Faculteit Industriële Ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Hybrid IGZO-Si 3T0C DRAM for Energy-Efficient Near-Memory MAC and Bitwise computing

Mohammed Murad Khalil Albayyouk
Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

PROMOTOR :
Prof. dr. ing. Kris MYNY

Gezamenlijke opleiding UHasselt en KU Leuven

Universiteit Hasselt | Campus Diepenbeek | Faculteit Industriële Ingenieurswetenschappen | Agoralaan Gebouw H - Gebouw B | BE 3590 Diepenbeek

Universiteit Hasselt | Campus Diepenbeek | Agoralaan Gebouw D | BE 3590 Diepenbeek
Universiteit Hasselt | Campus Hasselt | Martelarenlaan 42 | BE 3500 Hasselt

2024
2025

Faculteit Industriële Ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Hybrid IGZO-Si 3T0C DRAM for Energy-Efficient Near-Memory MAC and Bitwise computing

Mohammed Murad Khalil Albayyouk

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

PROMOTOR :

Prof. dr. ing. Kris MYNY

UHASSELT KU LEUVEN

Preface

This thesis, Hybrid IGZO-Si 3T0C DRAM for Energy-Efficient Near-Memory MAC and Bitwise computing, represents the culmination of my work over the past year. It explores a novel approach to integrating computation near DRAM arrays, aiming to reduce energy consumption.

I am grateful to my supervisors, Prof. Dr. Ing. Kris Myny, Ing. Djihad Narcereddine, Ing. Yari Nowicki, Ing. Sven Baerten and Ing. Jelle Biesmans. for their clear guidance and practical advice. Their expertise helped me refine both the experimental design and the overall scope of this project.

To my parents, thank you for your steady support and for always encouraging me to pursue my interests. Your confidence in my abilities provided the foundation I needed to tackle the challenges along the way.

I hope that the results presented here will offer useful insights for future developments in low-power memory architectures and inspire further research in energy-aware computing.

Contents

List of Tables

List of Figures

List of Abbreviations

CPU	Central Processing Unit
DRAM	Dynamic Random-Access Memory
IMC	In-memory Computing
NMC	Near-memory Computing
IGZO	Indium–gallium–zinc–oxide
SA	Sense Amplifier
3T0C	Three-transistor, Zero-capacitor
MAC	Multiply–Accumulate
PDK	Process Design Kit
WWL	Write Word Line
WBL	Write Bit Line
RWL	Read Word Line
RBL	Read Bit Line
a-Si:H	Hydrogenated Amorphous Silicon
LTPS	Low-Temperature Polycrystalline Silicon
TFT	Thin-Film Transistor
RRAM	Resistive RAM
CIM	Compute-In-Memory
1T1R	One-Transistor–One-Resistor
ADC	Analog-to-Digital Converter
DAC	Digital-to-Analog Converter
MRAM	Magnetoresistive RAM
MTJ	Magnetic Tunnel Junction
RISC-V	Reduced Instruction Set Computing-V
SoC	System on Chip
SN	Storage Node
WE	Write-Enable
RE	Read-Enable
ANDL	AND Word Line
ORL	OR Word Line
V_{DD}	Supply Voltage
V_{th}	Threshold Voltage
FSM	Finite-State Machine
PE_n	Precharge Enable (active low)
MOPS	Millions of operations per second
GOPS/W	Giga operations per second per Watt

Abstract

Data-intensive systems require low-power memory solutions, since conventional silicon DRAM incurs high refresh energy and conventional indium–gallium–zinc–oxide (IGZO) DRAM exhibits insufficient speed. This master's thesis investigates a hybrid IGZO–silicon 3T0C cell architecture for near-memory computing, aiming to design and validate a memory cell composed of an IGZO write transistor, an IGZO storage transistor, and a Si cascode transistor, as well as to assess its integration into a 32×32 array with full peripheral circuitry.

The decoder, sense amplifier, and controller were implemented in Si. Simulations were carried out using the Pragmatic IGZO PDK and the X-Fab 180 nm Si PDK, evaluating array write/read timings and retention, and characterizing peripheral circuit functionality and energy consumption.

Simulation results indicate a write access time of 55 ns with a total energy consumption of 131 pJ, while a complete READ operation takes 50 ns and consumes 116 pJ, of which 108 pJ is expended by the DRAM controller, with an estimated retention time exceeding 400 s. MAC operations deliver 290.9 MOPS at an energy efficiency of 111.1 GOPS/W, and bitwise operations achieve 11.76 MOPS at 4.31 GOPS/W. Row refresh consumes 131 pJ every 400 s (versus 64 ms in conventional Si), yielding an energy reduction of 818.6 nJ per refresh interval. Overall, the proposed architecture minimizes power consumption while maximizing throughput, demonstrating a scalable in-memory computing solution.

Abstract in Dutch

Data-intensieve systemen vereisen energiezuinig geheugen, aangezien conventioneel silicium-DRAM een hoog vernieuwingsenergieverbruik kent en IGZO-DRAM onvoldoende snel is. Deze masterproef onderzoekt een hybride IGZO–Si 3T0C-geheugencel voor near-memory computing. Die cel bestaat uit een IGZO-schrijftransistor, een IGZO-opslagtransistor en een Si-cascode-transistor, en is geïntegreerd in een 32×32-array met alle perifere circuits.

De decoder, de sense-amplifier en de controller zijn in silicium (Si) gerealiseerd. Simulaties met het IGZO-PDK van Pragmatic en het X-Fab 180 nm Si-PDK hebben de schrijf- en leestijd, de retentie, de functionaliteit en het energieverbruik van de perifere circuits geëvalueerd.

Simulatie-resultaten tonen een schrijftoegangstijd van 55 ns met een totale energie-consumptie van 131 pJ, terwijl een volledige lees-operatie 50 ns in beslag neemt en 116 pJ verbruikt, waarvan 108 pJ door de DRAM-controller, met een geschatte retentietijd van meer dan 400 s. MAC-operaties (multiply-accumulate) behalen een doorvoersnelheid van 290,9 MOPS bij een energie-efficiëntie van 111,1 GOPS/W, terwijl bitwise-operaties 11,76 MOPS bereiken bij 4,31 GOPS/W. Rijverversing vergt 131 pJ per 400 s (ten opzichte van 64 ms bij conventioneel silicium), wat neerkomt op een energiereductie van een 818,6 nJ per verversinterval. De voorgestelde architectuur minimaliseert het energieverbruik en maximaliseert de doorvoer, waarmee een schaalbare in-memory computing-oplossing wordt aangetoond.

Chapter 1

Introduction

The ever-widening gap between processor speeds and memory access times, called the 'memory wall', has become the main bottleneck in modern computing [1]. As data-intensive applications in machine learning, scientific simulation, and real-time analytics proliferate, the energy and latency costs of shuttling operands over the Central Processing Unit (CPU)-memory bus dominate both performance and power consumption [1]. In-memory computing (IMC) co-locates logic within the memory array, eliminating off-chip transfers and harnessing the high parallelism inherent in dense memory fabrics [2].

A hybrid Dynamic Random-Access Memory (DRAM) architecture is investigated that combines emerging oxide-semiconductor storage elements with conventional silicon logic to deliver low-leakage, high-throughput DRAM operations. Indium–gallium–zinc–oxide (IGZO) storage transistors exhibit ultra-low off-state leakage, dramatically extending data retention and reducing refresh energy. Silicon-based decoders, controllers, and sense amplifiers optimized for speed sustain high operational throughput despite IGZO's lower carrier mobility.

1.1 Context

Conventional DRAM cells remain separated from the CPU by a narrow, high-latency bus: each compute operation requires both a read and a write, amplifying energy and delay [3]. Digital IMC puts compute logic after the sense amplifier, while Near-memory computing (NMC) embeds it on the module buffer or logic die. However, both yield limited energy and latency gains because the logic stays outside the cell array and each read still triggers the sense amplifier (SA) [4]. Analog IMC approaches exploit charge-domain computation in multi-level cells for energy efficiency, but face severe design complexity, limited speed and sensitivity to process variations [4].

Thin-film transistor technologies based on IGZO offer orders-of-magnitude lower leakage compared to crystalline silicon, enabling dramatically extended retention and reduced refresh rates [5]. However, IGZO's lower mobility constrains its suitability for high-speed decoding and sensing functions, which remain the domain of silicon logic [6].

1.2 Problem Statement

No existing DRAM-compute proposal simultaneously achieves both minimal static power, through reduced leakage and infrequent refresh, and high operation speed via fast decode and sensing. Digital IMC suffers from limited energy-efficiency and latency gains [4]; analog IMC sacrifices speed and robustness [4]; IGZO-only designs lack the throughput required for high-rate logic [6]; silicon-only schemes optimize speed at the cost of leakage and refresh energy [4].

The central problem addressed here is:

- How can a DRAM macro be designed to exploit IGZO's ultra-low leakage for storage while retaining silicon's high-speed logic for decoding and sensing?
- Which cell topology, peripheral circuitry and activation schemes enable robust, low-energy operations entirely near the memory array without external data movement?

1.3 Objectives

The overall goal is a hybrid IGZO–Si three-transistor, zero-capacitor (3T0C) DRAM architecture capable of Near-DRAM bitwise and multiply–accumulate (MAC) operations with both low refresh power and high throughput. Specific objectives consist of:

1. **Architectural Design:** Definition and circuit realization of a 3T0C cell array combining IGZO storage transistors with silicon cascode devices, row decoder, DRAM controller, MAC-unit, Bitwise Logic Unit and strong-arm latch sense amplifiers.
2. **Energy-Retention Characterization:** Cadence-based simulations quantifying extended retention times and reduced refresh energy using the IGZO Process Design Kit (PDK) provided by Pragmatic and the Si PDK provided by XFab.
3. **Performance Evaluation:** Determination of the latency and throughput of bitwise and multiply–accumulate operations under representative workloads, demonstrating speed parity with silicon-only NMC despite IGZO's lower mobility.

1.4 Methodology

Three main phases structure the research:

- **Literature Survey and Gap Analysis:** Review of state-of-the-art DRAM topologies, emerging materials and IGZO device characteristics to identify design trade-offs.
- **Circuit and Array Design:** Implementation in Cadence Virtuoso of the hybrid 3T0C cell, writing the DRAM controller, row/column decoder, and bitwise logic unit in VHDL, synthesizing them using Cadence Genus synthesis tool and the standard-cell library provided by Xfab, importing the resulting netlists into Virtuoso, and designing the 3T0C memory cell and sense amplifier at the transistor level within Virtuoso.
- **Simulation and Characterization:** Device-level and macro-level simulations to extract leakage, retention, refresh energy, sense-amp offset, and operation latency.

1.5 Outline of the Thesis

The remainder of this document is organized as follows:

- **Chapter 2** reviews literature on IMC, NMC, DRAM cell topologies and emerging transistor materials, identifying the hybrid IGZO–Si opportunity.
- **Chapter 3** introduces three DRAM cell topologies, compares their pros and cons, and concludes that the hybrid 3T0C cell offers the best trade-off.
- **Chapter 4** presents the complete architecture of a memory array comprising hybrid 3T0C cells, silicon decoders, silicon sense amplifiers, and a silicon DRAM controller, and evaluates its performance against the state of the art.
- **Chapter 5** concludes with a summary of key findings and outlines possible directions for future research.

Chapter 2

IMC, NMC and Emerging DRAM Technologies: A Literature Survey

2.1 Von Neumann Architecture: Principles and Limitations

The Von Neumann architecture uses a single shared memory to store both instructions and data, which the CPU fetches over a common bus. The CPU comprises a control unit and an arithmetic–logic unit (ALU), with internal registers for temporary storage. Because the same bus is used for instruction fetch and data transfer, instruction and data accesses cannot occur simultaneously, creating the so-called *Von Neumann bottleneck*. Modern processors mitigate this with multi-level cache hierarchies, split instruction/data caches, prefetching, pipelining, and out-of-order execution, but the fundamental gap between processor speed and memory access remains severe [1]. As observed by [7], CPU performance has advanced faster than DRAM performance, causing idle cycles while awaiting data, the *memory wall* problem. As shown in Figure 2.1, processor performance has grown exponentially since 1980, whereas memory performance has only increased marginally.

Despite advances in interconnects and memory controllers, the memory wall persists for data-intensive applications such as scientific simulation, real-time analytics, and machine learning, where off-chip DRAM bandwidth and latency dominate system throughput.

2.2 IMC and NMC: Digital and Analog Approaches

IMC addresses the memory wall by performing computations within or adjacent to the memory array, thereby minimizing data transfer and exploiting the array's intrinsic parallelism.

Digital IMC approaches place the compute logic directly after the sense amplifier, while NMC schemes integrate it on the Dual In-line Memory Module (DDIM) buffer or the logic die of 3D DRAM; however, both yield only modest energy-efficiency and latency improvements because the logic remains outside the cell array and each access still triggers the SA [8], [4].

Analog IMC techniques leverage charge- or current-domain summation in memory crossbars to execute matrix-vector multiplications in a single step, achieving high energy efficiency but facing

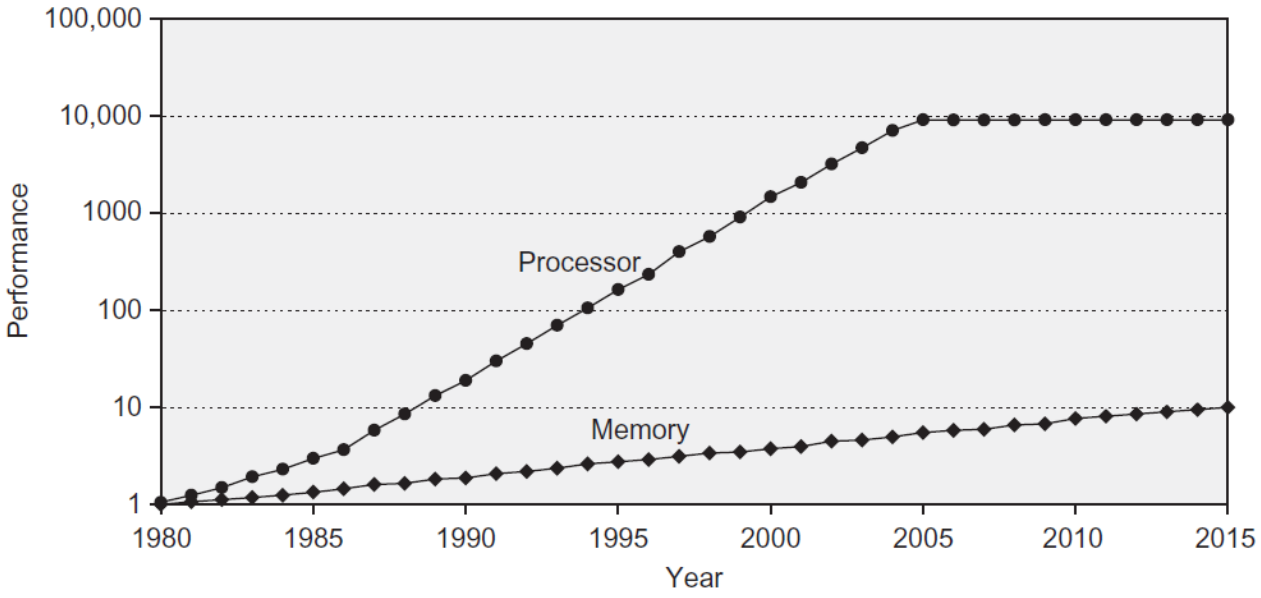

Figure 2.1: Processor vs. memory relative performance over the years [7, p. 80].

precision limits, process variation sensitivity, and Analog-to-Digital Converter (ADC)/Digital-to-Analog Converter (DAC) overhead [2]. Hybrid schemes combine analog computation with digital correction to balance efficiency and accuracy.

NMC lies between conventional IMC and standard architectures by placing lightweight processing elements in very close proximity to memory banks, often within the logic layer of 3D-stacked DRAM or on the same package, to reduce data movement overhead without the full area and retention penalties of in-array logic. Although NMC can deliver significant throughput and energy-efficiency gains, it still relies on explicit data transfers over short distances and introduces challenges in bandwidth contention, thermal management, and programming model support [9].

2.3 Emerging Transistor Materials and Technologies

This section reviews three thin-film semiconductor technologies relevant for memory arrays and peripheral circuits: amorphous silicon (a-Si), IGZO, and low-temperature polycrystalline silicon (LTPS).

2.3.1 Amorphous Silicon (a-Si)

Hydrogenated amorphous silicon (a-Si:H) can be deposited at 300 °C and is widely used in display backplanes [10], [11]. It exhibits low electron mobility (0.5–1 $\mathrm{cm}^2/(\mathrm{V{\cdot}s)}$) and an on/off current ratio on the order of 10^6 due to its high defect density [11], [12], and supports only n-type Thin-Film Transistors (TFTs), precluding CMOS logic integration [13]. a-Si:H TFTs are suitable for low-speed, large-area electronics such as pixel drivers and simple dynamic memories.

2.3.2 Indium Gallium Zinc Oxide (IGZO)

Amorphous IGZO combines low-temperature processing (below 250 °C), moderate electron mobilities (10–20 $\mathrm{cm}^2/(\mathrm{V{\cdot}s)}$), and ultralow off-currents, thanks to a wide bandgap of about 3.1 eV [14], [6].

IGZO decouples mobility from structural order, enabling long retention in capacitor-less DRAM cells (retention $> 400\,\mathrm{s}$) and reduced refresh energy [5]. Because IGZO TFTs are n-type only, peripheral CMOS logic must rely on silicon or LTPS. Their back-end-of-line compatibility makes IGZO prime for monolithic 3D memory integration [14].

