POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Elettronica

Tesi di Laurea Magistrale

Open-Source Design of a Bitline-Computing-SRAM for Neural Networks Acceleration at the Edge

Relatore:
Chiar.mo Prof. Mario Roberto Casu

Candidato:
Francesco Falconieri

Aprile 2022

A mia nonna Nina,
che piú di ogni altro
sognava questo giorno,
al suo ardente amore
per tutti i nipoti.

To my grandmother Nina,
she who most of all
dreamed of this day,
to her fervid love
for all her grandsons.

Table of contents

Summary XIII

1 The unbearable communication cost of modern-day computing 1

1.1 The von-Neumann bottleneck 1

1.2 Artificial intelligence in edge devices 3

1.2.1 Neural networks inference 3

1.2.2 Reducing computational effort in neural networks 6

1.3 A new computing paradigm 9

2 Classification of In-Memory Computing architectures 13

2.1 Analog & Mixed-Signal Processing 13

2.1.1 Analog computing fundamentals for in-memory inference . . . 14

2.1.2 Architecture overview 15

2.1.3 Boosting the classification accuracy 17

2.1.4 On-silicon characterization 18

2.1.5 Discussion 20

2.2 Digital processing 22

2.2.1 Bitline computing 23

2.2.2 Isolated read-ports 26

2.2.3 Interleaved bitlines 28

2.2.4 Sequential pulsed wordlines 31

2.2.5 Discussion 33

3 Democratizing hardware design 35

3.1 OpenRAM: an open-source memory compiler 37

3.1.1 Module structure 37

3.1.2 Technology files 39

3.1.3 Architecture overview 40

3.2 SKY130: Skywater 130nm process design kit 41

4 Architecture concept and design 46

4.1 Bitcell array 47

4.2 Sense amplifier array 50

4.2.1 Computing sense amplifier 50

4.2.2 Circuit-level design of the BLC logic 52

4.2.3 Carry-ripple logic sizing 54

4.2.4 A novel circuit topology for single-ended sense amplifiers . . . 56

4.2.5 Time-borrowing sense amplifier 63

4.2.6 Layout 71

4.3 Address port 75

4.3.1 Address decoders 75

4.3.2 Pulse generator 76

4.3.3 Closed-loop pulse gating technique based on replica bitline . . 81

4.3.4 Pulse generator compiler 84

4.3.5 Layout 92

4.4 Population-count network 95
4.4.1 CSA-based binary tree reduction HDL generator 95
4.4.2 Dalalah bit-counting network 100
4.4.3 Bit-counting networks comparison 100
4.4.4 Open-source synthesis and place & route 103
4.4.5 Layout 107
4.5 Control logic 107
4.5.1 Timing diagrams and pins configuration 109
4.5.2 Layout 109

5 Simulation and results **114**
5.1 Area occupation 114
5.2 Simulation results 118
5.3 Operating frequency and energy consumption 122

6 Conclusions and future perspectives **128**

A Custom modules for OpenRAM in SKY130 **132**
A.1 Schematics 132
A.2 Layouts 134

Bibliography **136**

Acknowledgments I

List of tables

4.1 CSA - Static CMOS Logic Functions Area Occupation 54

4.2 Sense Amplifiers Area Comparison 60

4.3 Computing Sense Amplifier Design - Area Summary 72

4.4 Dalalah Bit-Counting Network - Hardware Implementation Results . 107

4.5 Bitline Computing SRAM - Command Table 110

5.1 BLC SRAM - Area occupation and partitioning 115

5.2 BLC SRAM Characterization - $V_{DD} = 1.8V$, T=300K 125

5.3 BLC SRAM Characterization - Process Corner TT, Environmental Variations 126

5.4 BLC SRAM Characterization - TTTT, SSSS and FFFF Corners . . . 127

List of figures

1.1 Comparison between the cost of accessing memories and arithmetic operations in a 45nm CMOS technology. . . . 3

1.2 Analogies between artificial and biological neurons. . . . 5

1.3 Different activation functions for neural-networks. . . . 5

1.4 Hardware characterization of different neural-network implemented on an FPGA. . . . 8

1.5 Classification results on the ImageNet datasets of AdderNets . . . 9

1.6 Quantization and inference in Binary Neural Networks. . . . 10

1.7 System level representation of the IMC paradigm . . . 11

1.8 IMC introduced improvements in data-centric applications. . . . 12

2.1 Column weak classifier and architecture schematic . . . 16

2.2 WL DAC archictecture and characterization . . . 16

2.3 Error-Adaptive Classifier Boosting (EACB) concept and schematic view . . . 19

2.4 Calibration cells for SA offset compensation . . . 19

2.5 Classification accuracy of the system at different abstraction and simulation levels . . . 21

2.6 Chip-level simulation results . . . 21

2.7 Example of a digital column computing instance with 2 bitcells. Input signals in red, partial results in green, output nodes in blue. . . . 22

2.8 Behavioural representation of the bitline computing scheme. . . . 23

2.9 Bitcell's content corruption due to cell flipping. . . . 25

2.10 Issues arising from simultaneous 6T bitcells access 25
2.11 Different SRAM bitcell configurations. 27
2.12 8T BLC configuration and sensing scheme 28
2.13 4+2T bitcell configuration 29
2.14 DDC versus bulk technology simulation results. 29
2.15 Interleaved wordlines configuration 31
2.16 Bitcell's layout stretch for interleaved wordlines. 31
2.17 Sequential pulsed wordlines technique 33
3.1 Design technology crisis 36
3.2 The OpenRAM software framework 38
3.3 Block scheme view of the OpenRAM SRAM architecture 42
3.4 Timing diagram for OpenRAM read operation. 43
3.5 Timing diagram for OpenRAM write operation. 43
3.6 Process stack diagram for SKY130 45
4.1 Primitive Block Scheme of the Computing SRAM Architecture. . . . 48
4.2 6T Bitcell SKY130 Implementation - Schematic and Layout. 50
4.3 6T Bitcell Array - Layout View of a 2x4 Configuration. 51
4.4 Computing Sense Amplifier schematic (a), post-technology mapping area occupation distribution (b) and mode control signals table (c). . 53
4.5 Computing Sense Amplifier's critical (in red) and fast (in blue) paths. 54
4.6 Computing Sense Amplifier - Transmission Gate Logic. 55
4.7 Computing Sense Amplifier - Carry Ripple Logic Sizing Results. . . . 57
4.7 Computing Sense Amplifier - Carry Ripple Logic sizing result. 58

4.7 Computing Sense Amplifier - Carry Ripple Logic sizing result. 59

4.8 Proposed Single-Ended Sense-Amplifier's schematics. 61

4.9 Time-Borrowing Sense Amplifier Waveforms. 62

4.10 Sense Amplifiers area-delay-energy comparison. 64

4.11 Time-Borrowing Sense Amplifier's Pulse Resilience Simulation Results. 68

4.11 Time-Borrowing Sense Amplifier's Pulse Resilience Simulation Results. 69

4.12 Sense Amplifier Dynamic Threshold - Monte Carlo and Process Corners Simulations 70

4.12 Sense Amplifier Dynamic Threshold - Monte Carlo and Process Corners Simulations 71

4.13 Computing Sense Amplifier - Layout Views. 73

4.14 Computing Sense Amplifier Array - Layout Views. 74

4.15 Address Decoding Networks for Multiple Address Words 76

4.16 Pulse Generator - Schematics. 77

4.17 Pulse Generator - 2 Pulses Clock Chopper Characterization. 79

4.17 Pulse Generator - 2 Pulses Clock Chopper Characterization. 80

4.18 Closed Loop RBL Pulse Gating - Block Scheme. 83

4.19 Closed Loop RBL Pulse Gating - Replica Rows. 85

4.20 Closed Loop RBL Pulse Gating - Simulation Results. 86

4.21 Pulse Generator Compiler - Flow Structure 88

4.22 Clock Chopper Characterization - Single Pulse 90

4.22 Clock Chopper Characterization - Single Pulse 91

4.23 Closed Loop Bitline Discharge Times - Simulation Results and Fitting. 92

4.24 Pulse Generator Compiler - Layout Results for Different Configurations. 93

4.25 Address Port - Layout View for a 16 Rows Array Configuration. . . . 94

4.26 3D Array - CSA HDL Generator 97

4.27 Popcount Networks - 8 bits Dalalah. 101

4.28 Popcount Networks - 16 bits Dalalah. 102

4.29 Popcount Networks - Synthesis and Characterization. 104

4.29 Popcount Networks - Synthesis and Characterization. 105

4.30 Popcount Networks - Final Architecture. 105

4.31 Popcount Network - Layout View. 108

4.32 Timing Diagram - Waveforms for Pulsed Bitline Computing. 110

4.33 Control Logic - Logic Level Schematic View. 112

4.34 Control Logic - Layout View. 113

5.1 Layout view of the 32-bits 128x128 BLC SRAM array. 116

5.2 Layout view of the 32-bits 128x128 BLC SRAM array with components highlight. 117

5.3 Output waveforms for read and BLC operating modes: pass-transistor mux.Green waveforms represent the output of the SA array from the LSB (top) to the MSB (bottom). 119

5.4 Output waveforms for read , BLC and sum operating modes: transmission-gate mux. 121

5.5 Sense amplifier failure in nominal-threshold pass-transistor mux. . . . 123

5.6 Sense amplifier behaviour in low-threshold pass-transistor mux. . . . 123

5.7 Sense amplifier behaviour in transmission-gate mux. 123

5.8 BLC SRAM waveforms for BLC mode. 124

5.9 BLC SRAM waveforms for standard read mode. 124

A.1 6T SRAM bitcell, only FET widths in micrometers are reported, $L_{CH} = 0.15\mu$m 132

A.2 Sense amplifier, ratios near FETs are to be intended as absolute channel widths over channel lengths in micrometers. 132

A.3 Write driver, $W_N = 0.42\mu$m, $W_P = 0.84\mu$m, $L_{CH} = 0.15\mu$m. 133

A.4 Write driver, $W_N = 0.42\mu$m, $W_P = 1.35\mu$m, $L_{CH} = 0.15\mu$m. 133

A.5 6T SRAM Bitcell 134

A.6 Sense amplifier. 134

A.7 Write driver. 134

A.8 D-flip-flop 135

Summary

The vast majority of state-of-the-art computing platforms are based on the von-Neumann Architecture, which consists of two different spatial dwellings for handled data: the main computing unit (i.e., the CPU) and the data storage unit (i.e., the memory). This approach allowed for continuous performance improvements in the last decades but began to reach a dead-end as soon as datacentric applications started to become pervasive, like the ones involved, for instance, in artificial intelligence, cryptography and signal processing, due to the well-known communication cost and von-Neumann bottleneck. This problem led to questioning whether this approach could still be pursued, hence the concept of In-Memory Computing (IMC) began to attract interest. The IMC paradigm aims to relocate the computing center from the CPU to the memory units (e.g., SRAMs or caches) for datacentric applications, to reduce the number of memory accesses. IMC solutions based on the CMOS technology can be classified into two main categories: analog and digital implementations. Analog IMC implementations allow massive parallelism but yield approximated results due to their analog nature subject to process variations and mismatches. This limits their applications to error-resilient ones like neural networks. Digital IMC implementations, unlike their analog counterparts, always yield 100% accurate results, expanding their applications in a larger number of fields, like for instance, cryptography, at the expense of reduced parallelism. Several SRAM or cache memory architectures have been proposed, which embed logic gates in the memory cells, implement sequential peripheral circuitry for complex computations, and exploit bitline computation in a wired-or fashion. This approach works flawlessly especially with Binary Neural Networks (BNNs), requiring a single bit for both activations and weights, hence reducing their product to a simple XNOR. Due to the ever-growing interest in deploying neural network models in edge devices, the popularity of the IMC concept is expected to increase, but the fact that the design of IMC devices is challenging might hinder its practical adoption. Following the OpenROAD philosophy of "democratizing hardware design", an open-source flow will be used to design a Bitline-Computing-SRAM based on a customized version

of OpenRAM memory compiler, to take a step forward in the direction of making viable the adoption of IMC accelerators. Toward this goal, in this thesis, the concept of pulsed wordlines bitline computing will be resumed and expanded to solve the read-stability problem of simultaneous activation in a 6T bitcell array. A computing sense-amplifier array will be introduced suiting the acceleration of BNN and AdderNets. A new circuit topology for a single-ended sense-amplifier will be proposed to enhance the system's performance as well as limit the area overhead while providing a variation-tolerant design through pulse-width resilient behaviour. Pursuant to this, the design of a pulse generator will be carried on, ultimately resulting in a specifications-to-GDSII compiler script for a memory array implementation. Moreover, to unlock the acceleration of BNNs, different bit-counting networks will be reviewed and compared thanks to an automatic HDL generator developed to be integrated into OpenRAM. Finally, the customized array will be characterized according to the main figures of merit regarding digital circuits, ultimately comparing the obtained results with other bitline computing architectures based on the same technological node. The work targets SkyWater's SKY130 180nm-130nm process open-source PDK, which represents a sweet spot for commercial IoT applications.

Chapter 1

The unbearable communication cost of modern-day computing

1.1 The von-Neumann bottleneck

The vast majority of modern days state-of-the-art computing platforms rely on a computation archetype called the *von-Neumann architecture* [1], consisting of two different spatial dwellings for handled data: the main computing unit (i.e. the CPU) and the data storage unit (i.e. memory). The interfacing between the two follows a load-store approach, which boils down to the following steps: accessing memory for data retrieving, computing results and further accessing memory for data storing. This communication scheme allowed for continuous performance enhancement in the last decades; following a divide et impera approach, the two different instances experienced individual optimizations that overall increased the efficiency of the system.

Although designers have been pushing the boundaries of throughput and power consumption in nowadays special purpose or application-specific architectures, memory accesses have always been expensive from a performance perspective. Older primitive architectures fully relied on off-chip memories for data storage (apart from internal registers for intermediate results) [2], translating into off-chip communications. Off-chip communications introduce severe throughput limitations related, and not limited, to connections bandwidth and increased energy consumption arising from large capacitances of pads, wires and pins switched during the communication. This has been the fundamental reason that led to research over the years for a memory organization that could limit the number of off-chip events; the main result is

a *memory hierarchy* that exploits the *principle of locality* [3], both temporal and spatial, to embed a relatively small set of data (or instructions) in the CPU chip by integrating small memory arrays called *cache memories* [4]. This hierarchical approach has been responsible for a drastic reduction in off-chip memory access operations, limited to cache replacement events, together with a large speedup in the overall system's performance.

The *von-Neumann computing* paradigm reached a dead-end as soon as the involved applications running on platforms started to become *data-intensive* (or *data-centric*), like the ones involved, for instance, in *artificial intelligence* (AI) and signal processing. Older computing tasks can be classified as *application-centric*: the microprocessor was supplied with a certain set of instructions that were less likely to change with respect to data sets they were supposed to process. With the advent of big-data, the number of algorithms to be performed on the same inputs increased, shifting the stationary asset of the system from instructions to data, hence the epithet of *data-centric*. Working with a large amount of data weakens all those system optimizations gained through a hierarchical memory organization. This is because the required storage space for computations regarding a single context goes beyond the caching capabilities, thus resulting in an increment of memory accesses. The larger the data set, the larger the required memory area for storage, the farther the memory location from the computing cores, resulting in an ever-increasing communication cost. As a result, the cost of accessing memory arrays recently became larger than the one related to complex arithmetic floating-point operations (Fig. 1.1) [5]. This led to a shift in the optimization's target from CPU speed-ups towards data movement's efficiency.
All these collateral effects of today's data-centric applications on conventional architectures may be grouped under the so-called *memory wall* problem also known as *von-Neumann bottleneck* [6].

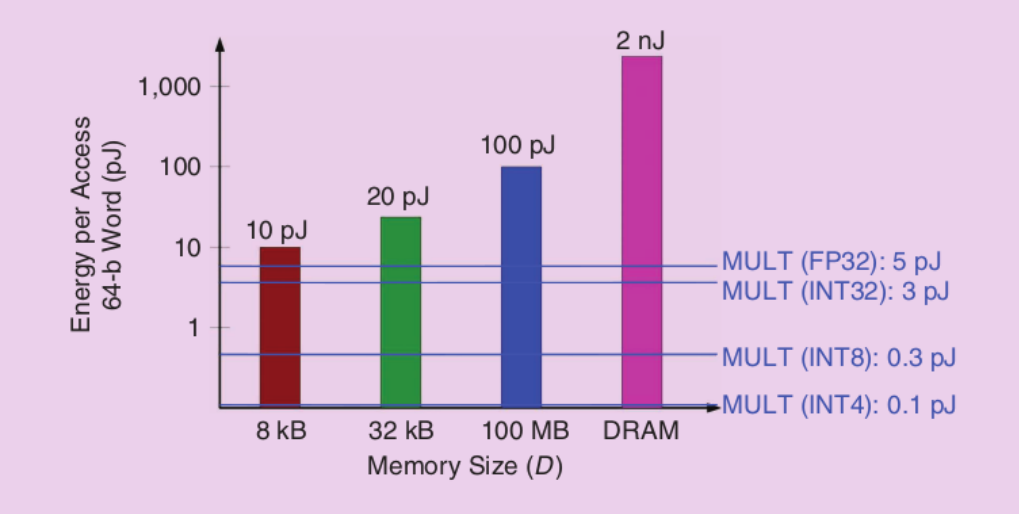


Figure 1.1: Comparison between the cost of accessing memories and arithmetic operations in a 45nm CMOS technology [5].

1.2 Artificial intelligence in edge devices

As the previous section introduced the concept of data-centric applications and processing, in the following a brief discussion about neural networks will be presented. Its main aim is to introduce the reader to useful *machine-learning* concepts for an easier understanding of the following sections. Key points of this description will be required computations involved in neural-networks acceleration, a highlight of the related computational effort and the delivery of an overview of proposed techniques to unlock artificial intelligence on low-computational resources such as edge devices.

1.2.1 Neural networks inference

Machine Learning (ML) is a branch of *Artificial Intelligence (AI)* whose aim is to provide computational models able to dynamically learn from a certain set of data how to perform some reasoning on a more general ensemble of inputs. Usually, these operations translate into object and sound detection, image classification, image and sound recognition.

Modeled on the human brain, a neural network consists of a large group of elementary processing unit nodes, called *artificial neurons* (Fig. 1.2). In the style of a biological one, each artificial neuron exhibits a certain number of inputs (*dendrites*) and a single output (*axon*). Before reaching the core of the node, each input is weighted according to the "strength" chosen for that particular connection (*synaptic-strength*). Finally, the processed inputs are combined and the output of the neuron is activated accordingly. To emulate the complex synaptic interconnections of the brain, each neuron's output translates into the input for a large number of nodes, thus resulting in an *artificial neural-network*.
Artificial neural networks are organized in stages and can grow in nodes count and interconnection's complexity according to their target application; the final stage of the net is the one that delivers the final decision of the *"machine reasoning"*, hence the classification's result. This process of evaluation is called *inference*. To give a basic understanding of how the aforementioned operation works, it is sufficient to describe the fundamental mathematical computations carried out by a single neuron.

The processing behind the evaluation of a fundamental node of the network can be summarized in two macro-steps: a *multiply-and-accumulate* (MAC) step and an *activation function* evaluation; a graph representation of these can be observed in Fig. 1.2. The MAC step can be further divided into an addition step and many multiplication steps. The neuron's input values, called *activations*, get multiplied by a set of constants, specific for the single node and called *weights*, defined during the *training* of the network; these are later combined through the last sum operation.
The final step in the neuron's processing is the application of an activation function to the MAC result. In older networks this translated into a comparison between the latter and a given value, called *threshold*, resulting in the sign function plotted in Fig. 1.3. If the sum yields a result whose value is larger than the specific threshold, the output of the single neuron "fires" (i.e. is 1), otherwise it keeps silent (i.e. is 0). With the advancement of mathematical models behind neural networks, activation functions evolved, assuming sophisticated shapes like the ones of sigmoids and hyperbolic tangents as in Fig. 1.3. This widens the range of possible output results improving the network's training.

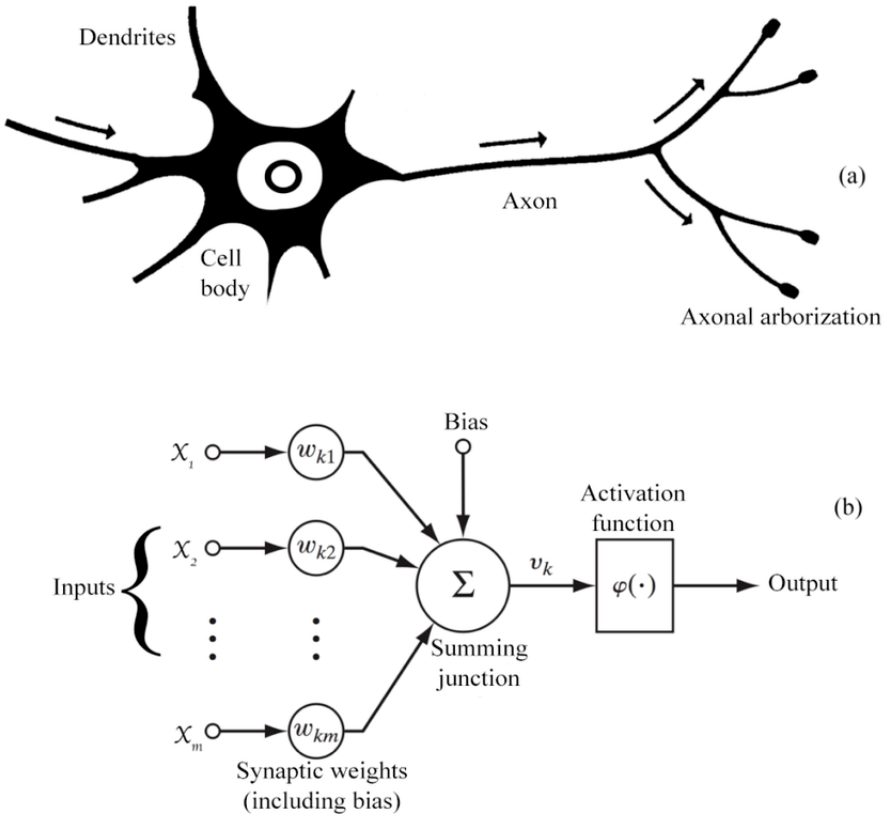


Figure 1.2: Analogies between artificial (b) and biological (a) neurons [7].

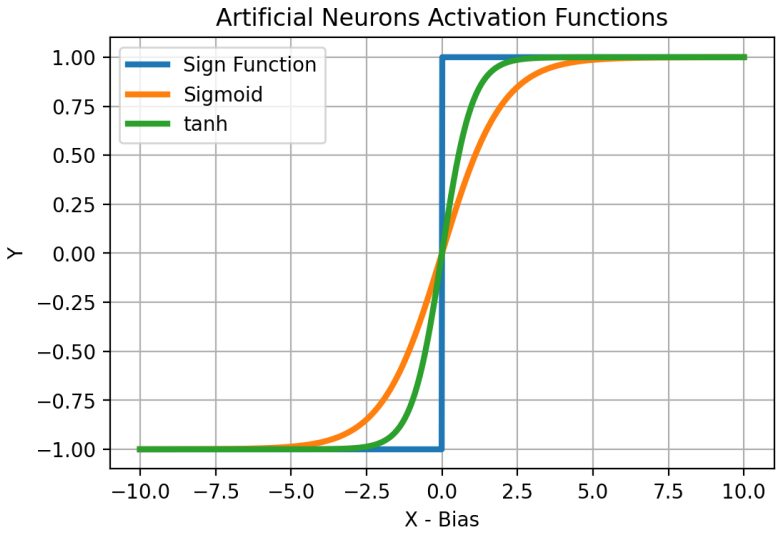


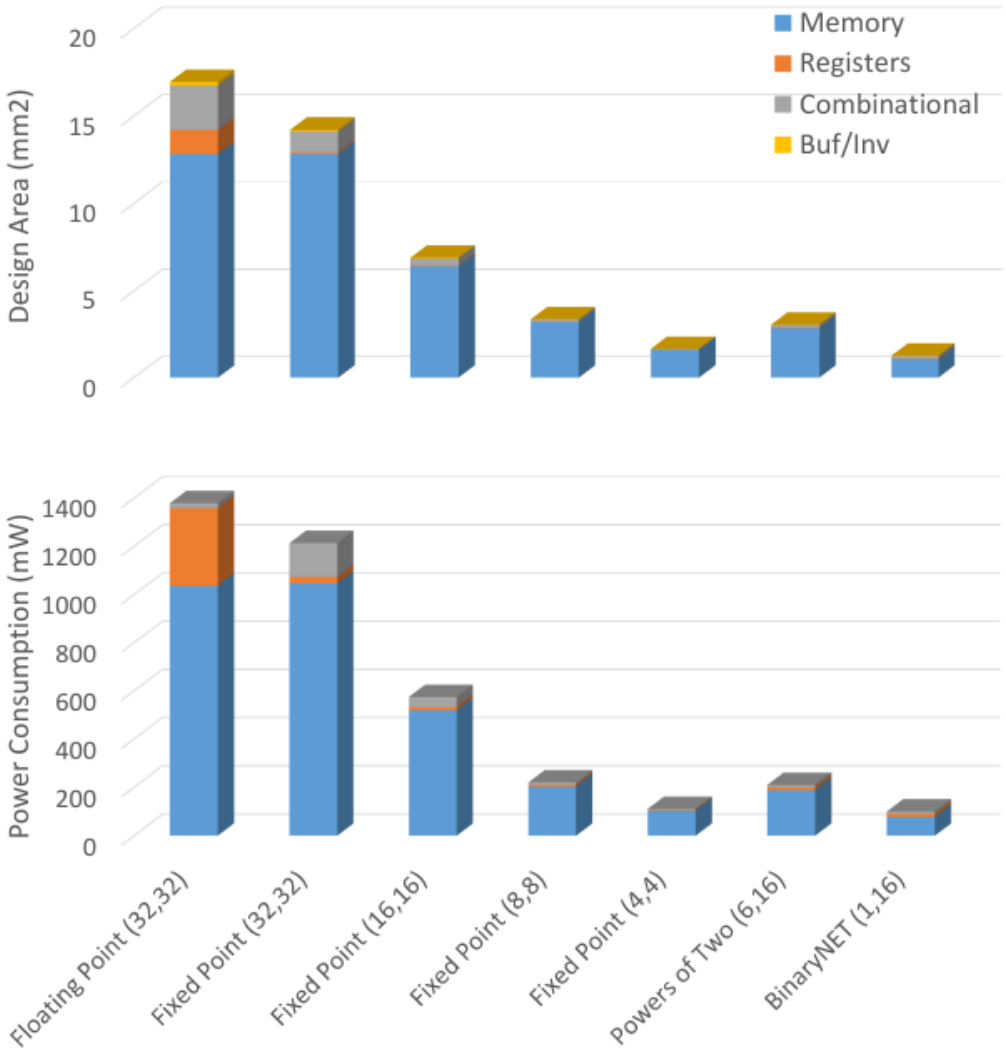
Figure 1.3: Different activation functions for neural-networks.

1.2.2 Reducing computational effort in neural networks

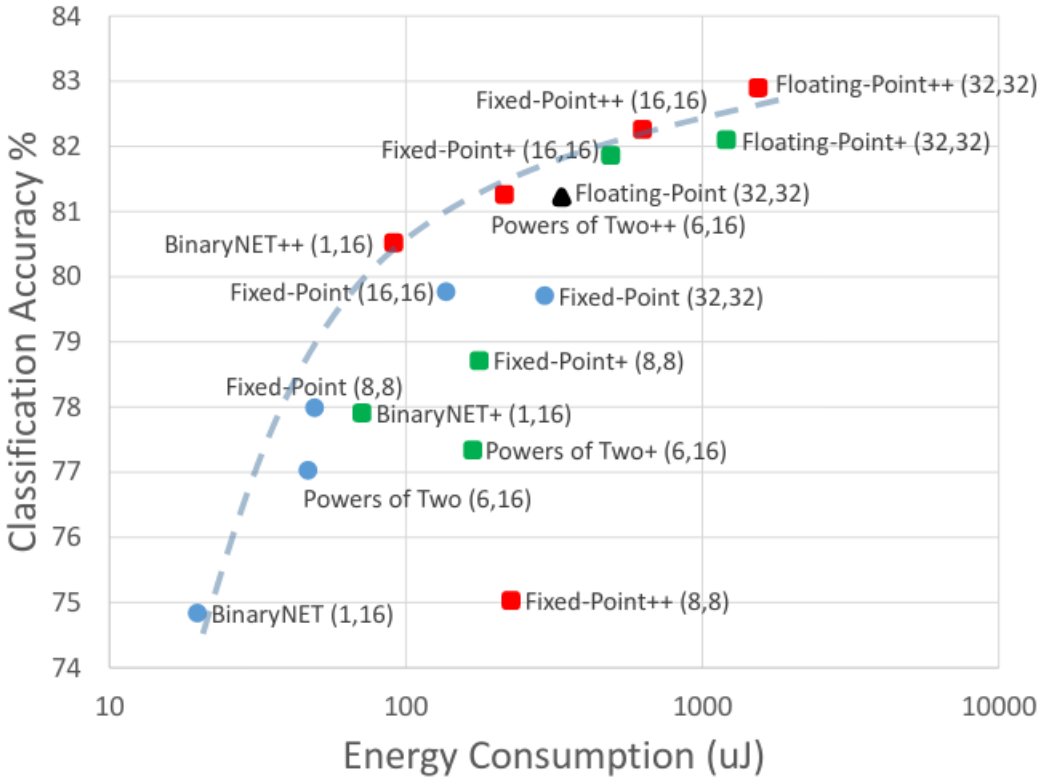
The introduced mathematical model of a processing node and, more in general, of a neural network, is both elegant and powerful from a software point of view but, from a hardware perspective, results in a complex netlist of entities, each performing heavy arithmetic computations in a floating-point representation. This yields a huge, slow and energy-hungry processing core, ultimately bounding artificial-intelligence applications to high-performance computing platforms like graphics processing units (GPUs) and high-performance computers (HPCs). However, at the expense of reduced accuracy, area, performance and energy consumption can be kept under tight control by employing a more "hardware-friendly" number representation as well as lowering the internal parallelism; this operation is called *quantization* of the network. Fig. 1.4 shows how number representation of weights and activations impacts the performances of a network at the expense of classification accuracy, ultimately unleashing the power of artificial intelligence in limited computational resources devices.

An extreme quantization boils down the number representation of both weights and activations to a single bit, limiting their values to +1 or -1 (logic 1 or logic 0 respectively). That's the concept behind *Binary Neural Networks* (BNN) [8, 9], a machine learning model that severely reduces the network's energy consumption, area occupation and required storage space at the expense of a severe loss of information (i.e. reduced accuracy) in the inference process (Fig. 1.6c).

As explained in [11], computational savings in a BNN (Fig. 1.6c) originate from the regression of required arithmetic operations into simpler ones due to single-bit quantization. In fact, given the number representation in Fig. 1.6a, the convolution process turns into the one shown in Fig. 1.6b. The 1-bit signed multiplication translates into a bitwise XNOR operation and the accumulation is substituted by a bit-count (population count, *popcount* hereafter). As a result, BNNs nowadays represent one of the keys to unlocking artificial intelligence in edge computing smart devices whose computational power is limited from power consumption and area occupation.



(a) Characterization of a neural-network implemented a 65nm CMOS technology node at a clock frequency of 250MHz with respect to area occupation and power consumption for different formats of number representation [10].



(b) Characterization of a neural-network implemented a 65nm CMOS technology node at a clock frequency of 250MHz with respect to classification accuracy and power consumption [10].

Figure 1.4: Hardware characterization of different neural-network implemented on an FPGA.

With the same purpose of BNNs, recently a new mathematical model has been proposed for reducing computational effort in neural networks. As stated previously, arithmetic operations involved in network inference boil down to multiplication and addition. However, a significant difference exists between the two both from latency and energy consumption perspectives. Multiplications, in fact, translate into complex and power-hungry hardware networks, increasing the power consumption of the system. Accordingly, authors in [12] developed a new family of networks called AdderNets, which trades massive multiplications with much cheaper additions to reduce the computation costs. As can be seen in Fig. 1.5, this new approach allows for a complete multiplication avoidance in the inference process, while achieving a

Model	Method	#Mul.	#Add.	XNOR	Top-1 Acc.	Top-5 Acc.
ResNet-18	BNN	0	1.8G	1.8G	51.2%	73.2%
	AddNN	0	3.6G	0	67.0%	87.6%
	CNN	1.8G	1.8G	0	69.8%	89.1%
ResNet-50	BNN	0	3.9G	3.9G	55.8%	78.4%
	AddNN	0	7.7G	0	74.9%	91.7%
	CNN	3.9G	3.9G	0	76.2%	92.9%

Figure 1.5: Classification results on the ImageNet datasets of AdderNets [12].

classification accuracy higher than binarized neural networks.

1.3 A new computing paradigm

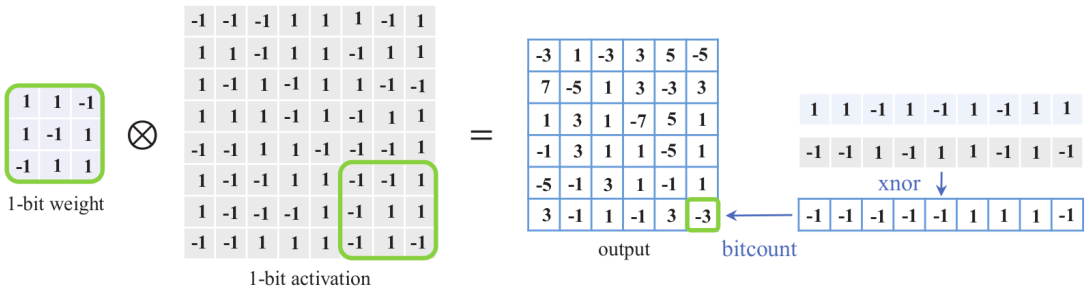
All the formerly introduced issues regarding von-Neumann architectures led to a huge effort over the last years for a change in the way of conceiving computing, one key result is the concept of *In-Memory Computing* (IMC).
The IMC paradigm aims at relocating the computing locus from the CPU to the memory array (Fig. 1.7) for data-centric applications, ultimately reducing the number of memory accesses to the ones needed to store input data and retrieve final results. The idea of exploiting memories for large computations comes from the intrinsically high parallelism of the array itself, due to the large storage space.

According to the adopted approach when it comes to embedding computation in the memory instance one can classify this new family of architectures in two different categories: *In-Memory Computing* and *Near-Memory Computing* (NMC).
Near-Memory computing can be seen as a brute force approach to the shift of computational effort; in fact, it only consists of integrating standard digital arithmetic and/or application-specific architectures in the memory macro nearby the array, exploiting concurrency between array read/write operations and auxiliary computations. However, there's no real innovation in the way of conceiving the two instances. The concept of NMC however is particularly well suited for large off-chip memory arrays and has been recently adopted by Samsung™ for AI acceleration under the

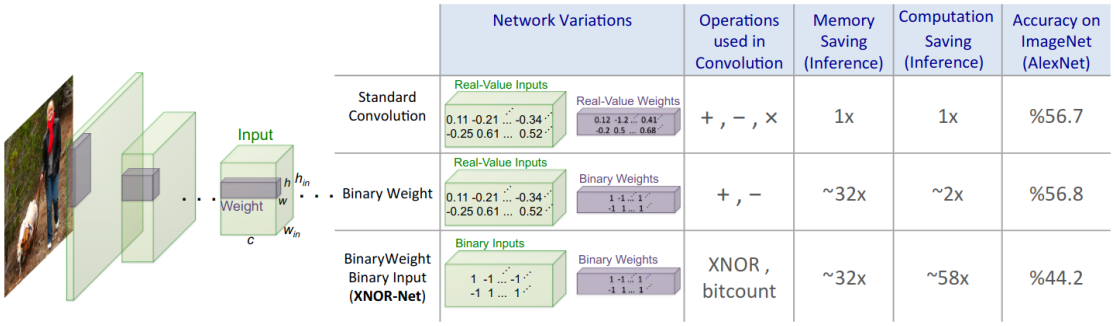
VALUE	
BINARY	DECIMAL
0	-1
1	+1

INPUT VALUES		MULT RESULT	
BINARY	DECIMAL	XNOR	DECIMAL
0 0	-1 -1	1	+1
0 1	-1 +1	0	-1
1 0	+1 -1	0	-1
1 1	+1 +1	1	+1

(a) Single bit quantization and XNOR multiplication in Binary Neural Networks



(b) Convolution process in Binary Neural Networks [11].



(c) Computational savings in Binary Neural Networks [11].

Figure 1.6: Quantization and inference in Binary Neural Networks.

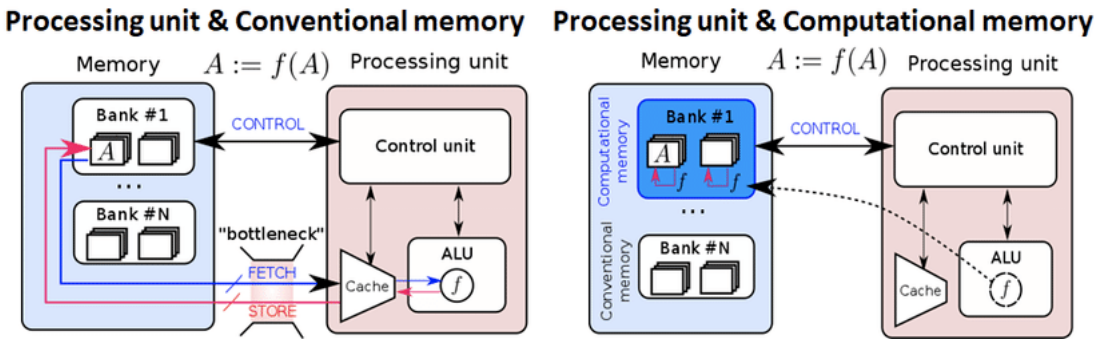
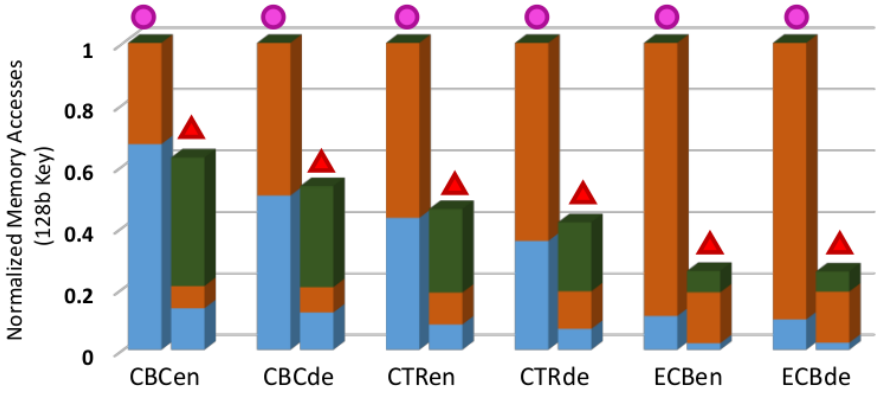


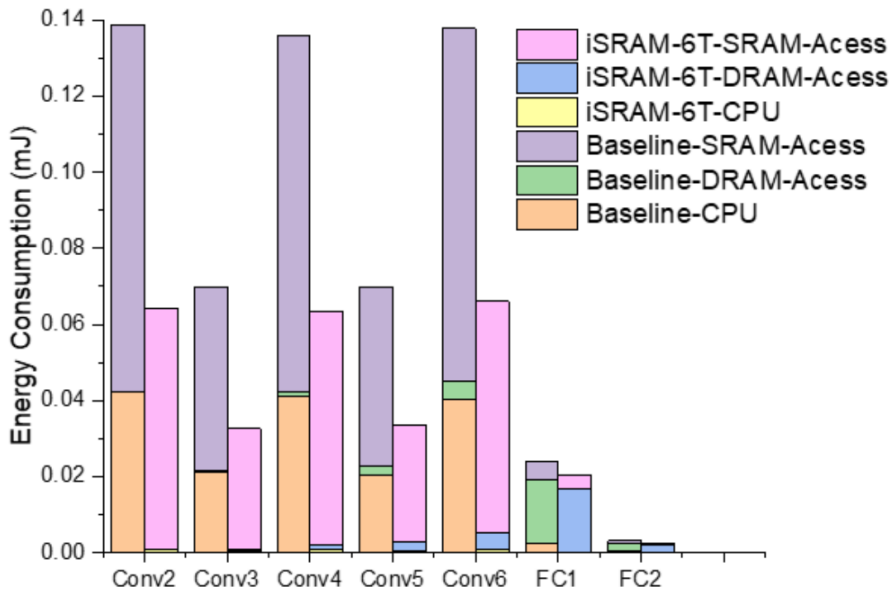
Figure 1.7: System level representation of the IMC paradigm [13]

name of *processing-in-memory* (PIM)[14].
In-Memory Computing instead aims at exploiting the array for computational purposes, ideally without the need for an off-array computing instance. This implementation demands more or less invasive array and peripherals customizations while promising an elevated throughput as well as a low area overhead.

The key result of this relocation of the computational effort is a drastic reduction in the number of memory accesses, ultimately reducing the overall system's power consumption and relieving the throughput degradation. Fig. 1.8a show the results of a system simulation of an IMC SRAM [15] exploited for the acceleration of cryptographic applications, where thanks to in-memory operations the number of memory accesses has been reduced by up to 74.7%. Similarly, Fig. 1.8b points out the power savings resulting from the application of a computing SRAM architecture in an AI environment, where a 2.16x energy reduction can be appreciated with respect to a baseline memory system. Due to these results, in-memory computing represents one key path to overcome the aforementioned memory wall in both high-performance and low-resources computing platforms. Thence, this work will deal with the investigation of different implementations of IMC approaches and the design of an SRAM architecture for AI acceleration. The flow of the project will be entirely based on open-source tools for IC design and will ultimately result in a memory compiler supplied with IMC features to allow a fast design space exploration and possibly boost the deployment of neural networks in edge devices.



(a) Memory access count of an IMC SRAM (triangle bar) system versus a baseline architecture (circle bar) in a 128-bit long key cryptographic application [15].



(b) Energy consumption comparison between an IMC SRAM system and a baseline architecture in a AI environment [16].

Figure 1.8: IMC introduced improvements in data-centric applications.

Chapter 2

Classification of In-Memory Computing architectures

As soon as memory wall limitations became unbearable, researchers and scientists challenged themselves in the development of innovative IMC implementations. Different proposals have been presented based on CMOS and beyond-CMOS technologies, exploiting both *Analog & Mixed-Signal (AMS) Processing* and *Digital Processing*. This chapter aims at delivering a detailed overview of state-of-the-art computing SRAMs architectures based on CMOS technology, highlighting the strengths and drawbacks of each proposal ultimately evaluating their compatibility with a memory compiler.

2.1 Analog & Mixed-Signal Processing

As explained in Sec. 1.2.2, computations in neural networks can be summarized into two main categories: MAC operations and *threshold comparisons (TC)*. Analog designers have been working out plenty of mixed-signal circuits to successfully achieve an accurate evaluation of the these computations. Concerning MAC operations, most popular processing methods rely on *charge-sum*[17, 18, 19] and *current-sum*[20, 21, 22], while for TC programmable comparators are widely exploited [20, 23, 22].

As a way to describe key concepts of analog IMC together with its strengths and drawbacks, in this section, the proposal in [20] will be investigated with an eye on final classification accuracy, performance and implementation cost.

2.1.1 Analog computing fundamentals for in-memory inference

The proposed MAC scheme is the one based on the current sum. Given the properties of a standard MOSFET device, a rudimentary model for an n-channel device's drain-source current while in the saturation region is:

$$I_{DS} = \beta_N \cdot (V_{GS} - V_{TH})^2 \cdot (1 + \lambda \cdot V_{DS}) \tag{2.1}$$

Linearizing the equation with respect to gate-source voltage and drain-source voltage one obtains the small-signal's equations for the device's output current, where transconductance and output's resistance respectively can be identified:

$$i_{ds} = g_m \cdot v_{gs} + g_o \cdot v_{ds} \quad \text{with} \quad g_m = f\left(\frac{W}{L}\right),\ g_o = \frac{1}{r_o} \approx f\left(\frac{1}{L}\right) \tag{2.2}$$

As one can notice, both quantities multiplying the injected signal's voltage are under the control of the designer: different transistor sizes and aspect ratios will result in different values of transconductances and output resistances respectively. Summing the current sank by N different nFETs, with different sizes and different gate-source applied voltages one obtains:

$$i_{TOT} = \sum_{n=0}^{N} g_{m,n} \cdot v_{gs,n} = (g_{m,0}v_{gs,0} + g_{m,0}v_{gs,0} + ... + g_{m,N}v_{gs,N}) = \vec{\mathbf{g_m}} \cdot \vec{\mathbf{v_{gs}}} \tag{2.3}$$

$$i_{TOT} = \sum_{n=0}^{N} g_{o,n} \cdot v_{ds,n} = (g_{o,0}v_{ds,0} + g_{o,0}v_{ds,0} + ... + g_{o,N}v_{ds,N}) = \vec{\mathbf{g_o}} \cdot \vec{\mathbf{v_{ds}}} \tag{2.4}$$

By looking at Eq. 2.3 and Eq. 2.4 one can easily point out an equivalence with the computational model of the neuron presented in Sec. 1.2.2, by linking for instance weights to conductances and input activations to input voltages. This unlocks an alternative to typical digital computing that has been widely explored in literature due to the memory array topology. In fact, the current sum formalized in Eq. 2.3-2.4 requires the drains of the nFETs to share the same node as in the case of array's bitlines, where the access transistors of the cells of a given column are connected to

the same line.

