Energy-Efficient Smart Embedded Memory Design for IoT and AI

by

Avishek Biswas

B. Tech. (Hons.), Indian Institute of Technology, Kharagpur (2012)
S.M., Massachusetts Institute of Technology (2014)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

Author .....
Department of Electrical Engineering and Computer Science
May 23, 2018

Certified by .....
Anantha P. Chandrakasan
Vannevar Bush Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Energy-Efficient Smart Embedded Memory Design for IoT and AI

by

Avishek Biswas

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2018, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Static Random Access Memory (SRAM) continues to be the embedded memory of choice for modern System-on-a-Chip (SoC) applications, thanks to aggressive CMOS scaling, which keeps on providing higher storage density per unit silicon area. As memory sizes continue to grow, increased bit-cell variation limits the supply voltage (V_{dd}) scaling of the memory. Furthermore, larger memories lead to data transfer over longer distances on chip, which leads to increased power dissipation. In the era of the Internet-of-Things (IoT) and Artificial Intelligence (AI), memory bandwidth and power consumption are often the main bottlenecks for SoC solutions. Therefore, in addition to V_{dd} scaling, this thesis also explores leveraging data properties and application-specific features to design more tailored and "smarter" memories.

First, a 128Kb 6T bit-cell based SRAM is designed in a modern 28nm FDSOI process. Dynamic forward body-biasing (DFBB) is used to improve the write operation, and reduce the minimum V_{dd} to 0.34V, even with 6T bit-cells. A new layout technique is proposed for the array, to reduce the energy overhead of DFBB and decrease the unwanted bit-line switching for un-selected columns in the SRAM, providing dynamic energy savings. The 6T SRAM also uses data prediction in its read path, to provide upto 36% further dynamic energy savings, with correct predictions.

The second part of this thesis, explores in-memory computation for reducing data movement and increasing memory bandwidth, in data-intensive machine learning applications. A 16Kb SRAM with embedded dot-product computation capability, is designed for binary-weight neural networks. Highly parallel analog processing inside the memory array, provided better energy-efficiency than conventional digital implementations. With our variation-tolerant architecture and support of multi-bit resolutions for inputs/outputs, $> 98\%$ classification accuracy was demonstrated on the MNIST dataset, for the handwritten digit recognition application.

In the last part of the thesis, variation-tolerant read-sensing architectures are explored for future non-volatile resistive memories, e.g. STT-RAM.

Thesis Supervisor: Anantha P. Chandrakasan
Title: Vannevar Bush Professor of Electrical Engineering and Computer Science

Acknowledgments

It has been a long, and sometimes uncertain, journey since I started in fall 2012 at MIT to finally writing this thesis in 2018. I am really grateful to my advisor Prof. Anantha P. Chandrakasan, who guided and motivated me throughtout this journey. Anantha gave me the golden opportunity to come to MIT (which has been my dream forever) and work with him. He has always believed in me, motivated me to think differently, push the limits and be a better researcher overall. I started off working on DC-DC converters in his group for my masters project. And now, I have worked on implementing machine learning algorithms inside memories. So it has been a great learning experience for me. Apart from technical feedback, I have also learned how to communicate my research to others more effectively and many other soft skills from him. In addition, Anantha has provided me with many opportunities outside MIT to build collaborations and work with people from the industry. He has been a great role model and a guide throughout these past 6 years at MIT and I am forever thankful to him.

I would also like to thank my thesis committee members, Prof. Vivienne Sze and Prof. Harry Lee, for their valuable technical feedback. Vivienne introduced me to the area of binary neural networks, which led to a very interesting and rewarding project of implementing computation in memory. I really enjoyed our technical discussions at the beginning of this project. She provided great insights, which helped improve the project. I also enjoyed discussing ADC's with Prof. Lee during that project.

Next, I would like to thank Intel for funding my research and also hosting me at their Hillsboro campus for 2 consecutive summers for internships. In particular, I would like to thank Fatih Hamzaoglu and Umut Arslan for all their feedback and help with test-chip fabrication and measurements for the STT-RAM sense amplifier project. I would also like to thank ST Microelectronics, in particular Andreia Cathelin, for their support and generously donating silicon area in their modern 28nm FDSOI process. Next, I would like to thank Edith Beigne for inviting me to spend a few months at CEA-Leti in France, to work on their in-memory computing project.

It was a great experience meeting so many wonderful people there, who generously included me as a part of their team.

Next, I would like to mention the wonderful Ananthagroup for being such helpful and supporting colleagues to work alongside. We have a lot of technical expertise in the group, so there is always someone I could go for help whenever I was stuck with some technical doubts. I got a lot of help on cadence issues from Arun, Phil and Nachiket. Mehul and Gilad's wonderful job on setting up the group wiki, also helped me navigate many technical issues. Apart from that, I really enjoyed the long discussions in the procrastination circles on various non-technical topics and all the outings to "free-food" events with fellow lab-mates: Priyanka, Bonnie, Chiraag, Frank, Chu, Preet, Sirma, Utsav, Mohamed, Alex, Miaorong, Saurav, Aya, Rabia, Ujwal and others. I would also like to thank Margaret, who is always so helpful in setting up meetings with Anantha, taking care of reimbursements etc., all with a smile. I would also mention Janet Fischer in the EECS graduate office and others at MTL for their help and support.

At MIT, far away from home, I had the pleasure of meeting some wonderful people, who have made this journey way more enojyable and fun, with all their friendship and warmth. I would like to thank Arijit, Suchismita, Abhishek, Siong-Thye, Yashovardhan, Naga, Devendra, Pritish, Ahmad, Nate and others for supporting and appreciating me in various ways and for being wonderful friends. I will cherish the time we have spent and the memories forever.

Last and most importantly, I am so grateful to my parents, Satyabrata Biswas and Chandrima Biswas, my brother, Avirup, and all my family for their eternal love, support and belief in me. They have made a lot of sacrifices for me and always stood by me through thick and thin. They are the ones who drive me to dream and achieve bigger goals and move forward in life. This thesis is dedicated to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Continuous CMOS scaling for the past few decades, following Moore's law [1], has increased transistor density on a silicon chip by more than 6 orders of magnitude. Today's micro-processors pack billions of transistors within a few mm^2 of silicon area, providing tremendous compute power, high processing performance and supporting a wide range of functionalities. This has enabled a plethora of applications, including portable electronics (e.g. laptops, smartphones), wearable health-monitoring systems, wireless sensor nodes etc. More and more of these devices around us have compute capabilities and are increasingly getting connected to each other, forming the "Internet-of-Things" (IoT). The tremendous boom in compute power, coupled with improvement in algorithms and availability of huge amounts of data, has also enabled modern computing systems to perform many "Artificial Intelligence" (AI) tasks e.g. image recognition [2, 3], speech recognition [4], natural language understanding [5] etc. One particularly interesting case for these recognition systems is that of "always-ON" sensing, which allows them to continuously collect data and monitor their surroundings. If the "always-ON" block (e.g. face detection) detects an event of importance (e.g. a human face), for further processing of data it can activate more complex systems (e.g. a face recognition system, which can be typically in "sleep" mode to conserve energy). However, many of these increasing number of "smart" and "always-ON" systems have limited energy budget, since they typically get powered by batteries or harvest ambient energy [6]. Hence, energy-efficiency of these computing

systems is a key concern, for their sustainable deployment in the real world.

1.1 Embedded Memories in Modern Computing Systems

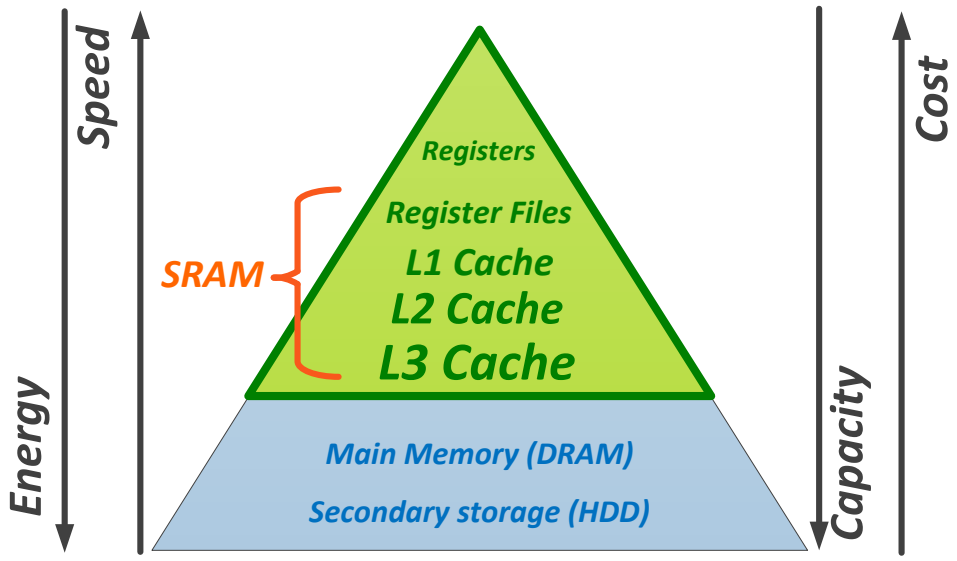


Figure 1-1: Typical memory hierarchy [7].

Memories are a critical component of any computing system, allowing to store data that need to be accessed for processing. Fig. 1-1 shows the typical memory hieachy, with increasing speed and decreased energy of access as we move up the pyramid. On the other hand, manufacturing cost decreases, along with increasing density of stored bits, as we move down the hierarchy. Hence, frequently accessed data are stored in faster and more energy-efficient embedded memories (caches), situated near the processing engines. Whereas, data that is not accessed very frequently, is stored further away in larger but slower memories, e.g. DRAM, Flash etc.. In this thesis, we mostly focus on embedded memories in the upper part of the hierarchy, especially Static Random Access Memories (SRAM), which are widely used for on-chip caches in modern micro-processors [8, 9, 10]. SRAMs occupy a significant portion of chip area in modern Systems-on-a-Chip (Soc). As seen Fig. 1-2, for the 14nm Zen processor [8]

almost 50% portion of the die is occupied by L2 and L3 caches, which are implemented with SRAMs. Figure 1-3 shows the general trend of increasing cache size in modern micro-processors, which can be as high as 54MB on a single die [10].

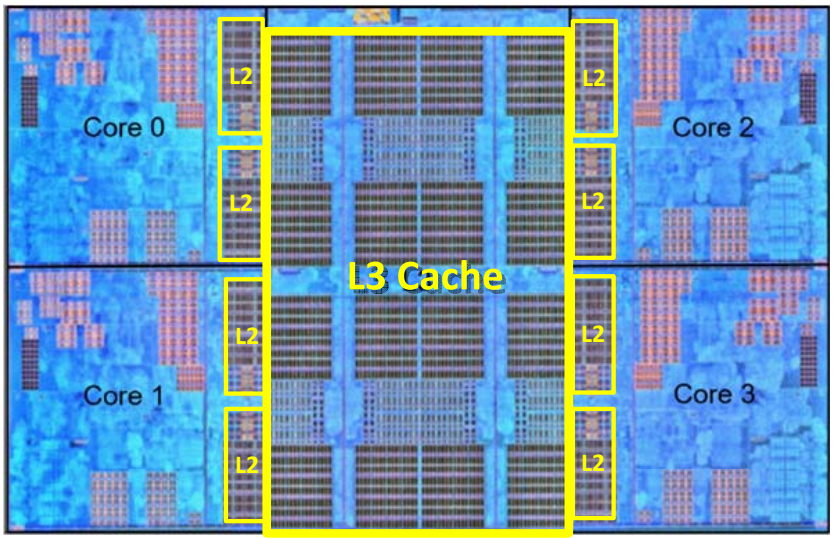


Figure 1-2: Area occupied by on-chip caches (L2, L3) for the 14nm Zen processor [8].

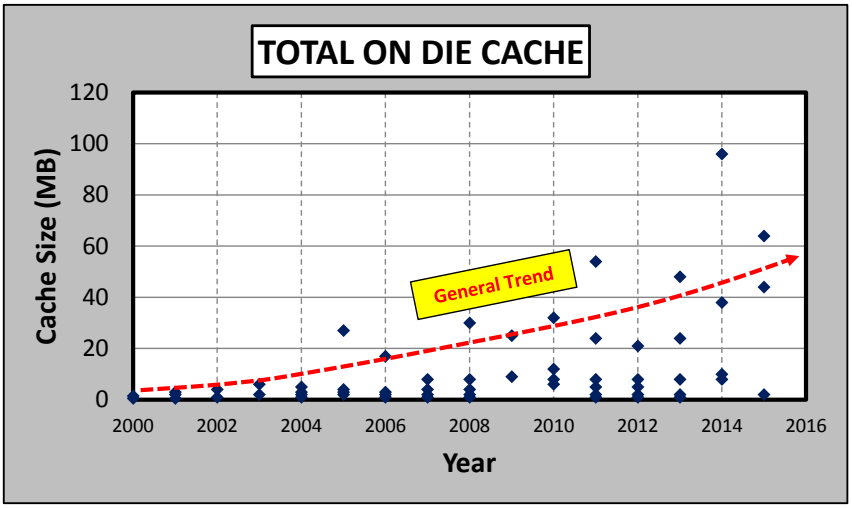


Figure 1-3: General trend of cache size. [source: ISSCC 2015 Trends]

In emerging applications, such as accelerators for machine learning (ML) algorithms like neural networks, embedded memory is even more crucial for improving performance. Google's Tensor Processing Unit or TPU chip [11] is one such example,

in which 37% of the chip area is occupied by memory (∼28MB), to buffer different types of data.

1.2 Memory-wall & the "von-Neumann bottleneck"

"Memory-wall" [12] is a term used to describe the increasing gap between CPU clock speeds and the memory access times. While CMOS scaling has resulted in smaller and faster transistors, which improve the CPU's speed, the overall processing times remained limited by the slow access times of memories. To continue Moore's law, multi-core processor designs started gaining popularity around 2005. However, with multiple cores operating in parallel, on-chip memory bandwidth and energy became more dominant issues. One of the major reasons for memory being a bottle-neck in modern computing systems is the traditional "von-Neumann" architecture, in which the memory and the processor are physically separate, with data flowing between them. This leads to a limited bandwidth for data transfer, which is dictated by the memory input/output (IO) capacity. Hence, in modern computing systems, a major portion of the time and energy is spent in moving data back and forth between the memory and the processing elements [13, 14]. Fig. 1-4 shows the relative energy consumption for different computation and memory access operations in a 45nm CMOS process [13]. As seen from the figure, the energy required to read a 32-b data from a 32KB SRAM cache can be 100× more than the energy needed for a 32-b integer add operation. This trend is more pronounced in data-intensive applications, which necessitates bigger memories. This is because, bigger memories lead to movement of data over longer distances on-chip, which amounts to higher power dissipation.

With the gaining popularity of data-intensive AI and ML applications like face recognition, image classification, speech recognition etc., the amount of storage needed is increasing tremendously. This can be easily seen from the size of the models (Table 1.1) [15] of typical Deep Neural Networks (DNN), which are one of the most popular ML algorithms. Hardware implementations of DNNs face major challenges in dealing with huge amount of data movement. Fig. 1-5 shows the different components of

Operation	Energy (pJ)
Integer ADD (8b)	0.03
Integer ADD (16b)	0.05
Integer ADD (32b)	0.1
Integer MULT (8b)	0.2
Integer MULT (32b)	3.1
8KB SRAM Read (32b)	5
32KB SRAM Read (32b)	10
1MB SRAM Read (32b)	50

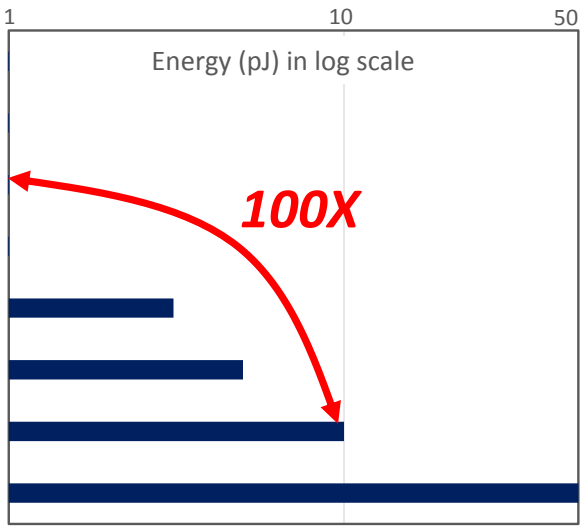


Figure 1-4: Energy consumption of various operations in a 45nm CMOS process at $V_{dd} = 0.9$V, recreated from [13].

energy for running DNNs in hardware, as estimated in [15]. As seen from the figure, the different memory access and data movement energies are the dominant ones, compared to the computation (ALU) energy.

Table 1.1: Summary of model sizes of popular DNNs [15]

Metric	**LeNet-5**	**AlexNet**	**Overfeat fast**	**VGG-16**	**GoogLeNet v1**	**ResNet-50**
Total # Filter Weights	60K	61M	146M	138M	7M	25.5M

There are different approaches to tackling the memory bottleneck problem in modern computing systems. Dynamic Voltage Scaling (DVS) has been proven to be an effective way to reduce energy consumption of circuits [16, 17]. Decreasing the supply voltage (V_{dd}) provides savings in the dynamic energy consumption ($\propto V_{dd}^2$), as well as a reduction in the leakage power consumption, at the expense of slower performance. However, SRAM V_{dd} do not easily scale as much as in logic circuits, due to the reduction in operating margins. The next section will discuss the challenges and opportunities in low-voltage SRAM design. Data-dependent SRAMs, which take advantage of certain data properties to provide energy savings, would be also discussed.

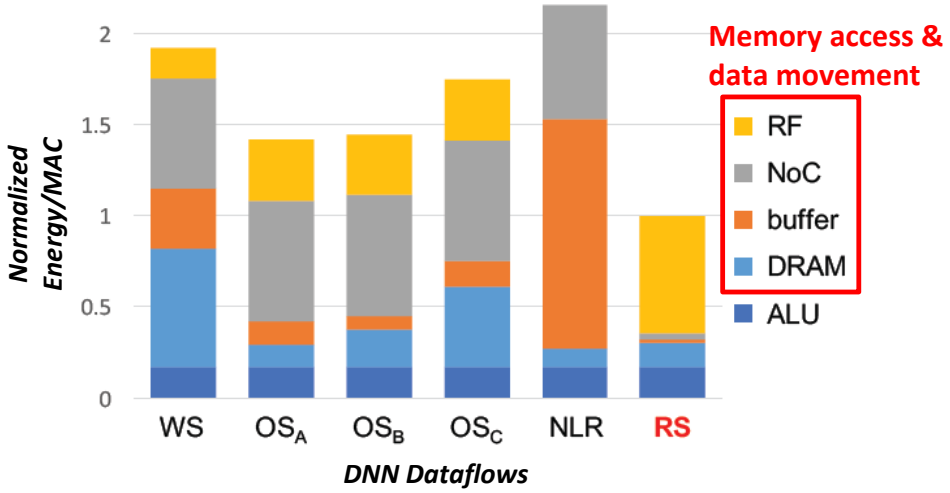


Figure 1-5: Breakdown of energy consumption for different digital implementations of a DNN algorithm [15].

Another promising approach to overcoming the von-Neumann bottleneck, is the concept of in-memory compute, which blurs the line between traditional memories and compute elements, by bringing computation features inside the memory. This will be discussed in the subsequent section.

1.3 Low-voltage SRAM

Static Random Access Memories (SRAM) are the most popular type of embedded memories and one of the most critical building blocks in modern SoCs. SRAMs are pre-dominantly used for register files and L1-L3 cache memories, in the embedded memory hierarchy (Figure 1-1). This is primarily because, SRAMs offer the best access-speed performance among other embedded memory technologies [7]. Furthermore, SRAMs are fully compatible with modern CMOS processes and operating voltage, and hence, can be easily integrated with logic circuits. With the scaling of device dimensions to sub-65nm regime, the variation in transistor threshold voltage (V_t) has become more severe. Since SRAM bit-cell size aggressively reduces with every technology node, the effect of random V_t variation makes it extremely challenging to reduce the V_{dd} of SRAMs, while maintaining sufficient stability margin for the

bit-cell. Figure 1-6 shows the recent trend in SRAM bit-cell size and the operating V_{dd}. As seen from the figure, the V_{dd} scaling has been essentially stagnant, in sub-65nm CMOS processes. Decreasing the supply voltage (V_{dd}) provides savings in the dynamic energy consumption ($\propto V_{dd}^2$), as well as a reduction in the leakage power consumption, at the expense of slower performance [16, 17]. Hence, V_{dd} scaling is very crucial to reduce energy consumption of SRAMs.

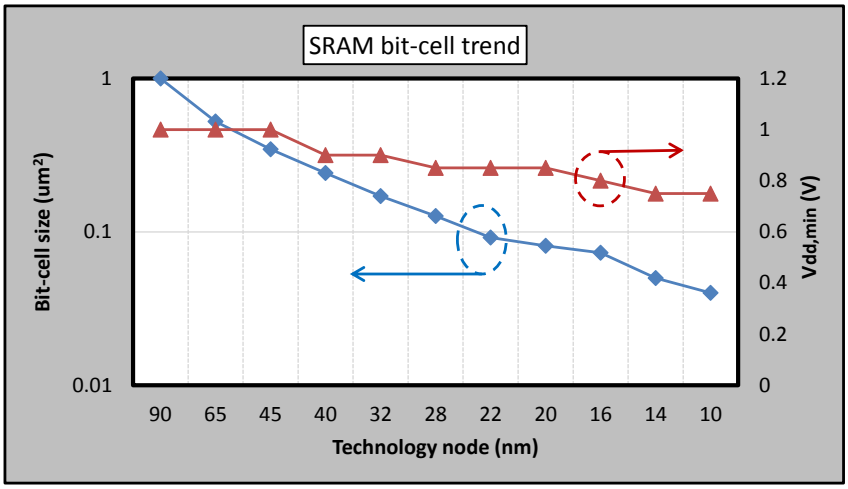


Figure 1-6: Scaling trends for SRAM bit-cell size and operating V_{dd}. [source: ISSCC 2016 Trends]

1.3.1 6T bit-cell based SRAM

Six transistor (6T) based bit-cell has been the workhorse for SRAM design, owing to the small cell area and compact lithography-friendly layout [18], resulting in high density memory arrays. As shown in Figure 1-7, each row of bit-cells share a common word-line (WL), while a pair of bit-lines (BL and BLB) are shared by multiple bit-cells in a column. The number of bit-line pairs (n) and the bit-width (m) of a single word, determine the column select ratio of $\frac{n}{m}$. A conventional 6T SRAM bit-cell (shown in Figure 1-7) consist of two back-to-back inverters (comprised of PU1, PD1 and PU2, PD2) and two access transistors (PG1 and PG2). The inverter pair is cross-coupled such that the output of one goes to the input of the other, and vice-versa. The resulting positive feedback of the inverter pair, can hold the desired data (states

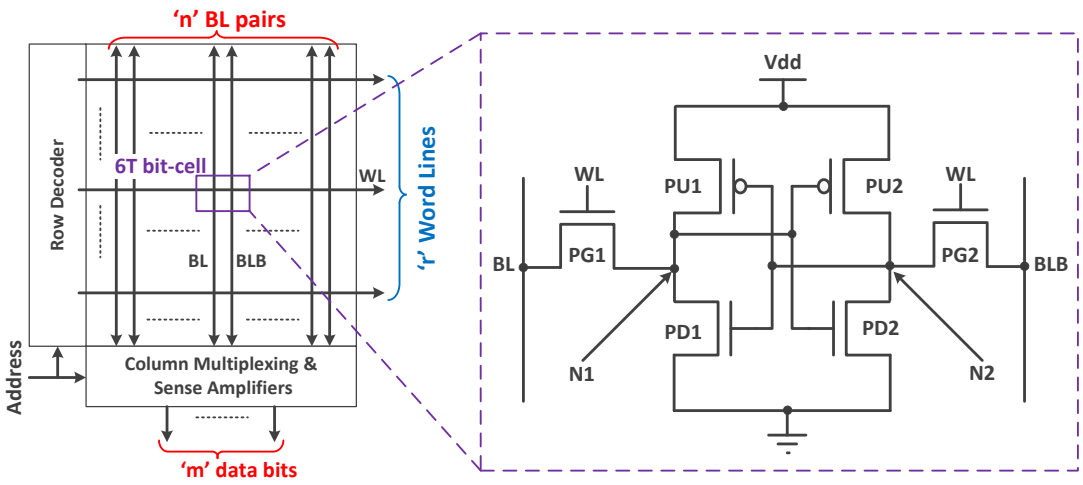


Figure 1-7: Conventional SRAM array architecture and 6T bit-cell.

"1" or "0") indefinitely at the internal nodes (N1 and N2), as long as the SRAM is powered up and the access transistors are turned-off. Access transistors are only turned-on during read and write operations, to connect the internal data nodes to the bit-lines (BL and BLB).

Fig. 1-8(a) shows the 6T bit-cell during a write operation. It starts with driving the bit-lines to the data value to be written. The WL is then asserted, turning-on the PG transistors. If the data to be written is opposite to the previously stored state (as shown in Figure 1-8(a)), the potential of the high internal node is lowered, depending on the drive strengths of the pull-up (PU) PMOS and the PG NMOS transistors. The ratio of the drive strengths of the PG and PU transistors is known as the γ-ratio and it is an important parameter in 6T SRAM design. The transistors need to be sized carefully so that the γ-ratio is high enough to lower the potential of the high internal node below the V_{TRIP} of the connected inverter. As shown in Figure 1-8(b), a low γ-ratio can lead to a write failure.

The read operation starts with pre-charging the bit-line pairs (BL and BLB) to a known voltage (typically V_{dd}). The bit-lines are then kept floating and the word-line (WL) is asserted. Depending on the data stored, one of the bit-lines (BL or BLB) starts discharging through the pass-gate (PG) and pull-down (PD) NMOS transistors, connected in series (Figure 1-9(a)). The bit-line differntial voltage is sensed by a sense-

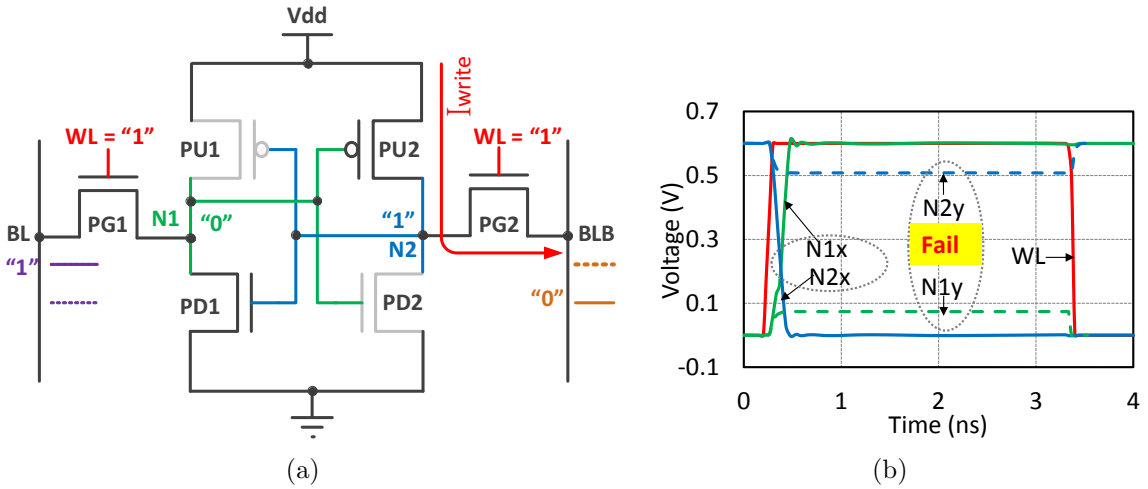


Figure 1-8: (a) 6T bit-cell during a write operation, (b) Waveforms during a write operation for two different γ-ratios: $(W_{PG}/W_{PU})_x = 1.25, (W_{PG}/W_{PU})_y = 1$). Write failure occurs when the γ-ratio is not high enough to lower the potential of N2 below the V_{TRIP} of the PU1-PD1 inverter.

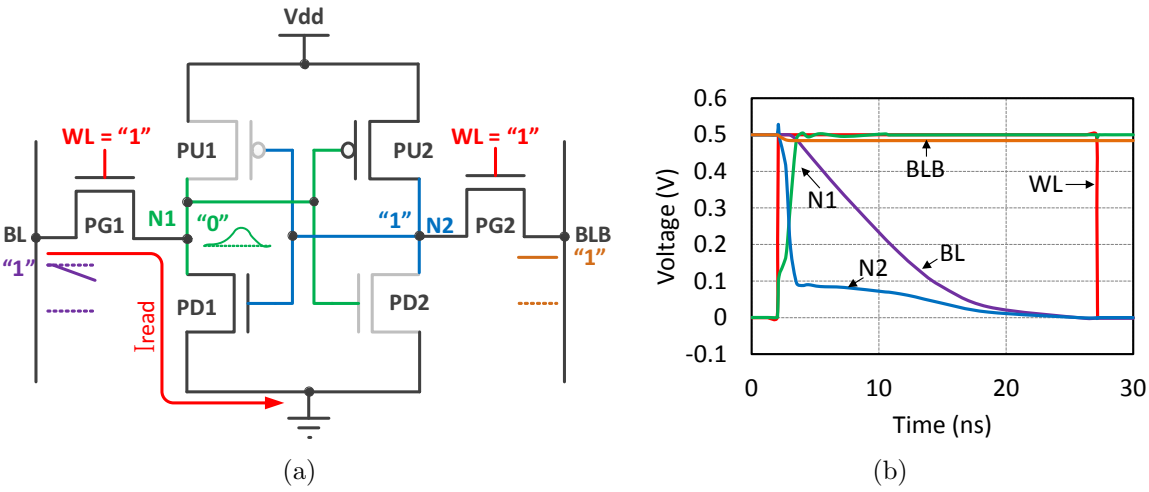


Figure 1-9: (a) 6T bit-cell during a read operation, (b) Waveforms showing a "read disturb" for a minimum sized bit-cell. Bit-cell flips since the disturbance at N1 is large enough to trip the inverter (PU2, PD2).

amplifier to output the data. During the read operation, the discharging current flows from the bit-line to the cell ground, on the side of the bit-cell storing a "0". This leads to an increase in the potential of the corresponding internal node (N1 in Figure 1-9(a)) and the amount of disturbance depends on the drive strengths of the PG and PD NMOS transistors. If this increased voltage goes above the trip-point (V_{TRIP}) of the connected inverter, the stored data is flipped. This event is known as '*read disturb*'

and it is shown in Figure 1-9(b). In order to prevent this, the PD NMOS needs to be stronger than the PG NMOS. The ratio of their drive strengths is known as the β-ratio, which is an important SRAM design parameter. Careful sizing of the NMOS transistors is required to achieve the desired β-ratio, which ensures successful read operations (i.e. without any 'read disturbs').

With CMOS scaling continuing in 10's nm regime, it becomes increasingly challenging to precisely control the transistor dimensions and also the channel doping concentration. Hence, the effect of random dopant fluctuation is exacerbated in modern CMOS processes, leading to a wide variation in the transistor threshold voltages (V_t). The effect of random V_t variation is more pronounced as V_{dd} scales down, since the overdrive voltage ($V_{dd} - V_t$) of the transistors is reduced. Thus, even though the required β and γ ratios are satisfied at higher V_{dd}'s, they might not be sufficient for every bit-cell when V_{dd} approaches close to V_t. This causes SRAM functional failure, limiting the minimum achievable V_{dd}.

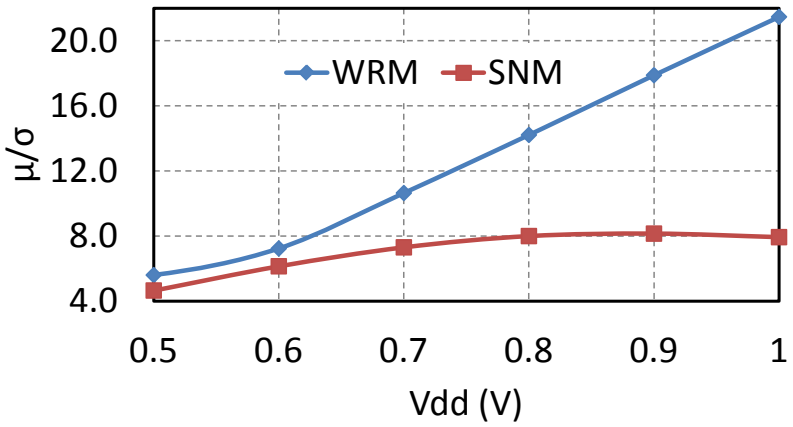


Figure 1-10: SNM and WRM dependence on V_{dd} for a 6T bit-cell in a 28nm CMOS process.

Figure 1-10 shows the effect of reducing V_{dd} on static noise margin (SNM) and write margin (WRM) of a 6T bit-cell that has been designed in a 28nm CMOS process, using regular-V_t transistors. As seen from the figure, the μ/σ ratio for both SNM and WRM decreases with V_{dd}. However, WRM exhibits a much stronger dependence on V_{dd} than SNM, especially at higher voltages. A μ/σ of more than 5 (i.e. < 1

error in 10^6 bits) is typically required, for high yield ratios in large sized SRAMs. Hence, new and improved read and write assist techniques are being increasingly used as design solutions, to reliably reduce the minimum operating voltage ($V_{dd,min}$) of SRAMs. Additionally, newer transistor structures, such as FDSOI [19, 20] and FinFET [21, 22], are also emerging as replacements for planar-bulk devices, to reduce device variations and further improve SRAM $V_{dd,min}$.

1.3.2 Assist Techniques Used in Modern 6T SRAMs

The issues of functional margin degradation with V_{dd} scaling are generally addressed by using peripheral assist circuits, to aid the read and write operations [23, 24]. [23] defines three modes of SRAM functional failure: read-ability, write-ability and read-stability. Assuming a differential sense-amplifier (SA) based read operation for 6T bit-cells, a read-ability failure occurs if the BL differential voltage (when the SA is triggered) is less than the offset voltage of the SA. A write-ability failure occurs when the desired data cannot be written at the end of the WL pulse. Read-stability failures can happen if the selected bit-cell (BC) data or the half-selected BC data is accidently flipped during a read or write operation, respectively. Here we summarize common assist techniques used to improve read-ability, write-ability and read-stability in modern 6T bit-cell based SRAMs.

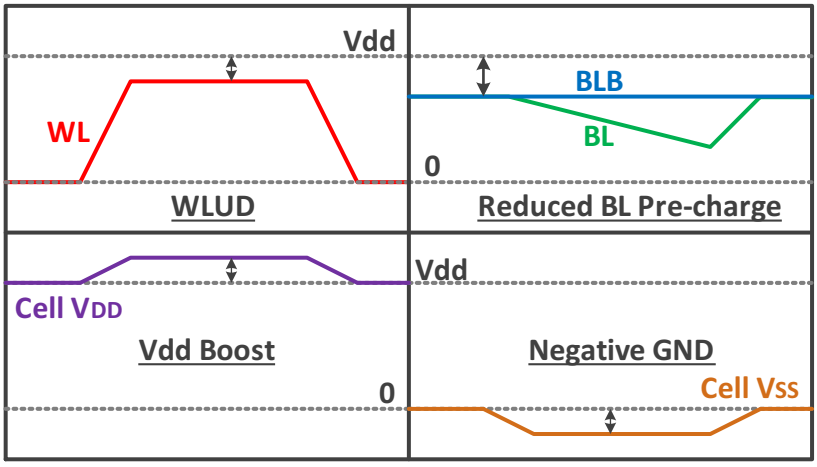


Figure 1-11: Conventional read assist techniques.

Fig. 1-11 shows the waveforms for the different read assist techniques used in previous works. Most of these techniques aim at improving the β ratio by making the PG NMOS weaker or the PD NMOS stronger. For the word-line underdrive (WLUD) technique [25, 26, 27], the gate-bias of the PG transistor is reduced, making it weaker than the PD transitor and improving read-stability. However, a reduced gate-drive decreases the BL discharge current, making read-ability worse. A reduced BL pre-charge (PCH) level can help in improving read-stability, without degrading the drive strength of the PG device (assuming negligible effect of V_{DS}). The work in [28] demonstrated a yield increase from 5 to 5.7 sigma, by pre-charging BLs to approximately 70% of V_{dd}. This technique has the overhead of generating and regulating the reduced PCH voltage. Precise regulation of the PCH voltage is necessary, since a low PCH level can create a pseudo-write operation scenario, which can overwrite the existing cell data. Boosting the cell V_{DD} makes the PD NMOS stronger than the PG device, improving read-stability. Driving the cell V_{SS} to a negative voltage level, simultaneously improves the strengths of the PG and PD devices. Hence the BL discharge current is increased, improving read-ability. These techniques can be implemented in a row-by-row [29] or column-by-column [30, 31] fashion.

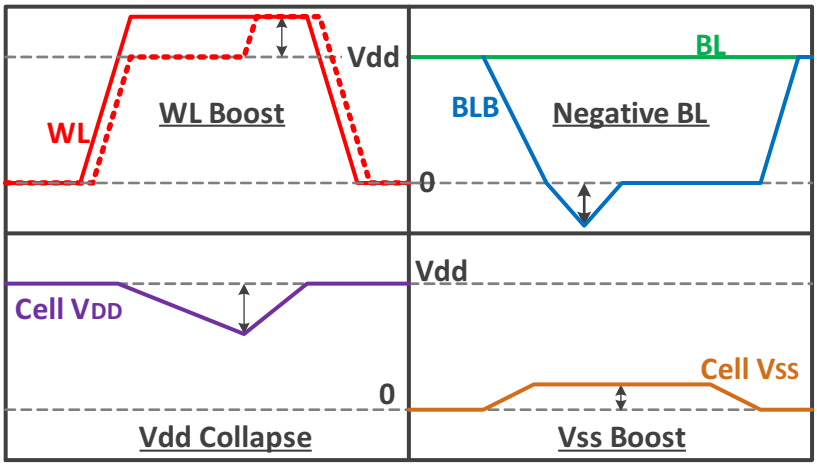


Figure 1-12: Conventional write assist techniques.

Figure 1-12 shows the waveforms for the conventional write assist techniques, which attempt to improve the γ ratio or weaken the bistability of the cross-coupled inverters. The word-line boosting technique improves write-ability by increasing the

gate-drive of the PG device, making it stronger than the PU PMOS. However, this has a detrimental effect on the read-stability of the half-selected bit-cells (i.e. bit-cells in the selected row and unselected columns) in a column-interleaved array. [32] addresses this issue by delaying the boosting phase with respect to WL turn-on. Hence, the half-selected bit-cells have already started reading and the BL voltage has sufficiently reduced when the WL boosting is applied. This technique incurs the area overhead of generating the boosted WL voltage. The negative BL technique [28, 27, 22, 33] increases the gate-drive (V_{GS}) of the PG transitor by reducing its source voltage and hence, improves write-ability. However, by decreasing the potential of one of the BLs below GND, there is a non-zero V_{GS} across the PG devices in unaccessed rows. If the internal node of an unaccessed bit-cell on this side is "1", then there is a chance of unintentional over-writing of that cell data. The non-zero V_{GS} also results in increased leakage from the PG devices and causes partial loss of the boost signal. This technique is also susceptible to voltage overstress in the write path at higher V_{dd} values [28, 25]. 'Cell-V_{DD} collapse' [26, 33] and 'Cell-V_{SS} boost' [34] techniques decrease the strength of the cross-coupled inverter pair holding the data and hence improves writability. However, the effect is much weaker [23] than the 'WL boost' and 'Negative BL' techniques, since the PG transistor's strength is not improved. Furthermore, these techniques when implemented column-wise, run the risk of violating the data retention voltage for unaccessed bit-cells, which can cause accidental loss of cell data. Whereas, if they are implemented row-wise, the read-stability of the half-selected bit-cells are degraded.