2.3.3 Low-Temperature Polycrystalline Silicon (LTPS)

Low-temperature polycrystalline silicon (LTPS) is created by laser annealing amorphous silicon into poly-Si at approximately $500\,^{\circ}\mathrm{C}$ via XeCl excimer laser annealing processes [15]. LTPS TFTs yield carrier mobilities of 50–100 $\mathrm{cm}^2/(\mathrm{V}\cdot\mathrm{s})$ and on/off current ratios around 10^7 [16]. Unlike a-Si, LTPS supports full CMOS logic integration and is used in high-resolution display drivers and system-on-glass applications [17]. However, its process complexity, cost, and substrate-size constraints limit large-area scalability. LTPS excels in high-speed peripheral circuits (decoders, sense amplifiers) above dense memory arrays.

2.4 Architecture of a Standard 3T0C DRAM Cell

2.4.1 Cell Topology and Storage Mechanism

In the 3T0C DRAM cell architecture (Figure 2.2), data are retained solely by the intrinsic parasitic capacitance of a dedicated storage transistor, thereby obviating the need for a separate capacitor [18]. The memory cell comprises three transistors:

1. **Storage transistor M1**: Its own parasitic capacitance serves as the charge-storage element that represents the stored logic state.
2. **Write access transistor M2**: When the write-word-line (WWL) is asserted, this device connects the storage node to the write-bit-line (WBL), allowing charge to be deposited or removed.
3. **Cascode (read) transistor M3**: Activated by the read-word-line (RWL), it couples the storage node to the read-bit-line (RBL) while concurrently isolating the storage node from voltage disturbances during read operations. This isolation ensures non-destructive sensing of the stored charge.

Because the effective capacitance of the storage node is inherently smaller than that of a discrete capacitor in a conventional 1T1C cell, the retention time is reduced and refresh cycles must occur more frequently to maintain data integrity.

2.4.2 Write Operation

During the write operation of a 3T0C cell, the controller asserts the WWL and drives the WBL to the appropriate logic level (high for a "1," low for a "0"). Under these conditions, the write transistor M2 becomes conductive, enabling the parasitic capacitance of the storage transistor M1 to be charged or discharged to reflect the intended logic state. Once the write cycle is complete, the WWL is deasserted, thereby isolating the parasitic capacitance and preserving its newly established charge.

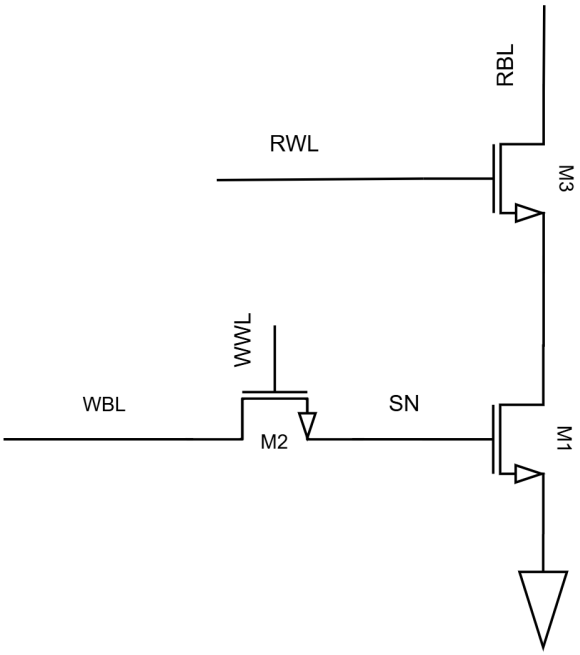

Figure 2.2: 3T0C DRAM

2.4.3 Read Operation

During the read operation of a 3T0C cell, the controller asserts the RWL, thereby activating the cascode transistor M3 and coupling the drain of the storage transistor M1 to a precharged RBL, which is typically biased at half the supply voltage. Depending on the charge stored on the parasitic capacitance of M1, the RBL either discharges (if the stored logic value is "1") or remains at its precharged level (if the stored value is "0," since M1 remains non-conductive). A sense amplifier subsequently monitors the RBL voltage and converts its deviation (or lack thereof) into a definitive logic output. Under ideal conditions, this sensing process does not perturb the potential at the storage node; however, in poorly optimized designs, parasitic coupling between the RWL and the M1 drain, and between the M1 drain and the storage node, can induce unwanted disturbances in the stored charge.

2.4.4 DRAM array architecture

The system-level organization of a 3T0C DRAM array follows a hierarchical, modular design that balances access parallelism, density, and ease of integration with a host controller (Figure 2.3). At the highest level, the memory array is partitioned into multiple banks, each of which can be independently activated, read, written, or refreshed. By interleaving accesses across banks, the controller can hide the intrinsic latency of individual row-activation and precharge operations, thereby improving overall throughput.

Within each bank, the storage region is further divided into a grid of subarrays, each of which houses a two-dimensional matrix of 3T0C cells. A global row decoder accepts a row address and routes it to the target subarray's local row decoder. Upon activation, the selected local decoder asserts the corresponding RWL or WWL across the entire row, enabling either the cascode transistor M3 (for reads) or the write transistor M2 (for writes) in every cell of that row.

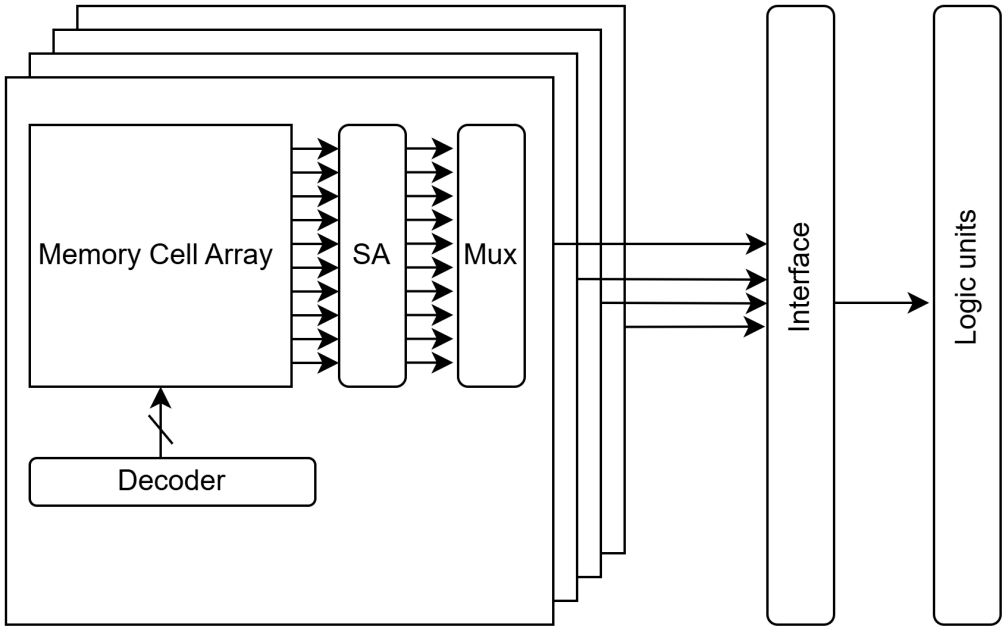

Figure 2.3: DRAM array architecture

Once a row is activated, the tiny voltage differential on each storage node is captured by its sense amplifiers, which serve a dual role as both read amplifiers and the bank's precharged row buffer. Following amplification, a column decoder and an associated multiplexer network select one or more bitlines from the row buffer according to the low-order column address. This column MUX collapses the wide row-buffer payload down to the fixed width of the external I/O bus, minimizing pin count and I/O logic complexity.

The time from request arrival to output signal in a read operation, often referred to as the random cycle time, is a critical performance metric because it directly determines the maximum sustainable bandwidth of the memory. Similarly, the write access time measured from the assertion of the WWL to the stable storage-node-voltage is key to understanding the peak write rate.

In their 2kb 3T0C eDRAM macro fabricated in a 0.18 μm Si CMOS process, Giterman et al. [19] report:

- A **write-access time** of 1.3 ns.
- An error-free **read random-cycle time** of 25 ns.

In their capacitor-less 2T0C DRAM cell comprising a-ITGZO TFTs, Ryu et al. [20] report:

- A **write–access time** of approximately 9.1 μs.
- An **error-free read random-cycle time** of approximately 9.1 μs.

2.5 RRAM- and MRAM-based Compute-In-Memory Architectures

2.5.1 RRAM-based IMC Architectures

Resistive RAM (RRAM) devices have become a workhorse for analog IMC due to their simple 1T1R (one-transistor–one-resistor) or 2T1R cell structures, high density, and ability to store multi-bit weights. In a typical RRAM IMC crossbar, each cell's conductance represents a matrix weight, and by applying analog voltages to wordlines while summing currents on bitlines, large-scale vector–matrix multiplications can be performed in one cycle [21], [22].

Dense RRAM IMC Macros Liu *et al.* fabricated a 28 nm, 576 kbit RRAM IMC macro that leverages a hybrid on-chip programming scheme to achieve an area efficiency of 2.82 TOPS/mm^2 and an energy efficiency of 35.6 TOPS/W [21]. This macro performs 128×128 analog MACs in parallel and uses a custom hybrid ADC for both weight programming and inference, reducing programming energy by 85 % compared to prior designs.

Ye *et al.* demonstrated a 28 nm hybrid 2T1R RRAM IMC core for energy-efficient AI edge inference, achieving sub-20 ns read/write times per cell and $>$ 50 TOPS/W for 6-bit–precision MAC operations at 10 MHz clock [22]. Their design uses a weighted hybrid 2T1R cell array with redundant subarray mapping to mitigate device non-idealities.

Scalability and Precision Earlier RRAM IMC works include Xue *et al.*, who reported a 22 nm, 4 Mb RRAM IMC macro capable of 8-bit MAC with 11.91–195.7 TOPS/W (depending on precision mode) [23]. Yoon *et al.* (2022) demonstrated a 40 nm, 64 kb RRAM IMC macro with read-disturb tolerance and an energy efficiency of 56.67 TOPS/W for 4-bit MACs [24]. These designs confirm that RRAM IMC scales from tens of kilobits to multiple megabits while retaining $>$ 10 TOPS/W energy efficiency, albeit at the cost of additional ADC/DAC overhead for high bit-width operations.

2.5.2 MRAM-based IMC Architectures

Magnetoresistive RAM (MRAM) offers non-volatility, near–SRAM-level speed, and endurance $> 10^{12}$ cycles, making it an attractive platform for IMC [25]. In traditional MRAM cells, each bit is a 1T1MTJ (one-transistor, one-magnetic tunnel junction) whose resistance toggles between a low (parallel) or high (antiparallel) state.

Analog MRAM Crossbar (Resistance-Sum) Jung *et al.* introduced the first 64×64 MRAM-based IMC prototype, which overcomes MRAM's inherently low resistance ratio by using a *resistance-sum* readout rather than conventional current-sum [25]. Their 28 nm prototype integrates read-out electronics and performs a two-layer perceptron for MNIST digit classification with 93.2 % accuracy (software baseline: 95.2 %). The array achieves 5 MHz analog compute rate (limited by ADC integration time), 20 ns per analog MAC, and $<$ 2 pJ/MAC inferred energy (excluding ADC), demonstrating that MRAM can perform in-situ multiply–accumulate operations with minimal additional circuitry.

Digital MRAM CAM/Bitwise Operations Deaville *et al.* presented a 22 nm, 128 kb MRAM row/column-parallel in-memory computing macro with memory-resistance boosting and multi-column ADC readout [26]. This design supports fully parallel matrix-vector multiplication in a 128 kb array, achieving robust performance through active compensation for MRAM non-idealities while retaining energy efficiency in the low pJ/bit range.

2.6 System-Level Integration with RISC-V and SoCs

To exploit IMC accelerators within conventional processing environments, recent works integrate RRAM/MRAM IMC blocks alongside RISC-V cores or embed them into full System on Chip (SoC) designs.

RISC-V + RRAM IMC Caon *et al.* proposed two NMC architectures, NM-Caesar and NM-Carus, that tightly couple RRAM IMC tiles with a 32-bit RISC-V core via an on-chip interconnect [27]. In post-layout 28 nm simulations, NM-Caesar achieves a 23.2× system-level energy efficiency improvement and 25.8× speedup over a baseline RV32IMC CPU for 8-bit matrix multiply workloads, peaking at 306.7 GOPS/W. NM-Carus trades some area for flexibility, allowing dynamic reconfiguration of RRAM IMC array shapes under software control.

SRAM-IMC + RISC-V SoC Guo and Chang introduced "CIMR-V," an end-to-end SRAM-based IMC accelerator with a RISC-V core that supports layer fusion, convolution/pooling pipelines, and custom IMC instructions for full AI model inference [28]. Fabricated in 28 nm, CIMR-V attains 3707.8 TOPS/W energy efficiency (8-bit MACs at 50 MHz) and 26.21 TOPS peak throughput, reducing keyword-spotting latency by 85.1 % versus a non-IMC approach.

Hybrid 3D-Stack SoC with Multiple Memory Technologies Yan *et al.* presented "Eq-CIM," a monolithic 3D IGZO-RRAM-SRAM integrated architecture that assigns each memory technology a specific role, IGZO for temporal activation storage, RRAM for high-density weight storage, and SRAM for accurate IMC [29]. By co-designing the 3D interconnect and peripheral circuits, their prototype achieves 5.05× area efficiency and 2.45× energy efficiency gains over single-type IMC architectures.

2.7 Summary

This chapter presented the classical Von Neumann architecture and the memory wall challenge, motivating in-memory and near-memory computing paradigms. Digital and analog IMC approaches were surveyed, 3T0C DRAM cell topology was detailed, and a new comparison of RRAM- and MRAM-based IMC architectures—including dense crossbar MAC implementations, digital bitwise operations, and system-level integrations with RISC-V cores and hybrid 3D-stack SoCs—was added. Emerging thin-film transistor materials (a-Si, IGZO, LTPS) and their roles in monolithic 3D integration were reviewed. These insights set the stage for the hybrid IGZO–Si 3T0C DRAM architecture developed in this thesis.

Chapter 3

Comparative Analysis of DRAM Cells

3.1 Overview of Design Variants

To explore the trade-offs between leakage, cell area, operational speed, and logic flexibility, four DRAM cell microarchitectures have been implemented and evaluated in Cadence. All variants exploit the storage capability of indium–gallium–zinc–oxide (IGZO) transistors but differ in cell topology and the degree of peripheral assistance required:

- **Analog 3-T IGZO Cell:** A fully IGZO-based, 3-transistor cell.
- **Hybrid 2-T IGZO + Si Cascode transistor:** A 2-transistor IGZO cell augmented by a silicon (Si) cascode transistor to enhance speed and discharge isolation.
- **Digital 4-T IGZO Cell:** A fully IGZO-based, purely digital 4T0C design supporting OR/AND logic.

Each variant is evaluated in terms of cell area, static leakage current, read latency, supported logic operations, and required refresh interval. Sections 3.2 through 3.3 describe the topology and operation of each cell, and Section 3.4 presents a quantitative comparison.

3.2 Analog 3-T IGZO Cell

In an initial attempt to realize a fully functional DRAM cell capable of performing bitwise operations by activating two rows simultaneously, a two-transistor–zero-capacitor (2T0C) IGZO cell was investigated. However, this topology exhibits several limitations in its read operations, as well as in its ability to perform in-memory bitwise functions. Based on the shortcomings identified, a three-transistor–zero-capacitor (3T0C) architecture is proposed to overcome these challenges.

3.2.1 2T0C IGZO Cell

Topology

Figure 3.1 depicts the schematic of the full-IGZO 2T0C DRAM cell [30]. It comprises two thin-film transistors: a write transistor (L = 1 μm and W = 4 μm), whose gate is connected to the write word line (WWL) and whose drain is tied to the write bit line (WBL), and a storage

transistor (L = 1 μm and W = 6 μm), whose gate serves as the storage node (SN). The logic state is held on the intrinsic gate capacitance of the storage transistor.

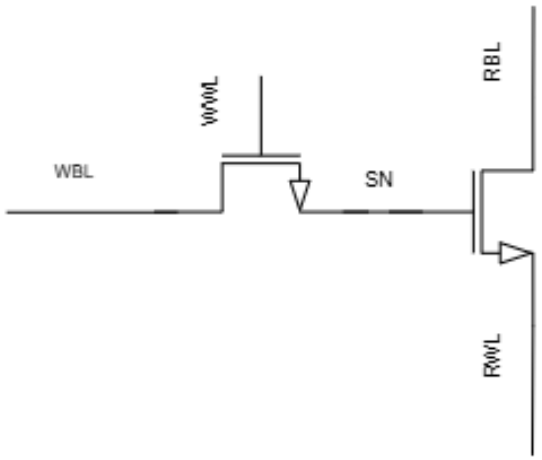

Figure 3.1: 2T0C DRAM

Limitations in Read/Write and Bitwise Operations

Figure 3.2 illustrates six 2T0C cells arranged in a 2 × 3 array, the read bitline was modeled with a capacitance of 250 fF. Simulations were carried out on this configuration to evaluate write and read operations and to assess the feasibility of performing in-memory bitwise operations.