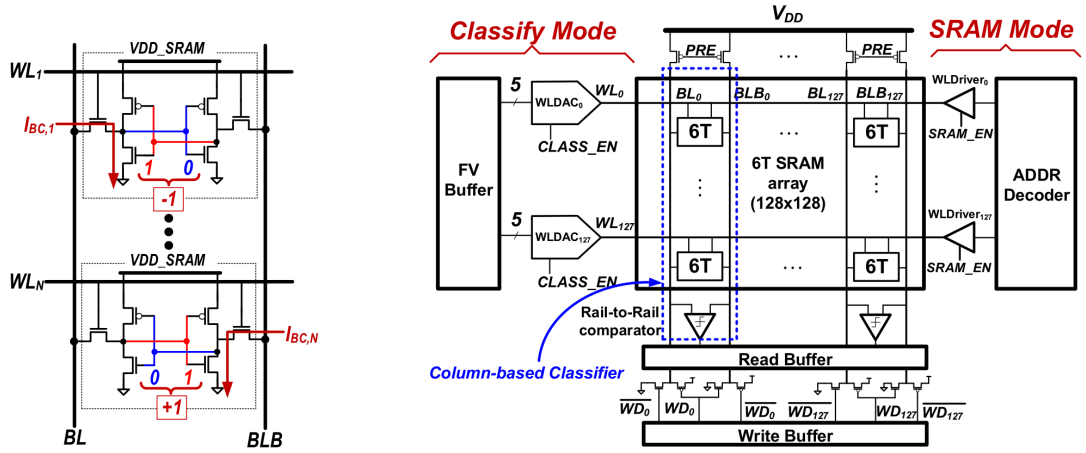
2.1.2 Architecture overview

The core of the architecture is a 6T SRAM array. The authors deliver a memory instance that can operate in two modes: standard SRAM operations (such as read and write) and "classify mode", where NN inference is performed. Classification is achieved by noticing that each column of the array can be seen as a *linear-classifier*, where the related output equation is:

$$d = sign\left(\sum_{i=1}^{N} w_i \cdot x_i\right) \tag{2.5}$$

where one can identify the two main operations introduced in former sections: multiply-and-accumulate (dot product) and threshold comparison (sign function applied to the MAC result). Given the analog processing scheme introduced previously, one can relate the accumulation result to the overall current of a column's bitline. The i-th weight is represented by the equivalent transconductance of the cell's access port (series connection of the access pass-transistor and the saturated FET of the inverter) while the i-th input is the input voltage of the access transistor (WL voltage). The threshold comparison (sign function) is addressed to a comparator at the end of the column. However, the sign of the equivalent conductance is dependent on the content of the cell: if the stored value is 1 the current will be sourced from the inverter's pFET, if the value is 0 it will be sunk by the nFET. The ultimate result of these two scenarios is to increase or decrease the bitline voltage respectively. Assuming the same input value for the wordline voltage of both configurations one obtains a binary quantization of the current values, resulting in a classifier with binary weights (Fig. 2.1a).

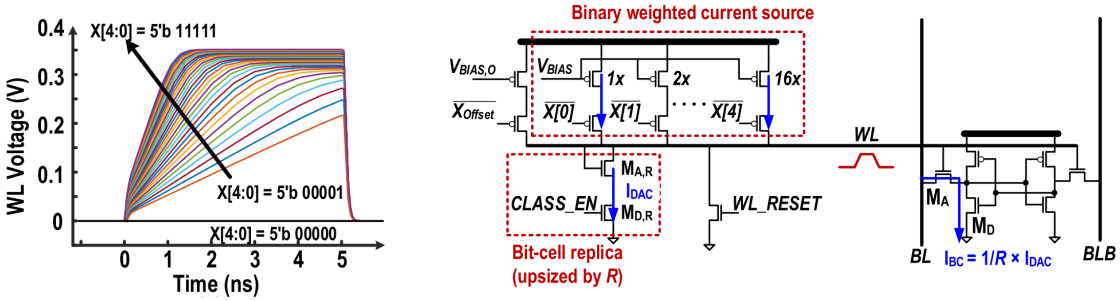
Input values, instead, are free to assume a broader range of values due to 5-bit quantization. Features of the classifier are loaded serially in a dedicated Frame Values Buffer and then fed to an array of WL DACs (Fig. 2.1b). During a classification cycle, unlike standard read/write operations, all the WLs are driven at once with a



(a) Column weak classifier with binary weights current configurations.

(b) Architecture block scheme.

Figure 2.1: Column weak classifier and architecture schematic [20].



(a) Output waveforms of the DAC.

(b) Schematic of the binary weighted current source DAC.

Figure 2.2: WL DAC archictecture and characterization [20].

different value coming from the DAC (Fig. 2.2b) array, resulting in different input voltages for the access transistors of different rows. This way, the magnitude of the current sank/sourced by each cell will depend on both the feature value and the weight value, resulting in a multiplication. To avoid the unwanted corruption of the cell's content, the DAC output range has been limited to 0.4V ($0.3V_{DD}$) (Fig. 2.2a).

2.1.3 Boosting the classification accuracy

The described architecture can support inference for 5-bit input neural networks and 1-bit weights. However, as anticipated, the mathematical model of the column processor results in a weak classifier, thus limiting the classification accuracy to 52% (Fig. 2.5). Moreover, process variations, device mismatches and BL discharge non-idealities additionally degrade the network's quality introducing non-linearities. To account for this degradation the authors adopted a *boosting* algorithm: an approach from machine learning for constructing a strong classifier from multiple base weak classifiers. This requires a further MAC operation on the weak classifiers' outputs, resulting in the insertion of additional logic for digital arithmetic operations (Fig. 2.3). The resulting scheme raises the ideal classification accuracy of the column classifier to 91%.

A further source of accuracy degradation is the sense amplifier's input offset voltage. The ultimate result of the described MAC operation is to deliver two different values for BL and BLB voltages depending on weights and features. To perform threshold comparison, the difference between the two is applied as a differential voltage to the inputs of a latch-based SA. Ideally, the comparator should exhibit a threshold equal to half the supply voltage; although, due to devices mismatches, an input offset voltage will shift the threshold value from ideality. This translates into a wrong classifier's activation function, deteriorating the SNM of the inference process. To restore the quality of threshold comparison, the authors supplied each column with additional cells. At the device's start-up, a calibration phase is performed: the amplifier is driven by the same input voltage (i.e. pre-charged BLs) and its output is sampled. If it shows a value different from $0.5V_{DD}$, a certain number of cells will

be set to 0 or 1 according to the sign of the output (Fig. 2.4). The result of this calibration is the insertion of a deterministic bias in the MAC operation that will compensate for the amplifier's input offset, enhancing the strength of the classifier.

2.1.4 On-silicon characterization

To characterize the performances of the architecture, authors delivered simulations performed on a 128x128 SRAM custom array implemented in a 130nm CMOS process. Thanks to the boosting algorithm and calibration cells, the final accuracy of the classification sits slightly above 90% (Fig. 2.5). Nonetheless, 18 iterations were required by the Error-Adaptive Classifier Boosting (EACB) to achieve this precision, severely increasing the latency of the system. Moreover, the additional circuitry for strong classifiers translates into a further source of area overhead. The aforementioned classification accuracy takes into account the presence of a certain number of calibration cells. In the prototyped configuration this translated into 32 additional rows in the array to achieve sufficient granularity for the SA's input offset compensation.

Results regarding the on-silicon characterization of the architecture can be seen in Fig. 2.6, where the area overhead due to IMC acceleration stands around 90% of the bitcell array. This is to be addressed mainly to the presence of the feature buffer and the DACs array. Another source of area increment is the presence of additional rows for the column calibration, which increases the size of the array of a factor of 1.25x. Ultimately, the SRAM exhibits significant differences in both access time and energy consumption regarding two modes: the classification cycle results 6x slower than normal SRAM operations while the energy per cycle increased by a factor of 3.

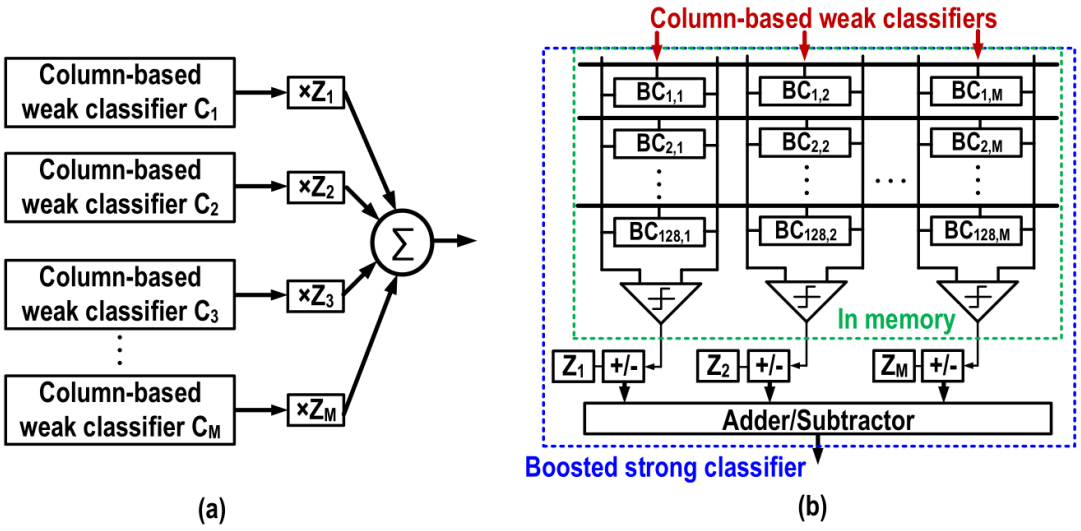


Figure 2.3: Error-Adaptive Classifier Boosting (EACB) concept and schematic view [20]

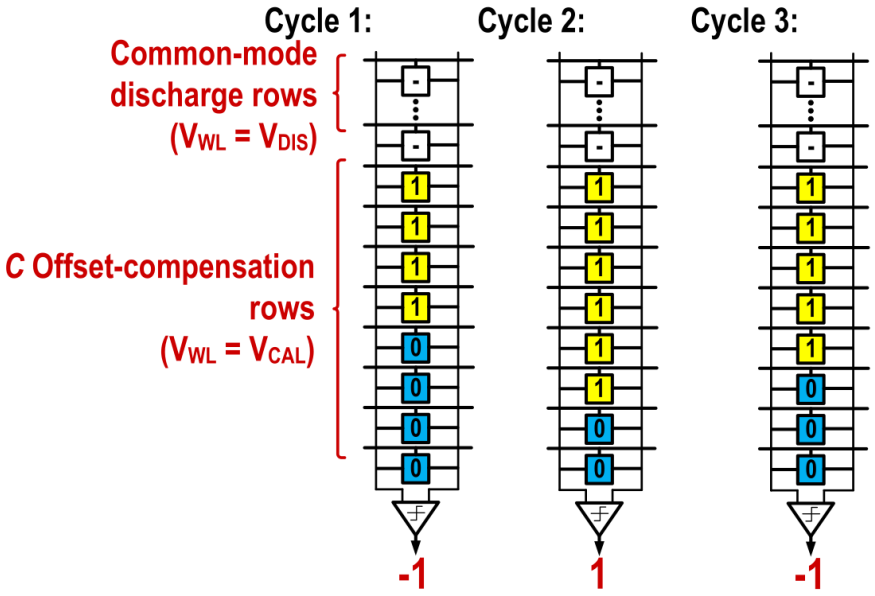


Figure 2.4: Calibration cells for SA offset compensation [20].

2.1.5 Discussion

This section presented a detailed overview of an IMC accelerated SRAM memory array based on analog computing. Albeit yielding a high classification accuracy, the architecture required the insertion of calibration schemes as well as a more complex training of the neural network. The resulting area of the system is large compared to the one of a standard SRAM array implemented in the same technology and both energy consumption and throughput exhibit a severe degradation. While the latter may hinder the practical adoption of the architecture from an efficiency perspective, the increased design complexity and its lowered regularity clash with a memory generator implementation. The presence of huge AMS peripherals is common to the vast majority of analog IMC architecture and unlocks the processing of feature words larger than the single bit, translating, however, into a major source of area-overhead. This, together with the pronounced sensibility to process variations, shifts the target of this work towards more reliable digital implementations at the expense of reduced parallelism. Accordingly, the following section will deliver an overview of state-of-the-art digital IMC architectures.

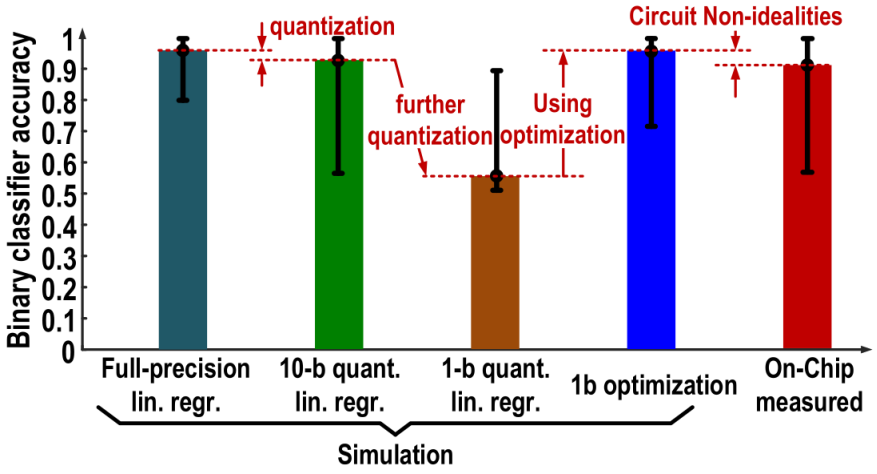


Figure 2.5: Classification accuracy of the system at different abstraction and simulation levels [20].

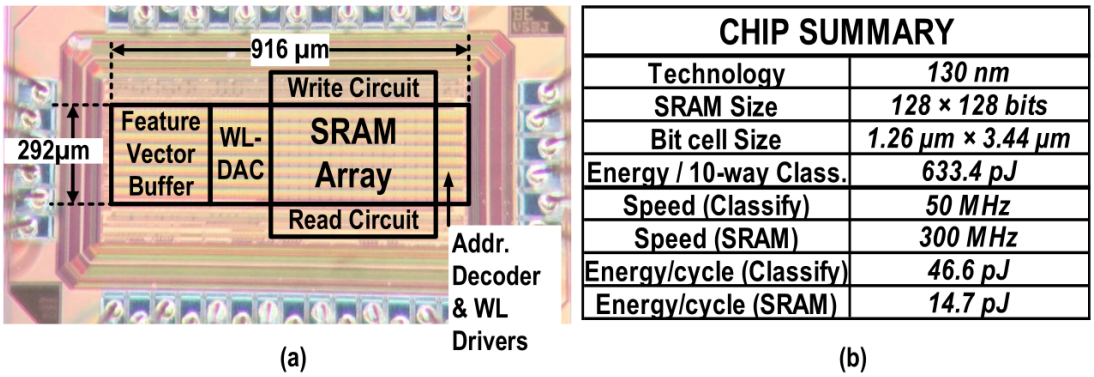


Figure 2.6: Chip-level simulation results [20].

2.2 Digital processing

According to the categorization used in this chapter, the term *digital processing IMC architectures* groups all those implementations that completely carry on computations in the digital domain. Meaning that the voltage of each node of the schematic ought to be coded on a single bit without any loss of information. The following sections present an overview of the current state-of-the-art digital IMC architectures, investigating different inference schemes. The compatibility of each proposal with a memory generator will be examined, eventually collecting features for the design of a novel architecture. To preserve the timing interface of the OpenRAM architecture, further analysis will only deal with single-cycle implementations. These memory arrays do not require stalling the entire system for complex computations, but their application is limited to BNNs. This is to be addressed to the structure of the array: from the analysis carried on in Sec. 2.1 one can identify the single column as a computing instance with a given number of input and output nodes. For instance, in a 6T SRAM column, inputs boil down to WL ports and bitcell's storage nodes (Fig. 2.7). BL instead represents the only output port as well as an intermediate computation node. As stated previously, all signals in a digital IMC architecture are constrained to own at most 1 bit of information; moreover, the need for accessing cells further reduces the number of input ports to be used for computation, thus limiting them to storage nodes solely. As a result, the elementary inference node translates into a 1-bit inputs 1-bit output device, ultimately limiting its target application to BNN only.

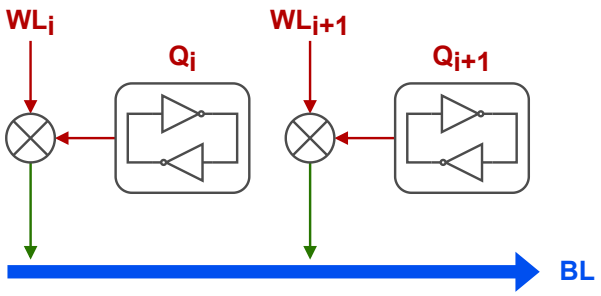
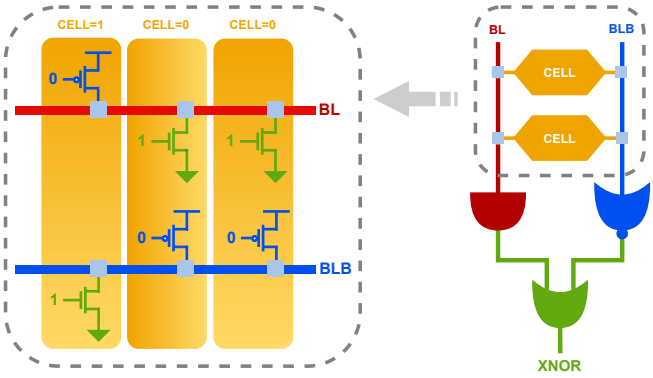


Figure 2.7: Example of a digital column computing instance with 2 bitcells. Input signals in red, partial results in green, output nodes in blue.



CELL 1	CELL 2	BL	BLB	XNOR
0	0	↓ 0	↑ 1	1
0	1	↓ 0	↓ 0	0
1	0	↓ 0	↓ 0	0
1	1	↑ 1	↓ 0	1

Figure 2.8: Behavioural representation of the bitline computing scheme.

2.2.1 Bitline computing

One promising approach for accelerating logical/arithmetical operations when it comes to BNN is *bitline computing* (BLC). As described in the previous chapter, bitline pairs can be exploited successfully for computations. However, their role is to support sum-like operations in the analog domain, according to the aforementioned current and charge accumulation methods. By extending the concept to the digital domain, column bitline pairs translate into a large fan-in wired-or line each, where inputs are the cells' contents (Fig. 2.8). If a given bitcell stores a logic 0, its activation will result in a pull-down device connected to the BL and a pull-up to the BLB, vice-versa for a cell storing a logic 1. By combining the effects of their concurrent presence on the BL, one can readily compute the truth table resulting from the activation of two words (Fig. 2.8). Through a deeper investigation, it is straightforward to identify the logic functions performed by this scheme, resulting in AND and NOR operations for BL and BLB respectively (Fig. 2.8). As stated in Sec. 1.2.2, multiply operations in BNN translate into the computation of a bitwise XNOR between weights and activations; this can be achieved readily by combining the bitline pair's results through an OR operation.

The only drawback of the introduced approach is the simultaneous access of two bitcells in the same column. A well-known issue in 6T (Fig. 2.11a) SRAM design is the one of read-stability. When a single bitcell is accessed, a low impedance path exists between the bitlines and the cell's storage nodes; for each read operation, one

of the column's BLs discharges through the pull-down nFET of the cell. During this transient, the current sank by the transistor will raise its drain's voltage, ultimately increasing the potential of the storage node corresponding to logic 0. If this value exceeds the threshold voltage of the inverters, the bistable loop flips, resulting in a destructive read. This led to investigating the consequences of accessing two cells simultaneously. Authors in [24] carried on a punctual analysis of the concerned scenarios, which boil down to two combinations: accessing two cells storing the same value or a different one. Two cells exhibiting the same content causes one of the BLs to discharge through a couple of nFETs. The current sank by the single transistor is therefore halved as the amount of charge accumulated on the drain nodes. This ultimately halves the node's peak voltage during the access transient, thus increasing the cells' read margin (Fig. 2.10a-b.1). On the other hand, if the cells store different values, the bitline charges and discharges simultaneously, creating a conductive path from V_{DD} to GND due to a pFET-nFET stack where both transistors conduct (Fig. 2.10b-a.1). As an ultimate result, the BL voltage will settle on an intermediate value linked to the FETs' relative sizing, with a large amount of current drawn from the supply. Moreover, as the cells store different values, both BL and BLB experience discharging and the storage nodes at logic 1 become vulnerable; if the BL's voltage exceeds the trip voltage of the two cross-coupled inverter pairs, the cells flip (Fig. 2.9). An accurate transient analysis of this scenario can be found in [24] under the concept of *dynamic read disturbance*.

The contamination occurring during simultaneous word accessing is the main factor that hinders a complete adoption of bitline computing. Researchers have been working out plenty of methods to overcome the read disturbance issue, hence the focus of the following sections will be to investigate these workarounds and ultimately pick the best ones for a reliable computing SRAM architecture. At the same time, the analysis will eventually describe and compare different peripheral circuits, unlocking the acceleration of various logic functions.

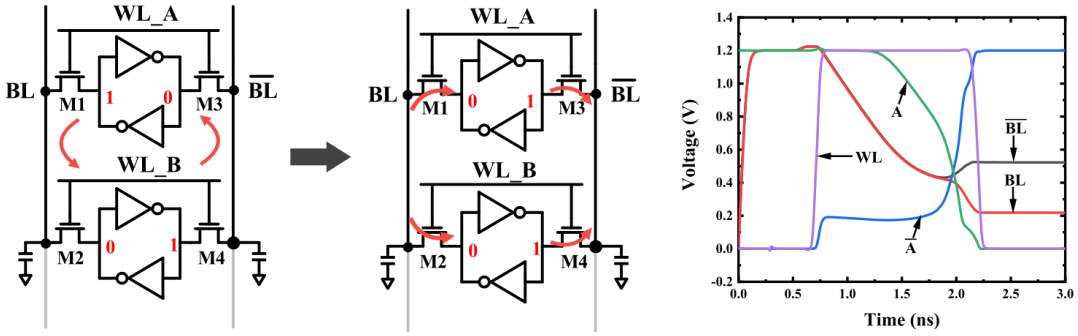


Figure 2.9: Bitcell's content corruption due to cell flipping [24].

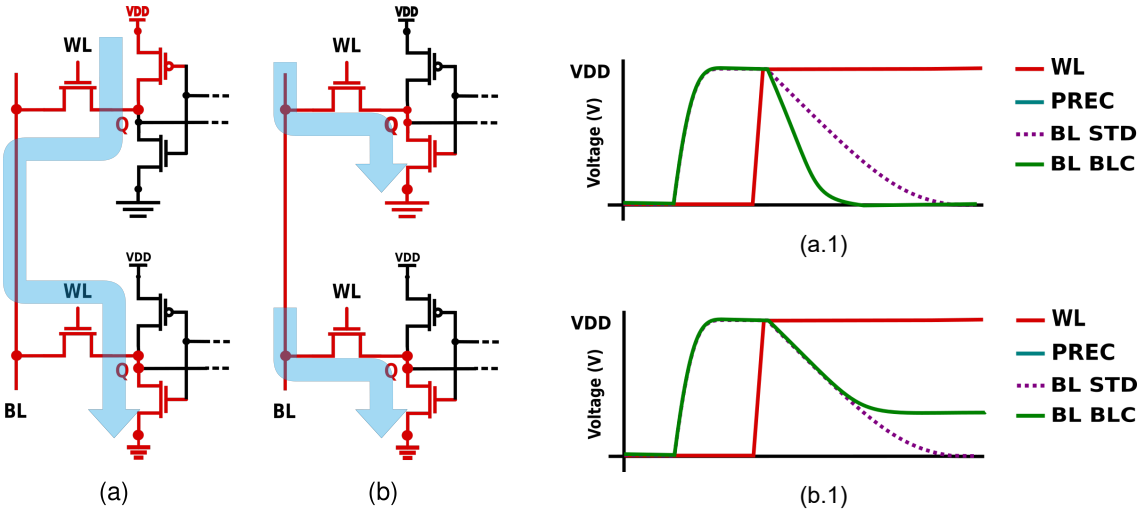


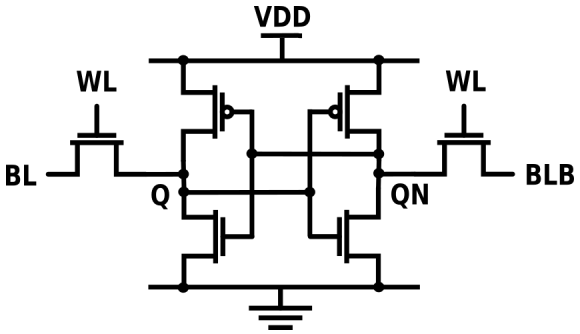
Figure 2.10: Cross conduction issue in simultaneous 6T bitcells access; scenarios consists of accessing complementary values (a, b.1) and equal values (b, a.1). Conducting devices are in red, current flow is visualized in blue.

2.2.2 Isolated read-ports

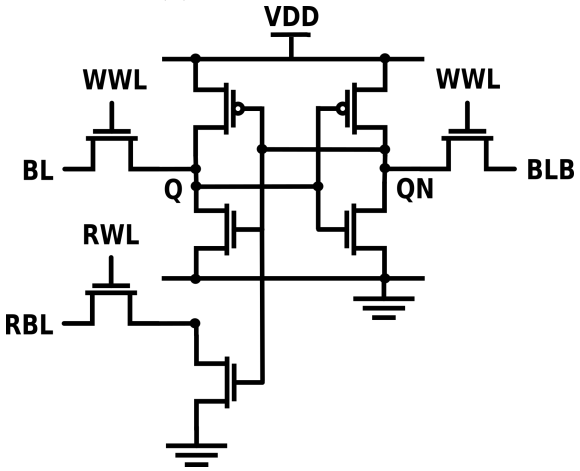
Because of the issue of read-disturbance arising from the adoption of a 6T bitcell as a storage element, one possible approach consists of shifting the circuit topology towards more compliant configurations. Supplying the bitcell with two isolated read ports eliminates the electrical connection between the BL and the storage nodes; that's the case of 10T bitcells. However, this configuration requires four additional FETs (Fig. 2.11c) to achieve the desired number of ports, translating into a source of area overhead which may be unbearable. An example of a 10T computing SRAM can be found in [25] where the authors sacrificed area for functionality.

A trade-off between area occupation and reliability is dropping one of the two read ports, thus moving towards an 8T bitcell configuration (Fig. 2.11b). The main drawback of this implementation is the absence of the complemented BL, hence a bitline computing result space limited to an AND operation only. However, the isolated read port allows for concurrent bitcell access without disturbance. By recalling the doubled discharge rate when accessing two cells storing the same value, authors in [15] decided to shrink the WL assertion pulse. This way, the BL will experience a full discharge only when the accessed values are '1-0' or '0-1'; otherwise the line will settle to an intermediate voltage (Fig. 2.12). This scheme allows the system to discriminate whether the two accessed cells store the same value or not, according to the final BL value. As a result, NAND and NOR computations can be performed by sensing the line with different skewed inverters (Fig. 2.12).

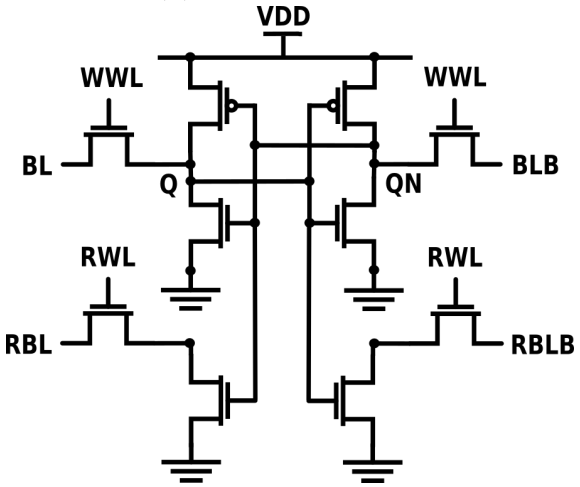
Albeit reliable, the increased bitcell's area of an 8T configuration might still clash with the tight area constraint of certain designs. Therefore, the transistor's count of the bitcell instance has to stick with the one of a 6T cell. To deliver a reliable architecture while retaining area occupation, authors in [26] designed a novel bitcell topology with isolated read ports without altering the FETs' count (Fig. 2.13). The proposal exploits the large body effect coefficient of deeply-depleted channel (DDC) technologies to achieve a write operation via the n-well. This way, the access transistors can be converted from input-output devices to output ones only; thus resulting in isolated read ports. Although elegant, this implementation



(a) 6T configuration.



(b) 8T configuration.



(c) 10T configuration.

Figure 2.11: Different SRAM bitcell configurations.

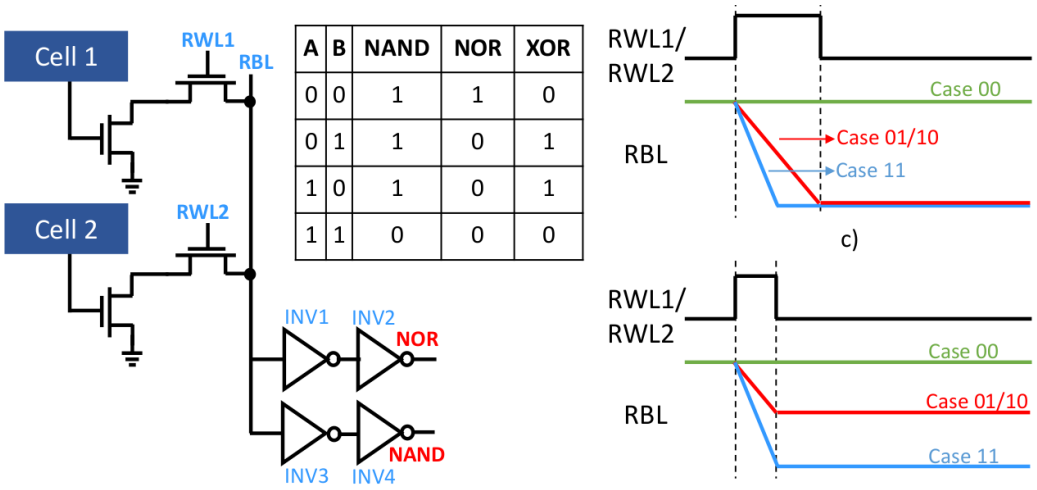


Figure 2.12: 8T BLC configuration and sensing scheme (left) and WL pulse width reduction waveforms for result identification (right) [15].

is strongly technology dependant and process variations sensitive. Fig. 2.14 shows the result of write margin simulation for bulk and DDC technologies. As stands out from the plot, bulk technologies severely deteriorate the system's reliability due to their high process variations and low body factor. This translates into the major issue hindering a practical compiler implementation of the architecture. On top of this, the proposed writing scheme requires supply voltages higher than the nominal one, thus giving rise to the need for level shifters and charge pump circuits which severely increase the system's area occupation.

2.2.3 Interleaved bitlines

A further approach to solving the read-disturbance issue for bitline computing focuses on the topological aspect of the computation instead. As previously stated, the disturbance arises from sharing the same BL for the accessed bits, resulting in a conductive path between their storage nodes. One way to avoid this connection is to split the BLs, as proposed in [16]. Instead of stacking weights and activations words vertically in the array, the two are *interleaved* as shown in Fig. 2.15a. This way, instead of asserting two rows of the array, only a single one is driven, thus

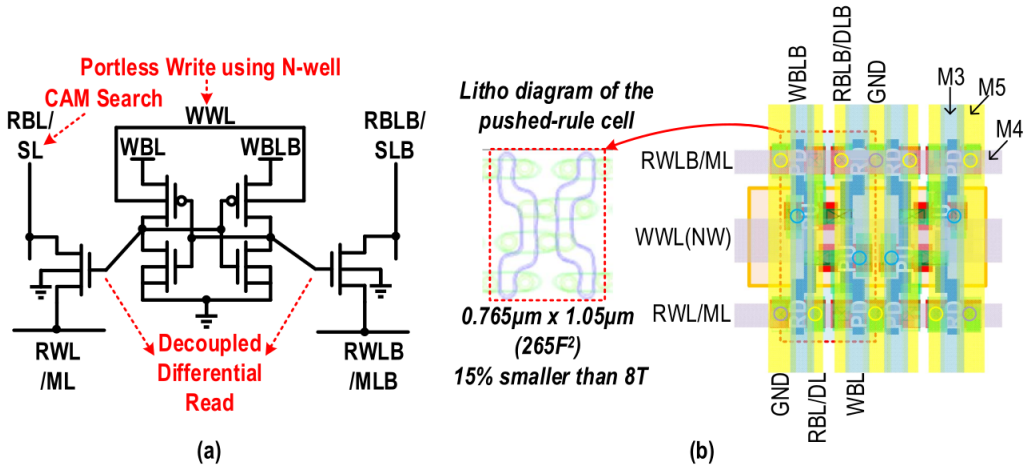


Figure 2.13: 4+2T bitcell configuration schematic (a) and layout (b) [26].

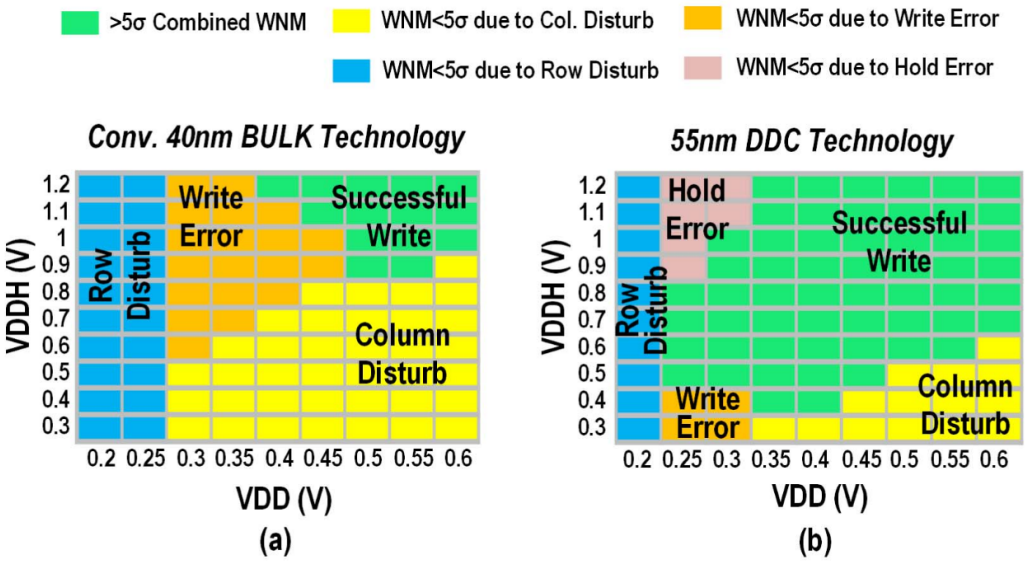


Figure 2.14: Results of write margin measurements using the proposed n-well scheme [26].

exhibiting a doubled number of bitlines. The avoidance of BL sharing vanishes the bitline computations, due to the intrinsic absence of a common node. This led the authors to address these missing operations to an additional stage in the column sensing circuitry. The outputs of two adjacent columns are sensed by individual SAs and sent to a logic circuit shared by the two amplifiers; this processes their results yielding the final computations (Fig. 2.15b). The ultimate result is a computing architecture behaving like a bitline computing one without effectively performing any computation in the single bitline, hence solving the read-disturbance issue.

Implementation results of this approach show an area increment of the array up to 11% with respect to a standard 6T SRAM, versus the 31% of an 8T array [27]. This is to be addressed to a stretch in the bitcell's layout, required by the presence of a further via for parallel wordline routing (Fig. 2.16). The resulting architecture is, therefore, free from read-stability issues typical of bitline computing; thus resulting in a viable adoption according to the purposes of this work. Another selling point is its pronounced design regularity, a strong advantage when dealing with memory compilers. One drawback of this implementation is its low operating frequency, which settles to 50MHz for a 64-bit word size and a 65nm technology node. The authors didn't investigate the sources of the delay overhead which may be due to the popcount network as well as an un-optimized computing circuit downstream of the SA array. Ultimately, the interleaved words arrangement severely reduces the flexibility of the system. In a standard bitline computing array, the communication between different cells of the same column is conditional: due to a shared line (BL), the user is free to choose between all the possible combinations of cells (and, by extension, words). In an *i-SRAM* instead, the communication is constrained to a single combination of two words, due to the hardwiring of the compute logic. However, a detailed study of this issue is missing. One possible implication would be the need for storing multiple times the same word in different locations for its usage with different operands. The ultimate result is data redundancy, leading to degraded storage capabilities.

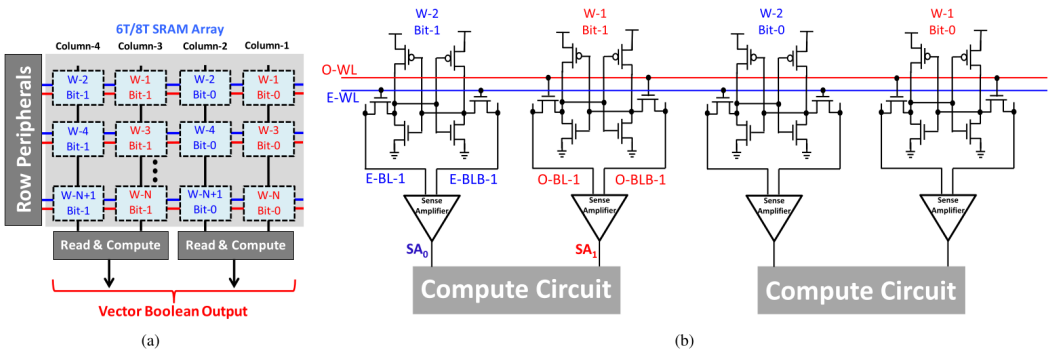


Figure 2.15: Interleaved wordlines array configuration (a) and compute circuit (b) [16].

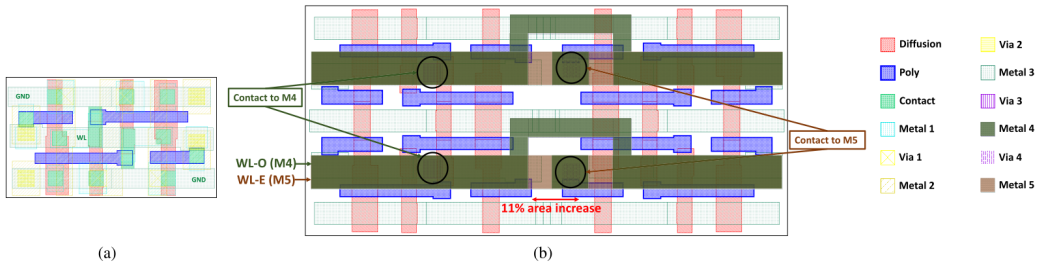


Figure 2.16: Bitcell's layout of a standard 8T configuration (a) and its stretched version for interleaved wordlines. [16].

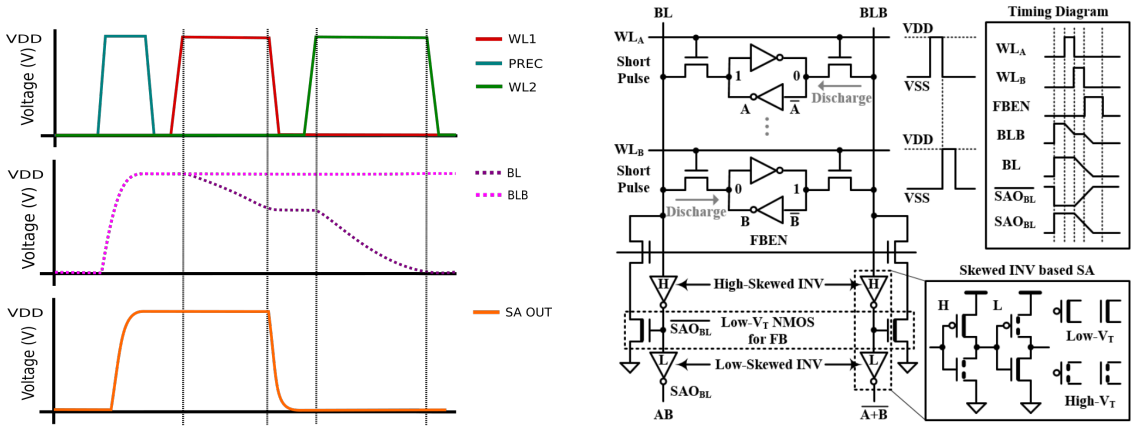
2.2.4 Sequential pulsed wordlines

One last approach to solving the read-disturbance issue is to adopt a pulsed WL assertion. This final technique is based on a change of conceiving the timing involved in bitline computing. A constant feature of forenamed proposals was the simultaneous access of two words; relaxing this assumption leads to sequential accessing the operands, avoiding their concurrent presence on the BL.

The first draft of this concept can be found in [15] and is exploited to unlock safe bitline computing in 6T arrays. A straightforward implementation of this scheme would result in two consecutive pulses sized to completely discharge the BL when reading a logic 0 from a cell. However, this leads again to read-stability issues when accessing the second bit. If two cells storing different values are accessed sequentially, the second one experiences its storage nodes shorted to BL and BLB. These

lines will therefore exhibit complementary values with respect to the ones of the cross-coupled inverters, potentially corrupting the content of the cell. Accordingly, the authors shrank the pulse duration to obtain a safe BL voltage after the first pulse. This way, the second cell will be shorted to a node whose value ($\sim V_{DD}/2$ in the proposal) is above the inverter trip voltage, thus avoiding flipping. Waveforms referring to sequential pulsed WL assertion can be found in Fig. 2.17a.
[28] further elaborates the concept of sequential pulsed WLs by delivering a dedicated single-ended sensing scheme based on skewed buffers. As soon as the first pulse ends, the BL reaches an intermediate voltage which slowly drives the switching of the first inverter of the chain. This latter's output node is the gate of a further nFET having its drain connected to the BL through a pass-transistor in a bootstrap fashion. When the second pulse has been issued, an enable signal shorts the feedback FET to the BL, completing its discharge (Fig. 2.17b).
Authors in [28] carried on a comparative analysis of the sequential WL assertion scheme and the wordline-underdrive (WLUD) technique. This latter consists of driving the bitcell's access transistor with a voltage lower than V_{DD}, thus increasing the impedance of the connection between the storage nodes and the BL. However, this leads to a degraded discharge rate of the BL which slows down the system. The study shows how the read access time experienced a 62.2% reduction with the proposed scheme at the expense of a 2.75% increment in energy consumption. The ultimate result is a 61.2% reduction of the energy-delay product for read/compute operations.

Sequential access is not the only fashion in which pulsed WL is exploited. [29, 30, 24] adopt this concept while simultaneously accessing the two words to limit the BL swing. The pulse width is constrained to the minimum required to make the SA switch if a discharging event of the line occurs. This technique aims at achieving a low BL voltage swing to avoid turning on the pFETs of cells storing a logic 1. Their conduction depends on their drain-source voltage, which translates into the difference between the potential of the storage node (V_{DD}) and the BL. As soon as this value exceeds the threshold voltage of the transistor, the device starts sourcing current, thus resulting in a conductive path from V_{DD} and GND when stacked with the nFET that's discharging the BL.