1.3.3 Alternate bit-cells for Low-V_{dd} Operation

Although assist techniques are useful in improving the operation margin of 6T SRAMs, they are fundamentally limited by the fact that the read and write design choices in 6T SRAMs are inherently contradictory. Hence, alternate bit-cell topologies [35, 36] have been explored to decouple read and write operations. Fig. 1-13 shows the popular 8-transistor (8T) based bit-cell design, which adds an extra read port, consisting of series NMOS transistors, that does not affect the internal data nodes ('Q' & 'QB') of

the bit-cell. Thus, there is no read-disturbance in the bit-cell and the read operation is essentially not limited by the bit-cell sizing. Therefore, a low voltage operation can be achieved by using more agressively designed write-assist techniques.

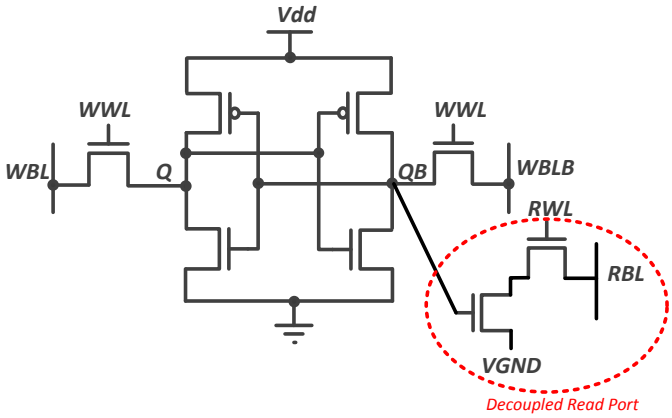


Figure 1-13: 8T SRAM bit-cell for low voltage operation.

1.4 Data-dependent SRAMs

The conventional 6T SRAM energy does not depend on the data read, because of the differential bit-cell structure with a shared word-line (WL). However, in many applications, the data bits stored are not entirely random and can have certain interesting properties which can be leveraged to get energy savings. For example, in neural networks, the input activations for the different layers can be sparse and have a high percentage of '0's [15]. Hence, 8T SRAMs can be used to store them. As shown in Fig. 1-13, the 8T bit-cell has a single-ended read port and hence, it only discharges the bit-line (BL) for one polarity of the data (can be chosen to be '1'). Therefore, if most of the data bits stored are '0's, then the BL switching energy (which is generally the most dominant component in the SRAM dynamic energy) can be substantially reduced [37]. Another application where data-dependent memory could be useful is motion estimation for video processing [38]. In motion estimation, when reading data from a reference video frame (i.e. an image), there can be a lot of correlation in the neighboring pixel values. Hence, to store the reference frame, [38] uses a 10T bit-

cell based SRAM with prediction signals, to prevent BL switching if the prediction matches the bit stored/read. The prediction signals, shared column-wise for each output data bit in [38], are updated less frequently because of the temporal correlation of the read pixel data. With this architecture, [38] achieved upto 1.75× reduction in energy/access with 100% correct predictions. [39] proposed another data-dependent 10T SRAM, in which the prediction logic is implemented at the periphery, rather than the bit-cell itself. It uses the last data read as the prediction signal for the next data. Therefore, if the last cycle's prediction was correct, there is no need to pre-charge the BLs for the next cycle, which reduces BL switching energy. If the next cycle detects a wrong prediction, it updates the prediction value at the peripheral read circuit and pre-charges the corresponding BL. Hence, if the activity factor in the read data is low (i.e. similar bits are read for many consecutive cycles), this prediction based SRAM [39] can provide upto 67% energy savings. [40] uses a similar concept of saving BL energy by not dynamically pre-charging the BLs every cycle. It rather uses a 10T bit-cell with a CMOS sensing inverter embedded in it, for a static read-out operation. Hence, if the activity factor in the read data is low, [40] can reduce BL switching to achieve 28-39% energy reduction, compared to 8T SRAMs with majority logic [37]. All the prior work mentioned above, use 8T/10T bit-cells, which have 1.3-1.7× more area than 6T bit-cells. In applications where SRAM density is crucial (e.g. GPUs), it would be useful to have data-dependent energy savings using standard 6T bit-cells.

Table 1.2: Summary of data-dependent SRAMs

Work	**Data-dependent Concept**	**Energy Savings**
Fujiwara '08 [37]	An 8T SRAM with majority data logic and bit reordering to reduce BL switching	28% (vs. no majority logic)
Noguchi '07 [40]	A non-precharge 10T SRAM with reduced BL switching for similar consecutive data reads	39% (vs. 8T with majority logic)
Sinangil '14 [38]	A 10T SRAM with prediction signals in bit-cells to reduce BL switching for correct predictions	43% (vs. 8T)
Duan '17 [39]	A 10T SRAM using previous data read as prediction to selectively pre-charge BLs	67% (vs. 8T)

1.5 In/Near-memory Computing Architectures

Although V_{dd} scaling can provide energy savings in memories, modern computing systems are inherently limited by the "von-Neumann" bottleneck, as explained before. Even with multiple cores, the system throughput and energy is limited by the memory size and I/O bandwidth. One of the promising approaches to circumvent this limitation is the concept of "in/near-memory" computation (Fig. 1-14). If the required computation can be done inside the memory array or in the periphery, very close to the array, then there would be significantly less data that needs to be transferrred over long distances on chip. Additionally, there is the potential to simultaneously access multiple memory addresses, which are involved in the computation. Thus the effective number of bits that can be accessed from the memory is not limited by its IO, which thereby, improves system throughput.

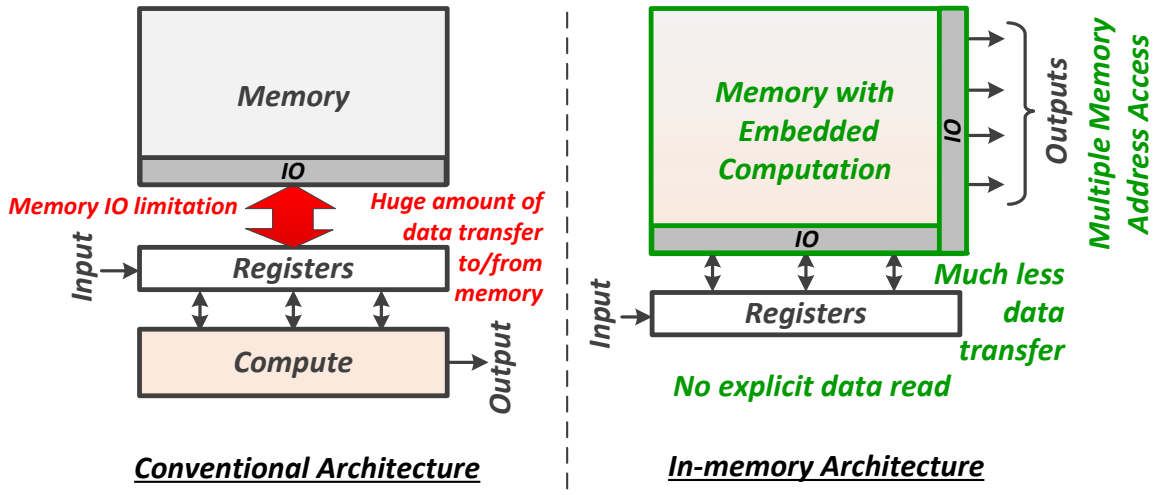


Figure 1-14: Comparison of conventional and in-memory architectures.

There have been a few recent works exploring in/near-memory computing architectures. They can be primarily divided into 2 categories, depending on whether the data stored in the memory is read explicitly and sequentially ("near-memory") or whether it is done implicitly ("in-memory"). Each of these can be further sub-divided into 2 categories, depending on whether the computation involved is done in the digital manner (bit-precise) or in the analog domain. There are many applications, e.g.

machine learning algorithms, in which the computation involved does not need to be bit-precise, as it would have been done with an ideal digital implementation. In these cases, low-precision analog computation with low signal-to-noise (SNR) ratio has been used to take advantage in the error resiliency of the algorithms involved. [41] describes a ML classifier (support vector machine or SVM) implemented with a standard 6T SRAM array. 5× improvement in speed and 10× improvement in energy consumption was reported, compared to a fully-digital implementation, for the handwritten digit recognition task (MNIST). [42] presents a multi-functional ML classifier chip, which implements face detection using SVMs, event detection using matched filters, face recognition using template matching and handwritten digit recognition using k-nearest neighbor algorithm. Although these systems achieve better energy-efficiency, they do not provide very accurate computations and rely on error tolerance of the ML algorithms. In other applications, bit-error rates might be more crucial e.g. video encoding/decoding, security algorithms etc. In these cases, the in/near-memory compute system needs to provide bit-accurate results in the digital domain. [43] presents an SRAM with AND, OR, XOR computation capacibility for running security algorithms e.g. AES. [44] describes a similar system for an image processing application, to gain 2× in speed. [45] presents a 6T SRAM architecture for in-memory logic operations, which can be reconfigured as a content-addressable-memory (CAM) for searching operations.

Table 1.3: Summary of recent works on in/near-memory computing

Work	**Comp. Mode**	**Brief Description**
Zhang '17 [41]	Analog	A 6T SRAM with in-memory dot-product computations using 5-b inputs and 1-b weights, producing 1-b outputs
Kang '18 [42]	Analog	A 6T SRAM with in/near-memory dot-product computations using 6-b inputs and 8-b weights, producing 6-b outputs
Zhang '18 [43]	Digital	A 10T SRAM to support in/near-memory bit-wise boolean operations for running security algorithms
Akyel '16 [44]	Digital	A 10T SRAM implementing in-memory bit-wise boolean operations and near-memory digital arithmetic operations
Jeloka '16 [45]	Digital	A 6T SRAM with in-memory logic operations, which can be also re-configured as a CAM for searching operations

1.6 Thesis Contributions

This thesis primarily focuses on low-power SRAM designs, utilizing data properties and in-memory computation capability, to address traditional memory bottlenecks and achieve high energy-efficiency, in energy-constrained IoT and AI applications. Furthermore, variation-tolerant circuits for promising furture memory technologies, e.g. STT-RAM, are also explored.

1.6.1 Low-Power 6T SRAM with Data-Dependent Energy Savings

Chapter 2 presents a low-voltage 6T bit-cell based SRAM architecture which can incorporate data-prediction in its read path, to provide dynamic energy savings. Using 6T bit-cells provides higher memory density than 8T/10T designs. This is very critical in many data-intensive applications, which needs large on-chip memories, dominating majority of the chip area. However, traditional 6T SRAMs do not operate at low supply voltages (V_{dd}) due to conflicting read and write design requirements. This work, in a 28nm FDSOI process, uses a dynamic forward body-biasing (DFBB) technique to increase the write margin of the SRAM and simultaneously improve the access speed for both write and read operations. A new array layout is proposed, which rotates the traditional 6T bit-cell by 90°, to align the n-wells horizontally (along the word access direction). This layout technique reduces the n-well capacitance switched per cycle for DFBB. In addition, it also eliminates the half-select problem (common in traditional 6T SRAMs), by supporting multiple word-lines per row. Solving the half-select issue, also helps in eliminating the dynamic bit-line power consumption of un-selected columns. In conventional 6T SRAMs, the un-selected columns switch every cycle unnecessarily, when their corresponding bit-cells are accessed, since they share a common word-line with the accessed bit-cells in a row. Further, savings in dynamic energy is obtained by incorporating data-prediction in the read path. We use a hierarchical array design, with local and global bit-lines (BL), and implement the prediction logic in the local read/write circuitry, rather than the bit-cell itself [38].

This enables using 6T bit-cells in the array to achieve higher memory density, while still saving global BL dynamic energy, when data is predicted correctly. This read architecture using data prediction can be also interpreted as enabling an in-memory X(N)OR computation capability, which could be beneficial in certain applications e.g. security algorithms [46] and low-precision neural networks [47, 48].

1.6.2 In-Memory Computation for Low-Power Neural Networks

Chapter 3 presents an energy-efficient SRAM architecture with embedded dot-product computation capability, intended for low-power neural-network (NN) based ML applications. The basic computation for neural networks can be boiled down to a multiply-and-accumulate (MAC) or dot-product operation. For every layer in the NN, the 3-D inputs are processed by multiple 3-D filters in parallel, producing a 3-D output, which is fed as the input to the next layer. The dot-product is inherently a parallel operation, and coupled with the fact that multiple dot-products share the same inputs/weights, there is a lot of scope for highly parallel implementations. However, in traditional systems, memory bandwidth would limit the degree of parallelism and also be a bottleneck in terms of energy. This is due to the need of constant shuffling of data, back and forth between the memory and the processing elements. With our architecture, the weights (1-b in this work) are stored in local arrays of an SRAM and are not explicitly read. Instead, the inputs (6-b) for a given NN layer, are sent to this SRAM as analog pre-charge voltages on bit-lines using DACs. The bit-cells multiply these analog voltages by the 1-b weights stored in them and then, the bit-line voltages are averaged. Finally, the analog averaged voltages are transferred back into the digital domain using ADCs. There are multiple advantages of this architecture. Firstly, the weights are not read explicitly and sequentially from each row of the memory. Instead, 16 rows are implicitly read in parallel, while also implementing multiplication by 1-b. This reduces the SRAM read energy and also improves throughput. Secondly, the highly parallel analog averaging operation is more

energy-efficient than digital accumulation, which needs large bit-width adders to sequentially add multiple partial products for each MAC operation. Finally, the highly parallel nature of the in-place computations significantly reduces data transfer to and from the memory, providing higher energy-efficiency. It should be noted that, the computation accuracy with an in-memory approach can be degraded due to variation in the SRAM bit-cells, which use near-minimum sized transistors for high density. In this work, we overcome this problem in 2 main ways. Firstly, we use the SRAM bit-cells just for evaluating the 1-b weights, using full-swing local BL discharge. The analog input voltages are not controlled by the SRAM bit-cells. Secondly, we use the capacitance of the BLs to implement the analog computations. Since, in a CMOS process, capacitance has much less variation than transistor V_t's, hence, our approach is more immune to variation and achieve better computation accuracy.

1.6.3 Variation-Tolerant Read Sensing Architectures for Non-Volatile Resistive Memories

While SRAM is the current embedded memory of choice in modern SoCs, its scaling would be limited in the future as CMOS processes scale down to sub 10-nm regime, with feature sizes approaching the dimensions of a few atoms. Hence, alternate memory technologies (e.g. ReRAM, STT-RAM, PCRAM etc.) are being actively investigated, as potential candidates for embedded memories. Most of these memories use a resistive device as the storage element, whose resistance can be modulated to have 2 (or possibly more) values. Chapter 4 presents a read sensing architecture targetted to improve yield. This is achieved in 2 main ways. Firstly, pseudo-differential 2-phase resistive divider circuit is used to improve the read signal margin. And secondly, an offset-cancellation technique is proposed which can tolerate more variation from the array. The differential nature of the read techniques, with an inherently single-ended bit-cell, provides more immunity to various types of common-mode noise and improves yield.

Chapter 2

Low-Power 6T SRAM with Output Data Prediction

The aggressive scaling of SRAM bit-cell size with every technology node makes it extremely challenging to reduce the $V_{dd,min}$ of SRAMs, due to the increasing effect of device variations. However, V_{dd} scaling is crucial in reducing the energy consumption of SRAMs, which is a significant portion of the overall energy consumption in modern micro-processors. Energy savings in SRAM is particularly important for battery-operated applications, which run from a very constrained power-budget. Recent works [49] have used negative bit-line (NBL) and word-line (WL) boosting techniques to improve the $V_{dd,min}$ for write operation. However, they are susceptible to voltage overstress issues at higher Vdd, or can cause read disturb issues in the half-selected bit-cells. In this work, we use dynamic forward body-biasing (DFBB) in an energy-efficient way to improve the write margin and increase operating speed. The proposed implementation significantly reduces the energy overhead of DFBB and also helps in reducing the switching energy for half-selected bit-lines. Further savings in energy/access is achieved by incorporating data-prediction in the 6T read path, which reduces bit-line switching. The proposed techniques have been implemented for a 128Kb 6T SRAM macro, designed in a 28nm FDSOI technology.

2.1 Dynamic Forward Body-Biasing (DFBB)

2.1.1 DFBB as a write-assist

Forward body-biasing of an NMOS device, refers to applying a positive body-to-source voltage ($V_{BS} > 0V$) across it. This reduces the threshold voltage of the NMOS, since the body-terminal acts like a second gate [19]. Fully-depleted silicon-on-insulator (FDSOI) technology [19, 20] offers the unique feature of applying FBB on NMOS devices without the need of a triple-well structure, which would be required in a bulk CMOS process. This is possible because of the electrical isolation of the source/drain of the transistor from the well/substrate by using a buried-oxide (BOX) layer. The ultra-thin BOX layer also improves the body-biasing efficiency ($\sim$ 85mV/V [19]) as compared to a bulk CMOS process ($\sim$ 25mV/V [19]), in which the BOX layer is not present. In this work, we use the LVT flavor of the FDSOI transistors [50], which is characterized by NMOS devices on n-well and PMOS devices on p-well, as shown in Figure 2-1. Since the n-well bias (GNDS) is controlled independently, the NMOS devices can be selectively forward body-biased, reducing their threshold voltage. The PMOS devices are already in FBB mode, since their body terminal is at 0V (same as the p-substrate).

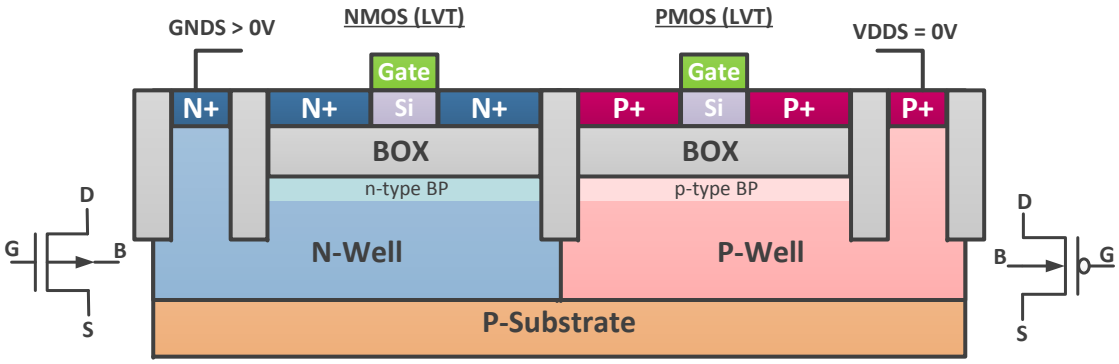


Figure 2-1: Cross-sectional view and circuit symbols of the LVT transistors [50] in the FDSOI process, used for the 6T SRAM design.

Fig. 2-2(a) shows the 6T bit-cell with FBB applied on flip-well FDSOI NMOS devices, during a write operation. FBB decreases the threshold voltage of the NMOS

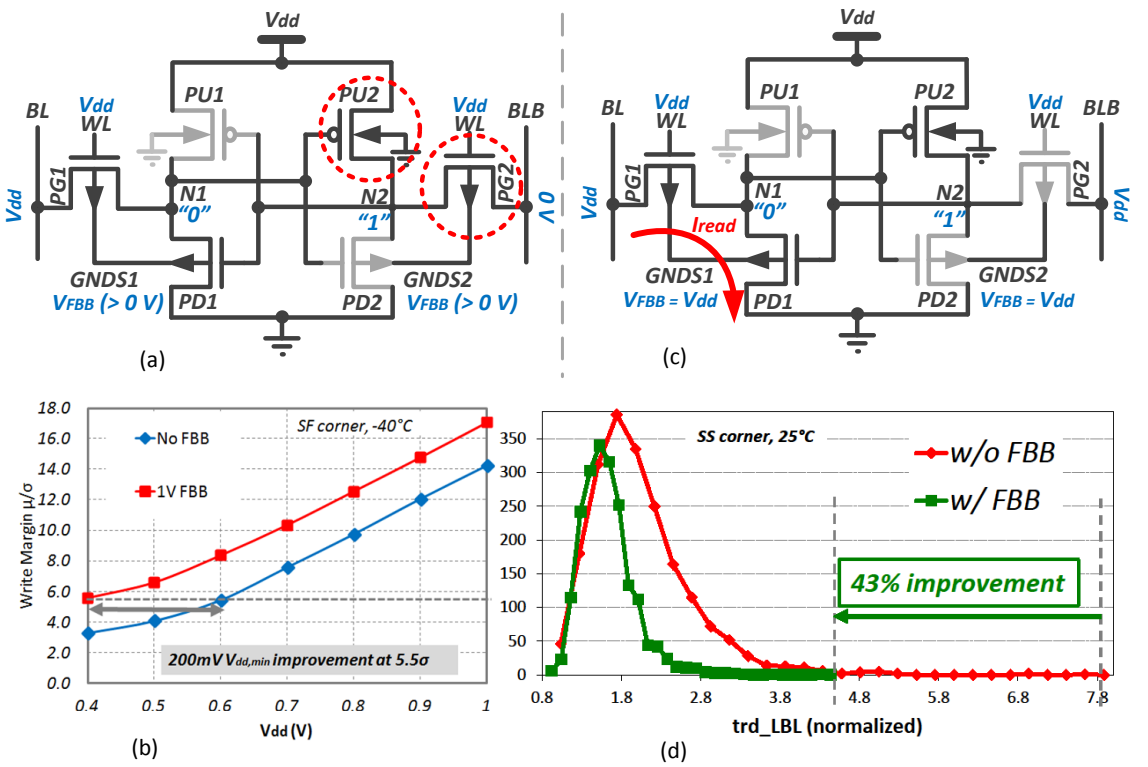


Figure 2-2: (a) 6T SRAM bit-cell with FBB applied during write operation (b) simulated write margin (c) 6T SRAM bit-cell with FBB applied during read operation (d) variation analysis of bit-line read discharge time.

transistors. Hence, the NMOS access transistor (PG2) becomes stronger than the PMOS pull-up transistor (PU2) and therefore, improves write-ability. Furthermore, a stronger NMOS helps to improve the write-speed as well. Fig. 2-2(b) shows the improvement in the write margin as a function of the supply voltage (V_{dd}), with the body-bias voltage (V_{FBB}) kept at a constant value of $1V$. It can be seen from the figure that, the μ/σ value of the write margin is consistently improved in the entire V_{dd} range of 0.4V to 1V. The conventional 6T bit-cell works down to 600mV with a $\mu/\sigma = 5.5$ (SF corner, $-40°C$), without any write assists. As seen from the figure, a 1V forward body-biasing can reduce the $V_{dd,min}$ for the write operation to 400mV, while maintaining a $\mu/\sigma = 5.5$. Fig. 2-2(c) shows the 6T bit-cell with FBB applied during a read operation. FBB improves the read speed by reducing the bit-line discharge time due to faster NMOS transistors (PG1 and PD1). As seen from Fig.1(d), 43% improvement in the simulated local bit-line discharge time can be

achieved by FBB ($V_{dd} = 0.4V, V_{FBB} = 0.4V$).

2.1.2 Energy-Efficient Array Layout for DFBB

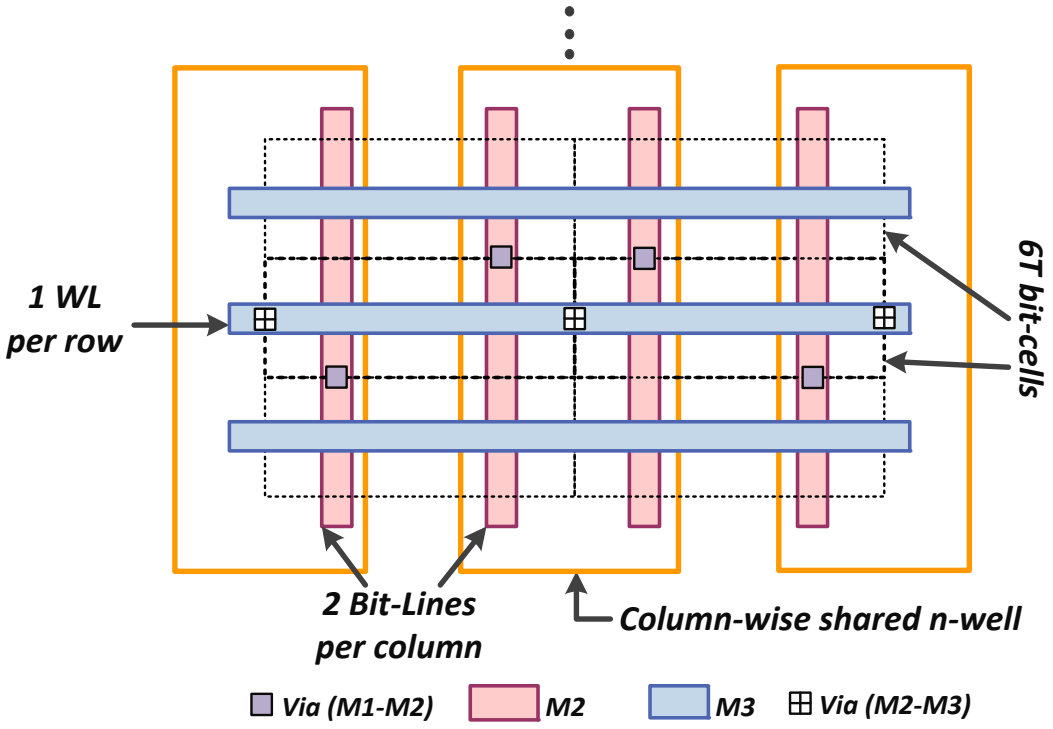


Figure 2-3: Conventional "thin-cell" array layout of 6T bit-cells, with column-wise shared n-wells, shown for 3 rows and 2 columns (not to scale).

Since the leakage of the bit-cell increases when FBB is applied, a huge leakage power penalty would be incurred if a DC FBB is applied to the whole array. This motivates a dynamic implementation of the FBB technique (DFBB). However, DFBB has its own share of power overhead due to n-well switching. For the conventional "thin-cell" 6T layout [51], shown in Fig. 2-3, the n-wells are shared vertically with other bit-cells in a column. Therefore, to apply body-biasing to a selected bit-cell, the entire n-well of the corresponding column needs to be charged up [51]. This translates into a significant capacitive-switching power overhead, since it scales with both the number of rows and columns. This n-well switching power overhead can limit the benefit of V_{dd} scaling, achieved using dynamic body-biasing. To address this issue, an alternate layout technique is proposed in this work, which shares the

n-wells horizontally across all the bit-cells in a row. The benefit of this technique stems from the fact that only one row in the memory array is accessed at a time. Hence, body-biasing can be applied to only the two n-wells in a selected row. This significantly reduces the amount of n-well capacitance switched per cycle, resulting in a more energy-efficient implementation.

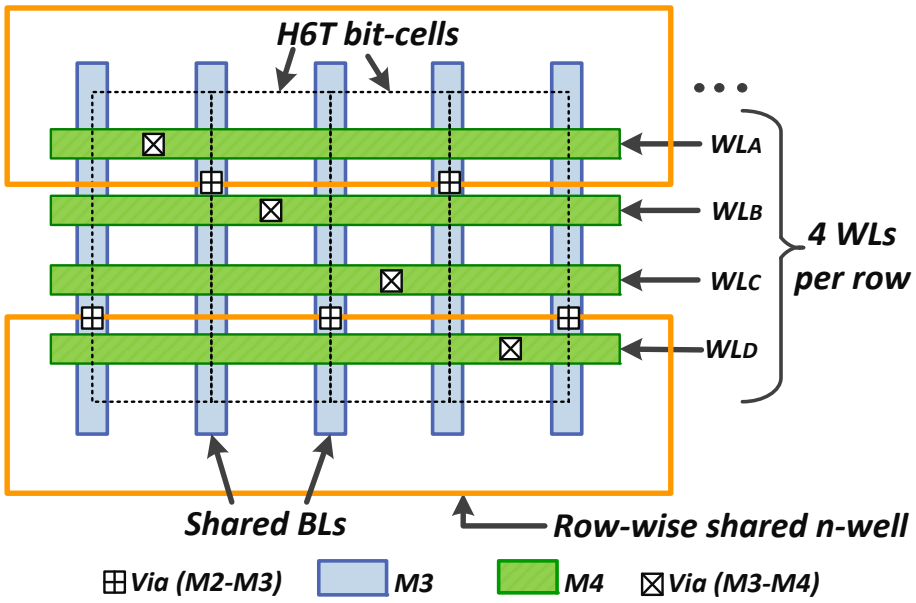


Figure 2-4: Proposed layout of a single row, showing row-wise sharing of n-wells, BL sharing between adjacent columns and multiple WLs per row (not to scale).

Fig. 2-4 shows the proposed layout technique, which shares the n-wells horizontally across all the bit-cells in a row. This is done by rotating the 6T bit-cell by 90°. The conventional "thin-cell" layout is used for the rotated 6T bit-cell (H6T), to minimize the new bit-cell/layout design effort. Traditionally, the bit-line (BL) diffusion contacts are shared with neighboring bit-cells in a column, reducing the effective cell area. However, this is not possible in the present scenario, since the diffusion regions run horizontally. Hence, the BL diffusion contacts are shared between bit-cells in adjacent columns, so that the effective bit-cell area is not increased and the lithography-friendly "thin-cell" layout structure is maintained. The "thin-cell" layout typically has an aspect ratio of approximately 3:1. Since the BLs are now routed in the longer dimension, the parasitic metal routing capacitance per BL is increased by a factor of $\sim 3\times$, compared to the conventional case. With 2 adjacent columns now

sharing a BL, it is necessary to have at least 2 word-lines (WL) for each row. This is to ensure that 2 adjacent bit-cells in a selected row, do not simultaneously drive a single BL. In this implementation, 4 WLs are routed for each row, taking advantage of the longer cell-height. The 4 WLs help in reducing unnecessary BL discharge in half-selected columns, as shown in Fig. 2-5. Therefore, the effective number of BL switching per cycle is reduced by a factor of 4. Due to these layout optimizations, the BL switching power is actually reduced to $\sim 0.75\times$ $(= \frac{3\times}{4})$ i.e. 25% reduction as compared to a conventional implementation, providing dynamic energy savings.

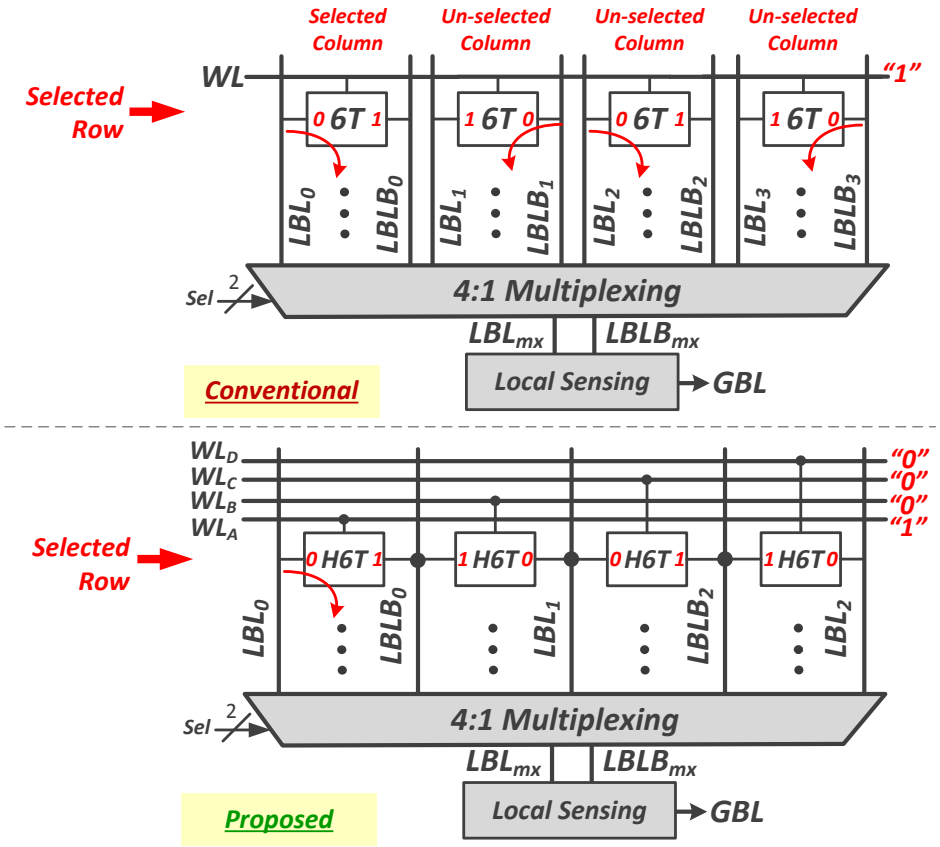


Figure 2-5: Comparison of the conventional (1 WL/row) and proposed (4 WLs/row) architectures for local BL discharge in unselected columns (shown for a group of 4 columns).

The proposed implementation ($H6T$) incurs a 2.5% increase in the effective cell-area, due to non-overlapping WL contacts between adjacent rows (Fig. 2-6). Normal logic design rules are used for bit-cell layout. 4 metal layers are used for routing the different signals. Metal-2 (M2) is used for the cell V_{dd} and GND, metal-3 (M3) is

used for bit-lines and finally, WLs are routed in metal-4 (M4) (Fig. 2-4).

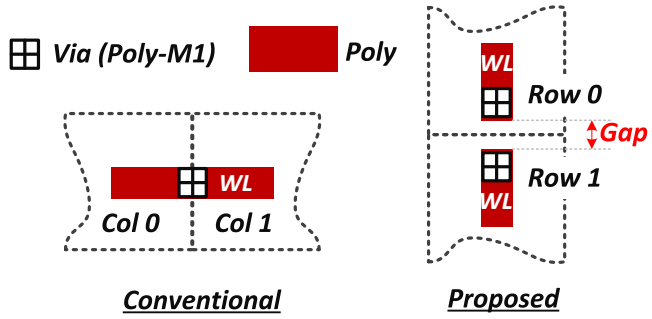


Figure 2-6: Comparison of the WL contact placement in the poly-Si layer for the conventional and proposed bit-cell/array layout.

2.1.3 DFBB circuit implementation

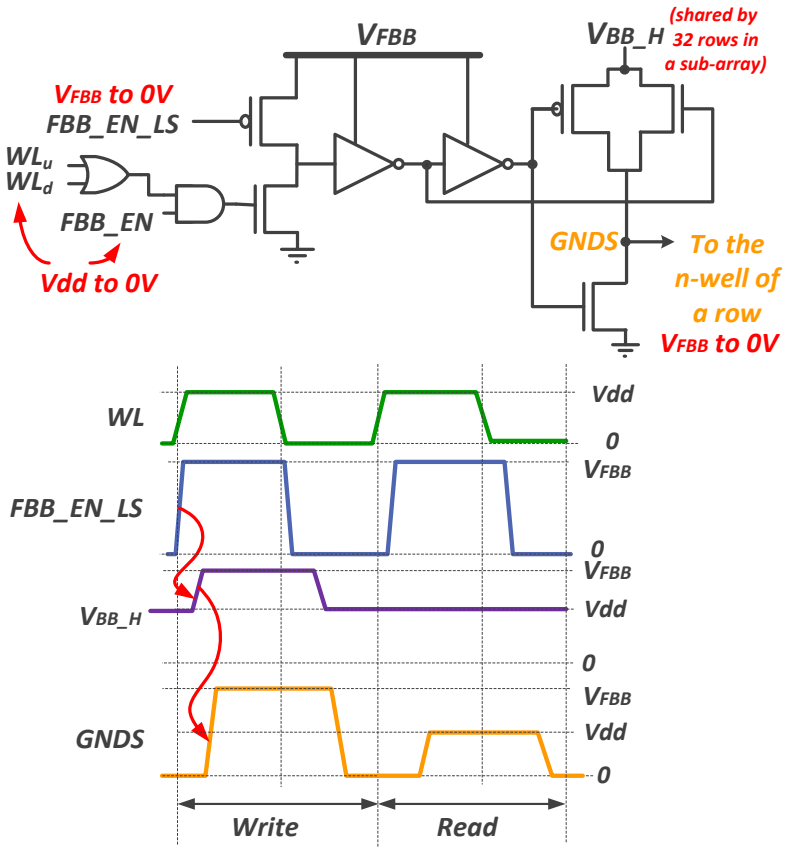


Figure 2-7: Circuit implementation of the proposed row-wise dynamic forward body biasing (DFBB) technique, and sample waveforms with FBB enabled during both write and read modes.

Fig. 2-7 shows the DFBB circuit and corresponding waveforms. Each n-well ($GNDS$) is shared between two consecutive rows in a sub-array. Hence, it is charged up when either of the rows (WL_u/WL_d) is accessed and FBB is enabled (FBB_EN_LS = '1'). The V_{BB_H} node voltage is modulated to either V_{FBB} or V_{dd} for write/read respectively. A higher FBB voltage of $V_{FBB}(> V_{dd})$ is used to improve the write operation, which limits the $V_{dd,min}$ of the array. A lower FBB voltage of V_{dd} is used during a read operation to speed up the local bit-line discharge process. Using a lower FBB voltage reduces the DFBB energy overhead for read operations. The high swing signals (FBB_EN_LS and V_{BB_H}) are shared for all the 32 rows in a sub-array, to amortize the area/power of the level-shifter circuits.

2.2 Overall Architecture

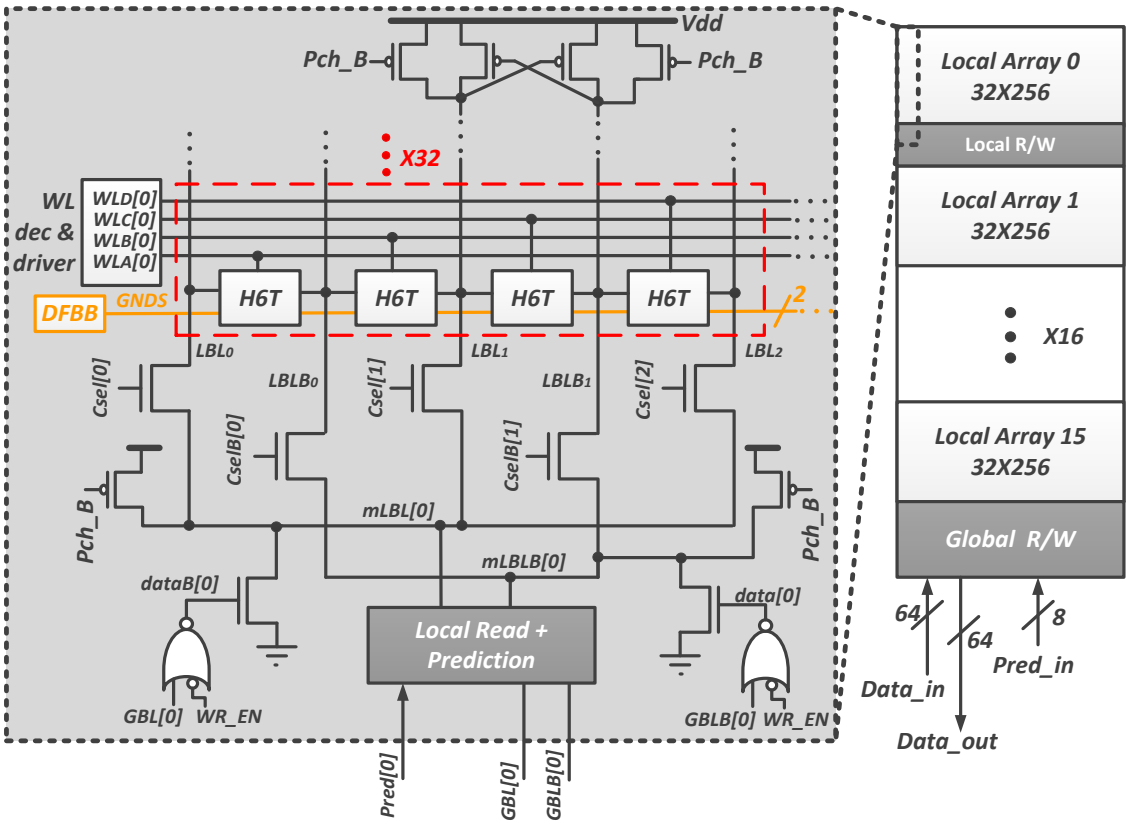


Figure 2-8: Array architecture of the 28nm FDSOI 128Kb SRAM macro.