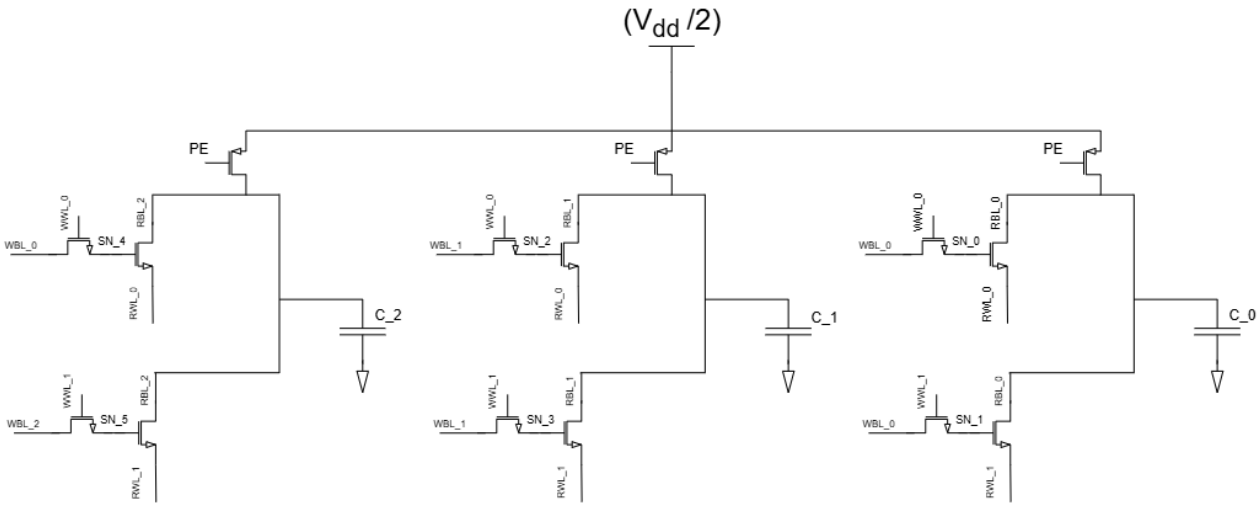

Figure 3.2: 2 × 3 array of Full IGZO 2T0C DRAM cells

Writing Writing to the cell proceeds analogously to conventional 1T1C DRAM. To store a logical value, the targeted row's WWL is asserted to 1.8 V, enabling conduction between the WBL and the SN. The data bit on WBL (1.8 V for logic '1' and 0 V for logic '0') is transferred to the SN capacitance over a pulse duration of approximately 50 ns. Upon deassertion of WWL (driven back to −0.8 V), capacitive coupling between the gate and source of the write transistor

causes the SN potential to shift: a stored '1' initially at 1.4 V equilibrates to 1.27 V and a stored '0' equilibrates form 0 V near −0.81 V.

Writes are performed on the schematic in Figure 3.2 one row at a time: the first row's WWL is asserted, then deasserted, followed by the second row. Figure 3.3 shows the simulation results of the writing operation of the cells from Figure 3.2.

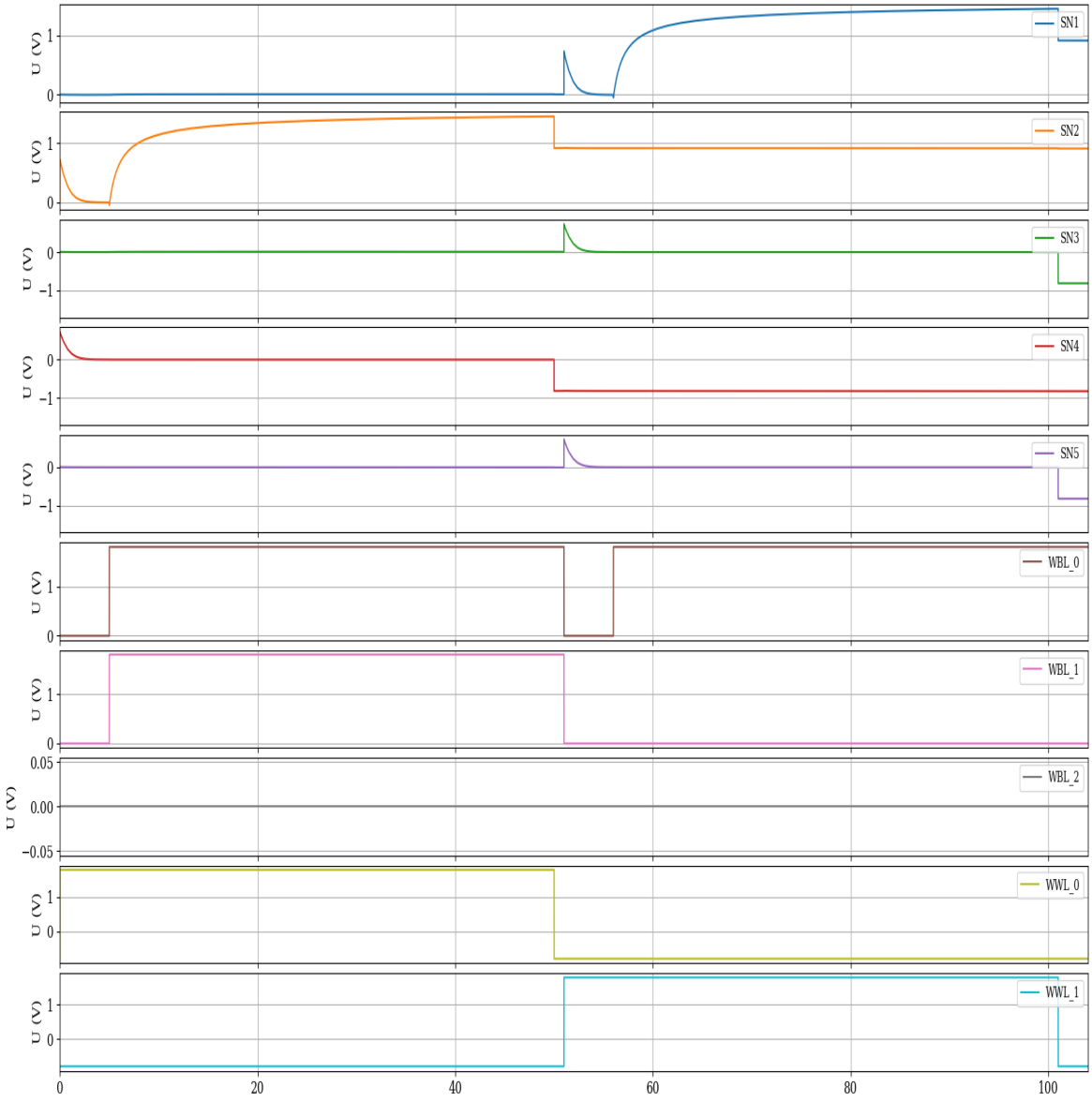

Figure 3.3: Writing a 2 x 3 matrix of 2T0C DRAM's

Reading and Bitwise operations For performing bit-wise operations between two bits from the same columns from Figure 3.2, all bit lines are precharged to $V_{DD}/2$. The read word line (RWL) is then driven low (i.e. 0 V enabled, 1.8 V disabled), thereby turning on all storage transistors in the two rows simultaneously. Depending on the stored bits in each column, the discharge rate of the bit line varies:

- **0 + 0:** no discharge,
- **1 + 0:** nominal discharge,

- **1 + 1:** accelerated discharge.

The objective was to terminate the bitline discharge by deactivating the rows at the exact point when the sense amplifier can reliably distinguish among the three cases.

However, capacitive coupling between the source and gate of the storage transistor induces a large voltage shift on the storage node, which effectively erases the "1" state and prevents any significant discharge even when a high voltage is stored. This phenomenon is observed under dual-row activation as well as single-row activation.

Figure 3.4 illustrates the failed read when both rows are enabled.

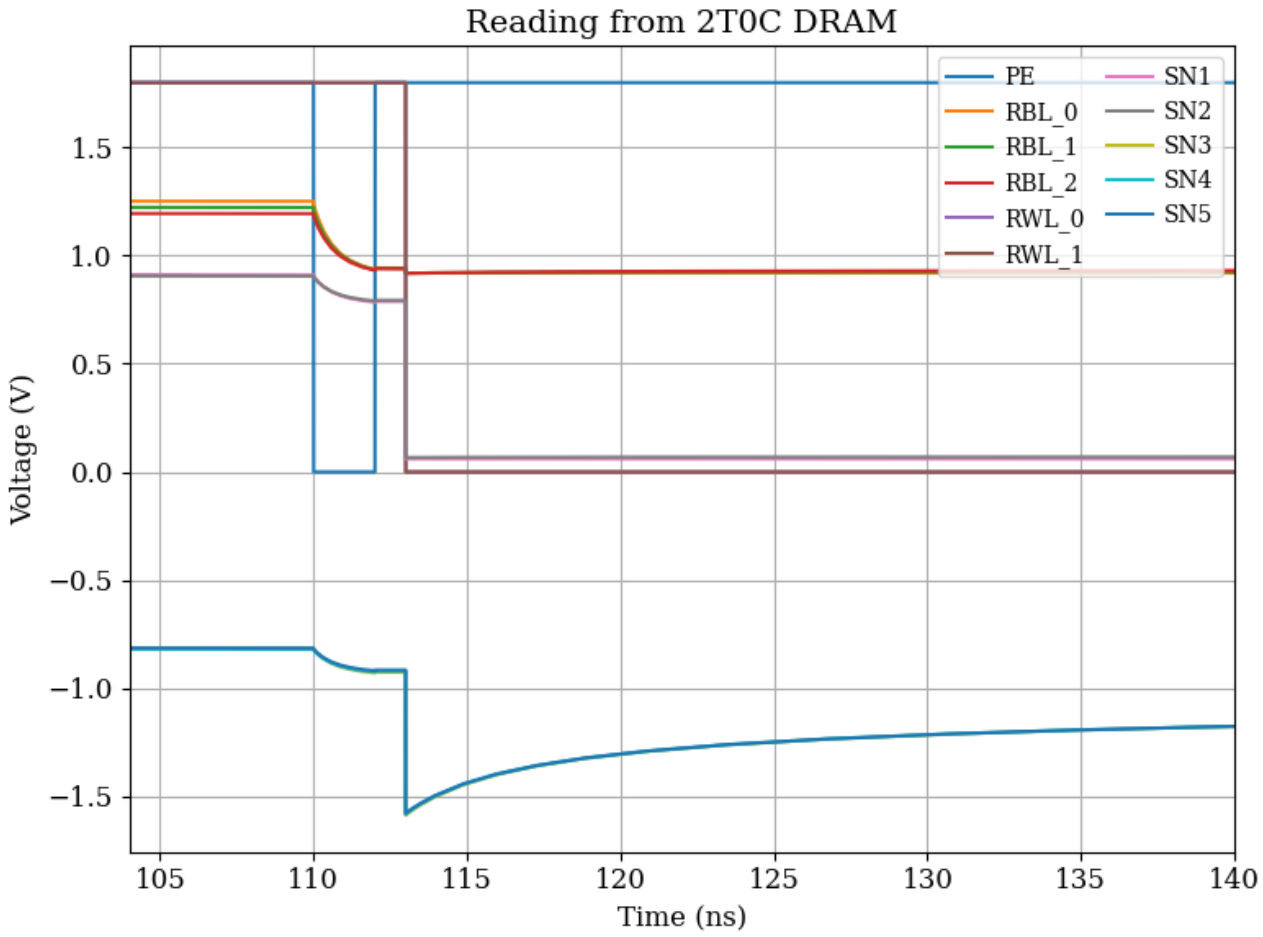

Figure 3.4: A failed read of a 2T0C DRAM

Introduction of Storage Capacitor To mitigate the coupling effect, a discrete capacitor is added at the storage node, resulting in a 2T1C cell. While this preserves the gate voltage during read operations, the bitline discharge profile still depends on the number of cells storing a logical '1' connected to the same bit line. Once the voltage on the bitline drops to the point where $V_g - V_{\text{bitline}} > V_{\text{th}}$, the floating bit line can act as a source for the read transistors of all connected '1' cells, inadvertently turning them on and partially recharging the line. Figures 3.5 and 3.6 illustrate the discharge behaviour with zero neighbouring '1' cells and with 28 neighbouring '1' cells, respectively.

These limitations motivate the design of a 3T0C IGZO cell, which is introduced in the next

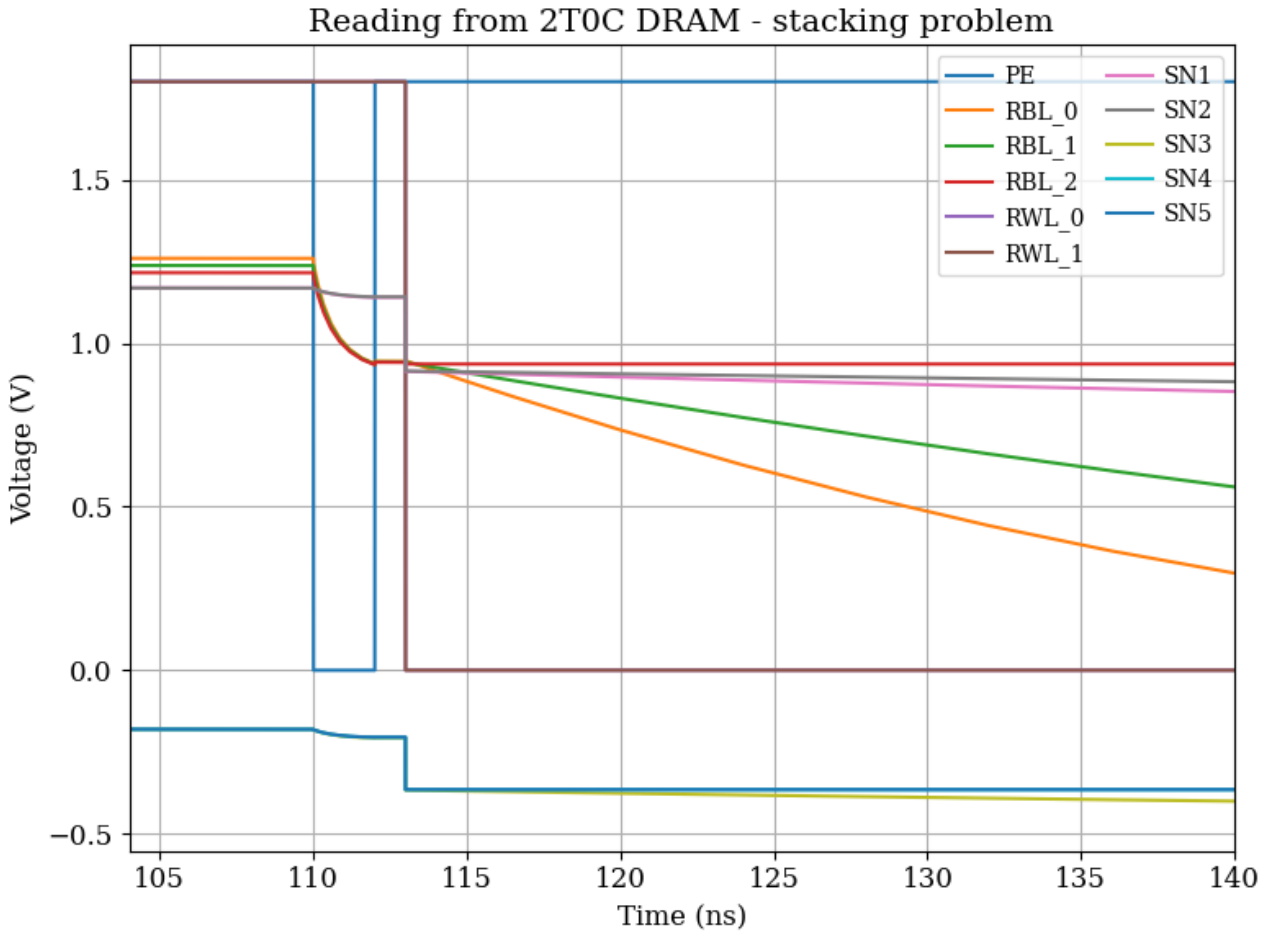

Figure 3.5: Reading from a 2 x 3 matrix of a 2T1C DRAM: only two cells are connected

subsection.

3.2.2 Proposed 3T0C Full IGZO Cell

Cell Topology

Figure 2.2 illustrates the schematic of the proposed 3T0C full IGZO DRAM cell, comprising three transistors. The write transistor M2 (L = 1 μm and W = 4 μm) has its gate connected to the WWL and its drain to the WBL. The storage transistor M1 (L = 1 μm and W = 6 μm) is arranged with its gate tied to the source of M2 and its source referenced to ground; the parasitic capacitance of M1 serves as the storage node. A cascode transistor M3 (L = 1 μm and W = 6 μm) provides the third element: its gate is driven by RWL, and its drain connects to the read bit line (RBL). By isolating M1 from the bit line, the cascode device overcomes the principal limitation of the conventional 2T0C DRAM cell, ensuring that bitline discharge no longer depends on the number of cells storing a logical '1' within the same column.