(a) (b)

Figure 2.17: Sequential pulsed wordlines waveforms (a), bitline boosting schematic for pulsed wordlines bitline computing (b) [28]

2.2.5 Discussion

This section provided a brief summary of the literature relating to digital architectures for BNNs acceleration. As stressed out in this study, the area efficiency of the SRAM array plays a crucial role in the architecture's performance, thus preferring a complex design to an increment in the bitcell's area occupation. This is to be addressed to target applications of IMC, mainly consisting of internet-of-things (IoT) nodes to enhance with artificial intelligence (artificial intelligence-of-things, AIoT), subjected to tight area and energy consumption constraints. A further aspect standing out from the analysis is the issue of read-disturbance, affecting all the 6T SRAM based IMC implementations. This affects the system's reliability, resulting in increased design time, failure rate and expertise required by the designers; translating into major factors retaining the deployment of IMC in a wider range of applications. Sequential pulsed WLs potentially represent a viable adoption to solving the read-stability issue, allowing the exploitation of a low-area 6T array. However, a full system implementation is missing, hindering a robust comparison with existing architectures. The next chapter will deal with the design of an IMC SRAM architecture based on BLC. Each section will introduce a building block of the system and define its architecture choosing among the features described in this

section.

Chapter 3

Democratizing hardware design

Information and communication technology (ICT) has changed our world. Since the invention of the first transistor, physical computing systems have been growing in complexity and performance, propelled by Moore's law. As the number of features of an electronic system increases, so does the expertise, resources and development time required by its design, thus constraining it to large industries and engineers teams. Commercial electronic design automation (EDA) tools have been supporting this growth for the last decades; however, the significant expertise required by current technology nodes is raising their cost, as well as the ones of the overall system's design and verification (Fig. 3.1). To make the design of new hardware devices quicker, easier and more accessible, in the last few years a new movement emerged under the theme of "democratizing hardware design". One key exponent of this is the OpenROAD Project [31, 32]. Launched in June 2018, it aims at bringing down barriers to circuit design by making system prototyping and manufacturing easier, for both experts and non-experts, even in advanced technologies. Accordingly, it provides a fully automated open-source "no human in the loop" hardware design flow, trying to unlock a 24-hour design time for complex IPs. The project is supported by a team of performers including members from Arm, Qualcomm and several universities such as UC San Diego and the University of Michigan. Currently under development, the flow is based on an open-source toolchain for the different steps of the VLSI design stream, providing facilities for a tapeout-ready result.

Several EDA tools, however, have to be aware of the targeted technology for the silicon implementation of the design. Accordingly, they are fed with foundry process design kits (PDKs), providing models for back-end digital design steps, SPICE

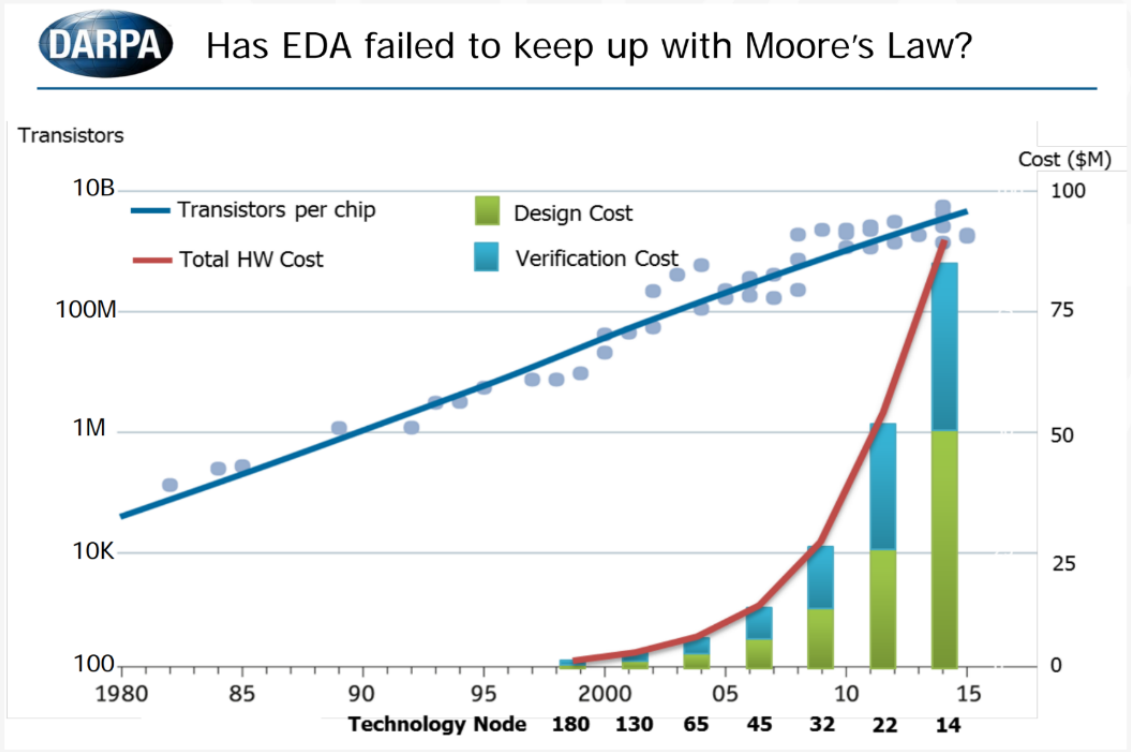


Figure 3.1: Design technology crisis [31].

simulations, parasitic extractions, design rules and process variations. Commercial PDKs are expensive, while freely available ones are either stuck to very old process nodes or non-manufacturable. To meet the need for a reliable open-source PDK, Google and Skywater teamed up for the release of SKY130 [33], a completely accessible 130nm process design kit for product development.

One last crucial aspect in electronic systems design is the development of memory arrays. An open-source flow for modern systems on chip (SoCs) has to take into account the presence of hard macros in the system like cache memories and SRAMs. Due to their pronounced layout regularity, the design of these elements is often carried on by memory generators, allowing users to benefit from a fast generation of silicon proved arrays. Due to dedicated layout solutions, most memory memory generators are strongly technology-dependent, while most open-source PDKs lack an array generator implementation. Several commercial tools instead, yield black-box instances or, however, do not allow customization of any of the memory components.

According to this issue, the OpenRAM Project [34] aims at delivering a customizable memory array design framework. Thanks to the presence of technology files, the compiler can be ported to different technologies while limiting the performance loss.

As previously stated, this work will deal with the compiler-aided open-source design of a computing SRAM architecture. Accordingly, this chapter introduces the main tools exploited to fulfil this purpose; the first section will deliver a brief overview of the OpenRAM framework structure, while the second one will introduce key properties of the Skywater 130nm PDK.

3.1 OpenRAM: an open-source memory compiler

The OpenRAM memory compiler is a memory design framework under development at the University of California Santa Cruz. The script follows an object-oriented programming approach and is completely based on Python, sticking to the philosophy of an open-source design flow. The framework can be divided into two main phases, a front-end and a back-end methodology (Fig. 3.2). The first is the effective generation of the array, starting from a user-defined configuration file (specifying word size, number of words, number of banks, etc) and producing SPICE, Verilog, GDSII and Liberty files, with estimated power and delay values resulting from analytical models. The back-end methodology is instead exploited by OpenRAM to yield a true and precise characterization of the architecture. It relies on external tools for simulation (e.g. ngspice) and parasitics extraction (e.g. Calibre), resulting in a timing and power annotated Liberty file (.lib).

3.1.1 Module structure

Each module of the design is derived from an ancestor class (namely `design.py`), which holds name, SPICE and layout information about the device. Each instance is further represented by some derived classes: `hierarchy_spice.py` and `hierarchy_layout.py`. These supply a set of methods for handling SPICE file

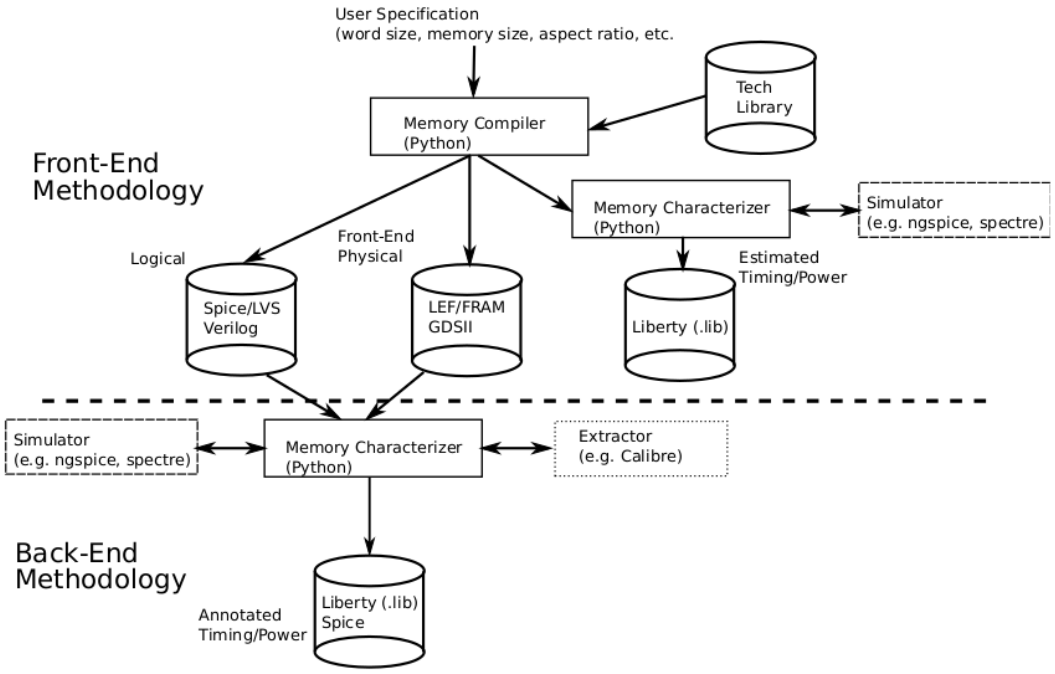


Figure 3.2: The OpenRAM software framework [34].

reading/writing and layout geometries drawing respectively.
OpenRAM relies on two types of modules: "hard" and parametrized ones. Formers have to be fed to the compiler both in SPICE and GDS files, resulting in the most customizable instances of the array. Belong to this category sense amplifiers, write drivers, flip-flops and bitcells. The exploitation of hard macros allows the designer to optimize the system performance as well as the array density. Latters instead are dynamically generated by the compiler and their sizing is made dependent on a set of input constraints. Belong to this category a whole set of logic gates primitives, including inverters and buffers.

For a new module to be instantiated, the primitive class has to be derived to create a new one, thus inheriting SPICE and layout file handling methods. Each module will have its constructor and might be customized with a set of input parameters if necessary. Concerning the SPICE file, it is loaded for the custom device and has to agree with the declared set of input/output/power pins during the module generation. Further functions are then invoked to connect these pins to the ones of

other modules to compose a netlist. The same thing applies to the layout view of the new device. Concerning geometries, these are stored in a software representation through shapes. Each shape has a given layer, net name and set of coordinates. More or less complex methods are present for an optimized drawing of the layout, speeding up processes like bus connection. Once the compiler ends the array generation, some special functions are invoked for translating the set of fictional shapes into effective geometries in a GDSII file. These functions belong to a dedicated library for GDS manipulation named GdsMill, developed at the University of Michigan. For hard modules, this library is exploited to load the layout view and translate it into fictional shapes for easier handling. Furthermore, the `hierarchy_design.py` side of the module provides methods for instance placing and pins position extraction. The global class in charge of constructing modules and instances is called *factory*.

3.1.2 Technology files

As stated previously, OpenRAM is, ideally, able to efficiently compile memory arrays on the majority of CMOS process nodes. This is achieved via a technology file called `tech.py`, which holds key characteristics of the given PDK. The file starts with a map that links GDS layers and planes to a more user and compiler friendly format, where the name of the layer is explicit and recalls its effective purpose. Then, design rules are listed for each layer, allowing the compiler to produce a violations free layout by issuing checks on the values passed to the methods. To ease the routing phase of the layout generation, each metal layer and the polysilicon has a user-defined preferred direction. By alternating these directions, tracks of nearby layers are made orthogonal, hence reducing their capacitive coupling. Another type of data structure is represented by the metal layer stacks. Stacks are arrays containing the name of two nearby layers in the hierarchy and the name of the via connecting the two. These structures are exploited by OpenRAM to readily switch layers during the routing as well as to identify which layer to be used for the supply grid.
To allow the automated generation of SPICE netlists, the technology files stores the device's model names and minimum allowed sizes for both pFETs and nFETs. Moreover, to unlock a parametric sizing of certain logic gates (e.g. drivers) the

values of FETs drain and gate capacitance are listed, together with delay and power parameters for the system's characterization. Finally, the file provides names and paths to auxiliary tools exploited by the compiler for parasitics extraction (PEX), design rules check (DRC) and layout versus schematic (LVS) correspondence.

3.1.3 Architecture overview

The reference implementation is an SRAM memory array, whose simplest block scheme architecture can be seen in Fig. 3.3. Among the options to be defined by the user one can identify:

1. number of read, write or read-write ports;
2. local array size for local bitlines;
3. write size for masked write operations;
4. number of banks (currently only one bank is supported);
5. words per row of the array;
6. whether to provide a ring-type supply network or not.

The possibility of exploiting isolated ports for reading or writing only is subjected to the availability of customized bitcells. OpenRAM can automatically generate a layout view for a standard 6T bitcell on any implemented technology; however, it would not translate into a minimum area occupation design due to the imposition of a DRC violation-free result. To enhance the array efficiency, the authors promote making the bitcell a hard-macro, thus requiring a dedicated design. Other custom modules include sense amplifiers, write drivers and flip-flops. Their layouts have been designed from scratch for SKY130 and can be found in App. A.

Concerning input-output signals, currently, the architecture relies on the followings:

1. **ADDR** for address bits;
2. **DIN** for input data bits;
3. **CLK** for the system clock;
4. **WEb** to select between read or write operation;
5. **CSb** to trigger the start-up of the device.

Accordingly, one obtains the timing diagrams for read and write operations shown in Fig. 3.4 and Fig. 3.5 respectively.

3.2 SKY130: Skywater 130nm process design kit

SKY130 is a hybrid 180nm-130nm technology node developed by Cypress Semiconductor and is now available through Skywater Technology Foundry. It is based on the 8th generation SONOS process node. As can be seen in Fig. 3.6, the process consists of 5 levels of metal and a layer of titanium nitride (TiN) for local interconnections. The feature size of the technology is 130nm, however, the minimum channel length for transistors is larger, resulting in 150nm. The minimum channel width is limited to 420nm, yielding a minimum aspect ratio of 2.8. The nominal supply voltage for the process is 1.8V, however different libraries are present to allow the interfacing with voltage domains up to 20V.
The content of the PDK is organized in libraries, each exhibiting different properties and trade-offs between power, performance and area:

1. **sky130_fd_sc**: foundry-provided (fd) standard cells (sc) libraries for digital design:
   (a) s**ky130_fd_sc_hs**: high speed, 0.48 x 3.33um site (about 11 met1 tracks);
   (b) **sky130_fd_sc_ms**: medium speed, 0.48 x 3.33um site (about 11 met1 tracks);

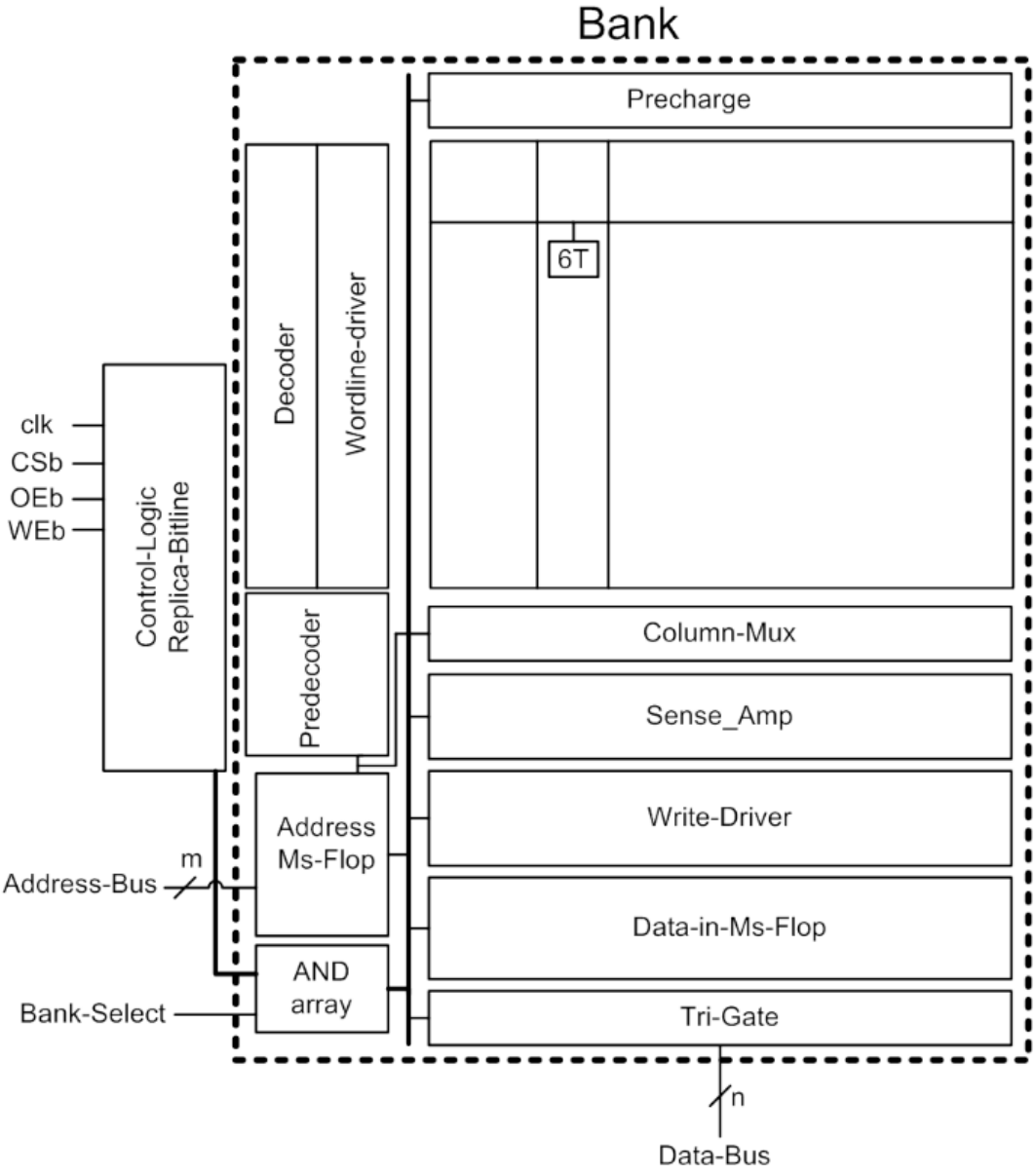


Figure 3.3: Block scheme view of the OpenRAM SRAM architecture [34].

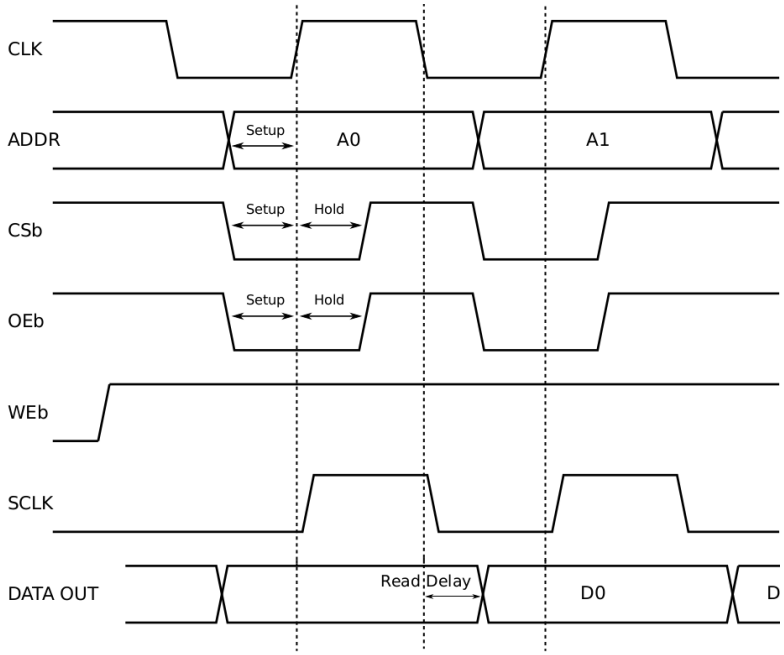


Figure 3.4: Timing diagram for OpenRAM read operation [34].

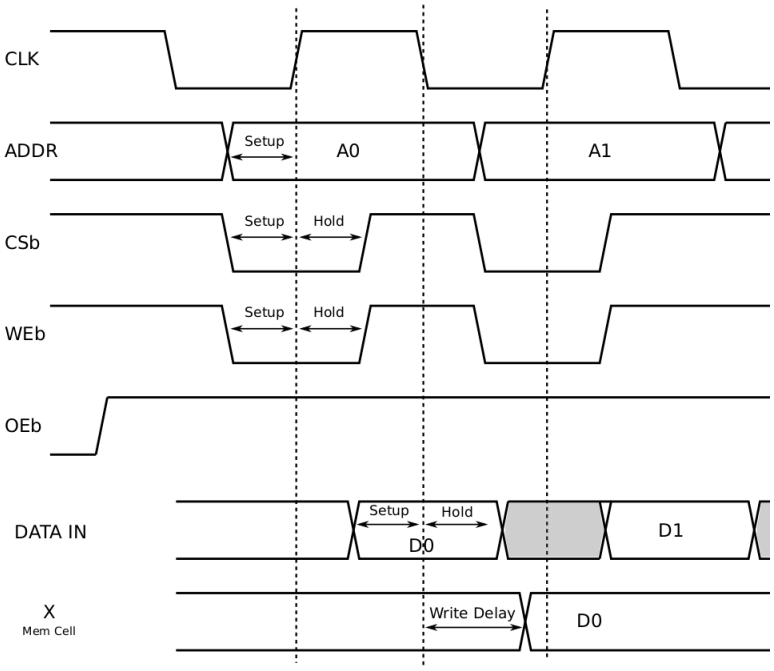


Figure 3.5: Timing diagram for OpenRAM write operation [34].

   (c) **sky130_fd_sc_ls**: low speed, 0.48 x 3.33um site (about 11 met1 tracks);
   (d) **sky130_fd_sc_lp**: low power, 0.48 x 3.33um site (about 11 met1 tracks);
   (e) **sky130_fd_sc_hd**: high density, 0.46 x 2.72um site (about 9 met1 tracks);
   (f) **sky130_fd_sc_hdll**: high density, low leakage, 0.46 x 2.72um site (about 9 met1 tracks);
   (g) **sky130_fd_sc_hvl**: high voltage (5V), 0.48 x 4.07um site (about 14 met1 tracks).

2. **sky130_fd_pr**: foundry-provided device primitives (pr), stores SPICE models of elementary devices like FETs, BJTs, resistors, capacitors, inductors, etc;
3. **sky130_fd_io**: foundry-provided input-output (io) and periphery cells, stores macros and SPICE models of peripherals like GPIOs and pads;
4. **sky130_fd_bd**: foundry-provided build spaces (bs), stores macros for special purpose applications like SRAMs and flash memories (none of them has been released yet).

The presence of such different libraries broadens the design space of target applications, allowing full-chip tapeout-capable implementation. Although wide, this PDK is not intended for commercial purposes and is to be intended as an experimental preview. Different designs have been manufactured and tested, however for prototyping purposes only. Each new batch of chips helps the developers fix bugs and model properties for the improvement of the design kit. Due to the open-source nature of the PDK, Skywater is open to collaborations to enrich the content of the libraries. New build spaces are currently under development like the one for SRAM arrays, featuring shrunken transistor sizes and optical proximity corrections for increased density.

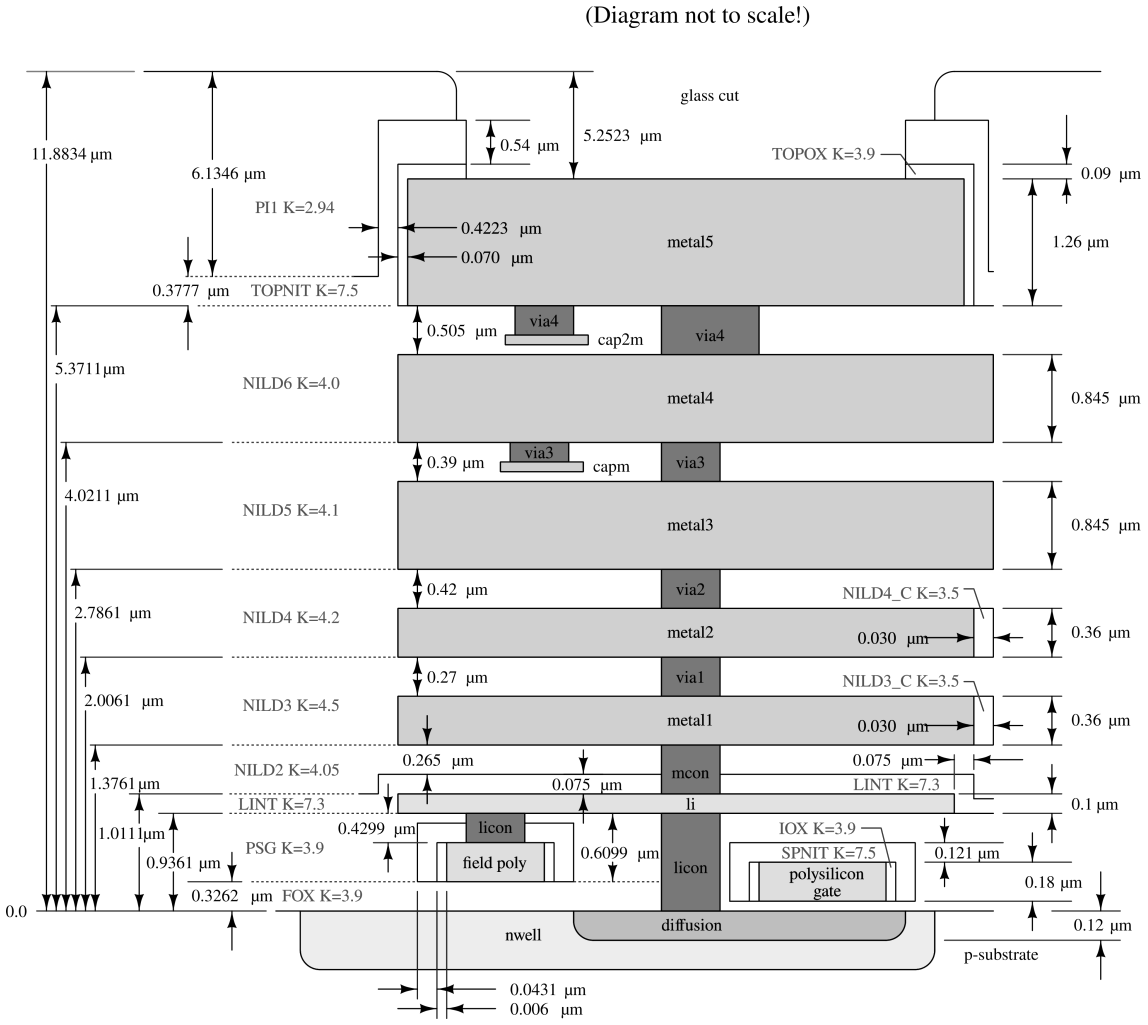


Figure 3.6: Process stack diagram for SKY130 [33].

Chapter 4

Architecture concept and design

Based on what has been previously discussed, an architectural concept of the computing SRAM to be implemented in SKY130 can now be investigated. Since, as previously stated, the aim is to exploit OpenRAM for the macro development and to introduce IMC features in a memory compiler, enhancements have been picked among the proposals to be least-invasive as possible, this for both preserving the compiler-introduced native optimizations and reducing the amount of additional code.

For the sake of the digital nature of this implementation and its target applications consisting of BNN accelerations, the fundamental computing scheme would be based on *bitline-computing* (BLC), given this one can highlight the building blocks of the memory architecture to be customized or designed from scratch:

1. **Bitcell Array**: needed for data storage, its density is determined by the type of bitcell (6T, 8T, 10T, Custom, etc) and the associated layout, resulting in the major area occupation source for an SRAM design.
2. **Address Decoder**: for BNN inference acceleration two addresses are needed at once: one for pointing at the word storing the weighs related to a neuron and pointing at its activations values.
3. **Sense Amplifiers Array**: probably the most crucial cell in an IMC memory implementation, it is in charge of unlocking key features for inference acceleration while keeping area overhead and performance loss as low as possible.
4. **Population-Count Network**: while the weight-activation product is addressed to bitlines and sense amplifiers array, results accumulation has to be

performed by a specific component described in RTL and implemented through logic synthesis. With respect to the array's word size, its design could severely impact on the overall area occupation of the architecture as well as on its resulting cycle time.

5. **Control Logic**: Responsible for control signals generation and instances synchronization, has to be equipped with additional intelligence for a smart inference acceleration.

The resulting block scheme representation of the architecture can be seen in Fig. 4.1. In the following sections, each of the listed components will be defined, starting from the implementation choices and down to the circuit layout generation through the complete VLSI design flow's steps.

4.1 Bitcell array

As aforementioned, the choice of the bitcell array structure has a huge impact on one of the most important figures of merit when it comes to memory macros: area occupation; an often employed quantity in design's quality estimation is the *Area-Cost-Per-Bit (ACPB)*, consisting of the ratio between the area of the array and the total number of stored bits. The key factor affecting ACPB is the bitcell's type together with its layout, in fact, the area occupation of a single cell (i.e. the area required to store a single bit) is largely due to its transistor's count under the assumption of a tight layout: a 6T array will result in a higher area efficiency than an 8T one, and so on. Given the additional circuitry required for IMC features unlocking, a 6T implementation of the bitcell has been elected as the fundamental storage element to limit the area increment arising from array customization to the absolute minimum.
As aforementioned, the choice of the bitcell array structure has a huge impact on one of the most important figures of merit when it comes to memory macros: area occupation; an often employed quantity in design's quality estimation is the *Area-Cost-Per-Bit (ACPB)*, consisting of the ratio between the area of the array and

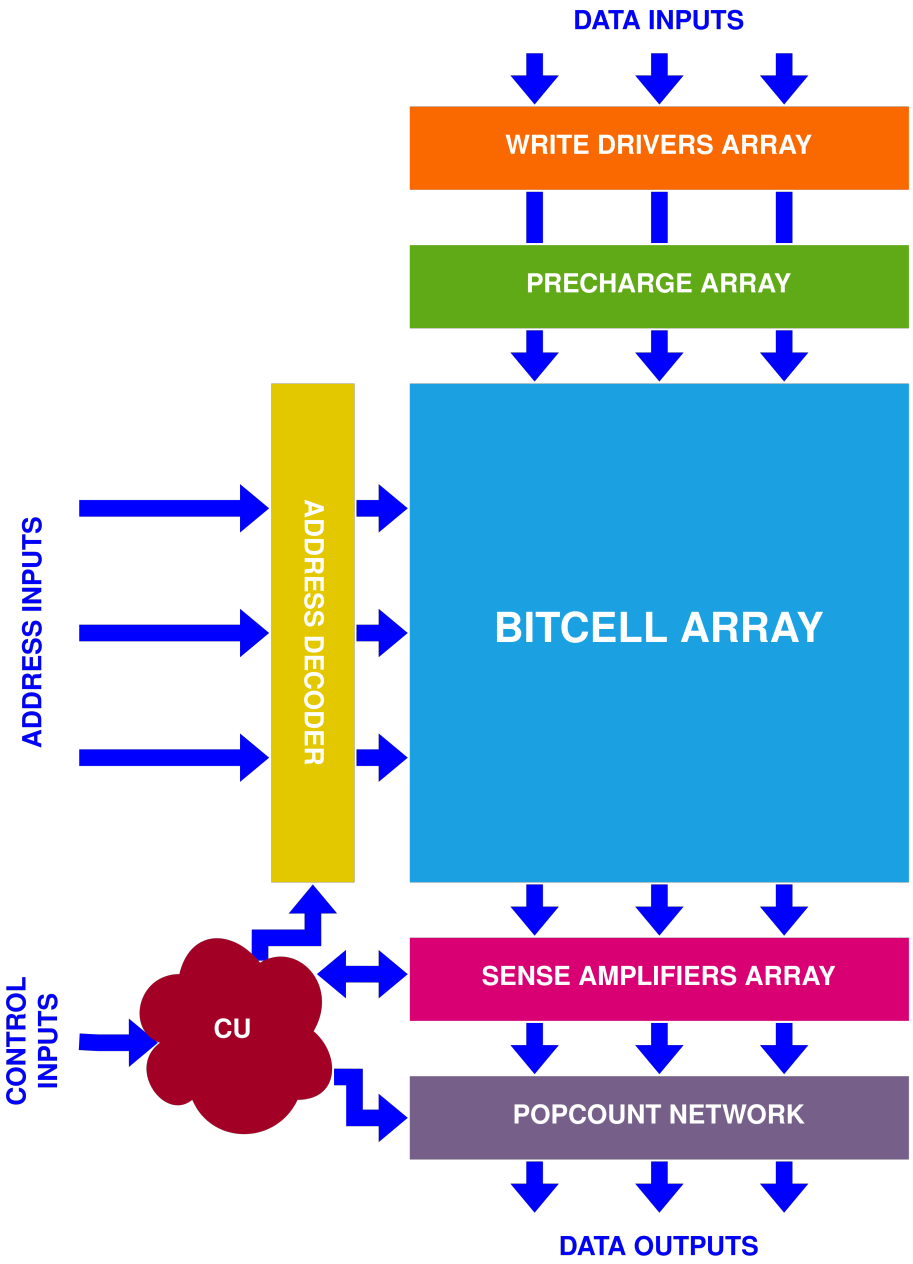
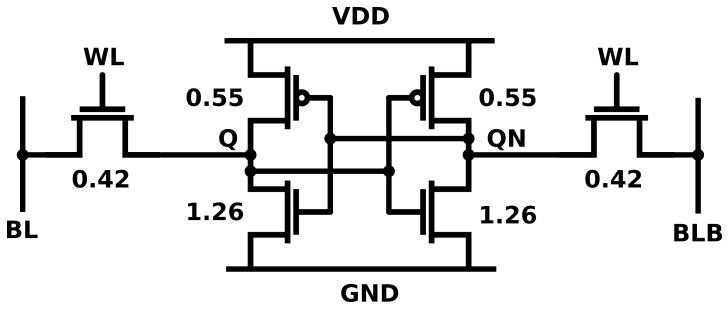


Figure 4.1: Primitive Block Scheme of the Computing SRAM Architecture.

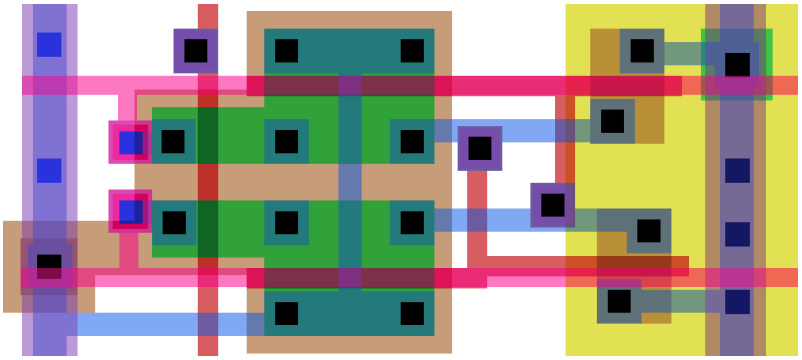
the total number of stored bits. The key factor affecting ACPB is the bitcell's type together with its layout; in fact, the area occupation of a single cell (i.e. the area required to store a single bit) is largely due to its transistor's count under the assumption of a tight layout: a 6T array will result in a higher area efficiency than an 8T one, and so on. Given the additional circuitry required for IMC features unlocking, a 6T implementation of the bitcell has been elected as the fundamental storage element to limit the area increment arising from array customization to the absolute minimum.
According to the overview performed in Sec. 2.2, the choice of adopting a 6T bitcell as a storage element limits the array's organization to two configurations: interleaved wordlines or a standard structure. As it was pointed out, wordlines interleaving leads to a loss in both storage efficiency and IMC operations flexibility. Accordingly, the structure of the array will stick to the one of a standard 6T SRAM, thus exhibiting a bitline pair for each column and a WL signal for each row.

Fig. 4.2 shows the schematic of the 6T bitcell (Fig. 4.2a), together with transistor sizes and the resulting layout (Fig. 4.2b), both taken as they are from an existing schematic/GDS database for an OpenRAM release ported to SKY130 [35]. The layout sticks to a conservative 6T bitcell design according to [36] while FETs are sized to optimize the cell for the well-known trade-offs between SNM, read-stability and writability. Due to the actual unpolished status of the PDK, designers are not allowed to draw transistor's geometries with a channel width smaller than the minimum one, which yields an aspect ratio of 0.42um/0.15um; this translates into strong limitations in area optimization of the array. Usually, due to the extreme regularity in the design of memory arrays, sub-min-size FETs are allowed and supported by lithographical enhancements such as OPC adjustments; however, Skywater is currently developing an SRAM build-space cell library to provide area optimized cells to improve array efficiency, which can be inserted in the compiler by simply updating the GDS and SPICE files. Fig. 4.3 depicts the layout view of a 2x4 array configuration in 6T cells.

Due to the choice of employing 6T bitcells as fundamental storage elements, the exploitation of simultaneous wordlines activation has to face the aforementioned



(a) Schematic view of the implemented 6T bitcell in SKY130. Annotated values are to be intended as channel widths in um, channel lengths always stick to the technology minimum and their value is omitted for sake of readability [35].



(b) Layout view of the implemented 6T bitcell in SKY130 PDK [35].

Figure 4.2: 6T Bitcell SKY130 Implementation - Schematic and Layout.

issues regarding read-stability, which may ultimately reduce the reliability of the system; to solve this issue this work will target a sequential pulsed WL assertion as introduced in the previous chapter.

4.2 Sense amplifier array

4.2.1 Computing sense amplifier

Arguably the most important component in a bitline computing-based IMC architecture, the sense amplifier combines the result of bitline computations to yield

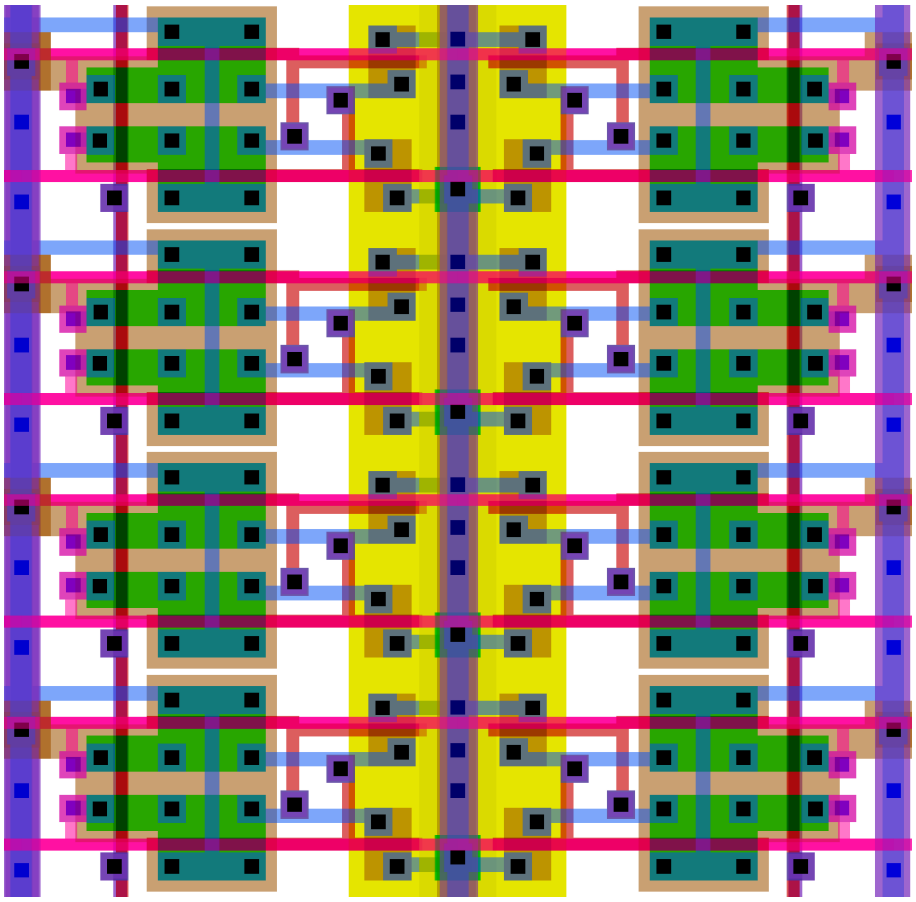


Figure 4.3: 6T Bitcell Array - Layout View of a 2x4 Configuration.

the required final results; these, being the final architecture designed for BNN and AdderNets acceleration, mainly consist of XNOR operation between the two accessed bits on the same column and a 2's complement addition.
After a double-word access, the BL will show a binary value equal to the AND between the two bits accessed sequentially in two cells belonging to the column, while the BLB the NOR. It is now sufficient to combine these two values through a further NOR operation to obtain the XOR value and complement it for the desired XNOR, which accounts for the BNN weight-activation product. Starting from this enhanced set of logic functions with respect to a standard sense amplifier, it is trivial to enrich the schematic of the device to allow for the computation of a 2's complement sum by inserting a further XOR gate for the sum output and the required carry-ripple logic.

Due to the presence of different values at the output of the SA, a multiplexer is required for the amplifier to be supervised by a control unit; according to this need, it has been decided to feed the MUX with two additional signals aside from XNOR and SUM ones: the BL value, needed for standard read operations and yielding the bitwise AND, and the complemented one (BLB), yielding the bitwise NOR. This latter choice is to be addressed to the AdderNets behaviour, in fact among the required operations one can identify the 2's complement subtraction and the absolute value computation, both requiring a 1's complement operation, hence a complemented read.

The resulting logic-level schematic of the *computing-sense-amplifier* (CSA) can be seen in Fig. 4.4a, where one can identify two instances of a sensing device, one for each BL. This is required because the differential sensing operation has to be replaced by two single-ended operations, thus requiring a duplication of the sensing component.

4.2.2 Circuit-level design of the BLC logic

The integration of logic gates in the SA increases the complexity of the device, thus rising the need for an accurate study on the circuit topology of each additional feature. Moreover, the computing amplifier has to undergo tight constraints concerning area occupation and propagation delay in order not to slow down the whole system.