Fig. 2-8 shows the overall array architecture for the 128Kb SRAM macro, con-

sisting of 16 local sub-arrays. The local sub-array consists of H6T bit-cells arranged in 32 rows by 256 columns. A 4:1 column interleaving ratio is implemented to obtain a 64-bit output data. For a group of 4 columns, there are 3 local BLs and 2 local BLBs. This is because, in this implementation, the local bit-lines are shared between adjacent columns. Therefore, 2 address bits ($A[1:0]$) are required for column multiplexing, to get a local BL pair ($mLBL, mLBLB$). The column-select signals, $Csel[2:0]$ and $CselB[1:0]$ are generated as follows:

$$\begin{gathered} Csel[0] = \overline{A[0]A[1]},\ Csel[1] = A[0] \oplus A[1],\ Csel[2] = A[0]A[1] \\ CselB[0] = \overline{A[1]},\ CselB[1] = A[1] \end{gathered} \tag{2.1}$$

As explained before, each row has 4 word-lines (WL_A, WL_B, WL_C and WL_D), only one of which is asserted based on the row-decoder's outputs and the column-select bits ($A[1:0]$). For an area-efficient implementation of 4 WL drivers per SRAM row, some of the common signals are shared by all the 32 rows in the local array, as shown in Fig. 2-9. A 2:4 decoder, in the local timing circuitry of a 32×256 array, produces the enable signals (ENB_A, ENB_B etc.) to choose one of A, B, C or D WLs. In addition, the PMOS pull-up transistors for the WL drivers are shared by the 32 rows, using the common pull-up lines (Vh_A, Vh_B etc.). In this way, the 4 WL drivers can be fit in 1 row pitch of the array layout, without incurring huge area penalty. It should be noted that, sharing the Vh signals increase their capacitances and hence, the corresponding switching energy. However, only one of them is switched every cycle for a 32×256 local array. Hence, that energy overhead is not significant. The selection of a 4:1 column interleaving ratio and the ability to route 4 WLs per row, eliminates the half-select issue in this implementation. The $DFBB$ circuit (explained before in Fig. 2-7) implements the row-wise dynamic body-biasing. The n-wells are shared horizontally, across all the 256 bit-cells in a row.

The local R/W circuitry per muxed-column, consist of inverter-based large-signal sensing for read and the prediction logic (explained later). In addition, the local write is implemented by the pull-down NMOS devices, controlled by *data* and *dataB*. During a write operation (WR_EN = "1"), *data* and *dataB* are locally generated

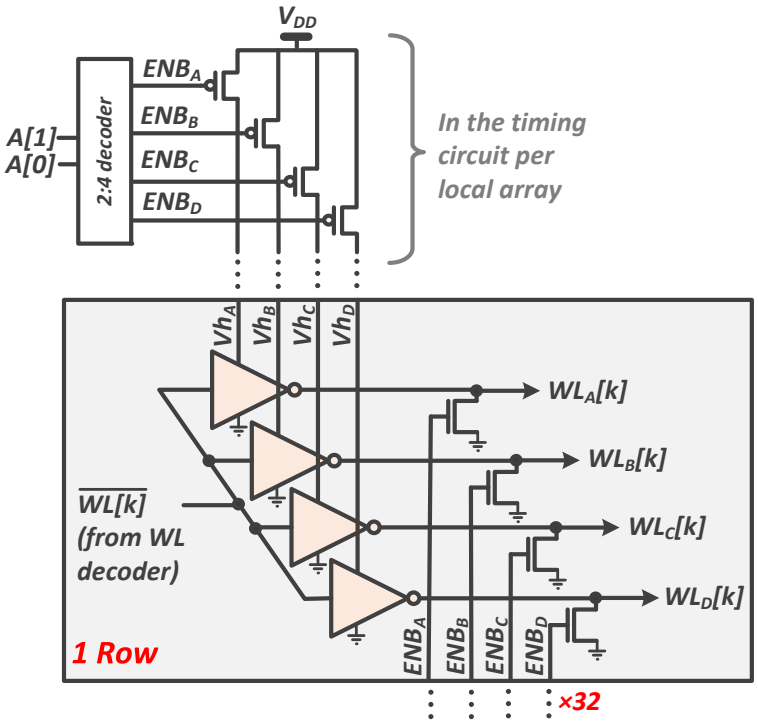


Figure 2-9: Schematic of the area-efficient driver design for the 4 WLs per row in a local array, using shared signals (Vh's and ENB's).

from the global BLs (which are driven to the data being written by the global R/W circuitry). During a read operation, the write path is turned off by driving *data* and *dataB* to "0". The pair of cross-coupled PMOS devices, connected to a local BL pair, maintains a differential signal level on the local BLs, during a write operation. Fig. 2-10 shows typical waveforms for a read "0" and a write "1" operation for one column in the local array. It should be noted that, the local R/W circuitry is shared between two local sub-arrays, which reduces its area overhead to only 9.1% in comparison to the bit-cell array.

All the global signals (including the global BLs and $Pred$, $PredB$ lines) are driven by the global R/W circuitry, which also incorporates small-signal sensing for the global BLs, during a read operation.

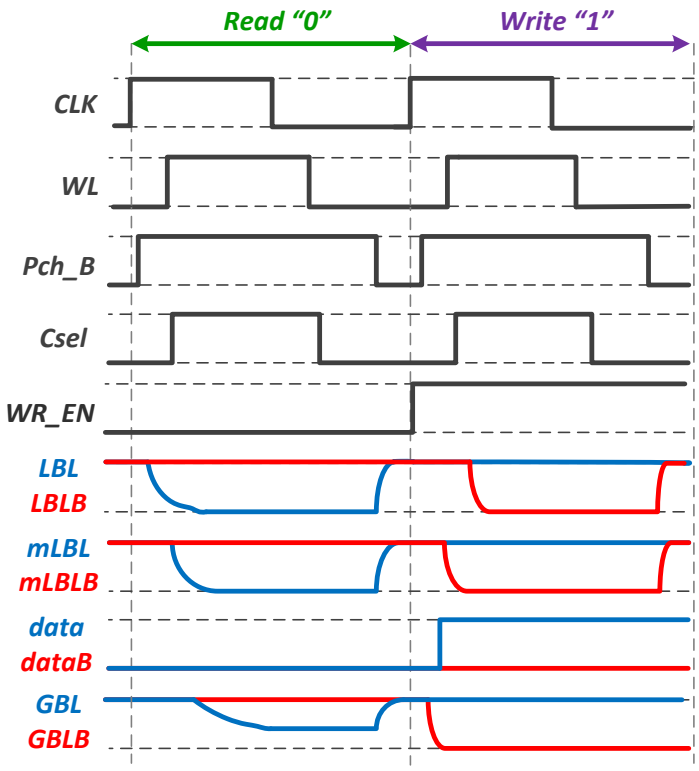


Figure 2-10: Typical operation waveforms for a read and a write cycle, showing the critical signals in the local R/W block.

2.3 Data Prediction in 6T SRAM

Application specific features can provide interesting data properties, which can be exploited to design a more tailored SRAM. [38] proposed a 10T SRAM bit-cell which uses prediction of data to reduce bit-line switching power during a read operation. It was targeted specifically towards motion-estimation in video processing applications. In motion estimation, the pixels from a small block of a video frame (reference buffer) is stored in the SRAM array and used in consecutive read cycles, before it is over-written. The correlation of the pixel data, stored in the reference buffer, can be exploited to predict the data during a read operation, using previously read values. If the prediction matches the actual data, the bit-line (BL) pair is not discharged. Thus, depending on the prediction accuracy, the BL switching power can be reduced. This can provide significant energy savings, since BL switching constitutes a major portion of the overall SRAM power consumption. [39] proposed another data-dependent 10T

SRAM, in which the prediction logic is implemented at the periphery, rather than the bit-cell itself. It uses the last data read as the prediction signal for the next data. Therefore, if the last cycle's prediction was correct, there is no need to pre-charge the BLs for the next cycle, which reduces BL switching energy. If the next cycle detects a wrong prediction, it updates the prediction value at the peripheral read circuit and pre-charges the corresponding BL. Since both the BLs can potentially be at 0V (due to wrong predictions), it can lead to unwanted write operations if regular 6T bit-cells, with common read and write BLs, are used. Hence, [39] uses a 10T bit-cell to prevent write disturb. One drawback of the prior prediction-based SRAM architectures is the need for 10T bit-cells, which consumes $> 1.5\times$ higher area than a 6T bit-cell. In many applications where SRAM area is of critical importance (e.g. GPU's), use of 10T bit-cells would not be feasible. To address this, we propose a prediction architecture which can use a 6T bit-cell. This is achieved by incorporating data prediction at the local array level, instead of embedding it in the 10T bit-cell as done in [38]. Thus, we get the area advantage of using smaller 6T bit-cells as compared to 10T designs [38, 39], while still saving BL switching power.

Fig. 2-11 shows the architecture implementing the prediction scheme for a 6T SRAM array. Two extra transistors (N_{p1}, N_{p2}) are added at the local sub-array level, which are controlled by the prediction signals: $Pred$ and $PredB$ (generated locally from $Pred$). N_{p1} and N_{p2} in turn, control the signal development at the '$int1$' and '$int2$' nodes, which drive the local sensing inverters. These inverters control the discharge of the global bit-lines using NMOS pull-down transistors, during a read operation (GWL is on). All the internal nodes are pre-charged to V_{dd} at the end of a clock cyle (using PMOS transistors, not shown in the figure). The operation waveforms of critical signals are shown in Fig. 2-12, for both correct and incorrect prediction cases. Let us assume the data to be read is "0". Hence, the LBL discharges to ground, during a read operation, while $LBLB$ stays at V_{dd}. If the prediction is correct, i.e. $Pred =$ "0" and $PredB =$ "1", N_{p1} is turned-off and the discharge of the LBL is not transferred to the '$int1$' node. Thus, both '$int1$' and '$int2$' nodes stay at the pre-charged level of V_{dd}. Hence, neither of the global bit-lines ($GBL, GBLB$)

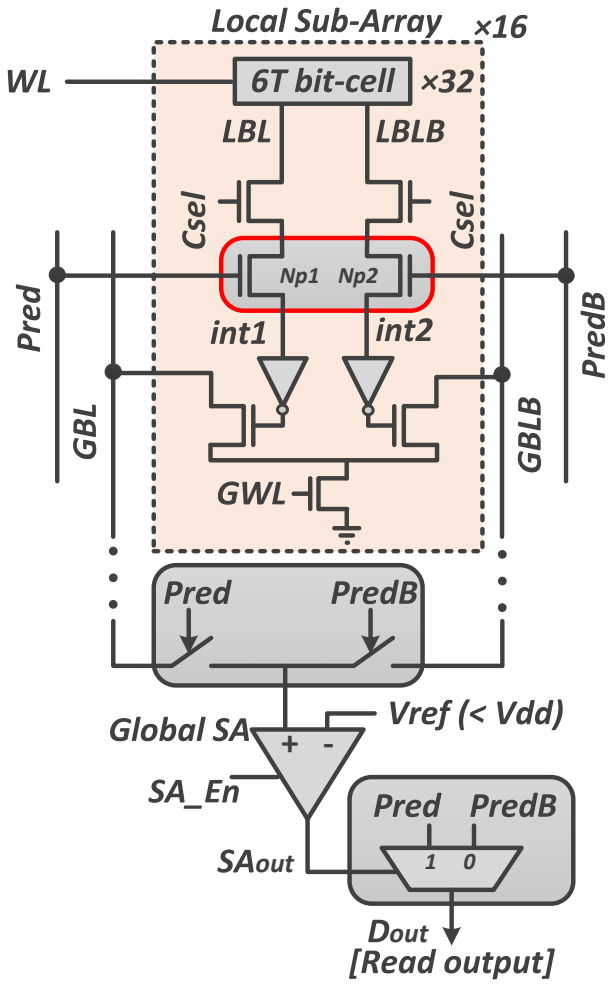


Figure 2-11: Proposed hierarchical data prediction architecture with 6T bit-cell array, to reduce read energy consumption. PMOS pre-charge transistors at every internal node are not shown.

are discharged from the pre-charged value of V_{dd}. The global sense-amplifier (SA) outputs a "1" and the correct prediction value, $Pred$ = "0", is chosen as the read output data. On the other hand, if the prediction is incorrect, i.e. $Pred$ = "1" and $PredB$ = "0", N_{p1} is turned-on and the discharge of the LBL is transferred to the '$int1$' node. Hence, GBL starts discharging and the global SA senses this, to output a "0". Therefore, $PredB$ (= "0") is chosen as the read output (since the prediction was incorrect). Thus, in either case, the correct value of the data (= "0" in this example) is obtained at the output. However, if the prediction was correct, the discharge of the global bit-line was avoided, which manifests into dynamic energy savings.

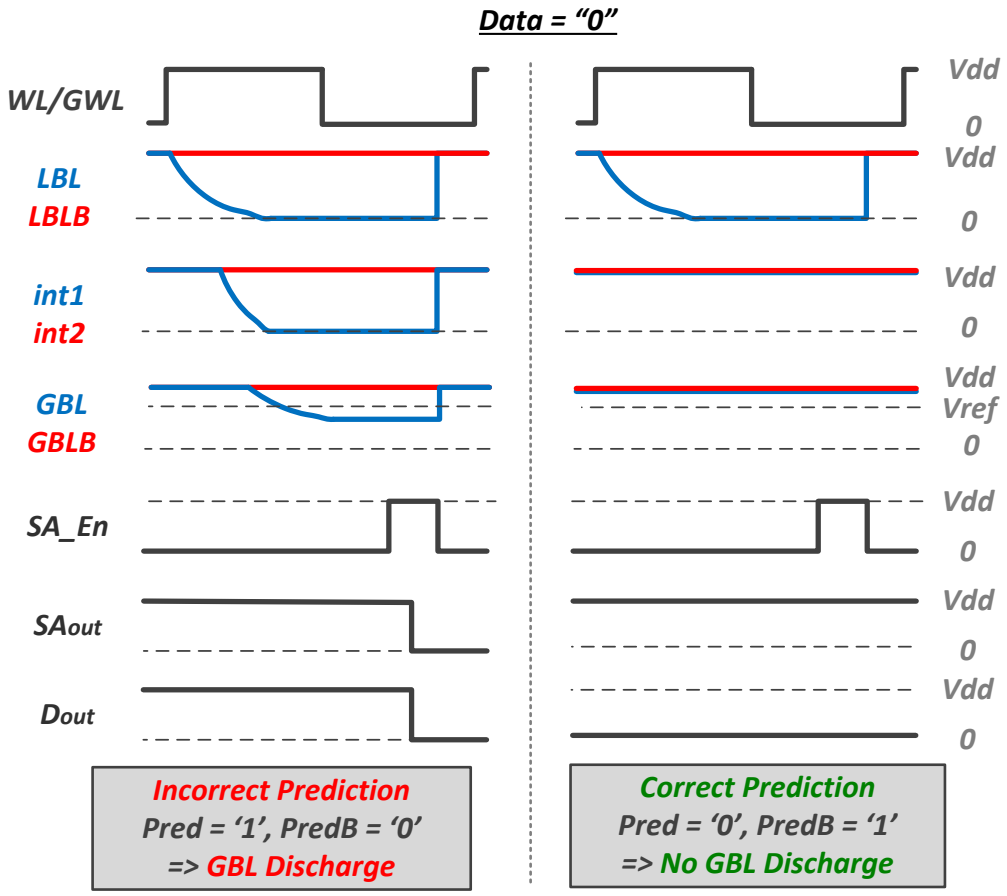


Figure 2-12: Typical operation waveforms during a read operation with the prediction architecture, shown for both correct and incorrect predictions (data stored: "0").

Therefore, in summary, without the data prediction technique, the BL switching energy per output data-bit for a hierarchical SRAM architecture, with single-sided read, is given by:

$$\begin{aligned} E_{BL}(w/o\ pred) &= E_{LBL} + E_{GBL} \\ &= N \times C_{LBL} \times V_{dd}^2 + p_1 \times C_{GBL} \times V_{dd} \times \Delta V_{RD,SA} \end{aligned} \tag{2.2}$$

where, C_{LBL} and C_{GBL} are the capacitances for 1 LBL and 1 GBL, respectively. N is the column-mux ratio, i.e. for every N local columns there is one global column. p_1 is the probability that the data read is a "1", assuming it is the polarity for which there is GBL switching with a single-sided read architecture. $\Delta V_{RD,SA}$ is the average GBL swing for the SA to detect the data correctly.

On the other hand, with the proposed data-prediction technique, the BL switching energy per output data-bit is given by:

$$\begin{aligned} E_{BL}(w/pred) &= E_{LBL} + E_{GBL} \\ &= N \times C_{LBL} \times V_{dd}^2 + (1 - p_{correct}) \times C_{GBL} \times V_{dd} \times \Delta V_{RD,SA} \end{aligned} \tag{2.3}$$

where, $p_{correct}$ is the probability of correct data prediction on the average, for a given application and a prediction scheme.

Although the local BL switching is not affected in this technique, the dominant component of the switching energy, which is due to the global BLs, can be reduced by using data prediction. Furthermore, the local BL switching energy is reduced due to the use of multiple word-lines in the proposed implementation, which eliminates BL switching energy for un-selected columns in a local sub-array.

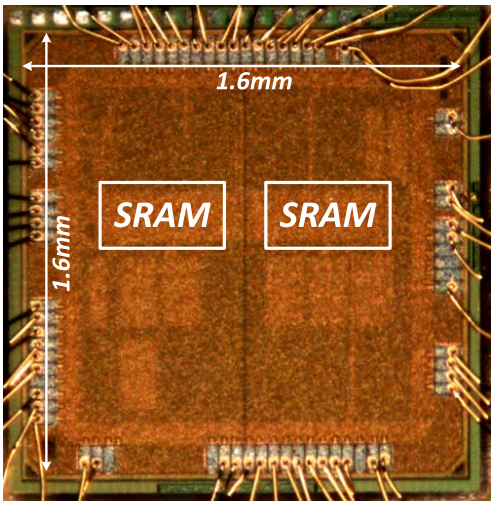
2.4 Using the Prediction Architecture for In-memory XOR/XNOR Computations

The SRAM read scheme with the prediction architecture presented in the last section, can be also interpreted as an XNOR computation between the data stored (D) in the memory bit-cell (which is to be read) and the *Pred* signal. This can be understood by observing the global SA output signal, SA_{out} (Fig. 2-11). For the case shown on the left in Fig. 2-12, SA_{out} = "0" after the SA evaluation, which is the same as $\overline{D \oplus Pred} = \overline{0 \oplus 1}$ = "0". Similarly, for the case shown on the right, SA_{out} = "1" after the SA evaluation, which is again the same as $\overline{D \oplus Pred} = \overline{0 \oplus 0}$ = "1". There are many applications in which, there is a need to perform bit-wise XOR/XNOR operations on wide bit-width words. For example, in crytographic algorithms such as AES [46, 52], during the "AddRoundKey" step, the state bits (derived from the plain-text bits) are XOR-ed with the secret key bits to generate the encrypted message bits. [43] implements an in-memory XOR computation architecture for accelerating cryptographic alogrihtms. However, it needs 10T SRAM bit-cells, which results in

3× larger memory area vs. compiled 6T SRAMs. On the other hand, our architecture still uses 6T bit-cells, which would lead to a very area-efficient implementation. Next, in machine learning applications e.g. binary neural-networks [47, 48], there is a need to perform bit-wise XNOR operation between the external input data (X: 1-b dot-product inputs) and the stored memory word (W: 1-b dot-product weights), to compute the output: $Y = \sum_i \overline{X_i \oplus W_i}$.

For all these applications, mentioned above, our SRAM architecture would be very beneficial to improve the speed of the overall operation. This is because both the memory read and the X(N)OR computation operations can happen in the same clock-cycle, compared to 2 cycles to do a regular memory read followed by the X(N)OR computation outside the memory array. In addition, the results of the X(N)OR computations can be written back to the same memory, without the need for data transfer between the memory and the processing engine (which would be needed in the conventional design). Thus, there can be potential energy savings due to reduction in the data movement.

2.5 Measurement Results



Technology	28nm FDSOI
Macro Size	128Kb (6T)
Macro Area	$0.046mm^2$
Macro Organization	(36×256)×16 sub-arrays
# output bits	64
Operating voltage	0.34V - 1.0V
Assist techniques	Dynamic FBB: upto 1V (Wr), upto V_{dd} (Rd)
Min. Energy/access	3.4pJ @ 0.45V, 40 MHz

Figure 2-13: Die photo and summary of the 128Kb 6T SRAM test chip fabricated in 28nm FDSOI technology.

Fig. 2-13 shows the die photo and test chip summary of the 128Kb 6T SRAM.

12 different macros from 3 different chips were tested. All of them work without any bit-error down to 360mV, (Fig. 2-14) without any assists. As seen from Fig. 2-14, FBB reduces the $V_{dd,min}$ by 20mV; beyond 340mV read-disturb errors start to occur. The improvement in $V_{dd,min}$ is not as expected from simulations. We suspect this is because, the fabricated chips that we received are in the typical (TT) process corner, whereas the write-margin simulations were done for the worst case corner (SF). Hence, the TT SRAMs already work down to low V_{dd}'s without any write-assists. Nevertheless, in cases where write operation limits the SRAM functionality, FBB would be helpful. The measured energy overhead due to DFBB in the proposed implementation is 4.5× lower compared to the conventional approach (column-wise n-wells). This demonstrates the benefit of the proposed layout with row-wise shared n-wells.

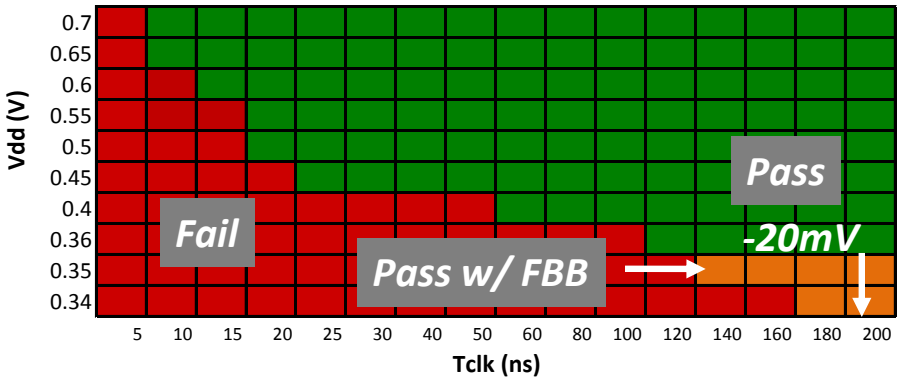


Figure 2-14: Shmoo plot of the 128Kb SRAM macro (25°C).

Fig. 2-15 shows the measured average energy per word access (64-bit) of the 128Kb 6T SRAM macro, along with the maximum operating frequency for correct memory functionality, at different V_{dd}'s. The macro achieves a minimum energy consumption of 3.36pJ/word-access i.e. 52.5fJ/bit-access, at 0.45V, 40 MHz. Compared to 0.7V, a 2× energy reduction is achieved at 0.45V, whereas, the SRAM can run at 3× higher frequency at 0.7V. On the other hand, at the minimum operating voltage of 0.36V (w/o FBB), the 128Kb macro still achieves a low energy of 4.6pJ/word-access i.e. 71.9fJ/bit-access, operating correctly up to 9.1MHz. This would be useful in IoT devices, with very limited energy budget and which can only run from very low

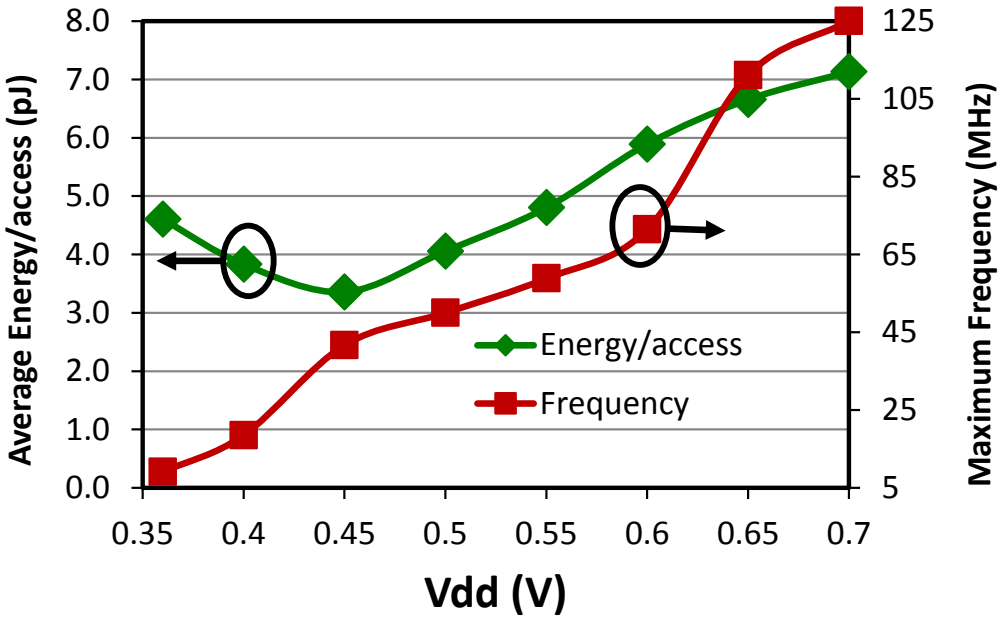


Figure 2-15: Average energy/word-access and maximum operating frequency vs. SRAM V_{dd}.

supply voltages (limited by the energy-harvesting capability of the device).

Data prediction is used during a read operation, to reduce the global BL switching. With a single-ended read architecture, the baseline approach (without prediction) would not consume any global BL switching energy for one polarity of the data (say '1'). Hence, in order to have a fair comparison, the SRAM array was configured with 50% 1's and 50% 0's . Fig. 2-16 shows the normalized dynamic read energy ($E_{read,dyn}$) at a V_{dd} of 400mV, as a function of the percentage of correct data prediction. Since, the conventional approach would only consume power for 50% of the data (equal % of 0's and 1's in the memory), the benefit of the prediction technique is only apparent for $>$ 50% correct prediction. With a 100% correct data prediction, a 36% reduction in $E_{read,dyn}$ is achieved. It may be noted that the 36% reduction is for the present array architecture with 32 rows per local bit-line, achieving only 9.1% area overhead. Increased benefits can be achieved if the local BL energy is further reduced. One simple way is to have fewer bit-cells per local BL. However, this comes with a decreased area-efficiency of the array, due to the need of more local read/write circuits.

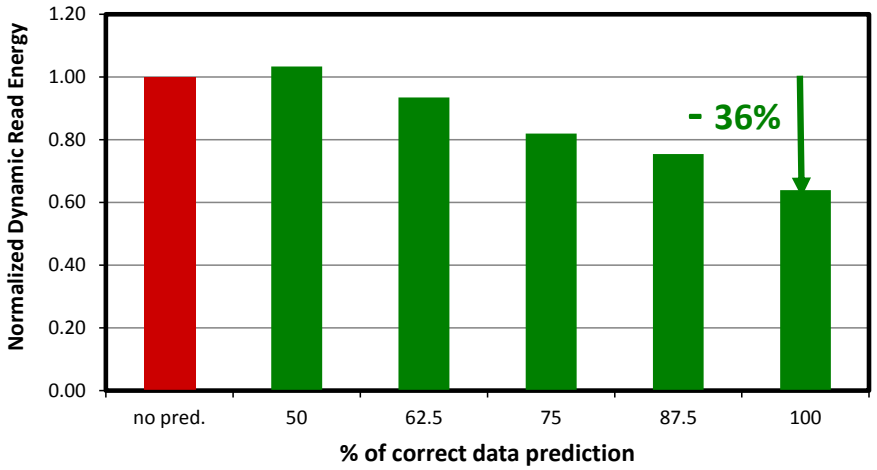


Figure 2-16: Improvement in dynamic read energy using data prediction.

Table 2.1 compares this work with state-of-the-art low voltage and low energy SRAM designs, in similar process technologies. As seen from the table, this work achieves the lowest energy/bit-access while still maintaining a low SRAM area by efficiently utilizing the 6T bit-cell. Compared to an implementation [53] (which uses larger 9T bit-cells) in the same 28nm FDSOI technology, our work performs better in terms of both area and energy consumption of the memory, while working at similar clock frequencies.

Table 2.2 compares the energy savings of our prediction-based hierarchical read approach with other works on data-dependent SRAMs. [38, 39] use 10T bit-cells and [37] uses 8T bit-cells, which have higher area overhead (33% - 67%) compared to our implementation (9.1%). However, [38, 39] provide more energy savings, because for correct predictions they can fully suppress BL switching. Whereas in our approach, even with 100% correct predictions, there is local BL switching due to the use of 6T bit-cells with 2-sided read ports. Compared to [37] our approach performs better, because [37] needs extra energy to selectively invert (to data '1') and store majority data-bits in the 8T bit-cells, which are hard-wired to not switch BLs for data '1'.

Table 2.1: Comparison with state-of-the-art low-V_{dd} SRAMs

Design	**This work [54]**	[53] ASSCC'17	[49] ISSCC'11	[55] VLSIC'13	[56] JSSC'13
Technology	28nm FDSOI	28nm FDSOI	28nm Bulk	20nm Bulk	65nm LP Bulk
Bitcell structure	6T	9T	6T	6T	6T
Bitcell area (um^2)	0.232[1]	N/A	0.12	N/A	1.092[1]
Memory Size	128Kb	16Kb	128Kb	128Kb	128Kb
Organization: rows × col.s × macros	512 × 256 ×1	256 × 64 ×1	128 × 256 ×4	512 × 64 ×4	64 × 512 ×4
Col. mux ratio	4	1	4	2	16
I/O Width	64-b	64-b	64-b	32-b	32-b
$V_{dd,min}$**(V)**	0.36	0.47	0.6	0.6	0.4
Freq. (MHz) @ $V_{dd,min}$	9	6.7	20	N/A	11.1
E_{min}**/bit (fJ)**	**52.5** @ 0.45V	105 @ 0.5V	2187.5 @ 0.6V	67.2 @ 0.6V	103.13 @ 0.4V

[1] Logic design rules

Table 2.2: Comparison with data-dependent SRAMs

Design	**This work [54]**	[38] JSSC'14	[39] JSSC'17	[37] TVLSI'08
Bitcell structure	6T	10T	10T	8T
Area overhead[1]	9.1%	66.7%[2]	66.7%[2]	33.3%[2]
Memory Size	128Kb	32Kb	16Kb	68Kb
Test case	50% 0's, 50% 1's	50% 0's, 50% 1's	Video Frames	Video Frames
Energy savings	up to 36%[3] vs. 6T with fixed prediction	up to 43% vs. 8T	up to 67% vs. 8T	28% vs. 8T with no majority logic

[1] vs. regular 6T with no prediction, no hierarchical structure in the macro.
[2] Estimated using number of transistors in the bit-cell.
[3] Dynamic energy only.

2.6 Conclusion

In this work we presented an energy-efficient, low-voltage 128Kb 6T SRAM macro. Forward body-biasing (FBB) is used dynamically to improve the write margin. Layout modifications are proposed for the array to reduce the energy overhead of dynamic FBB and to reduce unselected local bit-line switching by ∼25%. This work achieves a minimum energy/bit-access of 52.5fJ at 0.45V and zero-error rate at the minimum V_{dd} of 0.36V, while still having the area-advantage by using 6T bit-cells. Hence, this SRAM would be very useful in IoT applications, where limited energy budget necessitates low voltage and low power operations. Our work would be also useful in applications (e.g. GPUs) where SRAM density is of key concern and using 8T/10T bit-cells might not be preferred. We also implemented area-efficient logic which utilizes data prediction in the 6T read path to reduce bit-line switching energy. Upto 36% reduction in dynamic read energy is obtained with 100% correct data prediction. More power reduction can be obtained by using smaller local arrays, at the cost of decreased area-efficiency of the SRAM. The prediction-based read technique can also be utilized for in-memory bit-wise XOR/XNOR computations, useful in security (e.g. AES) and machine learning applications (e.g. binary neural-networks).

Chapter 3

In-Memory Computation for Neural-Network based Low-Power Machine Learning Applications

Artificial intelligence (AI) and machine learning (ML) are changing the way we interact with the world around us. Speech recognition [4] allows us to interact with "smart" devices using our voices. Facial recognition [2] enables using our faces to get access to devices in a more intuitive manner, instead of traditional passcodes. As we think about extending machine intelligence to more and more devices around us, in the "Internet-of-Things" (IoT), "edge-computing" i.e. computing on these edge devices vs. the "cloud" becomes increasingly important. There are a multitude of reasons for this. Firstly, "edge-computing" enables the devices to make fast decisions locally, without having to wait for the "cloud". Secondly, it can significantly reduce the communication traffic to the "cloud", by only sending the critical/relevant information and filtering out the rest of the massive amount of data the edge-devices may collect. Furthermore, "edge-computing" helps in improving the security of the data by keeping it local (within the devices), rather than having to send sensitive information to the "cloud". While "edge-computing" promises significant benefits for IoT devices, it also has certain requirements. The circuits to run the compute algorithms must be very energy-efficient, to extend the battery-life of these IoT devices, most

of which have a very limited energy budget. Additionally, in many applications, the local decision-making has to be done in real-time (e.g. self-driving cars), to make them practical.

Convolutional neural networks (CNN) provide state-of-the-art results in a wide variety of AI/ ML applications, ranging from image classification [3] to speech recognition [4]. However, they are highly computation-intensive and require huge amounts of storage. Hence, they consume a lot of energy when implemented in hardware and are not suitable for energy-constrained applications e.g. "edge-computing".

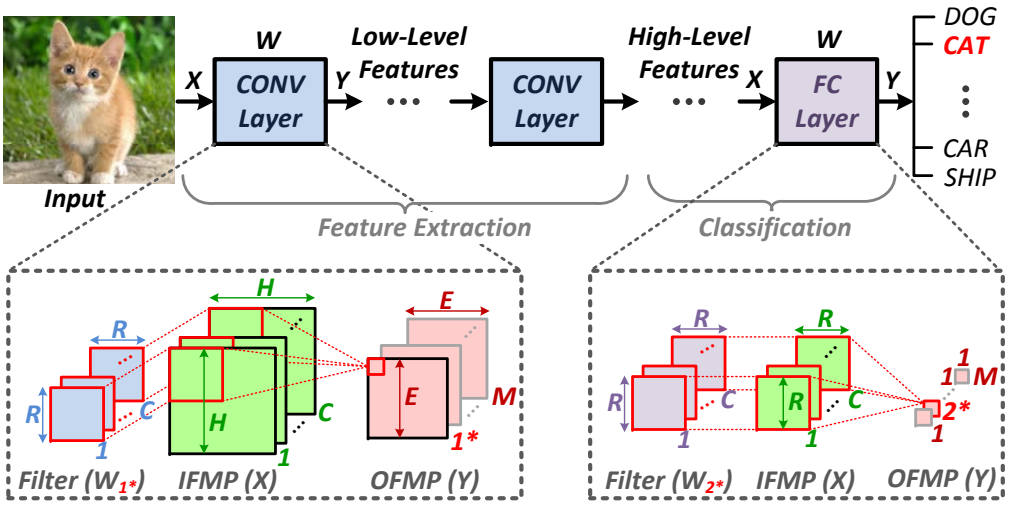


Figure 3-1: Basics of a typical convolutional neural network (CNN) for a classification problem, showing the structure for the CONV and FC layers [15].

CNNs typically consist of a cascade of convolutional (CONV) and fully-connected (FC) layers (Fig. 3-1), with some non-linear layers in between (not shown in the figure). The CONV layers extract different features of the input and the FC layers combine these features to finally assign the input to one of the many pre-determined output classes. For each of the CONV/FC layers, there is a set of 3-D filters (W_k), which are applied on the 3-D input feature-map (IFMP) to that layer and generate its 3-D output feature-map (OFMP). Each 3D filter/input consist of mutliple 2D arrays, each of which corresponds to a different "channel" (1 to C). When a 3-D filter (W_k) is applied to the input (X), an element-wise multiplication is performed, followed by addition of the partial products to compute the convolution output (Y_k). For CONV

layers the 3-D filter is applied on the shifted input to compute the next element in the 2-D OFMP. Each individual filter corresponds to a different channel in the 3-D OFMP. Therefore, the fundamental operation for both the CONV and FC layers can be simplified to a dot-product or a multiply-and-accumulate (MAC) operation, as shown in equation (3.1).

$$
\begin{aligned}
Y_{x,y,k} = \sum_{c=1}^{C}\sum_{j=1}^{R}\sum_{i=1}^{R} W_{i,j,c,k} \times X_{x+i,y+j,c} \\
1 \leq (x+i), (y+j) \leq H, \\
1 \leq x, y \leq H - R + 1 (= E), \\
1 \leq k \leq M
\end{aligned}
\tag{3.1}
$$

where, H is the width/height of the IFMP (with padding), E is the OFMP width/height, R is the filter width/height, C is the number of IFMP/filter channels and M is the number of filters/OFMP channels for a given CONV/FC layer. The width and height of the feature-maps/filters are assumed to be same for simplicity and also because it is very common in most of the popular CNNs.

In general, CNNs use real-valued inputs and weights. However, in order to their reduce storage and compute complexity recent work have strived towards using small bit-widths to represent the input/filter-weight values. [47] proposed a binary-weight-network (BWN), where the filter weights (w_i's) can be trained to be +1/-1 (with a common scaling factor per filter: α). This leads to a significant reduction in the amount of storage required for the w_i's, making it possible to store them entirely on-chip. BWN's also simplify the MAC operation to an add/subtract operation, since α is common for a given 3D filter and it can be incorporated after finishing the entire convolution computation for that filter. As shown in [47], this algorithm does not compromise much on the original classification accuracy of the CNN, obtained using full precision weights. BWN performs better than binary-connect [57], which does not incorporate the scaling factor of α per filter, and also binarized-neural-networks [48], where both weights and activations are constrained to ± 1.

