von Neumann and is the foundation for almost every modern computer. In this concept, programs and data are stored in a separate storage unit and the calculations are made in a separate computational unit. Whatever we do to enhance performance, we cannot get away from the fact that instructions can only be done one at a time (for each computing unit) and can only be carried out sequentially. Both of these factors hold back the competence of the CPU. This is commonly referred to as the ‘Von Neumann bottleneck’ [DMB14]. We can provide a Von Neumann processor with more cache, more RAM, or faster components but if original gains are to be made in CPU performance, then an influential inspection needs to take place of CPU configuration.

In Figure 2.1 we can see a basic CPU structure using the Von Neumann architecture. Over the past years we made severe improvements in every single one of these units, however with the recent emergence of data intensive applications such as AI the transfer of data itself proved to be a much bigger challenge than originally anticipated.

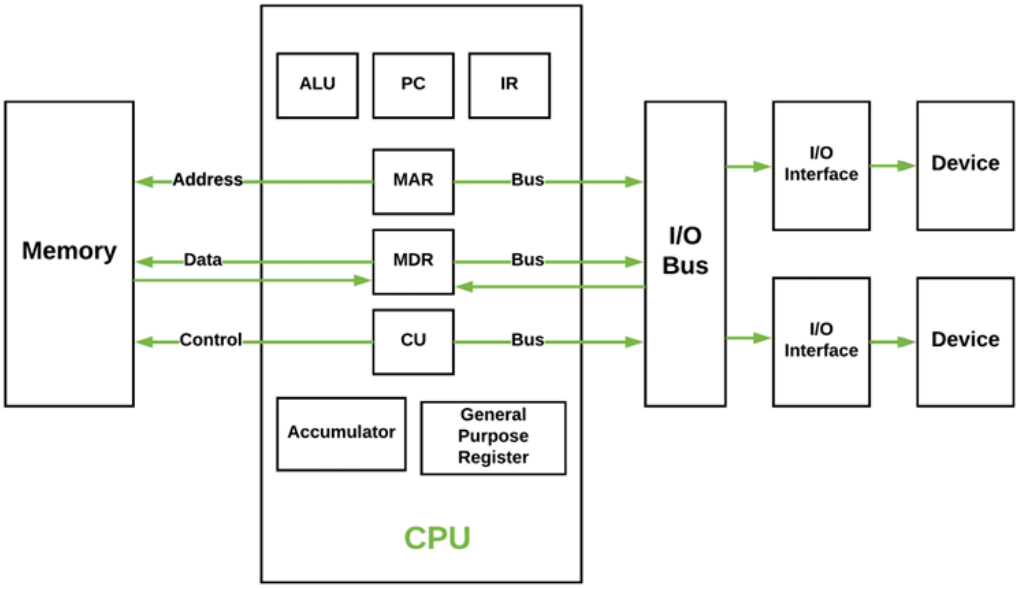


Fig. 2.1 Typical Von Neumann architecture

Today's memory hierarchy usually consists of multiple levels of cache, the main memory, and storage. The traditional approach is to move data up to caches from the storage and then process it. In contrast, near-memory computing (NMC) aims at processing close to where the data resides. This data-centric approach couples compute units close to the data and seek to minimize the expensive data movements. Figure 2.2 depicts the system evolution based on the information referenced by a program during execution, which is referred to as a working set [HXT+15]. Prior systems were based on a CPU-centric approach where data is moved to the core for processing (Figure 2.2 (a), (b)), whereas now with near-memory processing (Figure 2.2 (c)) the processing cores are brought to the place where data resides. The idea of processing close to the memory dates back to the 1960s [S70]; however the first appearance of NMC systems can be traced back to the early 1990s [ESS92]. Although promising results were obtained, the adoption of these NMC systems remained limited. One of the main reasons was attributed to technological limitations.

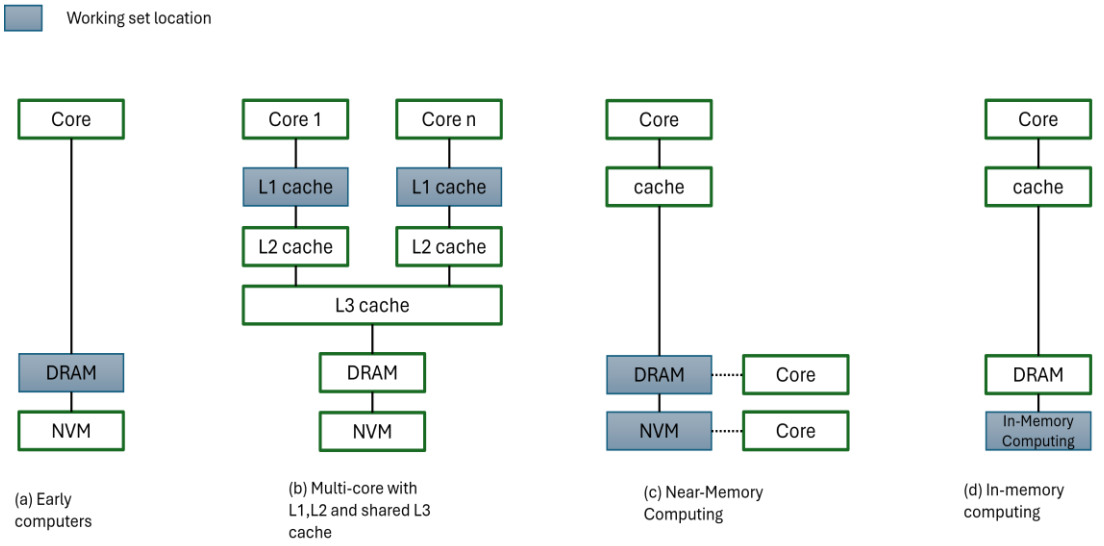


Fig. 2.2 Classification of computing systems based on working set location

After almost three decades research in NMC is regaining attention due to technological advancements in 3D integration technologies that blends logic and memory in the same package that make NMC more viable. Processing closer to the data can significantly diminish the data movement problem of data-intensive applications. However, this did not completely eliminate the need for transferring data even though it seriously reduced the bottleneck of data transfer. In the literature, NMC has manifested with names such as processing-in memory (PIM), near-data processing (NDP), near

memory processing (NMP), or in case of non-volatile memories as in-storage processing (ISP). However, all these terms fall under the same umbrella of near-memory computing (NMC) with the core principle of doing processing close to the memory. With the arrival of true on-site computing called in-memory computing, it would be prudent to merge all these synonyms mentioned above for the same concept under the umbrella of in/near-memory computing.

This necessitates a major departure from the established systems so that memory and computation are collocated. This contradicts the traditional computer architecture, which uses a central processing unit (CPU) to process data and execute programs that are retrieved from working memory, according to the Von Neumann model [ZSL18]. Therefore, in-memory computing (IMC) can be defined as a computer architecture in which data operations are carried out directly within the data memory array itself, rather than having to be transferred to CPU registers first. For data-intensive activities, the main memory, which is usually a dynamic random-access memory (DRAM), is placed on a physically separate module. In-Memory computing reduces the latency required for instruction and data fetching as well as memory write backs, solving the pipeline bottleneck of traditional von Neumann computers. Another significant benefit of IMC is high computational parallelism, due to the specific architecture of the memory array, where computation can occur simultaneously over several current pathways. IMC also benefits from the high density of the memory arrays with computational devices, which generally feature excellent scalability. Finally, analog computing is supported by the physical laws of memory circuits, such as the Ohm's law for product [IP18] and the Kirchhoff's law of current summation, as well as other memory-specific physical behavior such as nonlinear threshold-type switching, pulse accumulation, and time measurement [BAW+15]. Thanks to the combination of on-site, high-density, parallel, physical, and analog data processing, IMC appears as one of the most promising approaches for computing in the context of AI and big data.

Several memory technologies have recently surfaced as high-density, low-power, low-cost, high-speed computing and storage systems based on the materials they consist of, each with their own sets of advantages and drawbacks [IP18]. An overview of a few memory types that have been explored is shown in Figure 2.2. Some of those technologies include the resistive-switching random access memory (RRAM), the ferroelectric random-access memory (FERAM), the magnetic random-access memory (MRAM), and the phase-change memory (PCM) and finally, we will explain static random-access memory (SRAM) as well.

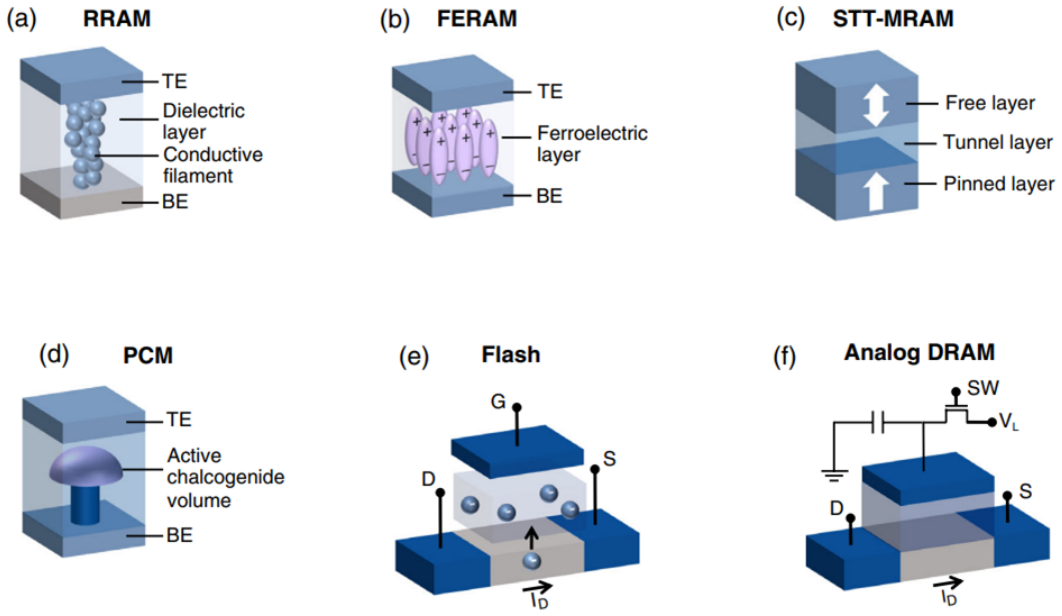


Fig. 2.3 Memory types commonly used for IMC

2.1.1 RRAM Devices

The RRAM device depicted in Figure 2.3(a) consists of a stack of metallic bottom electrode (BE), insulating metal-oxide layer and metallic top electrode. Due to the oxide layer's insulation, the resulting metal-insulator-metal (MIM) structure exhibits a comparatively high resistance. A different high-resistance material may occasionally be used in its place. First, a soft breakdown operation is used to electrically construct the MIM device. This process results in a localized change in the material composition or an increase in the concentration of defects, such as oxygen vacancies in the metal oxide. The device enters a low resistance state (LRS) as a result of the forming procedure, which often results in the accumulation of a conductive filament with a conductance greater than that of the initial insulating layer. The reset operation can electrically decrease the conductive filament's conductivity, resulting in the device's high resistance state (HRS), while the set transition can increase it to restore the LRS. While set and reset operations in unipolar RRAM devices have the same polarity, set and reset transitions in bipolar RRAM devices are driven by voltage pulses of opposite polarities. There are other RRAM devices with uniform switching in which the oxide layer

modification covers the entire region rather than just a small section of the filament. [IP20]

2.1.2 FERAM Devices

The FERAM device concept is based on ferroelectric (FE) materials where the electrostatic polarization can be reversibly switched by the application of an external electric field (Figure 2.3(b)). The materials used, however, are not very compatible with the CMOS process line, have a high leakage rate, and a relatively small bandgap. Recent breakthroughs in FE materials have rekindled interest in FE phenomena and materials for IMC and storage applications. [BMB+11]. An FE tunnel junction (FTJ), like an MTJ, works by connecting the FE switching layer in series with a dielectric layer to transform a residual FE polarization into a resistance signal. It is simple to program the FTJ structure by applying voltage pulses. Local differences in the coercive fields between different crystalline grains and domains within the FE material can impact FERAM uniformity even in the presence of nonfilamentary switching within the FE layer. [IP20]

2.1.3 MRAM

The MRAM device stores data by changing the magnetic polarization within a layer of ferromagnetic material (Figure 2.3(c)). MRAM cells consist of magnetic tunnel junctions (MTJ) which is a stack of two ferromagnetic layers separated by a thin insulating layer. One of the ferromagnetic layers has a fixed magnetic orientation (polarization), while the other has a free magnetic orientation. The resistance of the MTJ is comparatively low for parallel magnetization between two layers, while it is very high for antiparallel magnetization. Reading data from an MRAM cell depends on which resistance state it is in, ‘1’ for low resistance and ‘0’ for high resistance. Writing data requires applying a magnetic field generated by passing a current through a write line near the MTJ. The magnetic field influences the free layer’s magnetization, causing it to align with or against the fixed layer. Another writing method that relies on the phenomenon of spin-transfer torque can be used to alter the magnetization of the free layer. Those memories are otherwise identical to MRAM but they are called STT-RAM from their different write operation. Because of their ability to transition quickly STT-RAM devices are a great option to replace static RAM used as last-level cache (LLC). However, MRAM often exhibits a narrow resistance window which poses challenges for the implementation of some IMC algorithms. [IP20]

2.1.4 PCM

In the PCM device, the microstructure of a phase-change material can be reversibly switched between a crystalline phase and an amorphous phase (Figure 2.3.(d)). The crystalline phase has low resistivity, whereas the amorphous phase exhibits a disorder-induced high resistivity. As a result, a straightforward voltage/current sensing technique can be used to determine the PCM state. The PCM relies on the bulk characteristics of the active material as opposed to the filamentary switching process of the RRAM. This typically results in a broader resistance window and the possibility to operate the device using a multilayer cell (MLC) architecture. However, in order to speed up phase transitions like melting and crystallization, a significant Joule heating is typically required. This means that relatively large currents are required for programming or erasing the device. Scaling the PCM's active zone is necessary to lower the programming current. Resistance drift, in which the device resistance rises over time following programming as a result of the amorphous phase's structural relaxation, is a serious issue for the PCM. Improved stability against drift device solutions have been developed and demonstrated in IMC.

2.1.5 Flash Memory

The majority of nonvolatile memory devices used for high-density storage in solid state drives (SSDs) are based on Flash memory. A flash memory is based on a specific type of transistor known as floating-gate transistor, which is a type of metal-oxide-semiconductor field-effect transistor (MOSFET), see Figure 2.3(e). It consists of a control gate, a source, a drain, and a floating gate which is insulated from the other elements by a thin oxide layer. In a typical flash memory cell, information is stored by trapping electric charge on the floating gate. The charge affects the threshold voltage of the transistor, influencing its conductance. This attribute can be utilized in IMC since it can be used as a variable resistance [IP20].

2.1.6 DRAM

Another type of memory commonly used is Dynamic Random Access Memory (DRAM). DRAM is a type of volatile computer memory that stores each bit of data in a separate capacitor within an integrated circuit (Figure 2.3(f)). At transistor level, DRAM relies on the structure of a specific transistor known as a one-transistor capacitor (1T1C) cell. The transistor acts as a switch (access transistor) to control the flow of charge between the capacitor and the bitline. The capacitor stores electrical charge,

representing the binary state of the memory cell (charged or uncharged). The transistor's gate is connected to the wordline, and the source and drain are connected to the capacitor and bitline, respectively [IP20].

2.1.7 SRAM

Finally, Static Random Access Memory (SRAM) is also a volatile memory used in computers, and it retains data as long as power is supplied. Unlike DRAM, SRAM does not require periodic refreshing, and each bit of data in SRAM is typically stored using a more complex structure called a bistable latch, formed by a group of transistors. A bistable latch is essentially a pair of cross-coupled inverters. A basic SRAM cell consists of six transistors: two for each inverter (T1, T3 and T2, T4) and two access transistors (T5, T6). The inverters are connected in a cross-coupled manner, creating a positive feedback loop that allows the bistable latch to store a bit of information [IP20]. Figure 2.4 shows the most common six transistor (6T) SRAM memory cell. To write data into this SRAM cell, the wordline (connected to the gates of the access transistors) is activated, allowing write signals to reach the inverters from the bitline (BL) and bitline bar (BLB), setting its state. During a read operation, the wordline is activated, and the values from the latch are transferred to the bitlines, allowing the stored data to be sensed and read by an external sense amplifier. The positive feedback loop formed by the two inverters maintains the stability of the stored state. In general, SRAM memory cells are more stable and faster than DRAM cells, but this comes at a greater cost due to the 6T used greatly increasing the silicon surface, making it very expensive for larger memory capacities [YS20].

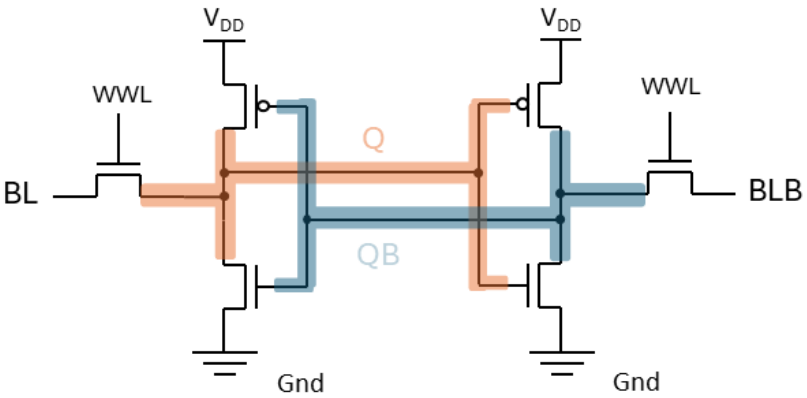


Fig. 2.4 A typical 6T SRAM cell

Chapter 3. SRAM-Based IMC

One of the key advantages of SRAM based IMC lies in its ability to perform parallel processing on a massive scale. By distributing computation across thousands or even millions of SRAM cells within memory arrays, complex operations can be executed in parallel, leading to unprecedented levels of performance. This parallelism is particularly well-suited for tasks such as machine learning, where vast amounts of data must be processed simultaneously. Operations within the array can be carried out using a few different methods among which Voltage accumulation, bitline discharging and charge sharing all of which have been previously utilized.