Given the logic-level view of the SA, one can classify the propagation paths in the device as critical and fast. Due to the presence of carry-ripple logic, the timing of the array is now dependent on the word size of the memory and two data-paths can be highlighted in the single amplifier (Fig. 4.5): one related to the device's individual bitwise computations (i.e. AND, NOR, XNOR), the other resulting from the carry ripple logic. While for amplifiers in the middle of the chain these two paths can be considered independent (Fig. 4.5b), it is well known for Ripple-Carry Adders that in the first and the last elements the two combine, thus sharing a certain amount of gates (Fig. 4.5c and 4.5a) and requiring further analysis. In Fig. 4.5a one can see how for the first amplifier the circuitry shared between the two paths consist of a

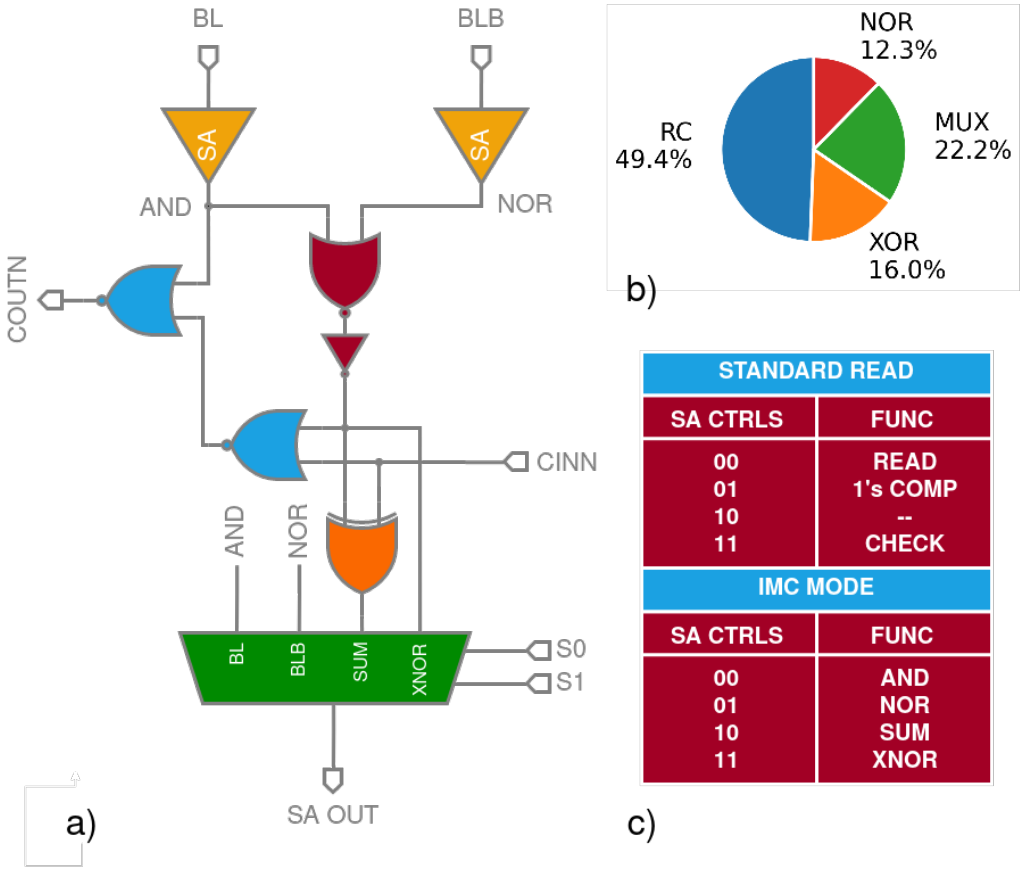
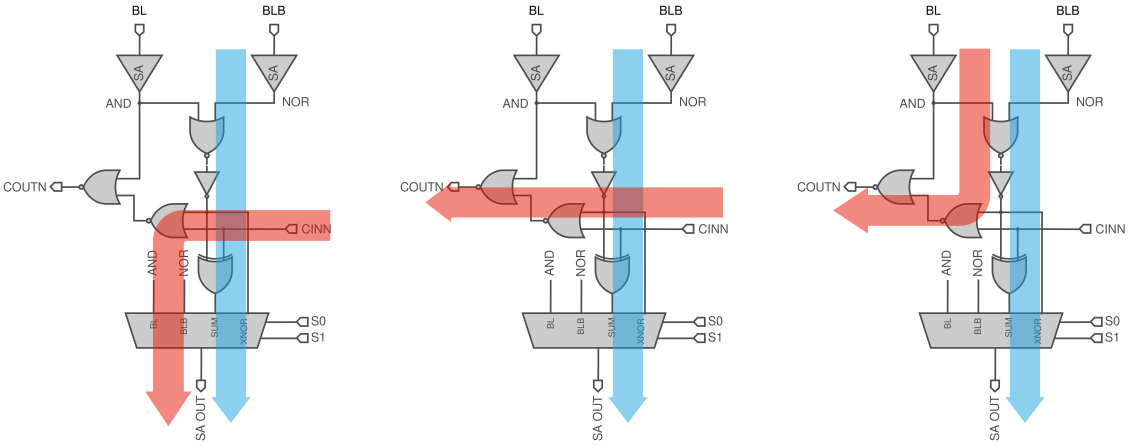


Figure 4.4: Computing Sense Amplifier schematic (a), post-technology mapping area occupation distribution (b) and mode control signals table (c).

NOR gate and a single inverter, for the last one, instead, the shared path results in an XOR gate and a 4x1 MUX. Analyzing the transistor's count of the shared circuitry in a pure static CMOS implementation one obtains the result in Tab. 4.1, where a significant area gap results between the shared gates of the first and last amplifier due to a complex circuit structure for XOR and MUX logic functions. Therefore, to keep the area overhead as low as possible, the NOR and INV related gates will be implemented in static CMOS topology, while XOR and MUX will be *transmission-gate* (TG) based; more in detail, the XOR gate will use the optimized circuit topology presented in [37] (Fig. 4.6a) while the 4x1 multiplexer will stick to a standard TG implementation as in Fig. 4.6b.



(a) Last CSA of the array. (b) Midway CSA of the array. (c) First CSA of the array.

Figure 4.5: Computing Sense Amplifier's critical (in red) and fast (in blue) paths.

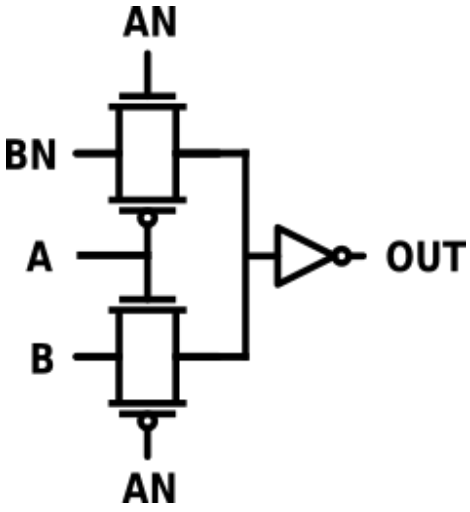
Table 4.1: CSA - Static CMOS Logic Functions Area Occupation

Area units	**Logic Function**			
	INV	**NOR**	**XOR**[a]	**MUX**[b]
min-FETs[c]	3	10	24	27
(μm^2)	0.189	0.630	1.512	1.702

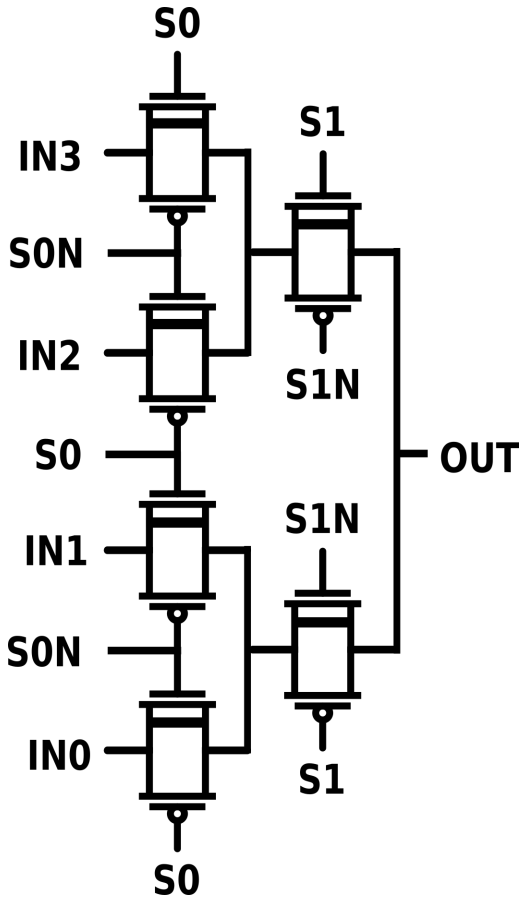
[a]Inverters for input signal generation are excluded.
[b]Inverters for control signal generation are excluded.
[c]SKY130 PDK - W_{min}=0.42um, L_{min}=0.15um.

4.2.3 Carry-ripple logic sizing

Formerly identified as the critical path in the CSA array propagation delay, the carry-ripple chain, whose circuit-level implementation has now been revealed, needs to be accurately sized in order not to degrade the system's performance during sum operations. According to this, propagation delay simulations have been performed for different word sizes and different drive strengths of the concerned logic gates. The first batch of simulations have been carried on imposing equal sizes for both the



(a) XOR gate schematic.



(b) 4-to-1 MUX gate schematic.

Figure 4.6: Computing Sense Amplifier - Transmission Gate Logic.

NOR2 gates, obtaining the results shown in Fig. 4.7a, where the trend for the 64-bits word length has been derived by interpolation of the lower parallelism's results according to the model in Eq. 4.1, which assumes equal delay for each additional amplifier in the chain; given the low values of approximation errors (Fig. 4.7b) this technique has been preferred to circuit simulations in order to ease the design process reducing the simulation time related to huge networks.

$$t_{pd}(n) = \frac{t_{pd}(8)}{8} \cdot n \quad \text{with} \quad n = \text{word-size} \tag{4.1}$$

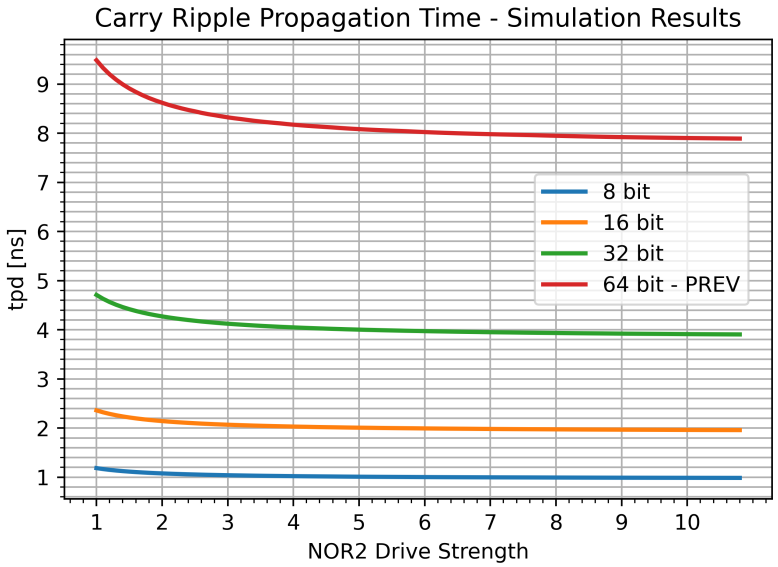
According to results in Fig. 4.7a-4.7c the obtained curve is monotone, hence no local minimum can be identified, meaning that as long as the drive strength of the gates

increases the propagation delay decreases, showing no self-loading behaviour and sticking to the ideal RC delay model for the aforementioned device. In order to increase the chances of discovering a local minimum, the assumption of equal-sized gates in the same amplifier has been relaxed and a new parametric simulation has been performed as shown in Fig. 4.7d. However, even in this scenario, a sweet spot was not found as the obtained 3D plot highlights how the minimum propagation times corresponds to an equal sizing of the two gates. Given these results, the only remaining design constraint for logic sizing is the area overhead introduced by increasing the drive-strengths of the devices. Fig. 4.7e depicts the relative delay reduction and area increment as function of the driving capabilities of the gates; given the obtained values, it has been decided to impose a drive strength for the carry-ripple logic equal to 2 in order to keep the area increment below the 50% while still obtaining a performance enhancement of 10% with respect to a unitary-sized logic chain. Resulting propagation delays for different word sizes are shown in Fig. 4.7f and final CSA logic area occupation and distribution with respect to logic function can be seen in Fig. 4.4

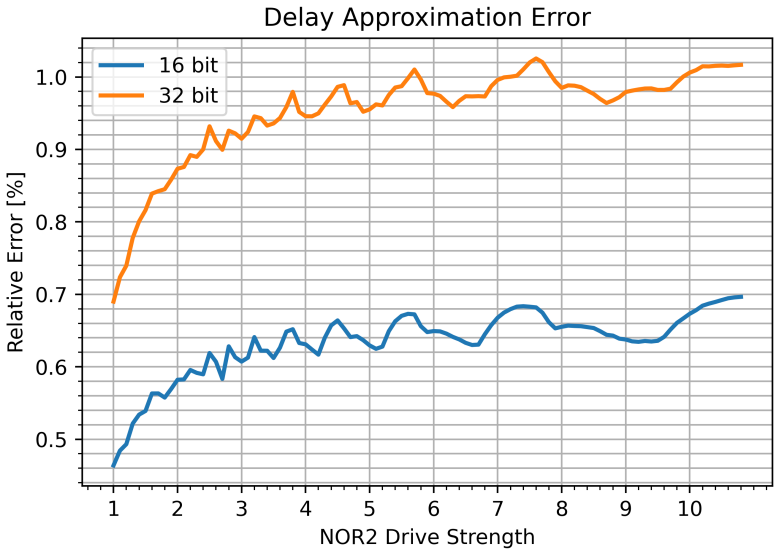
4.2.4 A novel circuit topology for single-ended sense amplifiers

In many previous works on IMC SRAMs, a standard implementation of the column sensing circuitry is used by simply duplicating the instance of a differential amplifier for each bitline pair. Since either a current latch SA (CLSA), voltage latch SA (VLSA) or differential stage (DS) is used, the area of the final bitline computing SA is large, mainly due to the transistors count of sensing components and their sizing for optimal read delay. Few works introduce single-ended sensing based on a high-skew buffer [29, 15]; being those designs optimized for speed though, they require very large transistor's aspect ratios and good control of the MOSFET threshold voltage, constraints that clash with the aim of a low area and technology independent compiler implementation.

To accommodate for the reduced area budget derived from logic insertion in the SA device, in this section a novel single-ended sense amplifier circuit topology will be

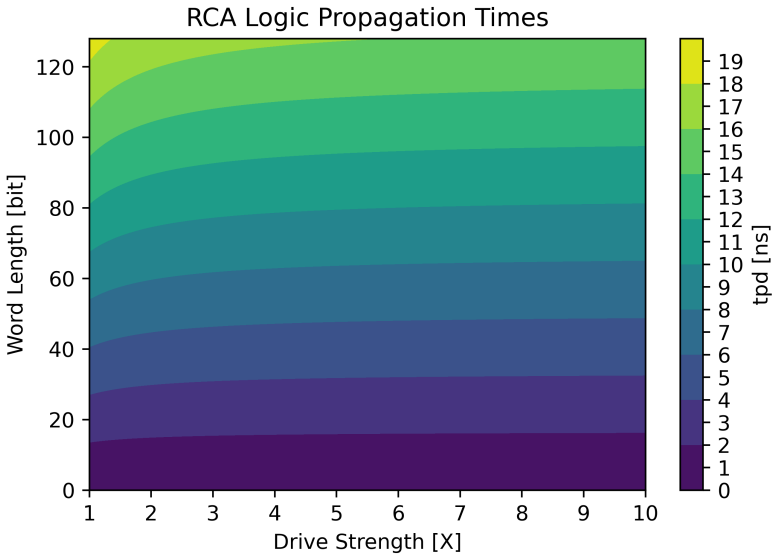


(a)

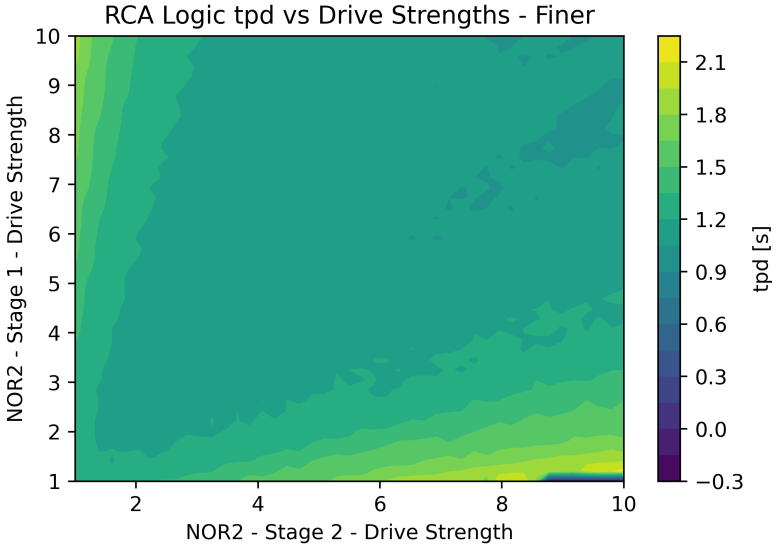


(b)

Figure 4.7: Computing Sense Amplifier - Carry Ripple Logic Sizing Results.

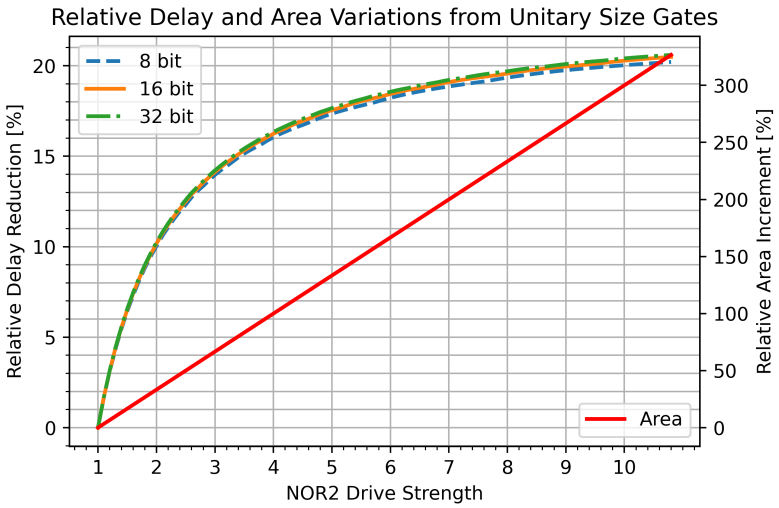


(c)

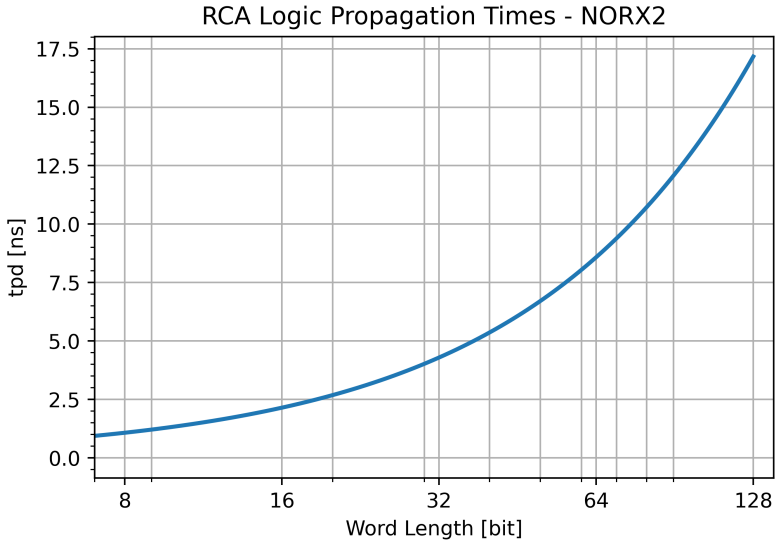


(d)

Figure 4.7: Computing Sense Amplifier - Carry Ripple Logic sizing result.



(e)



(f)

Figure 4.7: Computing Sense Amplifier - Carry Ripple Logic sizing result.

Table 4.2: Sense Amplifiers Area Comparison

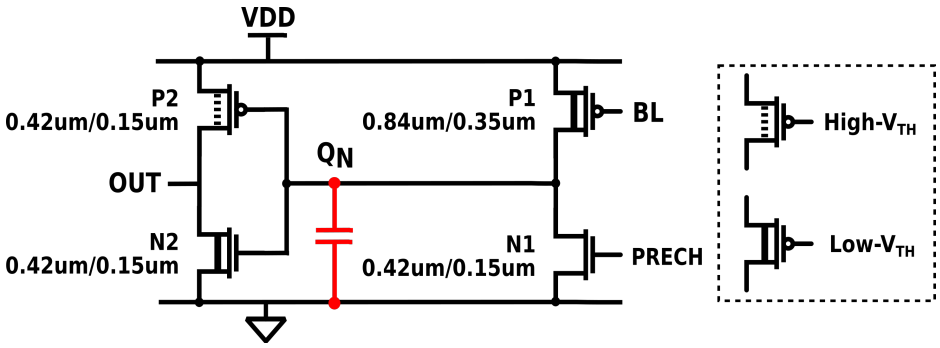
Area units	**Sense Amplifier Type**			
	Proposal	**VLSA**	**CLSA**	**DS**
(λ^2)	98.57	141.43	167.14	180
(μm^2)[a]	0.483	0.693	0.819	0.882

[a]SKY130 PDK - 0.15um channel length implementation.

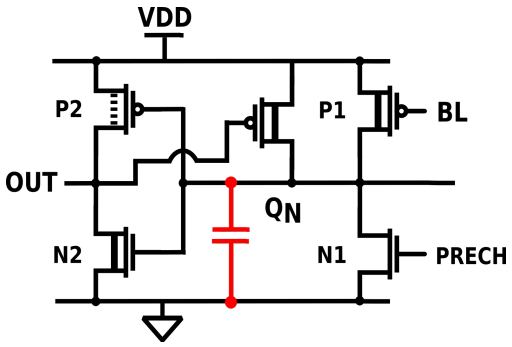
proposed.

Fig. 4.8a shows the area-optimized SA schematic, which consists of only two pMOS and two nMOS minimum sized transistors, apart from P1 which exhibits a larger width. The gate of the latter is connected to the bitline while its drain is Q_N, shared with the drain of N1 and the input node of a low-skewed inverter (P2 and N2). When the bitline is pre-charged to V_{DD}, P1 is off, Q_N is '0', and the output of the amplifier is '1', behaving as a buffer for the bitline value. As soon as the pre-charge signal is de-asserted and the bitline starts discharging due to the activation of a row in the array, P1 begins to turn on and slowly charges the capacitive node Q_N, as shown in Fig. 4.9a. When Q_N reaches the inverter's input threshold, the output of the amplifier goes to '0', yielding the correct read result. During the array's precharge phase, the node Q_N is discharged to '0' through N1, turned on by the signal PRECH. To enhance the amplifier's performance, P1 has been made a low-threshold voltage transistor, which speeds up its activation and ultimately reduces the charging time of Q_N. Increasing its aspect ratio will result in a larger charging current as well as a larger capacitance on Q_N, resulting in a trade-off for the design of the circuit. Tab. 4.2 shows a schematic-level area comparison between the proposed amplifier and minimum sized VLSA, CLSA and OpenRAM native DS where a significant reduction can be observed.

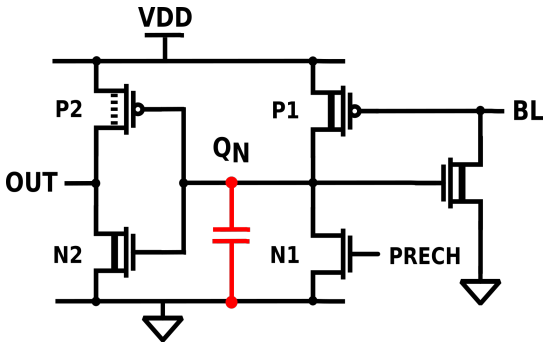
Based on the proposed sensing scheme one may enhance the circuit topology by inserting feedback bootstraps as in Fig. 4.8. In Fig. 4.8b a pFET is connected to the Q_N node with its gate driven by the amplifier's output, this boosts the charge of the capacitive node once the output voltage starts decreasing thus speeding



(a) Proposed SA schematic with explicit representation of equivalent capacitance on node Q_N.



(b) Amplifier's schematic with output boosting scheme.

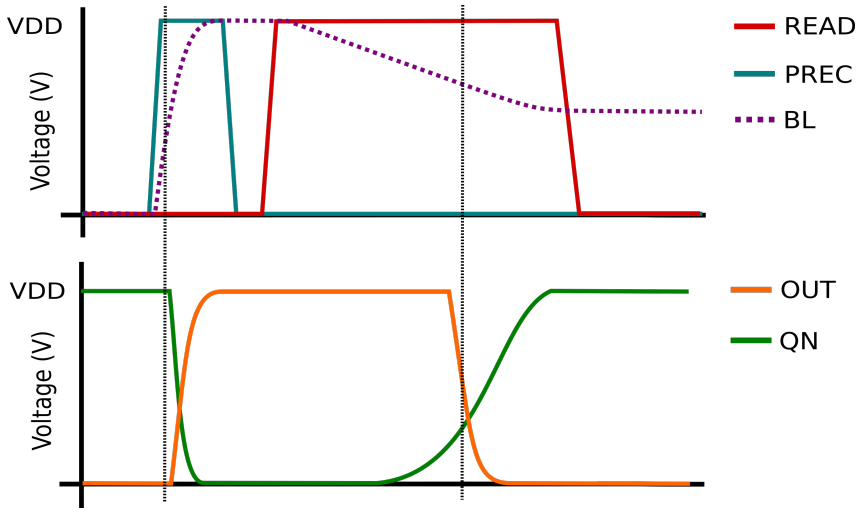


(c) Amplifier's schematic with BL boosting scheme.

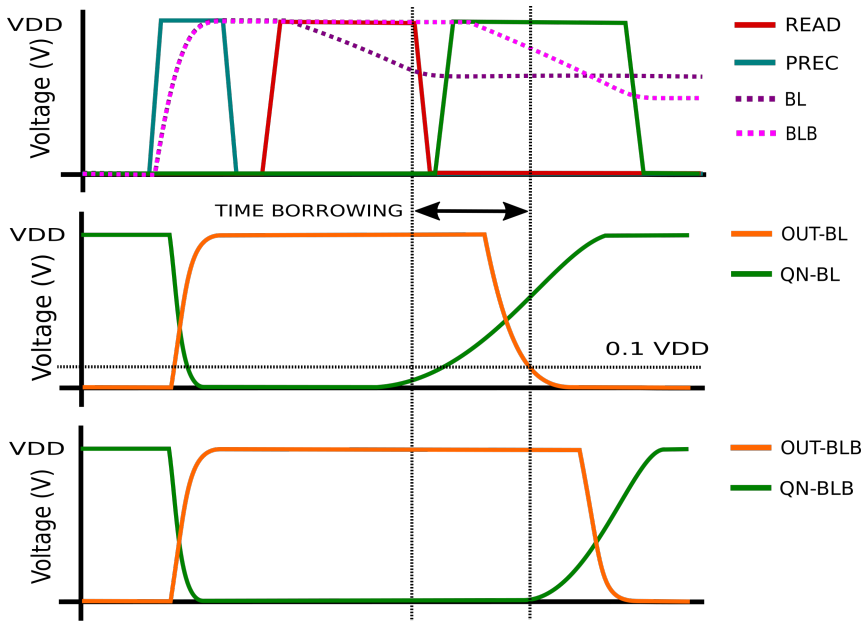
Figure 4.8: Proposed Single-Ended Sense-Amplifier's schematics.

up the amplifier's switching. Fig. 4.8c enriches the circuit with a bitline boosting scheme as in [28, 29, 30]; the nFET driven by Q_N turns on as the latter node voltage increases, thus boosting the discharge of the bitline, increasing the overdrive voltage of P1 and ultimately speeding up the charge of Q_N. This second-mentioned enhancement scheme, however, is not suitable for sensing arrays made up of 6T bitcells; the BL boosting nFET, once turned on, results in a further discharge path directly connected to the storage node, thus rising the same issues of an additional simultaneously accessed bitcell for what concerns read-stability. It could, en passant, represent a viable boosting scheme for all those arrays showing an isolated read port and requiring a full discharge of the BL.

Fig. 4.10 displays the results of a comparative simulation of different SA topologies in a 128x128 sized 6T bitcell array. Due to the strong dependency of their



(a) Proposed amplifier waveforms during standard memory read cycle together with control signals.



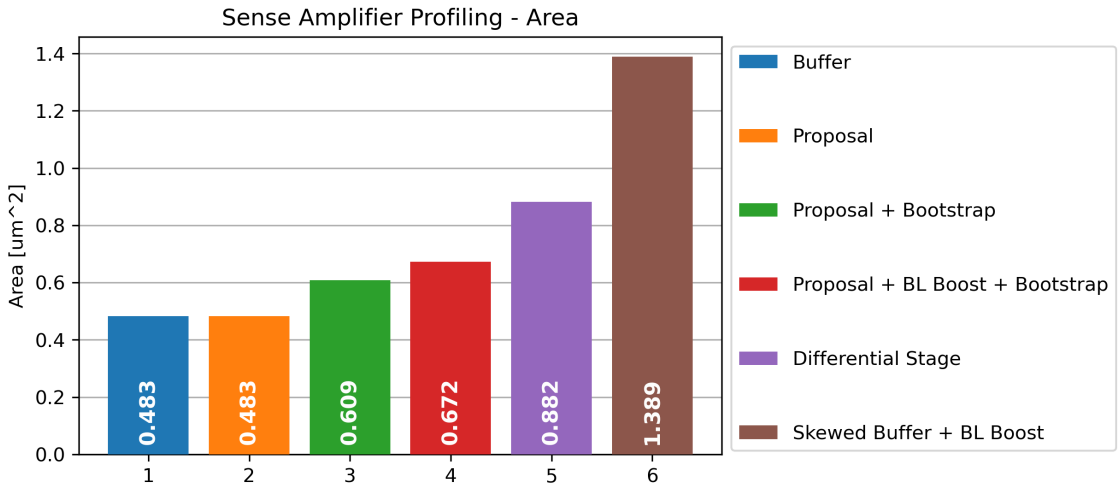
(b) Time borrowing waveforms during bitline computing cycle together with control signals.

Figure 4.9: Time-Borrowing Sense Amplifier Waveforms.

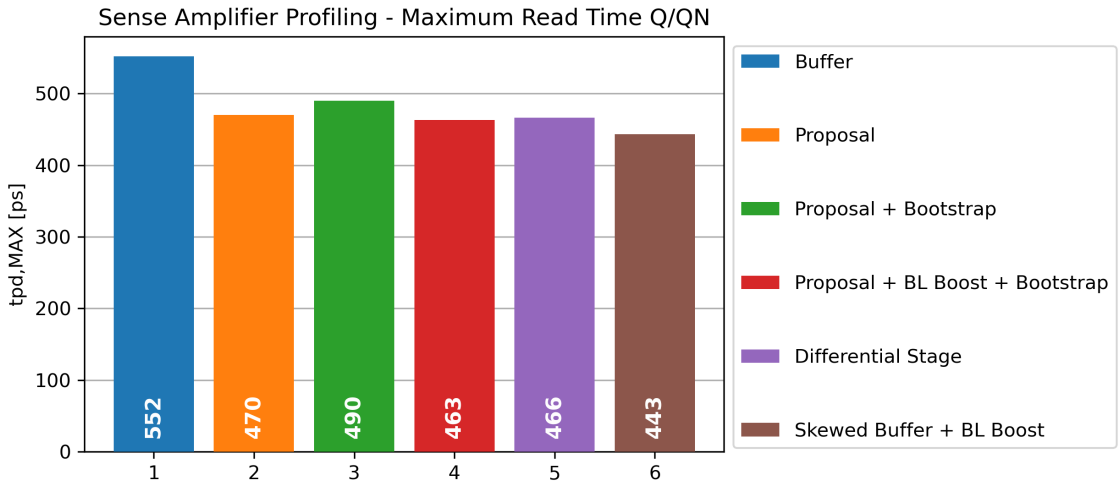
characterization from the generation of a well-calibrated sense-enable signal (SEN), VLSA and CLSA have been excluded from the simulation; DS-SA instead, despite requiring an enable signal too, does not rely on a latching scheme, and hence an inaccurate control signal generation only translates into an over-estimation of the device's power consumption. As stands out in Fig. 4.10a, the most competitive feature of the proposed amplifier is its area occupation, comparable with a min-sized buffer, way below the ones of standard SRAM's SAs. Another peculiar trait of this circuit topology is its read-delay (Fig. 4.10b), which competes with fast-access devices, like DS-SA and feedbacked BL-boosting skewed buffer [29], by degrading the performance merely by 0.9% and 6% respectively in its essential implementation. Nevertheless, when it comes to energy consumption, the proposal exhibits outstanding outcomes with respect to all the simulated competitors (Fig. 4.10c). In fact, thanks to the avoidance of a complementary pull-down/pull-up network simultaneously turned on by the BL signal (whose slew-rate is low for large arrays), the cross-conduction component of the CMOS gate current is neutralized, thus maximizing the charging current for the Q_N capacitive node. This translates into a 52% reduction in energy consumption with respect to the fastest of the simulated amplifiers; enhanced variants of the circuit, however, show a degraded energy saving at the expense of a reduced read-time, mainly due to transient cross-conduction currents during the pre-charge phase. Ultimately, the proposed amplifier exhibits excellent results when all the aforementioned figures of merit are taken into account at once (Fig. 4.10d), resulting in a solid pick in order to make up for the lowered area budget of the computing SA array.

4.2.5 Time-borrowing sense amplifier

Due to the adoption of a pulsed wordline assertion, a study of the amplifier's compatibility with this access scheme is crucial in order to obtain a reliable sensing scheme. As brilliantly identified and explained in [24] under the concept of *dynamic read disturbance*, the final value of the BL voltage plays a key role in the read-stability of the accessed cells, thus causing the need for a tighter design of the *pulse generator*

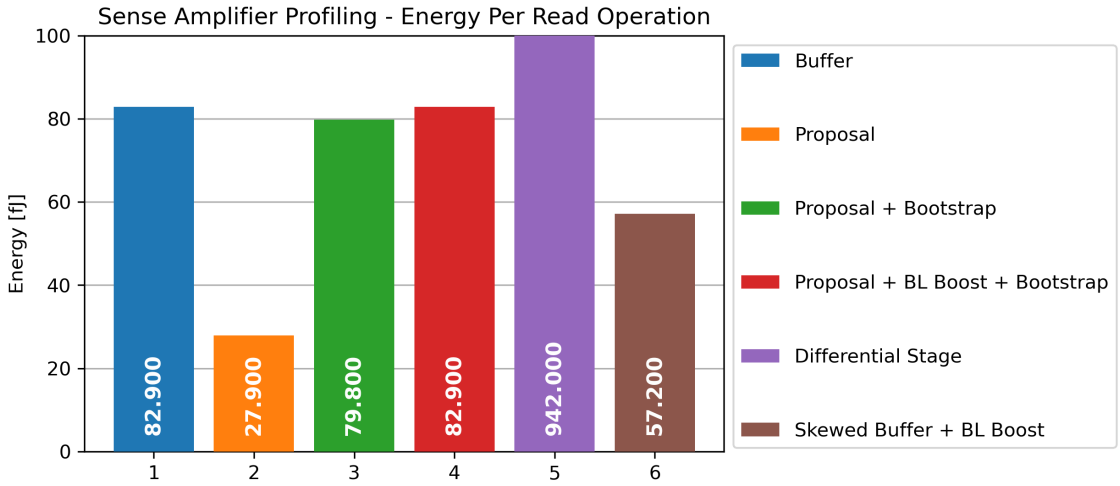


(a) SAs Area comparison.

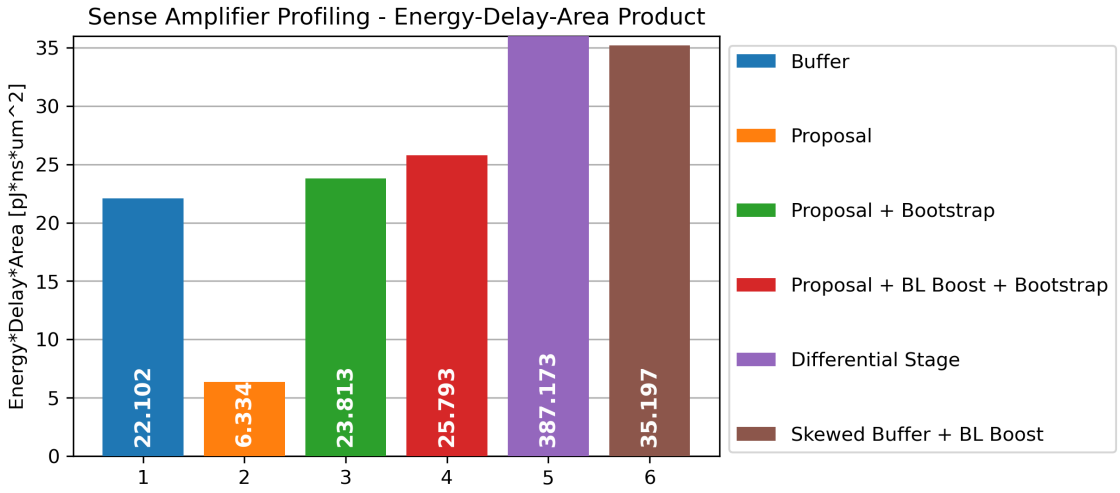


(b) SAs Read-Delay comparison.

Figure 4.10: Sense Amplifiers area-delay-energy comparison.



(c) SAs Energy-per-Operation comparison.



(d) SAs Area-Delay-Energy product comparison.

(PG). Additionally, since the BLC requires accessing two words per cycle, such a generator must produce two pulses, thus increasing the complexity of its design. All these discussed constraints, together with the large sensibility to process variations, result in uncertainties for the pulse width, leading either to performance degradation (in case of oversized, yet reliable pulse generator design) or increased failure probability (in case of a generator optimized for performance, whose actual pulse width may be too small for a correct switch of the SA). These topics will be covered more in detail in a dedicated section.

Assuming an inaccurate generation of the pulse, the behaviour of the proposed amplifier can be investigated in order to prove its compatibility. Even if not all the SRAM architectures rely on pulsed wordline assertion, a fictional pulse width can be defined as the time interval between the assertion of the wordline and the generation of the SEN signal, hence in this elapsed period of time, a defined amount of differential input voltage has to be developed at the input of the SA. Inaccurate pulse duration results in lessened input voltage, which may cause wrong latching or no switching, depending on the amplifier's topology, ultimately resulting in a failure. Unlike these implementations, called hereafter pulse-intolerant, with the proposed amplifier inaccurate pulse generation only results in performance degradation instead of a failure. This is because, in case a pulse duration shorter than the expected one, the BL voltage will be higher than required, thus resulting in P1 driven by a suboptimal gate-source (bitline) voltage: the pFET will still charge Q_N but at a slower rate, increasing the switching time on behalf of increased reliability. The results of a parametric simulation of the switching time versus the BL voltage driving P1 can be seen in Fig. 4.11 together with the related waveform of both OUT(Q) and Q_N (Fig. 4.11a-4.11b), where one can relate the asymptote of the obtained curve (Fig. 4.11d) with the pFET's threshold voltage in the interested process corner; this translates into the minimum voltage to be developed between the supply and the BL in order to turn on the transistor.
Due to the importance of the final BL voltage developed during the pulse duration, the amplifier has been further characterized with respect to this quantity with the help of a fictional *dynamic amplifier threshold*, defined as the BL voltage at which the device's output value falls to $0.9V_{DD}$ during a read operation in a 128x128 array.

Results of Monte Carlo simulations are reported in Fig. 4.12 together with a temperature sweep in the commercial range and process corners characterization; these will be used afterwards for the design of the pulse generator.

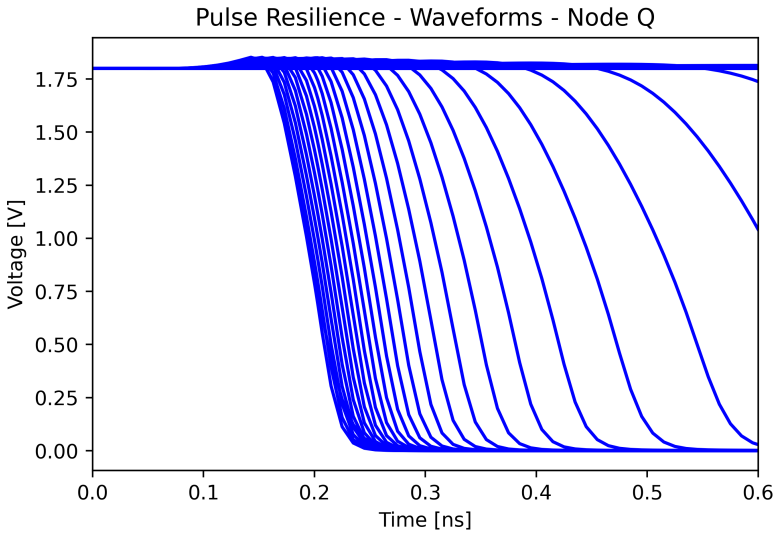
The aforementioned "relaxed" switching due to inaccurate pulse width generation can be even exploited in order to enhance the resulting array's access time. Given a two-pulse BLC SRAM structure as in Fig. 4.1, one can break down the overall access time from the control signals sampling in the following components:

$$T_{2P} = t_{CLK,Q} + t_{GEN} + t_{PREC} + t_{PW,1} + t_{NOVP} + t_{READ} + t_{PCNT} \tag{4.2}$$

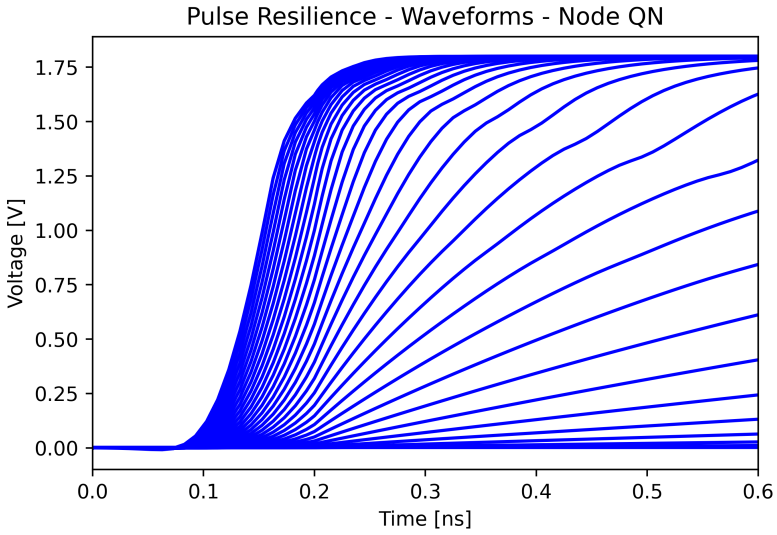
where:

1. $\mathbf{t_{CLK,Q}}$ represents the sequential overhead due to input signal's latching;
2. $\mathbf{t_{GEN}}$ includes all the introduced delays for the generation of the pulse from the read signal assertion;
3. $\mathbf{t_{PRE}}$ is the width of the pre-charge signal's pulse;
4. $\mathbf{t_{PW,1}}$ is the pulse width of the first decoder's enable signal;
5. $\mathbf{t_{NOVP}}$ is the non-overlap times between the two pulses;
6. $\mathbf{t_{READ}}$ is the propagation delay across the SA, which includes the BL discharge time;
7. $\mathbf{t_{PCNT}}$ stands for the propagation delay across the popcount network.

As can be easily noticed, the width of the second pulse does not appear in the equation. This is because a pulse duration larger than the SA read time is assumed, hence the only constraint on the latter pulse regards its maximum width and is linked to dynamic read disturbance and power consumption reduction. The first pulse instead is crucial for the performance of the system and any over-sizing could severely slow down the memory. A straightforward design would see t_{PW1} sized as the read time of the amplifier as reported in Fig. 4.10b, in essence: would wait for

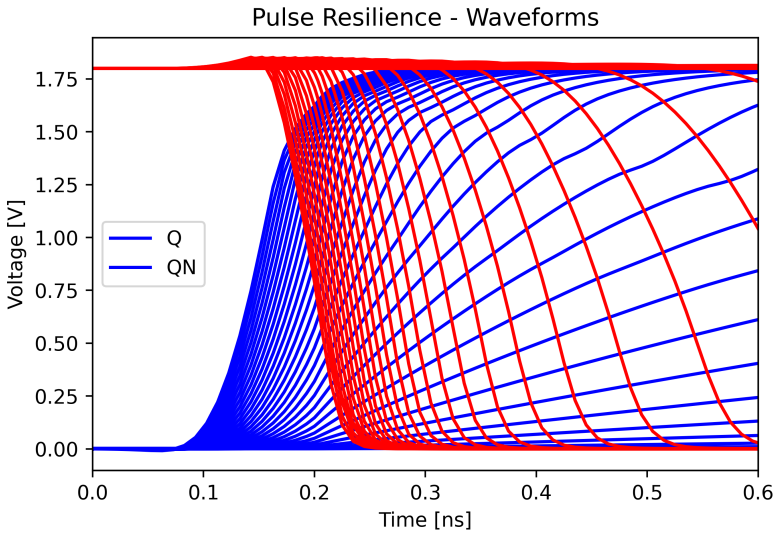


(a) Node Q waveforms.

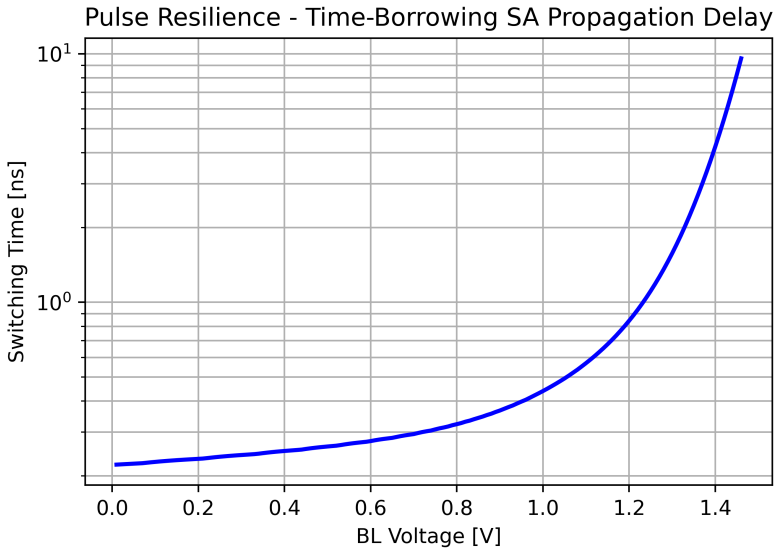


(b) Node QN waveforms.

Figure 4.11: Time-Borrowing Sense Amplifier's Pulse Resilience Simulation Results.