In the conventional all-digital implementation of CNNs [58, 59], with the mem-

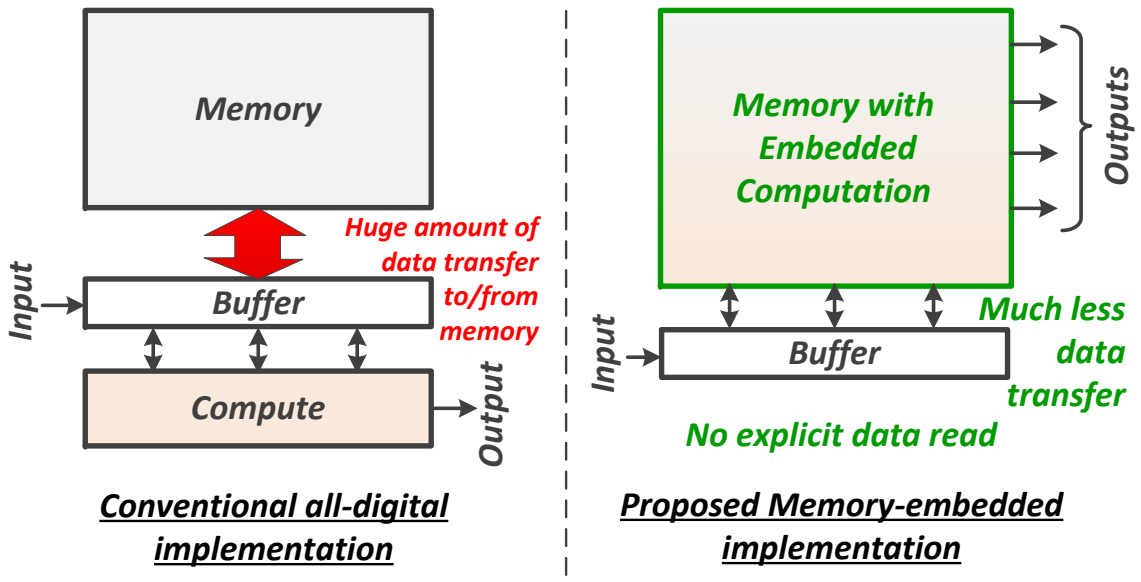


Figure 3-2: Comparison of conventional approach vs. proposed approach of memory-embedded convolution computation, for processing of CNNs.

ory and the processing elements being physically separate, reading the w_i's and the partial sums from the on-chip SRAMs lead to a lot of data movement per computation [15] and hence, make them energy-intensive. This is because in modern CMOS processes, the energy required to access data from memory can be much higher than the energy needed for a compute operation with that data [13]. To address this problem, we present an SRAM-embedded convolution computation architecture [60], conceptually shown in Fig. 3-2. Embedding computation inside memory has two significant benefits. Firstly, data transfer to/from the memory is greatly reduced, since the filter-weights are not explicitly read and only the computed output is sent outside the memory. Secondly, we can take advantage of the massively parallel nature of CNNs to access multiple memory addresses simultaneously. This is because we are only interested in the result of the computation using the memory data and not the individual stored bits. Therefore, a much higher memory bandwidth can be achieved with this approach, overcoming some of the major limitations posed by the conventional "von-Neumann bottleneck".

3.1 Concept of SRAM-Embedded Compute

The basic operation involved in evaluating convolutions (Y) for CNNs is the dot-product of the 3-D IFMP (X) and the filter-weights (W), as shown in equation (3.1). It can be re-written by flattening the 3-D tensor into a 1-D vector to obtain equation (3.2), where the 2-D subscripts (x, y) have been omitted for simplicity.

$$Y_k = \sum_{i=1}^{R\times R\times C} W_{k,i} \times X_i \tag{3.2}$$

Equation (3.2) can be further simplified for the case of binary filter-weights (w_i's) to get equation (3.3a), where α_k is the common coefficient for the k^{th} filter. If α_k is expressed as a ratio of two integers (M_k, N) then we get equation (3.3b).

$$Y_k = \alpha_k \sum_{i=1}^{R\times R\times C} w_{k,i} \times X_i, \quad w_i \in (+1, -1) \tag{3.3a}$$

$$= \frac{M_k}{N} \sum_{i=1}^{R\times R\times C} w_{k,i} \times X_i, \quad M_k, N \in I^+ \tag{3.3b}$$

Now, if we separate out the scaling factor of M_k (which can be incorporated after computing the entire dot-product), we get the expression for the effective convolution output (Y_{OUT}) as:

$$Y_{OUT,k} = \frac{1}{N} \sum_{i=1}^{R\times R\times C} w_{k,i} \times X_{IN,i} \tag{3.4}$$

where X_{IN} is the effective convolution input, i.e. scaled version of the original input X, limited to 7-b (includes 1-b sign). For energy-efficient computation with multi-bit values inside the memory, equation (3.4) has to be implemented in the analog domain, as shown in equation (3.5).

$$V_{Y\_AVG,k} = \frac{1}{N} \sum_{i=1}^{R\times R\times C} w_{k,i} \times V_{a,i} \tag{3.5}$$

The equivalence of equations (3.4) and (3.5), conceptually shown in Fig. 3-3, becomes apparent in 3 key steps. First, the digital inputs (X_{IN}'s) are converted into

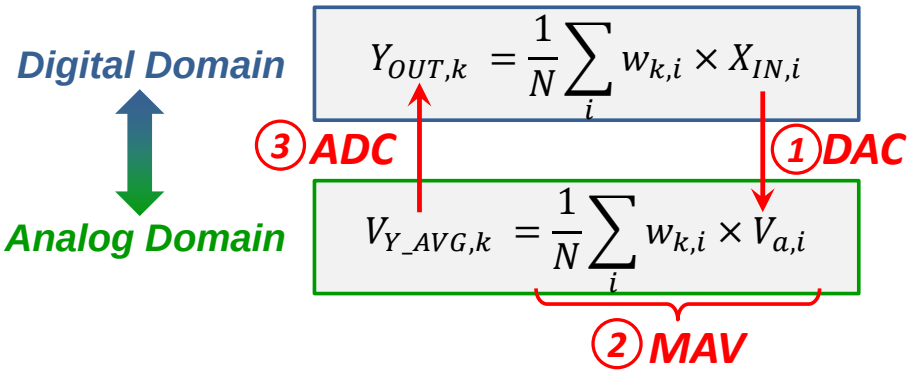


Figure 3-3: Concept of embedded convolution computation as averaging in SRAMs for binary-weight convolutional neural networks.

analog voltages (V_a's) using digital-to-analog converters (DAC). Then, the analog voltages are multiplied by the corresponding 1-bit filter-weights (w_i's), which are stored in a memory array. This is followed by averaging over N terms to get the analog-averaged convolution output voltage (V_{Y_AVG}). These constitute the second step: multiply-and-average (MAV). Finally, in the last step, the analog-averaged voltage is converted back into the digital domain (Y_{OUT}) using an analog-to-digital converter (ADC), for further processing. It may be noted that if the 3-D filter size ($R \times R \times C$) is greater than N, the above-mentioned 3-step process is repeated multiple (N_r) times using $R \times R \times C'$ ($\leq N$) elements in each cycle, where $N_r = C/C'$. The partial outputs (from the ADC) can then be further added digitally (outside the memory) to get the final convolution output.

3.2 Overall Architecture

Fig. 3-4 shows the overall architecture of the 16Kb conv-SRAM (CSRAM) array, consisting of 256 rows by 64 columns of SRAM bit-cells. It is divided into 16 local arrays, each with 16 rows. Each local array is meant to store the binary filter-weights (w_i's) for a different 3-D filter in a CONV/FC layer. w_i is stored in a 10T SRAM bit-cell as either a digital '0' or a digital '1', depending on whether its value is +1 or −1 respectively. Each local array has its analog averaging circuits (MAV_a's) and a

dedicated ADC to compute the partial convolution outputs (Y_{OUT}'s). Sharing these circuits for 16 rows in a local array reduces the area overhead. The IFMP values (X_{IN}'s) are fed into column-wise DACs, which convert the digital X_{IN} codes to analog input voltages on the global bit-lines ($GRBL$'s). The $GRBL$'s are shared by all the local arrays, implementing the fact that in CNNs each input is shared/processed in parallel by multiple filters. With this architecture, the 16Kb CSRAM array can process a maximum of 64 convolution inputs and compute 16 convolution outputs in parallel.

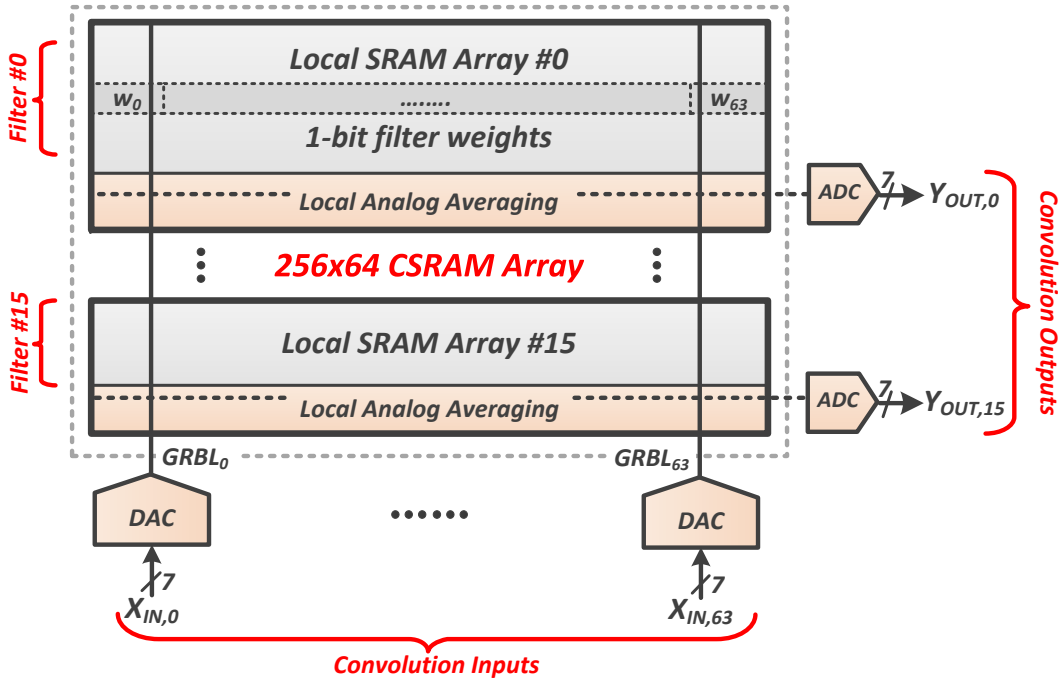


Figure 3-4: Overall architecture of the Conv-SRAM (CSRAM) showing local arrays, column-wise DACs and row-wise ADCs to implement convolution as weighted averaging.

Fig. 3-5(a) shows the simulated test error-rates for the MNIST dataset with the LeNet-5 CNN, consisting of 2 CONV layers (C1, C3) and 2 FC layers (F5, F6) [1]. The number of bits to represent the IFMP/OFMP values are varied from 8 to 4. Lower bit-width helps in reducing the area/power costs of the DAC and ADC circuits involved for the convolution computations. However, as seen from Fig. 3-5(a) the error rate starts to increase steeply for <7-b. Similar results (Fig. 3-6(a)) are observed for the

[1]Filter sizes: (C1) $5 \times 5 \times 1 \times 6$ (C3) $5 \times 5 \times 6 \times 16$ (F5) $5 \times 5 \times 16 \times 120$ (F6) $1 \times 1 \times 120 \times 10$

CIFAR dataset [61] (for small image classification) with a 5-layer CNN, consisting of 3 CONV layers (C1, C2, C3) and 2 FC layers (F4, F5) [2]. Hence, 7-b is chosen as the target bit-width for the DAC/ADC circuits. With 7-b (including the sign bit) the voltage resolution needed on a 1V scale is $1LSB = 1/2^6 \approx 15.6$ mV. Next, the effect of the averaging factor ('N') on the test error-rate is observed. A high value of 'N' would decrease the area/power overhead of the ADC by amortizing it over more MAV operations per clock-cycle. However, higher 'N' can also degrade the computation accuracy due to increased quantization by averaging. This is more crticial for CNN layers with smaller filter sizes. As shown in Fig. 3-5(b), for layer F6 of the LeNet-5 CNN, with a 3-D filter size of 120, the error-rate steeply increases as 'N' is varied from 15 to 120. Similarly for the CIFAR dataset, N>50 has a strong negative impact on the error rate of layer C2 (Fig. 3-6(b)). With a 2-D filter size of 5×5 for a given CNN layer, a minimum $N = 25$ is required to fit at-least 1 full filter channel per CSRAM row. We chose $N = 64$ to fit 2 channels for 5×5 filters, 4 channels for 4×4 filters and 64 channels for 1×1 filters, without sacrificing much on the error rate.

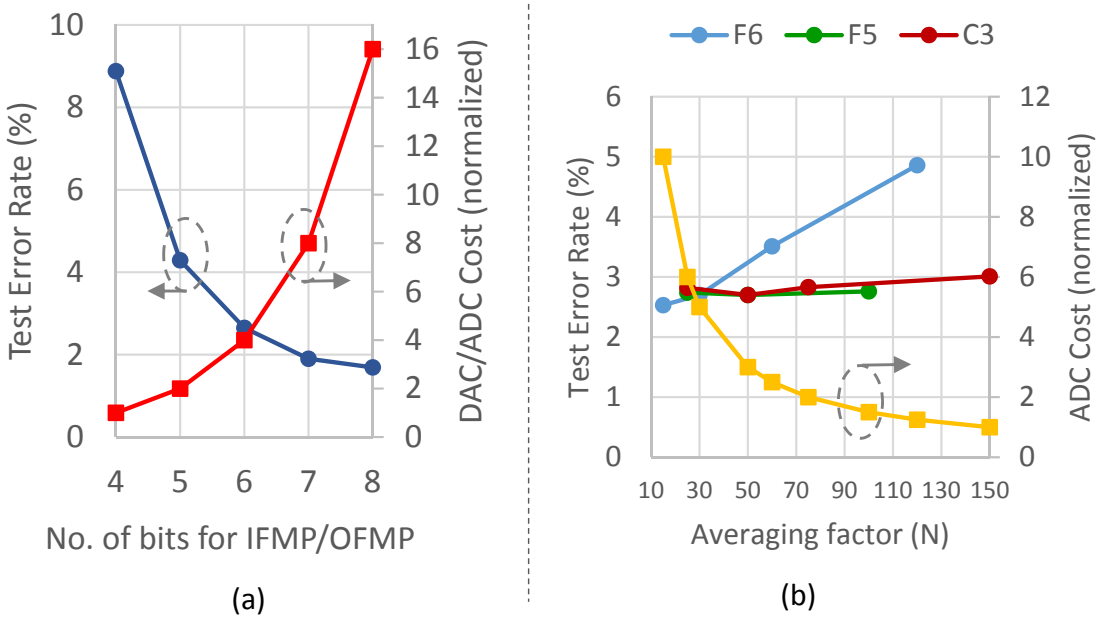


Figure 3-5: Simulated results for the MNIST dataset with the LeNet-5 CNN by varying: (a) bit-width to represent IFMP/OFMP values, (b) averaging factor (N).

[2]Filter sizes: (C1) $5 \times 5 \times 3 \times 32$ (C2) $5 \times 5 \times 32 \times 64$ (C3) $5 \times 5 \times 64 \times 64$ (F4) $4 \times 4 \times 64 \times 128$ (F5) $1 \times 1 \times 128 \times 10$

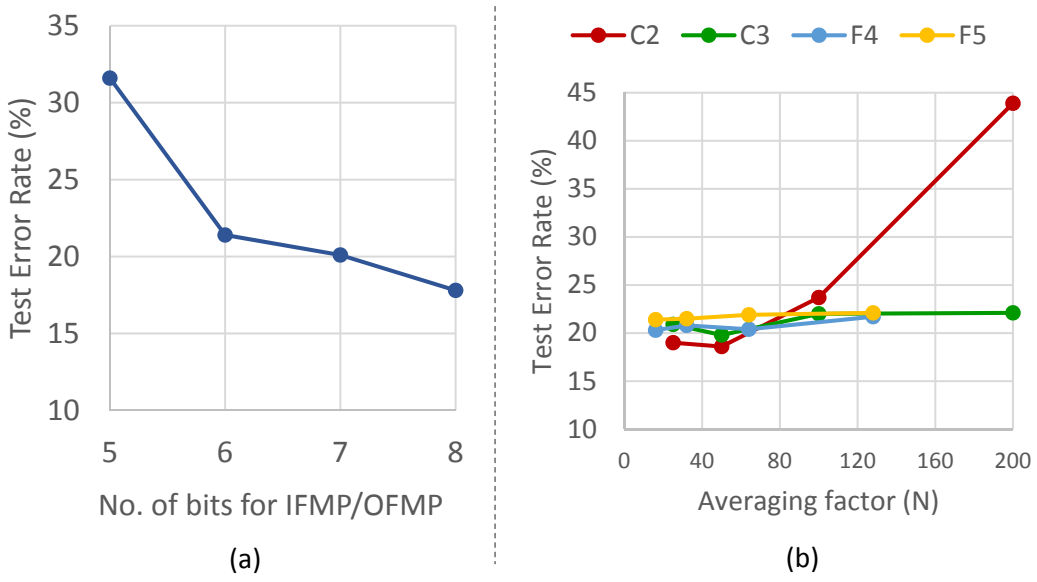


Figure 3-6: Simulated results for the CIFAR dataset with a 5-layer CNN by varying: (a) bit-width to represent IFMP/OFMP values, (b) averaging factor (N).

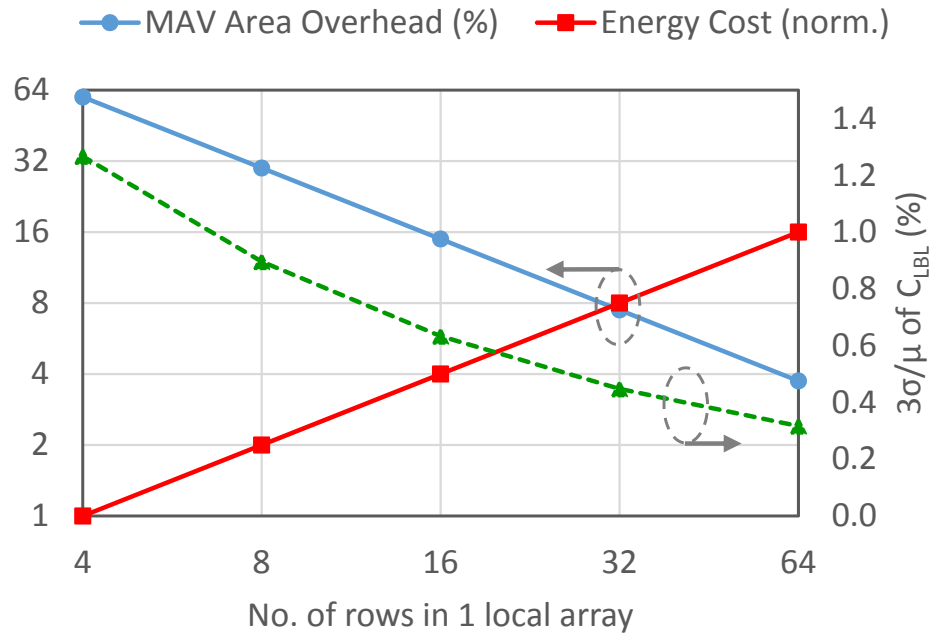


Figure 3-7: Effect of the number of rows in a local CSRAM array.

The number of rows (N_{rows}) per local array in the CSRAM determines the unit capacitance (C_{LBL}), which is used for all the analog operations required for the in-memory convolution computation. For every column in a local array, there is a corresponding MAV_a circuit. Hence, a higher value of N_{rows} would decrease the area-

overhead of MAV_a, by amortizing it over multiple rows. It also reduces variation of the C_{LBL} value (Fig. 3-7), which helps in improving accuracy of the computations. However, a high value of N_{rows} also means a high unit capacitance, which translates to increased energy costs. Therefore, $N_{rows} = 16$ is chosen as a trade-off. It may be noted that with $N_{rows} = 16$, the thermal noise ($\frac{kT}{C}$) is $< 1mV$, which is well below 1LSB = 15.6mV. Hence, the analog computations are not affected by thermal noise.

3.3 Key contributions of this work

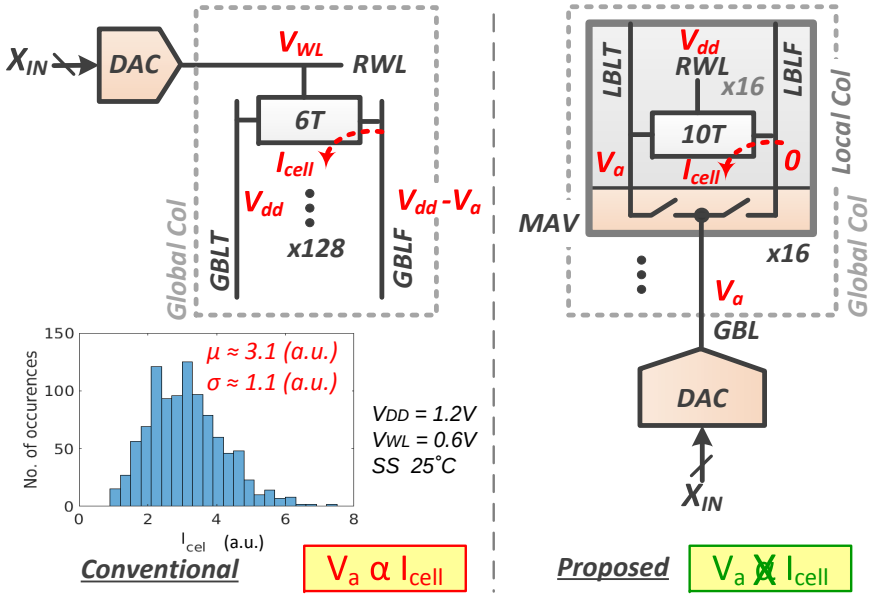


Figure 3-8: Comparison of the conventional and proposed approaches of using SRAM bit-cells for embedded analog computations.

While there are a few different approaches [41, 42, 62, 63] for in/near-memory computing, the proposed architecture has some key contributions, which provide significant benefits over prior work. The first key feature of our approach is the robustness to SRAM bit-cell V_t variations. SRAM bit-cells use near-minimum transistor sizes available in a given CMOS process and hence, suffer from transistor mismatch and variation. For example, if we consider the discharge current (I_{dis}) through an SRAM bit-cell (shown in Fig. 3-8) we can observe that it has a significant spread from its mean value ($\sigma \approx 30\%\mu$). Now, when I_{dis} is used to modulate the analog

voltage (V_a) on the bit-line [41, 42], there is a wide variation in the V_a value and it cannot be controlled very well. This compromises the computation accuracy and extra algorithmic techniques might be required to compensate for that. [41] uses the 'AdaBoost' technique, in which the results of many weak classifiers are combined to get a more accurate final result. However, this is not ideal for neural networks, since the number of computations is already very large and having multiple of them would only increase costs. [63] proposed an on-chip training to compensate for chip-to-chip variations. However, this is also not very feasible owing to the energy and timing penalty required to re-train the network corresponding to every single chip. In our approach (Fig. 3-8), the analog voltage (V_a) is directly sent to the bit-lines using global DACs at the periphery. Since the global DACs can be upsized, with their area being amortized over mutliple rows (256 in this case), the variation due to it is significantly less compared to that of the bit-cell. Furthermore, the SRAM bit-cell is only used to multiply V_a by the 1-bit filter weight (w_i) stored in it, using full signal swing locally. That means, the purpose of the SRAM bit-cell is to discharge one of its local bit-lines to 0, it is not used to control V_a. Hence, given enough time for the worst-case bit-cell discharge, the computation accuracy does not suffer from local bit-cell V_t variations.

The second key feature of our approach is the improvement of the dynamic voltage range for the analog computations without disturbing any bit-cell. In the conventional approach (with 6T SRAM bit-cells) [41, 42], where multiple word-lines (WL) are activated for the same bit-line, there might be a situation where one of the accessed bit-cells in that column is in pseudo-write mode (Fig. 3-9). This is because multiple activated bit-cells in that column can discharge a bit-line to a very low voltage, which could overwrite the data stored (Q_k = '1') in the disturbed bit-cell. Hence, the bit-line voltage range has to be limited to prevent any write-disturb. In our approach, 10T bitcells are used which de-couple the read and the write ports, to prevent any write-disturb. Furthermore, each bit-cell is read independently in parallel without sharing any bit-lines. And hence, the discharge on one bit-line cannot affect another accessed bit-cell. Thus, we can utilize a wide voltage range (close to full-rail) for the

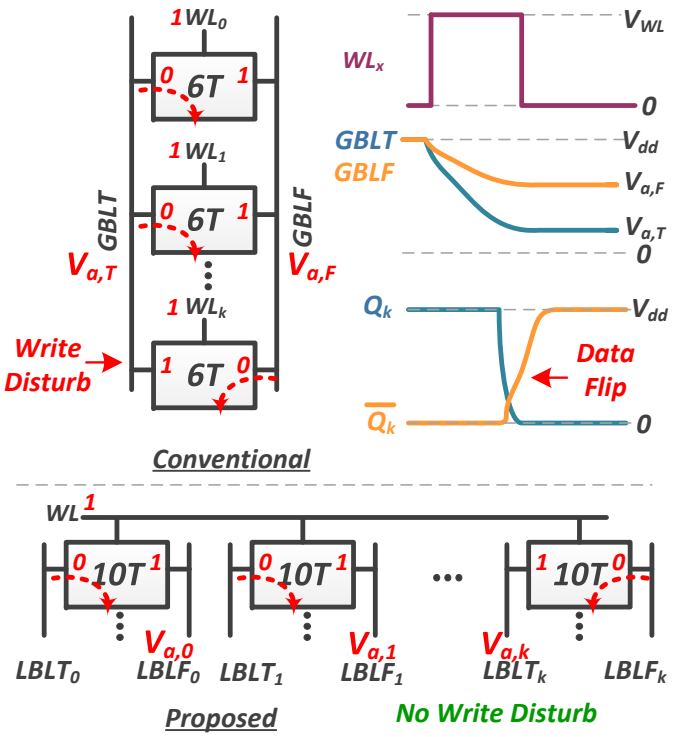


Figure 3-9: Comparison of the conventional and proposed approaches on write-disturb issue of SRAM bit-cells during compute mode.

analog computations, without disturbing any bit-cell. It may be noted that, with our architecture a 6T bit-cell can also be potentially used instead of a 10T. However, the word-line voltage (V_{WL}) should be kept low enough to not disturb the bit-cell when a low pre-charge voltage (V_a) is used during the compute mode. The disadvantage with a low V_{WL} is the increase in the bit-line discharge time. Therefore, a choice can be made between 6T/10T bit-cell designs, based on the area and timing requirements, without compromising on the computation accuracy. On the other hand, in the conventional approach a low V_{WL} decreases the overdrive voltage, leading to more variation in I_{cell} and V_a, as explained before.

The third key feature of our work, which distinguishes it from "near-memory" computing approaches [42, 62], is the use of the inherent bit-line capacitances in the SRAM array to implement the computations. This precludes the need for extra area-intensive capacitors, which would be otherwise required at the SRAM periphery [42] to implement some of the analog computations. Hence, it is an "in-memory"

computing architecture.

Finally, this work supports multi-bit resolution for the inputs and outputs of the dot-products, compared to [41] (output: 1-b) and [62] (both input/output: 1-b). This helps in achieving higher classification accuracy for a neural-network of a given size, as compared to [62] which needs a much larger network to compensate for the 1-b input/output quantization.

All the key features, described above, make our proposed architecture scalable, i.e. muliplte CSRAM arrays can operate in parallel to run larger neural networks.

3.4 Circuits for the 3-Phase Conv-SRAM Operation

3.4.1 Phase-1: DAC

During the first phase of the Conv-SRAM (CSRAM) operation the digital convolution input (X_{IN}) is converted into an analog voltage (V_a) using a column-wise digital-to-analog converter (GBL_DAC). The analog voltage is used to pre-charge the global read bit-line ($GRBL$) and the local bit-lines (not shown in Fig. 3-4) in the SRAM array. Each $GRBL$ is shared by all 16 local arrays and hence, they get the same value of the analog pre-charge voltage. This implements the fact that in a given CNN layer (CONV/FC) each input is processed simultaneously by multiple filters. Furthermore, all the 64 column-wise GBL_DACs operate in parallel and can send a maximum of 64 analog inputs to the CSRAM array in one clock cycle.

Fig. 3-10 shows the schematic of the proposed GBL_DAC circuit. It consists of a cascode PMOS stack biased in the saturation region to act as a constant current source. The $GRBL$ is charged with this fixed current for a time t_{ON}, which is determined by the ON pulse-width. t_{ON} is modulated based on the digital input code ($X_{IN}[5:0]$), using a digital-to-time converter. To achieve a very good linearity of V_a vs X_{IN} or t_{ON} vs X_{IN}, there should be a single continuous ON pulse for every input code, to avoid non-linearities due to multiple charging phases. This is not pos-

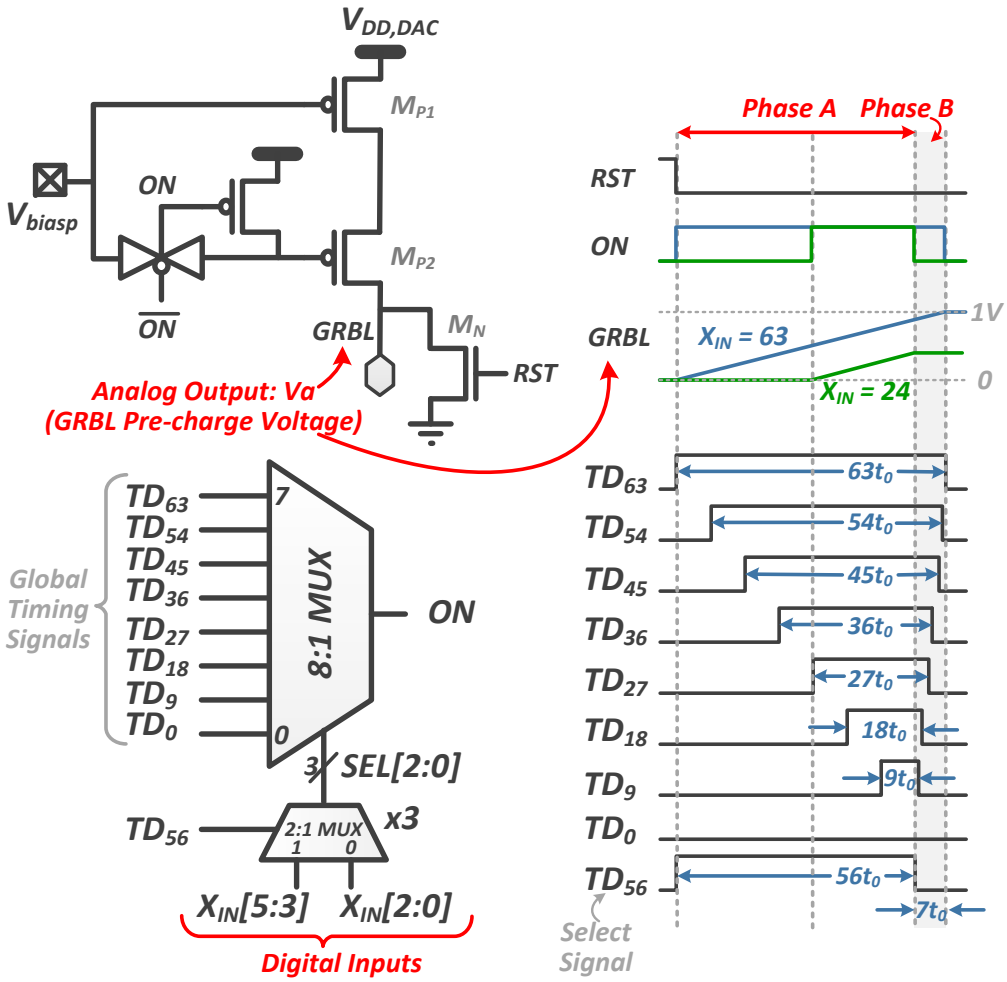


Figure 3-10: Schematic of the column-wise GBL_DAC circuit, showing the digital-to-time converter (bottom-left) and time-to-analog converter (top-left). Also shown are the timing signals and operation waveforms for 2 input codes (right).

sible to generate by simply using 6 timing signals with binary-weighted pulse-widths. However, it may be generated using 2^6 or 64 timing signals and a 64:1 mux. But that would consume a lot of area, which is not ideal for a circuit that needs to be replicated for each column of the SRAM array. To address this issue, we present a 2-phase architecture in which the 3 MSBs of X_{IN} are used to select the ON pulse-width for the first half of charging and the 3 LSBs for the second half. A control signal (TD_{56}) is used to choose between the 2 phases. In this way, an 8-to-1 mux, with 8 timing signals, can be shared during both phases, to reduce the area overhead and the number of timing signals to route. A tree-based architecture, using 2:1 unit

mux's, is used for the 8:1 mux to equalize the mux-delay for different control bits.

To design the pulse-widths of the 8 timing signals, we need to express X_{IN} in terms of its 2 components:

$$\begin{aligned} X_{IN,dec} &= 8 \times k_A + k_B, \\ k_A &= Decimal(X_{IN}[5:3]), \\ k_B &= Decimal(X_{IN}[2:0]) \end{aligned} \tag{3.6}$$

where, k_A and k_B are the decimal values for the 3 MSBs and the 3 LSBs of X_{IN} respectively. Since k_A and k_B can have any integer values from 0 to 7, the pulse-widths of the timing signals (TD's) are chosen as:

$$\begin{aligned} t_{TD_{9k}} &= 8 \times kt_0 + kt_0 = 9 \times kt_0 \\ k &\in (0, 1, .., 7) \end{aligned} \tag{3.7}$$

where, t_0 is the minimum time resolution. A delay-line architecture, with a controllable unit delay of t_0, is used to generate 64 time-delayed signals from the input clock. Then the appropriate signals are combined using NOR gates to generate the TD's. This is done at the global level and the generated TD's are buffered and routed to all the GBL_DACs.

To understand how the 2-phase charging technique works, let us consider two X_{IN} values of 24 and 63, as shown in Fig. 3-10. For $X_{IN} = 24 = 8 \times 3 + 0$, k_A is 3 and k_B is 0. Hence, $TD_{9\times3}$ or TD_{27} is used in phase A and TD_0 is used in phase B, to select the pulse-width of the ON timing signal. Similarly, for the code $X_{IN} = 63 = 8\times7+7$, both k_A and k_B are 7 and hence, TD_{63} is used in both the charging phases.

In addition to the linearity aspect of the DAC transfer function, this architecture also performs better in terms of device mismatch, compared to binary-weighted PMOS charging DACs [41]. This is because, here, the same PMOS stack is used to charge the global bit-line for all input codes, rather than having to use smaller PMOS devices for small input values. Furthermore, the pulse-widths of the globally generated timing signals have less variations typically, compared to those arising from local V_t-mismatch in the PMOS devices [41].

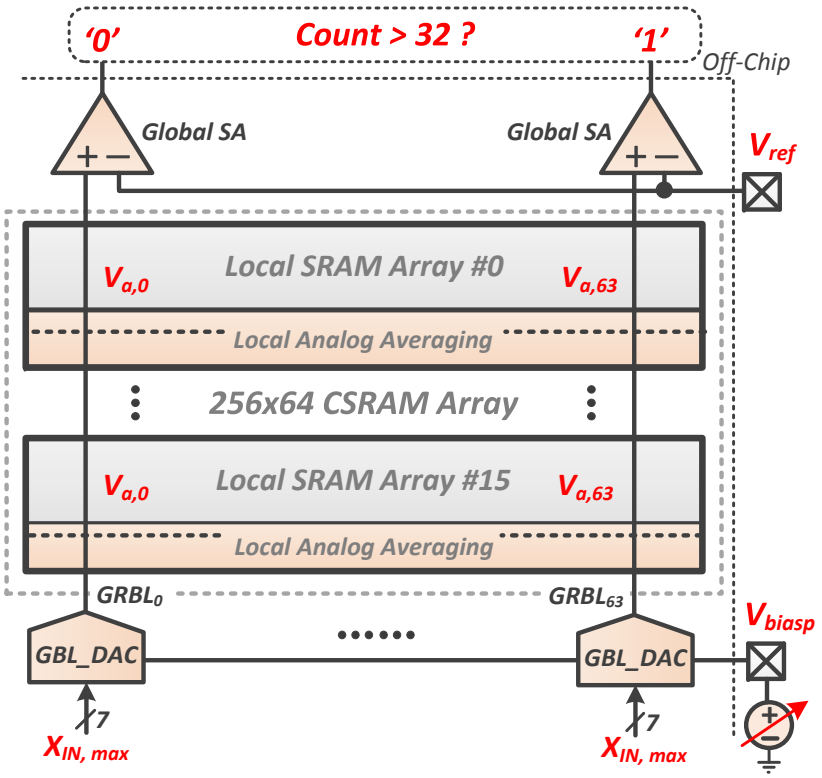


Figure 3-11: Schematic of the CSRAM array during the calibration mode for GBL_DAC.

It should be noted that, a one-time calibration is required to set the maximum value of the analog pre-charge voltage for the maximum input code ($X_{IN,max}$). The maximum pre-charge voltage should be kept lower than the supply voltage of the GBL_DAC, to ensure that the PMOS cascode stack is operating in the saturation region, as a constant current source. For a given t_0, the calibration can be achieved by tuning the externally provided bias voltage (V_{biasp}) of the PMOS stack. Fig. 3-11 shows the set-up of the CSRAM array in the DAC calibration mode. All DACs are fed the same value of the input code, $X_{IN,max}$. In a given clock cycle, first, the GBL_DAC pre-charges the $GRBL$ to an analog voltage (V_a). Then, V_a is compared to an externally provided reference voltage, V_{ref} (fixed at $1V$ in this work). The comparison is done by the column-wise sense-amplifiers (SA), which are already present for normal read-out of the SRAM. All the 64 SAs operate in parallel and use the same V_{ref} to provide 64 comparison outputs simultaneously. V_{biasp} is monotonically increased from $0V$ until majority of the SAs ($> 50\%$) flip their outputs ('1' to '0'), at which point

the calibration is achieved. In this work, a $5mV$ step-size is used to tune V_{biasp}.

3.4.2 Phase-2: Multiply-and-Average

The second phase of the Conv-RAM operation involves the mutliplication of the analog input voltages (V_a's) with the 1-bit filter weights (w_i's) and averaging over N values. This multiply-and-average (MAV) operation is done in parallel for all the 16 local arrays, each storing the w_i's for a different 3-D filter when running a CONV/FC layer.

$$V_{Y\_AVG,k} = \frac{1}{N}\sum_{i=1}^{N} w_{k,i} \times V_{a,i},\ 0 \leq k \leq 15, N \leq 64$$
$$V_{Y\_AVG} = Vp_{AVG} - Vn_{AVG} \tag{3.8}$$

Fig. 3-12 shows the details for the MAV operation for one local array. It starts by turning on the read word-line (RWL) for the selected row in the local array. This leads to discharging of one of the local bit-lines ($LBLT, LBLF$) in each column, depending on the w_i stored in the corresponding 10T bit-cell. A positive w_i ($+1$) is stored as a digital '0' and a negative w_i (-1) as a digital '1'. It should be noted that, the local bit-lines have been pre-charged to the same analog voltage ($V_{a,i}$) as its corresponding global bit-line ($GRBL$) during phase-1. Therefore, at the end of weight evaluation, the difference between the local bit-line voltages represents the product of the analog voltage ($V_{a,i}$) and the 1-bit weight (w_i). For example, the bit-cell in the '0^{th}' column stores a -1 and hence, $\Delta V_{LBL,0} = V_{LBLT,0} - V_{LBLF,0} = -V_{a,0}$.