In this chapter we will explore the state-of-the-art on SRAM-based IMC. [ZWV17] put the SRAM-based IMC implementation strategy into practice, considering ML applications. According to their approach, the weights are written in the memory array, so that multiply and accumulate (MAC) calculations can be accomplished by activating simultaneously several data word lines for local multiplications with the weights in the cell and accumulating the multiplication result as a current on a common bitline in each memory column. A current-domain IMC that supported 4bit weight was created by [KKS+14] by varying the WL signal's pulse width (WL signal ON time). According to their approach the Least Significant Bit's (LSB) WL is enabled for T0=Tmin seconds T1=2Tmin, T2=4Tmin and T3=8Tmin where T3 is the duration of the WL pulse for the Most Significant Bit (MSB). That way in a precharged common RBL (at VDD/2) the LSB will charge/discharge the RBL for the least time while MSB will charge/discharge for the most time. This translates to a ΔV_{LSB} voltage difference for the LSB on the RBL while the MSB has an $8 \times \Delta V_{LSB}$.Unfortunately, 6T SRAM may experience read disturbance while reading two or more rows. This means that data stored in the bit cells in the same column may be flipped when the bitline voltage level drops below the threshold during the accumulation operation.

To address this issue, [L20] suggested a programmable 8T SRAM that uses the double-WL structure and lowers the WL voltage to lower the likelihood of reading disturb (data damage). To maintain memory density and avoid read interference, [YYK+20] presented an 8T SRAM configuration that included extra read transistors to separate weight read and write operations from MAC calculations at the cost of three additional data lines. To address the issue of read interference, [BC19] suggested a 10T SRAM bit cell with two decoupling read ports. Furthermore, they created a charge-domain IMC, where accumulation is accomplished by charge-sharing among local capacitors and analog multiplication is carried out on local capacitors. Figure 3.1 illustrates different SRAM-based IMC cells that have been proposed to implement XOR and XNOR operations in particular.

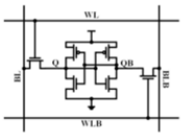
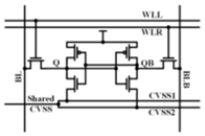
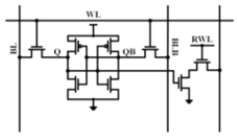
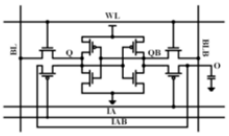
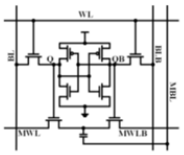
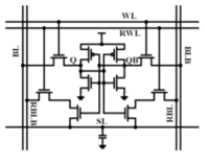
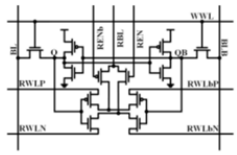
	[KKK+19]	[SKC+19]	[SEN+21]	[VRN+19]
SRAM Bitcell Schematic				
Bitcell Size	Moderate (Modified Split WL 6T)	Moderate (Modified Split WL and VSS 6T)	Moderate (Foundry 8T)	Large (8T+1MOSCAP)
Bitcell Logic	XNOR	XNOR	XOR	XNOR
MAC Operation	Voltage-Mode (Pull-Up & Pull-Down DRV)	Current-Mode (Bitline Discharging)	Current-Mode (Bitline Discharging)	Charge-Domain (Charge Sharing)
Key Features	(+) Compact Bitcell (-) Read Disturb	(+) Compact Bitcell (-) Read Disturb	(+) Foundry Cell, Read Disturb Elimination (-) Multiple Cells for Logic Operation	(+) Less Variation, Wide Dynamic Range (-) Low Throughput, Charge Injection
	[JYS+20]	[AJR+19]	[YS20]	
SRAM Bitcell Schematic				
Bitcell Size	Large (8T+1MOSCAP)	Large (10T+1MOSCAP)	Large (12T)	
Bitcell Logic	XNOR	XNOR	XNOR	
MAC Operation	Charge-Domain (Charge Redistribution)	Charge-Domain (Charge Sharing)	Voltage-Mode (Pull-Up & Pull-Down DRV)	
Key Features	(+) High Throughput, Less Variation (-) Area Overhead (MOSCAP)	(+) In-Situ XNOR Operation (-) Multiple Cells for Logic Operation	(+) Wide Dynamic Range (-) Area Overhead (12T)	

Fig 3.1 Comparison of SRAM-Based XOR/XNOR IMC cells

3.1 6T XNOR-SRAM with Split-WL

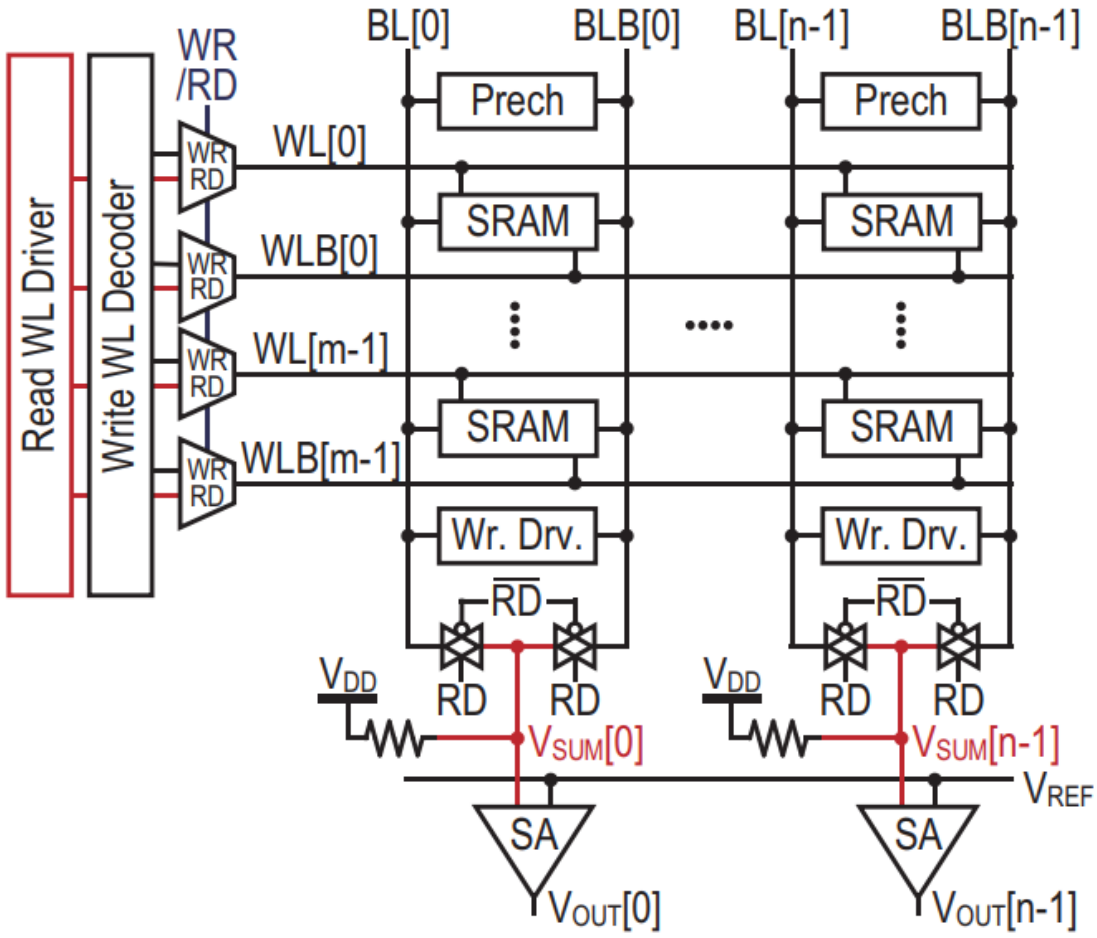


Fig. 3.2 IMC organization in [KKK+19]

Figure 3.2 presents the structure of the IMC design proposed in [KKK+19]. Each cell in the array is a 6T SRAM bitcell with split WL (Figure 3.3). The key idea to use a 6T SRAM cell for XNOR operation is to tie the bitline (BL) and the bitline-bar (BLB) during calculation mode (RD =1) using the transmission gates at the bottom of each column. To write in the cell the value of WR/RD has to be WR=1 (therefore RD=0), allowing the Write WL Driver to write the desired values into each cell. During calculation (RD=1,WR=0), multiple WLs are activated according to the input (data) values. The sense amplifier (SA) then compares the voltage level of the tied BL-BLB node (VSUM) to a fixed reference voltage (VREF) and produces a binary output (VOUT). By doing so, multiply and accumulate (VSUM) and activation (VOUT) for BNN are carried out simultaneously. Figure 3.4 shows the required input (WL/WLB) and weight (Q/QB) values to perform XNOR operation using this configuration. During read, a reduced voltage is used on WL (VDDL) to prevent possible data corruption, resulting in the two

NMOS transistors being less capable of driving Q-QB's voltage into BL-BLB but at the same time preventing the Q-QB voltage from flipping the value within the cell. Each SRAM cell pulls up (PU) VSUM voltage or pulls down (PD) it to GND, depending on Q (when WL=VDDL) or QB (when WLB=VDDL). Using a resistor connected between VDD and VSUM, the amount of PU and PD currents can be converted to VSUM. As a result, VSUM is determined by the number of PU rows and PD rows, which is directly proportional to multiply and accumulate (MAC) value of the m inputs and weights.

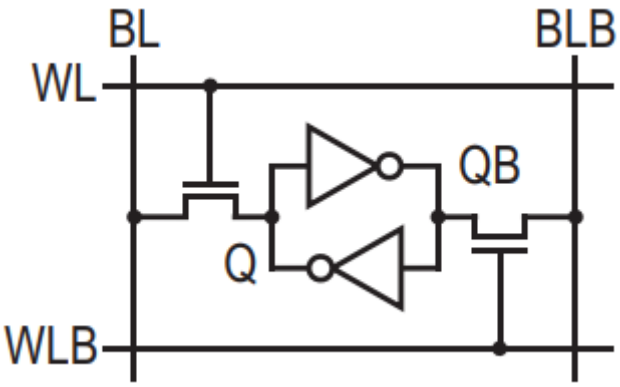


Fig 3.3 A 6T SRAM cell with split WL for XNOR operation proposed by [KKK+19]

The bitcell itself is an enhanced 6T SRAM cell with split WL for binary multiplication (XNOR operation) and is seen in Figure 3.3. The reason why two WLs are needed is because that way we can decide for each cell whether Q or QB will influence the value of VSUM allowing the four possible XNOR combinations to form for each cell. However this design has one major disadvantage which is that since a lot of cells share the same bitlines there is a chance that during the read procedure/calculation operation the value within one cell or more cells will be switched to the exact opposite because just as the cell influences the value of the BL , BL influences the value of the cell and if that influence crosses a certain threshold it leads to what is referred to as "read disturb".

Input	+1	+1	-1	-1
WL	V_{DDL}	V_{DDL}	GND	GND
WLB	GND	GND	V_{DDL}	V_{DDL}
Weight	+1	-1	+1	-1
Q	V_{DD}	GND	V_{DD}	GND
QB	GND	V_{DD}	GND	V_{DD}
V_{SUM}	↑ (PU)	↓ (PD)	↓ (PD)	↑ (PU)
Output	+1	-1	-1	+1

Fig. 3.4 Input/weight configurations for IMC XNOR

3.2 Dual-Split 6T SRAM for IMC

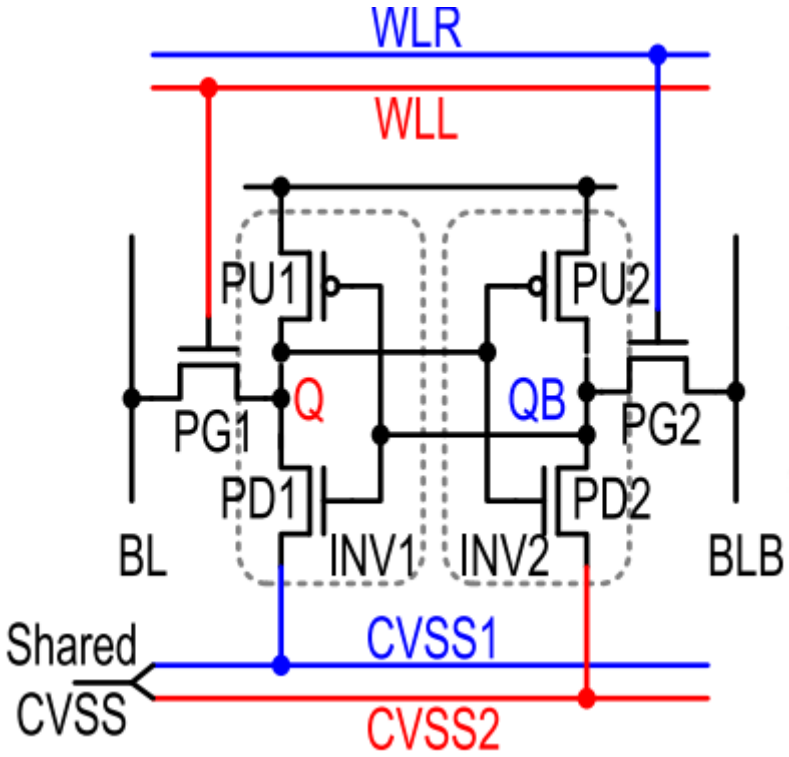


Fig. 3.5 A double-split controlled 6T SRAM cell proposed by [SKC+19]

Figure 3.5 presents a cell schematic of a proposed dual-split-control (DSC) 6T SRAM cell by [SKC+19] that supports the XNOR neural network (XNORNN) and the

modified binary neural network (MBNN). The cell uses split wordlines (SWL: WLL and WLR) and split cell-VSS (CVSS) lines (SCVSS: CVSS1 and CVSS2). Both techniques are used to implement read/write assist schemes in order to minimize area and power consumption. For example, the SCVSS is used to supply different voltage levels at the terminals of the NMOS transistors in the cross-coupled pair of the cell. By doing so the voltage difference between VDD and VSS becomes much smaller and from Ohm's law we know that less ΔV, means less current leaking through (for the same resistance). In this approach we can't use the same varying VSS voltage across both inverters (INV1 and INV2) because if VDD=1V and VSS=0.8V, then when Q=1V the leakage current is reduced, but at the same time as QB should be at 0V the corresponding VSS should be at Gnd. For normal write and read operations, WLL and WLR are shorted, so that these operations of the DSC6T cell are the same as those of conventional 6T SRAM cells.

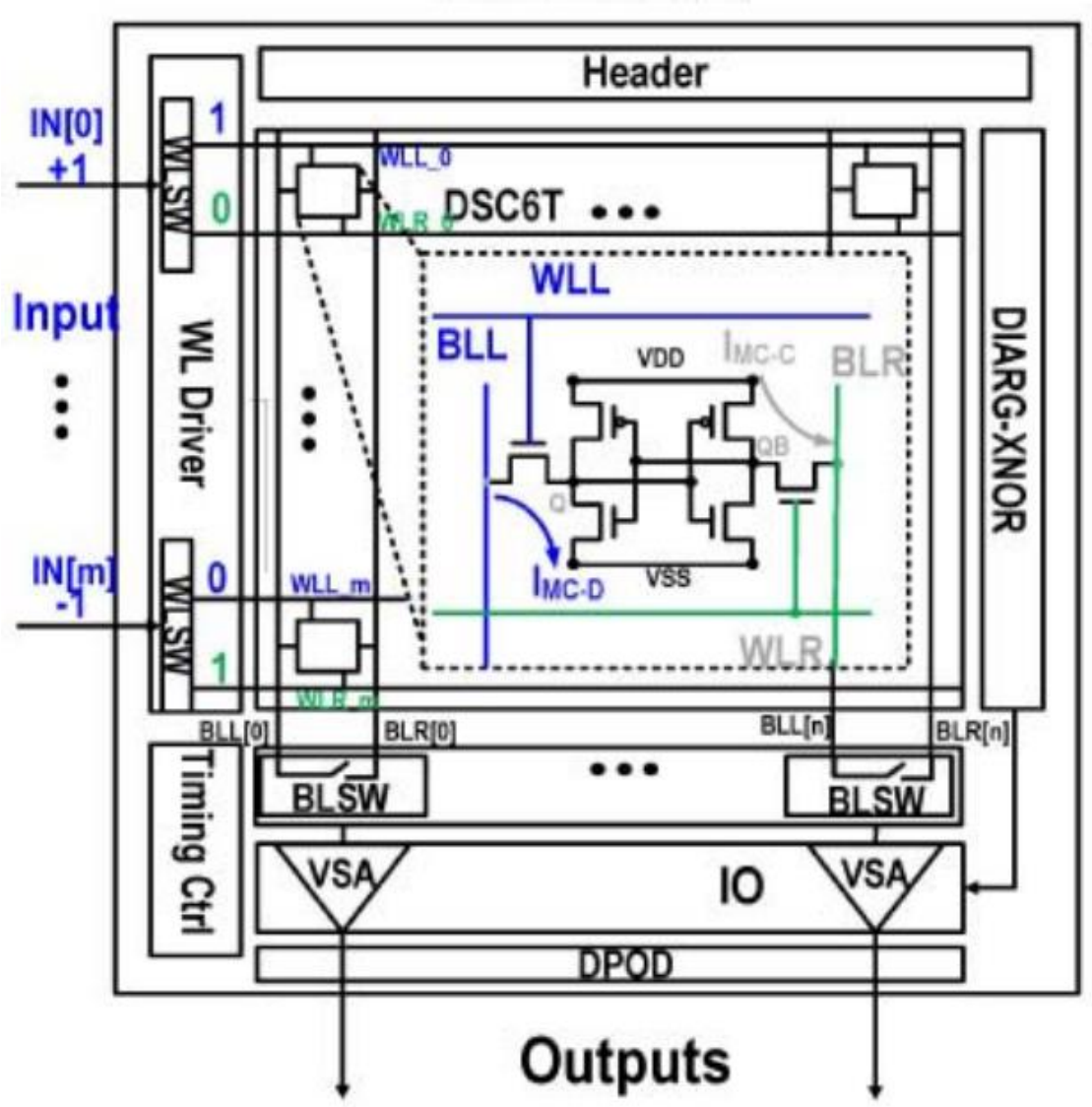


Fig. 3.6 Macro structure of XNORNN SRAM-IMC mode

Figure 3.6 illustrates the architecture of the XNOR neural network (XNORNN) SRAM-IMC macro proposed in [SKC+19]. This device consists of a dual-split-control 6T (DSC6T) bit-cell array and peripheral circuits for operation in two modes: SRAM and XNOR neural network (XNORNN). SRAM mode is activated to store the trained weights (write operation), whereas XNORNN mode is activated for XNOR-based IMC operations. In SRAM mode, only one row is activated for read and write operations.