Write Operation

In the write phase, the memory controller asserts a write-enable (WE) signal, which swings from 0 V to 1.8 V at its logic-high level. This WE pulse is level-shifted to a negative gate voltage of −0.8 V on the WWL via a dedicated shifter circuit. Initially, with the WBL held at 0 V to

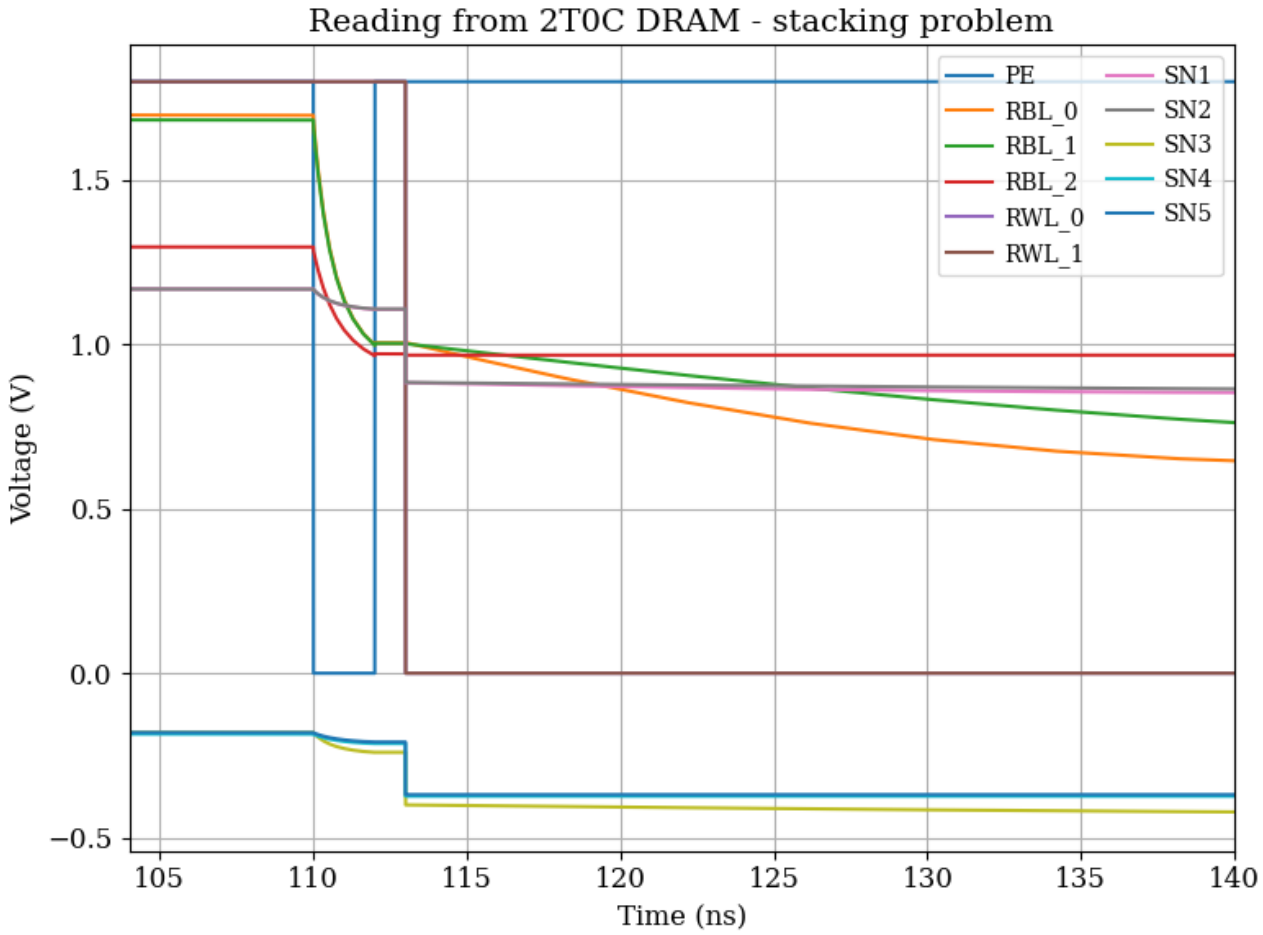

Figure 3.6: Reading from a 2 x 3 matrix of a 2T1C DRAM: 28 cells of logical one are connected to the same bitline

clear any previous state, After a latency of 5 ns, The controller drives the WBL to the desired data level—either 1.8 V for logic "1" or 0 V for logic "0"—and holds this potential for 50 ns to transfer charge onto the parasitic capacitance of transistor M1. At the conclusion of the write interval, the WE signal is deasserted, returning the WWL to −0.8 V and isolating the storage node. During this transition, capacitive coupling between the gate of M2 and the storage node of M1 induces a slight perturbation of the stored voltage: a written "1," nominally held at 1.5 V, shifts downward to approximately 0.964 V, whereas a written "0," nominally at 0 V, shifts to about −1.2 V.

Read Operation

During a read cycle, the controller asserts a read-enable (RE) signal, also from 0 V to 1.8 V, which is level-shifted to −0.8 V on the RWL to activate the cascode transistor M3. The RBL is precharged to $V_{DD}/2$. If the storage node holds a logic "1," transistor M1 conducts and pulls the RBL toward ground; if the storage node is at "0," M1 remains off and the RBL remains at $V_{DD}/2$. A sense amplifier detects this voltage deviation and resolves it into a full-swing logic output. Although an ideal design ensures non-destructive sensing, parasitic capacitances between the RWL and the M1 drain, and between the M1 drain and the storage node, can transiently disturb the stored voltage. Upon deassertion of the RWL, these perturbations are reversed,

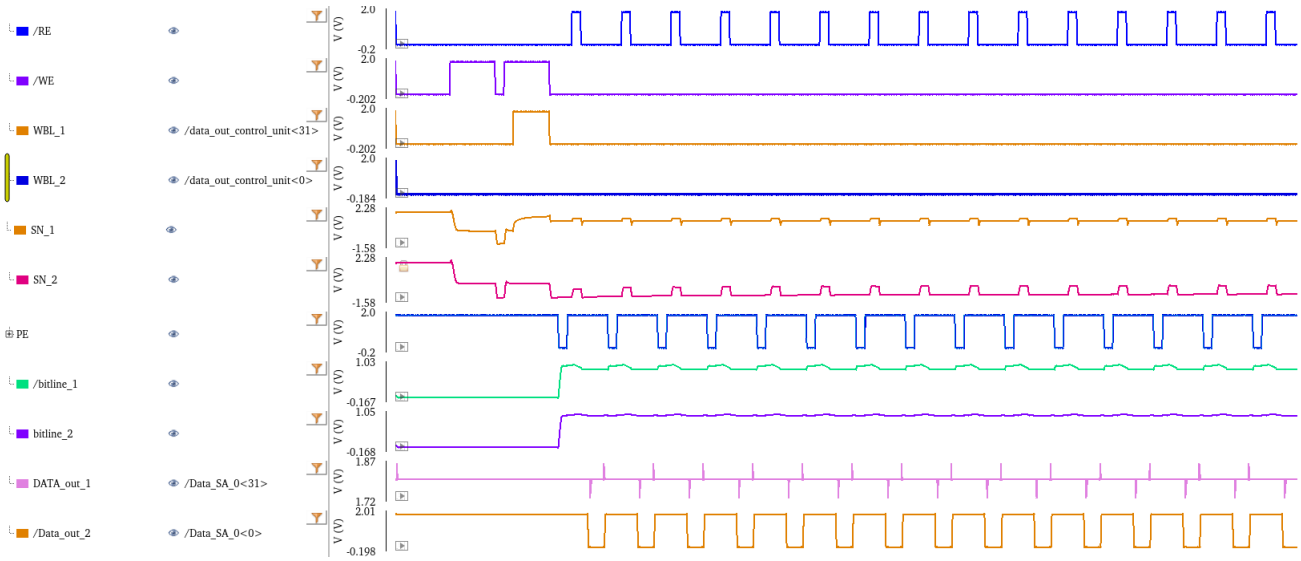

Figure 3.7: Writing and reading two Full IGZO 3T0C DRAM cells from one row

thereby restoring the storage node to its original pre-read potential; however, variations in the RWL signal can introduce deviations from this ideal restoration.

Figure 3.7 illustrates the simulation results obtained when two cells in the same row are written sequentially and then read continuously.

Bitline Sensing

To detect the discharge of the bitline, a strong-arm latch sense amplifier was employed (see chapter 4). The most critical parameter of this amplifier is its input-referred offset voltage, which for the current design is 17 mV. Consequently, to guarantee reliable sensing, the voltage differential between a discharged and a non-discharged bitline must exceed twice the offset, i.e. 34 mV.

A Monte Carlo analysis was performed to characterize the bitline discharge curves under process variation. After a discharge interval of $t_{\mathrm{d}} = 18\,\mathrm{ns}$, the voltage difference ΔV between discharged and non-discharged bitlines reaches 34 mV. At this point, the discharge phase can be terminated and the sense amplifier can be enabled. Figure 3.8 presents the simulation results of the Monte Carlo discharge curves.

Replacement of IGZO Cascode Transistor with Silicon Transistor

In order to improve mobility and reduce process-induced variation, the IGZO cascode transistor M_3 was replaced by a silicon (Si) transistor. Due to the higher electron mobility of crystalline Si, the device dimensions can be minimized ($L = 180\,\mathrm{nm}$, $W = 220\,\mathrm{nm}$) without degrading the bitline discharge speed. Furthermore, Si exhibits a lower Pelgrom's constant than IGZO, resulting in reduced variability of the discharge curves under process variation.

A Monte Carlo analysis was performed to compare the Si-cascode cell against the full-IGZO baseline (Figure 3.9). After a discharge time of $t_{\mathrm{d}} = 10\,\mathrm{ns}$, the minimum voltage differential ΔV between discharged and non-discharged bitlines is 38.7 mV, which exceeds the 34 mV threshold required by the strong-arm latch sense amplifier. This corresponds to an 8 ns improvement in

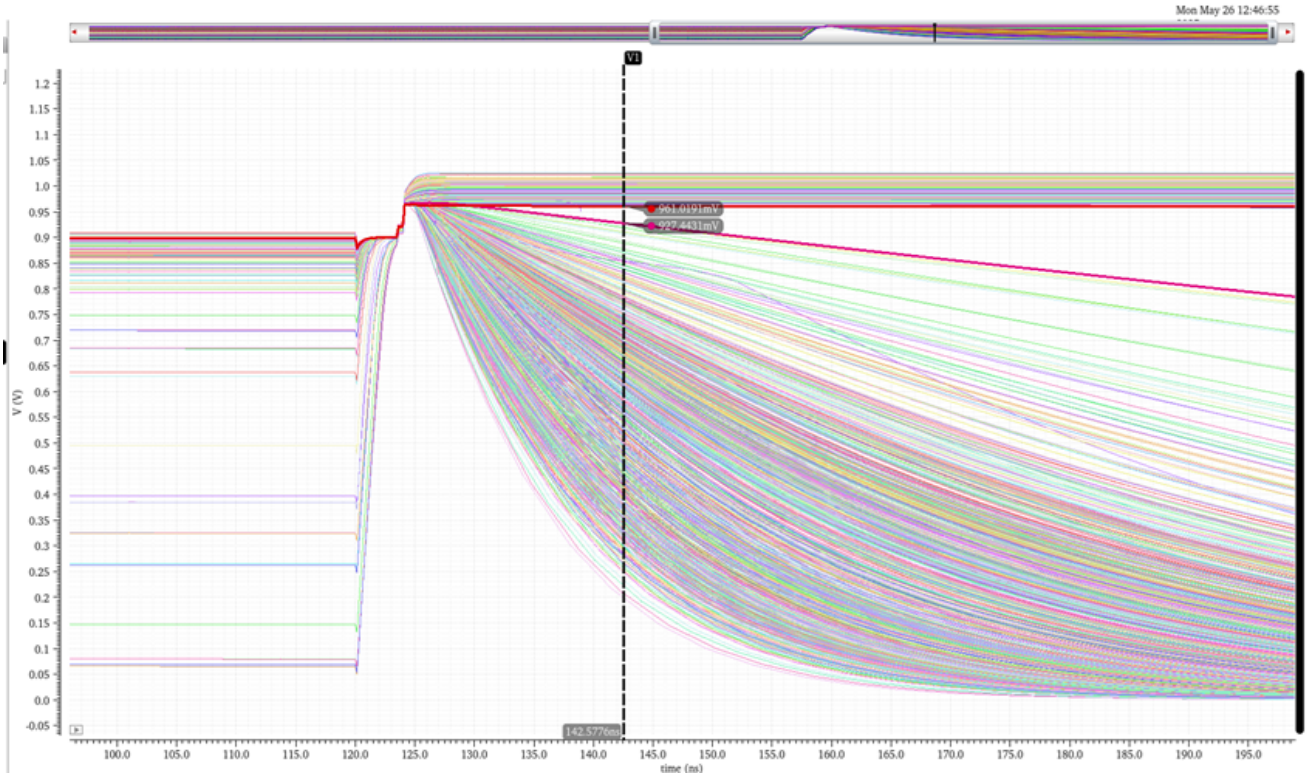

Figure 3.8: Monte Carlo simulation of bitline discharge curves under process variation.

sensing speed compared to the IGZO-only design.

The smaller gate-to-source capacitance of the Si transistor also minimizes parasitic coupling into the storage node, as evidenced in Figure 3.10. When the storage node is at logic "0," the read operation induces virtually no perturbation; when it is at logic "1," only a negligible transient deviation occurs.

From a layout perspective, the Si transistor is fabricated in a bottom layer, with the IGZO transistors M_{W} (write), M_{S} (storage) and M_{C} (cascode/read) stacked above. A full-IGZO cell thus comprises three devices with channel length $L = 1\,\mu\mathrm{m}$ and widths

$$W_{\mathrm{W}} = 4\,\mu\mathrm{m}, \quad W_{\mathrm{S}} = W_{\mathrm{C}} = 6\,\mu\mathrm{m}.$$

By removing the IGZO cascode transistor, the IGZO-plane footprint per cell shrinks by approximately 38 %, at the expense of adding one Si transistor beneath each cell. The trade-off is favorable: read latency is improved by 8 ns and storage-node stability is enhanced, beyond the Si cascode transistor, additional circuit blocks per bit cell can be integrated to perform vector–matrix multiplication or other digital computations, depending on the design of the peripheral logic.

Capability of Performing Bitwise Operations

The bit-wise operation in the 3T0C cell was intended to follow the same procedure as in the 2T0C array: all bit lines are precharged to $V_{\mathrm{DD}}/2$, and the RWL is driven high (1.8 V enabled, -800 mV disabled) to activate two cells in a column simultaneously. In principle, the different combinations of stored bits should yield distinct discharge rates:

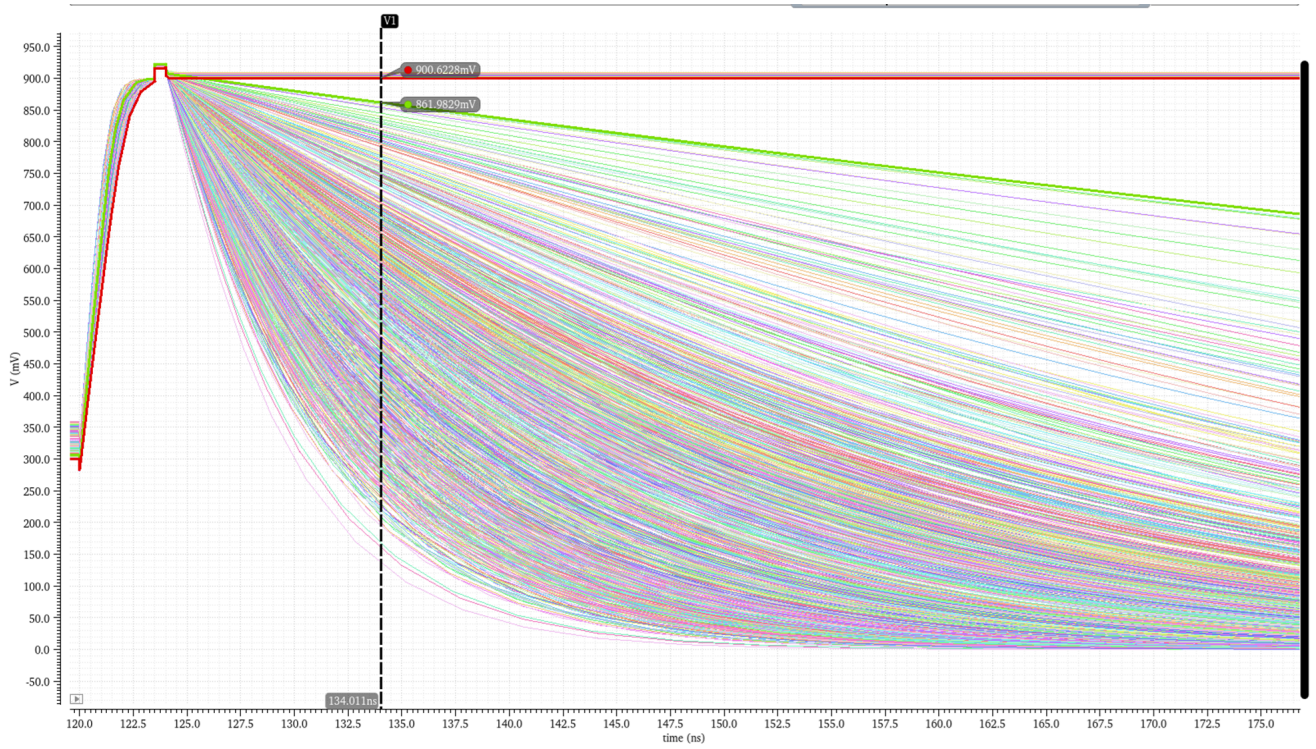

Figure 3.9: Monte Carlo simulation of bitline discharge curves for the Si-cascode cell under process variation.

- **0 + 0:** no discharge,
- **1 + 0:** nominal discharge,
- **1 + 1:** accelerated discharge.

However, in the IGZO and the hybrid 3T0C topology the discharge curves for the $0+1$ and $1+1$ cases overlap continuously, making it impossible to distinguish them at any sensing time. While enlarging the storage transistors could increase the difference in discharge rates, a rough estimate indicates that a size increase of at least $20\times$ would be required, resulting in a prohibitive area overhead.

3.3 Digital 4-T IGZO DRAM Cell

Topology

Figure 3.11 illustrates the schematic of an n-row by m-column array of 4T0C full-IGZO DRAM cells. Each column terminates in a silicon-based sense amplifier (SA). Two cells from different rows but sharing the same bitline are highlighted. Each row is driven by three wordlines: the WWL, the AND wordline (ANDL), and the OR wordline (ORL). The cell comprises four IGZO transistors: a write transistor, a storage transistor, an AND transistor, and an OR transistor. The SN is the parasitic capacitance of the read transistor.