The weight multiplication/evaluation step is completed by turning off the RWL. After that, the appropriate local bit-lines are shorted together horizontally to evaluate the average. The positive and negative parts of the average as obtained on two separate voltage rails, Vp_{AVG} and Vn_{AVG}, respectively. This is implemented by the local MAV_a circuits, which pass the voltages of the $LBLT$'s and $LBLF$'s to either the Vp_{AVG} or the Vn_{AVG} voltage rails, depending on the sign of the input X_{IN}. If the input for the particular column is positive ($X_{IN} > 0$) EN_P is turned on, otherwise EN_N is on. EN_P and EN_N are digital control signals which are globally routed and

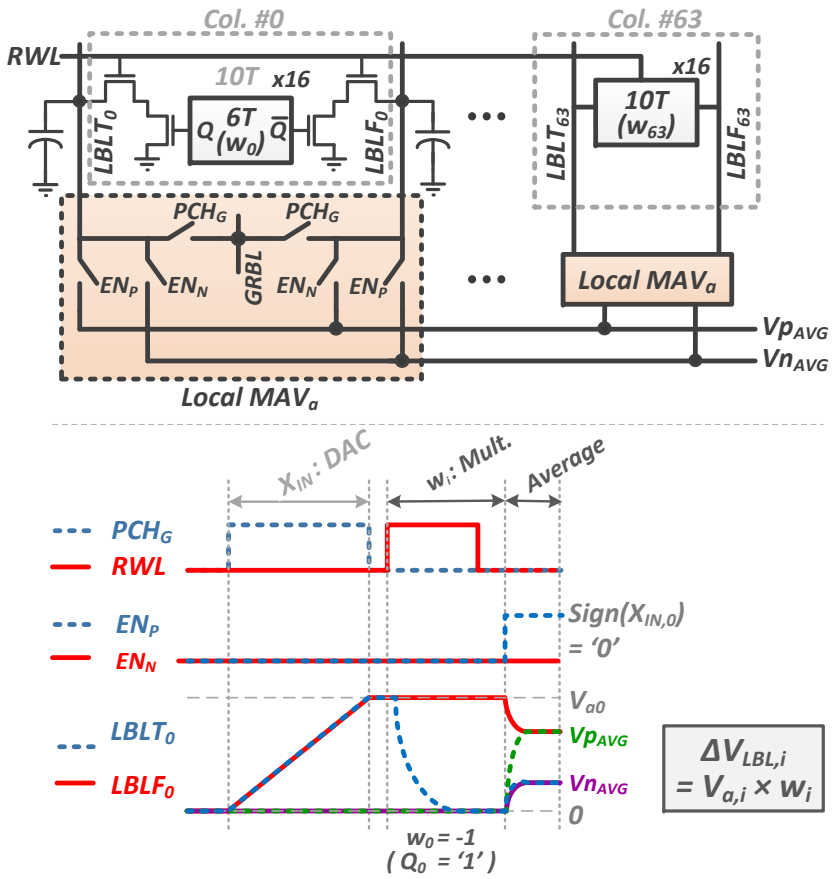


Figure 3-12: Architecture of a 16×64 local array of the Conv-RAM, showing the 10T bit-cells storing the filter weights and local analog multiply-and-average (MAV_a) circuits. Also shown are typical operation waveforms (bottom) for one column.

shared column-wise by all the 16 local arrays.

The fully-differential nature of the averaging architecture helps in mitigating many common-mode noise issues, e.g. clock coupling noise from the control switches, capacitance variation of the local bit-lines and the voltage rails due to different process corners, etc. This helps in improving the accuracy of the dot-product computations with our approach.

It should be also noted that during this phase, when the SRAM bit-cell is actually used for weight-evaluation, the time required does not have a large variation. Fig. 3-13 shows the simulated local bit-line discharge time ($t_{dis,LBL}$) in the slowest process corner (SS). As seen from the figure, even the 6σ value of $t_{dis,LBL}$ is merely 500ps, which is much smaller than the total clock period ($\approx 100ns$). This shows that bit-cell

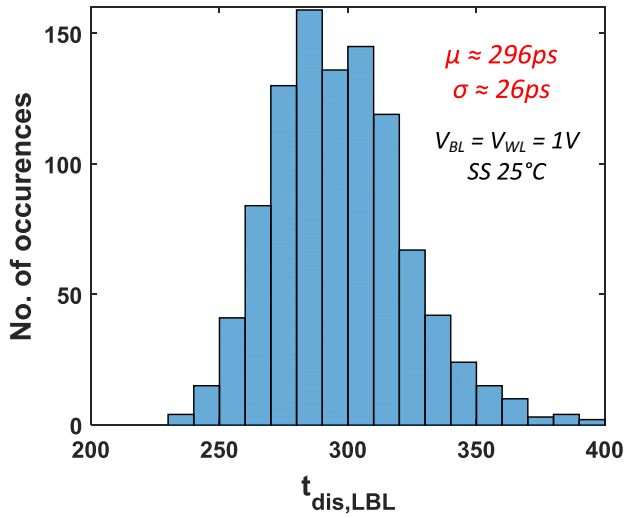


Figure 3-13: Variation of the local bit-line discharge time for weight evaluation/multiplication in phase-2.

V_t variations do not dominate the overall computation time. The longer clock period is facilitated by the highly parallel processing in the compute mode.

3.4.3 Phase-3: ADC

The third and last phase of the Conv-RAM operation is the analog-to-digital conversion of the dot-product outputs, with multi-bit resolution. The difference of the analog average voltages (Vp_{AVG} and Vn_{AVG}) is fed to an ADC to get the digital value of the computation (Y_{OUT}). This is done in parallel for all the 16 local arrays, producing outputs corresponding to 16 different filters simultaneously.

Choosing the ADC architecture is crucial since it would be replicated mulitple times in the CSRAM array. Hence, area and power consumption are key metrics to consider. In addition, the typical distribution of the ADC outputs (Y_{OUT}'s) should also be considered to find the more appropriate architecture. As seen from simulation results in Fig. 3-14, for a typical CONV layer with a full scale input range of ± 31, Y_{OUT} has an absolute mean value of ± 1.3 and is typically limited to ± 7. Hence, a serial integrating ADC architecture is more apt in this scenario, compared to other area-intensive (e.g. SAR) and more power-hungry ones (e.g. flash). In spite of its serial nature, in most cases we can expect the ADC to finish its operation within a

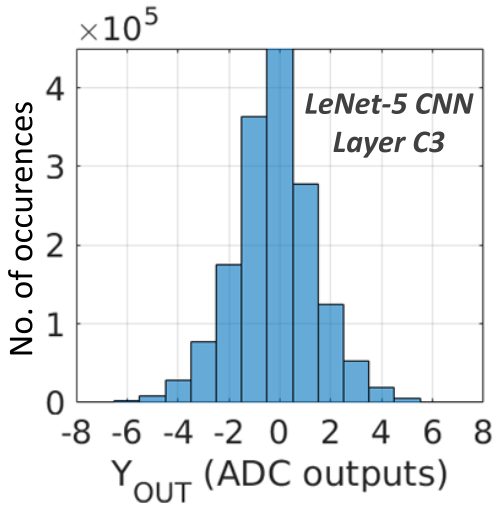


Figure 3-14: Simulated distribution of the partial convolution output from the ADC (Y_{OUT}), for a typical CONV layer (C3) in the LeNet-5 CNN.

few cycles, due to the particular Y_{OUT} distribution.

Fig. 3-15 shows the architecture of the proposed integrating ADC (CSH_ADC). It consists of 3 main parts: a charge-sharing based integrator, a sense-amplifier (SA) and a logic block. Capacitive charge-sharing with replica bit-lines is used to implement the integration. Using replica bit-lines help to track the local bit-line capacitance better in the presence of process and temperature variations. The SA has a standard strongARM latch-type architecture [64]. PMOS devices are chosen for the SA's input differential pair, since the common mode voltage of Vp_{AVG} and Vn_{AVG} signals are expected to be closer to the GND rail. The logic block provides the timing signals for the charge-sharing (PCH_R, EQ_P, EQ_N) and the SA comparison (SA_EN), using the globally provided timing signals (ϕ_1, ϕ_2). It also has a counter to count the number of cycles it takes to finish the ADC operation and that provides the digital output of the dot-product computation.

Fig. 3-15 also shows the waveforms for a typical CSH_ADC operation. It starts by sending a SA_EN pulse from the ADC logic block to the SA. The SA compares Vp_{AVG} and Vn_{AVG}, and sends its outputs (SAO_P, SAO_N) to the ADC logic block. The first comparison determines the sign of the output, e.g. for the case shown in Fig. 3-15, Y_{OUT} is positive since Vp_{AVG} is higher than Vn_{AVG}. After the first comparison, the

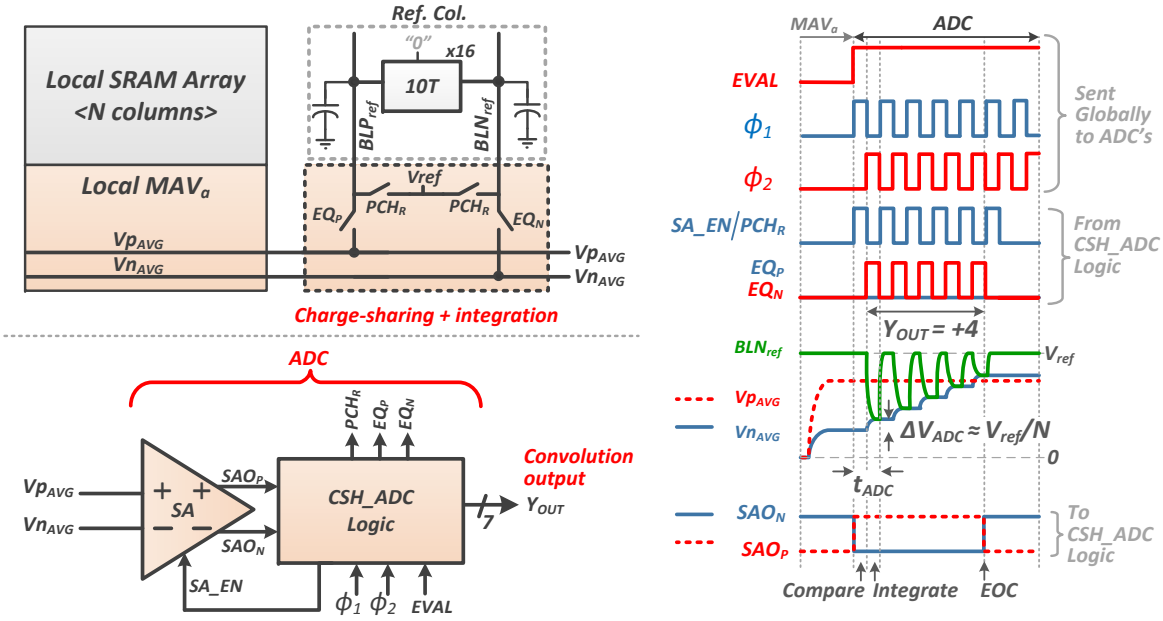


Figure 3-15: Architecture for the charge-sharing based ADC (CSH_ADC) for 1 local array of the Conv-RAM and typical waveforms for the digital output (Y_{OUT}) computation for the convolution (dot-product) operation.

lower of the 2 voltage rails (Vn_{AVG}) is integrated by charge-sharing it with a reference local bit-line (BLN_{ref}), using the equalize signal (EQ_N in this case). The reference bit-line, which replicates the local bit-line capacitance, was pre-charged during the SA comparison using the PCH_R signal to V_{ref} (= 1V in this work). Therefore, the step-size of the integration is $\approx \frac{V_{ref}}{N}$, where N is the number of SRAM local columns that were averaged. The pre-charge and equalize/integrate operations, along with the SA comparison, continue until the lower voltage rail (Vn_{AVG}) exceeds the higher one (Vp_{AVG}). When this happens, the SA outputs flip indicating the end-of-conversion (EOC). After this, no more timing pulses are generated. A counter in the ADC logic block counts the number of equalize pulses (EQ_N) it takes to reach EOC and that generates the digital value of the convolution/dot-product output (Y_{OUT}), which is +4 for the example shown in Fig. 3-15.

Fig. 3-16 shows the detailed schematic of the logic block in the CSH_ADC. The logic block uses the globally provided 2-phase ADC timing signals: ϕ_1 and ϕ_2 (generated from an on-chip free running VCO). ϕ_1 is used to generate the sense-amplifier

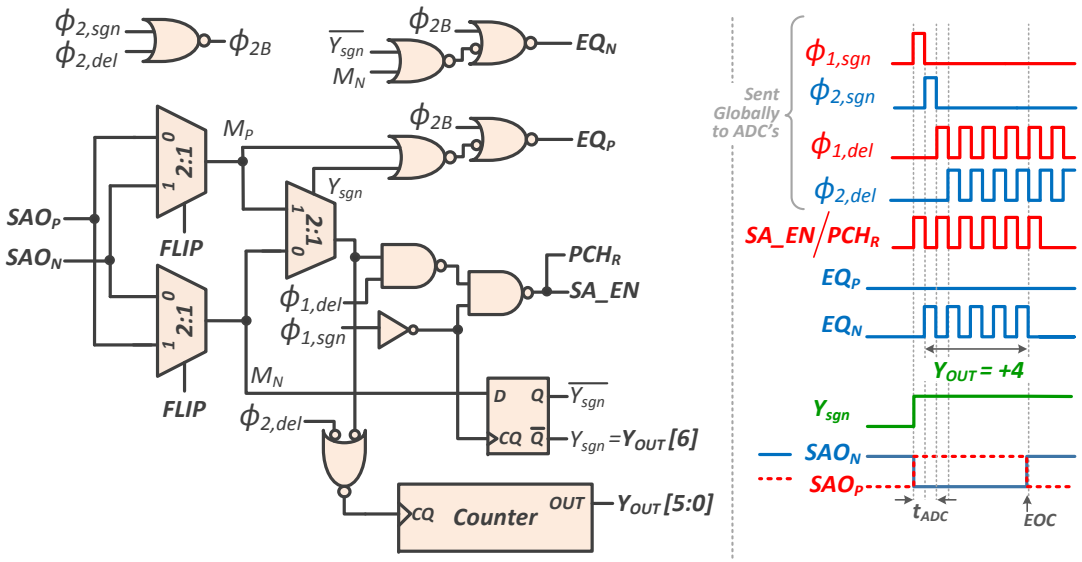


Figure 3-16: Schematic of the CSH_ADC logic block, to generate the timing signals for the ADC operation, using the globally provided signals: $\phi_{1,sgn}, \phi_{1,del}, \phi_{2,sgn}, \phi_{2,del}$. Typical waveforms are shown on the right, corresponding to the example in Fig. 3-15, with FLIP = '0'

enable (SA_EN) and pre-charge (PCH_R) signals. Whereas, ϕ_2 is used to generate the equalize signals (EQ_P, EQ_N) for integration. Each of ϕ_1 and ϕ_2 consists of a set of 2 signals: one to denote the first pulse ($\phi_{1,sgn}, \phi_{2,sgn}$) and the other one is for the rest of the pulses ($\phi_{1,del}, \phi_{2,del}$). $\phi_{1,sgn}$ is always needed in the ADC for the first SA comparison, to determine the sign of the output. And, $\phi_{2,sgn}$ is always needed for the first integrate operation, to determine if the magnitude of the ADC output is zero. Hence, $\phi_{1,sgn}$ and $\phi_{2,sgn}$ are sent separately, without having to be generated locally for each individual ADC. $\phi_{1,del}$ and $\phi_{2,del}$ signals are used for generating the timing signals, after the first ADC cycle. The $FLIP$ signal is used to flip the outputs from the SA during offset-cancellation, as explained below. Fig. 3-16 also shows the sample waveforms for the example discussed in Fig. 3-15, with $FLIP$ = '0'.

It should be noted that Y_{OUT} is directly affected by the SA offset voltage (V_{OS}), which can degrade the overall computation accuracy due to incorrect ADC outputs. To address this issue, we propose a simple 2-cycle offset-cancellation technique, using a flipping mux at the input of the SA (Fig. 3-17). During the first/even cycle of this 2-cycle period, $FLIP$ = '0'. Hence, Vp_{AVG} and Vn_{AVG} are passed to the

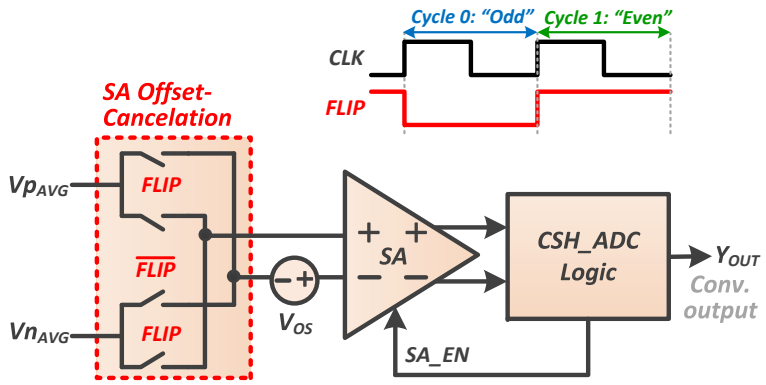


Figure 3-17: Circuit for the 2-cycle offset-cancellation technique for the SA in CSH_ADC.

positive and negative input terminals of the SA respectively. Therefore, $Y_{OUT,0} = f_{ADC}(Vy_{AVG,0} - V_{OS})$. The output in this cycle is exactly same as in the conventional case (without the flipping mux). However, during the next odd cycle, the inputs to the SA are flipped by setting $FLIP =$ '1'. Hence, a differential voltage of $(Vn_{AVG} - Vp_{AVG}) = -Vy_{AVG}$ is applied to the input of the SA. To get the correct polarity at the output of the ADC, another negation is applied by the ADC logic block. This results in $Y_{OUT,1} = -f_{ADC}(-Vy_{AVG,1} - V_{OS}) = f_{ADC}(Vy_{AVG,1} + V_{OS})$, as compared to $Y_{OUT,1} = f_{ADC}(Vy_{AVG,1} - V_{OS})$ for the conventional case. Finally, we add the Y_{OUT}'s for the 2 consecutive cycles to accumulate the partial results for the convolution. With our proposed 2-cycle approach the effect of V_{OS} is inherently canceled since:

$$\begin{aligned} Y_{OUT} &= Y_{OUT,0} + Y_{OUT,1} \\ &= f_{ADC}(Vy_{AVG,0} + Vy_{AVG,1}) \end{aligned} \tag{3.9}$$

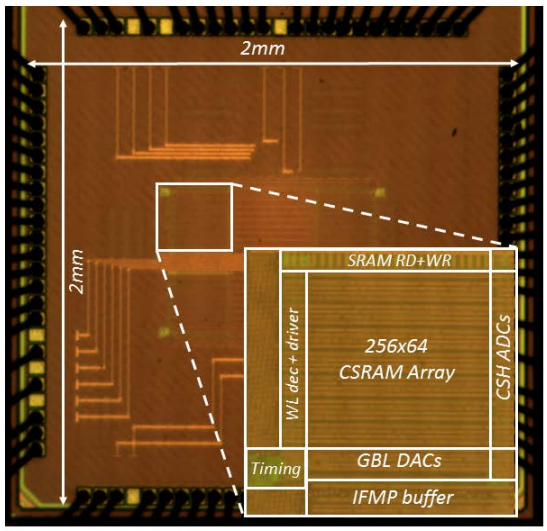
On the other hand, for the conventional case the effect of V_{OS} adds up since:

$$Y_{OUT} = f_{ADC}(Vy_{AVG,0} + Vy_{AVG,1} - 2 \times V_{OS}) \tag{3.10}$$

and this makes the accumulation result further inaccurate. It should be noted that, the benefits of this offset-cancellation technique comes without any extra timing and power penalty, as long as an even number of cycles are required to finish a full convolution computation. This can be easily expected for most CNNs.

3.5 Measured Results

The 16Kb CSRAM array was implemented in a 65nm LP CMOS process. The die photo in Fig. 3-18 shows the relative area occupied by the different key blocks. The bit-cell array (along with its peripheral circuitry) occupies 73.1% of the total CSRAM area, 8.2% is occupied by the GBL_DACs, 8.6% by the local MAV_a circuits, 7.3% by the CSH_ADCs and the rest by global timing circuits. The test-chip summary is also shown in Fig. 3-18.



Technology	65nm
CSRAM size	16Kb
CSRAM area	0.063mm^2
Array organization	256×64 (10T bit-cells)
# column DAC's	64
# row ADC's	16
Max. # MAV's	64×16
Supply voltages	1V (main), 1.2V (DAC), 0.8V (array)
Main clock freq. (compute mode)	5MHz
ADC clock freq.	250MHz

Figure 3-18: Die photo and summary of the Conv-RAM test-chip fabricated in a 65nm CMOS process.

Fig. 3-19 shows the measured waveforms for the critical signals, denoting the completion of different key steps during the convolution computation. CLK_main is the main input clock of the system, PCH_done denotes the completion of the

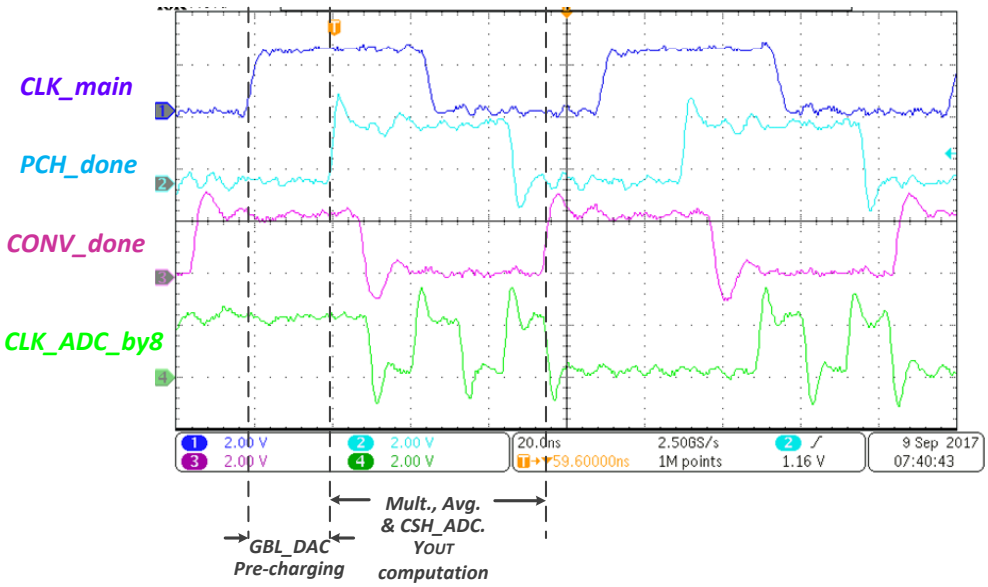


Figure 3-19: Measured oscilloscope waveforms of critical signals, for t_{clk} = 90ns.

GBL_DAC pre-charge phase, and $CONV\_done$ denotes the end of the CSH_ADC operation i.e. end of convolution computation. Finally, the clock signal for the ADC (frequency divided by 8) is CLK_ADC_by8. The ADC clock is generated from an on-chip VCO, which is activated when needed during the ADC operation.

3.5.1 Circuit Characterizations

Fig. 3-20 shows the measured transfer function for the GBL_DAC, which is used in the 5-b mode by setting the LSB of X_{IN} to '0' (the sign bit of X_{IN} does not affect the DAC analog output voltage). To estimate the DAC analog output voltage (V_a), V_{GRBL} for the 64 columns of the CSRAM are compared to an external voltage (V_{ref}) by column-wise SAs, as explained before for DAC calibration (Fig. 3-11). For each X_{IN}, the V_{ref} at which more than 50% of the SA outputs flip is chosen as an average estimate of V_a. As mentioned before, an initial one-time calibration is needed to set $V_{a,max} = 1V$ for $X_{IN} = 31$ (max. input code). The supply voltage for the DACs is fixed at 1.2V, to keep the PMOS stack in them operating in the saturation region (as a constant current source). It can be seen from Fig. 3-20 that, there is a good linearity in the DAC transfer function with $DNL < 1LSB$. Since the SAs

have NMOS input-pair, low values of V_a cannot be properly estimated. Hence, the characterization is done down to X_{IN} =16 (or $V_a \approx 500mV$).

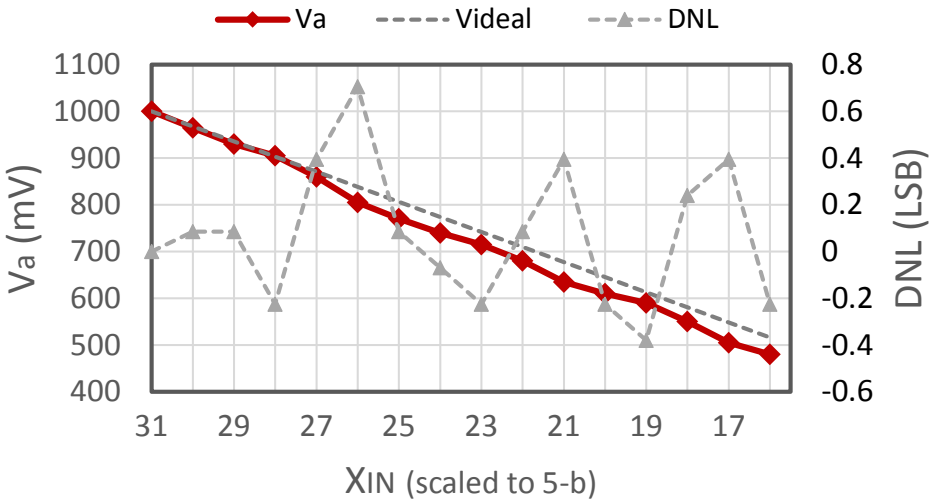


Figure 3-20: Measured transfer function of GBL_DAC at $V_{dd,DAC}$ = 1.2V, with V_{ref} = 1V and $t_0 \approx$ 250ps.

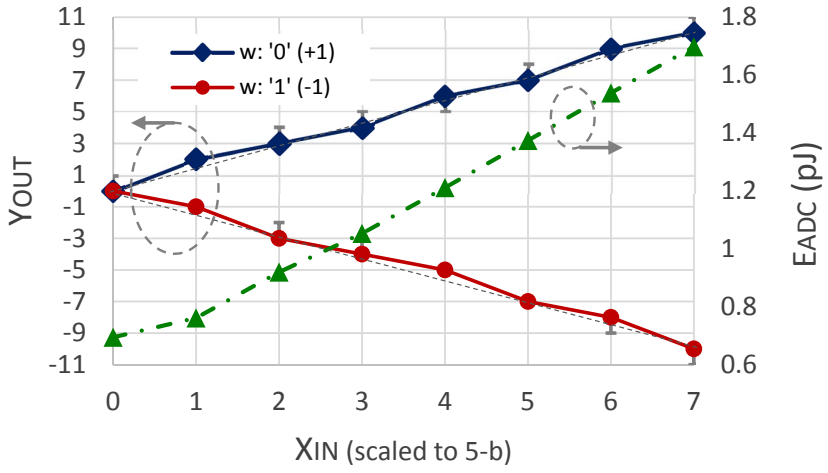


Figure 3-21: Measured transfer function and energy consumption of CSH_ADC at $V_{dd,ADC}$ = 1V, $V_{dd,ARY}$ = 0.8V and f_{ADC} = 250MHz.

Fig. 3-21 shows the transfer function of the CSH_ADC, operating at 1V and a clock frequency of 250 MHz (generated on-chip with a free-running VCO). The array voltage is kept at 0.8V to reduce the clock-coupling noise from the WL's, when reading the weights. To characterize the CSH_ADCs, all X_{IN}'s are fed the same input code, all w_i's are written the same value and then the ADC outputs (Y_{OUT} 's) are observed.

The measurement results show a good linearity in the overall transfer function and low variation in the Y_{OUT} values, which is due to the fact that the variation in BL capacitance (used in CSH_ADC) is much lower than transistor V_t-variation. It can be also seen from Fig. 3-21 that the energy/ADC scales linearly with the input/output value, which is expected for the integrating ADC topology.

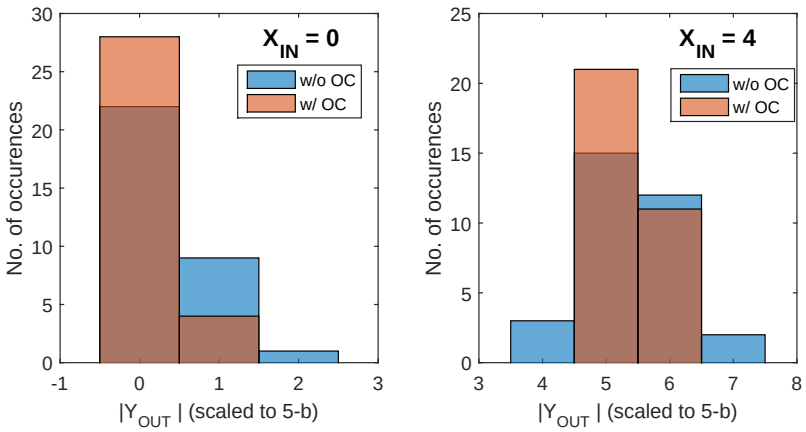


Figure 3-22: Measured distribution of convolution output values (Y_{OUT}) from CSH_ADC with and without the offset-cancellation (OC) technique, for two values of the input code (X_{IN}).

The effect of the offset-cancellation (OC) technique for the SA (in the CSH_ADC) is also characterized, as shown in Fig. 3-22 for two different input codes. It can be clearly seen that, the OC helps in reducing the variation of the Y_{OUT} values, leading to a better computation accuracy for the dot-products/convolutions.

3.5.2 Test Case: MNIST Dataset

To demonstrate the functionality for a real CNN architecture, the MNIST hand-written digit recognition dataset is used with the LeNet-5 CNN. As illustrated in Fig. 3-23, LeNet-5 consist of 2 CONV layers (C1, C3) and 2 FC layers (F5, F6), along with the non-linear max-pooling (S2, S4) layers. The ReLU layer (R5 after layer F5) is not shown in the figure for simplicity. Only the CONV/FC layers, which involve majority of the computations, are implemented on-chip by the CSRAM array. The non-linear layers are implemented in software (described later). Table.3.1 shows the

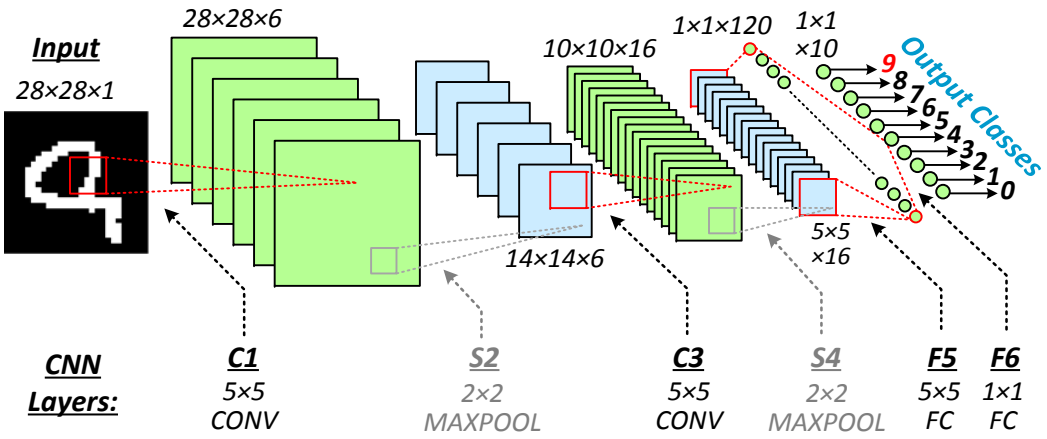


Figure 3-23: Architecture of the LeNet-5 CNN, showing the sizes of the feature maps (top) and the filters (bottom).

detailed mapping of the 4 CONV/FC layers to the CSRAM array to compute the convolutions. Let us first consider layer C3. It has a filter size of 5×5, with 6 input channels and 16 output channels (number of 3-D filters). Each of the 16 3-D filters are mapped to the one of the 16 local arrays in the CSRAM. Since each row in the local array has 64 bit-cells, hence, a maximum of 2 ($= \lfloor \frac{64}{5\times5} \rfloor$) input channels can fit per row. Therefore, 3 ($= \frac{6}{2}$) rows are required in each local array to fit the entire 3-D filter. In every clock cycle, 50 ($= 5 \times 5 \times 2$) X_{IN}'s are sent through a buffer (shift-registers) to the CSRAM array to compute 16 partial convolution outputs. Thus, the CSRAM array processes $50 \times 16 \times 2$ operations (1 MAV = 2 OPs: 1 multiply + 1 add/average), per clock cycle. For layer F5, the entire filter cannot be fit at once in the CSRAM array (due to its limited 16Kb size in the test-chip). Hence, the entire process, explained above, is repeated mulitple times to finish all the computations. However, having multiple CSRAM arrays operating in parallel, can easily alleviate this problem, by fitting all the filter weights together on-chip.

Fig. 3-24 shows the measured error rate on the 10K MNIST test-dataset, with each CONV/FC layers being successively implemented on-chip. 3 different chips are measured, each experiment is repeated mulitple times, and the average value of the error rate is reported. For this experiment, $V_{dd,DAC}$ is set to 1.2V, $V_{dd,ARY}$ is set to 0.8V and the rest of the circuits are operating from a 1V supply, with a 200ns clock period.

Table 3.1: Parameter mapping for the CONV/FC layers of LeNet-5 CNN to the CSRAM array

Parameters (↓)	**C1**	**C3**	**F5**	**F6**
3-D Filter size	5×5×1	5×5×6	5×5×16	1×1×120
# 3-D Filters	6	16	120	10
# Local ARYs used	6	16	15*	10
# IFMP channels/row	1	2	2	30 (15)
Rows, Col.s/local ARY	1, 25	3, 50	8, 50	4 (8), 30 (15)
# col.s for AVG (N)	32	50	50	32 (16)
# operations/cycle	25×6×2	50×16×2	50×15×2	30 (15)×10×2

* Repeated 8 times to cover all the 120 filters

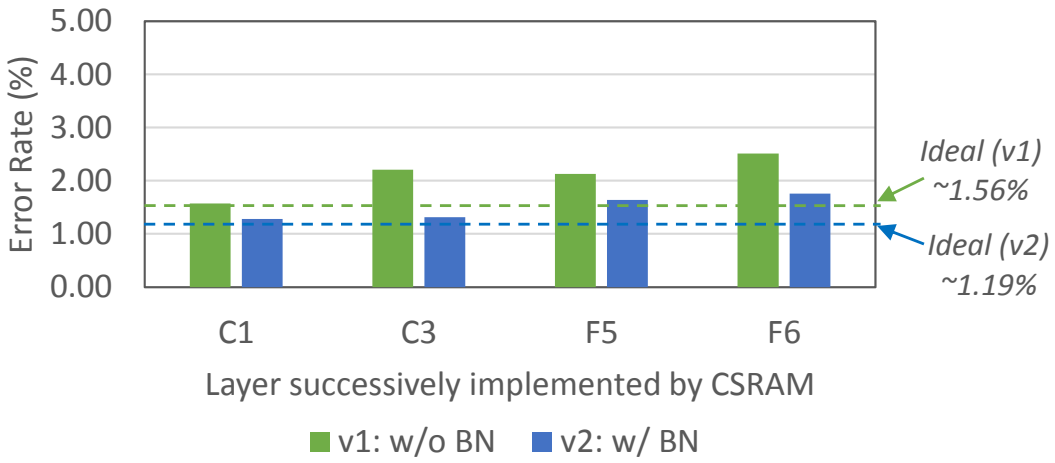


Figure 3-24: Measured error rate for the 10K test images in the MNIST dataset using LeNet-5 CNN, with and without batch-normalization, at $V_{dd} = V_{a,max} = 1$V.

We tested 2 different versions of LeNet-5: with and without Batch-Normalization (BN) layers preceeding the CONV/FC layers. Without BN layers ('v1') we achieve a classification error rate of 2.5% after all the 4 layers. The error rate is improved to 1.7% by using the BN layers ('v2'). This is mostly because BN normalize the convolution inputs for every layer, with a mean around '0' and also limits the maximum value of the inputs. Hence, after input quantization to 6-b, its features are better preserved compared to an un-normalized input distribution. The measured error rate, which is close the expected value from an ideal digital implementation, shows the robustness of the CSRAM architecture to compute convolutions. The error rate for the MNIST

dataset is improved by 8.3% compared to prior work on in/near-memory compute [41, 62], where a 10% error rate was achieved. Next, we tested functionality at a lower voltage setting of $V_{dd,DAC} = 1$V and the rest of the circuits operating at 0.8V, with a clock period of 400ns. The maximum DAC pre-charge voltage ($V_{a,max}$), corresponding to the maximum input code, is calibrated to 0.8V. Hence, the magnitude of 1LSB is $\sim 26mV$ (instead of 32mV for the previous case with $V_{a,max} = 1$V). Fig. 3-25 shows the measured error rate for this set of voltages. Due to reduced analog voltage precision, the error rates are slightly higher, with 'v1' achieving 3.4% and 'v2' achieving 1.9% for the 10K test dataset of MNIST.

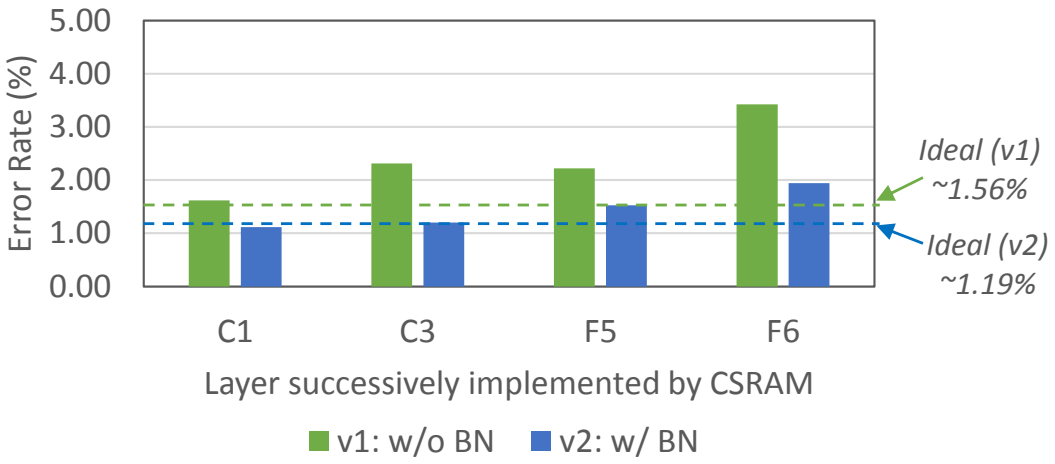


Figure 3-25: Measured error rate for the 10K test images in the MNIST dataset using LeNet-5 CNN, with and without batch-normalization, at $V_{dd} = V_{a,max} = 0.8$V.