The SRAM cell array can be accessed with both WLL and WLR on, which is like the read and write operations of a conventional SRAM. In such situations, the trained weights are stored in the dual split-control 6T (DSC6T) cell array via a write operation in SRAM mode under nominal-VDD. In XNORNN mode, multiple rows are activated at the same time, and in each row input data (IN) is pre-encoded on the two wordlines (WLL and WLR). The weights (W) are stored in consecutive DSC6T memory cells in the same column. When WLL or WLR is activated, the read current ($I_{Memory\ cell}$) of each activated memory-cell represents its input-weight-product (IWP = IN × W). All $I_{Memory\ cell}$ values of the activated memory cells are then summed at the bitlines (BLL and BLR are shorted at BLSW at the bottom of Figure 3.6) of the same column. When using a voltage-divider type sensing scheme through a PMOS header, BL voltage (VBL−XNOR) is the summation of all input-weight-products (IWP[m − 1 : 0]). Summing the read current of the storage cells that are activated in the same column yields the computation result. The updated 6T SRAM bit cells still have read interference because of the shared BL for read and write operations.

3.3 IMC supporting Multi-Bit Input, Weight and Output

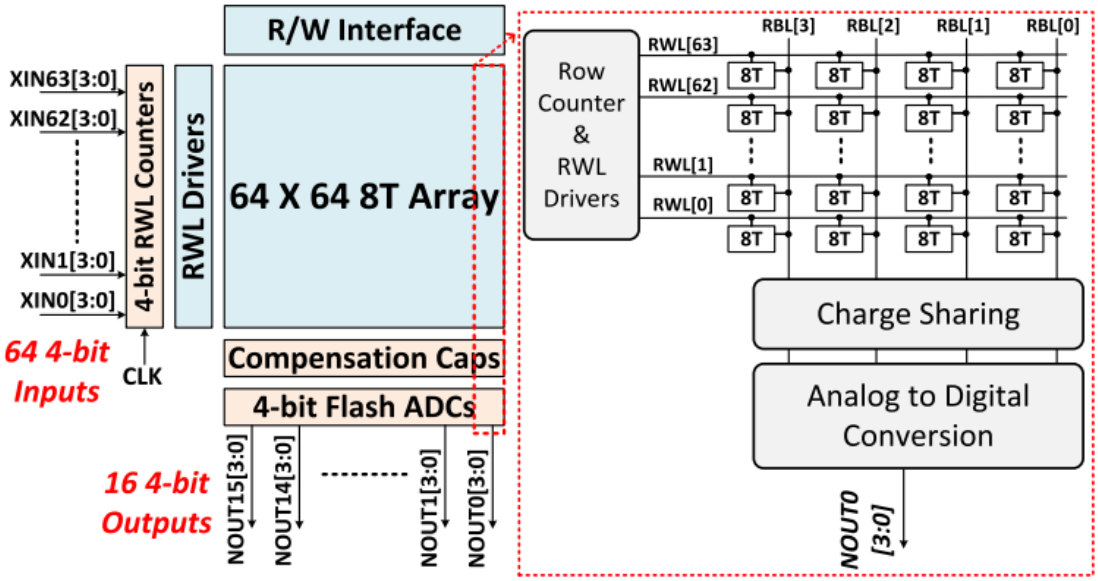


Fig 3.7 IMC macro utilizing bitline discharge proposed by [SEN+21]

In an attempt to create an IMC macro with 4-bit input and 4-bit weight MAC operation utilizing bitline discharge, [SEN+21] proposed an architecture based on foundry 8T SRAM (Figure 3.7). In order to implement 4bit inputs multiplied with 4bit weights to produce 4bit outputs sub-array of 4x64 8T cells is considered in a 64x64 8T array. Every cell in the same column shares the same RBL while every cell in the same row shares the same RWL.

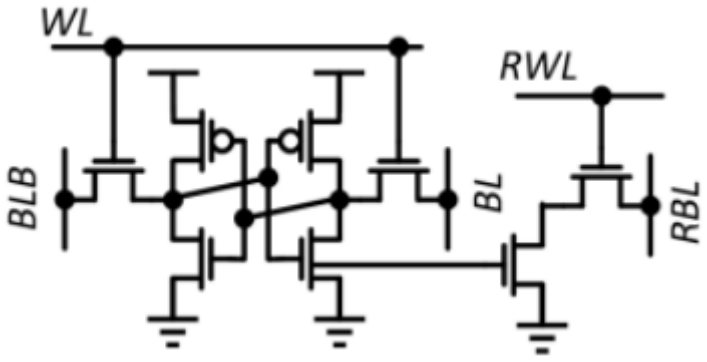


Fig 3.8 8T SRAM cell design proposed by [SEN+21]

Figure 3.8 shows the corresponding 8T SRAM cell. It consists of a standard 6T SRAM memory cell and two transistors that form the compute unit. Writing a value (weight)

into the cell follows the standard procedure for any 6T SRAM. Figure 3.9 describes the single cell operation where 1bit weight is stored in the cell and 1-bit input is applied as a WL pulse. Across the four possible cases, bitcell conducts current and discharges its Bitline (BL) only when both input and weight are "1". In other words, the voltage in the BL is equivalent to 1bit multiplication between input and weight. As mentioned above, this work supports 4-bit inputs, 4-bit weights, and 4-bit outputs. 4-bit inputs are realized by enabling the correct row's RWL giving the same input to all 4 bits across the row. This is implemented by placing row counters and necessary RWL drivers, as shown in Figure 3.7. 4-b weights are realized by sampling RBL voltage on binary-weighted capacitors and doing charge sharing to combine column-wise 1-b MAC results in a weighted fashion. Lastly, 4-b output is generated through a 4-b Flash ADC.

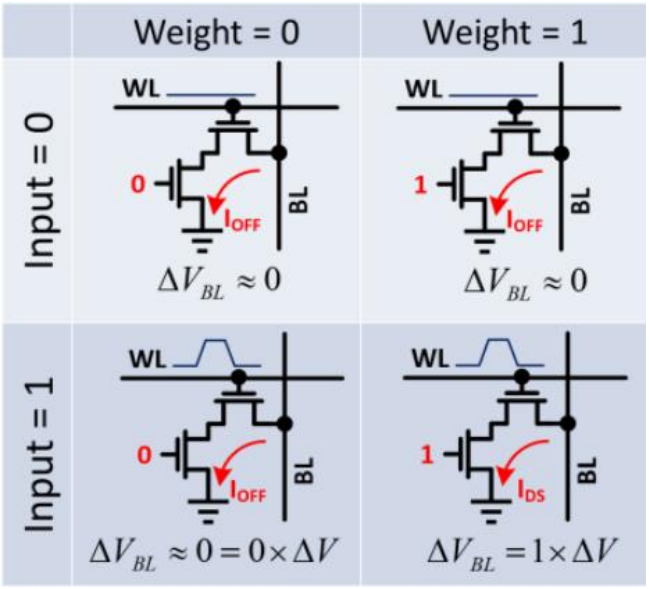


Fig 3.9 Bitwise multiplication concept

RBL voltages after the application of RWL pulses are proportional to bit-wise MAC of inputs and weights. To support 4-b weights, first, weights are stored in four neighboring bit-cells connected to the same RWL. After RBL voltage is developed, they are combined through binary-weighted capacitors such that the contribution of RBL voltages (MSB to LSB) is proportional to the location of the bits. Figure 3.10 shows two 4-b weights and two 4-b inputs (Xi[3:0]) applied from RWL[0] and RWL[1]. Shown in the same figure are two sets of binary-weighted capacitors; computation capacitors and compensation capacitors.

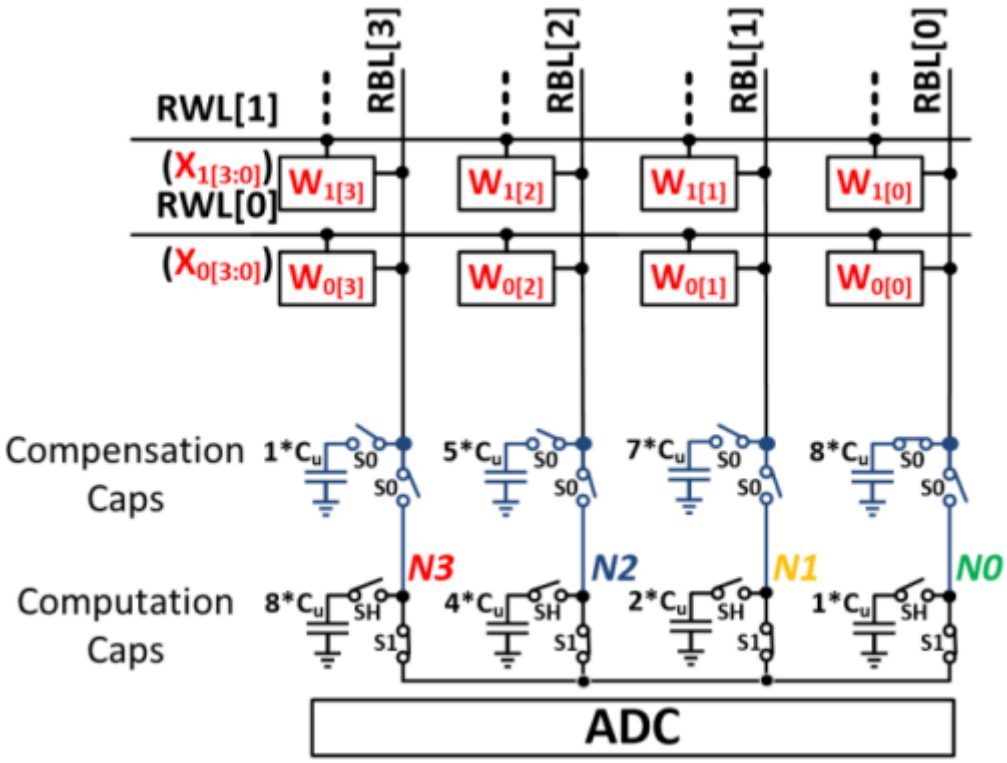


Fig. 3.10 An example array consisting of 2 inputs and 2 weights.

Figure 3.11 shows the IMC operation waveforms. At the beginning of the operation, all RBLs and capacitors are pre-charged to VDD. Next, during RBL Sampling phase, RWL pulses are applied and voltage is developed on RBLs. During this phase, RBL voltages are also sampled on binary-weighted computation capacitors. During the RBL voltage development and sampling, the load on RBL needs to be the same for all columns to avoid non-linearities so that's why compensation capacitors were added for each RBL. Up to this point the pass-gates S0 are closed (S0 = OFF) while SH and S1 pass-gates are open (S1= ON, SH= ON) essentially connecting each RBL only to the compensation capacitors. While S0 is still OFF, SH is enabled (S0 = OFF, SH = OFF, S1 =ON) charging the computation capacitor of each RBL. Once RWL pulses are applied, compensation capacitors and RBLs are disconnected from computations capacitors (S0 = ON, SH = OFF, S1 = OFF) and a charge sharing between binary-weighted computation capacitors is performed. At this point, charge shared voltage is proportional to the 4 bit × 4 bit MAC of inputs and weights. Lastly, an ADC is enabled and analog voltage is converted back to a digital representation.

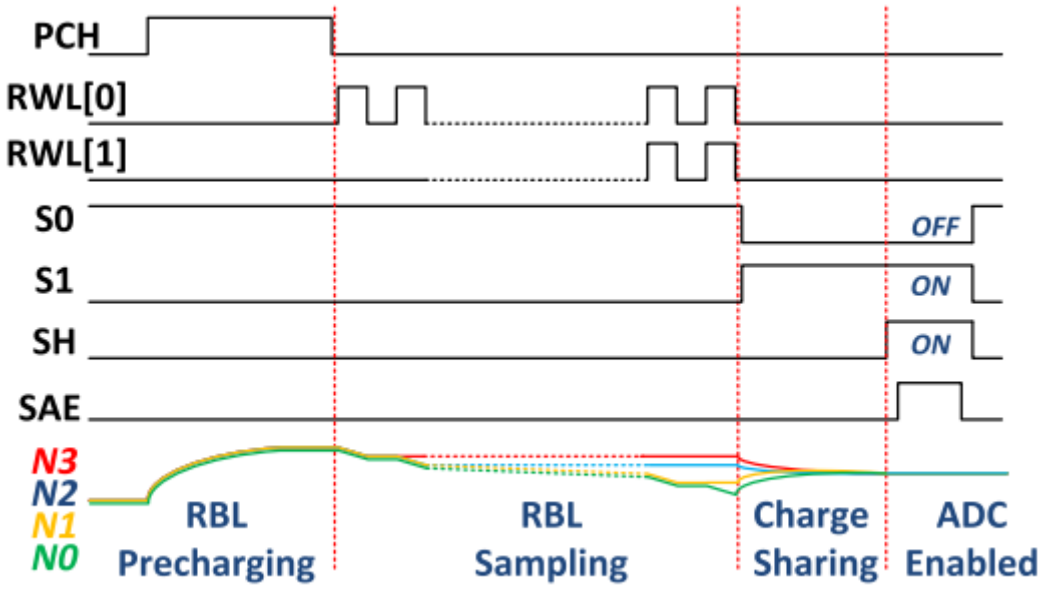


Fig 3.11 Timing diagram during different phases of the IMC operation

3.4 Charge-Domain SRAM for IMC

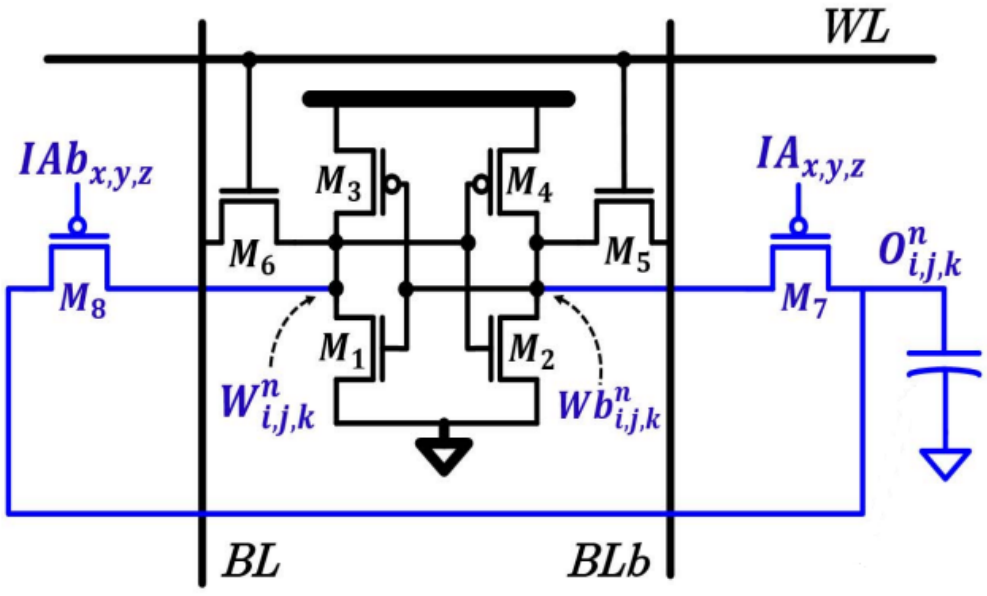


Fig. 3.12 8T1C SRAM bitcell whose multiplicative output is stored in a capacitor as proposed by [VRN+19]

Based on 6T SRAM, in [VRN+19] the 8T1C SRAM bit cell is proposed (see Figure 3.12) by adding two pmos and a capacitor. The structure of a memory array (in that

paper it is referred to as 'Neuron tile') is shown in Figure 3.13. Each neuron tile includes 64x64 neuron patches, each patch contains 3 x 3 Multiplying Bit Cells (M-BC) in order to process 3 x 3 binary Input Activations (IAs). 64 neuron patches in one column form a neuron filter. Within each neuron patch, every M-BC multiplies the corresponding 1-bit IA (input activation) with the 1-bit weight (value stored within the M-BC) and then stores the result into one local capacitor. Figure 3.12 depicts the M-BC circuit, where GND and VDD, respectively, stand for the binary values of '-1' and '+1'. Writing a value (weight) into the cell is achieved using a standard 6T SRAM. XNOR is implemented by adding two additional PMOS transistors that are driven by the IA signal. Finally, the capacitor O is responsible for storing the multiplication result.