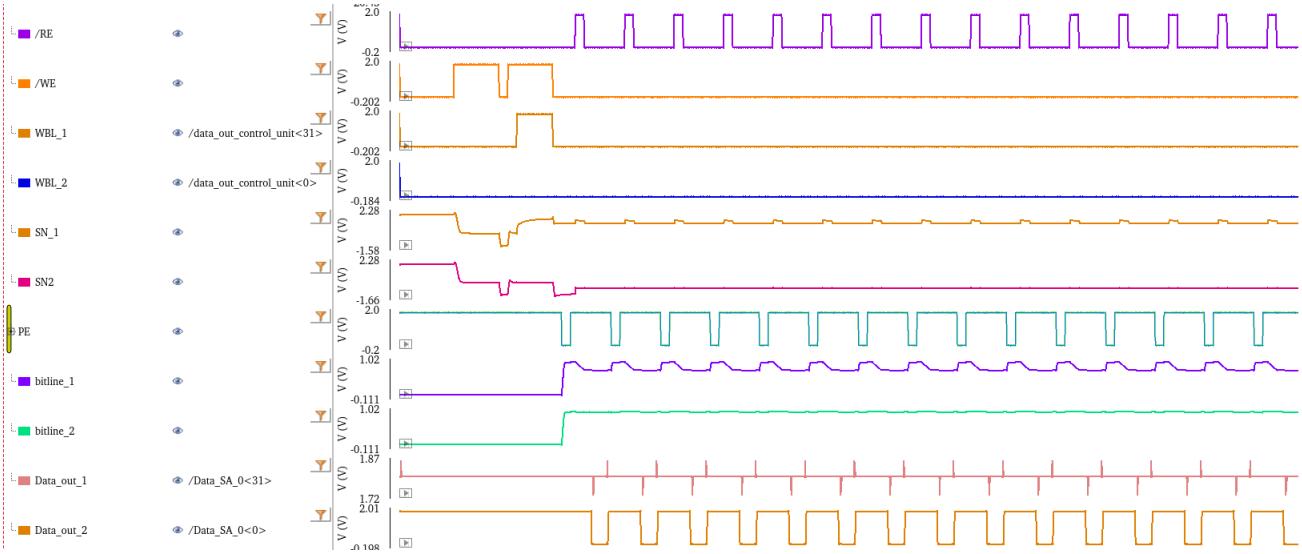

Figure 3.10: Transient disturbance of the storage node due to parasitic coupling when using the Si cascode transistor.

Transistor Sizing

All IGZO transistors use $L = 1\,\mu$m. The write device has $W = 4\,\mu$m to limit area and leakage (off-bias at -800 mV). The storage, OR and AND (cascode) transistors each use $W = 6\,\mu$m, balancing low on-resistance for fast BL discharge with modest area overhead.

In addition to its standard read and write capabilities, the cell can perform the logical OR, AND, NOR, and NAND operations.

Write Operation

The write operation in a 4T0C cell is identical to that of a 3T0C cell; no modifications are required. Due to time constraints, the parasitic coupling within the cell was not investigated. Instead, the focus was solely on the ideal digital interpretation of the logic function.

Read Operation

Similarly, the read operation also follows the exact same procedure as in a 3T0C design. The only additional requirement is that the ORL line must be held permanently off (-800 mV) during reading to ensure that the ORL transistor remains turned off. This guarantees that the sensing path is correctly established without interference from the ORL device.

OR Operation

For an OR computation, ANDL is asserted to (1.8 V), and ORL is driven by the input operand. Discharge of BL occurs if either the stored SN or the input operand is high. Table 3.1 summarizes the resulting outputs, where $X = \mathrm{OR}(A, B)$ and $Y = \overline{X}$.

AND Operation

During an AND computation, ORL is held low (-0.8 V) and ANDL receives the input operand. BL discharges only if both SN and the input operand are high. Table 3.2 presents the outputs,

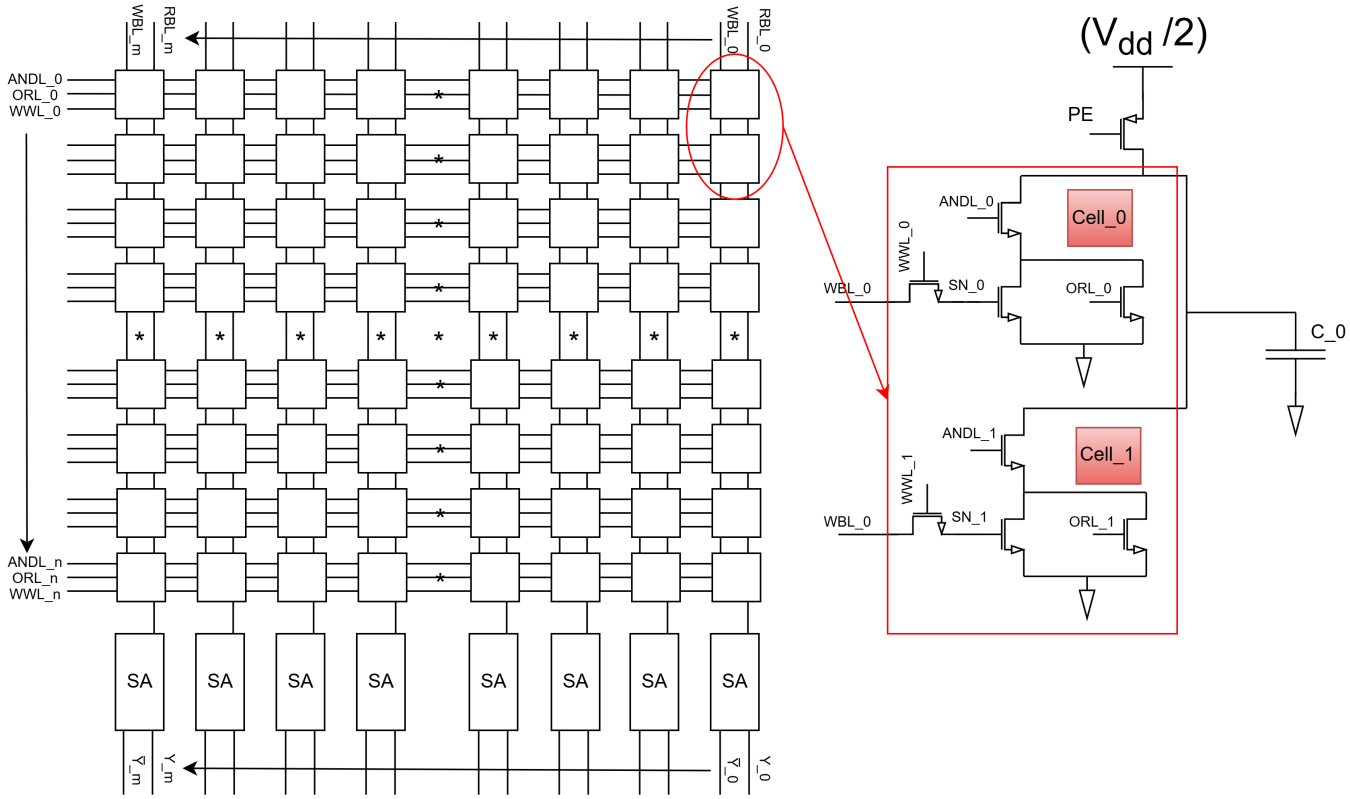

Figure 3.11: Schematic overview of an n-row by m-column array of 4T0C full-IGZO DRAM cells with WWL, ANDL, and ORL wordlines

Table 3.1: Truth table for OR/NOR operation.

A (SN)	B (ORL)	$X = A \vee B$	$Y = \neg X$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

where $X = \mathrm{AND}(A, B)$ and $Y = \overline{X}$.

3.4 Comparison of Variants

Area

No layout was performed, so absolute area values cannot be provided; however, the relative cell footprint can be ranked. The 4T0C variant, comprising four transistors per cell, exhibits the largest area. This is due to the presence of three word lines per row and a relatively large minimum spacing between adjacent lines, which forces the cells to occupy more matrix area. The 3T0C full-IGZO topology, with three transistors per cell and only two word lines per row, occupies less area than the 4T0C variant, resulting in a more compact layout. Finally, the 3T0C hybrid design attains the smallest cell footprint: it replaces the IGZO cascode transistor (600 nm L × 2 µm W) with a silicon cascode device (180 nm L × 280 nm W) fabricated above the IGZO layer and connected via a vertical via. This configuration is expected to reduce the IGZO-layer area by approximately 38 %, albeit at the expense of additional silicon area below the

Table 3.2: Truth table for AND/NAND operation.

A (SN)	B (ANDL)	$X = A \wedge B$	$Y = \neg X$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

cell array. Moreover, alongside the silicon cascode transistor, there is ample room to integrate additional circuitry per bit cell, such as blocks for vector–matrix multiplication or other digital computations, without impacting the reduced IGZO footprint.

Write Speed

The write speed is determined by the series combination of the SN capacitance (i.e., the gate capacitance of the storage transistor) and the on-resistance of the write transistor. Since all three variants employ identical write sequences, the same write-transistor dimensions, and the same SN capacitance, their write speeds are expected to be equivalent.

Read Speed

The read speed depends on the bitline capacitance, the on-resistance of the read transistor, and the on-resistance of the cascode transistor. The 4T0C and 3T0C full-IGZO cells share identical read and cascode transistors, and therefore exhibit the same read latency. In the 3T0C hybrid cell, the IGZO cascode device is replaced by a higher-mobility silicon transistor, which reduces the cascode on-resistance and thereby shortens the read time by approximately 8 ns compared to the 4T0C and full-IGZO 3T0C configurations.

Energy Consumption

Dynamic energy per operation scales with the supply voltage and the operating current. Because all variants use the same supply rails, current levels, and transistor dimensions during both read and write operations, their energy consumptions are expected to be essentially identical.

Retention

Retention time is governed by leakage through the write transistor. As all three cell topologies employ the same IGZO write transistor, their static leakage currents are the same, and thus their retention characteristics are equivalent. The retention time is expected to be $> 400\,\mathrm{s}$.

Because the 3T0C hybrid cell achieves both the smallest IGZO area and the fastest read speed, this variant will be adopted for the implementation of the full IMC-DRAM matrix architecture in the next chapter.

Chapter 4

System Architecture of a Hybrid 3T0C Memory Array

4.1 Introduction

This chapter presents the system architecture of a 32×32 hybrid 3T0C memory array and evaluates its performance against state-of-the-art designs. Section 4.2 provides a high-level architectural overview, including the system block diagram and detailed descriptions of the key silicon sub-blocks: the silicon sense amplifier and the silicon DRAM controller. Section 4.3 describes the evaluation methodology and reports latency, throughput, and energy-per-access metrics. Section 4.4 compares the hybrid 3T0C array with published designs via a benchmark table of key metrics and discusses relative strengths and weaknesses. This organization ensures a clear presentation of both design details and a rigorous assessment of performance trade-offs.

4.2 Architecture Overview

The proposed architecture features a 32 × 32 array of hybrid 3T0C DRAM cells, interfaced to two parallel rows of sense amplifiers. Each row of the array is served by two independent word-line paths, RWL and WWL, enabling row-by-row activation for read and write operations, respectively. Similarly, each column provides separate bit-line paths, RBL and WBL, to transfer data during the active interval of a selected row. The mechanisms for reading and writing follow the procedures detailed in Chapter 3.

Flanking the memory array are two silicon-implemented logic units. The first is a bitwise logic block: upon receipt of two row addresses and an operation code, the DRAM controller simultaneously loads the contents of the two addressed rows into the corresponding sense-amplifier latches, then enables the bitwise unit to perform the specified logical operation and present the result at its output. The second is a multiply-accumulate (MAC) unit, designed to execute dot-product operations between one row of the array—interpreted as up to eight signed 4-bit integers (32 bits total)—and an external input vector whose length is assumed to be less than or equal to the number of elements in the addressed row. Further details on both logic units are provided in Section 4.2.2. Figure 4.1 illustrates a block diagram of the complete system architecture.

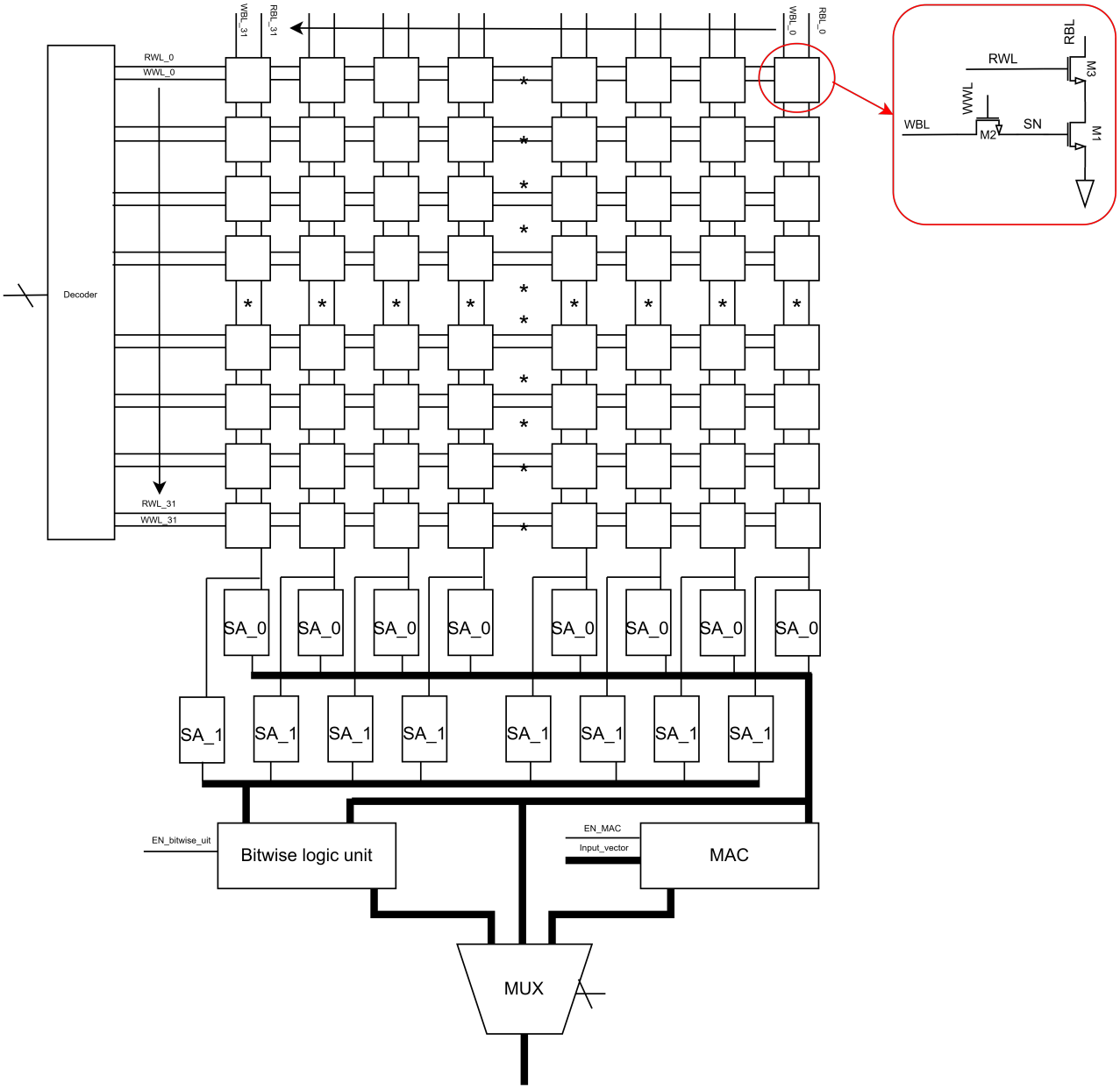

Figure 4.1: A block diagram of the complete system architecture.