Fig. 3-26 shows the overall energy consumption of the CSRAM array for running the different layers of LeNet-5, with $V_{dd} = 1$V, $f_{clk} = 5$MHz. Of the 4 CONV/FC layers in LeNet-5, the energy while running layers C1 and F6 are lower than layers C3 and F5. This is because layers C1 and F6 do not fully utilize the entire CSRAM array, due to their small filter sizes. However, that also translates to a lower energy-efficiency for them (table 3.2), since the energy is amortized over fewer MAV operations. A higher array utilization in layer C3 (all 16 local arrays) helps in achieving an energy-efficiency of 38.8TOPS/W ('v1') by consuming 41.3pJ of energy for $50 \times 16 \times 2 = 1600$ operations. Whereas, for 'v2' (with BN), layer F5 achieves the best energy-efficiency of 40.3TOPS/W, utilizing 15 of the 16 local arrays. Fig. 3-26 also shows the energy

breakdown for the 3 major components: GBL_DAC, array+MAV_a and CSH_ADC. The energy for GBL_DAC is limited by the bit-precision requirement for representing the IFMP/OFMP values. Whereas, the energy for the array, MAV_a and CSH_ADC circuits can be scaled down by scaling their supply voltages while sacrificing speed. Fig. 3-27 shows the measured energy consumption of the CSRAM array, at $V_{dd,DAC}$ = 1V, V_{dd} = 0.8V and f_{clk} = 2.5MHz. The reduced supply voltages help in decreasing the energy consumption, leading to better energy-efficiency numbers (table 3.3).

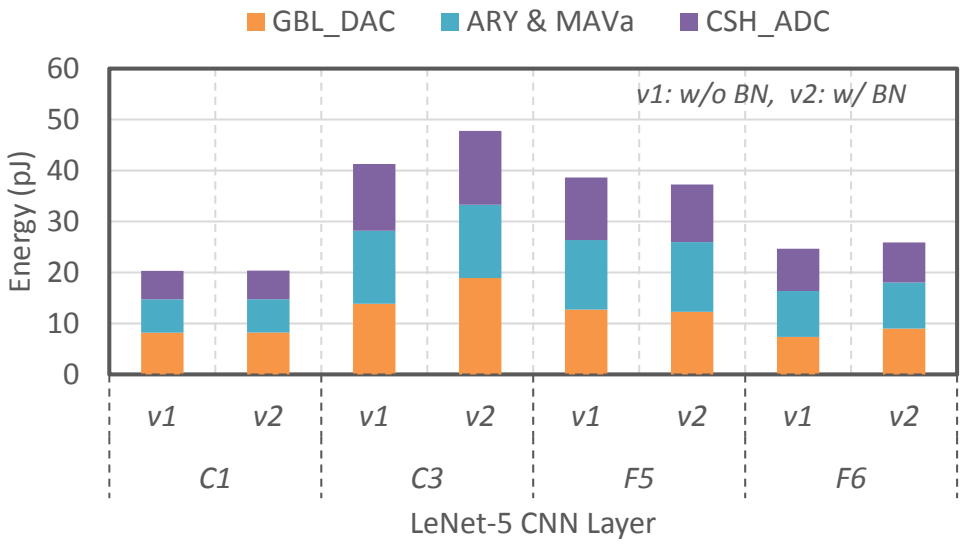


Figure 3-26: Measured energy consumption of the CSRAM array when running the 4 different CONV/FC layers of the LeNet-5 CNN, at V_{dd} = 1V, $V_{dd,DAC}$ = 1.2V, $V_{dd,ARY}$ = 0.8V and $f_{clk,main}$ = 5MHz.

Table 3.2: Measured energy-efficiency* (TOPS/W) for the CONV/FC layers of LeNet-5 CNN, at V_{dd} = 1V

Type (↓)	**C1**	**C3**	**F5**	**F6**
v1: without BN	14.8	38.8	38.8	24.3
v2: with BN	14.7	33.5	40.3	23.2

* 1 MAV = 1 multiply + 1 average = 2 operations

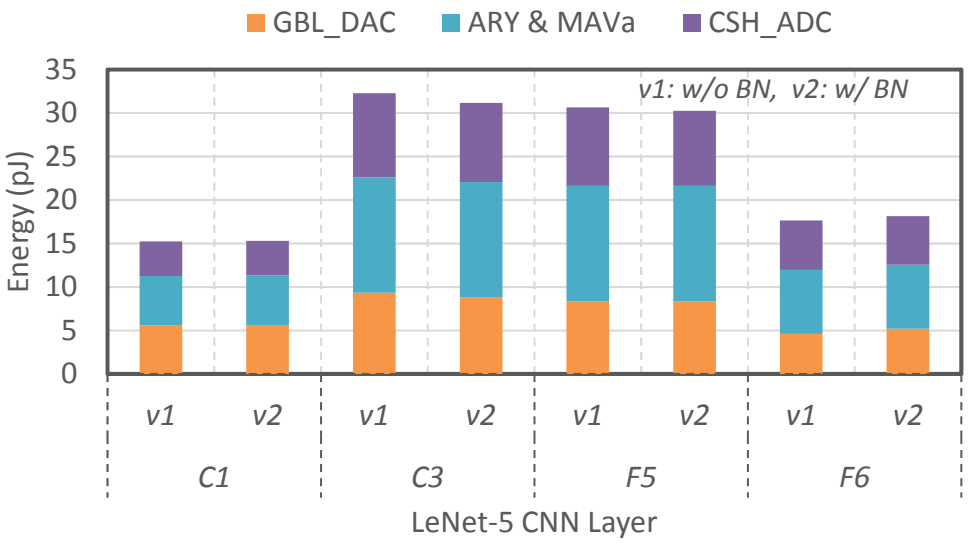


Figure 3-27: Measured energy consumption of the CSRAM array when running the 4 different CONV/FC layers of the LeNet-5 CNN, at $V_{dd} = 0.8$V, $V_{dd,DAC} = 1$V and $f_{clk,main} = 2.5$MHz.

Table 3.3: Measured energy-efficiency* (TOPS/W) for the CONV/FC layers of LeNet-5 CNN, at $V_{dd} = 0.8$V

Type (↓)	**C1**	**C3**	**F5**	**F6**
v1: without BN	19.7	49.6	48.9	17.0
v2: with BN	19.6	51.3	49.6	16.5

* 1 MAV = 1 multiply + 1 average = 2 operations

The measured distributions of the 6-b partial convolution outputs from the ADC (Y_{OUT}'s) are shown in Fig. 3-28, for all the 4 CONV/FC layers. For each of these layers, Y_{OUT} has a mean around $\approx 1LSB$ and is typically limited to $\pm 8LSBs$ (for a full-scale of ± 31), which justify the use of the serial ADC topology to compute it. Fig. 3-29 shows the measured distributions of the convolution inputs (X_{IN}'s) for the 4 layers. X_{IN}'s have been properly scaled and quantized to 6-b (including sign bit) before being sent to the CSRAM array to compute the convolutions. As seen from the figure, all the layers have a high proportion of 0's with low (absolute) mean values for the X_{IN}'s. This helps in reducing the GBL_DAC energy to convert and send them to the columns of the CSRAM array.

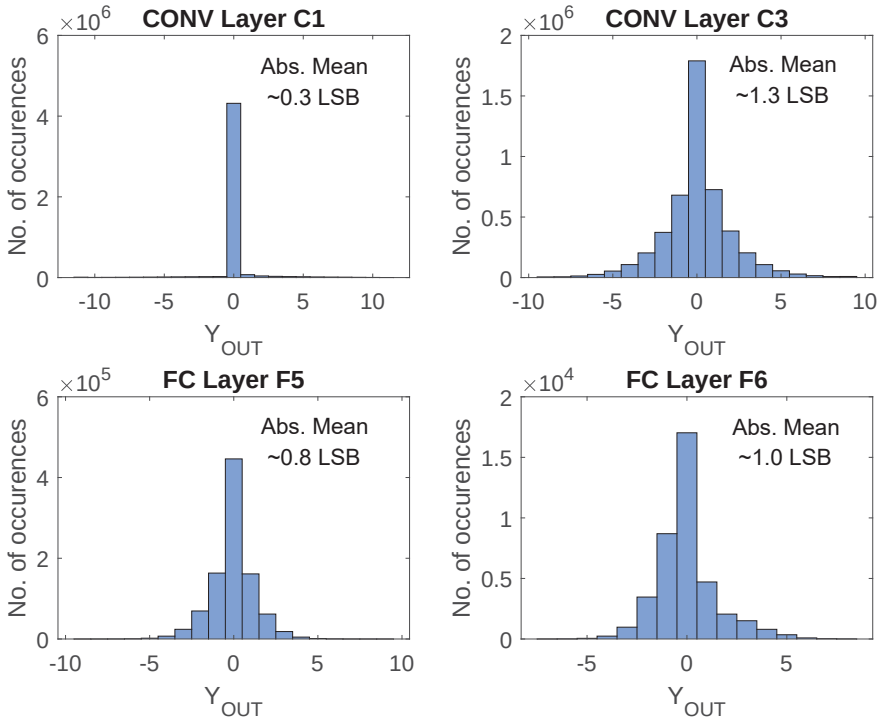


Figure 3-28: Measured distribution of the partial convolution outputs (Y_{OUT}'s) for the 4 different CONV/FC layers of the LeNet-5 CNN (V_{dd} = 1V).

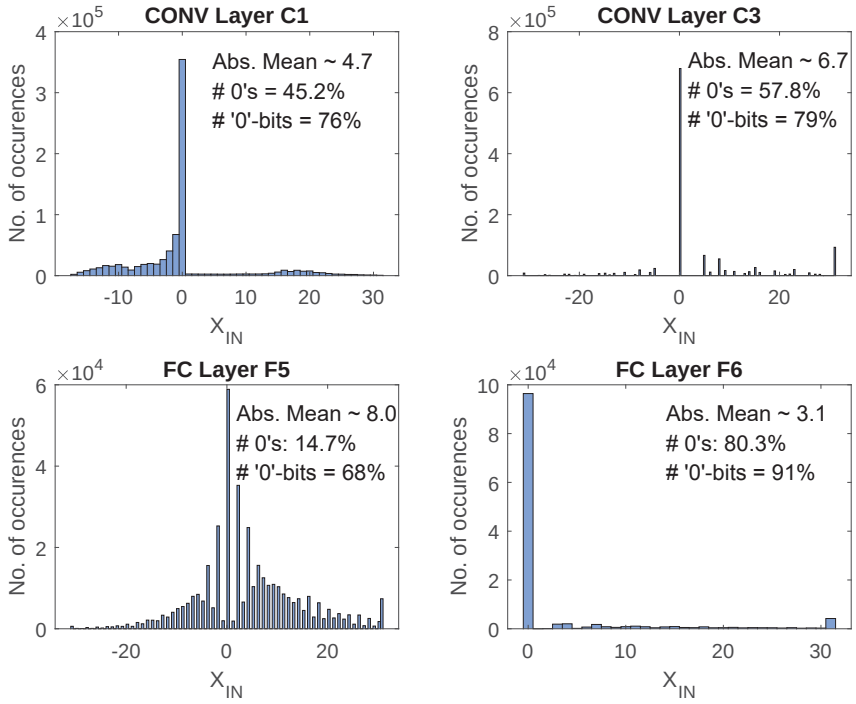


Figure 3-29: Measured distribution of the 6-b convolution inputs (X_{IN}'s) for the 4 different CONV/FC layers of the LeNet-5 CNN (V_{dd} = 1V).

Table 3.4: Comparison with prior work on low bit-width hardware implementations of ML algorithms

Metric	**This work [60]**	ISSCC'17 [65]	JSSC'17 [66]	JSSC'18 [42]	JSSC'18 [62]	CICC'18 [67]	ISSCC'18 [68]
Tech. (nm)	65	28	40	65	65	28	65
Voltage (V)	0.8, 1	0.715	0.8	1	0.55	0.66	0.65
Computation Mode	In-Memory, mixed-signal	Digital	Digital	In-Memory, mixed-signal	Near-Memory, digital	Digital	Digital
ML Algo.	CNN	FC-DNN (4-layer)	CNN	k-NN[1]	FC-DNN (12-layer)	CNN	CNN
ML Dataset	MNIST	MNIST	MNIST	MNIST	MNIST	MNIST	FER2013
# MAC(V)'s/ classification	406.8K	334.3K	406.8K	16.4K	768.1K	20M	-
Classification Accuracy	98.3% (1V) 98% (0.8V)	98.36%	98%	92%[1]	90.1%	97.4%	-
# bits for IFMP/OFMP	6	8	6	8	2	1	16
# bits for Weights	1	8	4	8	1	1	1
SRAM Size (KB)	2	1024	144	16	102.1	328	256
Peak Throughput (GOPS)[2]	8 (1V) 4 (0.8V)	10.7	102	10.2	380.2	90	368.6
Peak Energy Efficiency (TOPS/W)	40.3[3] (1V), 51.3[3] (0.8V)	1.86 (0.345)[4]	1.75 (0.663)[4]	1.94	6.0	230 (42.68)[4]	50.6

[1] k-NN: k-nearest neighbor, only 4-output classes (out of 10) were demonstrated with 100 test images
[2] We assume 2 operations (OPs) for 1 MAV (1 mult. + 1 avg.), similar to a MAC (1 mult. + 1 acc.)
[3] Does not include energy to access IFMP/OFMP memories
[4] Assuming a 65nm implementation and Energy $\propto$ (Tech.)^2

Recent hardware implementations [66, 65, 62, 69, 68, 67] for NNs have focused on reduced bit-precisions to achieve higher energy-efficiency. Table 3.4 presents comparison with prior work, both conventional digital [65, 66, 68, 67] and in-memory approaches [42, 62]. It should be noted that, while [65, 66, 62, 67, 68] are full systems, the main focus of this work was to demonstrate in-memory computation capability for

CNNs. Hence, ours does not include the energy for IFMP/OFMP memories. However, for the MNIST dataset with LeNet-5 CNN, we estimate (Appendix B) those to have only small contributions to the overall energy-efficiency per MAV operation, due to the high parallelism supported by our in-memory approach. Furthermore, as seen from Fig.s 3-29 and 3-28, both the inputs (X_{IN}'s) and the partial outputs (Y_{OUT}'s) have a high proportion of '0's. Hence, in future work, data-dependent memory architectures e.g. 8T SRAMs, [54, 39] can be used to store and access the inputs/outputs. [54, 39] take advantage of data properties to significantly reduce memory-access energy, which would be highly useful here. Compared to [65, 66], we achieve $> 27\times$ improvement in energy-efficiency, due to the massively parallel in-memory analog computations. Our work achieves similar energy-efficiency numbers as [67] (considering a simplified technology scaling model), while using 6-b for IFMP/OFMP, compared to 1-b in [67]. Whereas, we achieve similar classification accuracy as [67] on MNIST, using $\sim 50\times$ less MAC/MAV operations per classification. Our numbers are also comparable to the energy-efficiency of [68] (not quoted for MNIST), which uses 1-b for weights. Next, compared to a near-memory approach [62], which uses only 1-b for IFMP/OFMP, we still achieve $8.5\times$ improvement in energy-efficiency. This is because our approach exploits high parallelism of accessing multiple memory addresses simultaneously, without the need to sequentially and explicitly read out the data (filter weights) from the memory. We also achieve a higher classification accuracy compared to [62], because of using 6-b for inputs/outputs. Finally, our work also achieves better classification accuracy than prior in-memory approach [42], although it supports 8-b weights. This is because we reduce the effect of bit-cell variation when evaluating the weights. In addition, our approach benefits from more parallelism, by supporting 16 different dot-product computations per array per cycle, compared to 1 for [42].

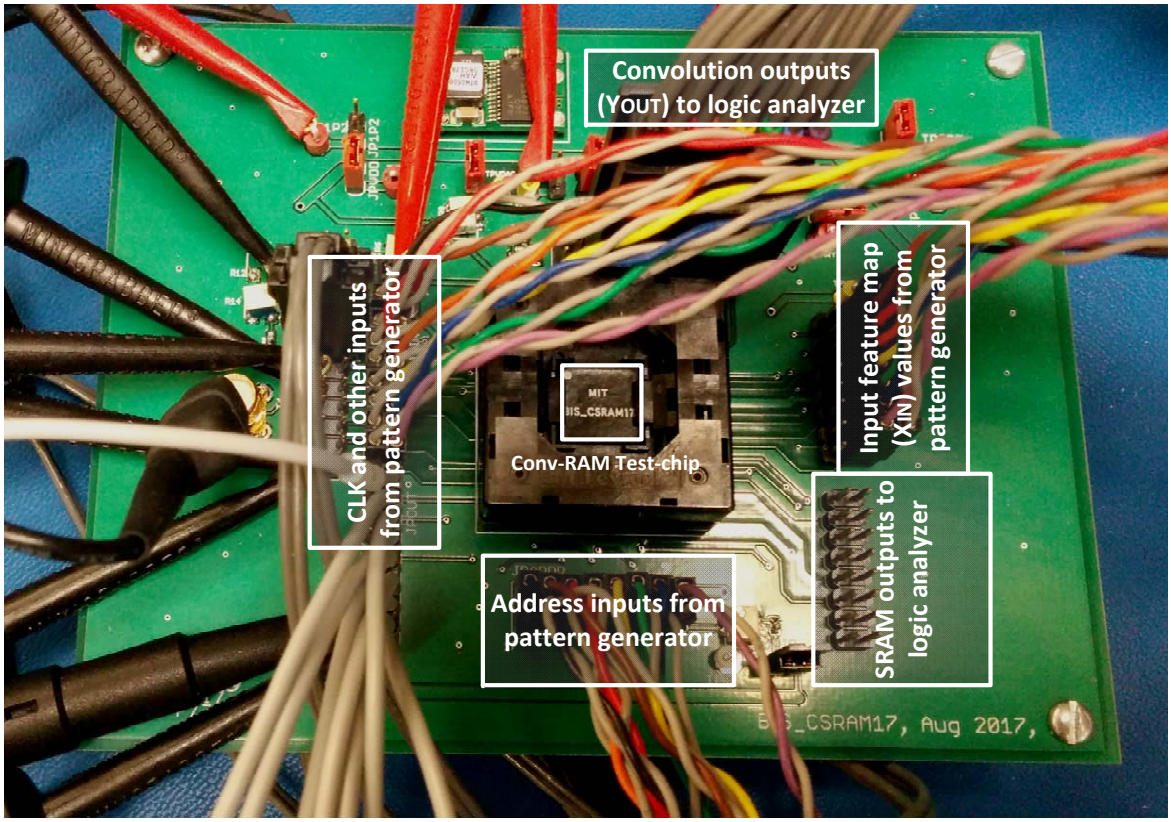


Figure 3-30: Test setup photograph, showing the chip under test in the center, digital inputs coming from a pattern generator and digital outputs sent to a logic analyzer for further processing.

3.6 Test/ Demonstration System

Fig. 3-30 shows the test setup for measuring the power consumption of the Conv-RAM. A pattern generator is used to continuously send all the digital inputs to the chip. Hence, it helps in measuring the average current consumption from the DC voltage sources. The digital outputs from the chip are sent to a logic analyzer, which captures them for multiple iterations. The logic analyzer output is exported as a text file and further analyzed/processed in MATLAB.

Although the pattern generator and the logic analyzer are useful to quickly test out the system, a more automated setup is required when a test has to be repeated for 10,000 different inputs (MNIST test dataset). Fig. 3-31 shows the setup for this. The inputs for a particular CNN layer are sent from the MATLAB environment, where it

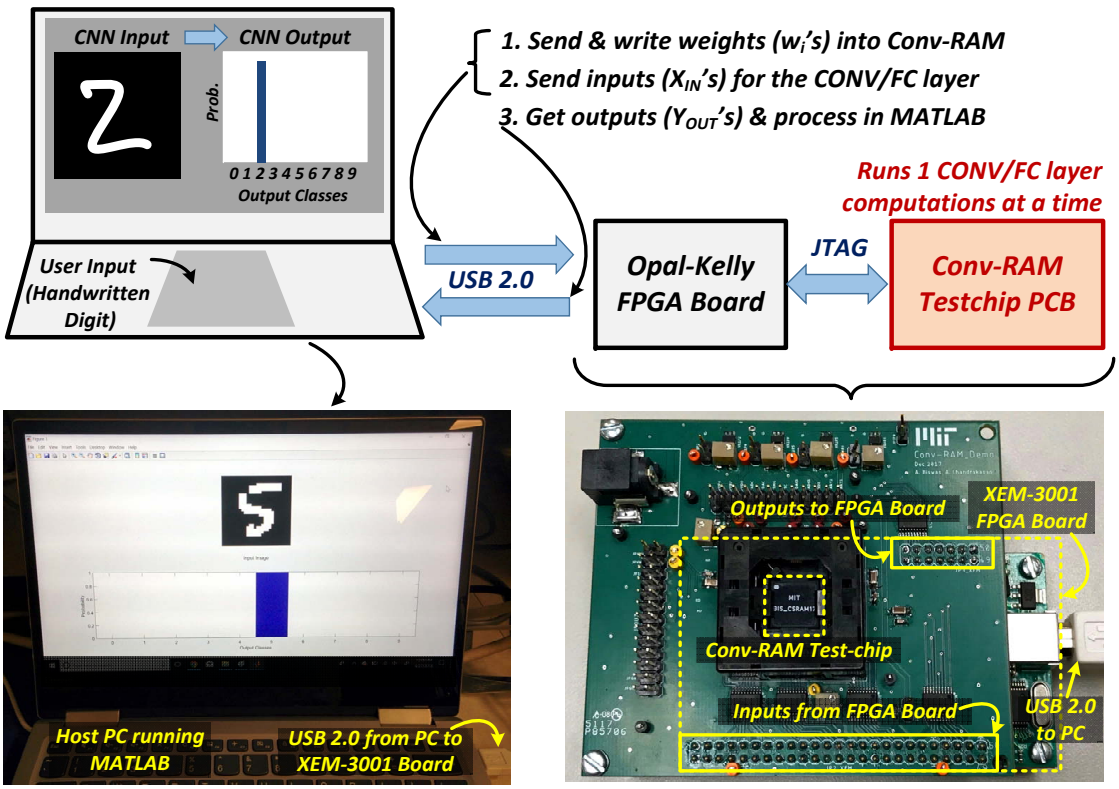


Figure 3-31: Demonstration system setup for automatically running the 4 CONV/FC layers of the LeNet-5 CNN for a given hand-written test input. Also used for measuring error rate for the 10,000 test images in the MNIST dataset.

is easy to process vectors as well as multi-dimensional arrays. An FPGA board (Opal-Kelly XEM3001) can interface with MATLAB over USB and receive digital inputs serially. A FIFO ($FIFO_I$) is used in the FPGA to convert this serial bit-stream from MATLAB to a multi-bit vector output. The $FIFO_I$ outputs are read using the FPGA's internal clock ('clk1'), and sent to the Conv-RAM test-chip. At the same time, the outputs from the Conv-RAM test-chip are sampled using 'clk1' and stored in an output FIFO ($FIFO_O$) in the FPGA. Finally, the FPGA communicates back to MATLAB and send those outputs serially over USB. The serial outputs are then interpreted in MATLAB and converted into the arrays of appropriate shapes/sizes, for a given CNN layer. This is done iteratively for all the 10,000 test inputs in the MNIST dataset, without any manual intervention. Furthermore, multiple CNN layers (each with different parameters) can be run one after the other using this setup. The outputs from the final layer F6 of the LeNet-5 CNN are used to determine the classification

results, from which the average error rate is obtained. This setup can also be used for demonstration with a user-provided test input: 28×28 image of a hand-written digit. The system, after running through all the 4 CONV/FC layers with the Conv-RAM hardware, would display the probablity for the 10 different classes, as shown in Fig. 3-32 for a few different inputs.

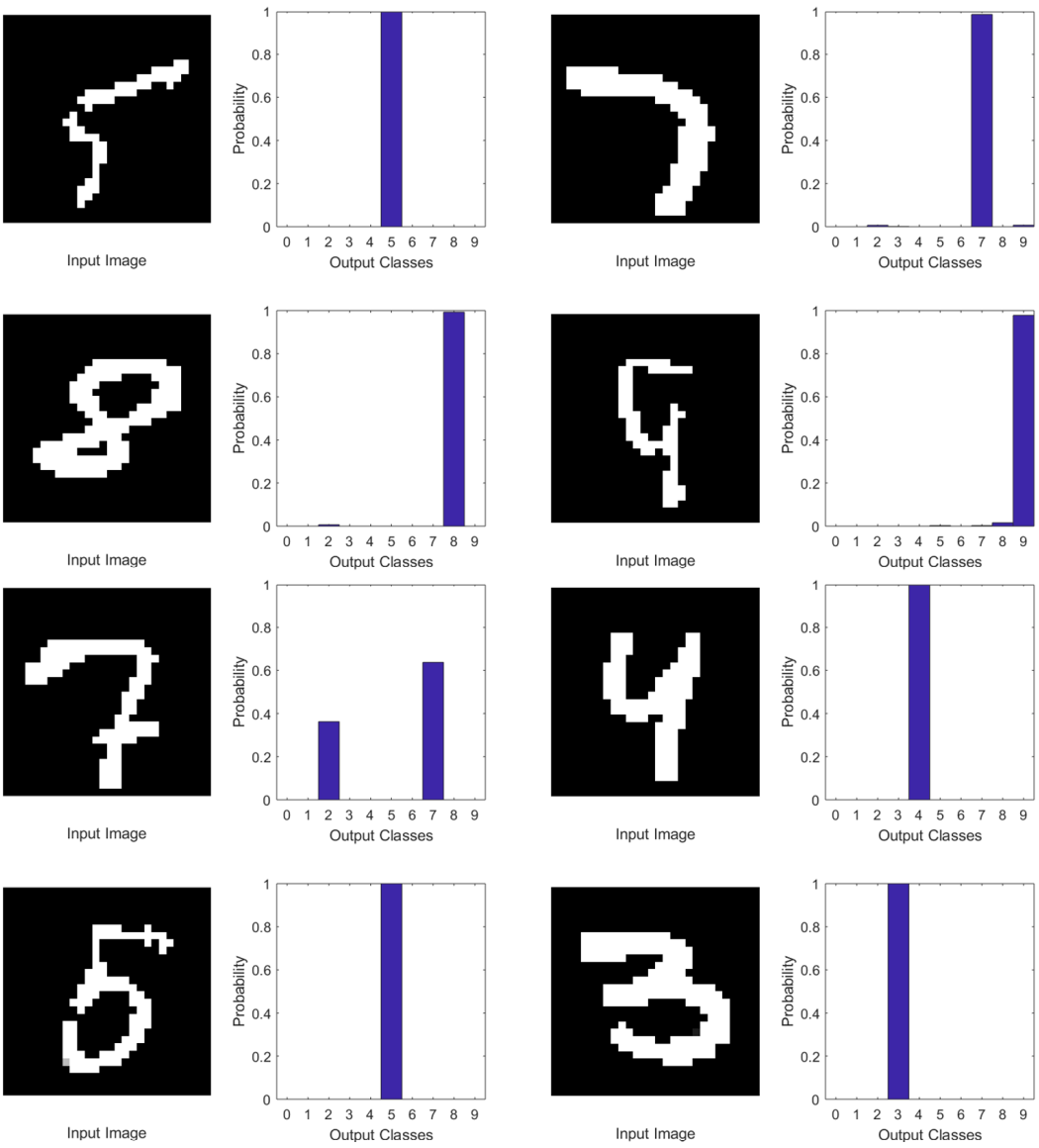


Figure 3-32: Probability distribution over the 10 different output classes for a few test inputs, obtained after running the 4 CONV/FC layers of the LeNet-5 CNN using the Conv-RAM test-chip and the setup shown in Fig. 3-31.

3.7 Conclusion

In this work, we presented an SRAM-embedded convolution (dot-product) computation architecture for running binary-weight neural networks. We demonstrated functionality with the LeNet-5 CNN on the MNIST digit-recognition dataset, achieving classification accuracy close to digital implementations and much better than prior in-memory approaches. This is made possible by our variation-tolerant architecture and also the support of multi-bit resolutions of input/output values. Compared to prior digital accelerator approaches using small bit-widths, we achieve similar or better benefits in combined energy-efficiency and throughput metrics, by overcoming some of the major limitations of memories in traditional computing paradigms. This is because our architecture can significantly reduce data transfer by running massively parallel analog computations inside the memory. The results indicate that the proposed SRAM-embedded architecture is capable of highly energy-efficient convolution computation that could enable low power ubiquitous ML applications for smart devices in the Internet-of-Everything.

Chapter 4

Variation-tolerant Read Sensing Technique for Non-volatile Resistive Memories

Resistive memories [70, 71] are a class of non-volatile embedded memories that have the potential to be a universal memory technology by providing the density of DRAM, the speed of SRAM and the non-volatility of Flash. One of the promising candidates among resistive memories is the Spin-Transfer Torque Random Access Memory (STT-RAM). STT-RAM offers a lower leakage consumption and up to 4× higher density compared to SRAM circuits (Table 4.1) [72]. Additionally, it achieves similar density as DRAM with the advantage of being non-volatile [73].

Table 4.1: Comparison of STT-RAM vs. other popular memory technologies [72, 73]

Metric (↓)	**SRAM**	**DRAM**	**STT-RAM**
Density	0.25×	1×	1×
Leakage	High	Low	Low
Write Latency	0.1×	0.5×	1×
Write Energy	Low	Low	High
Non-volatility	No	No	Yes
Sense Margin	High	High	Low

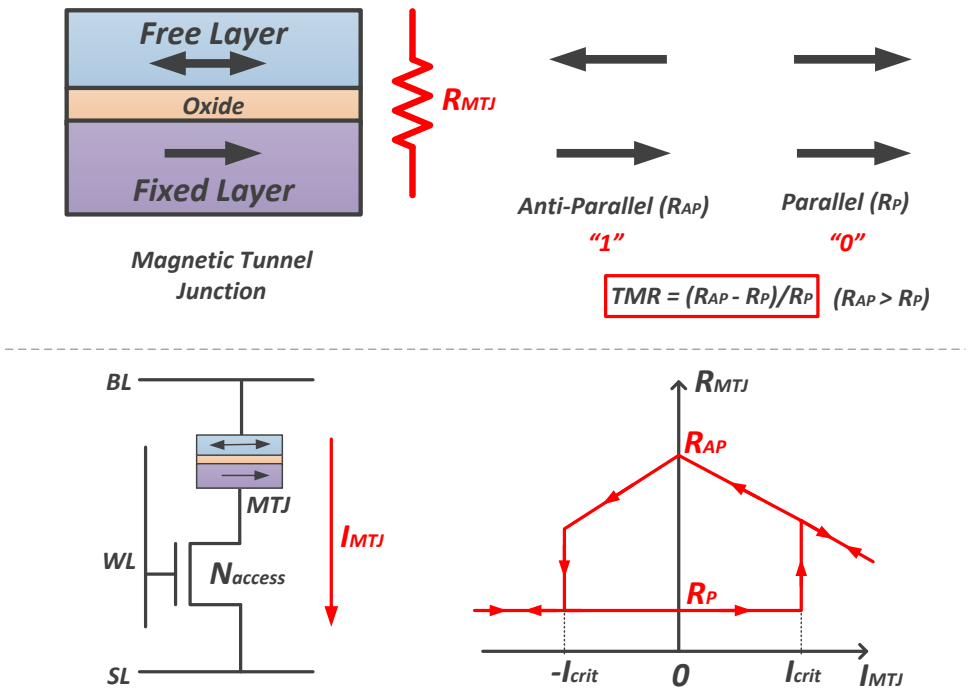


Figure 4-1: Basics of a STT-RAM bit-cell, showing the MTJ device and R-I hysteresis curve.

An STT-RAM bit-cell typically consists of a 1T-1R structure [74, 75], with a resistive storage device (based on magnetic tunnel junctions or MTJ) and an access transistor (Fig. 4-1). A 1T-1R bit-cell has 3 terminals: (1) a word-line (WL) to access the cell, (2) a bit-line (BL) and (3) a source-line (SL), to apply appropriate bias voltages to the MTJ device. An MTJ is made of two ferromagnetic layers separated by a thin oxide. If the oxide layer is thin enough (few nm's), then electrons can "tunnel" through it when a voltage is applied, and thereby conducting current. One of the ferromagnetic layers is "fixed" in terms of its spin-polarization, while the other ("free") layer's spin can be altered by passing a spin-polarized DC current through it. Hence, there are two possible resistance states: high-resistance (R_{AP} or '1'), when the spins are aligned anti-parallely and low-resistance (R_P or '0'), when the spins are aligned parallely. The difference between these 2 resistance states is characterized by a tunnel-magneto-resistance ratio (TMR), where $TMR = (\frac{R_{AP}}{R_P} - 1)$. For MTJ devices the TMR is typically around 100%-200%, depending on technology, temperature etc., which makes it challenging to distinguish the two resistance states correctly. Fig. 4-2

shows a typical distribution [76] of the 2 MTJ resistance states, with $\mu \sim 2.1K\Omega, \sigma \sim 4\%$ for R_P and $\mu \sim 4.1K\Omega, \sigma \sim 3\%$ for R_{AP}. Furthermore, the resistance of the access transistor adds to that of the MTJ, reducing the the read-margin between the '0' and '1' states.

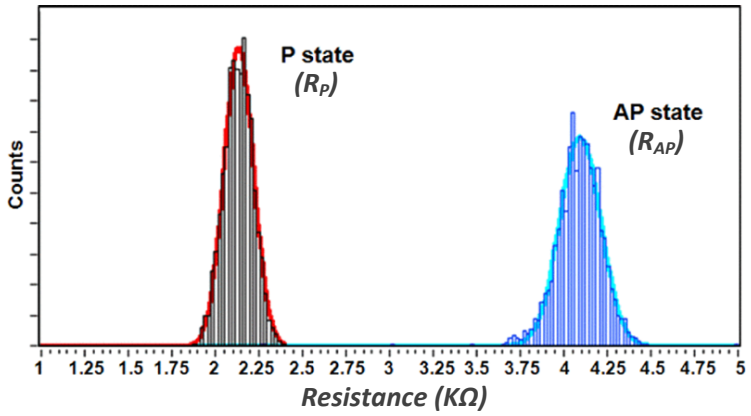


Figure 4-2: Distribution of the 2 resistance states of the MTJ: low (R_P) and high (R_{AP}), with 100% TMR [76].

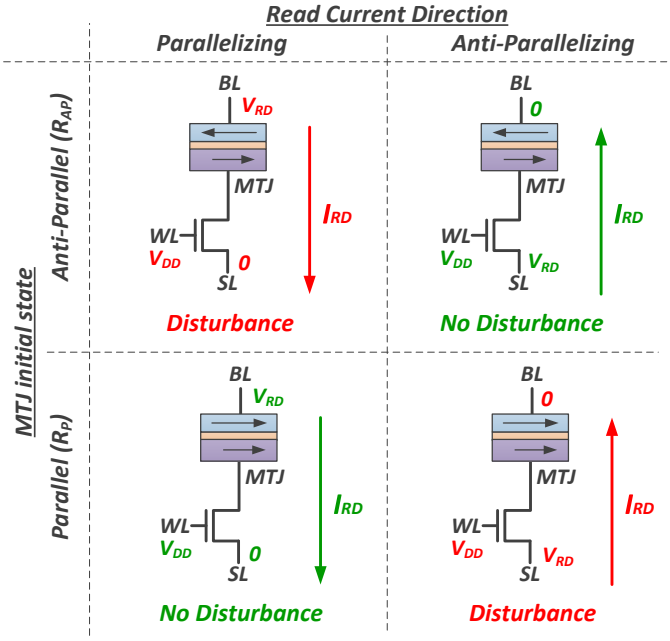


Figure 4-3: Possible scenarios for MTJ disturbance during a read operation.

Although the read margin can be improved by increasing the bias current through the MTJ, it is not ideal due to the possibility of a read disturbance. During a read

operation, the direction of the current through the MTJ is fixed. Hence, for one of the states there might be accidental flipping (Fig. 4-3), resulting in an unwanted write operation (i.e. a disturbance). This is true not only for the data MTJ, but also for any reference MTJ's, that would be needed for reference voltage/current generation during a read operation. Hence, the read current through the MTJ (I_{mtj}) should be kept as low as possible. Therefore, with increasing variations in sub-32 nm CMOS processes along with variation in MTJ resistance, it becomes extremely challenging to design a read-sensing scheme that achieves low read-disturbance and high yield ($> 5.5\sigma$).

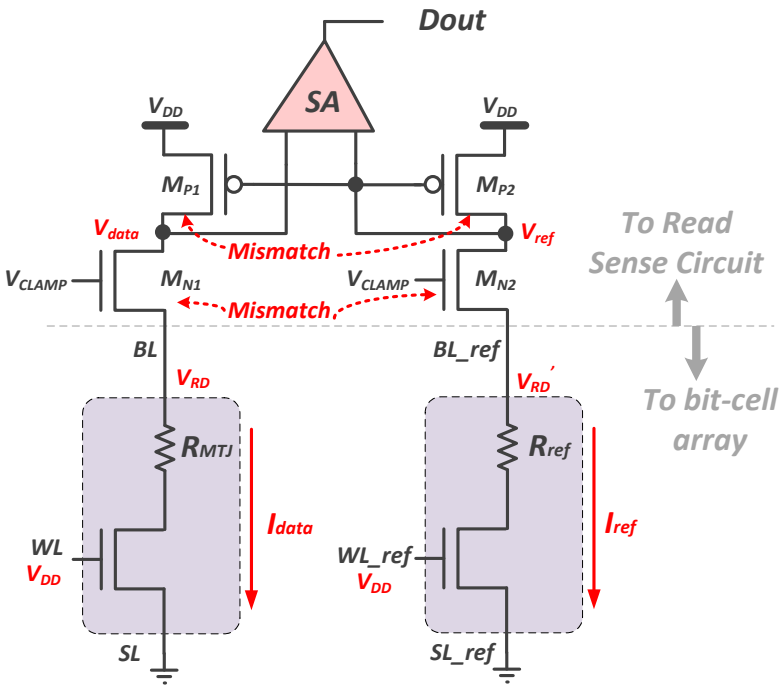


Figure 4-4: Conventional current-mode read sensing scheme for STT-RAM.

Previous work [74, 75, 77] have typically used current-mode read sensing schemes. Fig. 4-4 shows a typical current-mode sensing scheme for STT-RAM. It involves biasing the BL's of both the data and reference MTJ's to $\sim V_{RD}$, using clamp NMOS transistors (M_{N1}, M_{N2}). With the same bias voltages applied, the current difference between the data and reference MTJ's are converted into a voltage difference by the current-mirror PMOS transistors (M_{P1}, M_{P2}). Finally, the difference between V_{data} and V_{ref} is compared with a sense-amplifier (SA) to get a full-rail output (D_{out}). This

scheme suffers from any mismatch between M_{N1} and M_{N2}, since that would create different bias voltages for the data and the reference MTJ. It is also affected by the mismatch between M_{P1} and M_{P2}, since that might reduce the voltage difference between V_{data} and V_{ref}. That, in turn, would make it harder for the SA (with its own offset voltage) to detect the output correctly. Hence, this scheme does not have a very good yield. Furthermore, it typically operates from a high supply voltage due to voltage headroom requirements and hence consumes a lot of power.

In this work [78] we try to address these issues to design a robust read sensing circuit for STT-RAM, which would work for yield $> 5.5\sigma$ and reduce power consumption. The robustness to variations is achieved in mainly two ways. Firstly, due to the pseudo-differential nature (comparing data to two references 'ref1' and 'ref0') of the sensing scheme we get $2\times$ signal margin as compared to a single reference scheme [74, 77]. Secondly, the offset cancellation of the sense-amplifier (SA) makes it more suitable to tolerate variation from the array due to MTJ resistance variation. Also due to offset cancellation, we can use small sized devices for the SA, which results in lower area and less power. Thus the proposed technique benefits from technology scaling. Furthermore, since the sense amplifier has a differential topology it reduces the effect of supply noise and improves common mode rejection.