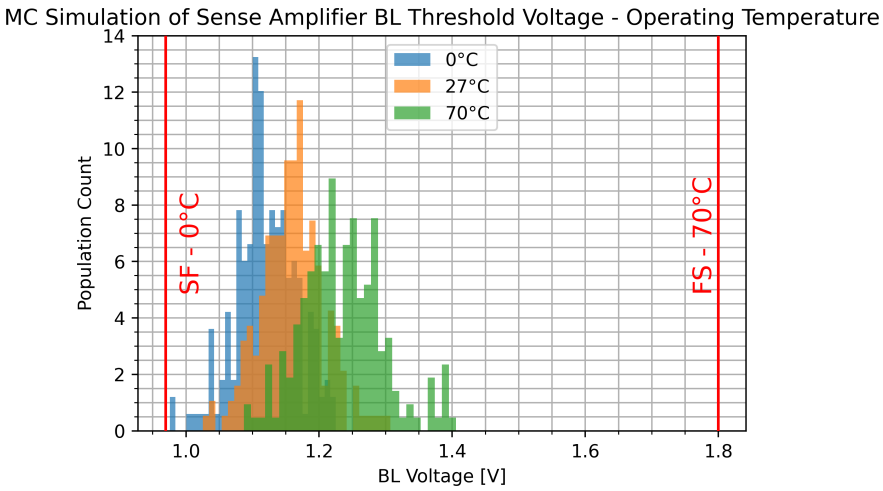


(c) Superposed waveforms.

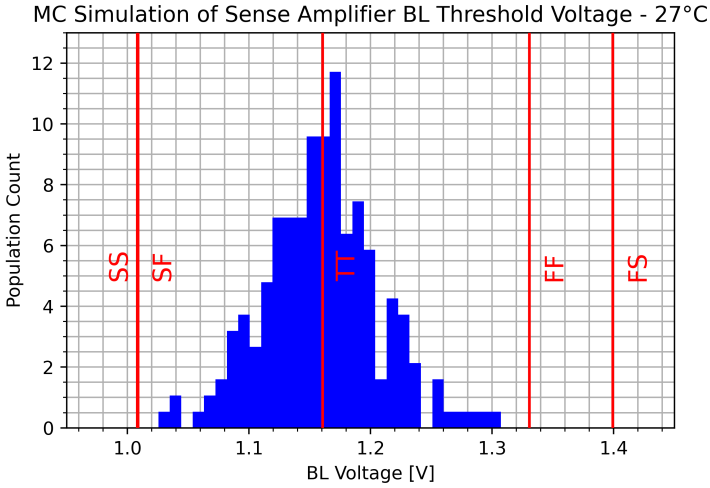


(d) Switching time versus final BL voltage.

Figure 4.11: Time-Borrowing Sense Amplifier's Pulse Resilience Simulation Results.

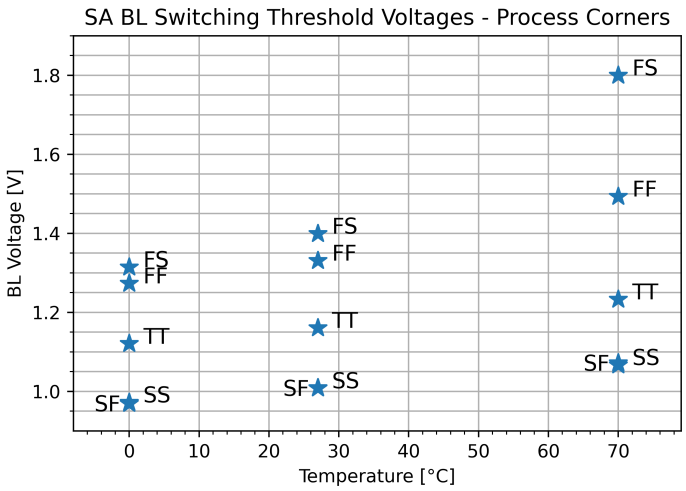


(a)



(b)

Figure 4.12: Sense Amplifier Dynamic Threshold - Monte Carlo and Process Corners Simulations



(c)

Figure 4.12: Sense Amplifier Dynamic Threshold - Monte Carlo and Process Corners Simulations

the device to completely switch. However, since the introduced amplifier can be driven by sub-optimal pulse widths at the expense of a longer switching time, one can reduce the pulse duration to the one required to make the device switch before the closing of the time window dedicated to access the bitcell array ($T_{2P} - t_{PCNT}$), shifting the end of the commutation in the next pulse. This behaviour, due to the resemblance with its digital counterpart in two-phase latch-based architectures, will be called hereafter *time-borrowing* and its qualitative waveforms can be seen in Fig. 4.9b.

4.2.6 Layout

As all the components of the CSA have been defined, a pre-layout area estimation can be performed; results are shown in Tab. 4.3 where one can acknowledge the lowered overhead introduced by the proposed sensing scheme.

Going on with the layout design of the device, the only constraint to take into account is the width of the resulting set of geometries; in fact, the macro has to

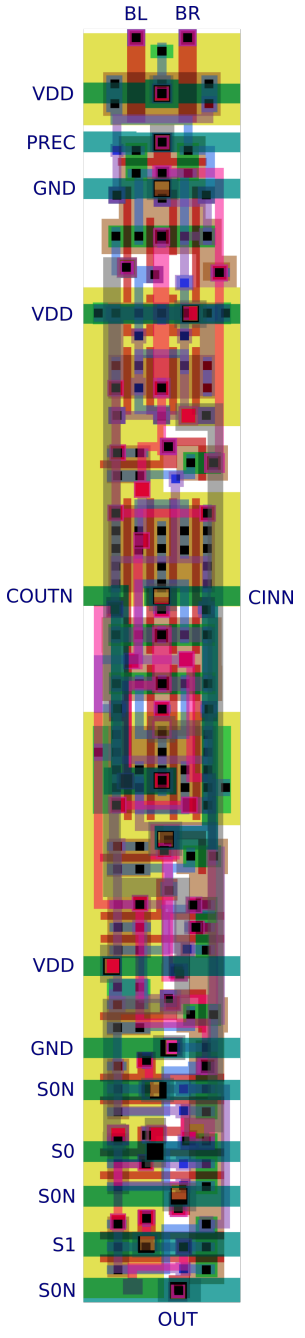
be pitch-matched to the bitcells to allow a side-by-side placement for those array configurations that do not require column multiplexing. The resulting layout view can be found in Fig. 4.13 with details of the main features. The main aspect that stands out from the design is the high aspect ratio of the amplifier (≈ 1:8), mainly due to the width constraint; moreover, this led to an extremely dense routing exploiting metal layers up to the second last (M4). The main consequence of this is that the sense amplifier array is now a *no-route zone*, meaning that the compiler should not be allowed to consider the area as available for supply and/or control signals routing. However, this does not translate into a real issue, since, due to foreseen design choices, both supply and control signals share the same y-coordinates in the resulting drawing, hence their routing reduces to a simple rail tracing, avoiding congestion. Fig. 4.14a depicts the layout view of a 16-bit SA array configuration in absence of column multiplexing, where the pitch matched design can be seen. Fig. 4.14b highlights the dedicated routing of the array while Tab. 4.3 summarizes the post-layout area occupation of the final device.

Table 4.3: Computing Sense Amplifier Design - Area Summary

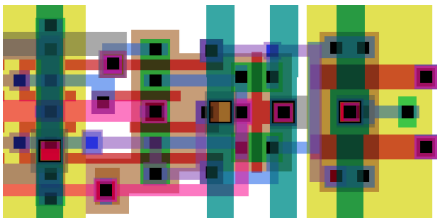
Area units	**Sense Amplifier Features**					
	Pre-Layout					
	TOT	**SA**	**NOR**	**RCA**	**XOR**	**MUX**
(λ^2)[b]	1140	98.571	128.571	514.284	167.143	231.428
(μm^2)[a]	5.586	0.483	0.630	2.520	0.819	1.134
%		8.6	11.3	45.1	14.7	20.3
Area units	**Post-Layout**					
	TOT	**SA**	**NOR**	**RCA**	**XOR**	**MUX**
(λ^2)[b]	15528	5712		4183	2567	3065
(μm^2)[a]	76.089	27.990		20.500	12.580	15.020
%	+1262	36.7		26.9	16.5	19.7

[a]SKY130 PDK - 0.15um channel length implementation.

[b] $L_{CH}=2\lambda$

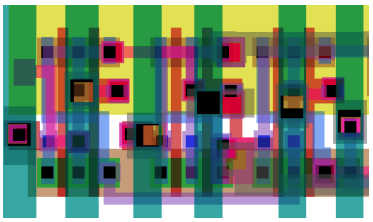


(a)



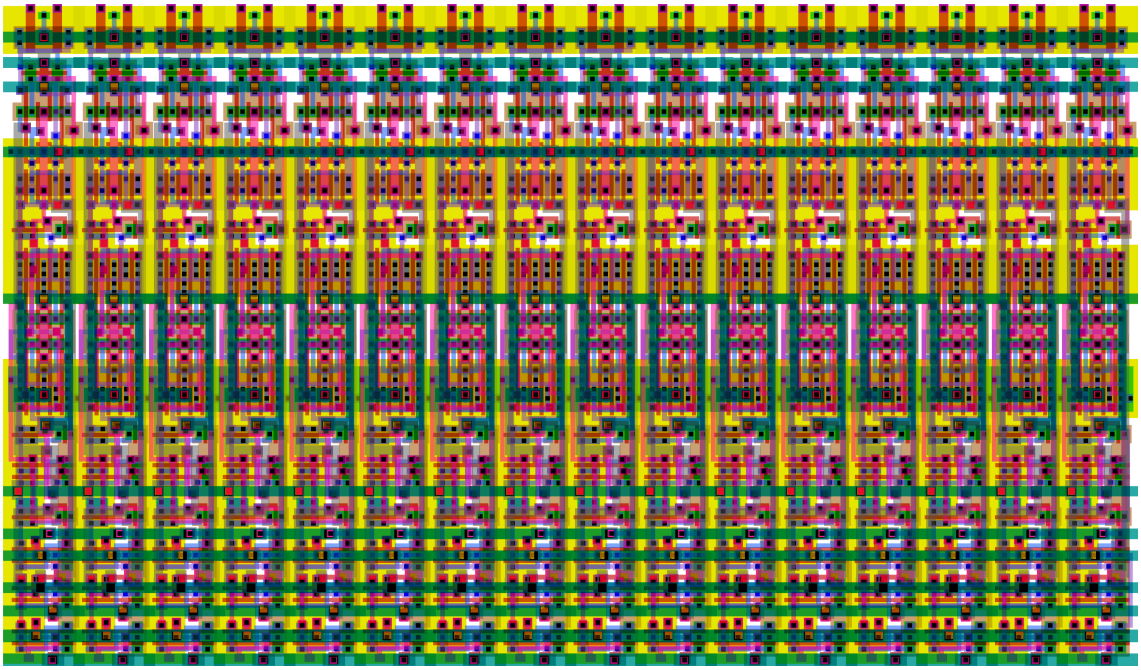
(b) Detail of the proposed sensing scheme. The layout is merged with the one of the NOR gate.

(c) Detail of the static CMOS RCA logic gates.

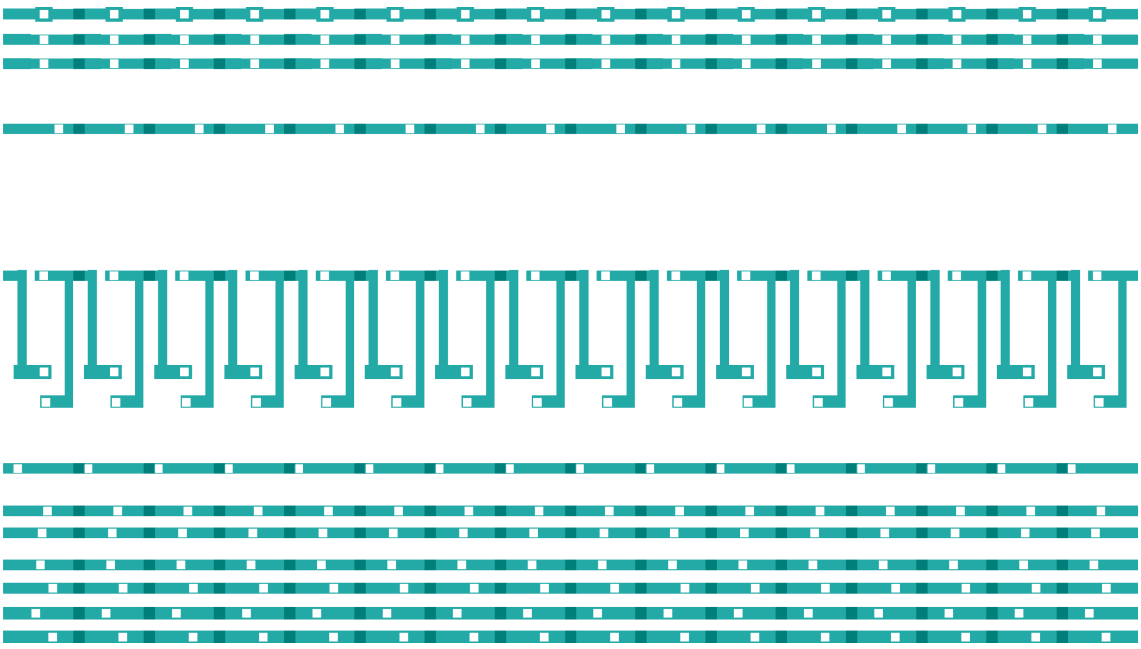


(d) Detail of the PT MUX gate.

Figure 4.13: Computing Sense Amplifier - Layout Views.



(a) Array of SA for a 16-bit word configuration with no column multiplexing.



(b) Detail of the dedicated rail routing of supply and control signals in the array.

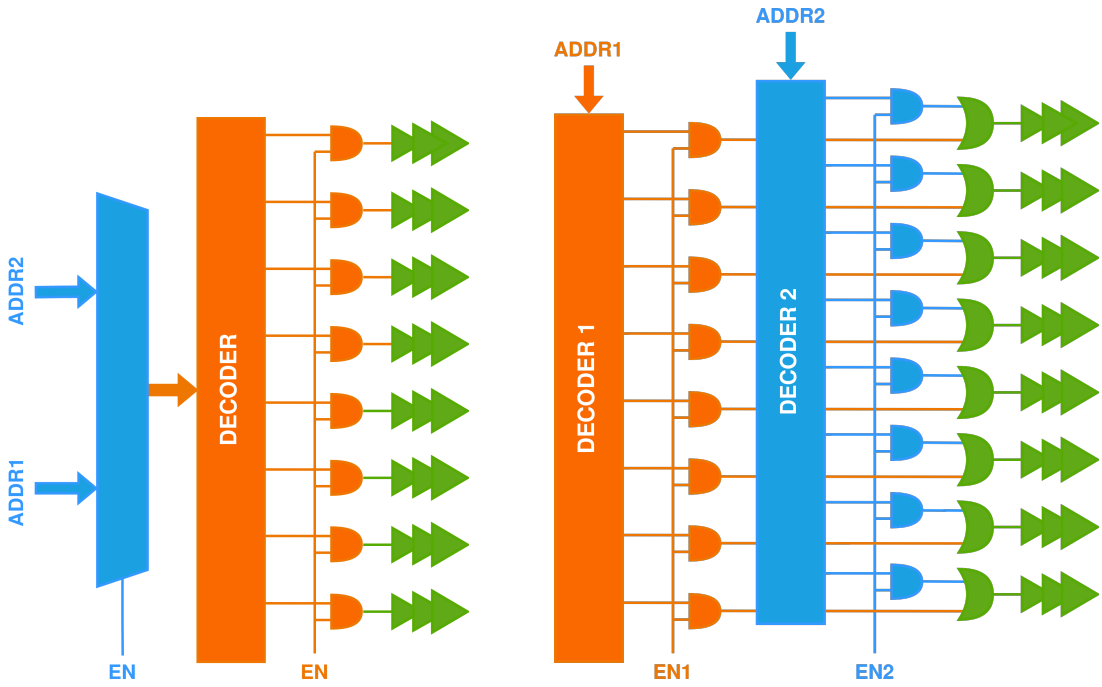
Figure 4.14: Computing Sense Amplifier Array - Layout Views.

4.3 Address port

Due to the nature of the operations required to accelerate neural networks, an IMC enhanced memory needs to be fed with at least two address words; according to the target applications of this work, one related to neuron weights, one to its input activations. Doubling the address words requires altering the address port's circuitry with respect to a standard memory implementation; moreover, the adopted pulsed WL assertion requires the decoding circuitry to be supplied with an enabling scheme to achieve the desired behaviour.

4.3.1 Address decoders

With respect to the targeted figure of merit to optimize, one can identify two configurations that prioritize area overhead and performance respectively. The former (Fig. 4.15a) exploit a single instance of the decoding circuitry multiplexed between the outputs of the two arrays of D flip-flops (DFFs) for the address words latching. Albeit inexpensive from an area perspective, this method leads to a significant performance loss since the decoder network has to be crossed twice, one for each of the addresses, due to multiplexing. This, linked to the synchronization with an enable signal, increases the size of the resulting non-overlap time between the pulses, lowering the performance of the array as explained in Sec. 4.2.5. Conversely, an implementation that promotes access time translates into a straight-forward doubling of the decoding network; this, given the gating circuitry related to the enable signal shown in Fig. 4.15b, limits the propagation delay overhead to two stages of gates only: the AND required for the enabling and a NOR for the sharing of the word-line drivers. Due to the increased access time linked to BLC circuitry, this latter implementation has been preferred to alleviate the performance loss and has been incorporated in the memory compiler.



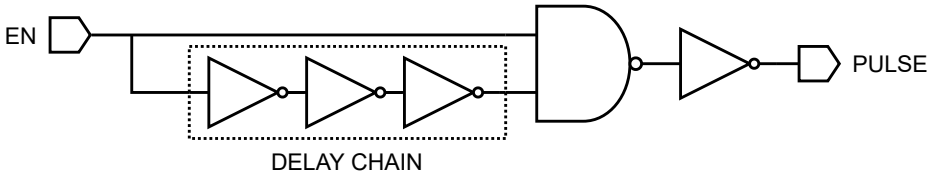
(a) Area-optimized implementation. (b) Speed-optimized implementation.

Figure 4.15: Address Decoding Networks for Multiple Address Words

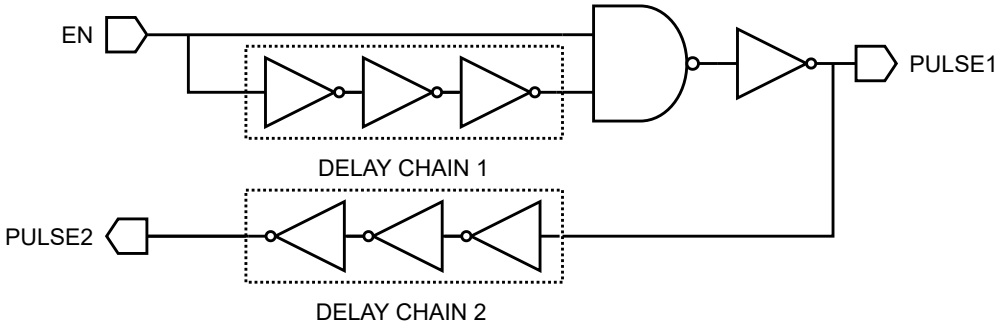
4.3.2 Pulse generator

As stated in previous sections, due to the adoption of a pulsed WL assertion to ease the issues of read-stability in 6T arrays, a pulse generator device is required.

The chosen circuit topology for the generator's implementation is the so-called *clock-chopper* [36] whose logic level schematic is shown in Fig. 4.16a. It consists of an AND gate driven by two versions of its enable signal: the first is directly connected to one of the gate's inputs, the second instead has to cross a *delay-chain* made of an odd number of inverters in charge of generating a complemented and delayed version of the input signal. At steady state the inputs of the AND gate differ, which means that its output is a logic 0; once the enable signal is asserted, its *fast input* turns to 1 while its *slow input* still shows the previous value (i.e. logic 1). This asserts the output of the AND gate as long as the input signal's propagation across the delay chain is not complete; once the chain's output value turns to its definitive



(a) Single pulse clock chopper.



(b) Double pulse clock chopper.

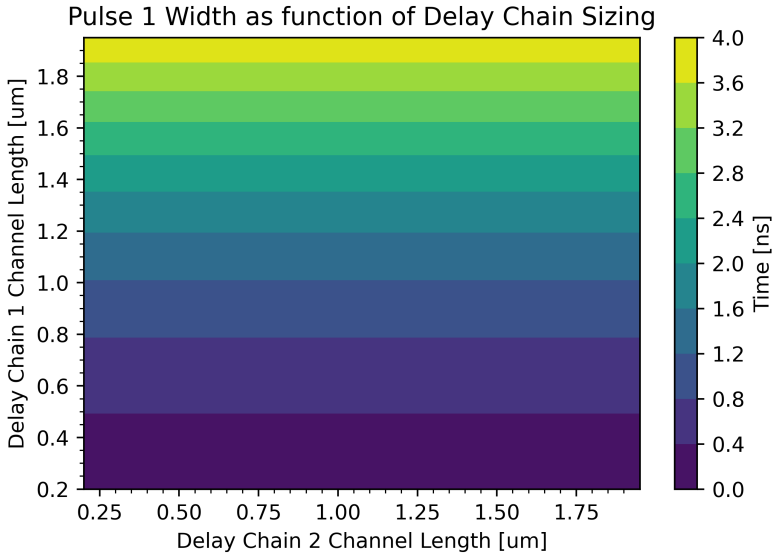
Figure 4.16: Pulse Generator - Schematics.

value, the gate finds itself with two different values as inputs, thus de-asserts its output "closing" the pulse. The pulse duration is the difference in the propagation delay of the two paths of the input signal, hence the propagation delay of the chain of inverters, which can be controlled by tuning the channel length of the involved transistors. Since both the accessed rows rely on pulsed WL assertion, the device must generate a further pulse, this is achieved by inserting an additional delay chain at its output to delay the pulse of the desired quantity. This allows for tight control on the non-overlap times which, as previously stated, is crucial for the memory access time. The final schematic can be seen in Fig. 4.16b.

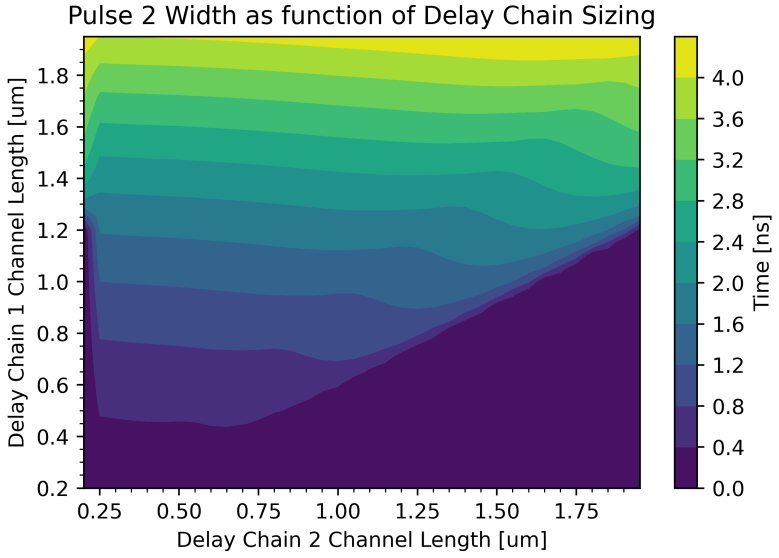
Fig. 4.17 shows the results of a parametric simulation aimed at characterizing the main design quantities of the device as functions of the FETs channel lengths of the two inverter chains. As expected, Fig. 4.17a depicts the trend of the first pulse width as insensitive with respect to the size of the second chain; reasonable result as the two involved paths are isolated. Conversely, Fig. 4.17b highlights the inter-dependency of the second pulse width from the first one. In particular, the

second pulse experiences a stretching in its duration as the delay introduced by the second delay line increases. That means that according to this design methodology, the second pulse width will always be larger than the first; this, however, doesn't translate into an issue either for the system's performance or its reliability, as the second pulse duration does not appear in the equation for the access time (Eq. 4.2) and the time stretch magnitude is not significant for the read-stability degradation. A further phenomenon that can be observed in Fig. 4.17b is that for small first pulse durations and long channels in the second chain, the latter acts as a filter with respect to the input pulse. This is to be addressed to the propagation delay across the chain being larger than the pulse duration, hence the input getting annihilated as it travels across the logic, according to the concept of inertial delay. Fig. 4.17c shows the second pulse's delay with respect to the first and sticks to the expected trend, being it related to the sizing of the second delay chain only; blank regions represent those scenarios in which measurements failed due to the absence of the pulse and have to be intended as infinite delay time. Finally, Fig. 4.17d depicts the resulting non-overlap times where one can notice how the concerned quantity is not free to be chosen given pulse width, in fact, due to time-stretch phenomenon and propagation delays inserted by the remaining gates of the device, the trend is far from being linear.

Due to the aforementioned non-linearities in the obtained trends, the design of the component becomes complex and may cause issues in its interfacing with the customized decoders as well as provide synchronization issues regarding the non-overlap times which may slow down the memory. To make matters worse, the device is susceptible to process variations; this, combined with the sensitivity of the SA's threshold from process variations, could severely degrade the reliability of the resulting system. To make up for these issues, a process-tolerant pulse generation technique will be introduced and investigated in the following section.

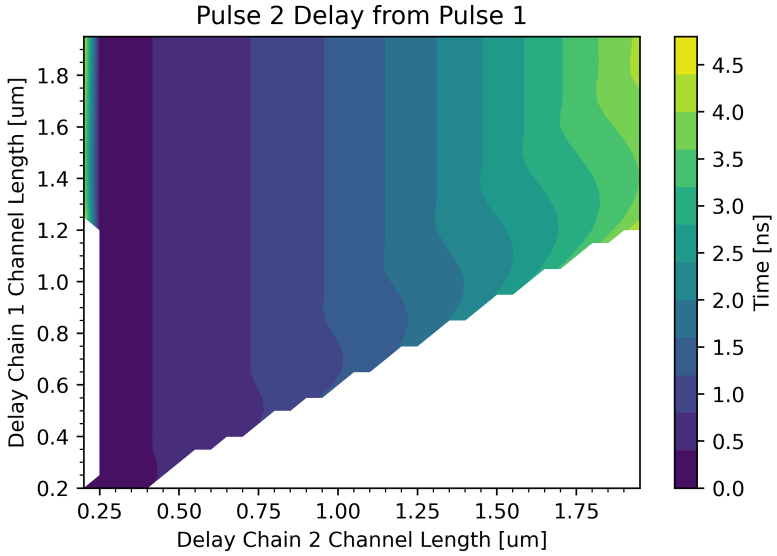


(a)

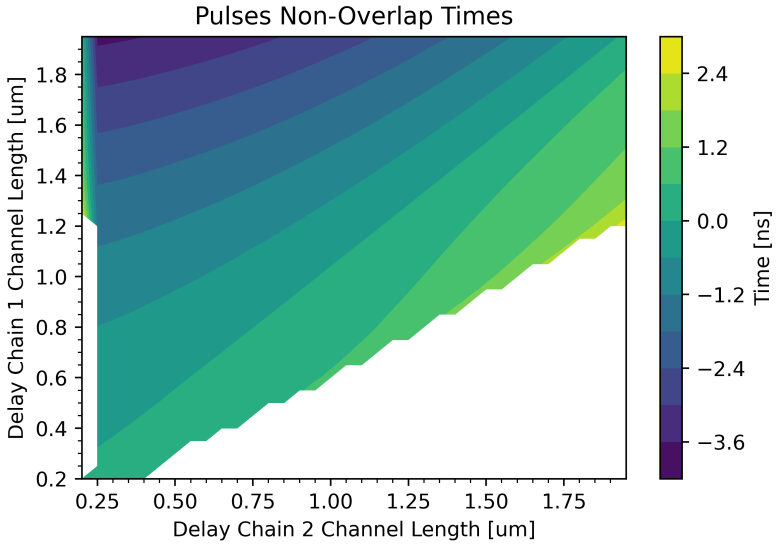


(b)

Figure 4.17: Pulse Generator - 2 Pulses Clock Chopper Characterization.



(c)



(d)

Figure 4.17: Pulse Generator - 2 Pulses Clock Chopper Characterization.

4.3.3 Closed-loop pulse gating technique based on replica bitline

To soothe their consequences and safely keep track of mismatches and degradation induced by process variations, the exploitation of a feedback loop is required. The concept is to involve in the loop all those quantities that introduce failure sources in the concerned scheme, this way the system will regulate and synchronise itself reducing the fault probabilities to the absolute minimum. A perfect example of this technique is the Replica Bitline (RBL) concept introduced in [38] which sees the control logic waiting for the switching of an extra inserted column in the array, namely *replica column*, to generate the SA enable signal. This way process variations in the array are taken into account and, by tuning the number of cells tapped to the RBL, the power consumption of the amplifier's array can be minimized by a precise synchronisation of its enable signal. In the same fashion, one can exploit an RBL (which comes for free in a memory compiler, being de-facto the standard for SRAM memories performance's enhancement) to decide when to "close" (i.e. de-assert) the WL pulse. This technique has been first introduced for BLC SRAM arrays in [24] where the feedback loop is employed to limit the BL voltage swing to the absolute minimum to reduce the dynamic read stability related failures. By tapping a different number of replica bitcells (RBC) to the RBL one can tune the ratio between the former and a normal BL discharge time; this allows to make the RBL SA switching faster than normal SAs in the array, thus accounting for the propagation delay of the path, that goes from the outputs of the amplifiers to the ones of decoders, needed to close the pulse. However, this implementation is based on a different RBL structure with respect to the one involved in OpenRAM; while the former sees a replica column made of dummy cells (not tapped to the related WL but to BL only, hence behaving as capacitances) with an extra RBC individually driven at the column's end, the latter instantiates RBC for all the cells of the column, this way, for each combination of the decoder's output, there will always be an RBC turned on to discharge the RBL. Ultimately the compiler adds a further RBC at the end of the replica column driven by a signal asserted each time a read operation is performed, resulting in two RBC discharging the RBL, halving the switching time.

According to this implementation, this section proposes a closed-loop RBL based pulse gating scheme that translates into non-invasive customization and reduces the amount of code to be added to the compiler.

Starting from the proposal in [24] one can extend the concept to two-pulses BLC SRAM arrays according to Fig. 4.18. The starting condition of the system is the time instant after the end of the pre-charge phase, hence all the BLs and the RBL are at logic 1, as the output of the RBL SA. Once the READ signal is asserted by the control logic, given the output of the RBL SA, the first decoder is enabled, thus triggering the wordline drivers and rising the selected WL to logic 1. Due to the aforementioned replica column structure, the RBL will discharge, thus turning the amplifier's output to logic 0. According to the scheme in Fig. 4.18, the discharge rate of the RBL will be the same as a standard BL due to the absence of additional RBCs, which means that the RBL SA is elected to represent the propagation delay of the whole SA array. Once the amplifier's output has switched there's no reason to keep the WL asserted, consequently, the first decoder's enable signal is de-asserted through the AND gate and the pulse of the second decoder can start. The latter is addressed to a simplified pulse generator, in charge of providing, this time, a single pulse; moreover, since the pulse width does not appear in the memory access time equation, the device can be over-sized to be reliable in all the process corners and temperature range while still avoiding a full BL discharge, ultimately limiting the system's power consumption. Nonetheless, as expected, the described scheme does not account for the propagation delays in the feedback loop; moreover, since the discharge rate of the RBL sticks to the one for normal BL, non-replica columns will experience a full BL discharge. This, as explained in Sec. 4.2 is unnecessary and is to be avoided to fully exploit the time-borrowing features of the proposed amplifier. Ultimately, the closed-loop pulse width shows itself over-sized, since a considerable amount of time exists between the amplifier's switching and the pulse closing instant (Fig. 4.20a).

To fasten the RBL discharge time to account for the pulse size overhead, an OpenRAM compiler-friendly answer is to insert additional RBCs at the end of the replica column. The working principle is the same of [24], but instead of varying the

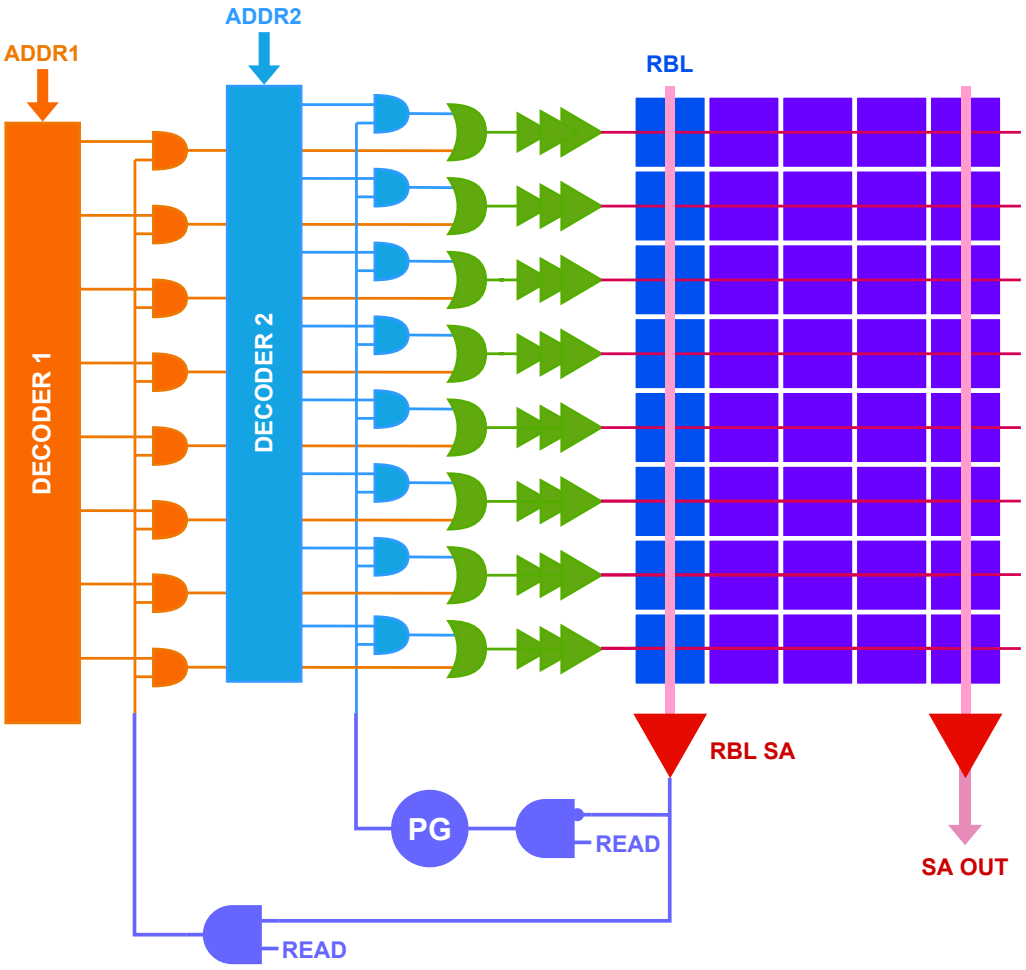


Figure 4.18: Closed Loop RBL Pulse Gating - Block Scheme.

capacitance on the RBL, this implementation varies the RBL discharge current by turning on multiple RBCs at the same time. To preserve the regularity of the array, a further replica bitcell insertion in a column ought to be followed by the completion of a whole row; since the additional cell is an RBC, the remaining elements of the row will be dummy cells, in order not to degrade the standard BLs discharge time increasing the equivalent capacitance on the line. The introduced rows will be called *replica rows* (RRW) and the related WLs *replica wordlines* (RWL).

Fig. 4.20 show the results of a simulation performed in a 128x128 array enhanced with different combinations of replica rows. Fig. 4.20a-4.20d depict the waveforms of the resulting closed-loop controlled pulse as well as the standard BL voltage compared to the RBL one. This technique allows for pulse resizing and subsequent performance enhancement by reducing the pulse duration up to 39% (Fig. 4.20e). Furthermore, the resulting BL voltage increases with the number of additional RRWs, increasing the reliability of the system and unlocking time-borrowing. The only drawback of the described technique is the area overhead due to array extension, which however sits below 3% for 3 additional rows. As stands out from the trend of both the pulse width reduction and final BL voltage, these enhancements tend to saturate as the number of RRWs increases, mainly due to the discharge time being inversely proportional to the discharge current, hence yielding a $\frac{1}{N}$ trend where N is the number of additional rows. However, a sweet spot is represented by using 3 additional RRWs, a configuration that achieves significant pulse width reduction while yielding a BL voltage that sits above half the supply voltage.

4.3.4 Pulse generator compiler

As stated in the previous section, the architecture of the pulse generator boils down to the one needed for the generation of a single pulse, thus with reduced complexity. Since the target of this work is the development of an enhanced version of OpenRAM which supports IMC features, the design of the concerned device has to be parameterized to allow the generation of memory macros of different sizes and needs to be made technology independent. This creates the need for a *pulse generator compiler*

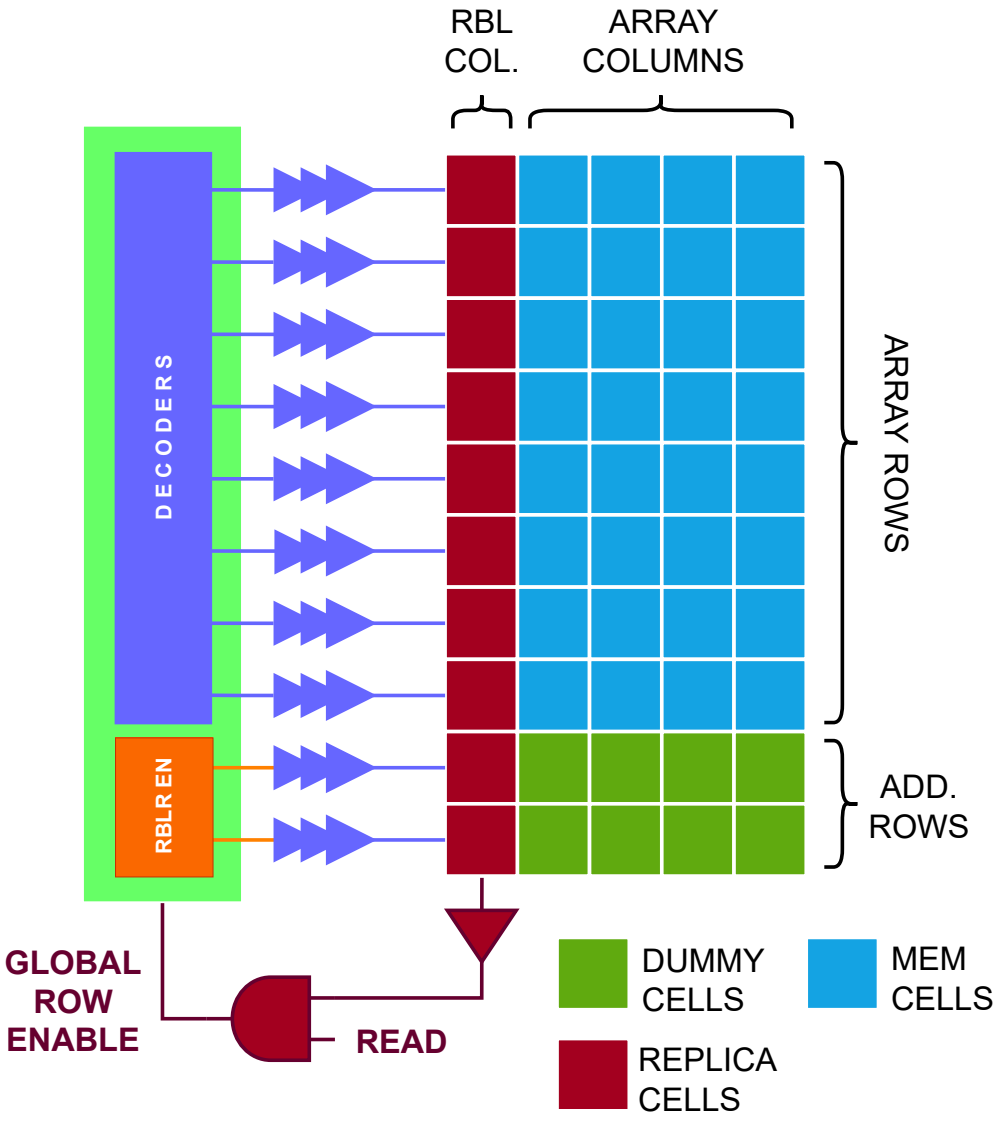


Figure 4.19: Closed Loop RBL Pulse Gating - Replica Rows.

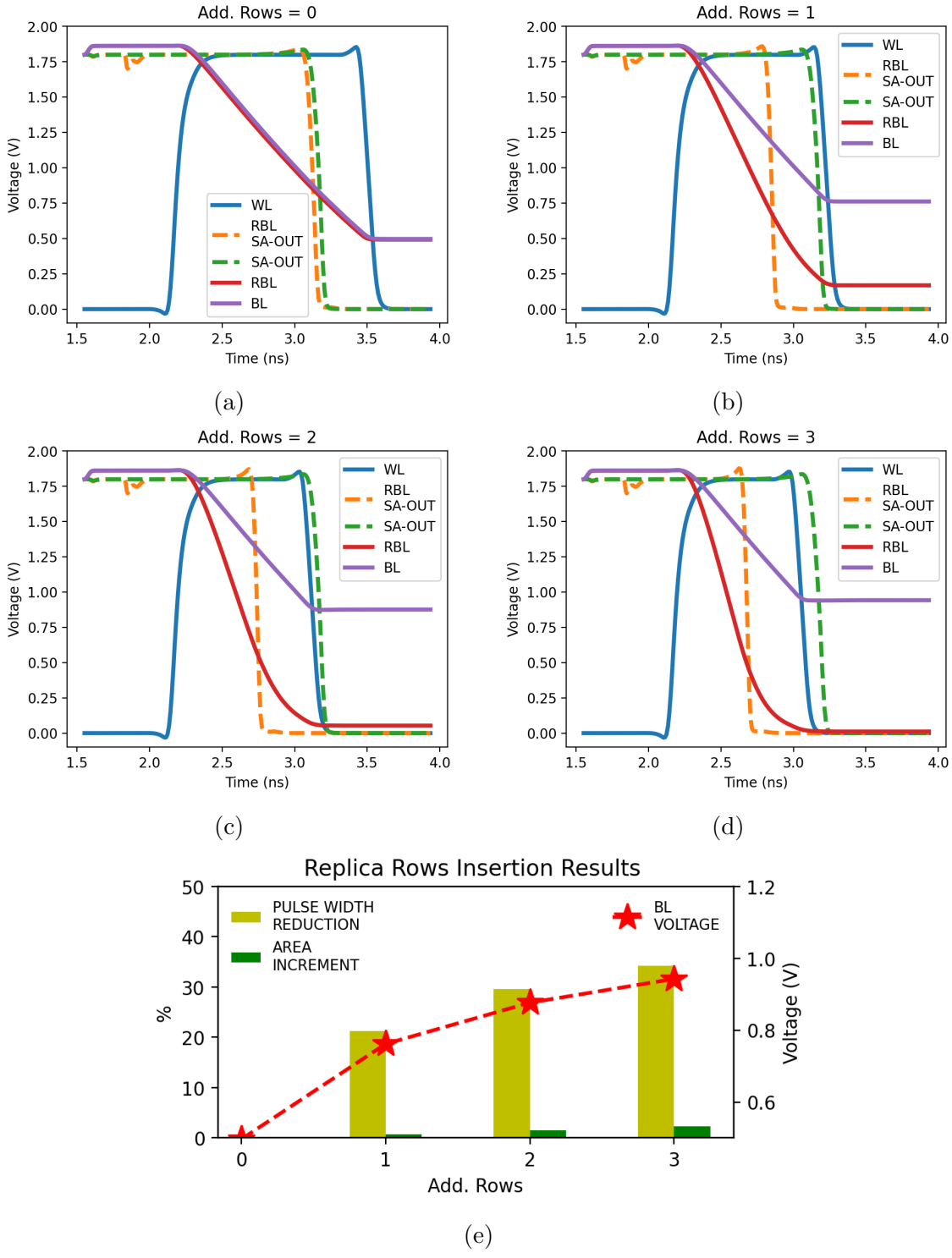


Figure 4.20: Closed Loop RBL Pulse Gating - Simulation Results.

whose development will be described in the forthcoming part of this section.