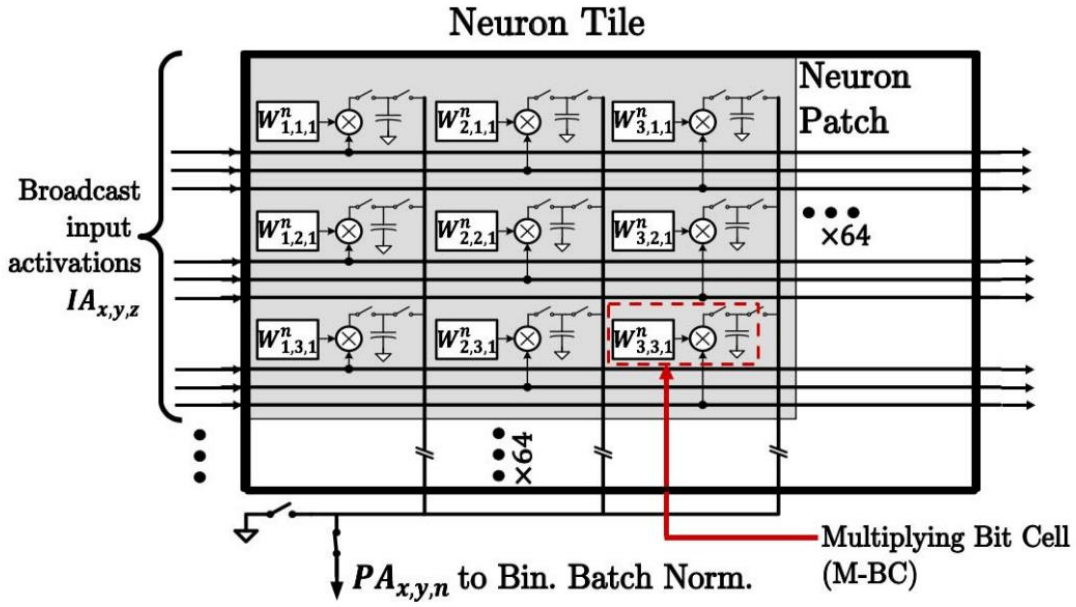


Fig. 3.13 Block diagram of a neuron tile. Each neuron tile includes 64x64 neuron patches, each patch contains 3 x 3 Multiplying Bit Cells (M-BC)

Calculation in this design is carried out in three steps. First is the reset phase during which all capacitors are unconditionally discharged to GND by enabling the TSHORT and PRE signals as seen in Figure 3.14. This is done while the two PMOS devices are deactivated therefore decoupling the bit-cell storage nodes. During the next phase the binary multiplication is calculated for each cell. The IA signals are driven complementarily, activating only one PMOS in each cell/M-BC and causing the capacitor to charge up to VDD or not, depending on the IA and the stored weight value on the cell, thus performing the XNOR operation. Finally, in the third step, known as the Accumulate phase the capacitors in one column sharing a common pre-activation value (PA) are shorted together to generate the analog value which is the final MAC result. The

multiplicative outcome is stored in a MIM capacitor that is placed above the bit cell, thus taking no additional area. Although this macro has a large dynamic range, it is also impacted by computational mistakes, brought on by charge injection from switching circuits, and insufficient throughput.

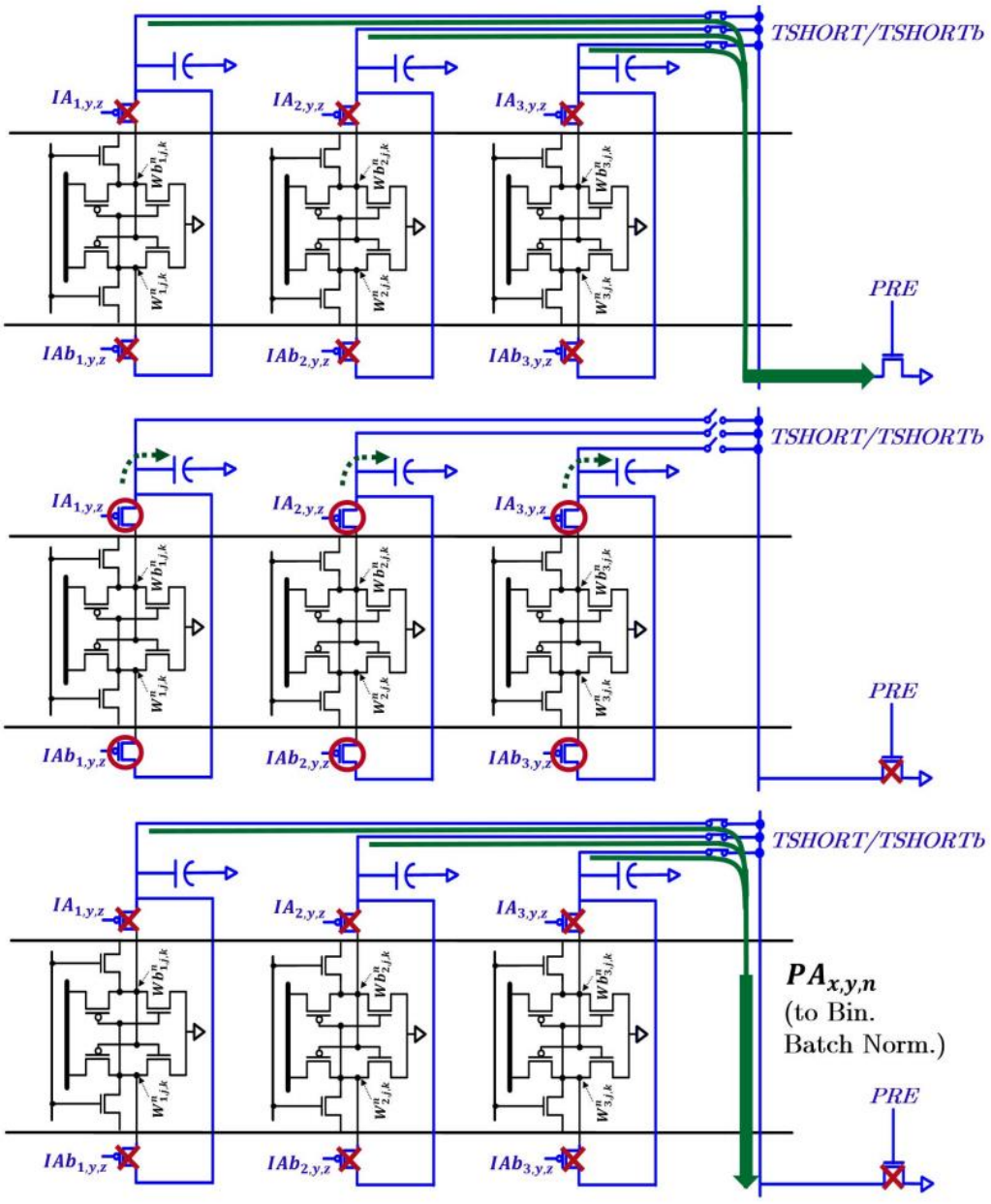


Fig. 3.14 XNOR computation by M-BCs, involving three signaling phases.

3.5 Capacitive Coupling Computing SRAM (C3SRAM)

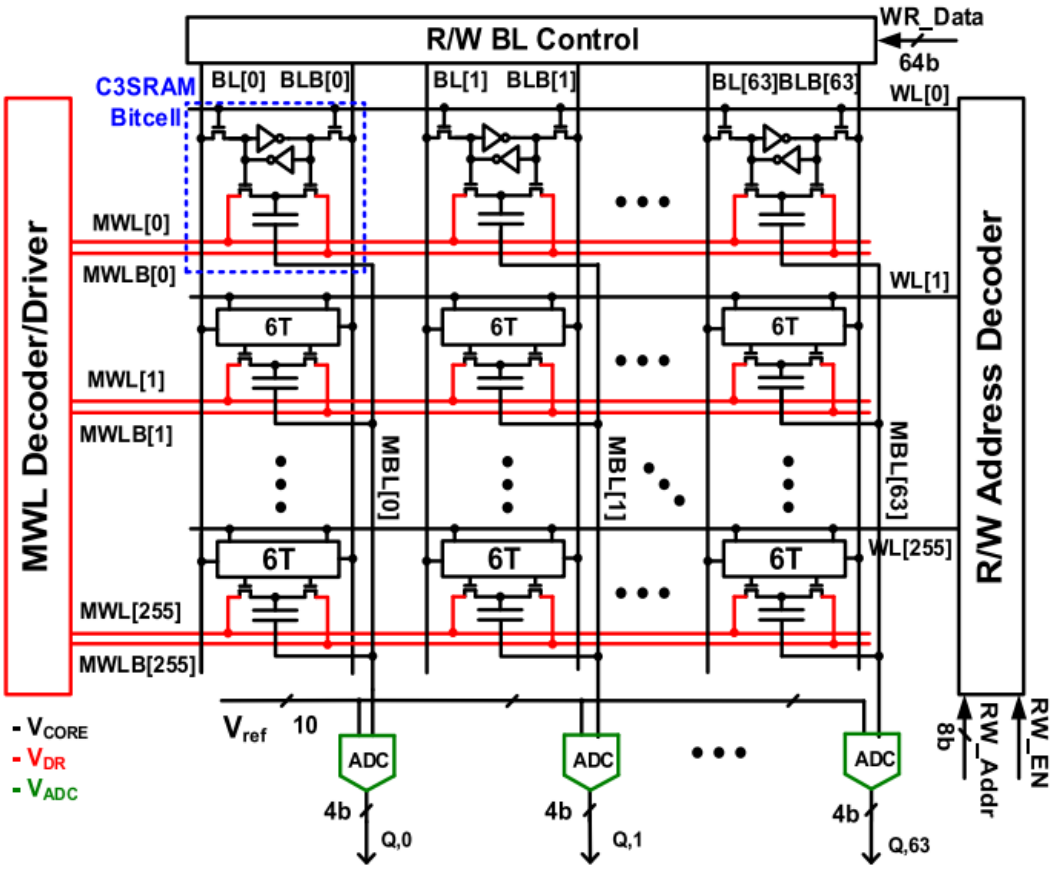


Fig. 3.15 Architecture of C3SRAM IMC macro.

Another 8T1C SRAM bit cell was proposed by [JYS+20]. The C3SRAM bitcell layout of the proposed design is based on capacitive-coupling computing (C3) hence named C3SRAM. C3SRAM performs a fully parallel vector-matrix multiplication of 256 x 64 binary weights (value stored within the cell). The macro (see Figure 3.15) consists of a 256 x 64 memory cell array, SRAM peripherals for read/write operations, input activation decoder/driver, and per-column flash analog-to-digital converters (ADCs).

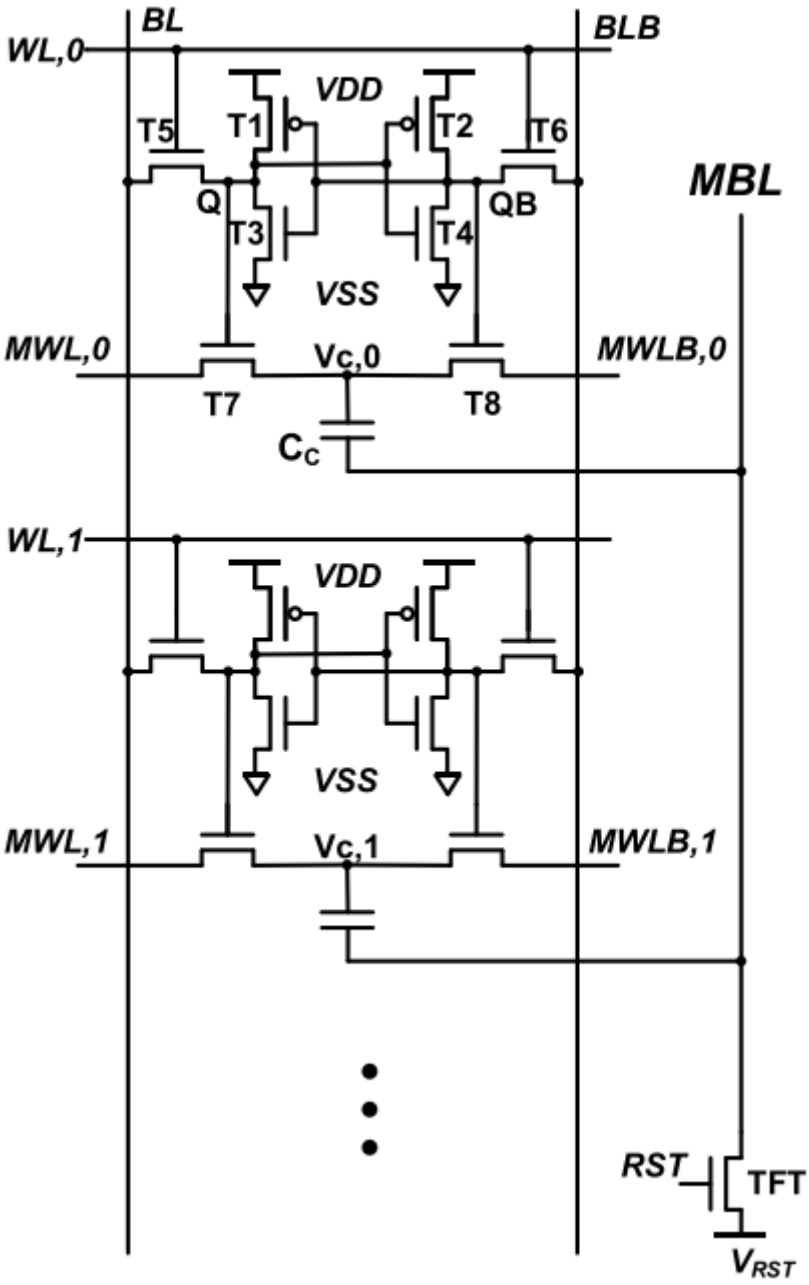


Fig. 3.16 Two C3SRAM bitcells sharing one common MBL, where the capacitor is in between two gate transistors complementarily enabled, proposed by [JYS+20]

Figure 3.16 shows a circuit diagram with two C3SRAM bitcells in a column. The capacitor is implemented as MOSCAP for high capacitive density. To perform binary dot product, the capacitor Cc is charged/discharged by MAC wordlines (MWL/MWLB), which play the role of datalines, via the pass transistors, which are gated by the stored weight and its complement. The binary MAC operation of their C3SRAM is carried out in two steps; each completed in half cycle duration. In the first step each column's MBL is precharged to VDD/2. In the same step, MWL and MWLB of each row are likewise reset

to VDD/2 such that there is no voltage potential on bitcell capacitors. In the second step, the footer transistor TFT is turned off. The 256 input activations are applied to 256 MWLs/MWLBs in parallel and therefore depending on the weight stored in the cell, the corresponding NMOS transistor is enabled, allowing current to flow through it and charge or discharge the capacitor. All capacitors are attached to the same MBL driving either up or down the MBL voltage from the previously set VDD/2 level. That value is then digitized with the use of a flash ADC per column. This approach offers a better throughput than the charge-sharing-based IMC macro, but because it uses mos capacitors (MOSCAP), this architecture takes up more silicon area.

3.6 XCEL-RAM

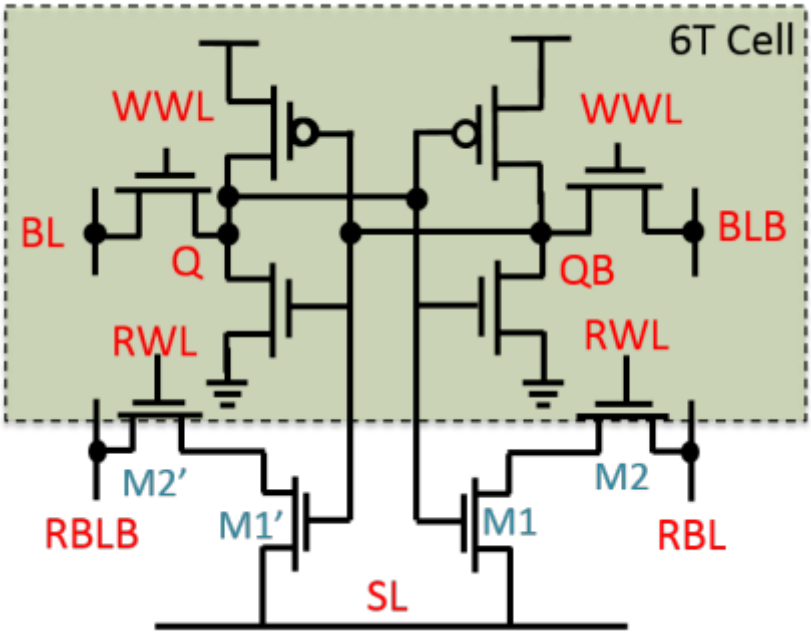


Fig. 3.17 A 10T1C SRAM bitcell based on the charge-sharing mode IMC proposed in [AJR+19]

Figure 3.17 shows a schematic of a 10T1C SRAM cell based on charge-sharing mode, proposed by [AJR+19], containing a basic 6T-cell as the storage unit, along with 4 transistors forming the differential read ports in pairs of two. Writing into a cell is similar to the 6T write operation. For reading, both RBL and RBLB are precharged to VDD, SL is connected to the ground, and RWL is enabled. If the cell stores a value of '1' (Q=VDD , QB= GND) then RBL discharges to GND and RBLB holds its charge and vice versa.A convolution operation is simplified to a bitwise XNOR, followed by a popcount/accumulation of the XNORed output in binary neural networks. They exploit the inherent SRAM structure, utilizing the internal parasitic capacitances to

perform the XNOR and accumulation operation of two vectors stored within the memory array. Although their approach to binary convolution is digital, the voltage used to evaluate the accumulation count is analog. They also added a dual read-wordline (Dual-RWL) scheme along with a dual-stage ADC to minimize the errors in the popcount output. When using a dual RWL technique each row in the memory array consists of two read-wordlines RWL1 and RWL2 (see Figure 3.18). Half of the cells along the row are connected to RWL1, while the other half are connected to RWL2. At a time, only one of RWL1, RWL2 are enabled, to ensure that only half of the cells participate in charge sharing, at a time, thereby reducing the number of voltage states on the SL to be sensed.