4.1 Proposed resistive-divider based Pseudo-differential Read Technique

4.1.1 Concepts and Modeling

Fig. 4-5 shows the concept of the proposed resistive-divider ('v1') based read sensing technique. The basic principle of sensing the MTJ resistance is comparing a resistive-divider output voltage V_O (formed by a ref-'1' MTJ and the data MTJ) to 2 reference voltages V_H and V_L. V_H and V_L are also generated from 2 resistive-divider networks using ref-'1' and ref-'0' cells. $R_{ref,H}$ and $R_{ref,L}$ are the resistances of the ref-'1' and ref-'0' devices respectively. R_{mtj} is the resistance of the data MTJ to be sensed.

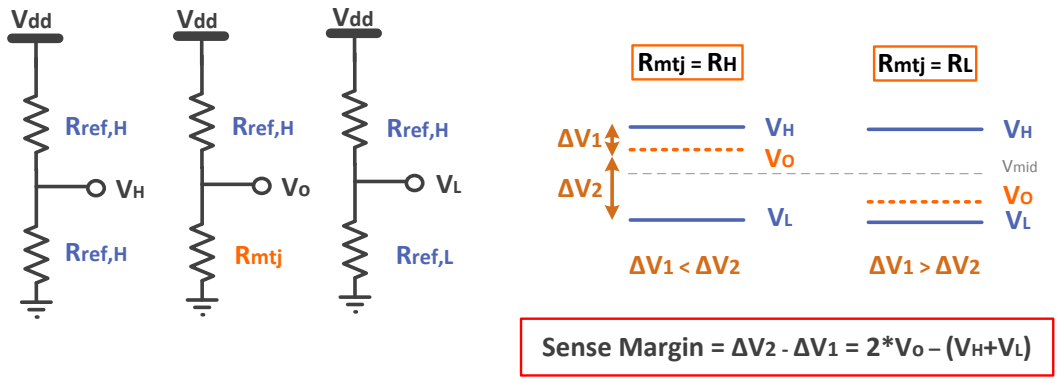


Figure 4-5: Concept of the resistive-divider read technique ('v1') for STT-RAM.

Therefore, the sense margin (SM) can be expressed as the difference of V_O from the 2 reference voltages (V_H, V_L):

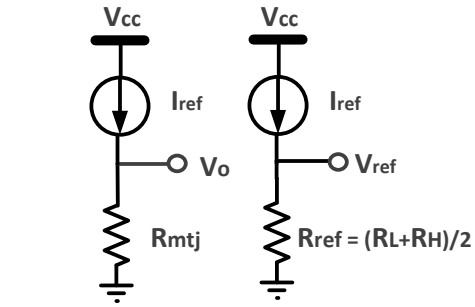
$$SM = \Delta V_2 - \Delta V_1 = (V_O - V_L) - (V_H - V_O) = 2 \times V_O - (V_H + V_L) \tag{4.1}$$

Assuming a TMR of 150%, i.e. $R_H = 2.5 \times R_L$, the nominal SM for data '1' and data '0' can be calculated as:

$$\begin{aligned} SM_1 &= 2 \times \frac{2.5R_L}{2.5R_L + 2.5R_L} V_{dd} - (0.5 + \frac{R_L}{2.5R_L + R_L}) V_{dd} = 0.214 V_{dd} \\ SM_0 &= 2 \times \frac{R_L}{2.5R_L + R_L} V_{dd} - (0.5 + \frac{R_L}{2.5R_L + R_L}) V_{dd} = -0.214 V_{dd} \end{aligned} \tag{4.2}$$

Now, let us compare this scheme with an ideal voltage sensing scheme, as shown in Fig. 4-6. R_L and R_H are assumed to be $4K\Omega$ and $10K\Omega$, respectively, with each having a resistance variation of $\sigma = 5\%$. We also assume the maximum read current (for the '1' state which may be potentially disturbed) is limited to $20\mu A$ (nominal). This implies a maximum stack bias voltage of $V_{dd} = 2 \times 10K\Omega \times 20\mu A = 400mV$. As seen from Fig. 4-6, the nominal SM is 1.43× higher for the proposed scheme, compared to the ideal voltage sensing method. This is because of the use of 2 reference voltages vs. 1 for the conventional approach. We also observed the effect of applying (gaussian) variations to the different resistors, on the sense margin. 10K monte-carlo

(MC) simulations are done in MATLAB to obtain the μ and the σ values. 2 different versions are simulated: one assumes the reference voltages being generated using only 1 (=N) set of resistors, whereas for the other version 16 (=N) sets of reference resistors are used to get the final reference voltage. As seen from Fig. 4-6, the μ/σ of the SM is almost the same for both the proposed and the conventional schemes. This is because both the μ and the σ of the SM are proportionately increased.[1]



Conventional voltage sensing (ideal)

$SM = |R_{mtj} - R_{ref}| * I_{ref}$

Sensing Scheme	Prop. (v1)	Conv.
# ref. MTJs	5	4
Nominal SM (mV)	85.6	60
SM μ/σ (N = 1)	5.1	5.5
SM μ/σ (N = 16)	6.0	6.1

R_H = 10 KΩ, R_L = 4 KΩ, σ_{rmtj} = 5%, I_{ref} = 20μA, V_{dd} = 0.4V

Figure 4-6: Comparison of the SM of the proposed ('v1') resistive-divider based read vs. a conventional (ideal) voltage sensing.

To improve the SM in the presence of MTJ resistance variation, a modified resistive-divider ('v2') based read technique is proposed, as shown in Fig. 4-7. The principal concept of comparing the data voltage to 2 reference voltages, still remains the same as 'v1'. However, the reference MTJ devices on the top of the resistive-divider stacks are re-used and time-shared in 2 phases: Φ_1, Φ_{1B}. Hence, the same reference MTJ (at the top of the resistor stack) is used to generate the voltage differences: $\Delta V_1, \Delta V_2$. This reduces the effect of reference MTJ resistance variation on the SM. Fig. 4-7 shows the typical operation waveforms of this technique. The voltage swings ($\Delta V_1, \Delta V_2$) are generated at the 2 nodes: 'in1' and 'in2'. In phase Φ_1, the node 'in1' is connected to V_O and the node 'in2' is connected to V_L. So V_{in1} is close to $V_{dd}/2$ (if we assume the data bit is '1') and V_{in2} is close to $V_{dd}/3$ (assuming $R_H = 2 \times R_L$). In the next phase Φ_{1B}, 'in1' is connected to V_H and 'in2' is connected to V_O. Hence,

[1]Given a random variable Y $\sim$ N(μ, σ), a new random variable Y' = kY follows N($k\mu, k\sigma$), where k is a scalar.

V_{in1} is equal to $V_{dd}/2$ and V_{in2} is close to $V_{dd}/2$. Thus the swing at the node 'in1' from phase Φ_1 to Φ_{1B} is smaller than the swing at 'in2' i.e. $\Delta V_1 < \Delta V_2$. Whereas if data is '0', $\Delta V_1 > \Delta V_2$. The sense margin is still defined as $(\Delta V_1 - \Delta V_2) = 2 \times V_O - (V_H + V_L)$, as before.

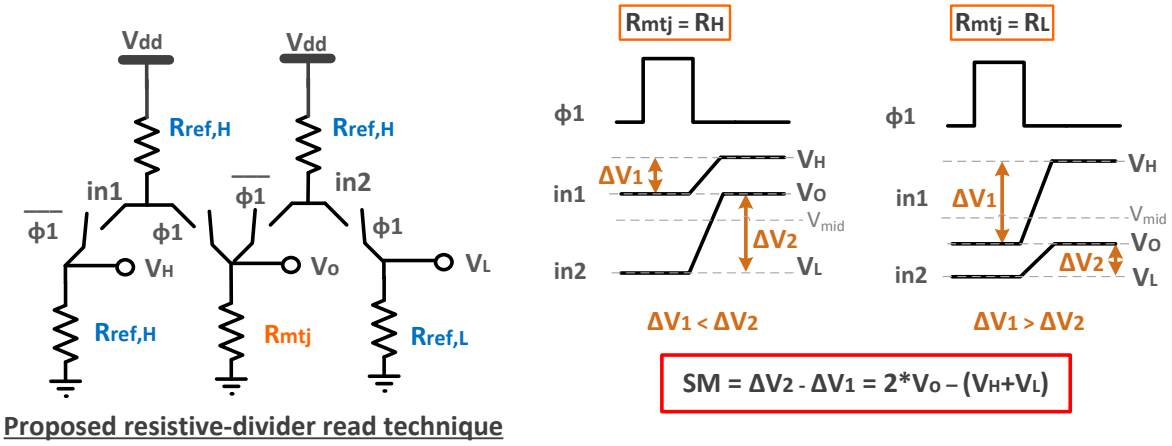


Figure 4-7: Modified resistive-divider ('v2') read technique for STT-RAM, which re-uses the top MTJ reference devices in 2 phases.

Table 4.2: Comparison of the different read techniques

Metric (↓)	**Ideal Voltage-Sensing**	**Resistive-divider 'v1'**	**Resistive-divider 'v1'**
# of ref. MTJ's	4	5	4
Nominal SM (mV)	60	85.6	85.6
SM μ/σ (MATLAB sim)	6.1[1]	6.0[1]	7.2
SM μ/σ (SPICE sim)	-	5.3	6.6
$\mu - 4\sigma$ (mV) (SPICE sim)	-	14.5	22.4

[1] Assuming N = 16 way averaging for generating the references

An analytical model of this scheme was simulated in MATLAB to analyze the effect of MTJ resistance variation on the sense margin. The μ and μ/σ of the sense margin (SM) is shown in the table 4.2, and compared with the approaches mentioned before. As seen from the figure, the modified sensing scheme ('v2') improves the μ/σ

of the SM vs. both 'v1' and the ideal voltage sensing approaches. This is because the voltages ΔV_1 and ΔV_2 are generated using the same $R_{ref,H}$ resistor in both the phases Φ_1 and Φ_{1B}, which reduces the effect of its variation on the sense voltage. The improvement is still maintained after incorporating actual device models and other parasitic effects in SPICE simulations of the 'v1' and 'v2' circuits. For the SPICE simulations, the array architecture (described in the next subsection) is used.

The proposed resistive-divider technique also burns less power since V_{dd} is smaller ($\sim$400mV) compared to the nominal V_{cc} ($\sim$1V, due to voltage headroom requirements) for ideal voltage sensing. Furthermore, the modified resistive-divider technique uses 4 reference MTJ's, which is the same as the ideal voltage sensing case ($\because R_{ref} = (R_H + R_L)/2 = (R_H + R_L)||(R_H + R_L)$). Hence, it does not incur any extra area overhead, compared to a single reference scheme. However, there is some timing penalty due to the 2-phase operation of the proposed scheme.

4.1.2 Array implementation

The reference devices needed for generating the V_{in1}, V_{in2} voltages (Fig. 4-7), can be implemented in the memory array itself and they can be shared by a sub-array (or sector) to reduce their area overhead. Fig. 4-8 shows the implementation of the four reference devices. There is one reference row per sector (driven by 'Ref_WL') and one reference column per global column (with 'BL_ref' and 'SL_ref'). The reference column and reference row implement the top and bottom devices respectively, of the resistor-divider stack in Fig. 4-7. Since the voltages at the nodes 'in1' and 'in2' are needed simultaneously, the unselected sector (shown in the bottom part of Fig. 4-8) is used to generate the reference voltages (V_H, V_L) by operating the switches in phases Φ_1 and Φ_{1B}, as described in Fig. 4-7. To further reduce the area-overhead of implementing the reference devices, instead of having two separate reference rows, the $R_{ref,H}$ and $R_{ref,L}$ devices (in the lower part of the resistor-divider stack) are shared between two adjacent global columns. They generate the reference voltages for the two adjacent global columns in a time-interleaved manner, i.e. the Φ_1 and Φ_{1B} signals for the global column<n> are opposite to that of the global column<n+1>.

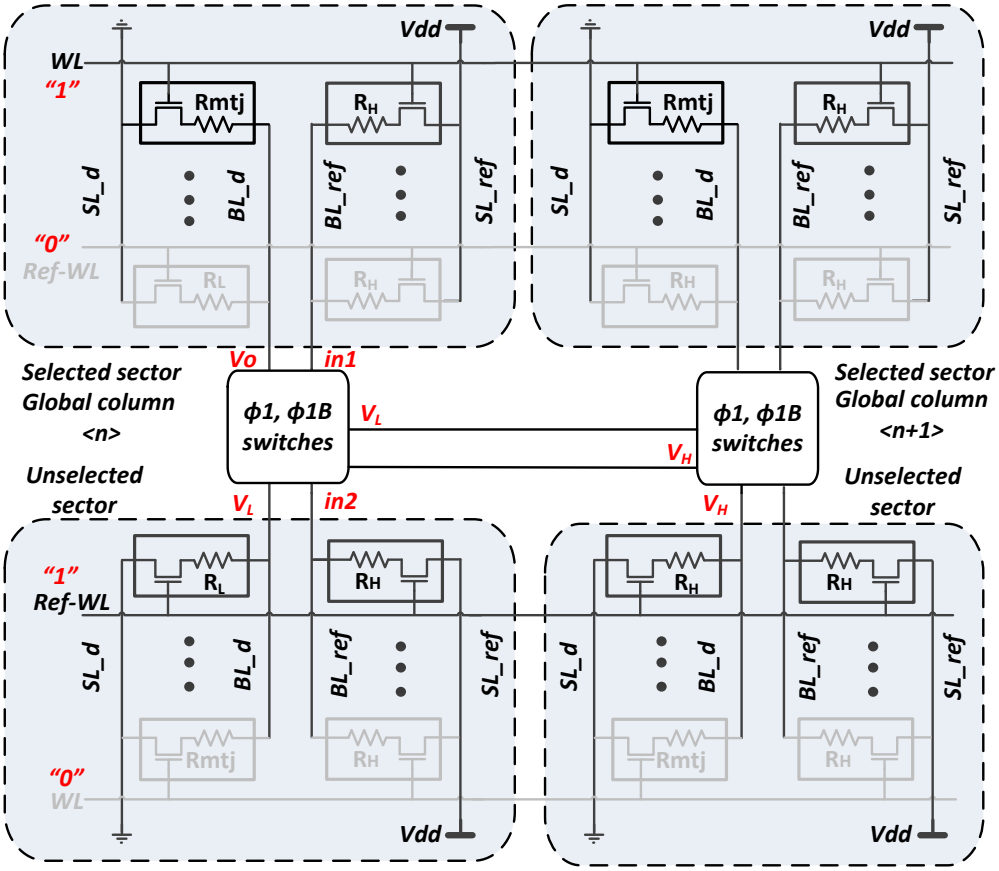


Figure 4-8: Array implementation of the reference devices (shown for two global columns), with the reference voltages (V_H, V_L) generated from the unselected sector.

4.2 Proposed Sense-Amplifier with Improved Offset-cancellation

The difference of V_O from the 2 reference voltages V_H and V_L are fed as the 2 inputs of a differential sense amplifier. The sense amplifier (SA) consists of 2 inverters which can be reconfigured both as an amplifier and a latch, similar to [79]. We propose a technique to fully cancel the trip-point (V_I) mismatch of the 2 inverters to reduce the sense-amplifier offset voltage.

Fig. 4-9 shows the proposed sensing circuit. The nodes 'in1' and 'in2' are from the resistive-divider network (Fig. 4-7), described above. During phase Phi_1, the two inverters 'inv1' and 'inv2' are biased at their respective trip points. This self-biasing scheme makes sure that the inverters are biased in their high gain (amplifying) region

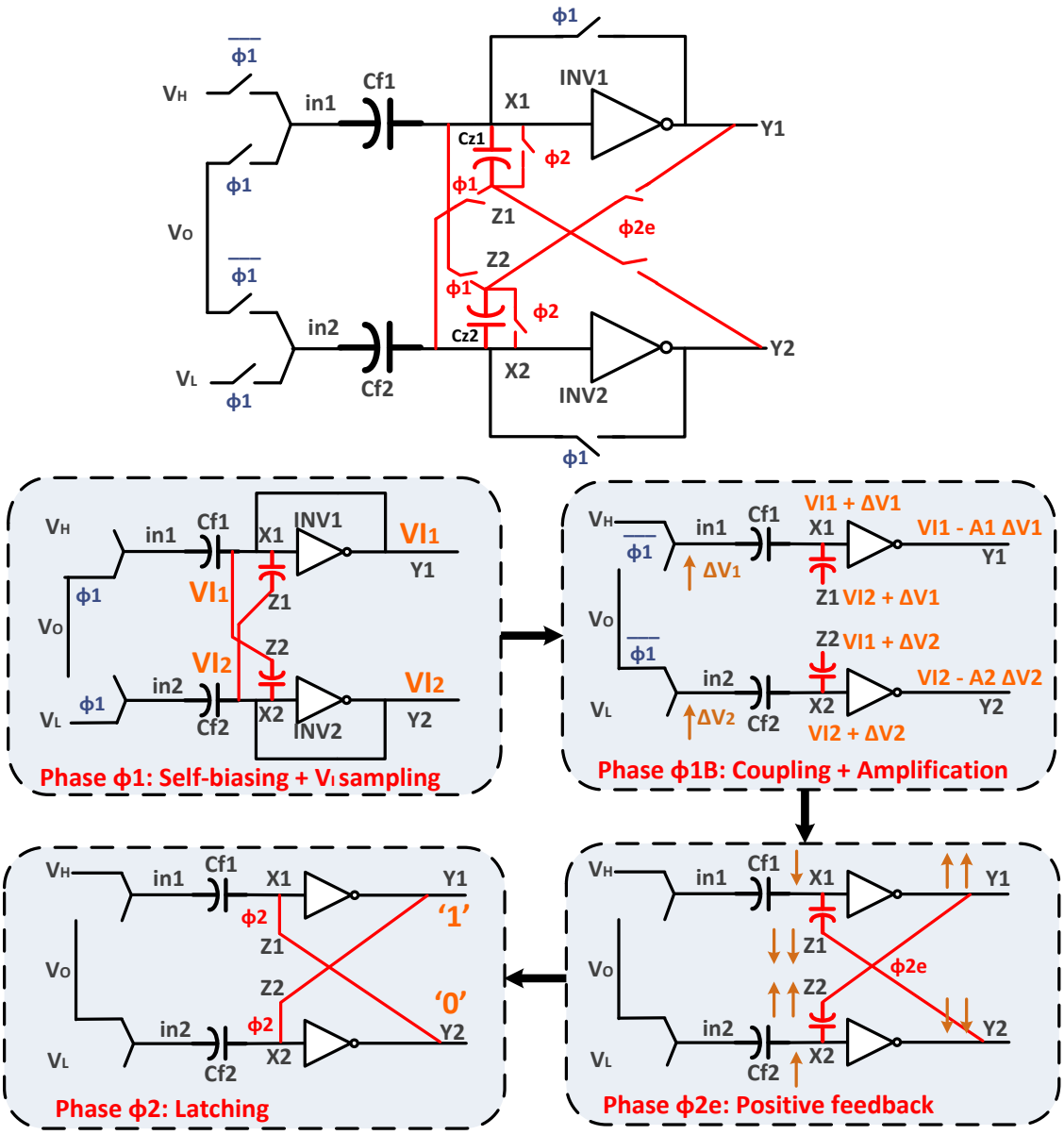


Figure 4-9: Proposed sense-amplifier with full offset cancellation of inverter trip point mismatch ($V_{I1} \neq V_{I2}$) by addition of an extra phase (Φ_{2e}) and using extra sampling capacitors (C_{Z1} and C_{Z2}). Shown for the case of $\Delta V_1 < \Delta V_2$, i.e. data '1'.

irrespective of device variation. During this phase the sampling capacitors (C_{z1} and C_{z2}) samples the difference of the trip points of the 2 inverters. Next, in phase Φ_{1B}, the negative feedback for the inverters is disconnected. The swing at the nodes 'in1' and 'in2' are coupled to the input of the inverters via the coupling capacitors (C_{f1} and C_{f2}). The swing at the input of the inverters is amplified by their gain (A) at

the outputs ('y1' and 'y2'). The nodes 'z1' and 'z2' of the capacitors C_{z1} and C_{z2} are floating in this phase, so they do not provide any extra loading at the input of the inverters. After this, in phase Φ_{2e}, 'z1' is connected to 'y2' and 'z2' is connected to 'y1'. This creates a positive feedback circuit (as long as $C_z > C_f/A$). So the inputs 'x1' and 'x2' will move in opposite directions (from the respective V_I points), depending on whichever of ΔV_1 and ΔV_2 is higher. Similarly, the outputs 'y1' and 'y2' will move in opposite directions. Finally, in phase Φ_2 the capacitors C_{z1} and C_{z2} are shorted, completing the latch and generating full-rail outputs. Fig. 4-10 shows the states of the SA's just before the positive feedback in enabled, for both the proposed and the conventional (without the C_z capacitors and phase Φ_{2e}) SA's. It can be seen that, even if the trip points of the two inverters are different, it does not affect the positive feedback for the proposed SA, since the difference of V_{I1} and V_{I2} was already sampled across C_{z1} and C_{z2} during phase Φ_1. By doing this we can fully cancel the offset due to V_t-mismatch in the 2 inverters. This is unlike the conventional SA, which suffers from mismatch between V_{I1} and V_{I2} when $X_1 - Y_2$ and $X_2 - Y_1$ connections are made for positive feedback (Fig. 4-10).

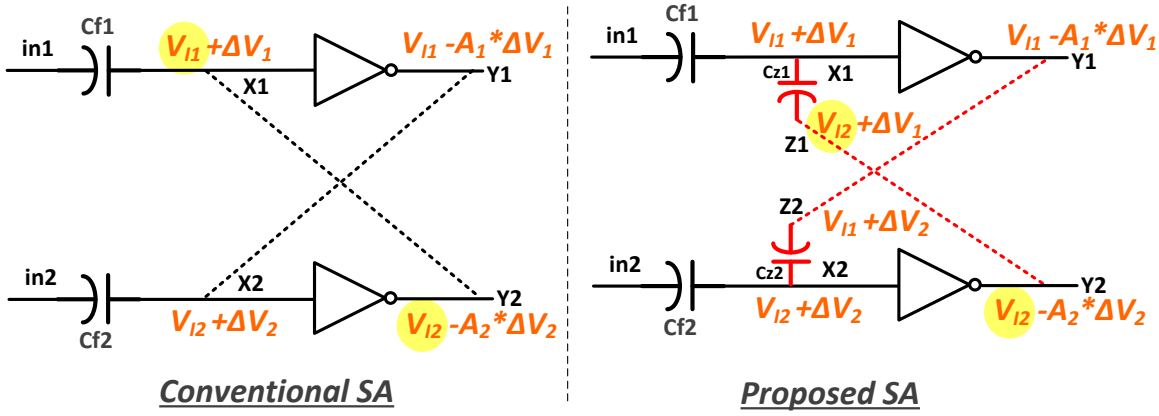


Figure 4-10: Comparison of the state of the conventional and proposed SA's before the positive feedback is enabled.

The waveforms of both the SAs are shown in Fig. 4-11 for a case where there is a mismatch of the V_I of the 2 inverters. As seen from the figure the proposed SA was able to resolve the data correctly (Y_1 = '1', Y_2 = '0'), due to the full offset cancellation technique, unlike the conventional SA. However, the proposed SA needs

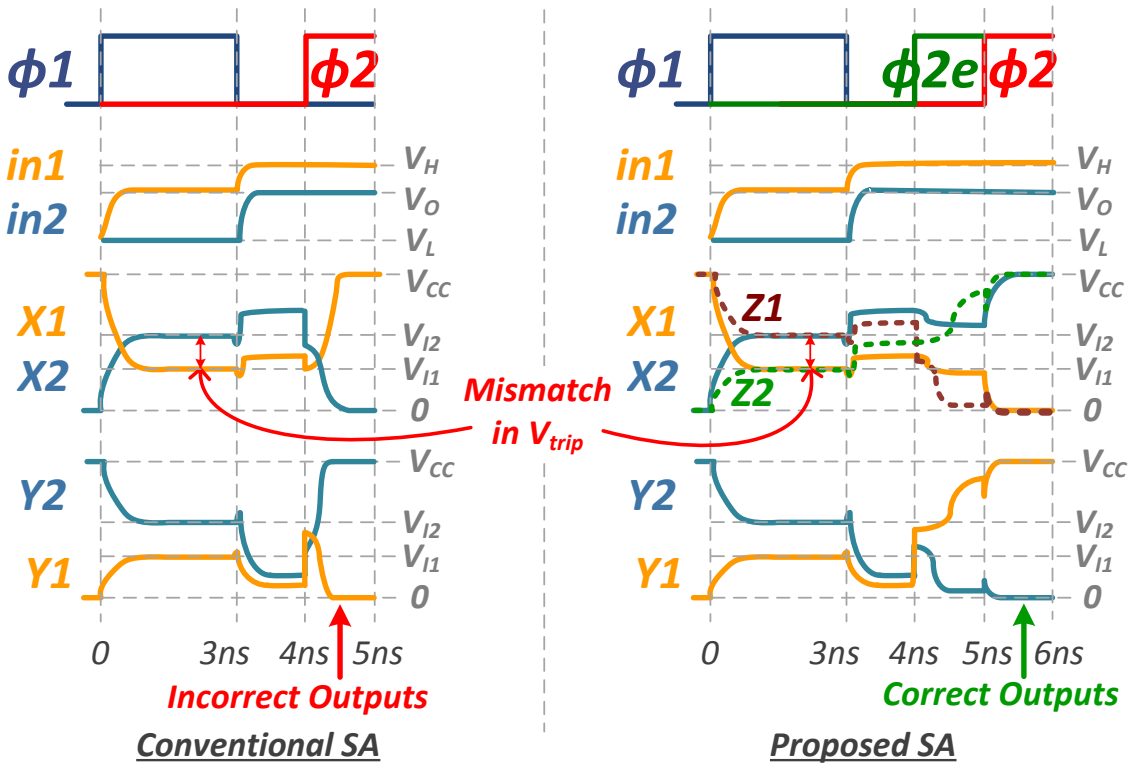


Figure 4-11: Simulation waveforms for the conventional and proposed sense-amplifiers when there is a mismatch of the V_I of the two inverters. Shown for the case of $\Delta V_1 < \Delta V_2$, i.e. data '1'.

a slightly longer sensing time, due to the addition of the extra phase (Φ_{2e}).

Table 4.3 shows the effect of variation on the simulated SA offset voltage (V_{off}). The proposed SA provides $\sim 3\times$ improvement in $\sigma(V_{off})$ over the conventional design. The residual offset is mostly due to other types of mismatches e.g. gains of the inverters, input/output capacitances, switch charge injection etc. The low sigma due to offset cancellation suggests that small sized transistors can be used for the sensing inverters and hence the SA power can be reduced. On the other hand, the SA area which is mainly dominated by the size of the sampling capacitors also reduces with CMOS scaling. This is because the capacitors can be implemented by regular MOS transitors. Hence, the proposed SA benefits from technology scaling in terms of both area and power metrics, which is very desirable when designing for future resistive memory technologies. Table 4.3 also shows the effect of supply noise on σ. For this analysis the V_{cc} of the SA is switched by ΔV_{cc} from phase Φ_1 to Φ_{1B}. This is because

changing V_{cc} in phase Φ_{1B} shifts the inverter trip points and this shift did not get sampled during phase Φ_1. Hence, this step jump in V_{cc} can have a negative impact on the SA operation. However, because the proposed scheme is differential it is more immune to supply noise (which is a common mode signal) than a single-ended sensing scheme [80].

Table 4.3: Variation analysis (simulated) and effect of supply noise on the proposed SA.

Offset Voltage (mV) (↓)	**Conventional SA**	**Proposed SA**
Mean (μ)	0.2	0.1
Standard Deviation (σ)	7.0	2.1
$\mu + 4\sigma$	28.2	8.5

% V_{cc} Noise	**σ of proposed SA (mV)**
0	2.1
5	3.4
10	4.4

4.3 Silicon Implementation and Measurement Results

The proposed SA was implemented in a 14nm CMOS technology [2]. The capacitors in the design (C_f, C_z) are realized using MOSFET devices to limit the technology cost while still achieving a compact footprint of $8\mu m^2$. To facilitate offset characterization, the resistive-divider output voltages (V_H, V_L, and V_O) are provided externally through the pads as shown in Fig. 4-12. A Schmitt trigger and a timing control block is integrated on-die to relax the tester requirements. After the DC inputs are set, START is asserted, and the timer generates all the phase clocks that are necessary for SA operation as described in Section III. The SA output becomes available on the

[2]Layout and chip testing were done at Intel (Hillsboro). We thank F. Hamzaoglu, U. Arslan, P. Jain and others in the Advanced Design team there for help.

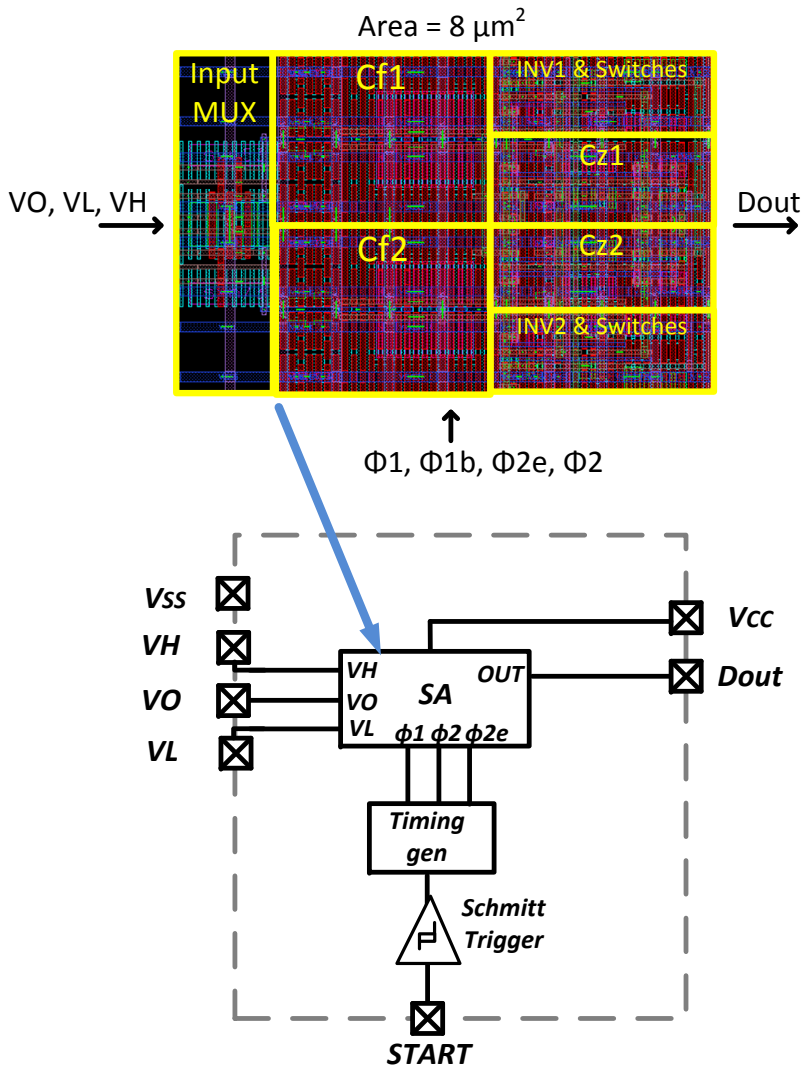


Figure 4-12: Sense amplifier layout and test structure including a Schmitt trigger and a timing generator block.

D_{out} pin at the end of Φ_2, and is held by the SA in the latch state until D_{out} value is recorded and START is deasserted for the next measurement.

The trip points of 71 SA instances were measured by stepping V_O from V_L (100 mV) to V_H (150 mV) with a 2mV resolution. Each instance was tested 16 times and the V_O levels that can reliably produce '0' or '1' at the output were recorded. Fig. 4-13 shows the distributions of the trip voltage (defined as $V_O - 0.5 \times (V_L + V_H)$) for both low-to-high and high-to-low transitions. The low-to-high trip point varies with a (μ, σ) of (11.3 mV, 2.6 mV), whereas the distribution of the high-to-low trip

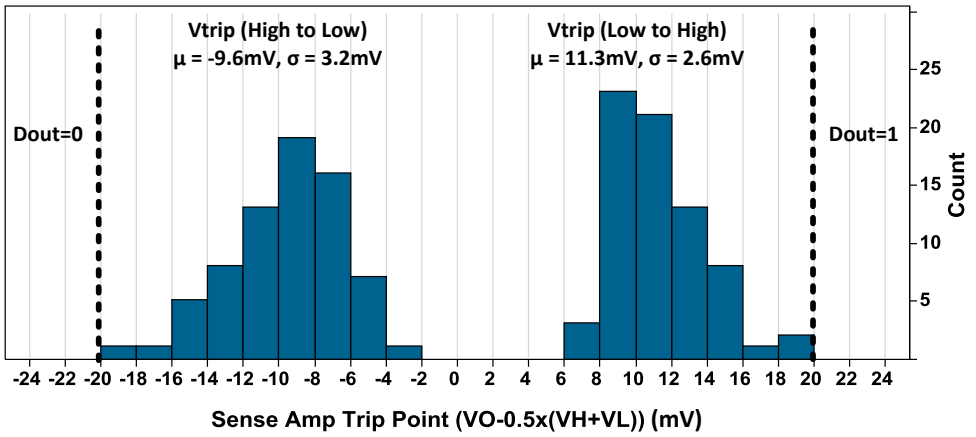


Figure 4-13: Measured SA trip point distributions showing the V_O levels that can reliably produce high and low at data out. Collected from 71 SA instances where each instance was tested 16 times.

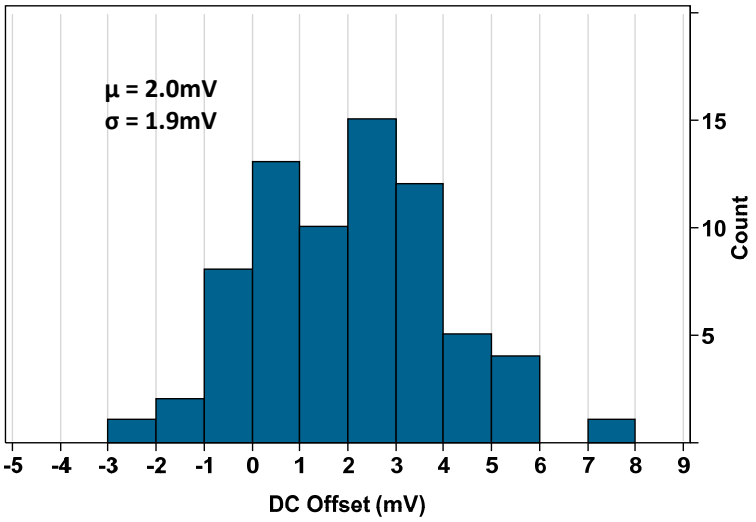


Figure 4-14: Measured DC offset from 71 SAs. Each SA was tested 16 times and random noise was removed from data by computing the average trip point. DC offset is then defined as twice the difference between the measured average and ideal trip points.

voltage has a (μ, σ) of (-9.6 mV, 3.2 mV). All the 71 SAs are able to reliably detect input signal levels beyond 20 mV in all 16 runs. We suspect that the data uncertainty within the 20 mV input range is partly caused by random AC noise events in our test environment, since the pre-silicon validation had confirmed the design to have a good tolerance to supply noise and low thermal noise. The AC noise is removed from the data by computing the average trip voltage of each SA from the 16 measurements.

The DC offset is then defined as twice the difference between the measured average and ideal trip points. Fig. 4-14 shows that the DC offset varies with a standard deviation of 1.9 mV, matching reasonably well to the expected value of 2.1 mV.

4.4 Conclusion

In this chapter we presented a resistive-divider based pseudo-differential read sensing structure for resistive memories. It achieves more signal margin due to the usage of 2 references, compared to the conventional single reference design. Techniques to mitigate variation effects of device resistances were proposed. A possible array implementation of the proposed resistive-divider based read scheme is also explained. Finally, a four-phase differential sense amplifier (SA) with an improved offset cancellation technique is proposed, to work with the resistive-divider read scheme. The SA was implemented in a 14nm CMOS process. 71 tested SA's achieve correct operation with 20mV input and achieve a DC offset σ of 1.9mV. This shows that the SA can tolerate large variations from the memory array, to achieve a high yield. On the other hand, due to the offset-cancellation technique the SA can be designed using small sized devices to achieve low area and power.

Chapter 5

Conclusions and Future Work

This thesis primarily explored avenues for improving the energy-efficiency of embedded SRAMs in modern computing systems, in which memory is often the main bottleneck. Firstly, we investigated techniques to reduce energy consumption of 6T SRAMs using: (1) assist circuits for voltage scaling, (2) alternate array layout implementations and (3) utilizing data properties for application-dependent further energy savings. 6T bit-cell based SRAM design was used, owing to its higher area-efficiency. We demonstrated very low voltage operations ($\sim$ 0.34V), inspite of using 6T bit-cells, whose operating voltage is generally limited due to conflicting requirements in read and write operations. We also achieved low access energy of 52.5fJ/bit (minimum) at 0.45V. The low voltage and low-energy SRAM would be ideal in many IoT applications, where the energy budgets are very limited. Our work would be also useful in applications (e.g. GPUs) where SRAM density is of key concern and using 8T/10T bit-cells might not be preferred.

The second approach for overcoming traditional limitations of memories, was to embed some computation capability inside the memory, for reduced data transfer and increased bandwidth. This is highly useful in data-intensive machine learning applications, in which the memory access and data movement energy dominates the energy of the arithmetic computation operations. We demonstrated the in-memory computation capability with a 16Kb SRAM array, running binary-weight neural network computations, used for the handwritten digit recognition task. We achieved better

energy-efficiency than conventional digital implementations, while still attaining similar classification accuracy. On the other hand, we improved classification accuracy compared to prior in-memory approaches, by addressing the impact of bit-cell variation on the analog computations. The measured results show that our approach could make it possible to run low-power ML algorithms in IoT devices, by energy-efficient local processing of data, instead of relying on the "cloud". This could enable "smart" decision making and "always-ON" sensing for these energy-constrained devices.

Finally, we also investigated read sensing circuits and architectures to improve yield for future non-volatile resistive memories, e.g. STT-RAM. These memory technologies have the potential to replace SRAMs for on-chip storage. However, they have unique design challenges compared to SRAM's, that need to be overcome before being adopted as a mainstream embedded memory technology.

5.1 Summary of contributions

This section summarizes the key contributions of this thesis.

5.1.1 Low-Power 6T SRAM with Data-Dependent Energy Savings

In this work, a low-voltage 128Kb 6T SRAM array was designed in a 28nm FDSOI process. Dynamic forward body-biasing (DFBB) was used to improve the write margin of the SRAM and to push the minimum operating voltage of the 6T SRAM to 0.34V.

- An energy-efficient array layout was proposed for implementing DFBB. The proposed layout shares the n-wells (body terminals) along the rows, compared to the conventional column-wise sharing. This reduced the n-well capacitance switched per cycle and consequently reduced its energy overhead by 4×.
- The proposed array layout allowed routing upto 4 word-lines per row. Since only 1 word-line is turned ON every cycle (due to 4:1 column-muxing ratio),

the bit-lines in the un-selected columns do not switch every cycle unnecessarily. Hence, the local bit-line (BL) dynamic energy was reduced by $\sim 25\%$.

- Data-prediction was incorporated at the local sub-array level (9% area overhead with 32 bit-cells/local bit-line), instead of using it at the bit-cell level, which would lead to a 10T design ($\sim 1.6\times$ overhead). Therefore, our architecture could still use area-efficient 6T bit-cells, while still saving global bit-line switching energy, when there is a correct data prediction. Upto 36% dynamic energy savings was demonstrated for the 128Kb SRAM, compared to a conventional single-ended read. The energy savings can be further improved by sacrificing a little on the area-efficiency, using smaller number of bit-cells per local bit-line.