The script is completely based on Python and supplies a specification-to-GDS flow according to the diagram in Fig. 4.21. Few sources have to be provided to the algorithm to operate successfully: characterization of the chosen architecture (in this case the single pulse clock chopper in Fig. 4.16a) for its sizing and characterization of the bitcell array in order to pick the optimal pulse width according to its size. To fulfil the first requirement the concerned component has been further characterized to obtain a curve able to relate the design quantity (the transistor's channel length of the delay chain) to the output quantity (the pulse width); results of parametric simulations can be seen in Fig. 4.22. Then, through curve fitting, an analytical expression of the output characteristic is obtained to allow for further manipulations by the compiler. Since the only aim of the component is to provide a pulse wide enough to cause the amplifier to switch in any possible scenario, the simulations took place in the FF process corner and refer to 90% to 90% pulse width, this causes the device sized by the compiler to always generate a pulse width slightly larger than the optimal, to achieve complete error resiliency. Fig. 4.22a show the results of the first curve fitting where the whole set of data shares the same analytical expression and coefficients:

$$t_{pd}(L) = \alpha + 3 \cdot \ln(2) \cdot \left(\beta L + \gamma L^2 + \omega L^3\right) \tag{4.3}$$

The equation refers to the Elmore delay RC model of a chain of three cascaded inverters where only the dependency from channel length has been made explicit. The latter, instead of a pure quadratic expression as provided by the model, has been turned into a third-grade polynomial to increase the fitting accuracy.
As can be observed from Fig. 4.22b, the model perfectly matches the simulation results only for large values of channel length, and fails for values around the minimum size, translating in modeling errors up to 40%. To further improve the model's accuracy, the data set has been split into two regions to be fitted individually with the same equation. The boundaries of the formers have been optimized to minimize the maximum relative error of the overall fitting (Fig. 4.22c), results can be seen in Fig. 4.22d. As one can observe, the composite fitting allowed for a delay model whose maximum relative error sits below 0.8%, resulting in a reliable analytical

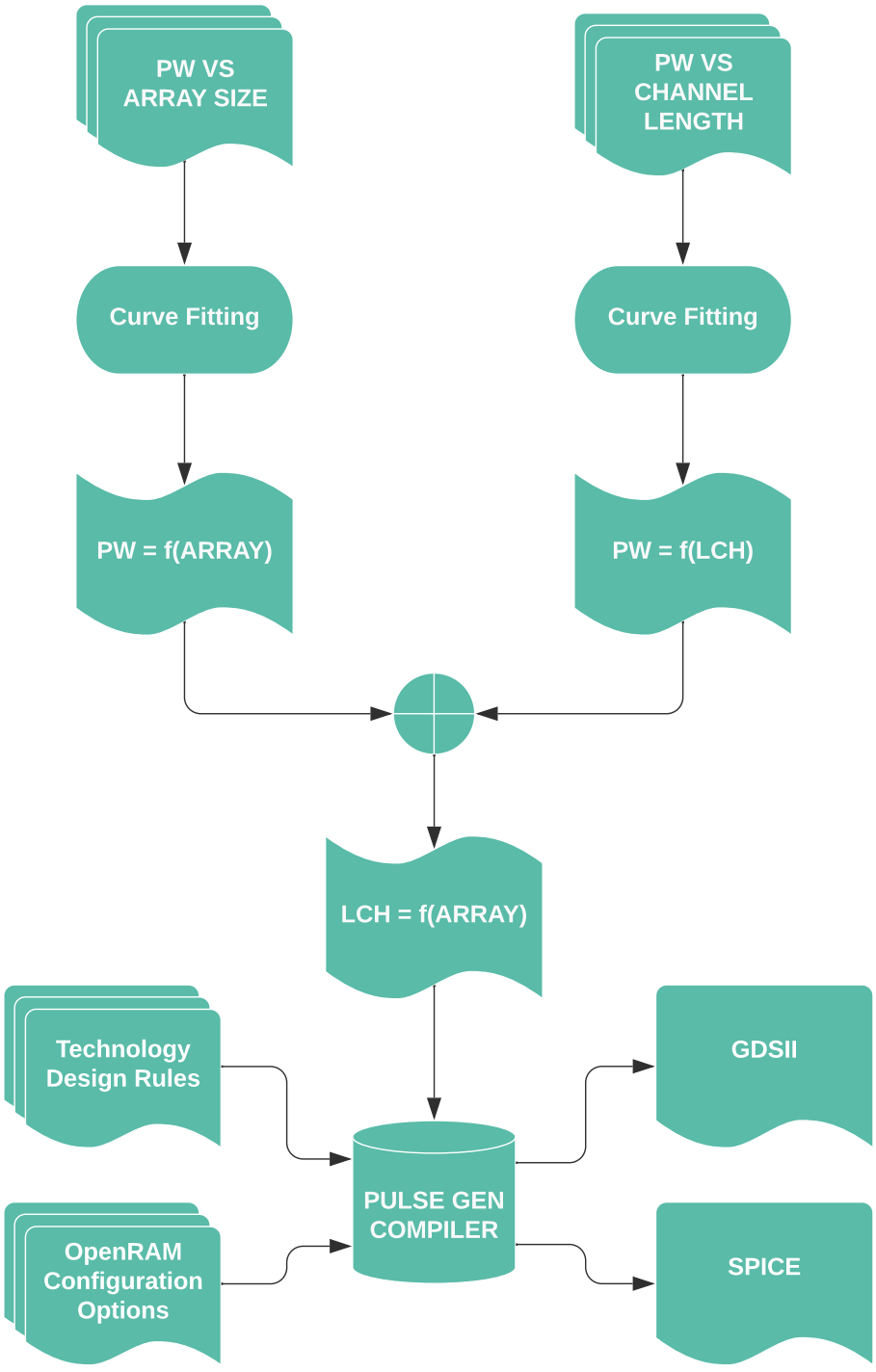


Figure 4.21: Pulse Generator Compiler - Flow Structure

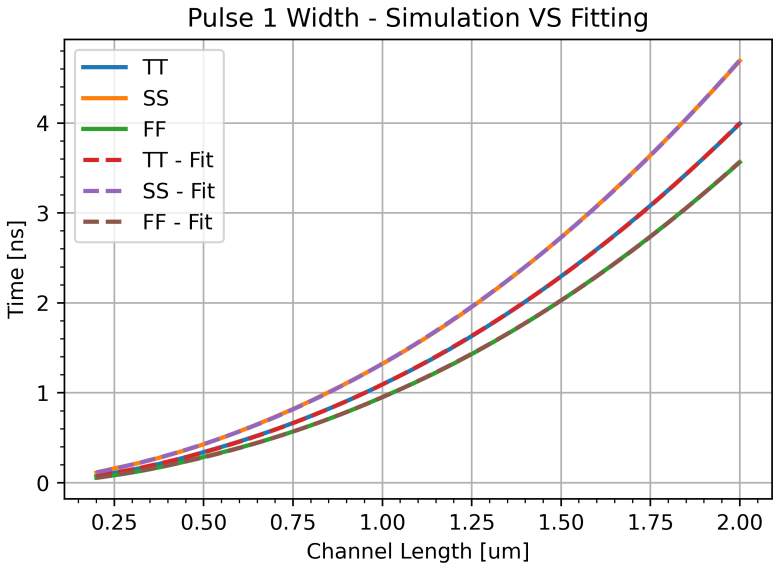
expression for the compiler algorithm.

The other characterisation needed by the compiler is the required pulse width versus the number of rows of the bitcell array (i.e. the column size). Once again different configurations of the memory have been simulated for row sizes up to 256. Following the same workflow as for the clock chopper's characterization, results have been fitted according to the following analytical expression:

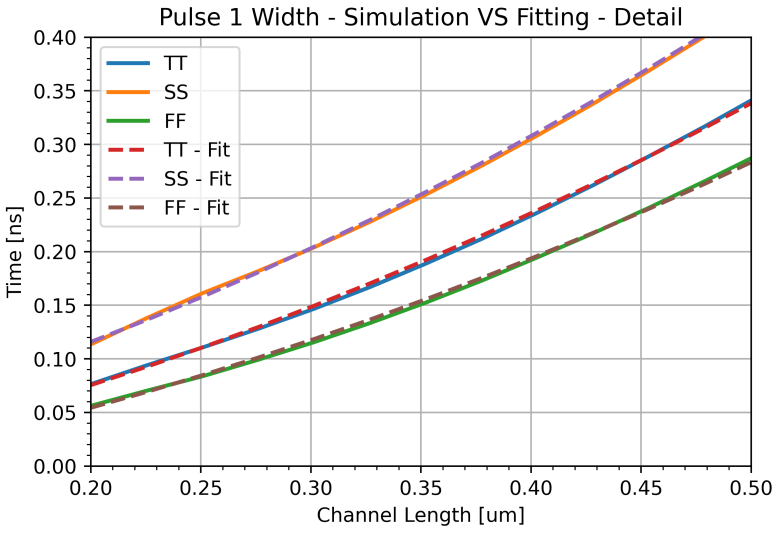
$$t_{SW,BL}(R) = \alpha + \beta\sqrt{R} + \gamma R \tag{4.4}$$

tuned to obtain always positive modeling errors, this translates into a further oversizing factor increasing the reliability of the device. Results are shown in Fig. 4.23 where the maximum relative error sits below the 3.5%.

Done with the required characterisations, the algorithm can be fed with the elaborated analytical expressions to automatically size the pulse generator during the memory array compilation. Once the channel lengths of the FETs have been computed, the layout of the component can be drawn; for this to be carried on, the compiler calls for MAGIC VLSI [39], an open-source layout editor tool, and issues the required commands to draw shapes for the geometries. For a layout to be drawn, the user has to be aware of the technology's design rules; to make the compiler technology-independent, the algorithm has been developed in order to be able to automatically interact with the OpenRAM technology files. More in detail, the tool scans the *tech.py* file of the desired technology to learn about design rules and layer names. As a result, the layout of the component is always 100% DRC errors free and achieves the lowest possible area occupation. Moreover, some features have been inserted in the layout editor as the option to individually choose the pFET and nFET sizes and select the multiplication factor of the inverters (drive strength). Fig. 4.24 shows the layout views results of different runs of the script with different channel lengths and transistor widths.

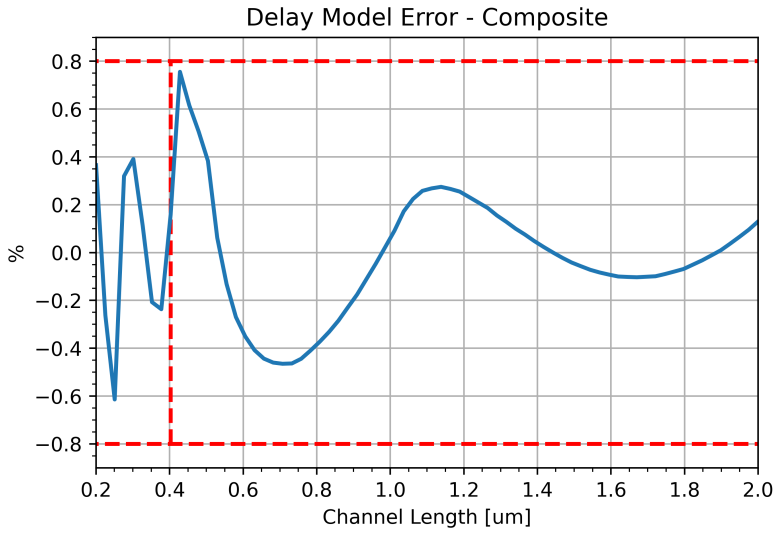


(a)

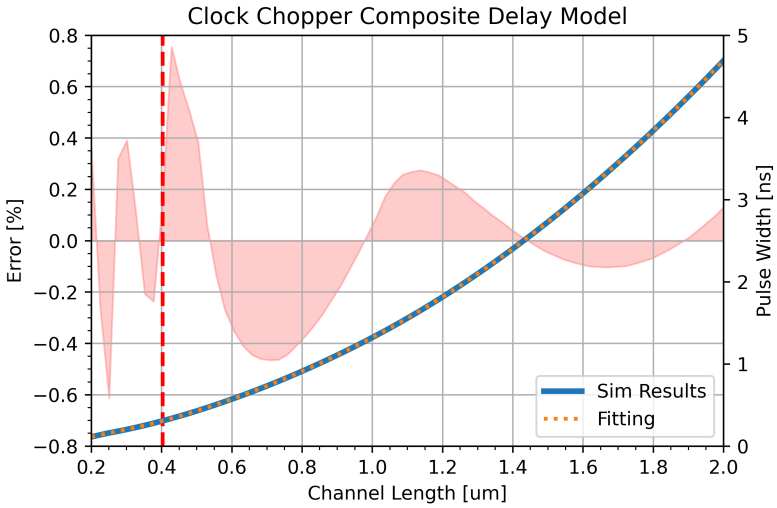


(b)

Figure 4.22: Clock Chopper Characterization - Single Pulse



(c)



(d)

Figure 4.22: Clock Chopper Characterization - Single Pulse

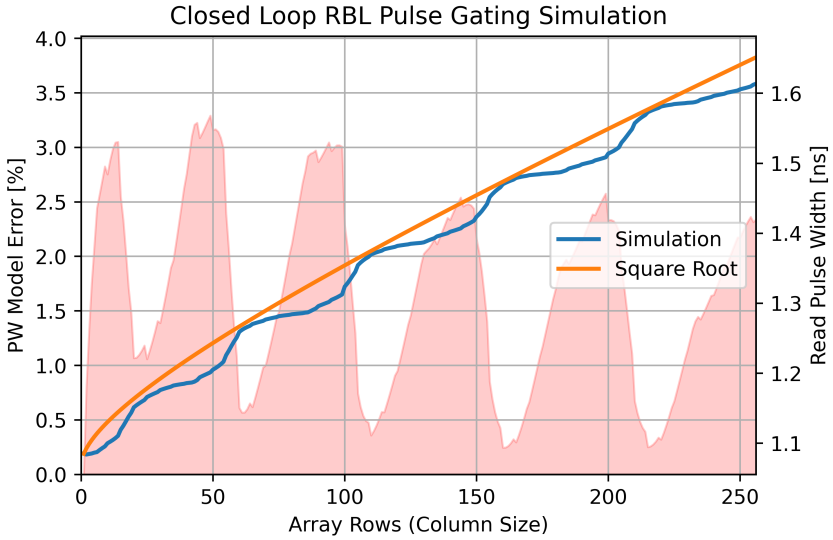
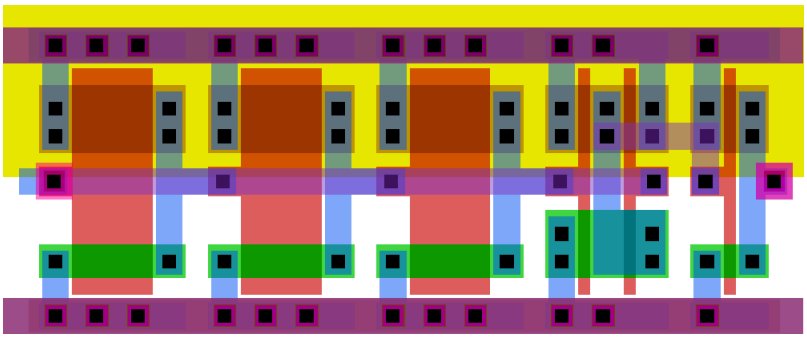


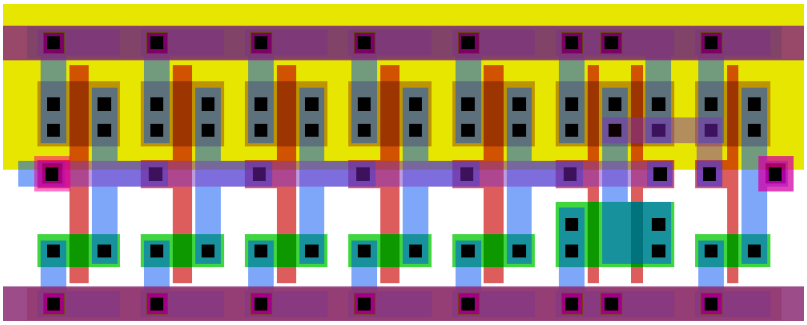
Figure 4.23: Closed Loop Bitline Discharge Times - Simulation Results and Fitting.

4.3.5 Layout

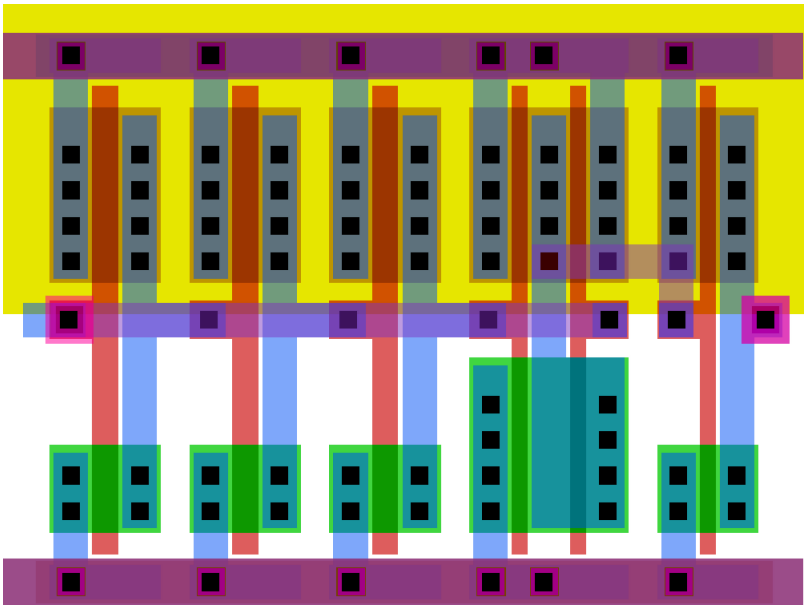
Introduced features have been fully implemented in OpenRAM, which now supports double address words for every configuration of the array's rows. Fig. 4.25 depicts the final layout of the address port composed by the compiler for an array made up of 16 rows; on the left and in the center the two decoders can be identified with their related pre-decoding stages, the bus routes at the bottom ease the overall cell interfacing with address DFFs by confining their routing to the leftmost border of the instance, thus avoiding routing congestion. The outputs of each decoder have been provided with an AND array for gating to allow pulsed assertion. Finally, in the rightmost part of the layout, one can appreciate the array of wordline drivers, where each row is composed of a NOR gate for decoders outputs blending followed by a chain of an odd number of inverters, automatically sized by the compiler to result in a delay optimized path for WLs driving. Beneath the lowermost wordline driver, a replica row driver can be found, whose enable signal is directly connected to the one of the decoder associated with the first issued pulse.



(a) L_{CH} = 1um - W_N = 0.42um, W_P = 0.84um



(b) L_{CH} = 0.25um - W_N = 0.42um, W_P = 0.84um



(c) L_{CH} = 0.25um - W_N = 0.84um, W_P = 1.68um

Figure 4.24: Pulse Generator Compiler - Layout Results for Different Configurations.

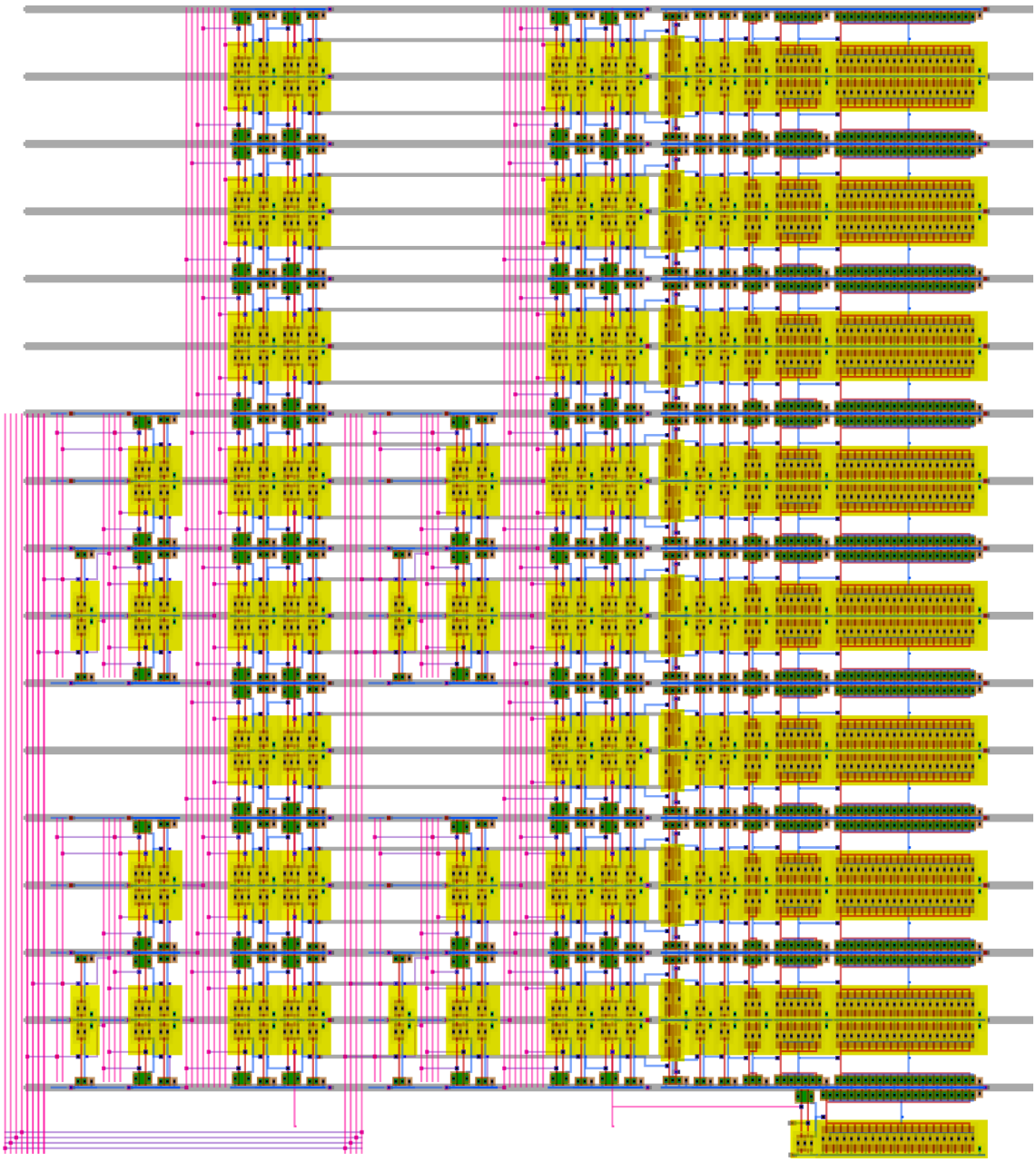


Figure 4.25: Address Port - Layout View for a 16 Rows Array Configuration.

4.4 Population-count network

As explained in Sec. 1.2.2, the use of binary quantization in neural networks inference allows for strong computational effort reduction, ultimately reducing the weight-activation product to an XNOR operation and partial results accumulation to a population count (*popcount* hereafter). The population count operation, from a hardware perspective, is nothing more than a 1-count in a given word, thus the resulting architecture translates in a bit-counting network. Far from being solved, the problem of optimized bit-counting has been brought back into focus by cryptography applications, thus bringing on a new wave of hardware solutions and dedicated microprocessor's instructions [40]. However, most of them are sequential circuits, which might not be compatible with IMC memory customization due to the need of stalling the system for a certain number of clock cycles; hence, to preserve the one-cycle timing of the resulting array, it has been decided to supply the SRAM with a fully combinational network. In [40], one promising approach to parallel popcount is the *Carry-Save-Adder* (CSA) based reduction tree for multi-operand addition, while [41] delivers a survey on existent architectures for parallel bit-counting and ultimately proposes a new network which is expected to reduce both complexity and propagation delay of the operation based on a hybrid RCA-CSA implementation. In this section, a comparative study will be carried on between the two introduced network topologies based on the involved figures of merit such as area occupation and maximum operating frequency, ultimately deeming their implementation in a memory compiler from an automated design point of view.

4.4.1 CSA-based binary tree reduction HDL generator

CSA trees are widely used in hardware multipliers for performing fast addition in a multi operand scenario. The fundamental instances of the network are full-adders (FA) and half-adders (HA), that behave as 3-to-2 and 2-to-2 compressors respectively; by exploiting the compression ratio of the FA one can reduce the number of input words to the network to two words within a certain number of compressing stages, the last step is to sum the obtained words with a standard carry-ripple

adder to achieve the desired computation. With respect to the adopted reduction algorithm, one can classify CSA trees in two: Wallace[42] and Dadda[43] types. The main difference between the two is the instantiation criterion for the compressors: the former allocate gates on each layer to reduce as soon and as much as possible the number of operands, while the latter tries to reduce the complexity of the resulting network by only allocating compressors to obtain the maximum number of allowed inputs for the next stage. A comparative analysis of the two architectures can be found in [44].

A possible approach to the problem of popcount is to see bit-counting as a peculiar multi-operand operation, more in detail a multi-operand sum of 1-bit words. According to this philosophy, CSA reduction trees can be employed to efficiently fulfil this need. To achieve a fully modular implementation in the memory compiler, the generation of the network ought to be both completely automated and parametrized with respect to the array's word size; hence the need for the development of a digital design flow to be equipped in OpenRAM.

The core of this additional feature is a hardware-description language (HDL) generator for the binary reduction network: the script performs a software version of the chosen hardware compression algorithm while labeling nets to produce a VHDL description of the resulting network. In the following, the description of the tool will be provided referring to a word size different from the single bit required in popcount applications, however, the algorithm behaves adaptively to the bit-width parameter, allowing for the reduction of words in any format. The starting point of the process is the placing of a dot representation of the input words in the data structure upon which the whole algorithm is based: a 3D array called *PP_tensor* (Partial Products) where the number of rows is the number of inputs, the number of columns is the word length of the final result while the number of "sheets" of the array is the number of levels of the tree. Each column represents a single weight in the binary representation of the final result (col 0: 2^0, col 1: 2^1, etc). A representative sketch of the 3D structure is shown in Fig. 4.26.

For each level, the matrix is analyzed taking into account two subsequent columns

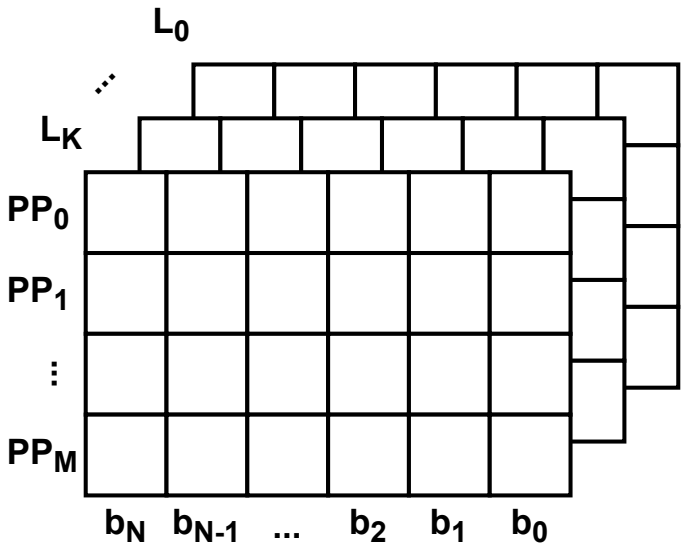


Figure 4.26: 3D Array - CSA HDL Generator

at a time. These are stored in temporary vectors, namely *tmp-column* and *tmp-column-next*, upon which the processing is made. For each element of these vectors three fields are present:

1. ***value***: 1 or 0, points out whether a single bit is present or not;
2. ***state***: *used* or *unused*, denotes whether the bit has been used in that level or not;
3. ***label***: represents the name of signal.

The content of the state field is analyzed for the whole vector to compute the number of operand bits present at that level for that particular bit weight and hence compute the *excess bits*, defined as the difference between the number of effective bits in the column and the maximum number of operands allowed for the next level. Once this quantity is known, a recursive function is invoked to allocate the compressor and reduce the number of bits. To decide whether to allocate a FA or an HA one needs to know how many bits to subtract from the column's number of elements (the *excess bits*) and how many bits in that column are available; the state field is used

to distinguish between the available and the unavailable ones. Once the decision has been made, the function only needs to cover the whole length of the vector searching for three available operands in the case of the FA or two available operands in the case of the HA. Once the desired triplet or duplet has been found, knowing the labels of the bits to compress, a VHDL component instance of the chosen compressor is generated and n-1 out of n inputs of the compressor get toggled to zero in the column getting labeled as *used*. The remaining input preserves its presence (a 1 in the *value* field) but gets labeled as *used* to take into account the sum bit of the compressor while not being available for further instances in the same level. Since the carry-out bit of both kinds of compressor has a higher weight with respect to the inputs and the sum bit, a 1 labeled as *used* is appended in the *tmp_column_next* to take into account the carry bit produced by the compression. When the number of operands in the column reaches the desired value, the processing on that particular bit weight is complete and as a result, the processed column gets stored in the matrix related to the next level, the column labeled as next becomes the column under processing and the algorithm starts again.

As aforementioned, the algorithm can compress the operands according to both the reduction criteria: Wallace and Dadda. This is achieved through manipulation of the *max operands per lvl* variable: by simply setting it to zero the script allocates compressors as long as available bits are present, according to the Wallace philosophy, by feeding the variable with a vector instead, each level will be related to a maximum allowed operands as the Dadda reduction algorithm requires.

The tool reduces the number of input operands down to two words to perform a final sum operation through a behavioural description of the adder for further optimization through synthesis constraints; however, by simply increasing by 1 the number of levels (K), the algorithm will recognize the underlying command and perform a final run relaxing the constraint on the unavailability of bits of neighbour weights, thus instantiating FAs and HAs to obtain a ripple carry adder.
The pseudo-code for the algorithm can be found in Alg. 1.

Algorithm 1: CSA Tree Reduction Algorithm

```
for (l = K ; l > 1 ; l −−) do
    tmp_column ← PP_tensor(:,0,l);
    for (b = 1 ; b < N ; b ++) do
        tmp_column_next = PP_tensor(:,0,l);
        avail bits per lvl ← numel(tmp_column);
        excess bits ← numel(tmp_column) − max operands per lvl;
        while (excess bits > 0 and avail bits per lvl ≥ 2) do
            num_FA ← floor(excess bits/2);
            if (num_FA > 0 and avail bits per lvl ≥ 3) then
                                /* a FA can be allocated */;
                for (r = 0 ; r < M ; r ++) do
                    if (tmp_column(r) is 1 and is unused) then
                        index.append(r)        /* append index of the chosen
                         elment to index vector content */ ;
                        if numel(index) == 3 then
                            tmp_column(index(1)) ← 0 and used;
                            tmp_column(index(2)) ← 0 and used;
                            tmp_column(index(3)) ← 1 and used   /* sum bit of
                             the FA remains active in the column */;
                            tmp_column_next(first available unoccupied position)
                             ← 1 and used      /* carry output bit of the FA
                             becomes active in the next column */;
                            excess bits − = 2;
                            printFA_VHDL();
                            break from last for loop;
                        end
                    end
                end
            end
            else if (avail bits per lvl ≥ 2) then /* an HA can be instantiated
             */
                same operations and loop as for the FA but with 2 inputs;
            end
            update_available_bits();
        end
        PP_tensor(:,b−1,l−1) ← tmp_col;
        tmp_column ← tmp_col_next;
    end
end
```

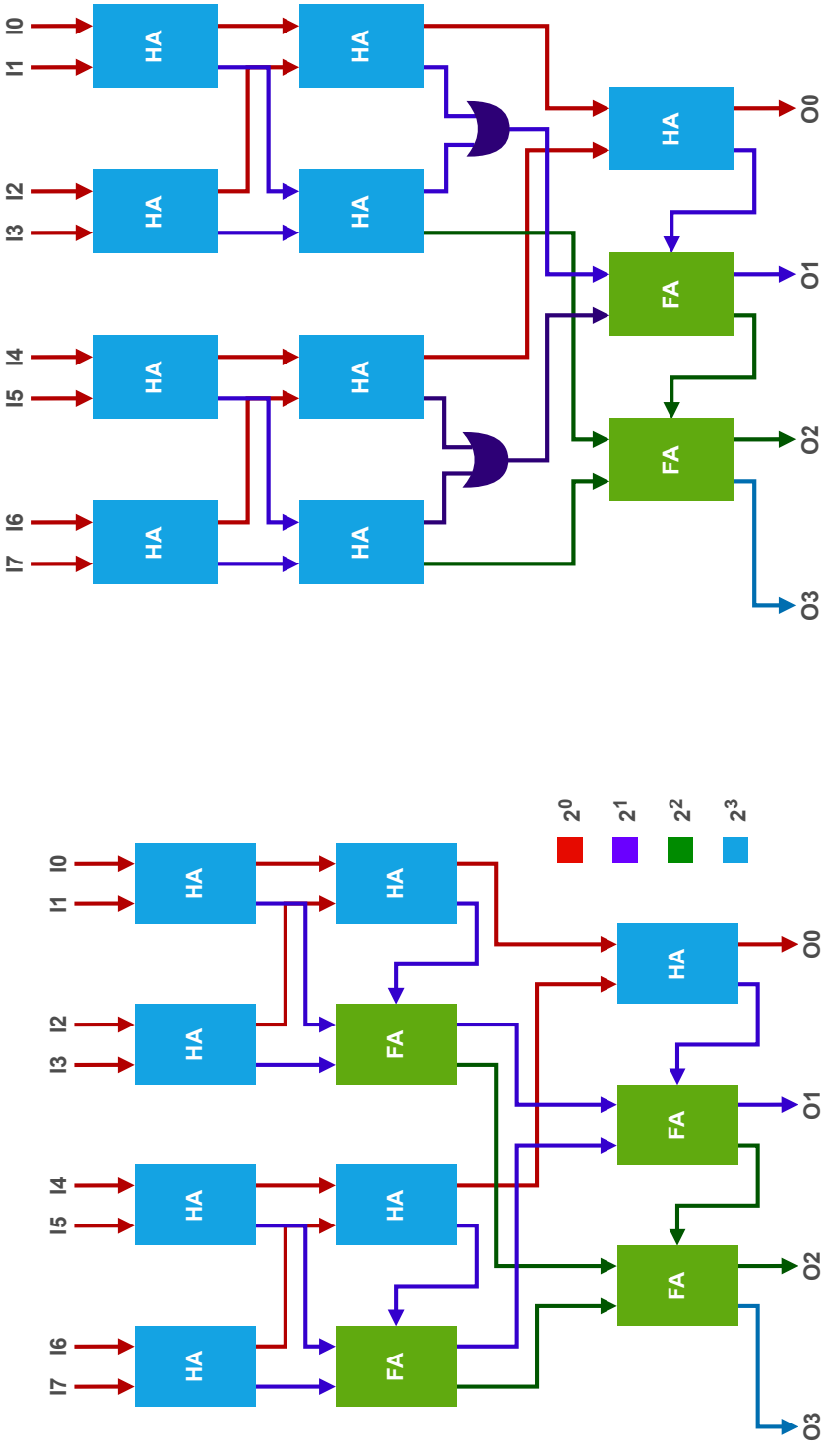
4.4.2 Dalalah bit-counting network

Another remarkable circuit for bit counting is the one proposed in [41]. In response to the lack of optimized parallel networks, the authors propose a new popcount architecture that blends the CSA and RCA philosophies. Analyzing the resulting architecture for an input word of 8 bits (Fig. 4.27a) one can observe how the latter gets split in two and each of its parts of 4-bits experiences a 2 stages compression in a CSA fashion to obtain, altogether, 2 words of 3 bits ($\lceil \log_2 ((8/2) + 1) \rceil$). These can be further summed in an RCA fashion through a chain of FAs.

As a way of both increasing performance and reducing complexity, the authors meditated on the effective role of the logic gates at a lower level of abstraction, investigating the elementary logic functions performed by HAs and FAs. Following a design by contraction approach, the netlist has been further simplified by noticing that not the whole circuitry behind the FAs of the second stage was effectively exploited. Same logic results can be achieved by replacing the aforementioned FAs with HAs equipped with a further OR gate for carry computations. The resulting network is shown in Fig. 4.27b.
Although not standing out from an 8-bit implementation, the hybrid compression scheme can be better appreciated in its larger word sizes versions. In fact, to obtain a 16 popcounter, for instance, it is enough to instantiate twice an 8 bits network and combine the outputs through an adder. This method can be extended for larger words resulting in popcounting networks for 32, 64, and 128 bits. An example of a 16-bit long input word configuration is depicted in Fig. 4.28.

4.4.3 Bit-counting networks comparison

Thanks to the developed script for the CSA-based binary reduction trees and the strong regularity of the Dalalah network design for larger word sizes, all the introduced popcounting networks have been described through VHDL, verified and finally synthesized and characterised through Synopsys Design Compiler for word lengths of 8, 16, 32 and 64 bits using the SKY130 High-Density Standard Cells Library. As



(a) Dalalah 8 bits - Base.

(b) Dalalah 8 bits - Modified.

Figure 4.27: Popcount Networks - 8 bits Dalalah.

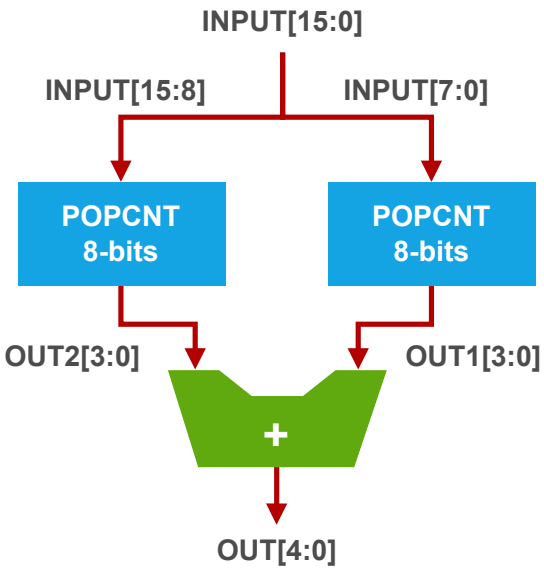


Figure 4.28: Popcount Networks - 16 bits Dalalah.

can be seen from Fig. 4.29a, area occupation for a given word length does not show sensitive variations from one architecture to the others, thus neglecting its possible election as the main figure of merit for the adoption of a given network. However, up to a 32 configuration, Dalalah stands out as the lowest area implementation. It is interesting to notice how the latter netlist shows a significant increment in cell count with respect to its counterparts (Fig. 4.29b) while, however, yielding comparable area occupation results. This may be due to the design simplification explained previously which increase the number of overall gates but reduces their complexity. Key results arise instead from a propagation delay characterization of the synthesized networks. As can be observed in Fig. 4.29c, this time a significant difference exists between the different architectures, proving the Dalalah bit-counting network to be the one to allow for the maximum operating frequency in all the investigated word length configurations.

However, from a design automation perspective, the latter network can only be exploited for word lengths sticking to the expression $8 \cdot 2^i$, with i in the range of natural numbers. This limits its application in a memory compiler like OpenRAM whose word size parameter is completely free to be chosen. However, to prioritize performance at the expense of modularity, the network to be implemented in the

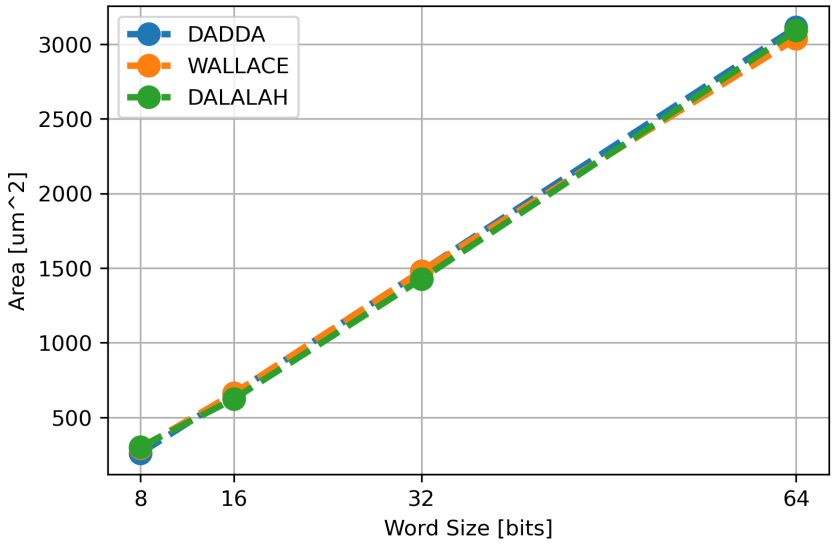
enhanced version of the compiler will be the Dalalah one.

Since, as introduced in Sec. 4.2, the resulting memory will be accountable for a large set of operations, and being the popcount network directly connected to the output port of the system, the latter has to be multiplexed between a standard output, consisting of the SA array's processed results, and the popcount result. To both ease the design and optimize the component for speed, the required multiplexer has been embedded in the hardware description of the bit-counting device. Moreover, with an eye on power consumption, the inputs to the architecture have been provided with a gating scheme. To this extent, the switching activity of the internal nodes of the network can be brought to zero by avoiding signal propagation from the inputs. These enhancements conclude the digital design phase of the bit-counting architecture and result in the final network's structure depicted in Fig. 4.30.

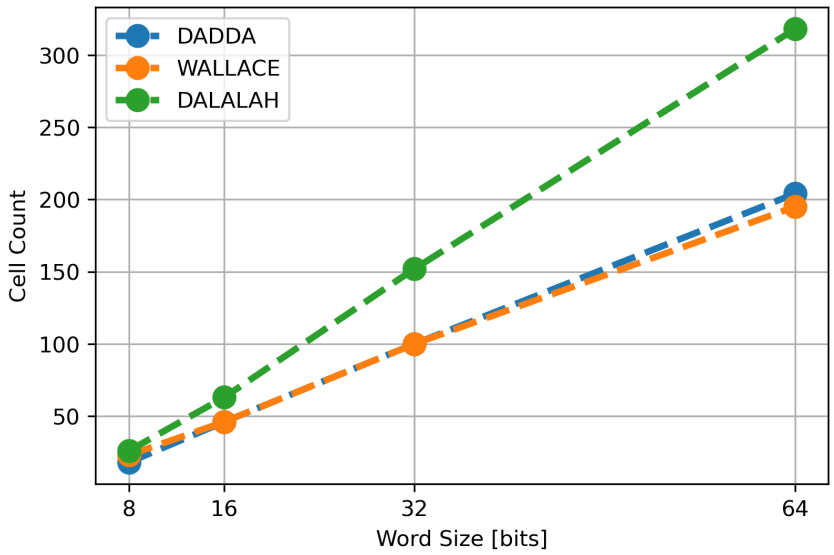
4.4.4 Open-source synthesis and place & route

With the aim of an open-source available memory compiler, the design flow of the bit-counting architecture ought to be achieved through non-commercial tools, unlike its characterization. Thence, as part of the OpenROAD project, the OpenLane [45] tool will be exploited to obtain an automated flow from specifications to physical design.

OpenLane is an automated RTL to GDSII flow based on several open-source components and several custom scripts for design exploration and optimization. The flow performs full ASIC implementation steps from RTL down to GDSII easing the IC design, allowing to overcome the limitations imposed by the absence of reliable non-commercial CADs. The starting point of the flow is a Verilog hardware description of the design and a configuration text file to be filled with constraints regarding the different steps of the stream. Synthesis constraints allow to target a specific figure of merit for the design optimization; due to the significant propagation delay overhead inserted by the popcount operation in the memory access time, it has been decided to settle on a delay-driven synthesis methodology. Other noteworthy synthesis constraints regard input pins capacitances, which have been

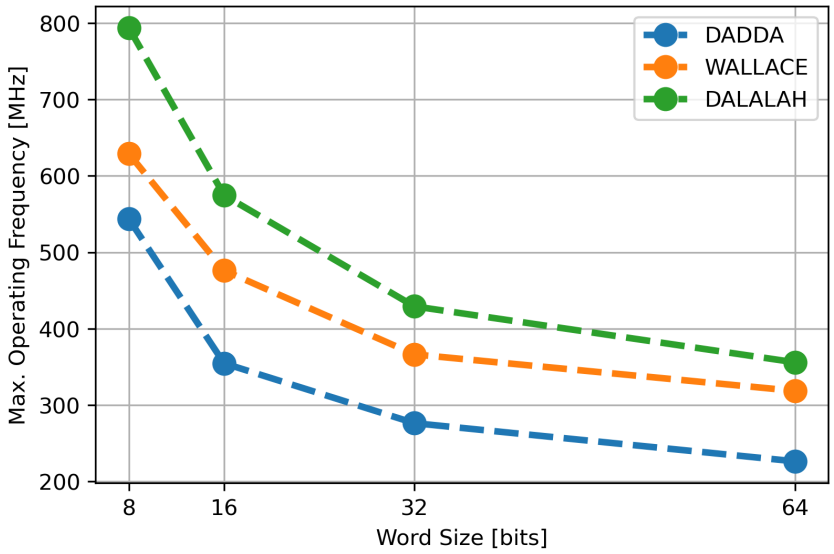


(a) Area characterization.