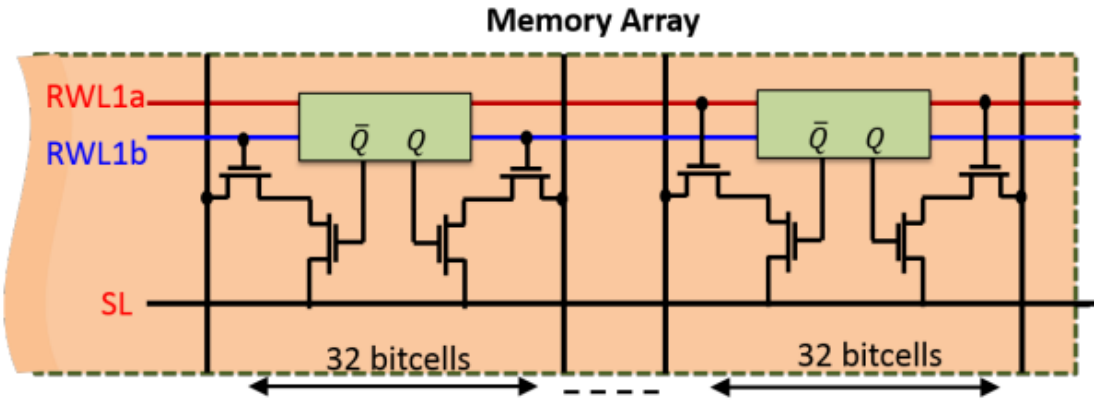


Fig. 3.18 Dual RWL technique for each row.

To compute the binary convolution within the memory array itself three steps are required. In the first step, a pseudo-read operation is performed on a row storing the binary vector inputs (A1). It is called pseudo-read due to the fact that we follow the standard procedure as when we want to read the value of the cell but this time we are not really interested in the value itself, we just want to multiply it with the value stored in another cell. During this stage, the cell has a value stored in it and WWL is disabled so the value can't be overwritten. All RBLs/RBLBs are precharged to VDD and then when the RWL corresponding to the row storing A1 is enabled, the precharged RBLs and RBLBs discharge depending on the data values stored in the cell. For example, in Figure 3.19 we can see that initially RBL/RBLB are precharged to VDD. When RWL is enabled either Q or QB will be '1' enabling the corresponding NMOS transistor (M1 or M1') and discharging the corresponding RBL line. In our Q=1 so through the M1 transistor RBL gets discharged while RBLB remains charged. That way after the pseudo-read operation the RBLs/RBLBs store the information of A1 as their respective voltages. In the second step, the RWL corresponding to the row storing K1 is enabled, and the one storing A1 is disabled (that's why the cells are greyed out in Figure 3.20). Now in our example the

RBLB was charged (1) and RBL was discharged (0) after the pseudo read operation. Depending on what value is stored in K1 (Q=1 or Q=0) the corresponding M1 or M1' transistor is enabled allowing charge flows into the SL or out of the SL. In Figure 3.20 we can see both the charge (top left cell) and the discharge (top right cell). That way we accomplished the calculation of A1 XNOR K1 and the result of this XNOR is seen in the truth table in Figure 3.21.

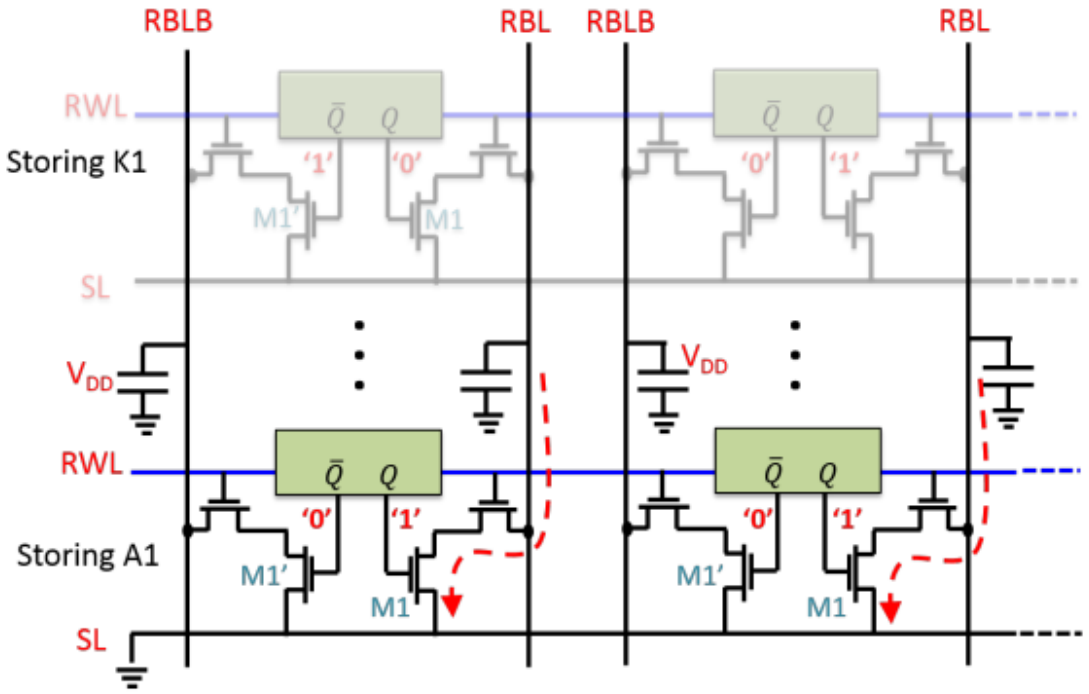


Fig 3.19 First step of calculation (Pseudo-Read)

Since the SL is shared by all the cells along the row, these pull-ups and pull-downs by each cell are accumulated upon each SL. Pull-ups count as 1 and pull-downs as 0, hence the popcount. Evaluation of the output is done by counting the 1s which in analog terms results in higher voltage for each 1 in the common SL. The 10T SRAM bit unit has a high calculation latency since it must perform the XNOR calculation in stages and using many SRAM cells.

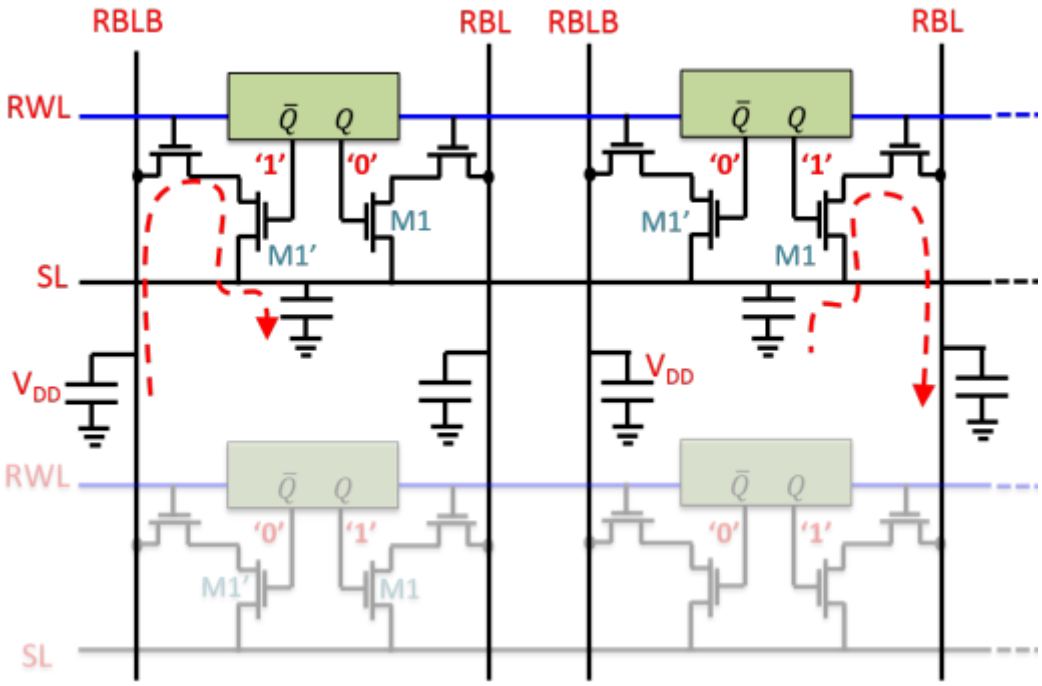


Fig 3.20 Second step of calculation (XNOR on SL)

RBL	Q	SL
0	0	↑
0	1	↓
1	0	↓
1	1	↑

A1	K1	Out
0	0	1
0	1	0
1	0	0
1	1	1

XNOR on SL

Fig. 3.21 XNOR truth table

3.7 XNOR-SRAM for binary/ternary Deep Neural Networks

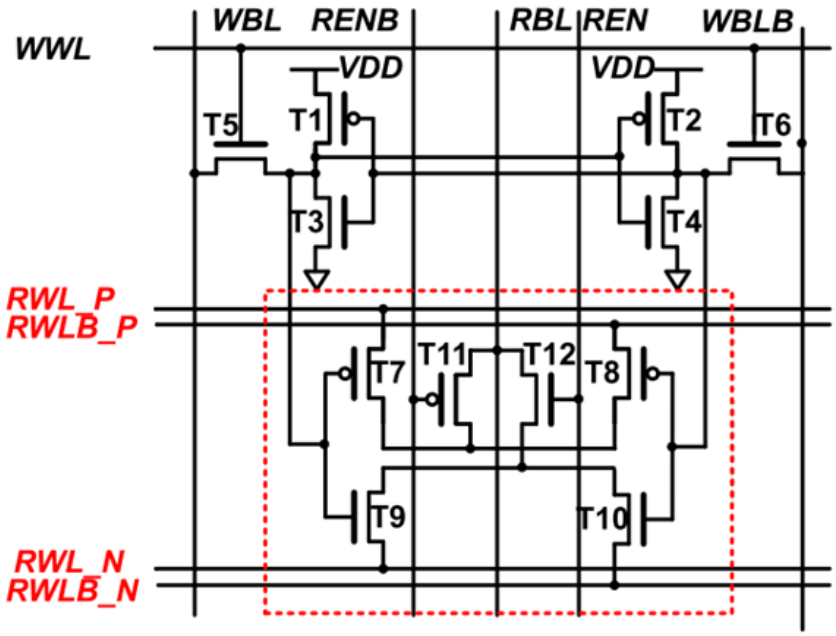


Fig 3.22 12T XNOR-SRAM bitcell for binary/ternary activations using voltage-mode(PU and PD) as proposed by [YS20]

Finally, [YS20] suggested an IMC macro based on the voltage mode SRAM (XNOR-SRAM), which performed binary and ternary XNOR-and-accumulate (XAC) computations using a 12T SRAM bit cell structure. Their proposed bitcell design as seen in Figure 3.22 consists of 12 transistors; 6 forming a regular 6T memory cell and another 6 used for calculating the XNOR result for each cell. For binary activations the bitcell produces the XNOR output of +1 with one strong PU by PMOS and one weak PU by NMOS. It produces the XNOR output of "−1" with one strong PD by NMOS and one weak PU by PMOS. The 256 bitcells in a column contribute such XNOR-output-controlled PU and PD circuits and essentially form a resistive voltage divider from the supply voltage to the ground, where RBL is the output. To support ternary activations, they added the activation of value '0' on top of -1 and +1. This 0 value is produced by either one weak PU through an NMOS and one weak PD through a PMOS or one strong PU through a PMOS and a strong PD through an NMOS. Which one is actually implemented depends on whether the row is even or odd, but it essentially contributes to the same result since the strong PU and PD or weak PU and PD cancel out each other. Deviation from the assumption that the number of even-row zeros and odd-row zeros are not equal, would cause VRBL

deviation, hence why they are different. The XNOR result of each cell in the same column, regardless of whether the calculation was binary or ternary, is shorted into one common RBL as an analog voltage and the ratio of PU to PD drivers determines the output level as seen in Figure 3.23 that depicts the macro architecture of this proposition. It requires a rather big area overhead, despite the fact that it can achieve a wide dynamic range and minimize read interference problems.

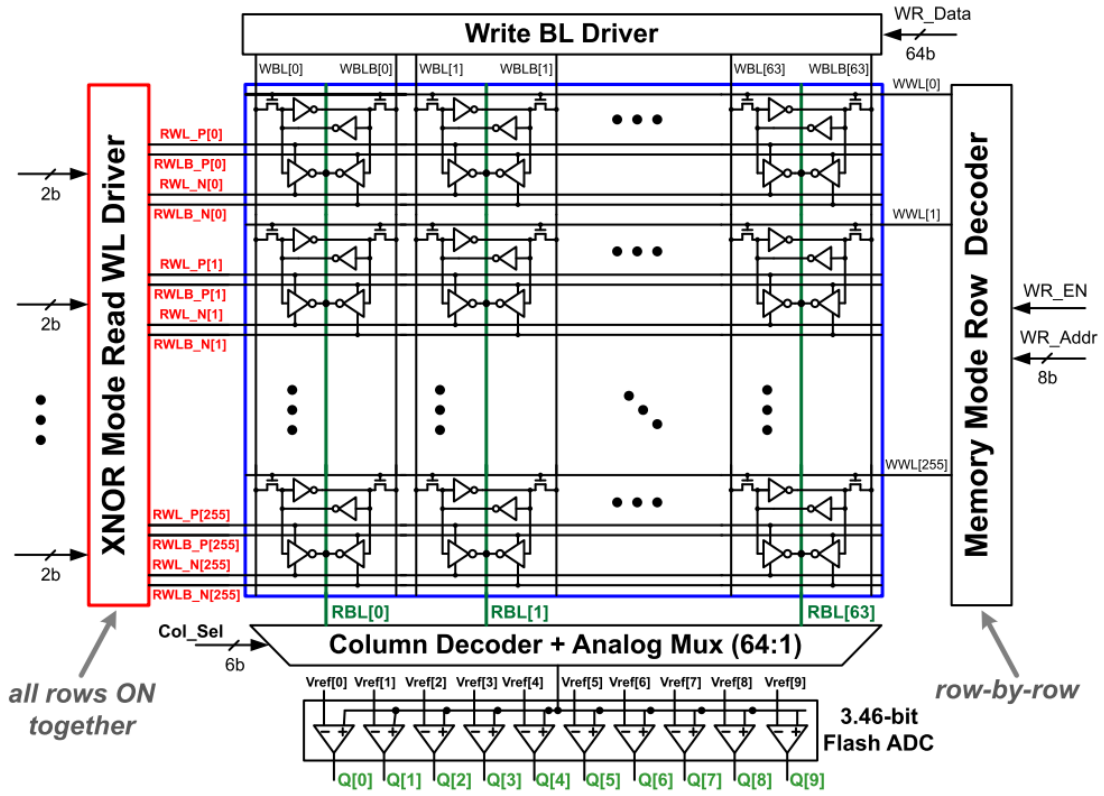


Fig. 3.23 XNOR – SRAM macro architecture as proposed by [YS20]

Chapter 4. XNOR-SRAM Macro Design and Optimization

4.1 SRAM-Based IMC

Large scale recognition tasks have seen previously unheard-of improvements in accuracy thanks to Deep Neural Networks (DNNs) and convolutional neural networks (CNNs). Nevertheless, DNN hardware's acceleration and energy efficiency are constrained by memory access and arithmetic complexity. To address this, weights and neuron activations are binarized to -1 or +1 in current algorithms. This way, the multiplication of a weight and an activation becomes a XNOR operation, and the accumulation of the XNOR operations becomes the bitcount of those XNOR outcomes.

In this chapter, we propose a modified 6T SRAM bitcell design that enables IMC operations, hence a further elaboration on SRAM-Based IMC is required. One approach to SRAM-Based IMC is the one proposed by Shihui Yin, Jae-Sun Seo [YS20] where a modified 6T SRAM memory cell is used for ternary/binary operations for deep neural networks. In this publication, the proposed bitcell consists of 12 transistors [see Figure 3.22], T1-T6 form a 6T cell, T7-T10 form complimentary Pull Up (PU) or Pull Down (PD) units for the computing mode (and memory mode read), and T11 and T12 as access/activation transistors when the corresponding column (256 bits) is selected.

In computing mode, depending on the value stored in the memory cell, either T7(pmos) or T9(nmos) will be enabled (be on) and due to the complementary nature of the 6T memory cell the exact opposite transistor (T10(nmos) or T8(pmos) respectively) will be also enabled. If the read enable signal is logic-1 (therefore read enable bar is logic-0) T11 and T12 both are enabled so that the cell affects the value of the Read Bit Line (RBL).

Given that the value stored in the memory cell is the binary weight and the digital values provided via the RWL_P, RWLB_P, RWL_N, RWLB_N inputs are the data, the digital multiplication (XNOR) is performed in the cell between weights and data (weight*data). The result of the multiplications in the cells of a column are accumulated on the RBL as we will see next.

4.2 Multiply and Accumulate Operations (MAC)

The operation we explained up to this point is a simple multiplication where the value stored in the cell is the weight and depending on the input values, we get a different output. However, in most DNNs we must calculate the sum of the products provided by each 12T cell (a1*w1 +a2*w2 +…+an*wn). In order to do this, the authors proposed to accumulate every result produced by each cell in a column into one single analog value. This is achieved by connecting every 12T cell of each column to a common RBL as shown in Figure 4.1. Thus, a Multiply and Accumulate (MAC) operation is performed and the result appears as an analog voltage value in the pertinent RBL due to the voltage divider that is formed among the cells of the RBL.