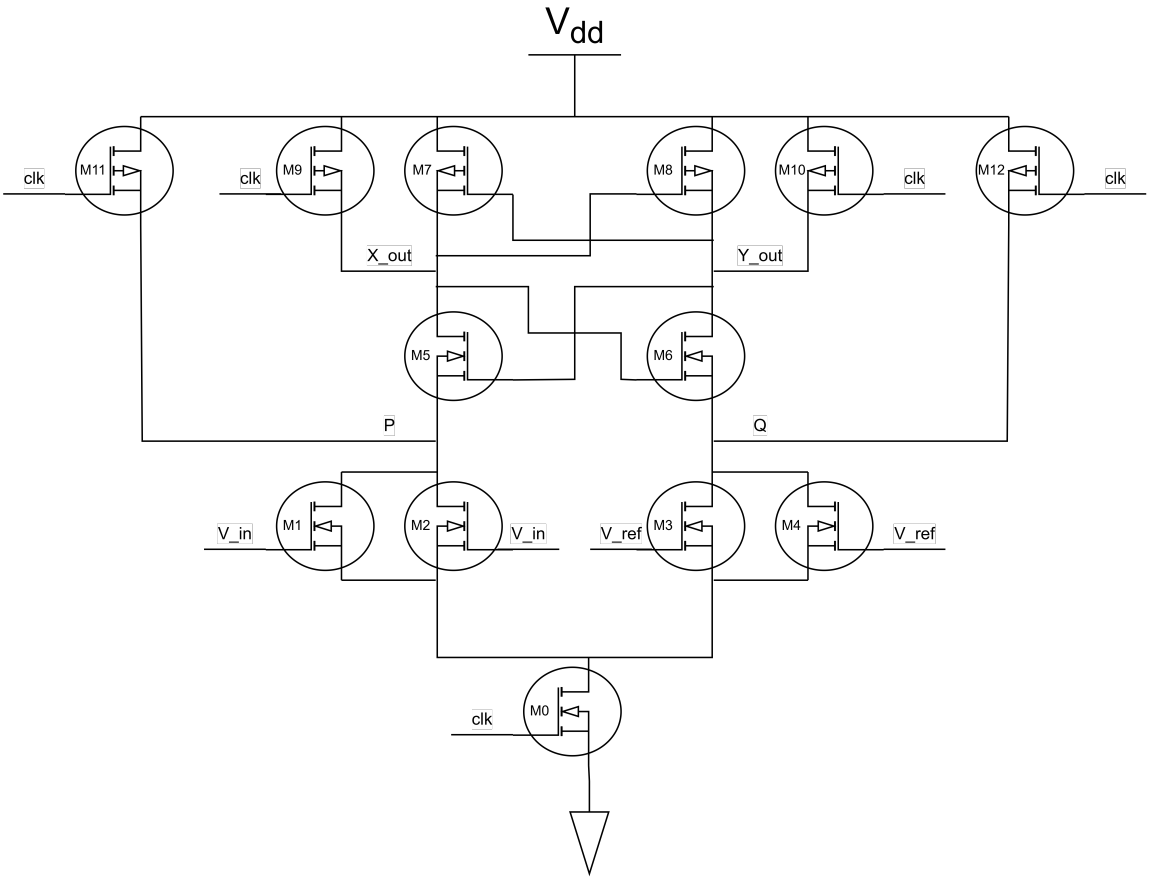

Figure 4.2: The used Strong arm latch SA

4.2.1 Silicon Sense Amplifier

To achieve an optimal sense amplifier (SA) for the DRAM memory architecture, high speed, low power consumption, minimal offset, and small area were pursued. A strong-arm latch SA was selected due to its favorable trade-offs: it operates purely dynamically without static current draw, resulting in low energy consumption, and occupies a compact layout. In the present design, the input-referred offset has been minimized to 17 mV, and the reference voltage V_{REF} is independent of the intrinsic transistor dimensions of the SA. A minimum V_{REF} of 600 mV is required. Figure 4.2 illustrates the SA schematic, while the detailed operation and function of each transistor are described in [31]. The transistor dimensions corresponding to the schematic in Figure 4.2 are listed in Table 4.1.

Figure 4.3 depicts a complete sensing cycle. Upon activation, the SA resolves the small voltage difference between V_{REF} and V_{IN} into complementary digital outputs. The sensing phase requires 2.5 ns, followed by a reset (precharge) phase of 1 ns to prepare the SA for the subsequent operation. Each complete sense–precharge cycle consumes 139 fJ of energy.

4.2.2 Silicon Logic Units

Due to limited development time, two digital logic units were described in VHDL and subsequently synthesized into standard cells using the Genus synthesis tool with a library provided by XFAB. These blocks serve as proof of concept; further optimization of the designs is required. Both units are combined into a single VHDL entity; the complete entity declaration is provided

Table 4.1: Transistor dimensions used in the SA

Transistor(s)	L (nm)	W (μm)
M0	360	1.44
M1–M4	360	1.54
M5–M6	360	1.68
M7–M8	360	0.96
M9–M10	360	2.16
M10–M11	360	0.96

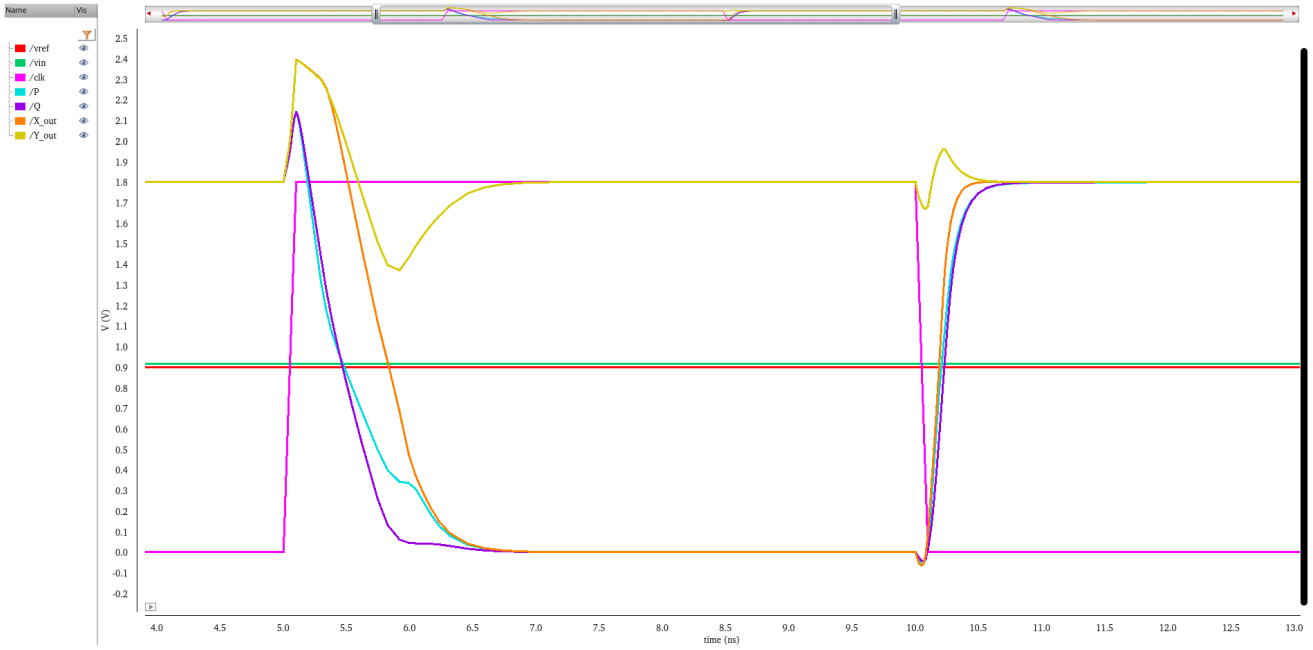

Figure 4.3: A cycle of enabling and precharging the SA

in Appendix A.

Bitwise Logic Unit

The Bitwise Logic Unit accepts four inputs:

1. A 32-bit vector from the first SA row.
2. A 32-bit vector from the second SA row.
3. A 3-bit operand specifying the bitwise operation to be performed.
4. A 1-bit enable signal indicating whether the SA inputs should be forwarded to the logic unit. This enable prevents dynamic evaluation during a normal array read.

When a bitwise operation is requested, the DRAM controller stores the two target rows into the SA rows and asserts the enable signal to activate the Bitwise Logic Unit. The unit then performs the specified bitwise operation on the two 32-bit inputs and produces a 32-bit result.

MAC Logic Block

The Multiply–Accumulate (MAC) Logic Block performs a dot-product operation between two input vectors whose maximum length matches the width of a memory-array row. The intended application is matrix–vector multiplication, where matrix weights (stored as signed 4-bit integers) are organized in consecutive array rows. Since each row can contain up to eight 4-bit values (32 bits total), each row represents one set of eight weights. The input vector is streamed in as a column vector.

For each MAC operation, the DRAM controller:

1. Loads a selected array row into the first SA row.
2. Provides an 8-bit enable mask, where each bit corresponds to one 4-bit element index. If the vector length is shorter than eight, only the required number of elements are precharged and read, reducing energy consumption. When the inputs are connected to the MAC, they pass through this mask so that only the elements up to the specified length propagate; any remaining positions are replaced by zero.
3. Triggers the MAC block, which multiplies each enabled 4-bit weight by the corresponding input element and accumulates the products into a signed result.

This process repeats for each matrix row until all weight rows have been processed. Figure 4.4 illustrates the MAC operation flow. This architecture can be extended to larger arrays, where row widths exceed eight elements, without additional modifications to the MAC logic (besides scaling).

4.2.3 Silicon DRAM Controller

Architecture

The silicon DRAM controller is implemented as a finite-state machine (FSM) that orchestrates all memory operations—read, write, bitwise, copy, and multiply-accumulate—on the hybrid 3T0C array. The controller interfaces to a 5 ns clock (`clk`), a synchronous active-high reset (`rst`), a 4-bit operation code (`operation`), two 5-bit row addresses (`addr_1` and `addr_2`), and a 32-bit data input (`data_in`). For MAC-specific operations, additional inputs specify the number of rows (`num_rows`) and the number of columns (`num_column`) involved in the dot-product. On the output side, the controller drives the read enable (`RE`), write enable (`WE`), and a 32-bit precharge enable bus (`PE_n`, active low) to the memory array, as well as selecting signals for sense-amplifier enables (`SA_0_E`, `SA_1_E`), bitwise logic enable (`enable_bw_rw`), and MAC enable (`enable_mac`). The 32-bit `data_out` bus from the DRAM controller is connected to one input of a two-to-one multiplexer (MUX_0). The second input of MUX_0 is driven by the 32-bit output of the array—that is, the data latched by the sense amplifiers. The `mux_0_sel` signal determines which source (controller or array) is forwarded to the array input during write or copy operations. Specifically, during a normal write, the controller's `data_out` is routed through MUX_0 to the bit lines. During a copy, the data residing in the sense amplifiers (the array's output) is selected by MUX_0 and written back into the destination row.

Upon deassertion of reset, all internal registers initialize to zero and the FSM enters the `IDLE` state with `busy` asserted. In `IDLE`, the controller samples the operation code and loads `addr_1`,

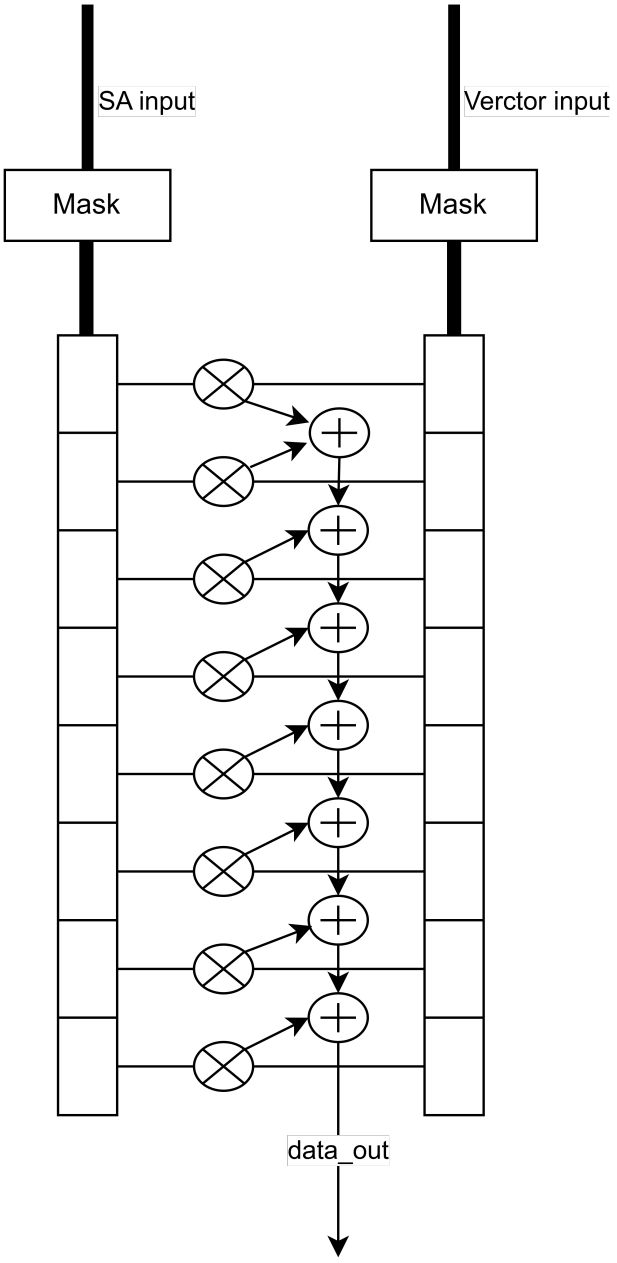

Figure 4.4: The MAC operation flow

addr_2, data_in, num_rows, and num_column into internal registers. Depending on the code, the FSM transitions to one of the following states: WRITE_START, READ_START, READ, bitwise, copy, or MAC. During this transition, busy is deasserted to acknowledge that a new transaction has begun.

Writing

In the WRITE sequence, the FSM transitions to WRITE_START, where WE is asserted and addr_1 is placed onto the address bus. The state then advances to WRITE_DATA: after a two-cycle delay, data_in is driven onto data_out, and WE remains asserted until the tenth cycle of the timer. At that point, WE is deasserted, data_out is cleared, and the FSM finally returns to IDLE.

Reading

In a READ or READ_NOT operation, the FSM first drives PE_n low for one cycle to precharge all bit lines, then releases PE_n high. Immediately thereafter, RE is asserted and addr_1 is placed on the address bus. Two cycles later, RE is deasserted to stop bit-line discharge. On the following clock edge, SA_0_E is enabled to latch the amplified sensed voltage. In the next cycle, mux_1_sel is set to the appropriate operation code (identity for READ, inverted for READ_NOT), and after two more cycles the FSM returns to IDLE.

Bitwise operations

The bitwise state performs two sequential reads using the mechanism described above. First, the data from row addr_1 is latched into the first sense-amplifier row. Next, the data from row addr_2 is latched into the second sense-amplifier row. Finally, mux_1_sel is set to the 3-bit bitwise operation code, enable_bw_rw is asserted for one cycle to execute the specified logic, and the FSM returns to IDLE.

Copy operation

The copy operation begins by reading the source row, latching the sensed data into the first sense-amplifier row. Next, mux_0_sel is set to '1', thereby connecting the array input to the array output (i.e., the latched SA row). A standard write sequence is then performed to the destination row: WE is asserted with addr_2 on the address bus, and after the fixed write latency WE is deasserted. Finally, the FSM returns to IDLE.

Vector–Matrix Multiplication

The MAC state is designed to perform matrix–vector multiplication: the weight matrix is stored in the 32×32 array so that each row of the matrix occupies one row of cells in the DRAM, and the input vector is supplied to the MAC unit. During each MAC microsequence, one row of weights (up to num_column signed 4-bit values) is read from the array and a dot product is computed with the corresponding input vector elements. To minimize dynamic energy, only the bit lines corresponding to active columns are precharged: PE_n is driven low for those bits and remains high for inactive bits. Next, RE is asserted to read the selected row into SA_0_E. On the following clock cycle, mux_1_sel is set to the MAC operation code and enable_mac is asserted with a

mask reflecting the number of active columns. The MAC unit then multiplies each 4-bit weight by the corresponding input element and accumulates the partial sums internally. Once that microsequence completes, `SA_0_E` and `enable_mac` are deasserted. If `row_counter` < `num_rows` − 1, the counter increments and the next row is processed; otherwise, the FSM deasserts `busy` and returns to `IDLE`.

The result of a read, bitwise, or MAC operation appears on `data_out` after the corresponding `mux_1_sel` selection. A final `busy` assertion ensures that no new command is accepted until the current operation completes, thus preserving data integrity and synchronizing with host logic.

The DRAM controller is implemented as a single VHDL entity; its full entity declaration can be found in Appendix B.

4.3 Performance Results

Throughout all states, the 5-bit `timer` register enforces precise multi-cycle timing for each control signal. With a 5 ns clock period, the `READ` sequence spans ten clock cycles (from asserting `PE_n` to latching the sense-amplifier output), for a total of $10 \times 5\,\text{ns} = 50\,\text{ns}$. A complete `READ` operation consumes 116 pJ, of which 108 pJ is expended by the DRAM controller; the remaining 8 pJ covers the sense amplifier, decoders, and bit-line precharge overhead. The `WRITE` sequence requires eleven cycles (one for `WRITE_START` plus ten in `WRITE_DATA`), totaling $11 \times 5\,\text{ns} = 55\,\text{ns}$. A full `WRITE` operation uses 131 pJ, almost entirely consumed by the DRAM controller (no sense amplifiers, MAC, or bitwise units are activated during this period).

A `bitwise` operation takes 85 ns and consumes 232 pJ in total; 223 pJ of that energy is drawn by the DRAM controller itself. Each dot-product (MAC) operation takes 55 ns and costs 144 pJ, where 108 pJ is spent by the controller and the remaining 36 pJ covers the MAC logic, sense amplifier, and decoder activity.

Table 4.2 presents the throughput in millions of operations per second (MOPS) and the corresponding energy efficiency (MOPS/W) for each NMC memory operation.

Table 4.2: Throughput and energy efficiency for various operations (in MOPS)

Operation	MOPS	GOPS/W
BITWISE	11.76	4.310
MAC	290.9	111.1

4.4 Comparison with State of the Art

As shown in Table 4.3, the write and read latencies, along with their corresponding energy consumptions, are compared between this work and the studies by [19] and [20].

The row-refresh operation requires 131 pJ of energy and occupies 55 ns, and it is performed once every 400 s. By comparison, a refresh in a pure silicon implementation requires 64 ms [20]. As a result, each refresh interval achieves an energy saving of approximately 818.6 nJ.

Table 4.3: Comparison to the state of Art

Metric	Giterman *et al.* [19]	Ryu *et al.* [20]	This work
Write access time	1.3 ns	9.1 μs	55 ns
Write Energy	N/A	N/A	131 pJ
Read time	25 ns	9.1 μs	50 ns
Read Energy	N/A	N/A	116 pJ
Retention	< 0.8 ms	> 1000 s	> 400 s

Discussion: Strengths and Weaknesses

By integrating IGZO storage and write transistors, the bitcell's retention time is now expected to exceed > 400 s. Compared to a silicon-only implementation, this represents an improvement by a factor of approximately 6451, which translates into significant energy savings—a major strength of the proposed design. Additionally, the inclusion of a silicon cascode transistor above the two IGZO transistors accelerates the read process by roughly 8 ns and reduces capacitive coupling, since its gate-to-source capacitance is minimized through the adoption of minimum-sized devices. This cascode configuration also mitigates discharge variations and, by enabling a more compact IGZO layer, reduces its IGZO footprint by approximately 38%.