5.1.2 In-Memory Computation for Low-Power Neural Networks

In this work, a 16Kb SRAM (Conv-RAM) with embedded dot-product computation capability was proposed for binary-weight neural networks. This architecture reduces data transfer by implementing in-place analog computations inside the memory array. The dot-product was modeled as an analog voltage averaging operation, which can be efficiently implemented inside the memory array.

- Analog inputs were directly sent on the global BLs, instead of using the WLs (conventional approach). This provided more linearity and better control of them using DACs at the SRAM periphery.

- The issue of bit-cell variations affecting the analog computations, is mitigated by using the bit-cell to only discharge one of its local BL to ground, to implement the 1-b multiplication. The analog input voltage was rather controlled by DACs, which were up-sized to have less variation. A hierarchical architecture (with local and global BLs), enabled de-coupling the weight multiplication and the analog averaging operations.

- Using the inherent BL capacitance for the analog operations, made them more immune to variation. This is because in a CMOS process, capacitance has much less variation than transistor V_t. Hence, the computation accuracy was improved.

- Support of 6-b precision for the inputs/outputs of the dot-products, allowed to achieve higher classification accuracy than 1-b inputs/outputs, for a neural network of a given size. We demonstrated $> 98\%$ accuracy for 10K test inputs on the MNIST dataset, compared to 90% achieved by prior in-memory approaches.

- With the highly parallel architecture (64 input, 16 outputs per cycle), $> 27\times$ benefits in energy-efficiency of the dot-product operations was achieved, compared to low-precision prior full-digital implementations.

5.1.3 Variation-Tolerant Read Sensing Architectures for Non-Volatile Resistive Memories

This work focused on read-sensing architectures for resistive memories, specifically STT-RAM, with a 1 transistor and 1 MTJ storage device (1T-1R) bit-cell structure.

- A pseudo-differential resistive divider based read-sensing scheme is proposed for the 1T-1R STT-RAM bit-cell array. This scheme compares the data voltage with 2 reference voltages to improve the signal margin, compared to a single reference scheme. The 2-phase architecture allows re-using some of the reference bit-cells in both phases, reducing the effect of their variation on the signal margin. An array architecture is proposed with shared reference rows and columns, to reduce their area overheads.

- An improved offset-cancellation (OC) technique is proposed for a sense-amplifier (SA), which uses inverters as both amplifiers and a latch. The proposed scheme fully cancels the mismatch in the trip points of the 2 inverters, by adding 2 small sampling capacitors and one extra phase, compared to the conventional OC technique. $3\times$ benefits in the sigma of the offset voltage was obtained in

simulation, compared to the conventional SA for iso-area. The proposed SA was implemented in a 14nm CMOS process and achieved a DC $\sigma_{offset} \sim 2$mV.

5.2 Future Work

With the exponential growth in data-intensive ML applications, e.g. neural networks, there are many opportunities in re-thinking the traditional memory design. As demonstrated in this thesis, for these applications, memory and computation do not necessarily need to be separated and huge benefits can be achieved by re-configuring the memory to implement in-place computations as well. Mentioned below, are some possible future areas to explore, which can expand on the ideas presented in this thesis.

- The proposed SRAM array for embedded dot-product computation currently supports usage of 1-b binary weights (+1/-1). It can be easily extended for 2-b ternary weights (+1/0/-1). However, it would be challenging and interesting to support more-precision in the weights, without compromising on the variation-tolerant read architecture. Supporting multi-bit weights could improve classification accuracy for a neural network of a given size.

- 6T bit-cells can be explored for in-memory dot-product computations instead of the 10T bit-cells used in this thesis. This would have obvious area advantage. However, techniques to not disturb the bit-cell during analog read operations should be investigated.

- The present Conv-RAM architecture implements the DAC, average and ADC operations in 3 consecutive phases. To reduce the clock period and improve the throughput, future work can also explore pipelining the DAC+average and ADC operations.

- Future work can also investigate into incorporating more computation capability with the same memory array, e.g. boolean logic, arithmetic operations etc. That could lead to more general purpose compute SRAMs.

- Future work can also explore ways to expand the in-memory computation capability to newer memory technologies, e.g. STT-RAM, ReRAM etc. These memories have the advantage of being non-volatile. Hence, the data stored (e.g. filter weights for neural networks) in them, can be retained even when they are powered off and do not be re-written once they are powered back up. Also, these memories have more density and some of them can store multiple bits in one cell. This could also be highly beneficial for neural networks, since all the filter weights can be stored on-chip, obviating the need for expensive off-chip DRAM accesses.

Appendix A

Energy Estimation for Digital Implementation of Binary-Weight CNNs

We estimate a rough energy model for the digital implementation of binary-weight CNNs, to compare with our in-memory computation approach. The architecture used for the estimation is shown in Fig. A-1, where the digital computation units are assumed to be situated next to the memory. We use the same size SRAM (256×64), as with the in-memory approach, to store the 1-bit filter weights. We assume an implementation, similar to the Conv-RAM, where each set of 64 inputs are re-used for 16 different filters. Hence, in each clock cycle, 64 1-bit filter weights (corresponding to a unique 3-D filter in a given CNN layer) are read. Then, they are multiplied by the corresponding 6-bit input values (X_{IN}'s), and the partial-products are added using a digital adder tree to get the dot-product output (Y_{OUT}). This is repeated for 16 filters, after which a new set of inputs (corresponding to a 1-D convolutional shift or to different input channels) are used for the computations. Hence, the total energy for a set of 16 filters, re-using the same set of 64-inputs, can be written as:

$$\begin{aligned} E_{dig} &= 16 \times (64 \times E_{SRAM,RD} + 64 \times E_{2s,COMP} + E_{Adder\_Tree}) \\ &= 16 \times (64 \times E_{SRAM,RD} + 64 \times E_{2s,COMP} + 72.5 \times E_{ADD}) \end{aligned} \tag{A.1}$$

where, $E_{SRAM,RD}$ is the average energy per bit access for a read operation of the SRAM (storing the filter weights), $E_{2s,COMP}$ is the average energy for the 1-b weight multiplication with the inputs (i.e. 2's complement computation of the inputs, X_{IN}'s) and E_{ADD} is the average energy for a 6-b pipelined adder. The energy of the adder tree is estimated using a linear relationship of the energy of an adder with its corresponding bit-width.

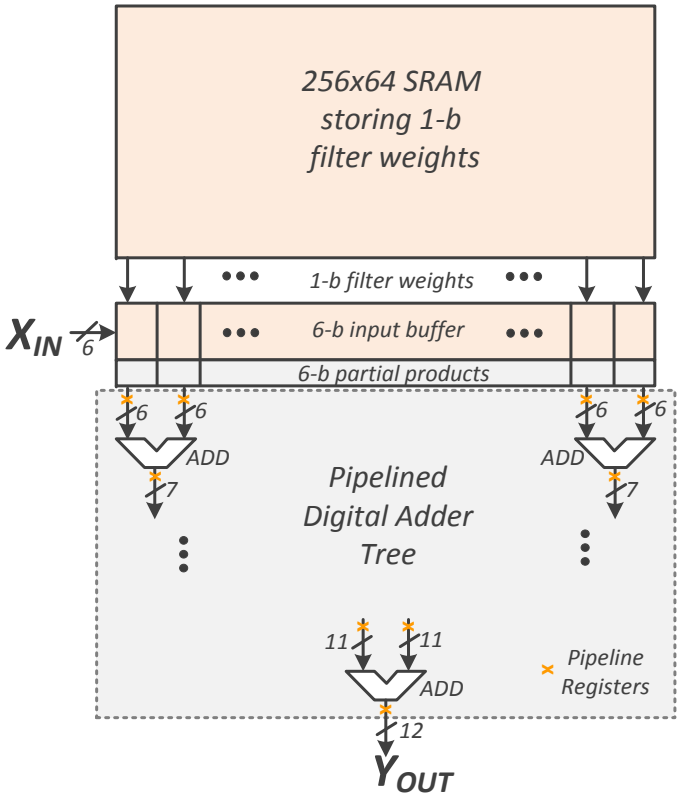


Figure A-1: Architecture for digital implementation of binary-weight CNNs.

Table A.1 shows the estimated energy for the different operations at 0.8V. $E_{SRAM,RD}$ is obtained from the measurements of the Conv-RAM read energy. An equal proportion of 0's and 1's are assumed for the 1-b weights. Hence, the SRAM would only consume bit-line switching energy for 50% of the cases. $E_{2s,COMP}$ is estimated from simulations with 25-50% activity factor. Similarly, the energy (E_{ADD}) for the 6-b pipelined adder is estimated from simulations with different sets of inputs.

Hence, from equation A.1 we can estimate the energy for doing 16 dot-products, each with 64 input pairs, using the traditional digital approach, as:

Table A.1: Energy (at V_{dd} = 0.8V) for the different digital operations, assumed in this estimation

$E_{SRAM,RD}$ ('1'): per bit	81fJ
$E_{SRAM,RD}$ ('0'): per bit	25fJ
$E_{SRAM,RD}$(avg): per bit	53fJ
$E_{2s,COMP}$: per 6-bit	20fJ
E_{ADD}: per 6-bit	80fJ

$$\begin{aligned} E_{dig} &= 16 \times (64 \times 0.053 + 64 \times 0.02 + 72.5 \times 0.08) \\ &= 167.6pJ \end{aligned} \tag{A.2}$$

Hence, it corresponds to an energy-efficiency of: $16 \times 64 \times 2OPs/167.6pJ$ = 12.2TOPS/W at V_{dd} = 0.8V. Compared to this, our measurements for the in-memory computation with the Conv-RAM (at 0.8V) achieved a peak energy-efficiency of 51.3TOPS/W for MNIST, which is $> 4\times$ better. This shows that our highly parallel in-memory analog computation approach is more energy-efficient than a near-memory digital implementation, even with binary filter weights.

Appendix B

Energy Model of Input/Output SRAMs for the LeNet-5 CNN

Here, we will estimate the SRAM energy required to read the inputs and write the outputs for the convolutions/dot-products, implemented with Conv-RAM (CSRAM). LeNet-5 CNN (shown in Fig. B-1) is used for the estimation. We will only analyze the CONV layer C3, which involves the maximum number of computations. The analysis can be easily extended for all the other layers of LeNet-5.

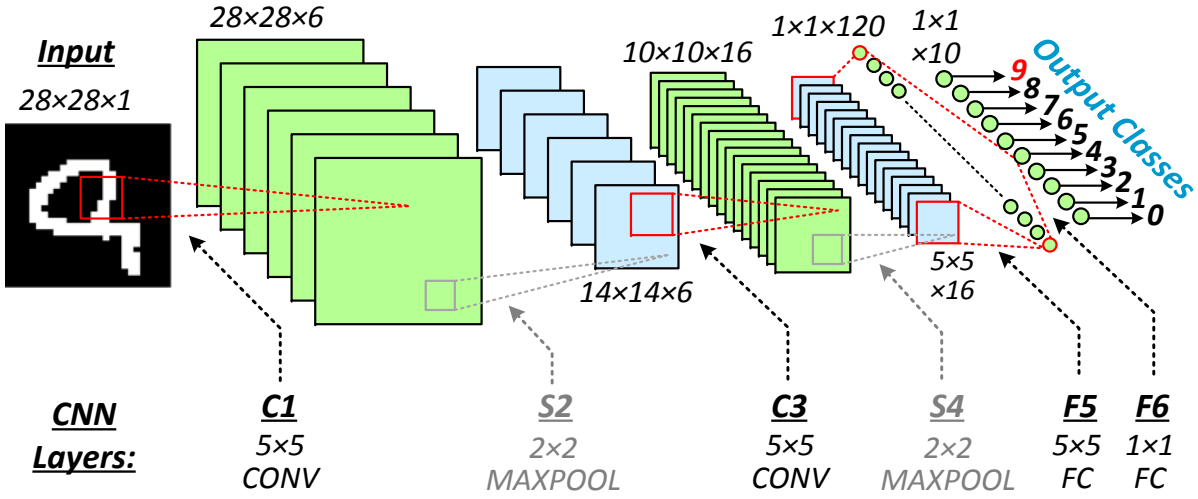


Figure B-1: Architecture of the LeNet-5 CNN, showing the sizes of the feature maps (top) and the filters (bottom).

The input feature map (IFMP) size, for the layer C3, is $14 \times 14 \times 6$, with 16 3-D

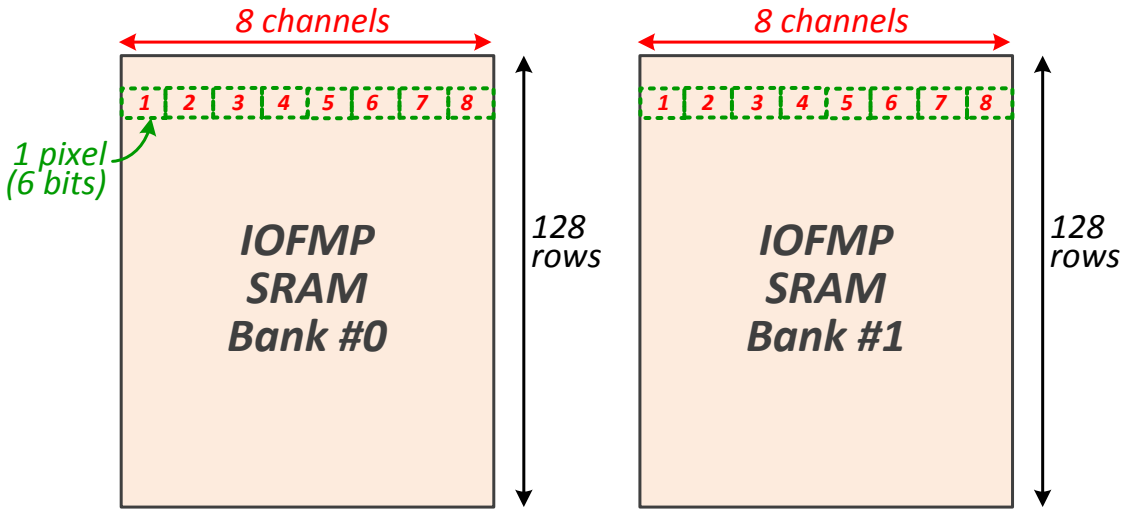


Figure B-2: SRAM architecture to store the IFMP/OFMP's, while processing the convolutions.

filters of size $5 \times 5 \times 6$. Hence, the 2-D output feature map (OFMP), corresponding to one filter, would have a size of $14-5+1 = 10$ elements in each direction. And, the 3-D OFMP size is $10 \times 10 \times 16$. Since, the sub-sampling layer S4, following CONV layer C3, reduces the size of the OFMP by $2\times$ in each 2-D direction, we only need to store $5 \times 5 \times 16 = 400$ elements/pixels. Fig. B-2 shows the structure of the SRAM, which can store the IFMP and OFMP elements. The IOFMP SRAM has 2 banks, one of which is accessed per cycle. Each row in a bank has 8 words (each 6-bit), corresponding to 8 different channels. For the IFMP of layer C3, each channel has $14 \times 14 = 196$ elements/pixels. Since, all the 6 (<8) IFMP channels can fit in 1 row, we need 196 rows to accommodate the entire IFMP. The OFMP (after sub-sampling) has $5 \times 5 = 25$ elements per channel. Since, each row can accommodate 8 channels, we need $25 \times (16/8) = 50$ rows, to fit all the 16 OFMP channels. Therefore, in total, we need 196+50 = 246 rows to fit both the IFMP and OFMP of layer C3. We assume 256 rows (128 rows per bank) to support this. It should be noted that, when processing one CNN layer at a time, we only need to store its corresponding inputs and outputs. The output of one layer is essentially the input to the next.

For estimating the energy for reading/writing the inputs/outputs for the convolutions, we assume an architecture with multiple CSRAM arrays working in parallel.

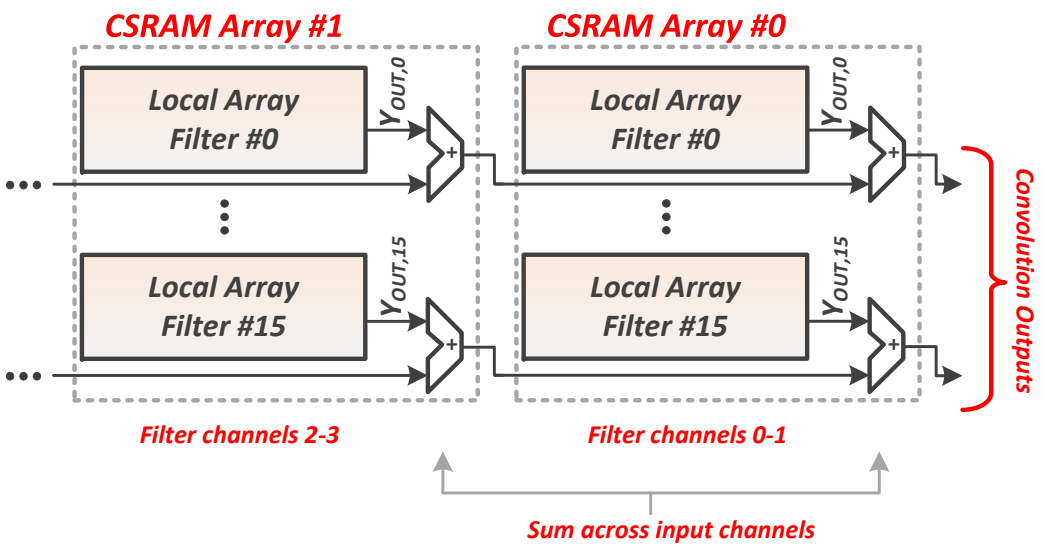


Figure B-3: Architecture of multiple CSRAM arrays working in conjunction, with 2 different input channels mapped to each CSRAM array (for CONV layer C3).

As shown in Fig. B-3, the partial convolution outputs (digital) from CSRAM array #1 (corresponding to 2 input channels: 2,3) can be passed on to the next CSRAM array #0 (corresponding to 2 other input channels: 0,1), where it is added to array #0's convolution output. In this way multiple partial convolution outputs (digital) can be combined together, before being written back to the IOFMP SRAM.

When processing a CONV layer, like C3, we can re-use most of the input feature map values when we evaluate the next output, corresponding to a 1-D shift. This is illustrated in Fig. B-4, showing for a 5×5 filter, only 5 new pixels (for each input channel) need to be read for the next computation cycle, for both horizontal and vertical shifts. A possible implementation for the IFMP FIFOs is also shown in Fig. B-4, to support the IFMP pixels re-use in the CONV layers. For vertical shifts, 5 new pixel values corresponding to a new row are fed to the input of the last FIFO. After 5 serial shifts, all the input FIFOs would have the updated IFMP values for the next Conv-RAM computation cycle. On the other hand, for vertical shifts, 5 new pixel values are parallely fed to the inputs of the individual 5-element FIFOs. With 5 FIFOs operating in parallel, one shift operation is needed to update the IFMP values for the next Conv-RAM computation cycle.

Now, 6 pixels for 6 different input channels can be simultaneously read from the

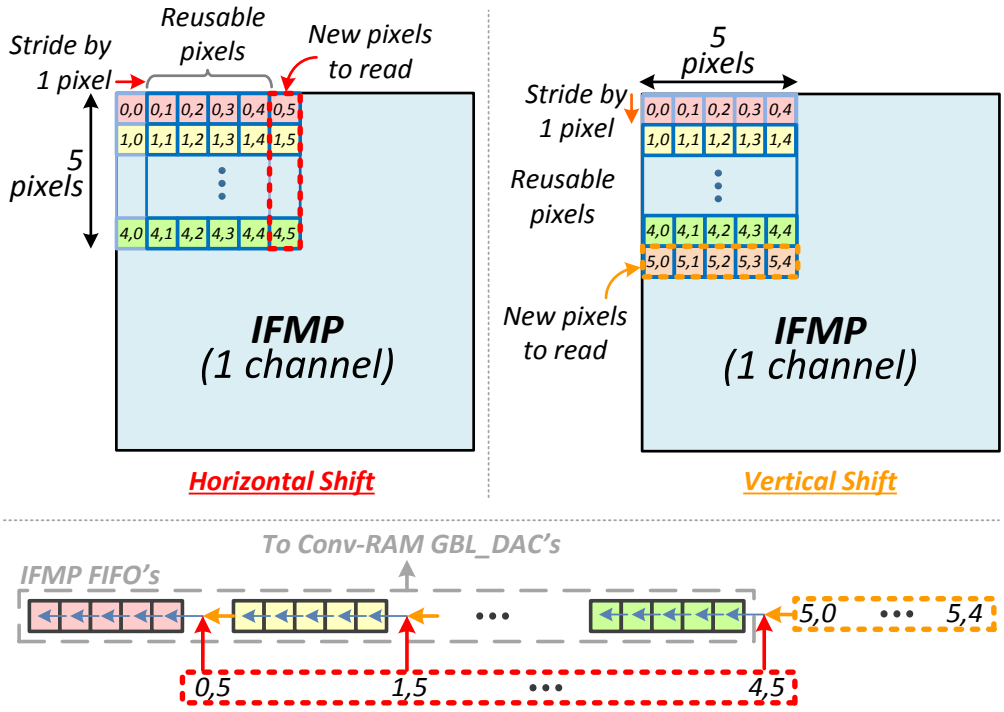


Figure B-4: IFMP pixels re-use when processing 2-D convolutions with sliding window (top), and the corresponding implementation for the IFMP FIFOs (bottom).

IOFMP SRAM per cycle and sent to 3 CSRAM arrays (each processing 2 channels as shown in Fig. B-3). Therefore, the IOFMP read energy is amortized by a factor of 3. Hence, the IFMP read energy, per computation cycle of 1 CSRAM array, can be expressed as:

$$E_{IFMP} = \frac{1}{3} \times 5 \times 8 \times 6 \times (p_0 \times E_{SRAM,RD,0} + p_1 \times E_{SRAM,RD,1}) \tag{B.1}$$

where, $E_{SRAM,RD,0}$ and $E_{SRAM,RD,1}$ are the read energy per bit of the SRAM for data '0' and data '1' respectively. The IOFMP SRAM is assumed to be data dependent [54, 39], i.e. it has lower energy consumption for one polarity of the data ('0' here) than the other ('1'). p_0 and p_1 are the average percentages of '0' and '1' bits in the input pixel values. We estimate the read energy from the Conv-RAM measured values at 0.8V (as shown in Appendix A), and calculate $p_0 = 79\%$ and $p_1 = 21\%$ from the measured input distributions for layer C3. Using those, we obtain:

$$E_{IFMP} = \frac{1}{3} \times 5 \times 8 \times 6 \times (0.79 \times 0.0125 + 0.21 \times 0.0405) \\ = 1.47pJ \tag{B.2}$$

For the OFMP, only 1 value needs to be written back per convolution output (Y_{OUT}) for 3 CSRAM arrays. In addition, while processing the 2-D convolutions, only 1 value needs to be stored for 4 neighboring OFMP pixels. This is because of the 2×2 sub-sampling/max-pooling operation (i.e. maximum of the 4 values) of layer S4, following CONV layer C3, in the LeNet-5 CNN. With 8 channels per IOFMP SRAM row, we need 2 cycles to write all the 16 CSRAM outputs corresponding to 16 different OFMP channels. Therefore, the OFMP write energy, per computation cycle of 1 CSRAM array, can be expressed as:

$$E_{OFMP} = \frac{1}{4} \times \frac{1}{3} \times 2 \times 8 \times 6 \times E_{SRAM,WR} \\ = 0.4pJ \tag{B.3}$$

where, $E_{SRAM,WR} = 50$fJ, is the write energy per bit, estimated from Conv-RAM measurements at 0.8V.

Therefore, the total estimated energy (at $V_{dd} = 0.8$V), for the IOFMP SRAM per computation cycle of 1 CSRAM array is = 1.47+0.4 = 1.87pJ, which is $\sim 6.2\%$ of the measured CSRAM energy of 30pJ at $V_{dd} = 0.8$V. This shows that, for the convolution implementation using CSRAM, the energy required to read and write the IFMP/OFMP values can be much lower than the energy of the CSRAM array, by taking advantage of data re-use. For bigger sized neural networks (with more input and output channels), the data re-use opportunity would be more, and hence, the IFMP/OFMP SRAM energy overhead can be further reduced.

Bibliography

[1] G. Moore, "Cramming More Components Onto Integrated Circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, Jan 1998.

[2] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 1701–1708.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097–1105.

[4] G. Hinton *et al.*, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov 2012.

[5] R. Sarikaya, G. E. Hinton, and A. Deoras, "Application of Deep Belief Networks for Natural Language Understanding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 4, pp. 778–784, April 2014.

[6] Y. K. Ramadass and A. P. Chandrakasan, "A battery-less thermoelectric energy harvesting interface circuit with 35 mv startup voltage," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 333–341, Jan 2011.

[7] H. Yamauchi, "Embedded SRAM Design in Nanometer-Scale Technologies," in *Embedded Memories for Nano-Scale VLSIs*, K. Zhang, Ed. New York: Springer, 2009.

[8] T. Singh *et al.*, "Zen: A next-generation high-performance ×86 core," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 52–53.

[9] S. M. Tam *et al.*, "SkyLake-SP: A 14nm 28-Core xeon® processor," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, Feb 2018, pp. 34–36.

[10] R. Riedlinger *et al.*, "A 32nm 3.1 billion transistor 12-wide-issue itanium processor for mission-critical servers," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, Feb 2011, pp. 84–86.

[11] N. P. Jouppi *et al.*, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17. New York, NY, USA: ACM, 2017, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/3079856.3080246

[12] W. A. Wulf and S. A. McKee, "Hitting the Memory Wall: Implications of the Obvious," *SIGARCH Comput. Archit. News*, vol. 23, no. 1, pp. 20–24, Mar. 1995. [Online]. Available: http://doi.acm.org/10.1145/216585.216588

[13] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb 2014, pp. 10–14.

[14] G. Kestor, R. Gioiosa, D. J. Kerbyson, and A. Hoisie, "Quantifying the energy cost of data movement in scientific applications," in *2013 IEEE International Symposium on Workload Characterization (IISWC)*, Sept 2013, pp. 56–65.

[15] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec 2017.

[16] T. Burd and R. Brodersen, "Design issues for Dynamic Voltage Scaling," in *Low Power Electronics and Design, 2000. ISLPED '00. Proceedings of the 2000 International Symposium on*, 2000, pp. 9–14.

[17] V. Gutnik and A. Chandrakasan, "Embedded power supply for low-power DSP," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 5, no. 4, pp. 425–435, Dec 1997.

[18] M. Khare *et al.*, "A high performance 90nm SOI technology with $0.992\mu m^2$ 6T-SRAM cell," in *Electron Devices Meeting, 2002. IEDM '02. International*, Dec 2002, pp. 407–410.

[19] P. Magarshack, P. Flatresse, and G. Cesana, "UTBB FD-SOI: A process/design symbiosis for breakthrough energy-efficiency," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, March 2013, pp. 952–957.

[20] N. Planes *et al.*, "28nm FDSOI technology platform for high-speed low-voltage digital applications," in *VLSI Technology (VLSIT), 2012 Symposium on*, June 2012, pp. 133–134.

[21] A. Carlson *et al.*, "SRAM Read/Write Margin Enhancements Using FinFETs," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, no. 6, pp. 887–900, June 2010.

[22] T. Song *et al.*, "A 14nm FinFET 128Mb 6T SRAM with VMIN-enhancement techniques for low-power applications," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, Feb 2014, pp. 232–233.

[23] B. Zimmer *et al.*, "SRAM Assist Techniques for Operation in a Wide Voltage Range in 28-nm CMOS," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 59, no. 12, pp. 853–857, Dec 2012.

[24] V. Chandra, C. Pietrzyk, and R. Aitken, "On the efficacy of write-assist techniques in low voltage nanoscale SRAMs," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, March 2010, pp. 345–350.

[25] J. Chang *et al.*, "A 20nm 112Mb SRAM in High-κ metal-gate with assist circuitry for low-leakage and low-VMIN applications," in *Solid-State Circuits Conference*

Digest of Technical Papers (ISSCC), 2013 IEEE International, Feb 2013, pp. 316–317.

[26] E. Karl *et al.*, "A 4.6 GHz 162 Mb SRAM Design in 22 nm Tri-Gate CMOS Technology With Integrated Read and Write Assist Circuitry," *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 1, pp. 150–158, Jan 2013.

[27] K. Nii *et al.*, "A 45-nm single-port and dual-port sram family with robust read/write stabilizing circuitry under dvfs environment," in *VLSI Circuits, 2008 IEEE Symposium on*, June 2008, pp. 212–213.

[28] H. Pilo *et al.*, "A 64 Mb SRAM in 32 nm High-k Metal-Gate SOI Technology With 0.7 V Operation Enabled by Stability, Write-Ability and Read-Ability Enhancements," *Solid-State Circuits, IEEE Journal of*, vol. 47, no. 1, pp. 97–106, Jan 2012.

[29] M. Yamaoka, K. Osada, and K. Ishibashi, "0.4-V logic library friendly SRAM array using rectangular-diffusion cell and delta-boosted-array-voltage scheme," in *VLSI Circuits Digest of Technical Papers, 2002. Symposium on*, June 2002, pp. 170–173.

[30] H. Fujiwara *et al.*, "A 20nm 0.6V 2.1μW/MHz 128kb SRAM with no half select issue by interleave wordline and hierarchical bitline scheme," in *VLSI Circuits (VLSIC), 2013 Symposium on*, June 2013, pp. C118–C119.

[31] K. Zhang *et al.*, "A 3-GHz 70MB SRAM in 65nm CMOS technology with integrated column-based dynamic power supply," in *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*, Feb 2005, pp. 474–611 Vol. 1.

[32] M. Sinangil, H. Mair, and A. Chandrakasan, "A 28nm high-density 6T SRAM with optimized peripheral-assist circuits for operation down to 0.6V," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, Feb 2011, pp. 260–262.

[33] S. Barasinski, L. Camus, and S. Clerc, "A 45nm single power supply SRAM supporting low voltage operation down to 0.6V," in *Solid-State Circuits Conference, 2008. ESSCIRC 2008. 34th European*, Sept 2008, pp. 502–505.

[34] A. Bhavnagarwala *et al.*, "A Sub-600-mV, Fluctuation Tolerant 65-nm CMOS SRAM Array With Dynamic Cell Biasing," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 4, pp. 946–955, April 2008.

[35] N. Verma and A. P. Chandrakasan, "A 65nm 8T Sub-Vt SRAM Employing Sense-Amplifier Redundancy," in *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, Feb 2007, pp. 328–606.

[36] B. H. Calhoun and A. Chandrakasan, "A 256kb Sub-threshold SRAM in 65nm CMOS," in *2006 IEEE International Solid State Circuits Conference - Digest of Technical Papers*, Feb 2006, pp. 2592–2601.

[37] H. Fujiwara *et al.*, "Novel Video Memory Reduces 45% of Bitline Power Using Majority Logic and Data-Bit Reordering," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 6, pp. 620–627, June 2008.

[38] M. Sinangil and A. Chandrakasan, "Application-Specific SRAM Design Using Output Prediction to Reduce Bit-Line Switching Activity and Statistically Gated Sense Amplifiers for Up to 1.9 × Lower Energy/Access," *Solid-State Circuits, IEEE Journal of*, vol. 49, no. 1, pp. 107–117, Jan 2014.

[39] C. Duan, A. J. Gotterba, M. E. Sinangil, and A. P. Chandrakasan, "Energy-Efficient Reconfigurable SRAM: Reducing Read Power Through Data Statistics," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 10, pp. 2703–2711, Oct 2017.

[40] H. Noguchi *et al.*, "A 10T Non-Precharge Two-Port SRAM for 74% Power Reduction in Video Processing," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI '07)*, March 2007, pp. 107–112.

[41] 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, no. 4, pp. 915–924, April 2017.

[42] M. Kang, S. K. Gonugondla, A. Patil, and N. R. Shanbhag, "A Multi-Functional In-Memory Inference Processor Using a Standard 6T SRAM Array," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, Feb 2018.

[43] Y. Zhang *et al.*, "Recryptor: A reconfigurable cryptographic cortex-m0 processor with in-memory and near-memory computing for iot security," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 4, pp. 995–1005, April 2018.

[44] K. C. Akyel *et al.*, "DRC^2: Dynamically Reconfigurable Computing Circuit based on memory architecture," in *2016 IEEE International Conference on Rebooting Computing (ICRC)*, Oct 2016, pp. 1–8.

[45] S. Jeloka, N. B. Akesh, D. Sylvester, and D. Blaauw, "A 28 nm Configurable Memory (TCAM/BCAM/SRAM) Using Push-Rule 6T Bit Cell Enabling Logic-in-Memory," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 4, pp. 1009–1021, April 2016.

[46] J. Daemen and V. Rijmen, *The design of Rijndael: AES — the Advanced Encryption Standard*. Springer-Verlag, 2002.

[47] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," *ArXiv e-prints*, Mar. 2016.

[48] I. Hubara *et al.*, "Binarized Neural Networks," in *Advances in Neural Information Processing Systems 29*, 2016, pp. 4107–4115.

[49] M. E. Sinangil, H. Mair, and A. P. Chandrakasan, "A 28nm high-density 6T SRAM with optimized peripheral-assist circuits for operation down to 0.6V," in *Proc. IEEE ISSCC*, Feb. 2011, pp. 260–262.

[50] J.-P. Noel *et al.*, "Multi-V_T UTBB FDSOI Device Architectures for Low-Power CMOS Circuit," in *IEEE Transactions on Electron Devices*, vol. 58, no. 8, August 2011, pp. 2473–2482.

[51] M. Yamaoka, R. Tsuchiya, and T. Kawahara, "SRAM Circuit With Expanded Operating Margin and Reduced Stand-By Leakage Current Using Thin-BOX FD-SOI Transistors," *Solid-State Circuits, IEEE Journal of*, vol. 41, no. 11, pp. 2366–2372, Nov 2006.

[52] "Specification for the Advanced Encryption Standard (AES)," Federal Information Processing Standards Publication 197, 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[53] M. S. M. Siddiqui, Z. C. Lee, and T. T. H. Kim, "A 16kb column-based split cell-VSS, data-aware write-assisted 9T ultra-low voltage SRAM with enhanced read sensing margin in 28nm FDSOI," in *2017 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, Nov 2017, pp. 165–168.

[54] A. Biswas and A. P. Chandrakasan, "A 0.36V 128Kb 6T SRAM with energy-efficient dynamic body-biasing and output data prediction in 28nm FDSOI," in *ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference*, Sept 2016, pp. 433–436.

[55] H. Fujiwara *et al.*, "A 20nm 0.6V 2.1 μ W/MHz 128kb SRAM with no half select issue by interleave wordline and hierarchical bitline scheme," in *2013 Symposium on VLSI Circuits*, June 2013, pp. C118–C119.

[56] S. Miyano *et al.*, "Highly Energy-Efficient SRAM With Hierarchical Bit Line Charge-Sharing Method Using Non-Selected Bit Line Charges," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 4, pp. 924–931, April 2013.

[57] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training Deep Neural Networks with binary weights during propagations," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 3123–3131.

[58] J. Sim *et al.*, "A 1.42TOPS/W deep convolutional neural network recognition processor for intelligent IoE systems," in *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, Jan 2016, pp. 264–265.

[59] B. Moons and M. Verhelst, "A 0.3-2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets," in *2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, June 2016, pp. 1–2.

[60] A. Biswas and A. P. Chandrakasan, "Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, Feb 2018, pp. 488–490.

[61] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," University of Toronto, Tech. Rep., April 2009.

[62] K. Ando *et al.*, "BRein Memory: A Single-Chip Binary/Ternary Reconfigurable in-Memory Deep Neural Network Accelerator Achieving 1.4 TOPS at 0.6 W," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 4, pp. 983–994, April 2018.

[63] S. K. Gonugondla, M. Kang, and N. Shanbhag, "A 42pJ/decision 3.12TOPS/W robust in-memory machine learning classifier with on-chip training," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, Feb 2018, pp. 490–492.

[64] B. Razavi, "The StrongARM Latch [A Circuit for All Seasons]," *IEEE Solid-State Circuits Magazine*, vol. 7, no. 2, pp. 12–17, Spring 2015.

[65] P. N. Whatmough *et al.*, "A 28nm SoC with a 1.2GHz 568nJ/prediction sparse deep-neural-network engine with >0.1 timing error rate tolerance for IoT applications," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 242–243.

[66] B. Moons and M. Verhelst, "An Energy-Efficient Precision-Scalable ConvNet Processor in 40-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 903–914, April 2017.

[67] B. Moons *et al.*, "BinarEye: An always-on energy-accuracy-scalable binary CNN processor with all memory on chip in 28nm CMOS," in *2018 IEEE Custom Integrated Circuits Conference - (CICC)*, April 2018.

[68] J. Lee *et al.*, "UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, Feb 2018, pp. 218–220.

[69] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable Convolutional Neural Network processor in 28nm FDSOI," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 246–247.

[70] Y. Chen and C. Petti, "ReRAM technology evolution for storage class memory application," in *2016 46th European Solid-State Device Research Conference (ESSDERC)*, Sept 2016, pp. 432–435.

[71] Y. Lu *et al.*, "Fully functional perpendicular STT-MRAM macro embedded in 40 nm logic for energy-efficient IOT applications," in *2015 IEEE International Electron Devices Meeting (IEDM)*, Dec 2015, pp. 26.1.1–26.1.4.

[72] A. Jog *et al.*, "Cache revive: Architecting volatile STT-RAM caches for enhanced performance in CMPs," in *DAC Design Automation Conference 2012*, June 2012, pp. 243–252.

[73] E. Kltrsay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating STT-RAM as an energy-efficient main memory alternative," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, April 2013, pp. 256–267.

[74] M. Jefremow *et al.*, "Time-differential sense amplifier for sub-80mV bitline voltage embedded STT-MRAM in 40nm CMOS," in *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, Feb 2013, pp. 216–217.

[75] C. Kim *et al.*, "A covalent-bonded cross-coupled current-mode sense amplifier for STT-MRAM with 1T1MTJ common source-line structure array," in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, Feb 2015, pp. 1–3.

[76] E. Chen *et al.*, "Advances and Future Prospects of Spin-Transfer Torque Random Access Memory," *IEEE Transactions on Magnetics*, vol. 46, no. 6, pp. 1873–1878, June 2010.

[77] J. P. Kim *et al.*, "A 45nm 1Mb embedded STT-MRAM with design techniques to minimize read-disturbance," in *2011 Symposium on VLSI Circuits - Digest of Technical Papers*, June 2011, pp. 296–297.

[78] A. Biswas, U. Arslan, F. Hamzaoglu, and A. P. Chandrakasan, "An offset-cancelling four-phase voltage sense amplifier for resistive memories in 14nm CMOS," in *2017 IEEE Custom Integrated Circuits Conference (CICC)*, April 2017, pp. 1–4.

[79] B. Giridhar, N. Pinckney, D. Sylvester, and D. Blaauw, "A reconfigurable sense amplifier with auto-zero calibration and pre-amplification in 28nm CMOS," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb 2014, pp. 242–243.

[80] N. Verma and A. P. Chandrakasan, "A High-Density 45nm SRAM Using Small-Signal Non-Strobed Regenerative Sensing," in *2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, Feb 2008, pp. 380–621.