(b) Cell count characterization.

Figure 4.29: Popcount Networks - Synthesis and Characterization.



(c) Propagation delay characterization.

Figure 4.29: Popcount Networks - Synthesis and Characterization.

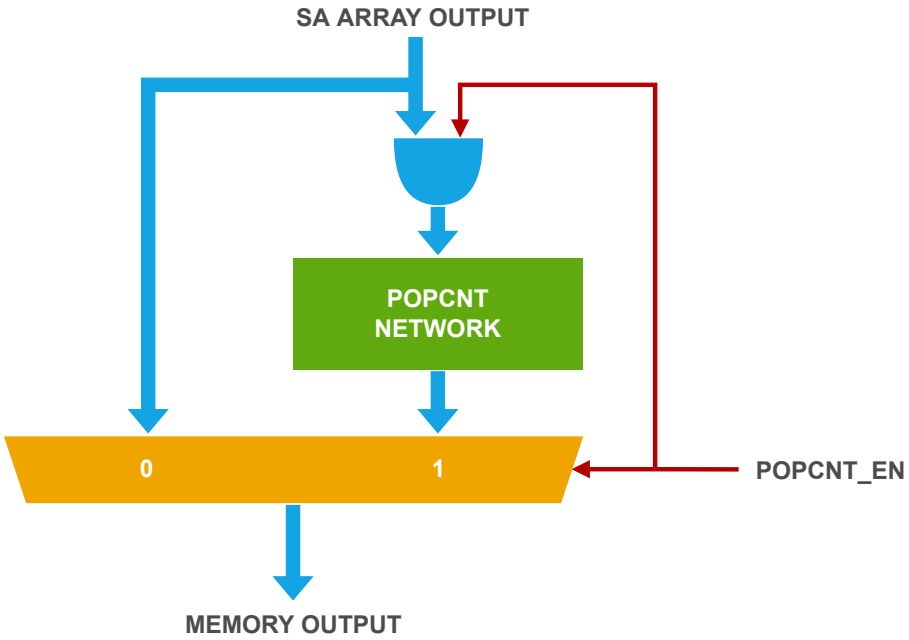


Figure 4.30: Popcount Networks - Final Architecture.

set to the input one of an X4 and X2 inverter for data pins and the enable signal respectively. The formers match the load capacitance used in the SA characterization while the latter eases the design of the control unit drivers. Constraints on the output pins capacitance instead have been tightened up since the output of the component translates into the main output of the system, hence has to offer a good driving capability; for this reason, also the output load capacitance has been set to the input one of an X4 inverter. Tab. 4.4 resumes the post-synthesis results of the given device for a 32-bit word size.

Moving to the physical phase of the design flow, relevant constraints regard the die area and aspect ratio. These can be computed knowing the size of the bitcell array: all the column circuitry in OpenRAM (sense amplifiers, write drivers and pre-charge transistors) are carefully aligned to fit below the storage bank without exceeding the boundaries of the given region. Hence, this allows to automatically size the horizontal dimension of the die at runtime while compiling the memory. Unfortunately, it is not equally straightforward for what concerns the vertical dimension. Wrong sizing of the die may result in flow convergence failure while performing placing and/or routing, this lessens the design automation of the bit-counting device, requiring a human-in-the-loop to tune the die size to successfully conclude the design of the component. Although not specific for each possible configuration of compiled memory, results of post place and route device characterization can be found in Tab. 4.4 to provide hints on how this step may worsen the performance of a netlist in a die with a characteristic aspect ratio. The design flow applies to a 128-bits column bitcell array of 32-bits words, resulting in a 4:1 multiplexing factor and a width of 332µm. To efficiently succeed at the device's physical design, a vertical dimension of 50µm was required, resulting in an aspect ratio of 6.64 and a total die area of $16.6 \times 10^3 \mu m^2$.
As stands out from the results, performing place and route on a die with a high aspect ratio might hinder an efficient floorplanning optimization as it happens for typical sizing (aspect ratios in the range of 1-2), resulting in routing congestion and additional buffers insertion which raise both area occupation and propagation delay.

Table 4.4: Dalalah Bit-Counting Network - Hardware Implementation Results

Step	**Area (µm²)(%)**[a]	**Cells**	$\mathbf{t_{pd,MAX}}$ **(ns)**	$\mathbf{f_{MAX}}$ **(MHz)**
Post-Synthesis	3827 (27%)	241	3.57	280.11
Post-P&R	13594 (97%)	326	4.16	240.38

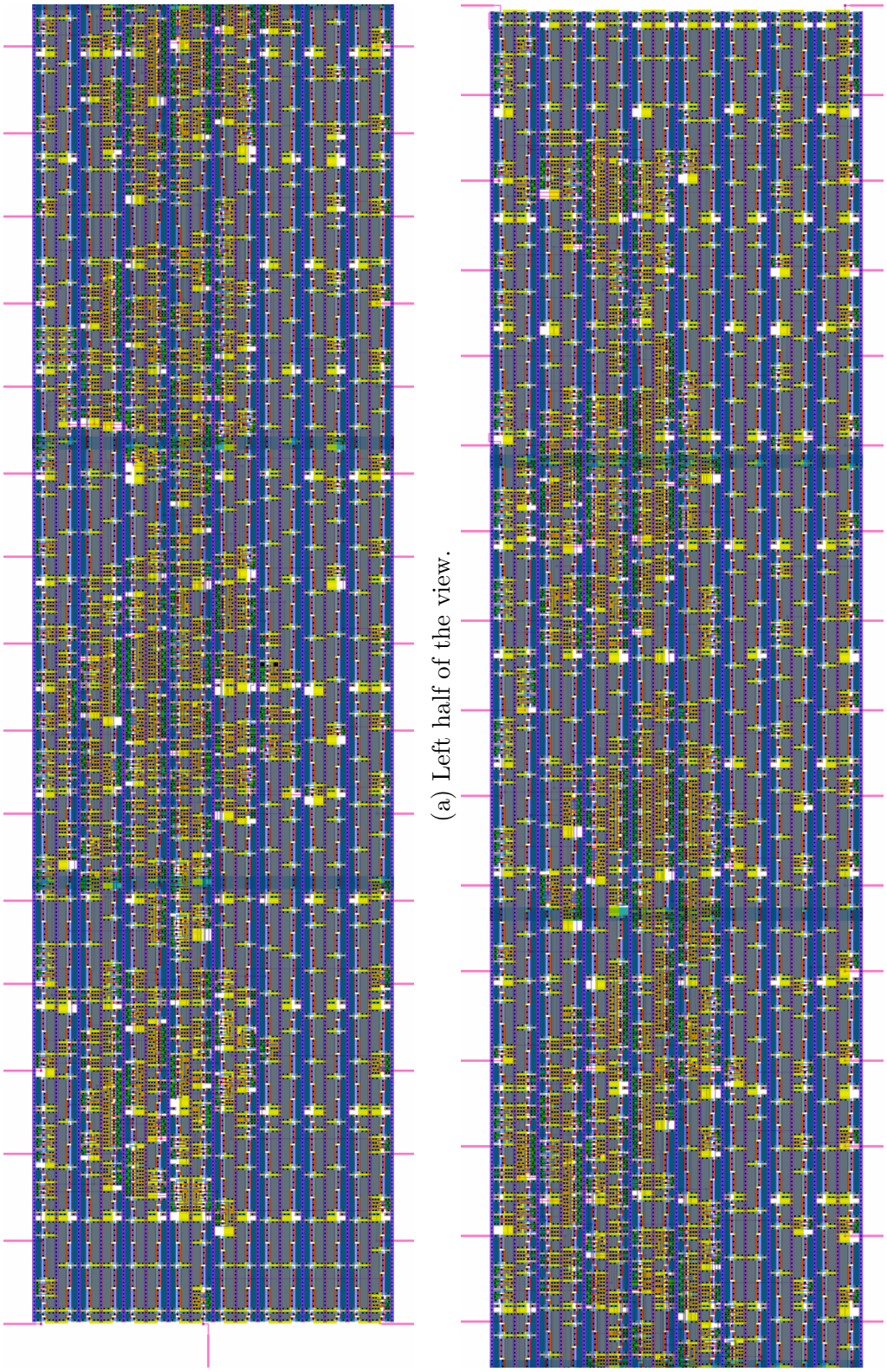
[a]Die area utilization percentage.

4.4.5 Layout

Finally, Fig. 4.31 depicts the layout view of the processed netlist. Due to the aforementioned remarks on its placing inside the overall memory macro, data input pins have been placed on the upper side of the die and their spacing matches the one of SAs in the array, allowing for a straightforward routing of the two components. The same thing applies to data output pins, with a spacing equal to the one of the data DFF array below the popcount network. The enable signal is on the left side of the die to ease the connection to the main control bus.

4.5 Control logic

The final building block to customize to unlock IMC features in the SRAM array introduced in this work is the control logic. In charge of providing the correct signal generation for optimal timing of the overall system, the control logic can be analyzed by looking at the array's required signals and the system's inputs.
The compiler's base input control signals consist of: a *chip select* (CS) for the component wake up, a *write enable* (WE) to choose between a read or a write operation and a *clock* (CLK) for the device's synchronization. Due to the availability of different operations to be performed by the computing SA array, 3 additional signals have to be fed to the control logic: a 2-bit *MODE* word for the SA's 4-to-1 mux select bits and a *CINN* signal for the 2's complement operation. Their relation with the SA array control signals is trivial and the resulting logic elements can be



(a) Left half of the view.

(b) Right half of the view.

Figure 4.31: Popcount Network - Layout View.

seen in Fig. 4.33. The same considerations apply for the popcount network's enable signal, to be asserted only when the XNOR mode of the array is selected. Given the feedback loop behaviour explained in Sec. 4.3.3, the control logic implementation for the decoders enable signals is straightforward and is shown in Fig. 4.33.
The set of operations that can be performed by the memory can now be classified into two groups: standard read and write and IMC computations. Due to the difference in the protocols between the two, a further control signal is required to turn off some circuitry during standard cycles; by loading a logic 0 in the DFF related to the *PULSE* signal, the second decoder is gated and only the first pulse will be issued. A further operation performed by the *PULSE* signal is to de-activate the pulsed assertion of the first decoder during a write operation; a short pulse as the one required for BLC may not be wide enough to safely alter the content of the bitcell, hence the system has to revert to a standard write operation protocol to increase reliability.

4.5.1 Timing diagrams and pins configuration

Done with the control logic enhancement one obtains the pins truth table in Tab. 4.5 and the timing diagram in Fig. 4.32. Diagram for standard read and write operations are omitted in this section and can be found in Sec. 3.1.

4.5.2 Layout

OpenRAM allows for easy customization of the control section by providing a regular structure based on different rows for each signal's generation. Starting from a buffered DFF, the input control signals and their complemented versions get connected to a vertical bus spanning across the whole height of the section. Each track of the bus is a "primitive" signal used by one or more rows for the generation of a definitive one. Thanks to the *factory* introduced in Sec. 3.1, a minimal set of logic gates can be automatically generated in both spice instances and layout geometries,

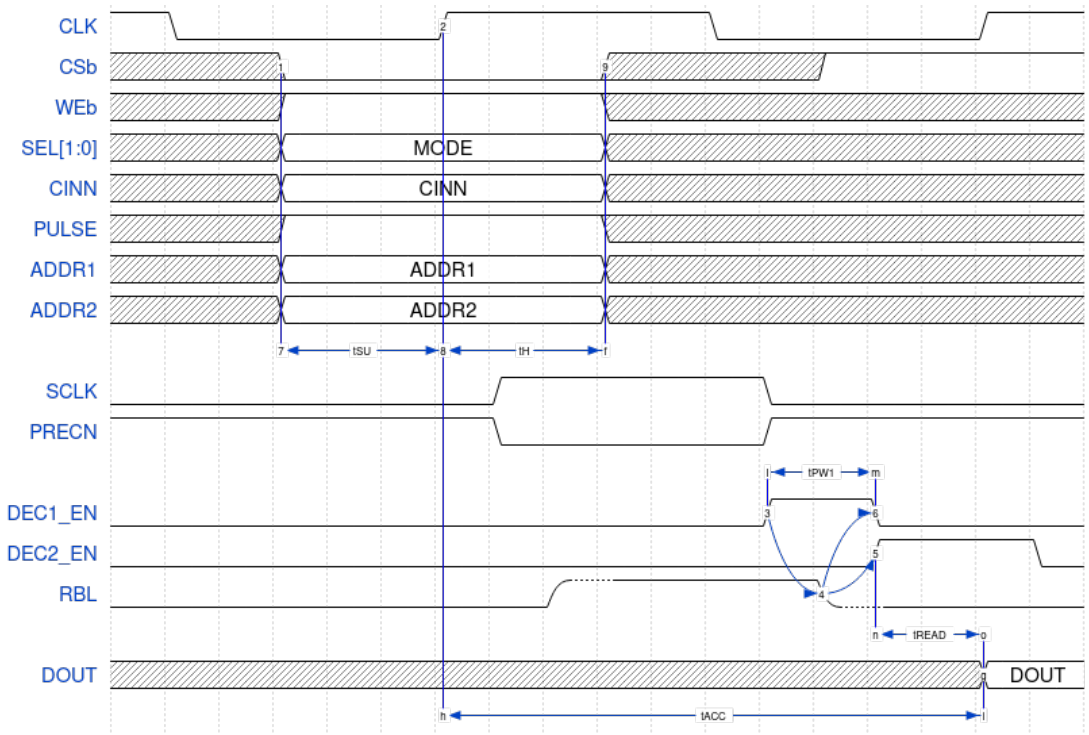


Figure 4.32: Timing Diagram - Waveforms for Pulsed Bitline Computing.

Table 4.5: Bitline Computing SRAM - Command Table

PULSE	**WEn**	**SEL[0]**	**SEL[1]**	**CINN**	**Operation**
0	1	0	0	-	Standard read
		0	1		1's complement
		1	0		–
		1	1		Read check
0	0	-	-	-	Standard write
1	1	0	0	-	Bitwise AND
		0	1		Bitwise NOR
		1	0	0	Sum + Increment
				1	Sum
	1	1	1	-	Bitwise XNOR

also the required logic drivers can be automatically sized to optimize the propagation delay across the system knowing the load capacitance of a given signal. With help of a library of methods, the desired logic gates can be placed alongside one another and connected both between themselves and the vertical bus. Since the height of the rows is a design parameter and the one of the pulse generator cell is moderate, the latter has been inserted in the row accountable for the generation of the first decoder's enable signal, in order to save area.

Fig. 4.34 shows the final layout of the control logic instance. In the lower-left corner one can appreciate the array of buffered DFFs, at center, spanning across the cell, the vertical bus, in the lower right corner the drivers for clock buffering and at the right of the bus all the rows for signal generation.

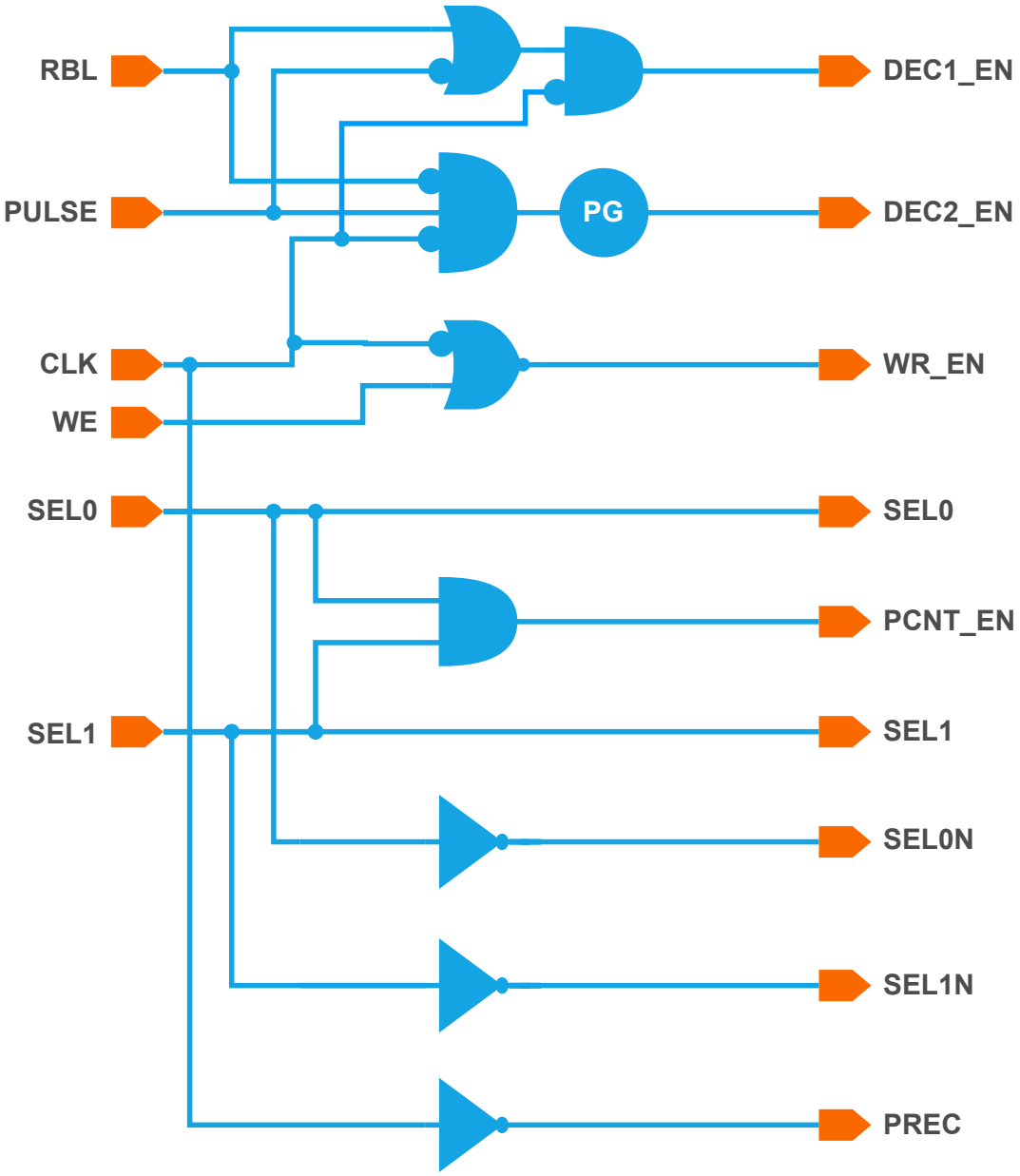


Figure 4.33: Control Logic - Logic Level Schematic View.

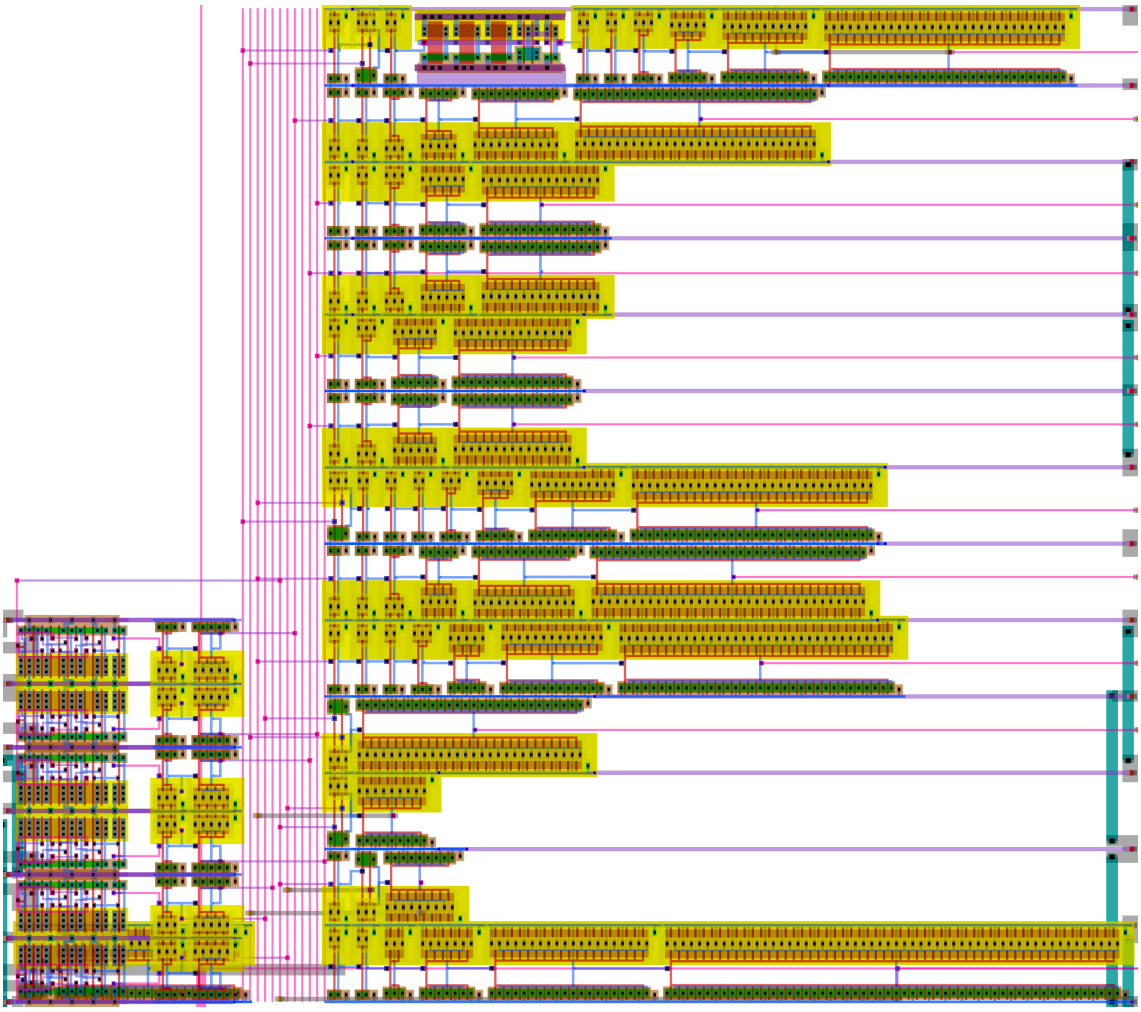


Figure 4.34: Control Logic - Layout View.

Chapter 5

Simulation and results

All the features presented in the previous chapter have been implemented in the OpenRAM memory generator to produce a 128x128 bits array of 32-bit long words. This resulted in a 4:1 column multiplexing and a 32-bits Dalalah popcounting network as stated in Sec. 4.4. The final number of address pins is 16, 7 for each row decoder, while the two LSBs of the word are shared between the two addresses. This limits the flexibility of choosing a two words combination while performing bitline computations but is due to the presence of a column multiplexing. Given the data word size, an array of 32 DFFs is present for input bits while performing write operations. Also, the number of control flops increased due to the presence of additional signals for SAs multiplexers and bitline computing mode enabling.
This section will present a characterisation of the described memory array in terms of area occupation, operating frequency and energy consumption, comparing results regarding the two different operating modes: bitline computing and standard read. All simulations have been performed in SPICE on a schematic level netlist of the system; to speed up the simulation process, only the bitcell array has been described in terms of parasitics and modelled through RC lumped elements obtained from the electrical and geometrical properties of the PDK layers.

5.1 Area occupation

The layout view of the architecture can be seen in Fig. 5.1, while Fig. 5.2 breaks down the location of the various components in the macro. The first thing standing out from the layout view is the high aspect ratio of the bitcell array (≈ 1.95). The

ultimate goal of adopting a column multiplexing scheme is to achieve an array shape where the row side length is roughly the same as column one while the size of the array increases. This avoids the presence of excessively long BLs or WLs which may ultimately slow down the system's performance. A key factor defining the shape of the array is the one of the single bitcell; by recalling the layout view of the 6T cell in Fig. 4.2b one obtains an aspect ratio of approximately 2.26. Although the memory compiler tries to achieve the best multiplexing scheme to yield an array shape as square as possible, it has to face a limited number of multiplexing factors. This, combined with an aspect ratio of the bitcell larger than one, results in an array's shape far from the desired one.

Tab. 5.1 beaks down the area occupation of the whole macro as well as the single components, ultimately delivering the partitioning percentages. To evaluate the effects of BLC features insertion on the size of the instance, a standard 6T SRAM has been generated via OpenRAM, yielding an area occupation of 327 844 μm^2, which the insertion of BLC features increases by 16.63%.

Table 5.1: BLC SRAM - Area occupation and partitioning

Cell	**Area (um^2)**	**%**
Bitcell array	227 340	59.55
Popcount network	16 599	4.35
Col. circ. + SAs	17 283	4.53
Decoder 1	33 066	8.66
Decoder 2	33 066	8.66
Control logic	10 723	2.81
BLC-SRAM	381 752	

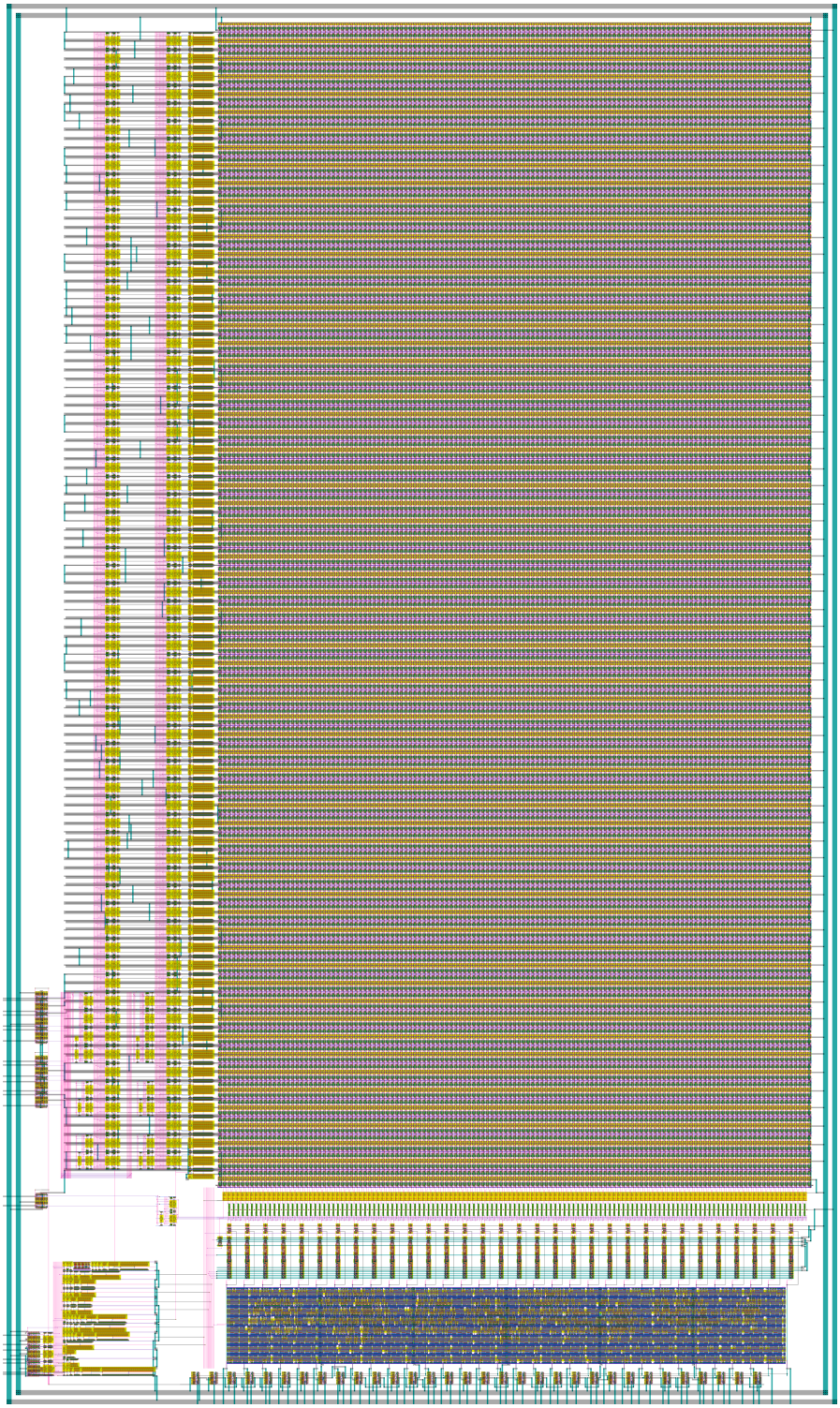


Figure 5.1: Layout view of the 32-bits 128x128 BLC SRAM array.

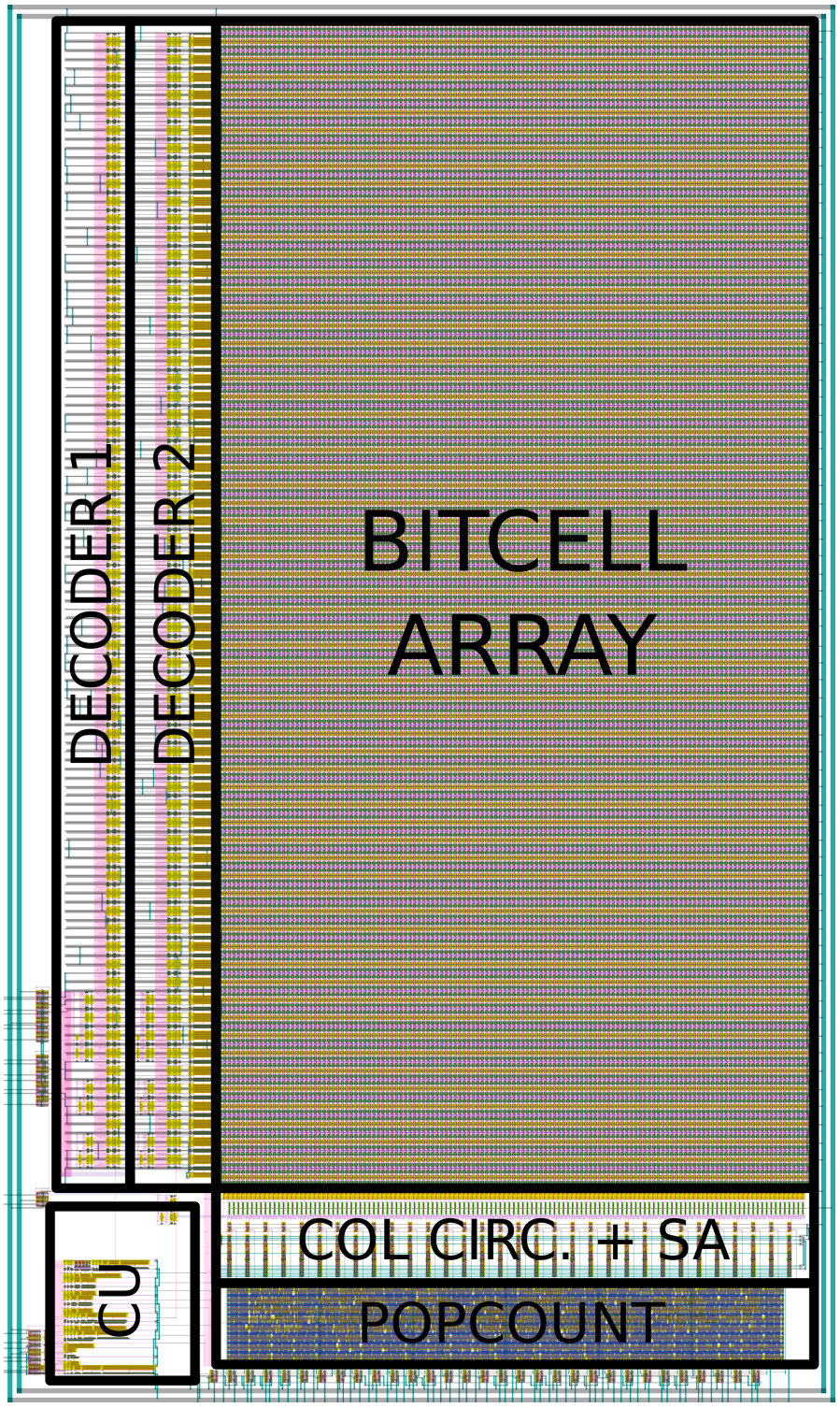
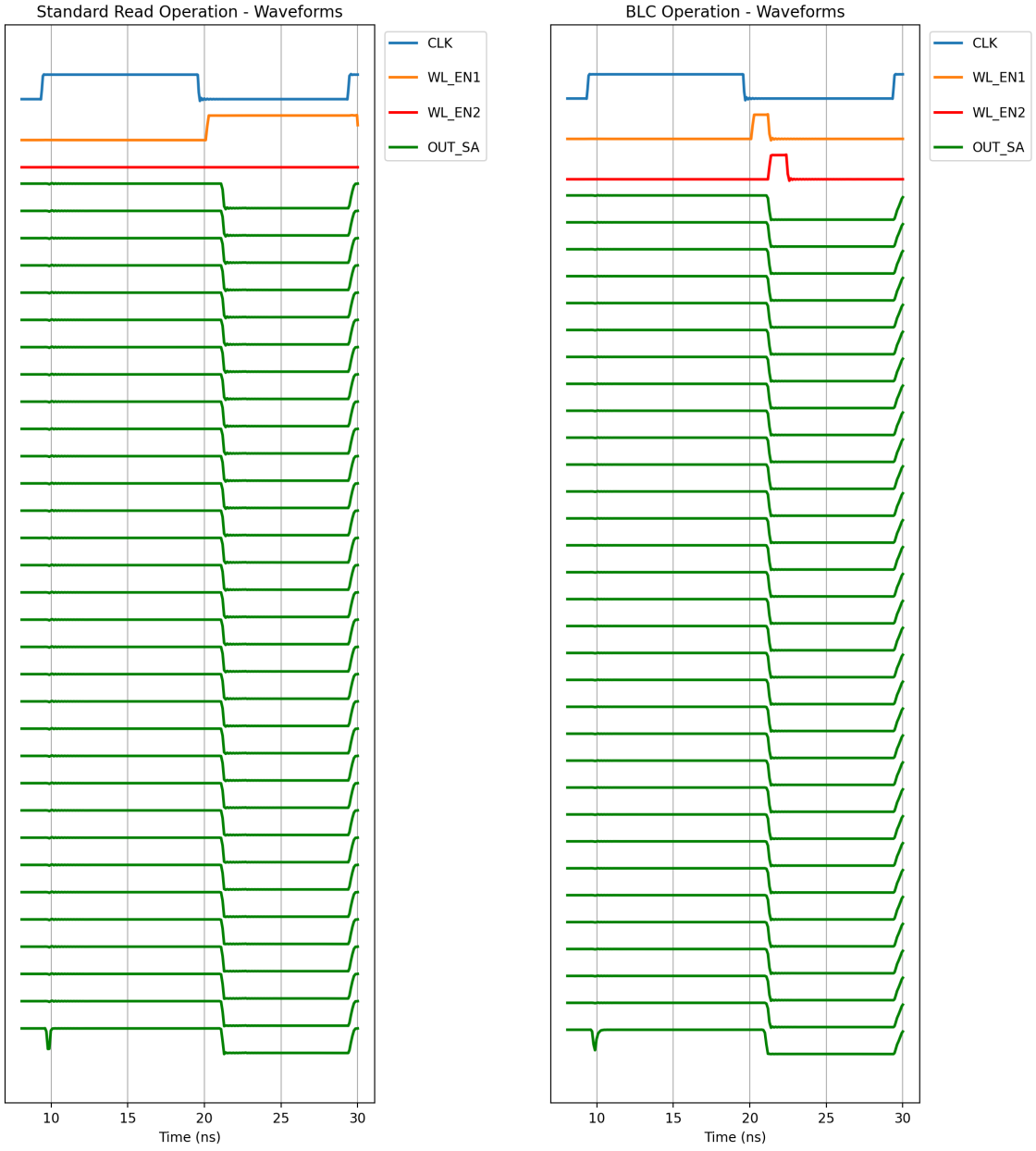


Figure 5.2: Layout view of the 32-bits 128x128 BLC SRAM array with components highlight.

5.2 Simulation results

The SPICE waveforms of the simulated architecture can be seen in Fig. 5.3b and Fig. 5.3a for BLC mode and standard read mode respectively. Green waveforms represent the output of the SA array from the LSB (top) to the MSB (bottom). The content of the words to access was tuned to test the BLC scheme in all the possible combinations of 2-bits (00, 01, 10, 11) within a single operation. For standard read operation instead, the array was initialized at random values. Simulations were performed in the nominal corner (TT, typical-nFET and typical-pFET) with a 1.8V supply voltage and a clock frequency of 50MHz.

One thing standing out from both simulations is that even for BLs experiencing no discharge, the output of the SA switched to logic 0 after a certain time. Investigating deeper, one can observe how the Q_N node of the amplifier charges even when the BL is at logic 1 (Fig. 5.5). This is due to the voltage drop caused by the n-channel pass transistor of the column mux circuit. When propagating a logic 1 (i.e V_{DD}) the output of the multiplexer will exhibit a degraded value, resulting from the transistor's threshold voltage. This doesn't usually translate into an issue in differential sensing schemes, but given the novel amplifier's topology introduced in this work, a study is required. Being the output of the multiplexer the input of the amplifier, the input pFET of the latter will experience a gate-source voltage equal to $V_{DD} - V_{TH,n}$ when the BL is at logic 1. Since the threshold voltage of the pFET has been lowered to increase its sensibility to BL swings, the voltage drop of the nFET is now sufficient to drive the former transistor into the near-subthreshold region. The resulting sub-threshold current proved itself enough to charge the capacitive node Q_N up to the inverter's trip voltage before the end of the cycle, ultimately causing the switching of the amplifier (Fig. 5.5). To reduce the amount of sourced current, the nFETs of the column multiplexer have been made low threshold transistors, soothing the degradation of the propagated value. Simulation waveforms of this new configuration can be seen in Fig. 5.6 for a standard read operation when accessing a word storing the value "1". As stands out from this last simulation, the lowered threshold voltage of the nFET is enough to keep the pFET off, avoiding the charge of Q_N. This solution, however, relies on the difference between the threshold



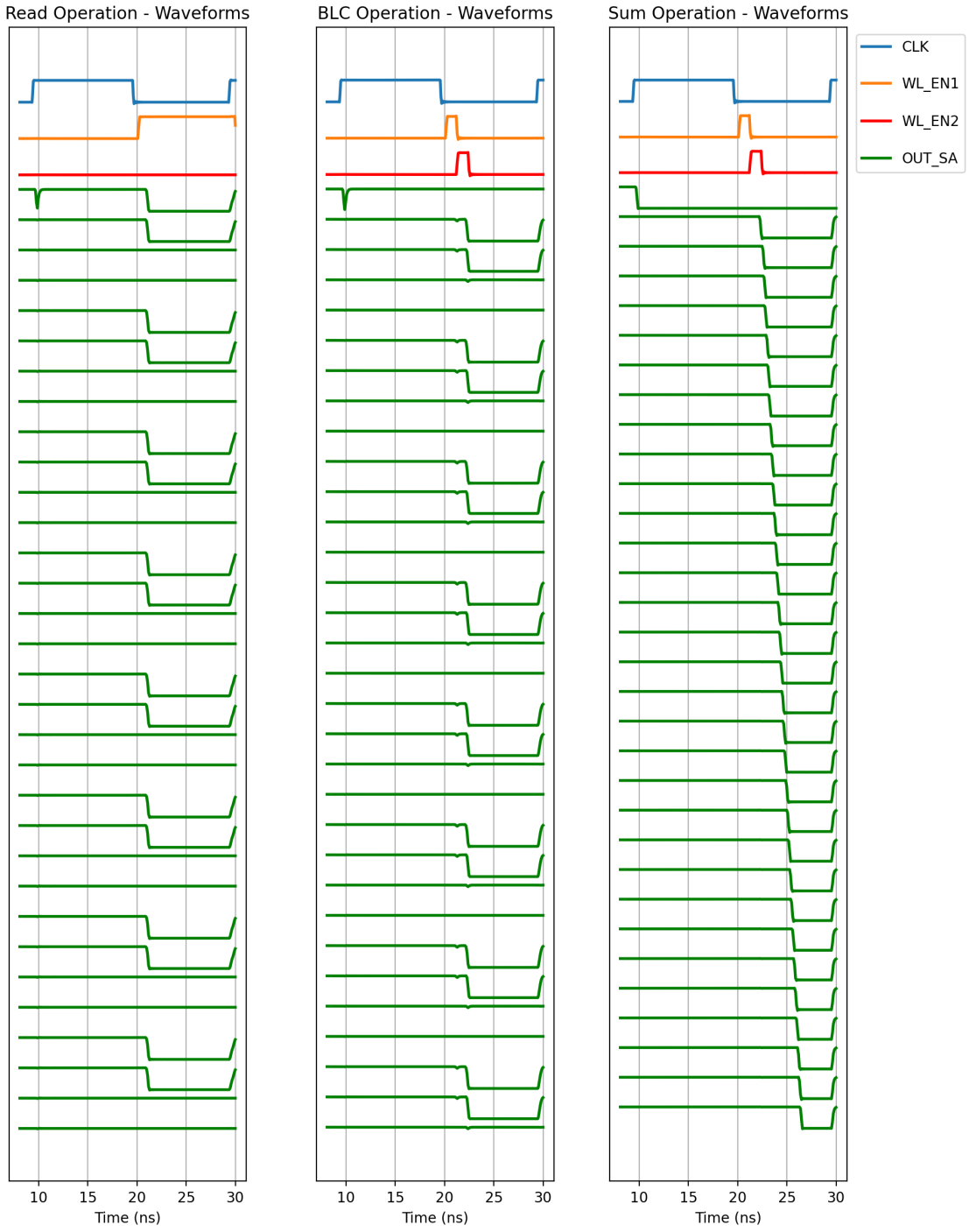
(a) (b)

Figure 5.3: Output waveforms for read and BLC operating modes: pass-transistor mux.Green waveforms represent the output of the SA array from the LSB (top) to the MSB (bottom).

voltage of the two FETs, strongly subjected to process variations. The condition to be satisfied is $V_{TH,n} < V_{TH,p}$, which could be violated in the SF process corner (slow nFET, fast pFET). To increase the reliability of the system, it was decided to allow a "strong" propagation of logic 1 across the column multiplexer, hence shifting the circuit implementation from a pass-transistor to a transmission-gate one. The size of the mux's pFETs was chosen to match that of the nFETs; thanks to the slow discharge rate of the BL this allowed to avoid unnecessary area overhead. Results of this latter simulation are in Fig. 5.7. Finally, this last configuration avoided the unwanted charge of the amplifier's capacitive node, thus completely switching off the input pFET in all the process corners. New simulation waveforms of read, BLC and sum mode can be seen in Fig. 5.4

Concerning BLC mode, obtained waveforms stick to ones assumed during the design phase. Fig. 5.9 shows resulting waveforms for all the 2-bit combinations of a computing column: (1,1), (1,0), (0,1) and (0,0) from bottom to top. As desired, the closed-loop replica bitline scheme successfully de-asserts the enable signal of the first decoder while triggering the pulse generator. Moreover, the BL does not experience full discharge after the first pulse as desired, although reaching a lower voltage value with respect to the one predicted in Sec. 4.3.3 due to control logic propagation delays. The developed BL voltage is now sufficient to turn on the pFET of the second accessed cell if the latter stores logic 1. The current sourced by the latter leads to a transient when the line's voltage is raised to $V_{DD} - V_{TH,n,WL}$. This turns the access transistor of the bitcell off and isolates it from the BL. However, thanks to the closed loop gating, the transient duration is small compared to the cell's access time, thus avoiding data corruption.

Regarding a standard read operation instead, the enable signal of the first decoder is stretched to boost the array performance, hence reducing the read-access time. The second decoder, instead, is gated to prevent double word access. For testing the sum mode the words to access have been tuned to allow for a complete carry propagation from the CINN signal to the COUTN one of the SA array. This latter propagation can be seen in Fig. 5.9.



(a) (b) (c)

Figure 5.4: Output waveforms for read, BLC and sum operating modes: transmission-gate mux. Green waveforms represent the output of the SA array from the LSB (top) to the MSB (bottom).

5.3 Operating frequency and energy consumption

The designed architecture has been characterised in all the available operating modes both in propagation delay and energy consumption. Simulations have been performed using SPICE simulators on the schematic-level view of the memory where only the bitcell array has been annotated with parasitic devices. Results for nominal supply voltage and temperature can be found in Tab. 5.2 for TT, FF and SS process corners. Tab. 5.3 presents results for supply voltage (±10% of nominal value) and temperature (commercial applications range: 0°C - 70°C) variations in the typical corner. Finally Tab. 5.4 delivers the BLC SRAM characterization in the TTTT, SSSS and FFFF performance corners.