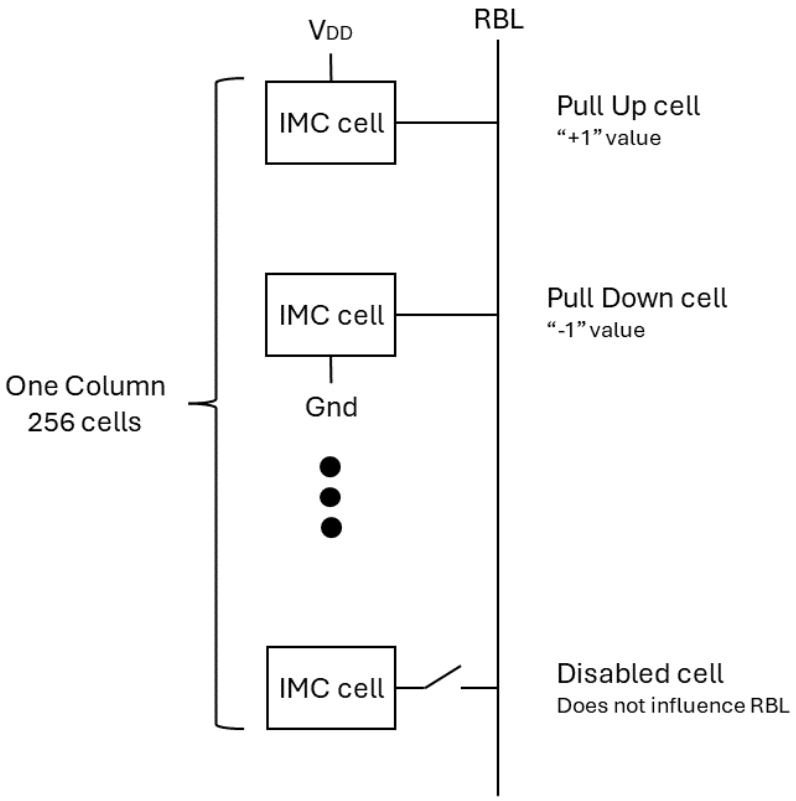


Fig 4.1 Connection of a column's cells to a common RBL

With both weights and activations (data) binarized to +1 or -1 in BNNs, the high-precision multiply-and-accumulate (MAC) operations can be replaced by XNOR and bit-counting operations. In one column (256 bits) with one common RBL the value range of RBL are the continuous values in [GND, VDD] where RBL = GND when XAC = -256 and RBL = VDD when XAC=+256. That means that if exactly 128 cells pull RBL's voltage down to GND and another 128 cells pull it up to VDD, we should expect to see a value of VDD/2 in the RBL.

4.3 The proposed 11T SRAM cell for IMC XAC operations

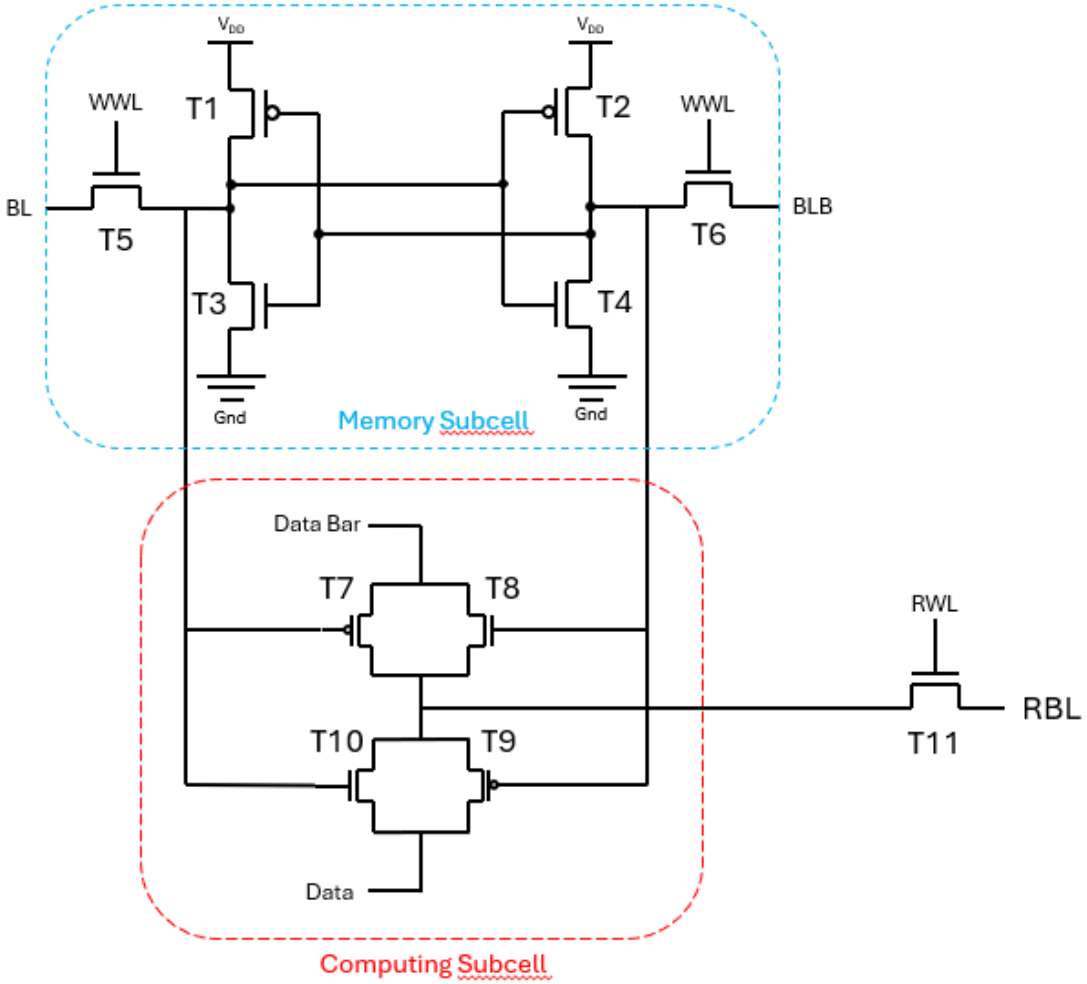


Fig 4.2 Proposed 11T SRAM cell

Figure 4.2 shows the proposed bitcell for XNOR-SRAM. T1-T6 form a regular 6T cell, T7-T10 form two full pass-gates which are complementarily enabled according to

the value stored in the cell and finally T11 is an NMOS transistor that acts as an NMOS pass-gate between the cell and the Read Bit Line (RBL). In the memory subcell the two NMOS transistors T3 and T4 are slightly sized up. In the computing subcell, except T7 and T9 PMOS transistors, all other transistors in the bitcell use the minimum size. We sized up those two PMOS transistors to match their strength to their NMOS counterparts. In the XNOR mode T11 is turned on and T7-T10 perform the XNOR operation between the Data/Data_bar values (+1, -1 / VDD,GND) and the binary weight stored in the cell (+1,-1 / VDD,GND). The RBL voltage settles into a continuous value between GND and VDD which is then digitized by an analog-to-digital converter (ADC) to the XAC value of all the cells that contributed to the calculation.

4.4 Alternative 12T SRAM cell for IMC XAC operations

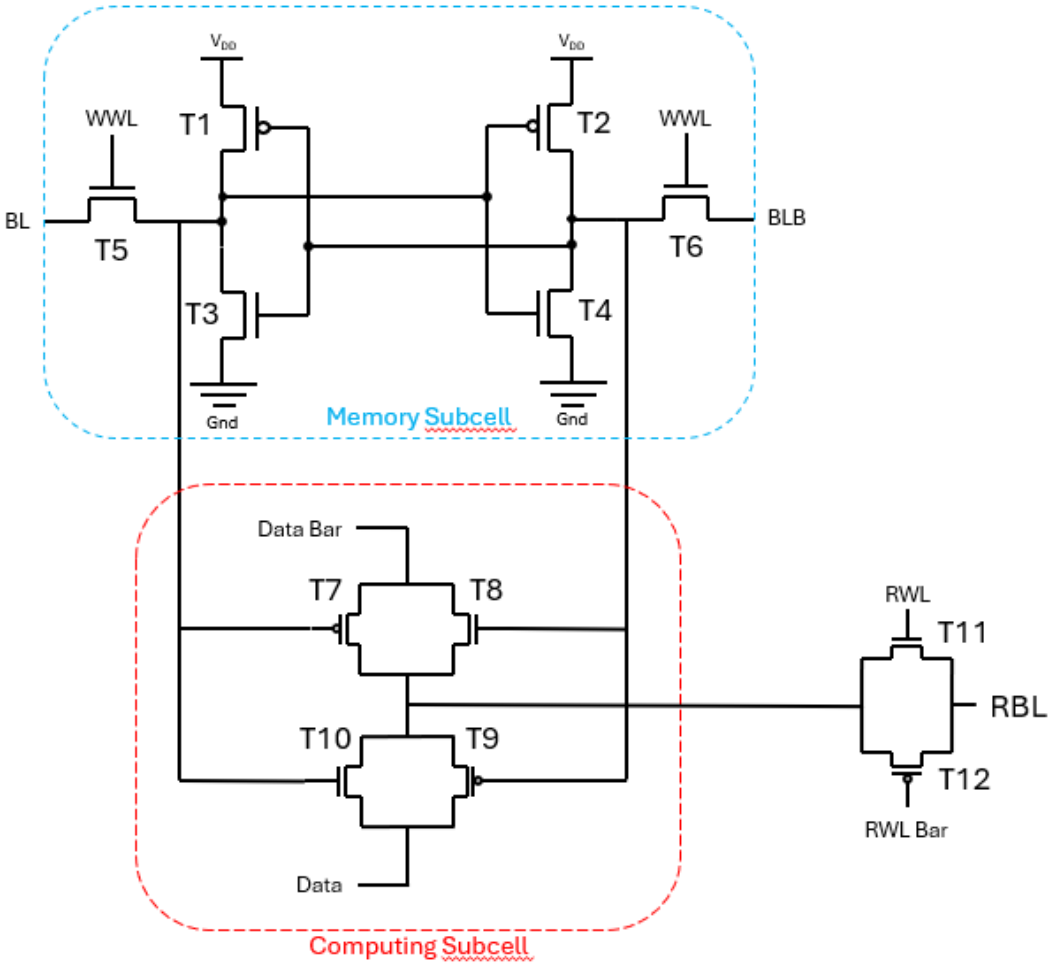


Fig 4.3 Proposed 12T SRAM cell.

Figure 4.3 shows an alternative SRAM cell with 12 transistors. The bitcell we previously suggested has one drawback whose gravity is not easily measured. In Figure 4.2, the current flow between the cell and the RBL is through an NMOS pass-gate. This does not affect in any way the XAC calculation if the result between the weight and the data line is -1 (GND) but it does impact it when the value is +1 (VDD) since we try to charge RBL to VDD through an NMOS transistor, i.e. through a higher resistance path.

This inefficiency can be mitigated by applying a higher-than-nominal voltage to the gate of that NMOS transistor (overdrive it). This higher-than-nominal voltage may be obtained by stepping down a higher voltage source that is provided in the entire processing unit. This approach is commonly used in DRAMs but assumes the presence of a higher voltage in the chip (e.g. chip pads). If such voltage is not present, then we have to produce it from a lower voltage such us V_{DD}, possibly using a charge pump, which will increase silicon area cost and power consumption. To address this problem, we considered an alternative bitcell design that exploits a full pass-gate by adding a PMOS transistor (T12) in parallel to T11. The gate of this extra PMOS transistor needs to be supplied with a signal RLW_Bar which in turn leads to an additional input line across the entire row of cells in the array. In this implementation our bitcell has one more PMOS per cell as well as a control line to drive the gate of the transistor, which is a reasonable increase in silicon area, but it has no need for any higher than VDD voltage source in the entire circuit leading to less design complexity and possibly power consumption.

4.5 XNOR-SRAM Array Organization

The designs under consideration are an extension of the 6T memory modified accordingly so that it enables IMC. The array architecture (Figure 4.4) consists of a 256-by-64 custom bitcell array, a write BitLine (BL) Driver, an XNOR mode WL driver, as well as a Memory mode Row Decoder. The XNOR-SRAM has two operation modes, a memory mode and an XNOR mode. Memory mode performs row-by-row digital read/write, just like a regular 6T SRAM cell, while in XNOR mode it executes in-memory XAC calculation asserting all rows concurrently.

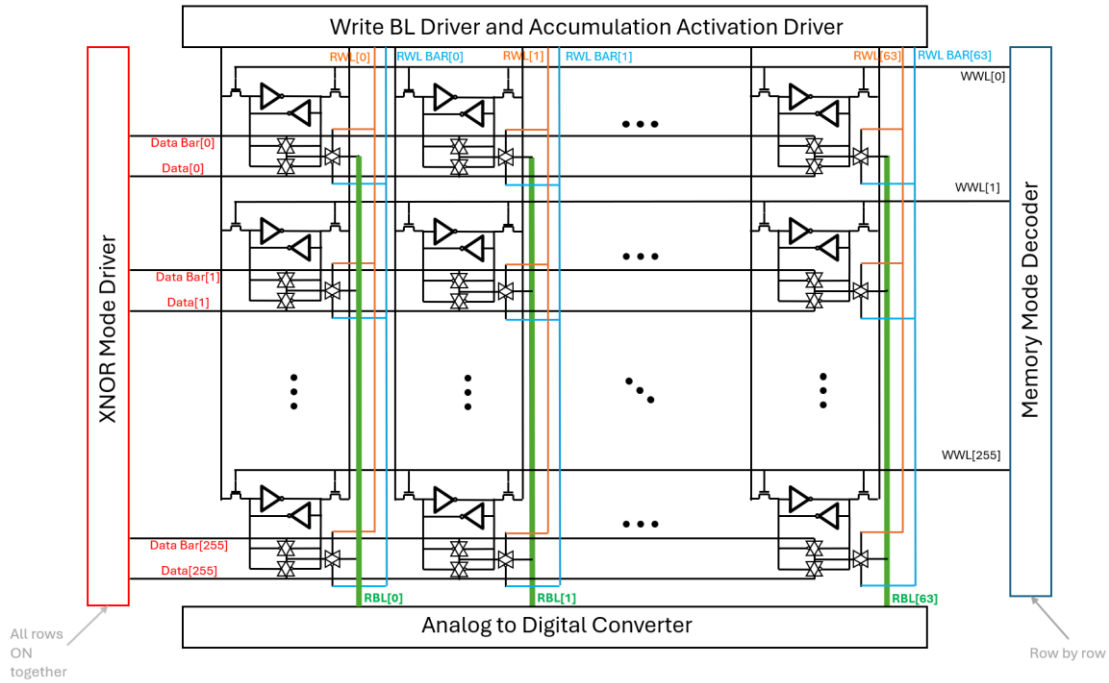


Fig. 4.4 The memory array organization under consideration.

4.6 Proposed IMC Operation and Analysis

Tables 4.1/4.2 shows the operations calculated by each cell in both binary values (+1/-1 or logic-1/logic-0) and voltage values. In the tables we use the logic-0/logic-1 format but moving on we will use the -1/+1 format since it is easier to comprehend when counting how many cells pull up or pull down the RBL's voltage. The cell produces a "+1" value if either the value stored in the cell (weight) is "-1" and the data value provided is "-1" or if the weight is "+1" and the data value is "+1" (since this makes the data_bar value equal to "-1"). The 256 bitcells in a column contribute either with a "-1" or a "+1" value on the RBL and essentially form a resistive voltage divider from the Supply Voltage (VDD) to the ground (GND) where RBL is the output. If the cells participating in the calculation are equally distributed in "-1" and "+1" values, meaning that there are the same number of cells pulling down the RBL towards GND as these that are pulling it up towards VDD, then the RBL voltage should be exactly at the average between those two values. (VDD+GND)/2 = VDD/2. This voltage should not vary regardless of whether 1 cell pulls down, 1 cell pulls up and the other 254 are turned off or if 128 pull down and 128 pull up as long as the PMOS transistors are properly sized up to match the strength of their NMOS counterparts. The RBL voltage should be a symmetric and monotonic function of the XAC value if the pull up (PU) and pull down (PD) resistances are equal; however considering that process variations may affect this

operation. In Figure 4.5 we can see how different weights and data affect the RBL's voltage.