One drawback of introducing a silicon transistor is the additional silicon area required below the IGZO layer. To address this, peripheral logic circuits—including the DRAM controller, sense amplifier, and decoders—are manufactured in 180 nm silicon technology and placed adjacent to the array on the silicon layer (below the reduced-area IGZO layer). Implementing these functions in silicon accelerates both read and write operations: in a purely IGZO architecture, access times can reach up to 9.1 μs, whereas a silicon-only design achieves 1.3 ns for writes and 25 ns for reads. In our hybrid architecture, the total access times are 55 ns for writes and 50 ns for reads.

When comparing the throughput and energy efficiency of the proposed design with those reported in Sections 2.4 and 2.5, it is evident that our implementation achieves on the order of GOPS/W, whereas state-of-the-art accelerators attain TOPS/W and higher raw performance. This discrepancy can be attributed to three primary factors:

1. **Technology node and peripheral overheads.** The multiplier–accumulator (MAC) array and DRAM controller in our design consume up to 85 % of the total energy, and are implemented in a 180 nm process, compared with 28 nm or smaller technologies used by leading designs.

2. **High-level synthesis inefficiencies.** Both the DRAM controller and MAC units were described in VHDL and synthesized with Cadence Genus, resulting in suboptimal area–power trade-offs; a bespoke RTL implementation could yield significant energy reductions.

3. **Lack of in-memory compute.** Whereas state-of-the-art architectures perform vector–matrix multiplications directly within the memory array—activating the sensing amplifiers once for the entire operation—our system must read and process each row sequentially, incurring additional data-transfer energy.

To close the energy-efficiency gap, the logic and memory peripherals will be migrated to a more advanced technology node, reducing both dynamic and leakage power. In addition, by adding

supplementary blocks per cell adjacent to the silicon–cascode transistor, the design can integrate dedicated vector–matrix multiplication units or other digital compute modules directly at the bit-cell level, without enlarging the reduced IGZO footprint. This approach is expected to further enhance overall throughput and energy efficiency.

This heterogeneous approach also enables efficient compute-near-memory: By placing digital logic blocks in close proximity to the array, read operations can be performed and subsequent logic operations carried out near the memory without incurring a high refresh rate—only a single refresh is required every 400 s. In essence, data can be written once and subsequently accessed and processed rapidly and energy-efficiently right next to the memory array.

Chapter 5

Broader Economic, Societal, and Sustainability Implications

5.1 Economic Aspects

Although the dual-layer IGZO–Si 3T0C DRAM incurs higher fabrication cost, its ultra-long retention ($\sim$ 400 s vs. $\sim$ 64 ms) reduces overall energy consumption dramatically. In large AI-centric data centers, where memory can represent 40–50% of hardware cost and 30–40% of power draw, eliminating most refresh cycles yields substantial savings in energy and cooling, thereby reducing total cost of ownership despite a higher per-chip price [32], [33]. Furthermore, in-memory bitwise and MAC operations cut data-movement overhead, improving performance-per-watt and further enhancing economic value.

5.2 Societal Implications

By lowering the energy barrier for AI workloads, IGZO–Si DRAM can help democratize compute-intensive analytics, narrowing today's "AI divide" [34]. Edge-deployed AI for healthcare diagnostics or personalized education becomes more feasible in resource-constrained settings when memory power and cooling demands are minimized. This aligns with the "Green AI" movement: reducing carbon footprint in training and inference makes ethical, sustainable AI more accessible [34], [33].

5.3 Sustainability Implications

Operationally, the design's drastic energy reduction contributes directly to lower data-center emissions (data centers now consume 1–2% of global electricity and rising) [33]. However, the IGZO layer relies on indium—a scarce byproduct of zinc mining with limited recycling ($\approx 10 - 15\%$), posing material-supply and end-of-life challenges [35]. Mitigating this requires improved indium recovery and circular-economy strategies. On the positive side, reduced power dissipation lowers average device temperature, potentially doubling module lifetime for every 10 °C reduction in operating temperature [36], and IGZO endurance tests exceed 10^{11} cycles without degradation [37]. Together, these factors suggest that IGZO–Si DRAM can deliver a net

sustainability win, if coupled with responsible material sourcing and recycling.

Chapter 6

Conclusion and Future Work

Conclusion

In this work, we have presented a hybrid IGZO–Si 3T0C DRAM architecture that leverages the ultra-low leakage of indium–gallium–zinc–oxide (IGZO) storage transistors together with high-speed silicon peripherals. By combining an IGZO write transistor, an IGZO storage transistor, and a minimum-sized silicon cascode transistor ($L = 180\,\text{nm}$, $W = 220\,\text{nm}$), we achieve a bitcell retention time exceeding 400 s, which is approximately 6451× longer than a comparable silicon-only 3T0C DRAM cell. This dramatic increase in retention directly translates into substantial energy savings, since refresh operations can be performed only once every 400 s instead of every 64 ms (in pure silicon).

The silicon cascode device also accelerates read access: swapping an IGZO cascode for the minimum-sized Si transistor reduces read latency by roughly 8 ns (from 58 ns in the IGZO-only cell to 50 ns). Moreover, the smaller gate-to-source capacitance of the silicon device mitigates parasitic coupling into the storage node, thereby preserving data integrity and reducing discharge variability under process variation. From a layout standpoint, integrating the Si cascode below the two IGZO transistors reduces the IGZO layer's area by approximately 38%, at the cost of additional footprint in the silicon plane.

Peripheral logic—including the DRAM controller, the strong-arm latch sense amplifiers, decoders, and compute units—was implemented in 180 nm silicon. This heterogeneous approach yields access times of 55 ns for writes and 50 ns for reads, with energy per access of 131 pJ (write) and 116 pJ (read). By comparison, a purely IGZO 2T0C cell exhibits 9.1 μs for both read and write, while a silicon-only 3T0C cell achieves 1.3 ns (write) and 25 ns (read). Thus, our hybrid design strikes a balance: retaining most of IGZO's static-power advantages without sacrificing a large throughput.

We also demonstrated near-memory computing capabilities by embedding two digital logic blocks adjacent to the array. A 32-bit bitwise unit performs Boolean operations on two entire rows within 85 ns and 232 pJ, and a multiply–accumulate (MAC) unit implements an 8-element dot product in 55 ns and 144 pJ. Because only one refresh is needed every 400 s, data can be written once and subsequently accessed or computed upon with minimal energy overhead—enabling efficient compute-near-memory (CNM) operation.

In summary, this thesis demonstrates that a hybrid IGZO–Si 3T0C DRAM cell can:

- Extend retention to $> 400\,\text{s}$, reducing refresh energy by more than three orders of magnitude compared to silicon.
- Achieve 50ns/55ns read/write access times, competitive with purely silicon designs.
- Realize bitwise and MAC operations near the memory array.
- Reduce IGZO layer area by $\approx 38\%$ through 3D stacking of the Si cascode below the IGZO cell.

These results confirm that the hybrid IGZO–Si 3T0C architecture is a promising approach for low-power, high-throughput in-memory computing.

Future Work

While the benefits of the hybrid IGZO–Si 3T0C DRAM architecture are clear, several avenues remain to strengthen and extend this research:

1. **Array Scaling and Layout Optimization.**
   - *Large-scale arrays:* Evaluate the performance, yield, and variability when scaling beyond 32×32 to kilobit or megabit arrays. Assess the impact of increased interconnect capacitance, IR drop, and bitline parasitics on read/write latency and energy.
   - *3D integration:* Explore fabricating a monolithic 3D stack where multiple IGZO layers are vertically integrated with silicon logic. Investigate TSV (through-silicon via) or BEOL (back-end-of-line) technologies to reduce inter-layer routing overhead and latency.
   - *Layout cell libraries:* Develop a detailed standard-cell library for the IGZO–Si hybrid cell, including optimized placement of the Si cascode via local interconnects and minimized metal layers.
2. **Advanced Sense Amplifier and Peripheral Design.**
   - *Low-offset, low-power sensing:* Further reduce the input-referred offset of the strong-arm latch SA (currently $17\,\text{mV}$) via careful transistor sizing or offset-cancellation techniques. Lower offset would allow shorter discharge times, improving read latency below $50\,\text{ns}$.
   - *Adaptive biasing:* Implement adaptive bias circuits that adjust sense-amp precharge levels or thresholds based on temperature and process corners to optimize both speed and energy.
   - *Column-level parallelism:* Design multi-bank/multi-bank column architectures that permit simultaneous activation of multiple columns, boosting aggregate throughput for bulk reads or CNM tasks.
3. **Enhanced Near-Memory Compute (CNM) Architectures.**

- *Custom logic primitives:* Explore adding lightweight analog compute primitives—such as bitline-level charge summation or majority gates—within the IGZO array to accelerate common bitwise or approximate computing kernels.
- *Per-bit logic blocks adjacent to Si cascode:* Implement a small logic block per bit next to the Si cascode transistor (e.g., a pull-up or pull-down network driven by the cell value). This would enable analog matrix-vector multiplication across the entire array in a single operation.
- *Programmable logic:* Design a small reconfigurable array (e.g., a coarse-grained reconfigurable fabric) in silicon atop or beside the memory array to support a wider range of CNM tasks (e.g., filtering, encryption) with minimal data movement.
- *Compiler and ISA support:* Develop compiler-driven instruction extensions for a host CPU to offload CNM tasks (e.g., bitwise CNN layers) automatically to the hybrid IGZO–Si memory macro, abstracting the low-level control signals.

4. **Application-Level Evaluation.**
   - *Workload benchmarks:* Simulate end-to-end performance and energy on target applications (e.g., sparse matrix–vector multiplication, bitwise neural networks, graph analytics) to quantify system-level gains over conventional DRAM + CPU baselines.
   - *Co-design with accelerators:* Investigate coupling the hybrid DRAM macro with emerging accelerators (e.g., RISC-V cores, domain-specific ASICs) in a system-on-chip (SoC) to measure holistic area, power, and throughput trade-offs.
   - *Security and data retention:* Examine how the ultra-long retention and analog-like storage node might enable new threat models or side-channel vulnerabilities, and propose mitigation strategies (e.g., row scrambling, refresh randomization).
5. **Alternative Device Materials and Architectures.**
   - *Emerging TFT semiconductors:* Study alternative oxide semiconductors (e.g., IGTO, ZnO) or 2D materials (e.g., MoS_2) for the storage transistor to further increase retention or reduce write energy.
   - *Multilevel storage:* Investigate multilevel storage using analog programming of the IGZO storage node for high-density in-memory computing, trading off retention margin for increased bit density.

By pursuing these directions, the hybrid IGZO–Si 3T0C DRAM can evolve from a proof-of-concept macro to a robust, manufacturable technology for next-generation, energy-efficient in-memory computing systems.

Bibliography

[1] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *ACM SIGARCH Comput. Archit. News*, vol. 23, pp. 20–24, Mar. 1995.

[2] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory," *ACM SIGARCH Comput. Archit. News*, vol. 44, pp. 27–39, Jun 2016.

[3] V. Seshadri and O. Mutlu, "The Processing Using Memory Paradigm: In-DRAM Bulk Copy, Initialization, Bitwise AND and OR," *CoRR*, vol. abs/1610.09603, Oct 2016.

[4] S. Kim and H.-J. Yoo, "An overview of computing-in-memory circuits with dram and nvm," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 71, pp. 1626–1631, Mar. 2024.

[5] A. Belmonte and H. Oh, "Capacitor-less, long-retention (>400 s) dram cell paving the way towards low-power and high-density monolithic 3d dram," in *2020 IEEE International Electron Devices Meeting (IEDM)*, pp. 4.2.1–4.2.4, 2020.

[6] T. Kamiya, K. Nomura, and H. Hosono, "Present Status of Amorphous In–Ga–Zn–O Thin-Film Transistors," *Science and Technology of Advanced Materials*, vol. 11, p. 044305, Apr 2010.

[7] J. L. Hennessy, D. A. Patterson, and A. C. Arpaci-Dusseau, *Computer Architecture: A Quantitative Approach*. San Francisco, CA: Morgan Kaufmann, 6 ed., 2019.

[8] M. He, C. Song, I. Kim, C. Jeong, S. Kim, I. Park, M. Thottethodi, and T. N. Vijaykumar, "Newton: A dram-maker's accelerator-in-memory (aim) architecture for machine learning," in *Proceedings of the 53rd IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 372–385, 2020.

[9] G. Singh, L. Chelini, S. Corda, A. J. Awan, S. Stuijk, R. Jordans, H. Corporaal, and A.-J. Boonstra, "Near-memory computing: Past, present, and future," *Microprocessors and Microsystems*, vol. 71, p. 102868, Nov. 2019.

[10] K. H. Cherenack, A. Z. Kattamis, B. Hekmatshoar, J. C. Sturm, and S. Wagner, "Amorphous-Silicon Thin-Film Transistors Fabricated at 300 °C on a Free-Standing Foil Substrate of Clear Plastic," *IEEE Electron Device Lett.*, vol. 28, pp. 1004–1006, Nov 2007.

[11] M. J. Mirshojaeian Hosseini and R. A. Nawrocki, "A review of the progress of thin-film transistors and their technologies for flexible electronics," *Micromachines*, vol. 12, no. 6, p. 655, 2021.

[12] J. B. Choi, D. C. Yun, Y. I. Park, and J. H. Kim, "Properties of hydrogenated amorphous silicon thin-film transistors fabricated at 150 °C," *Journal of Non-Crystalline Solids*, vol. 266–269, pp. 1315–1319, May 2000.

[13] S. M. Venugopal, *Flexible Active Matrix Displays and Integrated Amorphous Silicon Source Drivers*. PhD thesis, Arizona State University, 2007.

[14] K. Nomura, H. Ohta, A. Takagi, T. Kamiya, M. Hirano, and H. Hosono, "Room-temperature fabrication of transparent flexible thin-film transistors using amorphous oxide semiconductors," *Nature*, vol. 432, no. 7016, pp. 488–492, 2004.

[15] T. Sameshima, S. Usui, and M. Sekiya, "Xecl excimer laser annealing used to fabricate poly-si tfts," in *MRS Online Proceedings Library*, vol. 71, pp. 435–440, 1986.

[16] K. Sera, H. Hosoya, O. Sugiura, and M. Matsumura, "High-performance tfts fabricated by xecl excimer laser annealing of hydrogenated amorphous-silicon film," *IEEE Transactions on Electron Devices*, vol. 36, no. 12, pp. 2868–2872, 1989.

[17] S. Uchikoga, "Low-temperature polycrystalline silicon thin-film transistor technologies for system-on-glass displays," *MRS Bulletin*, vol. 27, no. 11, pp. 881–886, 2002.

[18] F.-C. Hsu, R. J. Huang, C.-H. Chang, R.-P. Tsay, J.-H. Chang, and I.-W. Huang, "New 1T1C and 3T0C Cells in the 3D X-DRAM Family: Advancing 3D NAND-like DRAM Technology Using IGZO," white paper, NEO Semiconductor, May 2025.

[19] R. Giterman, A. Teman, P. Meinerzhagen, L. Atias, A. P. Burg, and A. Fish, "Single-supply 3t gain-cell for low-voltage low-power applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 1, pp. 358–362, 2016.

[20] S. Ryu, M. Kang, K. Cho, and S. Kim, "Capacitorless Two-Transistor Dynamic Random-Access Memory Cells Comprising Amorphous Indium–Tin–Gallium–Zinc Oxide Thin-Film Transistors for the Multiply–Accumulate Operation," *Adv. Mater. Technol.*, vol. 9, p. 2302209, Aug 2024.

[21] S. Liu, S. Wei, P. Yao, D. Wu, L. Jie, S. Pan, J. Tang, B. Gao, H. Qian, and H. Wu, "A 28 nm 576 kbit rram-based computing-in-memory macro featuring hybrid programming with area efficiency of 2.82 tops/mm^2," *Journal of Semiconductors*, vol. 46, no. 6, 2025.

[22] W. Ye, C. Dou, L. Wang, Z. Zhou, J. An, W. Li, H. Gao, X. Xu, J. Yue, J. Yang, J. Liu, D. Shang, J. Tian, Q. Liu, and M. Liu, "A 28 nm Hybrid 2T1R RRAM Computing-in-Memory Macro for Energy-efficient AI Edge Inference," in *Proceedings of the 2022 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, Nov. 2022.

[23] C.-X. Xue, J.-M. Hung, H.-Y. Kao, Y.-H. Huang, S.-P. Huang, F.-C. Chang, P.-C. Chen, T.-W. Liu, C.-J. Jhang, C.-I. Su, W.-S. Khwa, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, Y.-D. Chih, and T.-Y. Chang, "A 22 nm 4 mb 8 b-precision reram computing-in-memory macro with 11.91 – 195.7 tops/w for tiny ai edge devices," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 246–247, Feb 2021.

[24] J.-H. Yoon, M. Chang, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, and A. Raychowdhury, "A 40 nm, 64 kb, 56.67 tops/w voltage-sensing computing-in-memory/digital rram macro

supporting iterative write with verification and online read-disturb detection," *IEEE Journal of Solid-State Circuits*, vol. 57, pp. 68–79, Jan 2022.

[25] S. Jung, D. Ham, S. J. Kim, and et al., "A crossbar array of magnetoresistive memory devices for in-memory computing," *Nature*, vol. 602, pp. 582–588, Feb 2022.

[26] P. Deaville, B. Zhang, and N. Verma, "A 22 nm 128 kb mram row/column-parallel in-memory computing macro with memory-resistance boosting and multi-column adc readout," in *2022 IEEE Symposium on VLSI Technology and Circuits*, pp. 268–269, Jun 2022.