As stands out from the characterization, a significant difference exists between the system's performance in different operating modes. As expected from a common SRAM architecture, while performing standard read operations the clock period is constrained by the time needed to access the bitcell array. For bitwise BLC operations instead, the latter has to be accessed twice. The degradation of the operating frequency by a factor of 1.35 instead of roughly 2 points out the benefit from a double decoder implementation, avoiding crossing the whole decoding circuitry for a second time. The same applies from an energy consumption perspective, resulting in an increment by a factor of 1.62.
Popcount and 2's complement sum instructions instead exhibit a large overhead in propagation delay due to bit-counting network and RCA logic respectively. The ultimate result is a propagation time across these circuits way larger than the bitcell's access time, stalling the whole system. Differences between the operating modes stand out also from an energy consumption comparison. Due to the significant increase in the network's nodes switching activity when propagating results across both bit-counting and ripple-carry networks, the dynamic power consumption experiences a significant increment, resulting in a 2x energy consumption with respect to a standard read operation in the nominal corner.

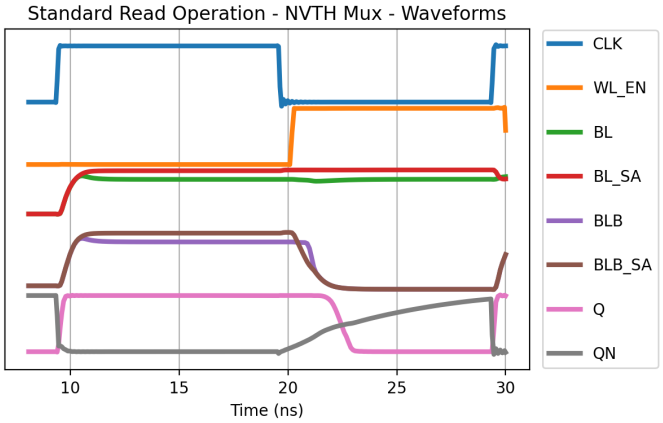


Figure 5.5: Sense amplifier failure in nominal-threshold pass-transistor mux.

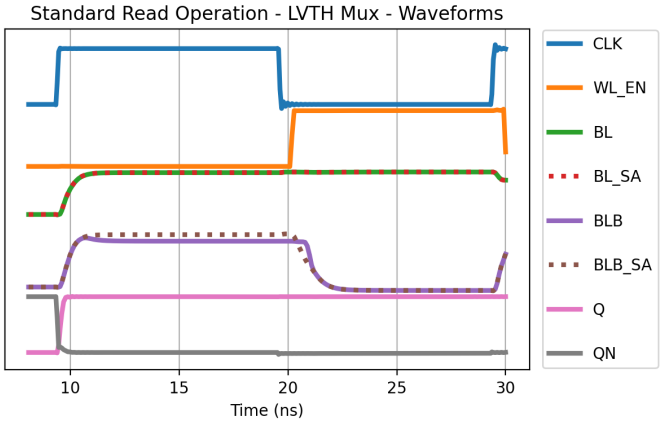


Figure 5.6: Sense amplifier behaviour in low-threshold pass-transistor mux.

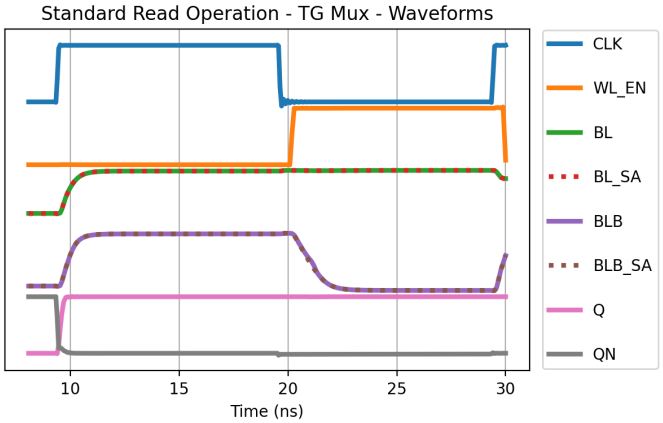


Figure 5.7: Sense amplifier behaviour in transmission-gate mux.

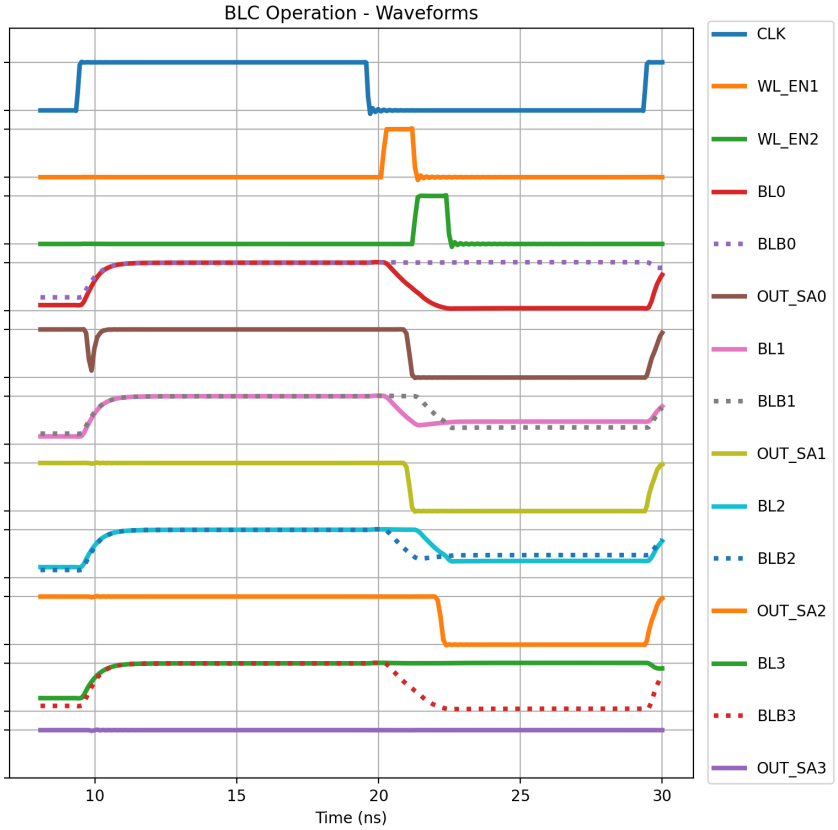


Figure 5.8: BLC SRAM waveforms for BLC mode.

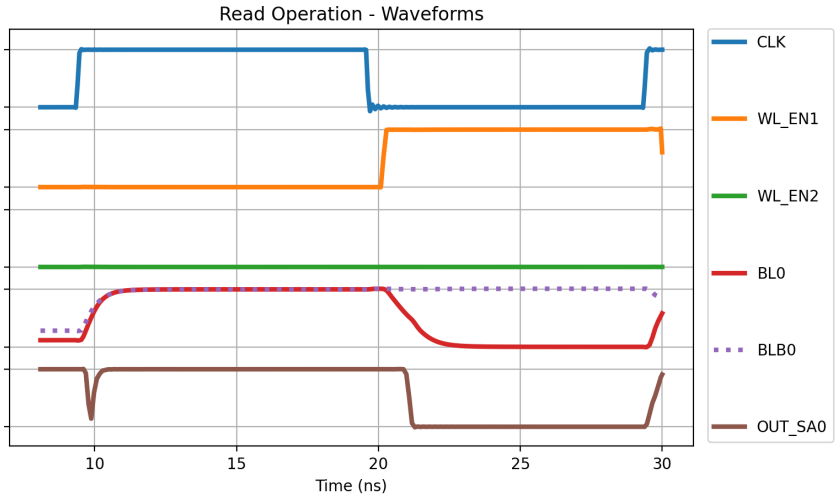


Figure 5.9: BLC SRAM waveforms for standard read mode.

Table 5.2: BLC SRAM Characterization - $V_{DD} = 1.8V$, T=300K

	Process Corner - TT			
Mode	**f_{MAX} (MHz)**	**t_{ACC}[a] (ns)**	**E/op (pJ)[b]**	**P_{LEAK} (mW)**
Std. Read	237	2.11	29.31	1.47
Bitwise Ops.	176	2.83	47.46	
Pop-Count	85	5.83	59.82	
Sum	65	7.69	58.63	
	Process Corner - SS			
Mode	**f_{MAX} (MHz)**	**t_{ACC}[a] (ns)**	**E/op (pJ)[b]**	**P_{LEAK} (mW)**
Std. Read	176	2.83	16.45	0.61
Bitwise Ops.	108	4.64	25.55	
Pop-Count	65	7.64	37.90	
Sum	48	10.45	25.58	
	Process Corner - FF			
Mode	**f_{MAX} (MHz)**	**t_{ACC}[a] (ns)**	**E/op (pJ)[b]**	**P_{LEAK} (mW)**
Std. Read	313	1.594	29.64	1.54
Bitwise Ops.	213	2.344	43.59	
Pop-Count	94	5.332	55.95	
Sum	91	5.491	44.29	

[a]From CLK falling edge to DOUT stable.
[b]$T_{CLK} = 20$ns.

Table 5.3: BLC SRAM Characterization - Process Corner TT, Environmental Variations

	V_{DD}+10%, T=300K			
Mode	**f_{MAX} (MHz)**	**t_{ACC}[a] (ns)**	**E/op (pJ)[b]**	**P_{LEAK} (mW)**
Std. Read	274	1.82	49.68	1.79
Bitwise Ops.	178	2.81	76.90	
Pop-Count	86	5.81	91.73	
Sum	80	6.23	70.13	
	V_{DD}-10%, T=300K			
Mode	**f_{MAX} (MHz)**	**t_{ACC}[a] (ns)**	**E/op (pJ)[b]**	**P_{LEAK} (mW)**
Std. Read	210	2.38	19.71	0.79
Bitwise Ops.	131	3.83	32.02	
Pop-Count	73	6.83	41.90	
Sum	57	8.71	35.59	
	V_{DD}=1.8V, T=273K			
Mode	**f_{MAX} (MHz)**	**t_{ACC}[a] (ns)**	**E/op (pJ)[b]**	**P_{LEAK} (mW)**
Std. Read	246	2.03	31.72	1.19
Bitwise Ops.	155	3.23	43.04	
Pop-Count	80	6.23	57.86	
Sum	69	7.29	43.75	
	V_{DD}=1.8V, T=343K			
Mode	**f_{MAX} (MHz)**	**t_{ACC}[a] (ns)**	**E/op (pJ)[b]**	**P_{LEAK} (mW)**
Std. Read	247	2.03	23.15	0.76
Bitwise Ops.	154	3.24	34.97	
Pop-Count	80	6.25	44.86	
Sum	68	7.32	34.87	

[a]From CLK falling edge to DOUT stable.

[b]T_{CLK} = 20ns.

Table 5.4: BLC SRAM Characterization - TTTT, SSSS and FFFF Corners

	Process Corner TT, V_{DD}=1.8V, T=300K			
Mode	**f_{MAX} (MHz)**	**t_{ACC}[a] (ns)**	**E/op (pJ)[b]**	**P_{LEAK} (mW)**
Std. Read	237	2.11	29.31	1.47
Bitwise Ops.	176	2.83	47.46	
Pop-Count	85	5.83	59.82	
Sum	65	7.69	58.63	
	Process Corner SS, V_{DD}-10%, T=343K			
Mode	**f_{MAX} (MHz)**	**t_{ACC}[a] (ns)**	**E/op (pJ)[b]**	**P_{LEAK} (mW)**
Std. Read	149	3.36	21.85	0.91
Bitwise Ops.	89	5.65	27.71	
Pop-Count	58	8.64	36.36	
Sum	39	12.71	26.52	
	Process Corner FF, V_{DD}+10%, T=273K			
Mode	**f_{MAX} (MHz)**	**t_{ACC}[a] (ns)**	**E/op (pJ)[b]**	**P_{LEAK} (mW)**
Std. Read	339	1.47	56.54	1.94
Bitwise Ops.	255	1.96	51.18	
Pop-Count	101	4.97	67.24	
Sum	106	4.74	51.38	

[a]From CLK falling edge to DOUT stable.
[b]T_{CLK} = 20ns.

Chapter 6

Conclusions and future perspectives

Von-Neumann architectures fueled the age of modern computing for decades. However, the advent of data-intensive applications already started to raise new challenges in circuit and system design. In-Memory Computing is considered a worthy alternative to conventional computing platforms to overcome the memory-wall and boost the system's performances. Efficient and reliable IMC implementations are currently a popular branch of research but most of them require ad-hoc design tunings together with pronounced designers' expertise. Another factor hindering the IMC deployment in commercial applications is the low reliability of certain implementations sensitive to process variations or dealing with read-disturbs.

To contribute to the research going on and to unlock a fast design space exploration for innovative systems relying on IMC, this work delivered the complete design of a BLC 6T-SRAM on a 130nm technology node together with a memory-generator implementation. The widespread problem of read-stability in concurrent word accessing was suppressed by adopting a sequential pulsed WL assertion, resulting in the absence of cells content corruption in all the simulated process corners. A reliable pulse generation was achieved by employing an RBL-based closed-loop control of the pulse width. This scheme proved itself a valuable choice as it avoided the superposition of the two pulses in every process corner, thus unlocking a safe pulsed WL assertion even when dealing with process variations.

Some novel techniques were proposed throughout this work with the aim of boosting the performance of the array. A novel sensing scheme has been introduced which

reduces the footprint of the amplifier by 45% as well as the energy consumption by 52%. This allowed easing the constraints on the computing part of the component, required for low computational effort neural networks acceleration. Furthermore, the circuit topology of the amplifier allowed the introduction of a novel technique for boosting the performances of pulsed BLC architectures called time-borrowing. To unlock this new feature, the first pulse width has to be tuned to under-drive the sense amplifier. Due to the RBL-determined pulse generation, this work proposed the insertion of replica rows to tune the RBL discharge rate based on the number of simultaneously active replica bitcells per access. This allowed for a pulse width reduction of the 39% while introducing minimal area overhead. However, due to control logic propagation delays, this scheme didn't allow for proper exploitation of the time-borrowing technique. This is to relate to the saturating $1/N$ trend of the RBL discharge rate when adopting the proposed replica rows technique, where N is the number of additional replica bitcells. To unlock a finer-grain tuning, the proposed control mechanism could be merged with existing ones, like reducing the number of dummy cells tapped to the RBL, thus allowing for linear control of the discharge time.

This work also provided the complete design of a popcount network based on an existing optimized architecture. This unlocked the single-cycle acceleration of BNNs inference at the expense of degraded system performance. Investigating deeper the delay distribution across the array, one can understand how the propagation times of the bit-counting network are the key factor slowing down the memory. To soothe this issue, alternative population-count architectures can be adopted with the aim of reducing the logic depth of the network. Another option is to relax the constraint on a single-cycle SRAM architecture. Breaking down the popcount instructions into simpler operations one obtains an XNOR and a bit-count phase. Due to the bitwise nature of the first one, the results will be ready at the same instant at the output of the SAs. Breaking the combinational path by inserting a memory elements array here would address the remaining popcount operation to the next cycle, while allowing the exploitation of the array for other computations. This way the system will behave as a two-cycles architecture only when a POPCNT instruction is issued.

Another configuration slowing down the system performance is the one related to a 2's complement addition. According to simulations, this translated into the operation requiring a longer clock period, more than three times larger than the one involved in a standard read instruction. This degradation is to be addressed to the structure adopted for achieving the desired sum operation. In fact, the constraint of equal-sized gates for the carry-ripple logic severely limits the optimization of the given path. Unfortunately, moving towards a two-cycles implementation is not allowed due to the nature of the implemented adder. Optimizing the sum instruction requires a change in the way of conceiving the computation itself, moving towards different algorithms of circuit implementations.

From a design automation perspective instead, different optimizations can be introduced in the memory-generator. As previously stated, one thing retaining the potentiality of the time-borrowing technique is the propagation delay inserted in the closed-loop control by the control logic. This can be addressed to the lack of proper sizing of the logic gates when automatically generated by OpenRAM. In fact, only the output drivers of these logic functions are optimized with the logical-effort method; internal gates instead exhibit unitary sizes. A further optimization concerning the memory-generator is the routing of the data port, a region of the instance that includes SAs, write drivers, precharge array, column mux and popcount network. The native OpenRAM floorplanning sees all these components tiled at the bottom of the bitcell array (excluding the popcount network), thus adopting a channel routing of the bitlines spanning across the tile. Further routes are needed to connect the popcounter to the output port and the write drivers to the input DFFs. Due to the SA array and popcounter exhibiting routing layers up to M4 (last available metal layer for routing before pads), an efficient routing needed a dedicated tuning, thus inserting a "human-in-the-loop". A way to solve this issue would be to exploit the top of the bitcell array for input circuitry and precharge array while addressing the SA array and popcount network to its bottom. The only drawback of this customization would be the doubling of the column multiplexer for both sides of the bitcell array, thus potentially increasing the area of the memory.

In an age where big-data and artificial intelligence seem to burden the pace of

Moore's law, the struggle for innovative computing architectures is giving birth to brilliant solutions to continue advancing technology for mankind. Being In-Memory Computing considered becoming one of the strongholds of the modern computing era, this work hopes at contributing to its construction with a tiny brick, providing a further understanding of a new way of conceiving hardware computing.

Appendix A

Custom modules for OpenRAM in SKY130

A.1 Schematics

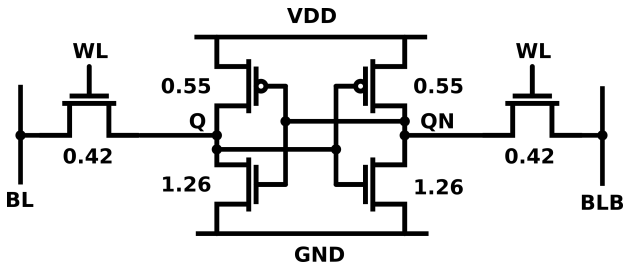


Figure A.1: 6T SRAM bitcell, only FET widths in micrometers are reported, $L_{CH} = 0.15\mu m$ [35].

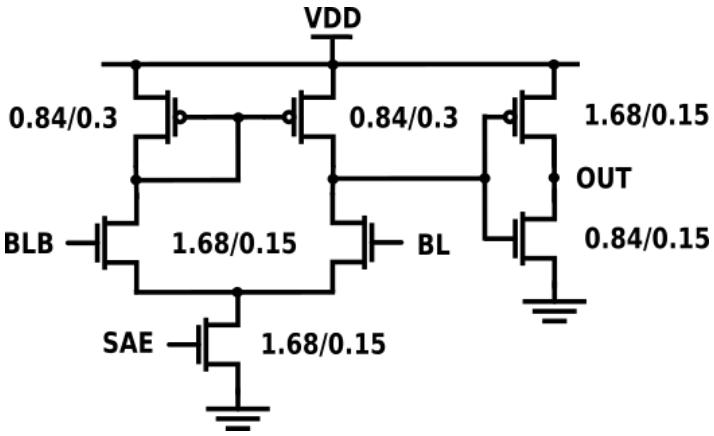


Figure A.2: Sense amplifier, ratios near FETs are to be intended as absolute channel widths over channel lengths in micrometers.

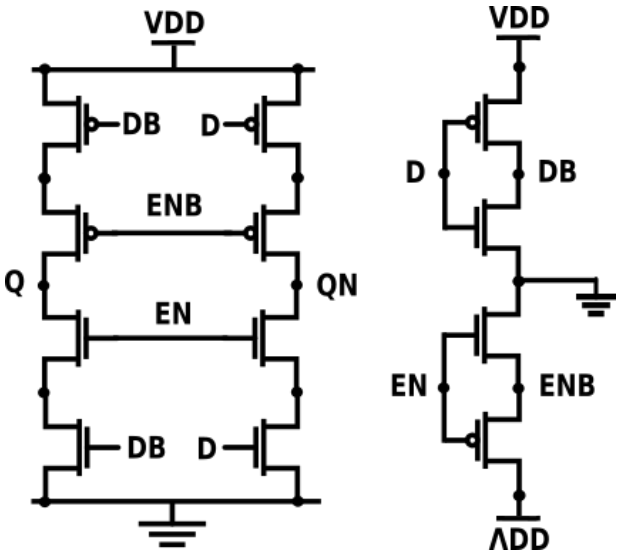


Figure A.3: Write driver, $W_N = 0.42\mu m$, $W_P = 0.84\mu m$, $L_{CH} = 0.15\mu m$.

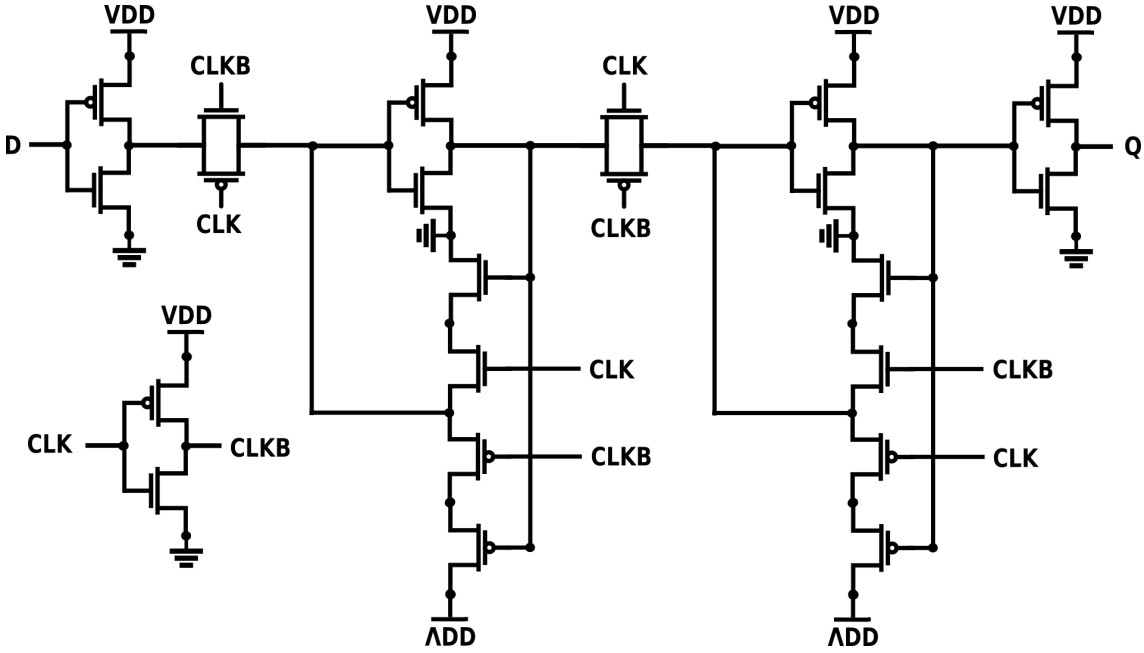


Figure A.4: Write driver, $W_N = 0.42\mu m$, $W_P = 1.35\mu m$, $L_{CH} = 0.15\mu m$.

A.2 Layouts

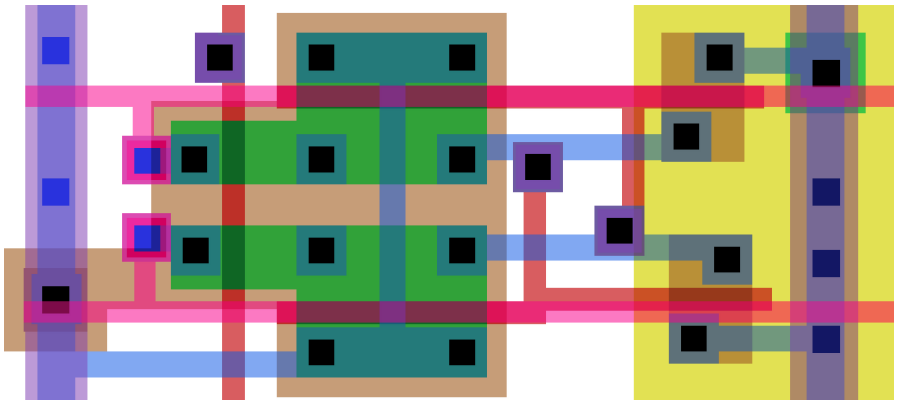


Figure A.5: 6T SRAM Bitcell [35].

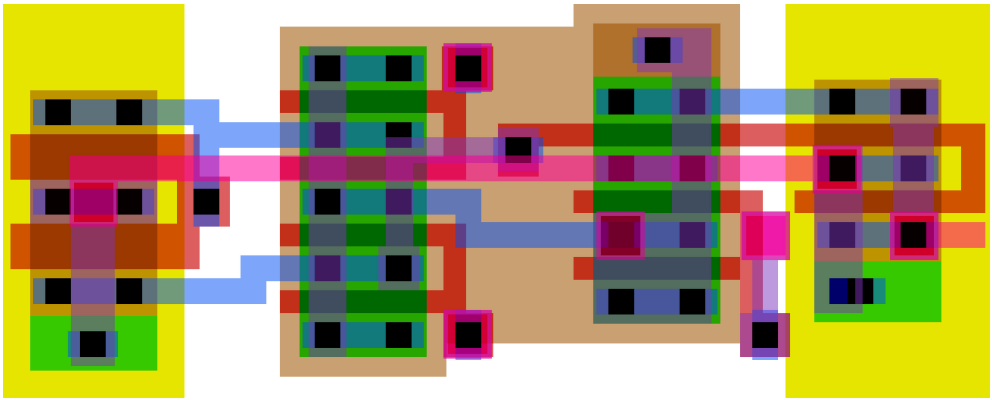


Figure A.6: Sense amplifier.

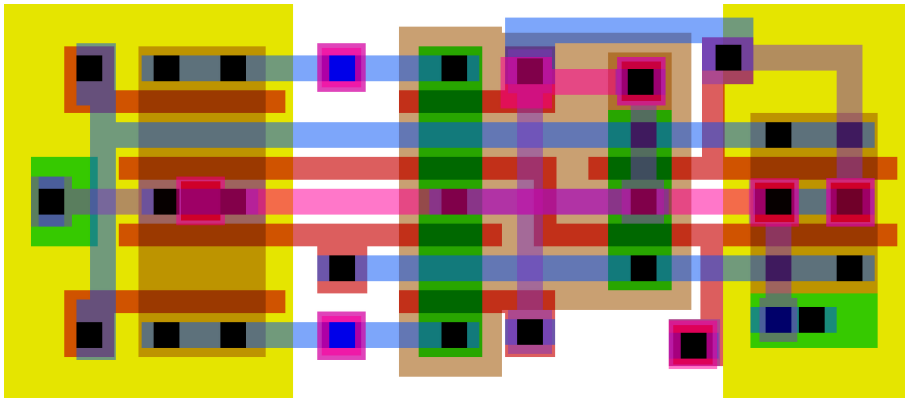


Figure A.7: Write driver.

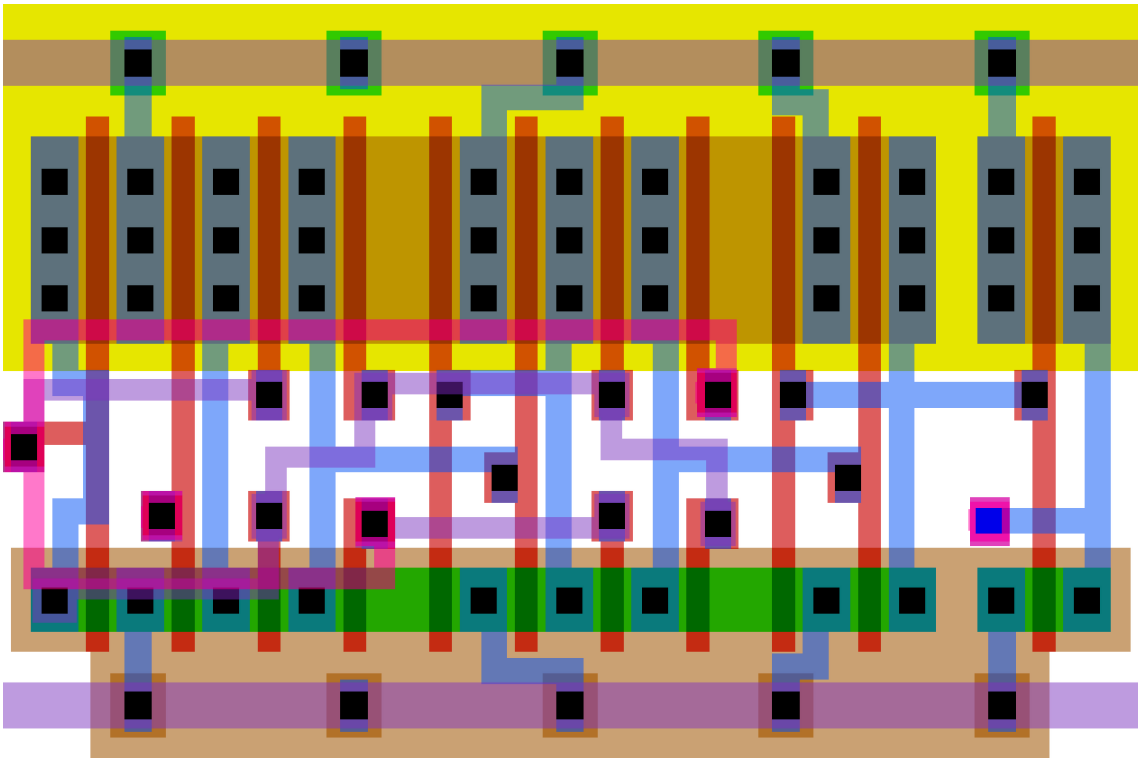


Figure A.8: D-flip-flop.

Bibliography

[1] J. von Neumann, "First draft of a report on the edvac," *IEEE Annals of the History of Computing*, vol. 15, no. 4, pp. 27–75, 1993.

[2] M. Shima, "The birth, evolution and future of the microprocessor," in *The Fifth International Conference on Computer and Information Technology (CIT'05)*, pp. 2–2, 2005.

[3] P. J. Denning and S. C. Schwartz, "Properties of the working-set model," *Commun. ACM*, vol. 15, p. 191–198, mar 1972.

[4] K. Kaplan and R. Winder, "Cache-based computer systems," *Computer*, vol. 6, no. 3, pp. 30–36, 1973.

[5] N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L.-Y. Chen, B. Zhang, and P. Deaville, "In-memory computing: Advances and prospects," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3, pp. 43–55, 2019.

[6] J. Backus, "Can programming be liberated from the von neumann style? a functional style and its algebra of programs," *Commun. ACM*, vol. 21, p. 613–641, aug 1978.

[7] E. Akgun and M. Demir, "Modeling course achievements of elementary education teacher candidates with artificial neural networks," *International Journal of Assessment Tools in Education*, vol. 5, 01 2018.

[8] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, "Binary neural networks: A survey," *Pattern Recognition*, vol. 105, p. 107281, Sep 2020.

[9] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *NIPS*, 2016.

[10] S. Hashemi, N. Anthony, H. Tann, R. I. Bahar, and S. Reda, "Understanding the impact of precision quantization on the accuracy and energy of neural networks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, (Lausanne, Switzerland), pp. 1474–1479, IEEE, Mar. 2017.

[11] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," *arXiv:1603.05279 [cs]*, Aug. 2016. arXiv: 1603.05279.

[12] H. Chen, Y. Wang, C. Xu, B. Shi, C. Xu, Q. Tian, and C. Xu, "AdderNet: Do We Really Need Multiplications in Deep Learning?," *arXiv:1912.13200 [cs]*, July 2021. arXiv: 1912.13200.

[13] A. Sebastian, T. Tuma, N. Papandreou, M. Le Gallo, L. Kull, T. Parnell, and E. Eleftheriou, "Temporal correlation detection using computational phase-change memory," *Nature Communications*, vol. 8, p. 1115, Dec. 2017.

[14] S. Lee, S.-h. Kang, J. Lee, H. Kim, E. Lee, S. Seo, H. Yoon, S. Lee, K. Lim, H. Shin, J. Kim, O. Seongil, A. Iyer, D. Wang, K. Sohn, and N. S. Kim, "Hardware architecture and software stack for pim based on commercial dram technology : Industrial product," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pp. 43–56, 2021.

[15] A. Agrawal, A. Jaiswal, C. Lee, and K. Roy, "X-SRAM: Enabling In-Memory Boolean Computations in CMOS Static Random Access Memories," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, pp. 4219–4232, Dec. 2018.

[16] A. Jaiswal, A. Agrawal, M. F. Ali, S. Sharmin, and K. Roy, "i-sram: Interleaved wordlines for vector boolean operations using srams," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 4651–4659, 2020.

[17] A. Agrawal, A. Kosta, S. Kodge, D. E. Kim, and K. Roy, "CASH-RAM: Enabling In-Memory Computations for Edge Inference Using Charge Accumulation and Sharing in Standard 8T-SRAM Arrays," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, pp. 295–305, Sept. 2020.

[18] R. Khaddam-Aljameh, P.-A. Francese, L. Benini, and E. Eleftheriou, "An SRAM-Based Multibit In-Memory Matrix-Vector Multiplier With a Precision That Scales Linearly in Area, Time, and Power," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, pp. 372–385, Feb. 2021.

[19] X. Si, Y.-N. Tu, W.-H. Huang, J.-W. Su, P.-J. Lu, J.-H. Wang, T.-W. Liu, S.-Y. Wu, R. Liu, Y.-C. Chou, Z. Zhang, S.-H. Sie, W.-C. Wei, Y.-C. Lo, T.-H. Wen, T.-H. Hsu, Y.-K. Chen, W. Shih, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, N.-C. Lien, W.-C. Shih, Y. He, Q. Li, and M.-F. Chang, "15.5 A 28nm 64Kb 6T SRAM Computing-in-Memory Macro with 8b MAC Operation for AI Edge Chips," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, (San Francisco, CA, USA), pp. 246–248, IEEE, Feb. 2020.

[20] J. Zhang, Z. Wang, and N. Verma, "In-Memory Computation of a Machine-Learning Classifier in a Standard 6T SRAM Array," *IEEE Journal of Solid-State Circuits*, vol. 52, pp. 915–924, Apr. 2017.

[21] X. Si, J.-J. Chen, Y.-N. Tu, W.-H. Huang, J.-H. Wang, Y.-C. Chiu, W.-C. Wei, S.-Y. Wu, X. Sun, R. Liu, S. Yu, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, Q. Li, and M.-F. Chang, "24.5 A Twin-8T SRAM Computation-In-Memory Macro for Multiple-Bit CNN-Based Machine Learning," in *2019 IEEE International Solid-State Circuits Conference - (ISSCC)*, (San Francisco, CA, USA), pp. 396–398, IEEE, Feb. 2019.

[22] J. Kim, J. Koo, T. Kim, Y. Kim, H. Kim, S. Yoo, and J.-J. Kim, "Area-Efficient and Variation-Tolerant In-Memory BNN Computing using 6T SRAM Array," in *2019 Symposium on VLSI Circuits*, (Kyoto, Japan), pp. C118–C119, IEEE, June 2019.

[23] Y. Wang, J. Chen, Y. Pu, and Y. Ha, "Energy-Efficient Arbitrary Precision Multi-Bit Multiplication with Bi-Serial In/Near Memory Computing," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, (Seville, Spain), pp. 1–5, IEEE, Oct. 2020.

[24] J. Chen, W. Zhao, Y. Wang, and Y. Ha, "Analysis and Optimization Strategies Toward Reliable and High-Speed 6T Compute SRAM," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, pp. 1520–1531, Apr. 2021.

[25] Y. Zhang, L. Xu, K. Yang, Q. Dong, S. Jeloka, D. Blaauw, and D. Sylvester, "Recryptor: A reconfigurable in-memory cryptographic Cortex-M0 processor for IoT," in *2017 Symposium on VLSI Circuits*, (Kyoto, Japan), pp. C264–C265, IEEE, June 2017.

[26] Q. Dong, S. Jeloka, M. Saligane, Y. Kim, M. Kawaminami, A. Harada, S. Miyoshi, M. Yasuda, D. Blaauw, and D. Sylvester, "A 4 + 2T SRAM for Searching and In-Memory Computing With 0.3-V $V_{\mathrm {DDmin}}$," *IEEE Journal of Solid-State Circuits*, vol. 53, pp. 1006–1015, Apr. 2018.

[27] M. Kutila, A. Paasio, and T. Lehtonen, "Comparison of 130 nm technology 6T and 8T SRAM cell designs for Near-Threshold operation," in *2014 IEEE 57th International Midwest Symposium on Circuits and Systems (MWSCAS)*, (College Station, TX, USA), pp. 925–928, IEEE, Aug. 2014.

[28] J. Jeong and J. Park, "Fast 6T SRAM Bit-Line Computing with Consecutive

Short Pulse Word-Lines and Skewed Inverter," in *2020 International SoC Design Conference (ISOCC)*, (Yeosu, Korea (South)), pp. 292–293, IEEE, Oct. 2020.

[29] S. Cheon and J. Park, "A Bit-Line Boosting Technique for Fast Bit-Line Computation without Read Disturbance," in *2020 International SoC Design Conference (ISOCC)*, (Yeosu, Korea (South)), pp. 294–295, IEEE, Oct. 2020.

[30] K. Lee, J. Jeong, S. Cheon, W. Choi, and J. Park, "Bit Parallel 6T SRAM In-memory Computing with Reconfigurable Bit-Precision," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, (San Francisco, CA, USA), pp. 1–6, IEEE, July 2020.

[31] T. Ajayi, D. Blaauw, T. Chan, C. Cheng, V. Chhabria, D. Choo, M. Coltella, S. Dobre, R. Dreslinski, M. Fogaça, *et al.*, "Openroad: Toward a self-driving, open-source digital layout implementation tool chain," *Proc. GOMACTECH*, pp. 1105–1110, 2019.

[32] T. Ajayi, V. A. Chhabria, M. Fogaça, S. Hashemi, A. Hosny, A. B. Kahng, M. Kim, J. Lee, U. Mallappa, M. Neseem, *et al.*, "Toward an open-source digital flow: First learnings from the openroad project," in *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1–4, 2019.

[33] Google and Skywater, "Skywater sky130 pdk documentation." https://skywater-pdk.readthedocs.io/en/main/index.html.

[34] M. R. Guthaus, J. E. Stine, S. Ataei, B. Chen, B. Wu, and M. Sarwar, "OpenRAM: an open-source memory compiler," in *Proceedings of the 35th International Conference on Computer-Aided Design*, (Austin Texas), pp. 1–6, ACM, Nov. 2016.

[35] S. Taware, "Design of 1024x32 sram (32kbits) using openram and sky130 pdks." https://github.com/ShonTaware/SRAM_SKY130.git, 2021.

[36] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison Wesley, 2011.

[37] S. A. Mondal, S. Talapatra, and H. Rahaman, "Analysis, modeling and optimization of transmission gate delay," in *2011 3rd Asia Symposium on Quality Electronic Design (ASQED)*, pp. 246–253, 2011.

[38] B. Amrutur and M. Horowitz, "A replica technique for wordline and sense control in low-power SRAM's," *IEEE Journal of Solid-State Circuits*, vol. 33,

pp. 1208–1219, Aug. 1998.

[39] J. Ousterhout, G. Hamachi, R. Mayo, W. Scott, and G. Taylor, "Magic: A vlsi layout system," in *21st Design Automation Conference Proceedings*, pp. 152–159, 1984.

[40] A. Suciu, P. Cobarzan, and K. Marton, "The never ending problem of counting bits efficiently," in *2011 RoEduNet International Conference 10th Edition: Networking in Education and Research*, pp. 1–4, 2011.

[41] A. Dalalah, S. Baba, and A. Tubaishat, "New hardware architecture for bit-counting," in *Proceedings of the 5th WSEAS International Conference on Applied Computer Science*, ACOS'06, (Stevens Point, Wisconsin, USA), p. 118–128, World Scientific and Engineering Academy and Society (WSEAS), 2006.

[42] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Transactions on Electronic Computers*, vol. EC-13, no. 1, pp. 14–17, 1964.

[43] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza*, vol. 34, pp. 349–356, 1965.

[44] W. J. Townsend, J. Swartzlander, Earl E., and J. A. Abraham, "A comparison of Dadda and Wallace multiplier delays," in *Advanced Signal Processing Algorithms, Architectures, and Implementations XIII* (F. T. Luk, ed.), vol. 5205 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 552–560, Dec. 2003.

[45] M. Shalan and T. Edwards, "Building openlane: A 130nm openroad-based tapeout- proven flow : Invited paper," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–6, 2020.

Acknowledgments

Questo elaborato è la sublimazione di quello che, sono convinto, è stato e forse rimarrà il periodo più intenso della mia vita. Così pregno di emozioni che non basterebbero altrettante pagine di questa tesi per descriverlo come meriterebbe, ma un tentativo va fatto.

Prima di concedermi a plateali sentimentalismi desidero mantenere, ancora per qualche riga, le sembianze di un rigore accademico per ringraziare il Prof. Mario Roberto Casu. Per l'attenzione dimostratami, la dedizione e la passione per l'elettronica e l'insegnamento; tutte provate non solo durante il percorso di tesi ma anche nelle interessanti lezioni. La ringrazio per avermi dato l'opportunità di dare il meglio di me, fine ultimo e sacro della nobile arte dell'insegnante.

Riprendendo le poche (e imbarazzate, ndr) parole pronunciate al Fante ormai più di due anni fa: "...se il valore di un uomo si misura dalle persone di cui si circonda, non potrei essere più felice di trovarmi qui in mezzo a tutti voi." -autocit. Ancora fermamente convinto di ciò, dedico l'imminente fiume di parole a tutti coloro che, in un modo o nell'altro, sono parte integrante della mia vita.

Ai miei genitori, per gli innumerevoli sacrifici aspettando questo giorno, nella speranza di ripagare ogni lacrima versata con la promessa di dedicare la vita a rendervi orgogliosi dell'uomo che sto diventando.
A mia madre, per tutto ciò che sono e sarò. Per non aver mai mollato, per avermi insegnato l'importanza delle emozioni e del prendersi cura del prossimo.
A mio padre, per avermi instillato l'amore per la scienza fin da bambino. Alle sue ripetizioni di fisica nella stanza d'albero la notte prima del test d'ingresso, senza le quali non so se avrei mai attraversato il colonnato del Politecnico.
A mia sorella Sara, alla dolcezza di un abbraccio nel momento del bisogno e di un consiglio quando la rotta sembra essere persa. Per la certezza di un porto dalle acque tranquille nella burrasca della vita.

E se la vita è una traversata non posso non menzionare quel branco di pirati di Titanium: Alessandro D.M., Alex C., Andrea M., Andrea P., Angelo R., Antonio D.B., Gianmarco F., Giorgio L., Lorenzo C., Marco A., Matteo D.G., Matteo S., Mauro F., Roberto S. e mio cugino Francesco. Vi ringrazio per ogni singola volta in cui una risata sul nostro gruppo ha risollevato giornate fin troppo uggiose. Per i simposi alcolici fino alle quattro di notte da Materia, per le albe ed i tramonti in riva al mare, per le infinite confidenze, i consigli e quelle due parole giuste dette al momento giusto.
Non posso non ripetermi nel menzionare ancora una volta mio cugino Francesco. Alle passioni comuni ed alla fratellanza che ci ha sempre legati.

Qualsiasi persona di mia conoscenza arrivata fin qui a leggere questo smielato torrente di parole sa quanto io possa (molto ndr.) spesso essere scostante, irritante ed insopportabile. Se al mondo ci sono persone la cui visione di noi stessi è offuscata dall'amore incondizionato nei nostri confronti o da amicizie secolari, ce ne sono altre che invece ci vedono benissimo e decidono di utilizzare l'altro braccio della bilancia riempiendolo con ciò che di buono abbiamo da offrire.
Il primo tra queste persone che voglio e devo ringraziare è Michele, costante della mia quotidianità da quasi cinque anni. Grazie per aver reso casa quattro mura di una città grigia a mille chilometri da Brindisi.
Questo percorso accademico rimarrà costellato di gioie immense, di successi e di sconfitte, di spensieratezza e sacrifici. Un tale sosteneva che la felicità è vera solo quand'è condivisa, io mi permetto di espandere il concetto a tutte le emozioni provate tra le mura del Politecnico e per le strade di Torino insieme a coloro che chiamare colleghi sarebbe riduttivo. Grazie a Davide E. e Luigi G., per le notti insonni passate davanti ad un PC, ad interpretare ondine verdi e a saltare da una mattonella all'altra. Grazie a Davide D.S., Thomas G. e Luca F., per le mille risate, per le scorribande in laboratorio, per le carbonare saltate per aria e la pizza kebab prima degli esami. Grazie anche ad Alessia D., per le sedute di psicanalisi con uno spritz in mano e ad Emanuele D.L., compagno di cinismo.

Termino questo dovuto piagnisteo dedicando la fine di questo percorso a tutti i

familiari, gli amici, i conoscenti che, pur non figurando in queste pagine, sono stati presenti in questi anni. Con un sorriso, davanti ad un calice di vino o semplicemente con una pacca sulla spalla. Grazie per ogni singolo gesto.

All'importanza delle relazioni umane nel mondo post-pandemia.