Weight	Input	RBL	Weight	Input	RBL
0	0	1	0 V	0 V	1.2 V
0	1	0	0 V	1.2 V	0 V
1	0	0	1.2 V	0 V	0 V
1	1	1	1.2 V	1.2 V	1.2 V

Table 4.1 Truth table for 12T design

Weight	Input	RBL	Weight	Input	RBL
0	0	1	0 V	0 V	1.2 V
0	1	0	0 V	1.2 V	0 V
1	0	0	1.2 V	0 V	0 V
1	1	1	1.2 V	1.2 V	1.2 V

Table 4.2 Truth table for 11T design

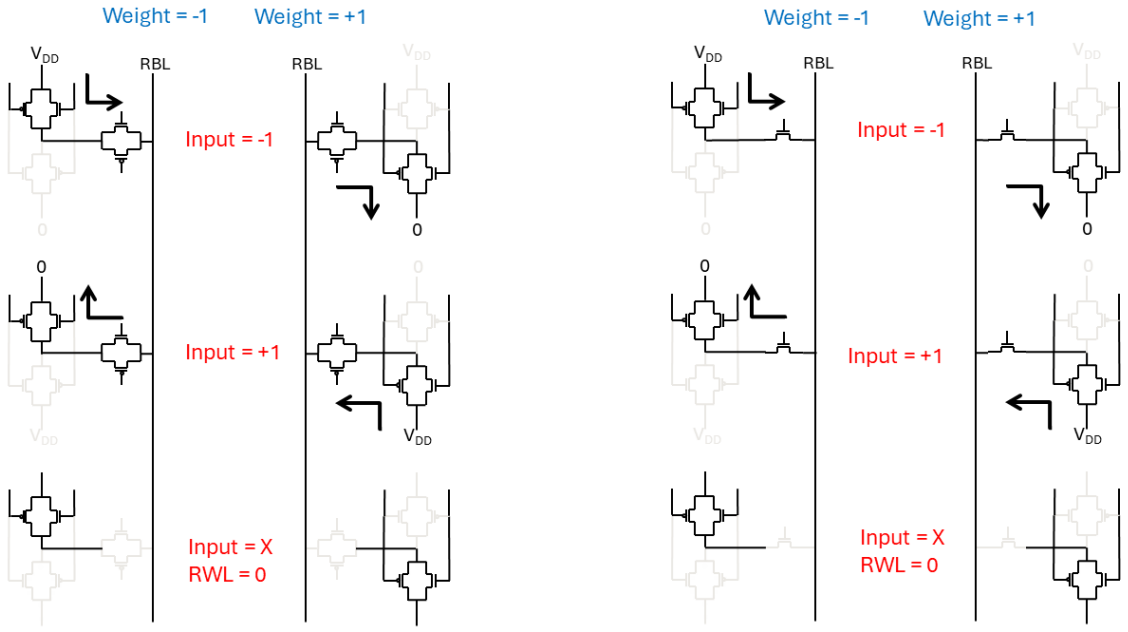
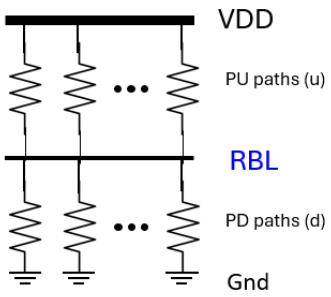


Fig. 4.5 Charge or discharge of RBL according to different weights and data in both proposed designs

The following is the first-order analysis of the relationship between the RBL voltage and the XAC value. The range of XAC is from -N to +N if there are **N** rows. Assume that **d** represents the number of PD cells and **u** represents the number of PU cells among N cells in a column. N can be represented as the sum of u and d, as Eq. 1 in Figure 4.6 demonstrates. The XAC result, which adds up all of the cells' XNOR outputs, is expressed as Eq. 2 since each PU and PD cell represents the bitwise XNOR output of "+1" and "−1," respectively.



$$N = u + d \qquad \textbf{Eq. 1}$$

$$\text{XAC} = u - d \qquad \textbf{Eq. 2}$$

$$VRBL = V_{DD} \times \frac{u}{u + d} \qquad \textbf{Eq. 3}$$

Fig. 4.6 Resistive voltage divider formed on the RBL

Moreover, using the equation for the resistive voltage divider, VRBL can be expressed as shown in Eq. 3. Keep in mind that as long as the same XAC value is produced in one column then the RBL's voltage should be the same regardless of how many cells were utilized to produce that result. For example, if 238 cells are turned off (RWL = 0 for those rows) 10 cells pull up and 8 cells pull down:

$$XAC1 = 10 - 8 = 2$$

This leads to an RBL voltage value of v1 so $V_{RBL1} = v1\,V$. If we had another column with 254 cells turned off, 2 cells pull up and 0 cells pull down:

$$XAC2 = 2 - 0 = 2$$

As long as XAC1 = XAC2 then $V_{RBL1} = V_{RBL2}$.

Chapter 5. Simulation Results

We designed the proposed XNOR-SRAM IMC architecture using the 90-nm CMOS technology of UMC and the Virtuoso platform of Cadence. The Low Leakage Low Threshold Voltage transistors of the technology were exploited. Those transistors are expected to work normally under 1.2V supply and they have a minimum width of 120nm. Simulations were carried out using Spectre. All measurements were calculated under the same time constraints for write cycles and computing (XNOR mode) cycles. According to the simulations, in Figure 5.1 we can see the observed voltage level on the RBL across different XAC values when operating at 1.2V. As expected, this graph proves that the RBL's voltage is a symmetric and monotonic function of the XAC value. Furthermore, an increased linearity is noticed in the middle of the graph at around XAC = [-128,+128] which is useful since this is the XAC value range that most computations will revolve around. This leads to an easier voltage distinction among the most commonly encountered XAC values. XAC value from 0 to 32 leads to an approximately 0.2 V difference when at the same time from XAC = 224 to XAC = 256 (again a XAC difference of 32) is a 0.08V difference in the RBL voltage.

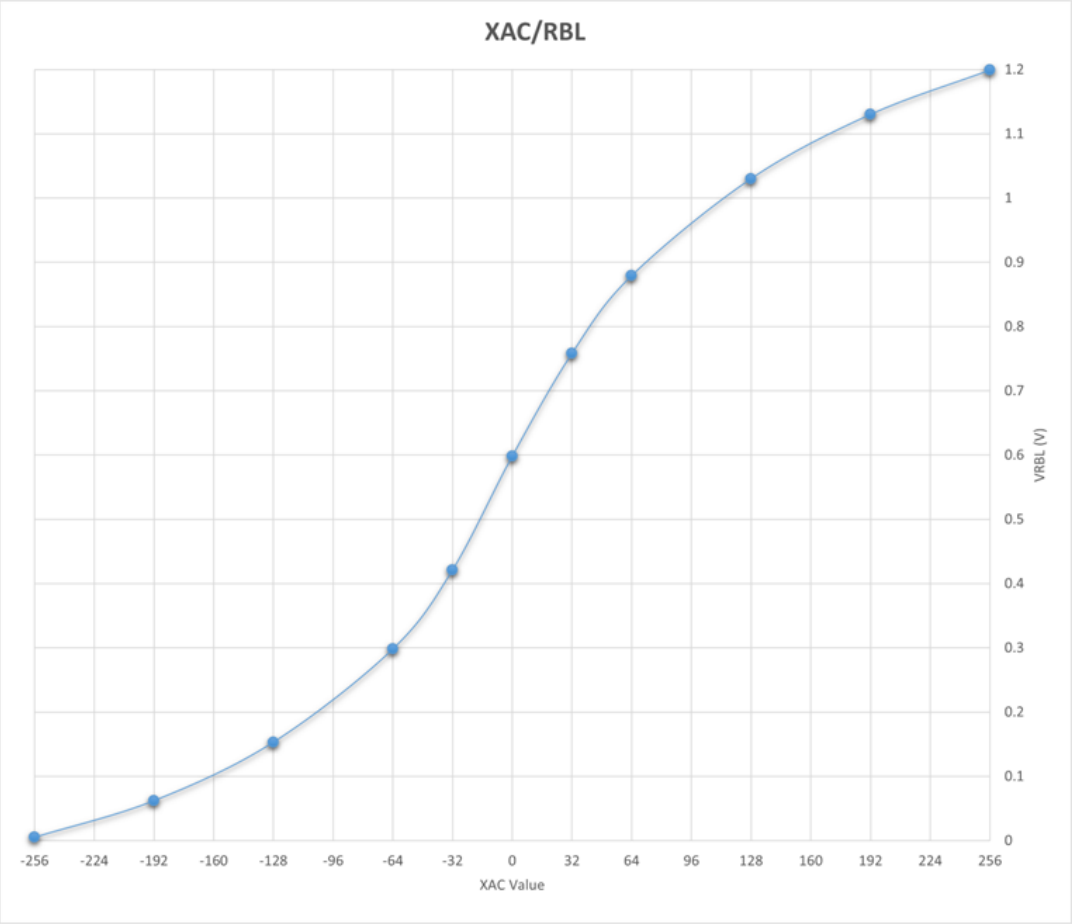


Fig. 5.1 RBL voltage according to RBL's XAC value

5.1 Memory Array Periphery

As mentioned earlier, one problem we faced was the fact that a higher voltage needs to be provided to the gate of the NMOS will need to be able to provide a higher-than-nominal voltage if we do not use a full passgate (12T design) connected approach. Another difficulty we encountered is related to the size of the transistors in the data buffers in order to achieve the required driving strength to feed multiple cells in a row as well as the RBL. Improper driving strength may result in incorrect calculations. On the other hand, oversizing the buffers negates any area advantages we may have gained by using less transistors in the 11T cell.

5.2 Power Consumption and Performance Measurements

We measured the energy consumption in a 256 cell column using a variety of metrics to demonstrate the different ways that those cells can be operated at. Initially, the energy consumption varies depending on the XAC value. For example, having all 256 cells operating either as pull up or down devices, no current paths from VDD to ground are formed. However, it is extremely rare to have a XAC value of -256 or +256. In any other case there is at least one path from the power supply VDD to the ground. This causes a significant current leakage which in turn affects the total energy consumption. The case in which the same number of cells operate as pull up and down devices (XAC value = 0) poses the worst case for energy efficiency according to Figure 5.2, where we can see that the least energy consumed is noticed at the extreme cases of XAC=-256 and XAC=+256 since no current path from VDD to GND is formed. Energy consumption peaks at the XAC value of 0 which makes sense since there is an equal number of cells pulling up and pulling down forming the most possible open paths from VDD to GND. Also , another noteworthy fact is that most of the energy consumed comes from current leakages during calculation (when there are open paths from VDD to GND) since $\frac{Worst\ energy\ consumption}{Best\ energy\ consumption} \cong \frac{1.7}{0.2} = 8.5$. All energy measurements in this graph refer to a 1.2V VDD.

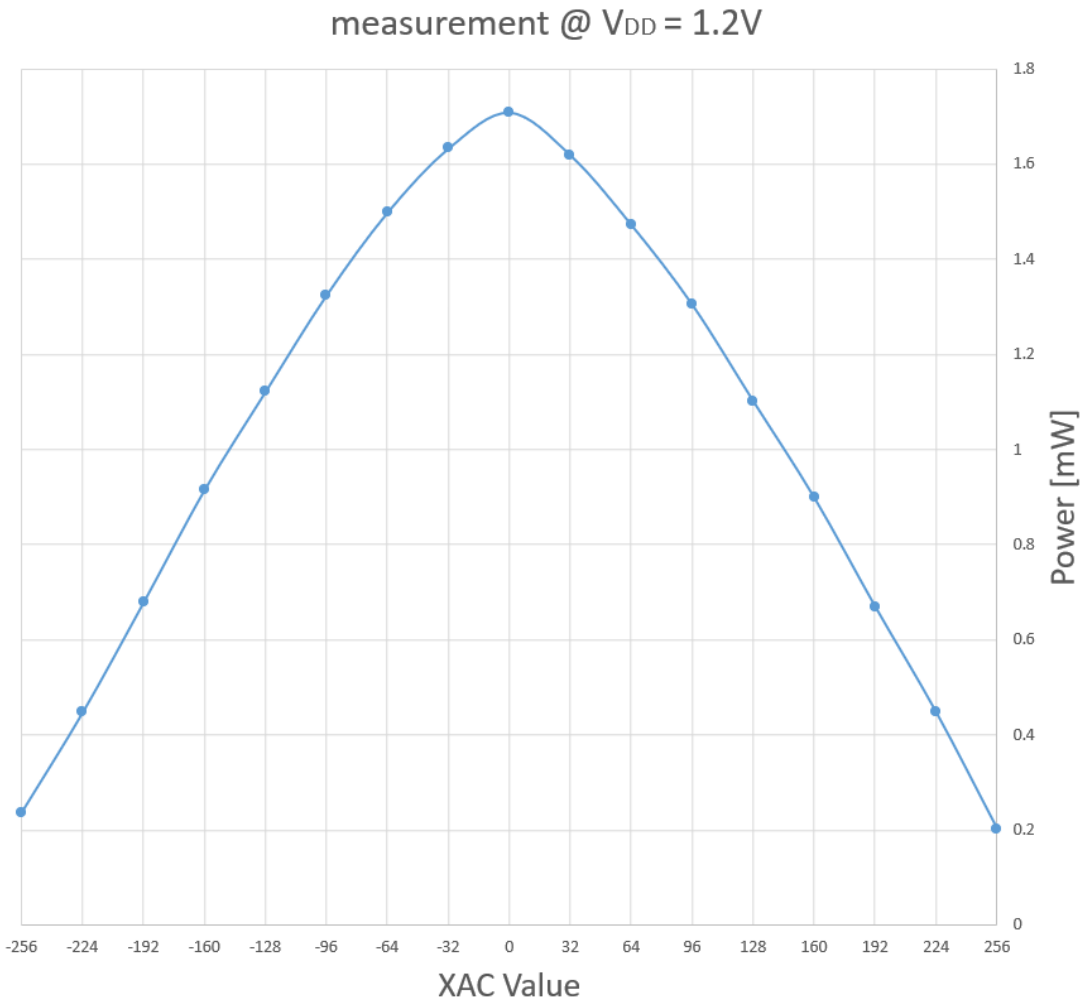


Fig. 5.2 Power consumption for a single column with respect to the XAC value

For XAC value equal to 0 we also measured the energy consumption across different power supply (VDD) voltage levels from 1.2V to 0.6V with a step of 50mV (0.05V), aiming to explore energy efficiency at low voltage operation as a tradeoff with respect to computation delay (Figure 5.3), in both proposed designs. Moreover, different voltages levels, higher than VDD, were considered for the activation of the NMOS pass-gate in the 11T cell design. In Figure 5.3 we observe that the 12T design performs similarly with the 11T highly overdriven designs. It is also evident that all designs converge both in terms of energy consumption as well as delay when we get to the lowest VDD operation points. Each design also has a point of diminishing returns after a certain voltage since below 0.8V the circuit doesn't reduce energy consumption that much but the computation delay increases drastically. In Figure 5.4 we see how the circuit with the 12T cell design performs under different VDD voltages in terms of delay

and energy consumption for an array of 256x64 cells. Again, it is fairly obvious that after the energy consumption reaches a point of around 100pJ energy gains start to reduce while computational delay increases significantly.

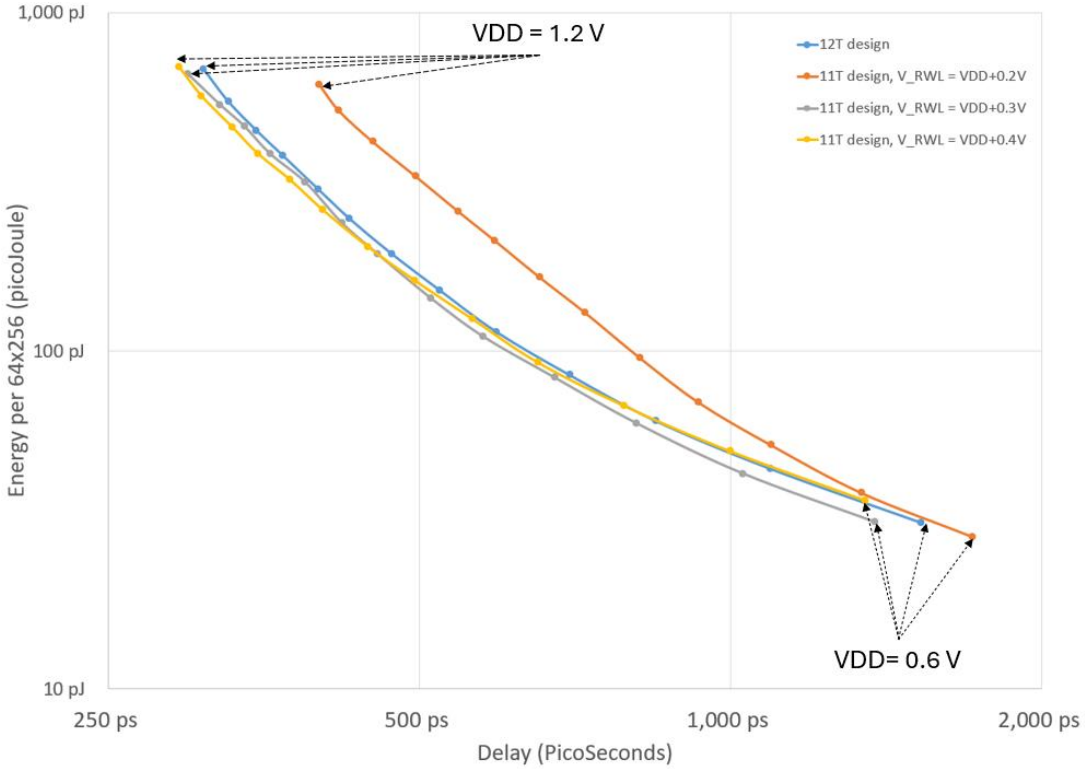


Fig 5.3 Energy consumption and computation delay for the entire array at different VDD operation points for both designs.