[27] M. Caon, C. Choné, P. D. Schiavone, A. Levisse, G. Masera, M. Martina, and D. Atienza, "Scalable and RISC-V Programmable Near-Memory Computing Architectures for Edge Nodes," *IEEE Transactions on Emerging Topics in Computing*, vol. 13, pp. 1–15, Jan 2025.

[28] Y.-C. Guo, T.-S. Chang, C.-S. Lin, B.-C. Chiou, C.-M. Lai, S.-S. Sheu, W.-C. Lo, and S.-C. Chang, "CIMR-V: An End-to-End SRAM-based CIM Accelerator with RISC-V for AI Edge Device," in *Proceedings of the 2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2024.

[29] S. Yan, Z. Cong, Z. Wang, Z. Dai, Z. Guo, Z. Qian, X. Li, X. Zheng, C. Chen, N. Lu, C. Dou, G. Yang, X. Xu, D. Geng, J. Yue, L. Wang, L. Li, and M. Liu, "A monolithic 3d igzo-rram-sram-integrated architecture for robust and efficient compute-in-memory enabling equivalent-ideal device metrics," *Science China Information Sciences*, vol. 68, pp. 122404:1–122404:16, Feb 2025.

[30] L. Zheng, Z. Wang, Z. Lin, and M. Si, "The Impact of Parasitic Capacitance on the Memory Characteristics of 2T0C DRAM and New Writing Strategy," *IEEE Electron Device Letters*, vol. 44, Aug 2023.

[31] B. Razavi, "The strongarm latch: A circuit for all seasons," *IEEE Solid-State Circuits Magazine*, vol. 7, pp. 12–17, June 2015.

[32] D. Patel, "Ai server cost analysis – memory is the biggest loser." SemiAnalysis (online), 2023. Accessed 2025-06-09.

[33] S. Chen, "Data centers will use twice as much energy by 2030—driven by ai." Scientific American, Apr. 10, 2025. Accessed 2025-06-09.

[34] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," *Proc. 57th Annual Meeting of the ACL*, pp. 3645–3650, 2019.

[35] H. U. Sverdrup, O. Van Allen, and H. V. Haraldsson, "Modeling indium extraction, supply, price, use and recycling 1930–2200 using the world7 model," *Natural Resources Research*, vol. 32, pp. 2285–2313, 2023.

[36] R. Wilcoxon, "Does a 10 °c increase in temperature really reduce the life of electronics by half?." Electronics Cooling Magazine, Aug. 2017. Accessed 2025-06-09.

[37] A. Belmonte, B. Govoreanu, S. Sutar, K. Barla, and *et al.*, "Tailoring igzo–tft architecture for capacitorless dram, demonstrating $> 10^3$ s retention and $> 10^{11}$ cycles endurance," in *2021 IEEE International Electron Devices Meeting (IEDM)*, pp. 10.6.1–10.6.4, 2021.

Appendix A

Appendix - VHDL Implementations of the Multiply–Accumulate and Bitwise Functional Units

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity MAC is
    Port (
        Data_SA_0   : in  std_logic_vector(31 downto 0);
        Data_N_SA_0 : in  std_logic_vector(31 downto 0);
        Data_SA_1   : in  std_logic_vector(31 downto 0);
        Data_N_SA_1 : in  std_logic_vector(31 downto 0);
        Data_vector : in  std_logic_vector(31 downto 0);
        sel         : in  std_logic_vector(2 downto 0);
        enable      : in  std_logic;
        enable_mac  : in  std_logic_vector(7 downto 0);
        Data_Out    : out std_logic_vector(31 downto 0)
    );
end MAC;

architecture Behavioral of MAC is

begin
    process(Data_SA_0,Data_N_SA_0 , Data_SA_1 , Data_N_SA_1, Data_vector,
sel, enable , enable_mac)

        variable sum      : signed(7 downto 0);
        variable a        : signed(3 downto 0);
        variable b        : signed(3 downto 0);
        variable prod     : signed(7 downto 0);

    begin
        if enable = '1' then
            case sel is
                when "000" => Data_Out <= Data_SA_0 xor Data_SA_1;
                when "001" => Data_Out <= Data_SA_0 xnor Data_SA_1;
                when "010" => Data_Out <= Data_SA_0;
                when "011" => Data_Out <= Data_N_SA_0;
                when "100" => Data_Out <= Data_SA_0 or Data_SA_1;
                when "101" => Data_Out <= Data_SA_0 nor Data_SA_1;
                when "110" => Data_Out <= Data_SA_0 nand Data_SA_1;
                when "111" => Data_Out <= Data_SA_0 and Data_SA_1;
                when others => Data_Out <= (others => '0');
            end case;
        elsif enable_mac /= "00000000" then
            -- Multiply-and-Accumulate over 8 signed 4-bit elements
            sum := (others => '0');
            for i in 0 to 7 loop
                if enable_mac(i) = '1' then
                    a    := signed(Data_SA_0((i*4)+3 downto i*4));
                    b    := signed(Data_vector((i*4)+3 downto i*4));
                    prod := a * b;
                    sum  := sum + prod;
                end if;
            end loop;
            -- Extend sum (8 bits) to 32 bits
            Data_Out <= std_logic_vector(resize(sum, 32));
        else
            Data_Out <= (others => '0');
        end if;
```

```
    end process;
end Behavioral;
```

Appendix B

Appendix - VHDL Implementations of the DRAM controller Functional Unit

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity dram_controller_MAC_bitwise is
    Port (
        clk           : in  std_logic;               -- 5 ns clock
        rst           : in  std_logic;               -- synchronous
reset, active high
        operation     : in  std_logic_vector(3 downto 0);
            -- 0000: IDLE
            -- 0001: WRITE
            -- 0010: READ
            -- 0011: READ_NOT
            -- 0100: OR
            -- 0101: NOR
            -- 0110: NAND
            -- 0111: AND
            -- 1000: XOR
            -- 1001: NXOR
            -- 1101: copy
            -- 1011: MAC
        data_in       : in  std_logic_vector(31 downto 0);
        addr_1        : in  std_logic_vector(4 downto 0);
        addr_2        : in  std_logic_vector(4 downto 0);

        -- MAC specific signals
        num_rows      : in  std_logic_vector(4 downto 0);
        num_column    : in  std_logic_vector(4 downto 0);
        enable_bw_rw  : out std_logic;
        enable_mac    : out std_logic_vector(7 downto 0);
        -- control outputs
        RE            : out std_logic;
        WE            : out std_logic;
        PE_n          : out std_logic_vector(31 downto 0);
-- precharge enable, active low
        data_out      : out std_logic_vector(31 downto 0);
        mux_0_sel     : out std_logic;               -- select bitwise
output mux
        mux_1_sel     : out std_logic_vector(3 downto 0);
        SA_0_E        : out std_logic;
        SA_1_E        : out std_logic;
        addr_bus      : out std_logic_vector(4 downto 0);
        busy          : out std_logic
    );
end dram_controller_MAC_bitwise;

architecture Behavioral of dram_controller_MAC_bitwise is
    type state_type is (
        IDLE,
        WRITE_START, WRITE_DATA,
        READ_START, READ,
        COPY,
        bitwise,
        MAC
    );
    signal state            : state_type;
```

```
    signal timer             : unsigned(4 downto 0);

    signal RE_reg            : std_logic;
    signal WE_reg            : std_logic;
    signal PE_n_reg          : std_logic_vector(31 downto 0);
-- precharge enable, active low
    signal data_out_reg      : std_logic_vector(31 downto 0);
    signal mux_0_sel_reg     : std_logic;               -- select bitwise
output mux
    signal mux_1_sel_reg     : std_logic_vector(3 downto 0);
    signal SA_0_E_reg        : std_logic;
    signal SA_1_E_reg        : std_logic;
    signal addr_bus_reg      : std_logic_vector(4 downto 0);
    signal enable_bw_rw_reg  : std_logic;
    signal enable_mac_reg    : std_logic_vector(7 downto 0);

    signal operation_reg     : std_logic_vector(3 downto 0);
    signal num_rows_reg      : std_logic_vector(4 downto 0);
    signal num_column_reg    : std_logic_vector(4 downto 0);
    signal data_in_reg       : std_logic_vector(31 downto 0);
    signal addr_1_reg        : std_logic_vector(4 downto 0);
    signal addr_2_reg        : std_logic_vector(4 downto 0);

    signal row_counter       : unsigned(4 downto 0);
    signal mac_stage         : integer range 0 to 1;  -- 0 = start next
read, 1 = in read-microsequence


begin
    -- FSM
    process(clk)
    begin
        if rising_edge(clk) then
            if rst = '1' then
                    state      <= IDLE;
                    timer <= (others => '0');
                    RE_reg         <= '0';
                    WE_reg         <= '0';
                    PE_n_reg       <= (others => '1');
                    data_out_reg   <= (others => '0');
                    mux_0_sel_reg  <= '0';
                    mux_1_sel_reg  <= (others => '0');
                    SA_0_E_reg     <= '0';
                    SA_1_E_reg     <= '0';
                    addr_bus_reg   <= (others => '0');
                    operation_reg <= (others => '0');
                    data_in_reg <= (others => '0');
                    addr_1_reg <= (others => '0');
                    addr_2_reg <= (others => '0');
                    enable_bw_rw_reg <= '0';
                    enable_mac_reg <= (others => '0');
                    row_counter <= (others => '0');
                    mac_stage   <= 0;
                    busy <= '1';
            else
                case state is
                    when IDLE =>
                        -- default outputs
```

```
    timer <= (others => '0');
    RE_reg         <= '0';
    WE_reg         <= '0';
    PE_n_reg       <= (others => '1');
    data_out_reg   <= (others => '0');
    mux_0_sel_reg  <= '0';
    mux_1_sel_reg  <= (others => '0');
    SA_0_E_reg     <= '0';
    SA_1_E_reg     <= '0';
    addr_bus_reg   <= (others => '0');
    -- start new transaction
    operation_reg <= operation;
    num_rows_reg <= num_rows;
    num_column_reg <= num_column;
    enable_bw_rw_reg <= '0';
    enable_mac_reg <= (others => '0');
    data_in_reg <= data_in;
    addr_1_reg <= addr_1;
    addr_2_reg <= addr_2;
    row_counter <= (others => '0');
    mac_stage   <= 0;

    case operation is
          when "0000" =>
            state <= IDLE;
          when "0001" =>
            state <= WRITE_START;
          when "0010" =>
            state <= READ_START;
          when "0011" =>
            state <= READ_START;
          when "0100" =>
            state <= bitwise;
          when "0101" =>
            state <= bitwise;
          when "0110" =>
            state <= bitwise;
          when "0111" =>
            state <= bitwise;
          when "1000" =>
            state <= bitwise;
          when "1001" =>
            state <= bitwise;
          when "1101" =>
            state <= copy;
          when "1011" =>
            state <= MAC;
          when others =>
            state <= IDLE;  -- fallback
      end case;
    busy <= '0';

-- WRITE sequence
when WRITE_START =>  -- time 0
    WE_reg       <= '1';
    addr_bus_reg  <= addr_1_reg;
    timer    <= to_unsigned(1,5);
    state    <= WRITE_DATA;
```

```
    busy <= '1';

when WRITE_DATA =>    -- time 4 ns
    if timer = to_unsigned(2,5) then
        data_out_reg  <= data_in_reg;
    end if;
    if timer = to_unsigned(10,5) then
        WE_reg        <= '0';
        data_out_reg  <= (others=>'0');
        state    <= IDLE;
    else
        timer <= timer + 1;
    end if;
    busy <= '1';

-- READ & READ_NOT sequence
when READ_START =>
    PE_n_reg  <= (others => '0');
    timer    <= to_unsigned(1,5);
    state    <= READ;
    busy <= '1';
when READ =>
    if timer = to_unsigned(2,5) then
        PE_n_reg  <= (others => '1');
    elsif timer =  to_unsigned(3 ,5) then
        RE_reg  <= '1'; addr_bus_reg  <= addr_1_reg;
   elsif timer =  to_unsigned(5,5) then
        RE_reg  <= '0';
    elsif timer =  to_unsigned(6,5) then
        SA_0_E_reg  <= '1';
    elsif timer =  to_unsigned(7,5) then
        mux_1_sel_reg  <= operation_reg;
        enable_bw_rw_reg <= '1';
    elsif timer = to_unsigned(9,5) then
        state <= IDLE;
    end if;
    timer <= timer + 1;
    busy <= '1';

-- BITWISE OPERATION
when bitwise =>
    -- latch first row
    if timer = to_unsigned(0,5) then
        PE_n_reg  <= (others => '0');
    elsif timer = to_unsigned(2,5) then
        PE_n_reg  <= (others => '1');
    elsif timer =  to_unsigned(3 ,5) then
        RE_reg  <= '1'; addr_bus_reg  <= addr_1_reg;
    elsif timer =  to_unsigned(5,5) then
        RE_reg  <= '0';
    elsif timer =  to_unsigned(6,5) then
        SA_0_E_reg  <= '1';

    -- latch second row
    elsif timer = to_unsigned(7,5) then
        PE_n_reg  <= (others => '0');
    elsif timer = to_unsigned(9,5) then
        PE_n_reg  <= (others => '1');
```

```
                    elsif timer =  to_unsigned(10,5) then
                        RE_reg  <= '1'; addr_bus_reg  <= addr_2_reg;
                    elsif timer =  to_unsigned(12,5) then
                        RE_reg  <= '0';
                    elsif timer =  to_unsigned(13,5) then
                        SA_1_E_reg  <= '1';

                    -- bitwise calculation
                    elsif timer =  to_unsigned(14,5) then
                        mux_1_sel_reg  <= operation_reg;
                        enable_bw_rw_reg <= '1';
                    elsif timer = to_unsigned(16,5) then
                        state <= IDLE;
                    end if;
                    timer <= timer + 1;
                    busy <= '1';


                -- COPY OPERATION
                when copy =>
                    if timer = to_unsigned(0,5) then
                        PE_n_reg  <= (others => '0');
                    elsif timer = to_unsigned(2,5) then
                        PE_n_reg  <= (others => '1');
                    elsif timer =  to_unsigned(3 ,5) then
                        RE_reg  <= '1'; addr_bus_reg  <= addr_1_reg;
                   elsif timer =  to_unsigned(5,5) then
                        RE_reg  <= '0';
                    elsif timer =  to_unsigned(6,5) then
                        SA_0_E_reg  <= '1';
                    elsif timer =  to_unsigned(7,5) then
                        mux_1_sel_reg  <= "0010";
                        enable_bw_rw_reg <= '1';
                    elsif timer =  to_unsigned(8,5) then
                        WE_reg       <= '1';
                        addr_bus_reg  <= addr_2_reg;
                    elsif timer = to_unsigned(10,5) then
                        mux_0_sel_reg  <= '1';
                    elsif timer = to_unsigned(18,5) then
                        WE_reg        <= '0';
                        data_out_reg  <= (others=>'0');
                        state    <= IDLE;
                    end if;
                        timer <= timer + 1;
                        busy <= '1';


                  -- MAC operation ->
                  when MAC =>
                        case mac_stage is
                          -- begin van een nieuwe read-microsequence
                          when 0 =>
                                -- bereken en zet het adres van deze
iteratie
                                addr_bus_reg <=
std_logic_vector(unsigned(addr_1_reg) + row_counter);
                                timer        <= to_unsigned(0,5);
                                mac_stage    <= 1;
```

```
                                when 1 =>
                                      if timer = to_unsigned(0,5) then
                                              for i in 0 to 31 loop
                                                  if i <
to_integer(unsigned(num_column_reg)) * 4 then
                                                      PE_n_reg(i) <= '0';
                                                  else
                                                      PE_n_reg(i) <= '1';
                                                  end if;
                                              end loop;
                                      elsif timer = to_unsigned(2,5) then
                                          PE_n_reg <= (others => '1');
                                      elsif timer = to_unsigned(3,5) then
                                          RE_reg       <= '1';
                                      elsif timer = to_unsigned(5,5) then
                                          RE_reg       <= '0';
                                      elsif timer = to_unsigned(6,5) then
                                          SA_0_E_reg   <= '1';
                                      elsif timer = to_unsigned(7,5) then
                                          mux_1_sel_reg     <=
operation_reg;
                                          enable_mac_reg <=
std_logic_vector(
                                           to_unsigned(

(2**to_integer(unsigned(num_column_reg))) - 1,
                                             8
                                           )
                                         );
                                      elsif timer = to_unsigned(9,5) then -
- verbeter dit als de multiplyer meer tijd nodig heeft
                                          -- microsequence klaar: kies of
we nog een rij moeten doen
                                          SA_0_E_reg   <= '0';
                                          enable_mac_reg <= (others =>
'0');
                                          if row_counter <
unsigned(num_rows_reg) -1 then
                                              row_counter <= row_counter +
1;
                                              mac_stage   <= 0;      --
start volgende rij
                                          else
                                              state       <= IDLE;   --
alles gelezen, klaar
                                          end if;
                                      end if;
                                      timer <= timer + 1;
                                      busy  <= '1';
                                end case;

                        when others => state <= IDLE;
                    end case;
                end if;
            end if;
        end process;

         RE <= RE_reg;
```

```
      WE <= WE_reg;
      PE_n <= PE_n_reg;
      data_out <= data_out_reg;
      mux_0_sel <= mux_0_sel_reg;
      mux_1_sel <= mux_1_sel_reg;
      SA_0_E  <=SA_0_E_reg;
      SA_1_E <= SA_1_E_reg;
      addr_bus <= addr_bus_reg;
      enable_mac <= enable_mac_reg;
      enable_bw_rw <= enable_bw_rw_reg;

end Behavioral;
```