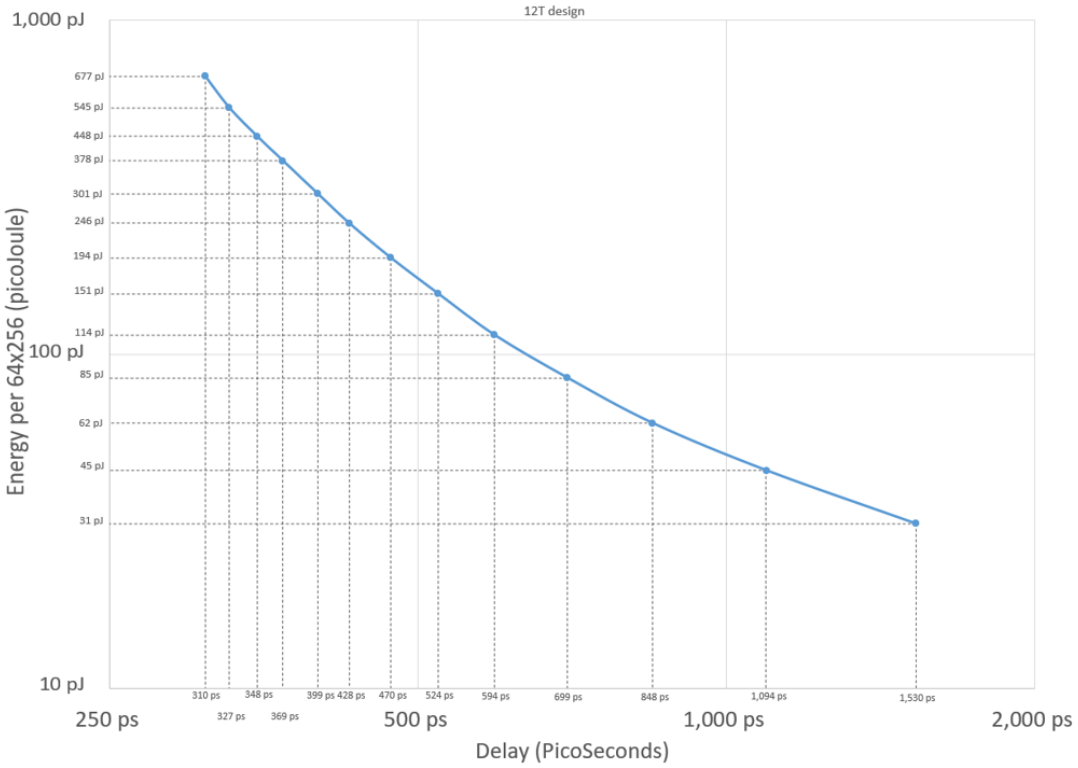


Fig 5.4 Energy consumption and computation delay at different VDD power supply levels for the entire array in the case of the 12T cell design.

5.3 Comparisons

In order to have a frame of reference we are now going to compare both of our designs to already existing work and specifically the 12T XNOR-SRAM cell proposed by Shihui Yin et al. [YS20]. Their proposed architecture was designed using a 65nm technology so comparing straight up with their results would be unrealistic since our technology is at 90nm. So, using their publication as reference, we designed their IMC SRAM in the same 90nm technology we used for our designs and tested it under the same conditions as the ones we previously used. PMOS transistors were scaled up properly to account for the lack of strength compared to their NMOS counterparts and we made no changes to the buffers. That means that any differences observed rely solely on the SRAM array and consequently to the design of each cell.

When it comes to the silicon area required for an entire grid of each implementation, we have already a significant advantage compared to [YS20]. In both our schematics we use only two data lines to provide the input data to each row of cells

(*data* and *data_bar).* On the other hand [YS20] uses four data lines namely *RWL_P, RWL_P_BAR, RWL_N, RWL_N_BAR.* Those two less datalines contribute to a decrease in area since they have to span the entire array 256 times providing data to 64 cells each. Our 11T design further improves upon that, by needing one less control line as well. It is not easy to estimate the precise silicon area savings without actually designing the circuit at the layout level. A rough estimate of the silicon area of the proposed designs could be as follows by considering the sum of the WxL of all transistors in the memory array.

$$\left[[180nm \times 90nm \times 2] + [120nm \times 90nm \times 7] + [520nm \times 90nm \times 2]\right] = 0.2016\ \mu m^2\ per\ cell$$

$0.2016\ \times 256 \times 64 = \mathbf{3303.0144}\ \boldsymbol{\mu m^2}$ for the entire array of 11T SRAM

$$\left[[180nm \times 90nm \times 2] + [120nm \times 90nm \times 7] + [520nm \times 90nm \times 3]\right] = 0.2484\ \mu m^2\ per\ cell$$

$0.2484\ \times 256 \times 64 = \mathbf{4069.7856}\ \boldsymbol{\mu m^2}$ for the entire array of 12T SRAM

$$\left[[180nm \times 90nm \times 2] + [120nm \times 90nm \times 7] + [520nm \times 90nm \times 3]\right] = 0.2484\ \mu m^2\ per\ cell$$

$0.2484\ \times 256 \times 64 = \mathbf{4069.7856}\ \boldsymbol{\mu m^2}$ for the entire array of [YS20]

In our 12T SRAM design the silicon area related to the cell transistors has no difference compared to [YS20] since it uses exactly the same number and size of both PMOS and NMOS transistors for the 6T memory subcell and the 6T computing subcell so it benefits only from using two less datalines, which as we mentioned above is no minor advantage. The 11T SRAM design, however, uses 6T for the memory subcell but uses only 5T for the computing subcell and on top of that this one less transistor is a PMOS which inevitably requires to be sized up to match the strength of the NMOS. Thus, the required silicon area is reduced at least by 18.84% with respect to [YS20].

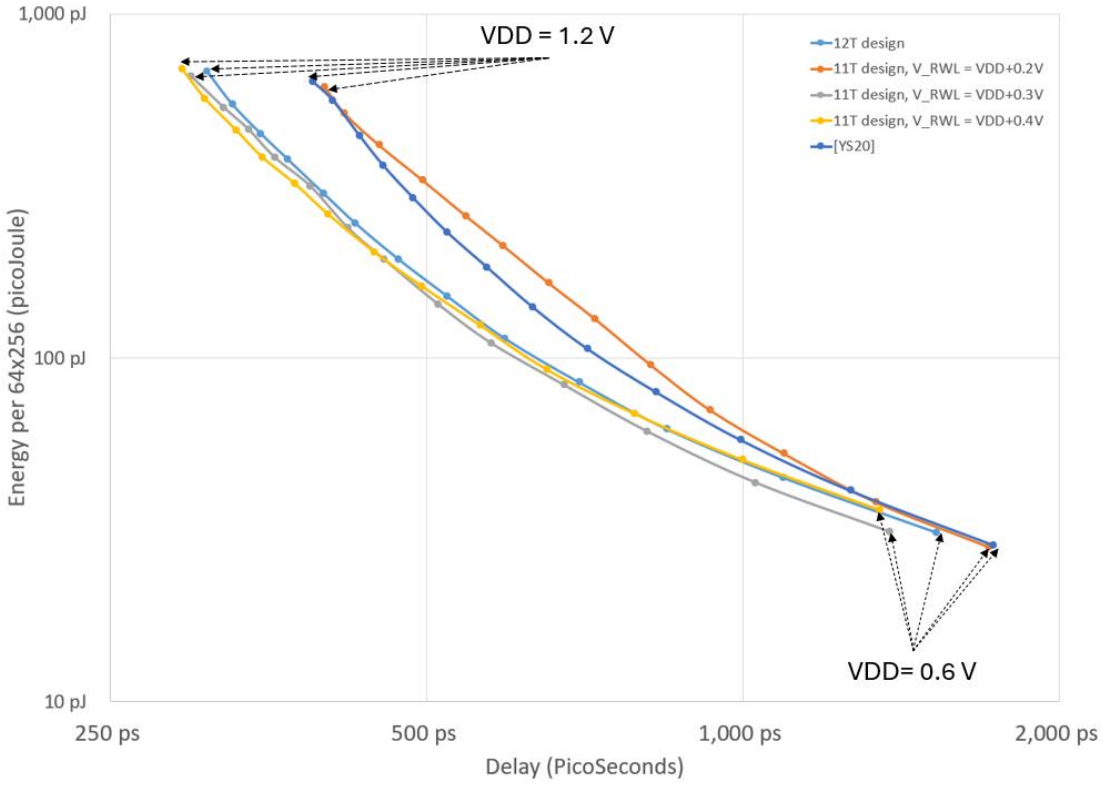


Fig. 5.6 Comparisons of energy and delay among the proposed designs and the design in [YS20]

In terms of energy consumed by the entire array we will once again exploit the graphs used in chapter 5.2 to compare our designs with the design in [YS20]. As we can see in Figure 5.6, [YS20] approach ensures similar delay results as our 11T SRAM design with RWL's voltage set to VDD + 0.2V while simultaneously consuming slightly more energy for higher VDD voltages. Its energy consumption is on par with our 12T SRAM design and this becomes obvious when examining Figure 5.7. In Figure 5.7 we notice that [YS20] design has a similar energy-delay product (EDP) to our 11T design with V_RWL = VDD +0.2V. Ideally, we want to minimize the EDP meaning that the desired computation can be achieved with the least energy expenditure and in the shortest time possible. All designs converge into a similar EDP value when supplied a voltage less than about 0.8V. In the higher VDD voltages the best EDP values come from our 12T design as well as our 11T designs whose RWL voltage is $\geq VDD + 0.3V$. [YS20] and our 11T design with V_RWL= VDD +0.2 have the highest EDP values since one is slightly faster in terms of time delay but the other consumes the least energy, providing overall similar EDP values.

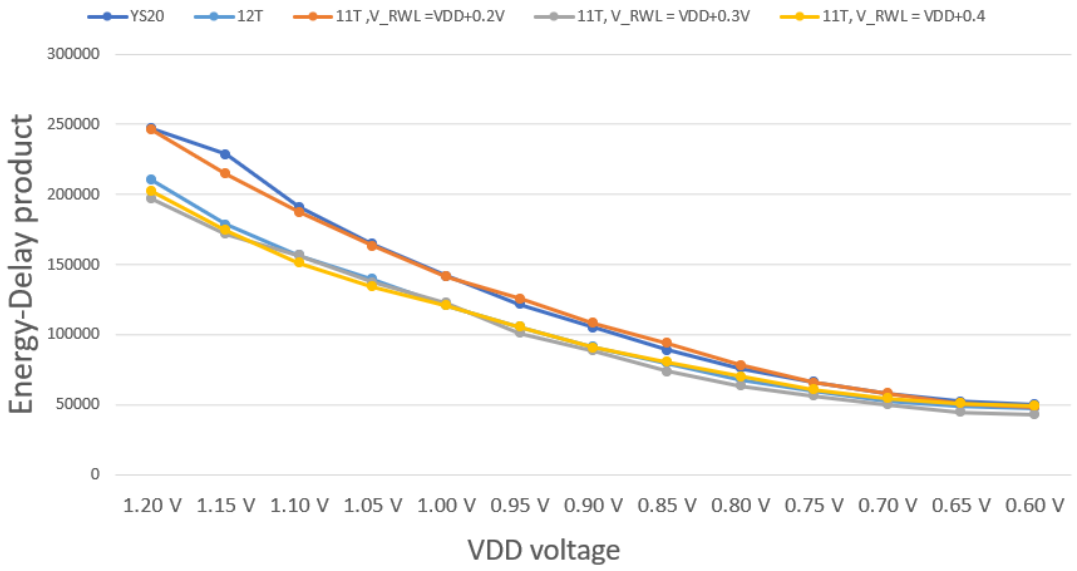


Fig. 5.7 Energy-delay product for different VDD voltages.

Chapter 6. Conclusions

This thesis presents two new SRAM-based IMC architectures that compute binary XNOR and Accumulate operations with possible applications in Binary Neural Networks and Convolutional Neural Networks with a balance of high energy efficiency, decreased silicon area and computational speed. Our 256 x 64 XNOR SRAM macro can assert all rows simultaneously in order to perform a 256 input XAC in a single cycle by accumulating the bitwise XNOR results of each cell in a common column line (read bitline – RBL) as a voltage value. This voltage can then be digitized by an Analog to Digital Converter implemented in the memory array periphery.

Our 11T XNOR SRAM cell design features lower silicon area requirements due to the fact that it needs one control line and two data lines less compared to a well-known work in the literature [YS20] to compute the bitwise XNOR result, but it does require a extra higher-than-nominal voltage to drive the used NMOS pass-gate. Using one more PMOS transistor to compose a full pass-gate, our 12T design eliminates the need for the extra voltage supply but this does come with an increase in silicon area since PMOS transistors have to be sized up while a second control line is also required. However, also in that case the design requires two data lines less compared to [YS20].

Overall, both designs are highly competitive both in terms of energy efficiency and computational delays when compared to [YS20]. Our 11T design achieves energy consumption reduction while still having reasonable computational delays while on the other hand if the voltage on its gate NMOS transistor is driven high enough it can still reduce time delays significantly but at the cost of more energy. Our 12T design accomplishes high computational speeds without requiring a higher-than-nominal voltage while maintaining its energy consumption at reasonable levels.

References

[ZSL18] M. Zidan, J. P. Strachan, W. Lu. *The future of electronics based on memristive systems,* Nat. Electron., pp. 1-22, 2018.

[ZWV17] 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.

[KKS+14] M. Kang, M.-S. Keel, N. R. Shanbhag, S. Eilert, and K. Curewitz, *An energy-efficient VLSI architecture for pattern recognition via deep embedding of computation in SRAM*, in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), pp. 8326–8330, 2014.

[L20] Z. Lin et al., "In-memory computing with double word lines and three read ports for four operands," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 28, no. 5, pp. 1316–1320, 2020.

[YYK+20] C. Yu, T. Yoo, T. T. Kim, K. C. Tshun Chuan, and B. Kim, *A 16 K current-based 8T SRAM compute-in-memory macro with decoupled read/write and 1–5 bit column ADC* in Proc. IEEE Custom Integr. Circuits Conf. (CICC), pp. 1–4, Mar. 2020.

[KKK+19] J. Kim, J. Koo, T. Kim, Y. Kim, H. Kim, S. Yoo, J.J. Kim. *Area-efficient and variation-tolerant in-memory BNN computing using 6T SRAM array,* in Proc. Symp. VLSI Circuits, 2019, pp. C118–C119.

[SKC+19] X. Si, W. S. Khwa, J. J. Chen, J. F. Li, X. Sun, R. Liu. *A dual-split 6T SRAM-based computing-in-memory unitmacro with fully parallel product-sum operation for binarized DNN edge processors* IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 66, no. 11, pp. 4172–4185, 2019.

[DMB14] Marijana Despotović-Zrakić, Veljko Milutinović, Aleksandar Belić. *Handbook of Research on High Performance and Cloud Computing in Scientific Research and Education,* Published by IGI Global, Chapter 13, pp. 308 – 340, 2014.

[BC19] 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.

[BMB+11] T. S. Böscke, J. Müller, D. Bräuhaus, U. Schröder, U. Böttger, in IEDM Technical Digest, IEEE, San Francisco, CA, USA 2011, p. 547.

[SGK+20] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, E. Eleftheriou. *Memory devices and applications for in-memory computing, pp. 1-16. Also available at https://www.nature.com/articles/s41565-020-0655-z*

[IP20] D. Ielmini, G. Pedretti. *Device and Circuit Architecture for In-Memory Computing,* Journal Advanced Intelligent Systems, pp. 1-19, 2020. Also available at https://onlinelibrary.wiley.com/doi/epdf/10.1002/aisy.202000040

[SCC+19] Gagandeep Singha, Lorenzo Chelini, Stefano Corda, Ahsan Javed Awan, Sander Stuijk, Roel Jordans, Henk Corporaal, Albert-Jan Boonstra. *Near-memory computing: Past, present, and future.* Microprocessors and Microsystems, vol 71, pp 1-15, 2019

[M65] G. Moore, *"Cramming more components onto integrated circuits"* MDPL Electronics magazine, Vol.38, Number 8,pp. 37-54, 1965.

[SEN+21] M. E. Sinangil, B. Erbagci, R. Naous, K. Akarvardar. *A 7-nm compute-in-memory SRAM macro supporting multi-bit input, weight and output and achieving 351 TOPS/W and 372.4 GOPS*, IEEE J. Solid-State Circuits, vol. 56, no. 1, pp. 188–198, 2021.

[HXT+15] S. Hamdioui, L. Xie, H.A.D. Nguyen, M. Taouil, K. Bertels, H. Corporaal, H. Jiao, F. Catthoor, D. Wouters, L. Eike, J. van Lunteren, Memristor based computation-in-memory architecture for data-intensive applications, in: 2015 Design, Automation and Test in Europe Conference (DATE), pp. 1718–1725, 2015.

[VRN+19] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-Mb in-memory-computing CNN accelerator employing chargedomain compute," IEEE J. Solid-State Circuits, vol. 54, no. 6, pp. 1789–1799, Jun. 2019.

[JYS+20] Z. Jiang, S. Yin, J. 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, 2020.

[AJR+19] A. Agrawal, A. Jaiswal, D. Roy, B. Han. *Xcel-RAM: Accelerating binary neural networks in high-throughput SRAM compute arrays*, IEEE Trans.

Circuits Syst. I, Reg. Papers, vol. 66, no. 8, pp. 3064–3076, 2019.

[H14] M. Horowitz, in 2014 IEEE Int. Solid-State Circuits Conf. Digest of Technical Papers (ISSCC), pp. 10–14, 2014.

[S70] H.S. Stone, *A logic-in-Memory computer*, IEEE Transactions on Computers C19 (1) 73–78, 1970.

[ESS92] D.G. Elliott, W.M. Snelgrove, M. Stumm, Computational ram: a memory-SIMD hybrid and its application to DSP, in: 1992 Proceedings of the IEEE Custom Integrated Circuits Conference, 1992, pp. 30.6.1–30.6.4, doi:10.1109/CICC.1992. 591879.

[IP18] Daniele Ielmini, H.-S. Philip Wong. *In-memory computing with resistive switching devices,* Nature Electronics*, Vol. 1,* pp. 1–11, 2018.

[BAW+15] S. Balatti, S. Ambrogio, Z. Wang, D. Ielmini. *True Random Number Generation by Variability of Resistive Switching in Oxide-Based Devices*, *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*. pp 1-8, 2015.

[YS20] S. Yin, J.S. Seo. *XNOR-SRAM: In-Memory Computing SRAM Macro for Binary/Ternary Deep Neural Networks,* In IEEE Journal of Solid-State Circuits, Vol. 55, No. 6, pp. 1733-1743, 2020.