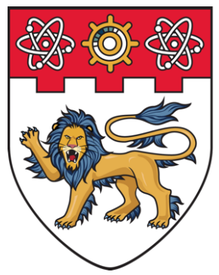


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Low-power Circuits for Neuromorphic Vision Sensor based Internet of Video Things

ZHANG Xueyong

School of Electrical and Electronic Engineering

2021

Low-power Circuits for Neuromorphic Vision Sensor based Internet of Video Things ZHANG Xueyong 2021

Low-power Circuits for Neuromorphic Vision Sensor based Internet of Video Things

Zhang Xueyong

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2021

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

26-07-21

.....

Date

.....

Zhang Xueyong

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

26-07-21

.....

Date

.....

Prof. Gwee Bah Hwee

Authorship Attribution Statement

This thesis contains material from 3 publications published/ accepted in the following peer-reviewed journals/ conferences in which I am listed as an author.

Chapter 3 contain parts of material published as **X. Zhang**, V. Mohan and A. Basu, "CRAM: Collocated SRAM and DRAM With In-Memory Computing-Based Denoising and Filling for Neuromorphic Vision Sensors in 65 nm CMOS," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 5, pp. 816-820, May 2020.

The contributions of the co-authors are as follows:

- A/Prof Arindam Basu provided the initial project direction and edited the manuscript.
- I contributed to proposing the innovation idea and designing the circuit, characterizing the chip, writing the codes, designing the experiments, interpreting the results, and drafting and revising the manuscript.
- Vivek Mohan contributed to proving the data set for testing.

Chapter 4 is accepted as a provisional patent as **X. Zhang** and A. Basu, "A High Efficient Region Proposal Algorithm and Hardware Implementation in 65 nm CMOS for Computer Vision Applications,". The contributions of the co-authors are as follows:

- A/Prof Arindam Basu provided the initial project direction and reviewed the manuscript drafts.
- I conceptualized, designed and characterized the integrated circuit. I also wrote the testing codes, carried out the experiments, and wrote the manuscript drafts.

Chapter 5 contains parts of material published as **X. Zhang**, J. Acharya and A. Basu, "A 0.11–0.38 pJ/cycle Differential Ring Oscillator in 65 nm CMOS for Robust Neurocomputing," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 2, pp. 617-630, Feb 2021.

The contributions of the co-authors are as follows:

- A/Prof Arindam Basu provided the initial project direction and edited the manuscript.
- I contributed to proposing the innovation idea and designing the circuit, characterizing the chip, writing the testing codes, designing the experiments, interpreting the results, and drafting, and revising the manuscript.

- Jyotibdha Acharya contributed to neural network simulation and the corresponding manuscript.

26-07-21

.....

Date

.....

Zhang Xueyong

Acknowledgements

I owe deep thanks to many people for their support and help of my research and career journey. First and foremost, I am deeply grateful to my supervisor, Prof. Arindam Basu, for his continuous guidance, support, and suggestion throughout my Ph.D. and research associate in NTU. Without his patient guidance and support, I would not have been able to finish my Ph.D. study and have led an enriched research life. He is always considerate and kind. Due to his trust in me, I have been allowed to explore various research topics. I am amazed and inspired by his immense inter-disciplinary knowledge as well as the endless passion for research. He not only improved my problem-solving skills but also taught me to communicate with others effectively. I am incredibly fortunate to have him as a mentor in both academic and non-academic life. He is really a great role model, and I am forever thankful to him for the opportunity he provided in the past years.

Special thanks to my supervisor Prof. Gwee Bah Hwee for the final support of my thesis submission and all his kind help for the last journey of my Ph.D. period.

I want to thank Dr. Chen Yi, Dr. Wang Zeng, Dr. Bapi Kar, and Sumon Kumar Bose for their supports, encouragement, and contributions to my work. Dr. Chen Yi and Dr. Wang Zeng gave me a lot of help at the beginning of my research here. Dr. Bapi Kar and Sumon broadened my circuit design horizon by introducing me to the digital flow, especially back-end physical design. I would also like to acknowledge and thank Lei Zhang, Sandeep, Aakash Patil, Vivek Mohan, Deepak Singla, Shoeb, Nimesh, and Pradeep for their contributions and collaborations.

I would also like to thank Kok Seng, Du Yanxian from Singapore Technologies Electronics Engineering (STEE) and collaborators from National University of Singapore (NUS) for the project support. I also want to thank my friends and colleagues for making my journey at NTU enjoyable, unforgettable and fun. Thanks also to the staffs in VIRTUS IC Design Centre for their help with the technical and administrative support.

Finally, I am always indebted to my parents for their unconditional love and care. I also extend my gratitude to the whole family and all my relatives for their continued care and support.

Xueyong, April 2021

Abstract

There has been a tremendous growth in the number of sensors under the paradigm of the Internet of Things (IoT) spurred by the advent of 5G communication. Among such sensors, video cameras hold a special role due to their rich information content. However, due to the huge volume of such data, it requires a special paradigm — internet of video things (IoVT) — to process and handle scalability of such networks. This method leverages the huge success of deep learning to use machine learning accelerators at the sensor node to perform "Edge" computing thus reducing the wireless transmission bottleneck. However, von Neumann architectures take unbearable energy and latency costs for deep learning accelerator hardware, because of the separation of data storage device and computing unit. The increasing demands on memory storage capacity and computational capability make it challenging to deal with huge data on resource limited platforms such as portable products and remote sensory devices.

In computer vision (CV) applications for traffic surveillance and monitoring, image frames from a camera undergo several processing steps such as image denoising, region proposal, object classification, and object tracking. However, implementation of this data-intensive computing based on traditional von Neumann architecture involves a huge energy dissipation and more tedious execution time due to the enormous data movement between computing unit and storage block. Neuromorphic vision sensors (NVSs) hold promises for such applications due to its ability to reduce data at the source by optimal sampling. NVS only records active event data and ignore the stationary background reducing the redundant data significantly. Recent work has shown a hybrid frame-event approach that processes event based binary images (EBBI) created out of events from a NVS allows for efficient denoising and region proposal (RP) operations. However, no hardware implementation was reported.

To overcome the bottleneck of the system performance and implement the integrated circuits for image or video processing in the application of traffic surveillance, near

memory computing and in memory computing were proposed to reduce or even remove the data movement between memory device and processor unit. We first proposed a novel Collocated Random Access Memory (CRAM) based analog in memory computing (IMC) architecture to achieve parallel image denoising and image filling. The process of image denoising and filling is essential for NVS because of the inherent random noise (spurious events) due to thermal noise in transistors, shot noise and junction leakage current of the photodiode etc. The proposed approach is tested with the binary image frames from a Dynamic and Active-pixel Vision Sensor (DAVIS) setup and achieves around 10000× lesser energy cost compared to conventional non-IMC approach in the same process (in 65 nm CMOS). The fully parallel natural diffusion architecture reduces the processing time at least to 20 ns and average power consumption to 170 pJ per frame, leveraging large throughput and energy efficiency.

The second part of this thesis explores low power region proposal (RP) algorithms and hardware implementations. We propose an edge event driven RP (EEDRP) approach with programmable parameters for the event-based binary image to exploit spatial redundancy in the valid frames. The proposed EEDRP network can quickly find out the bounding boxes of each object in the image to reduce the computation complexity of the successive deep neural network (DNN) by confining the computing region to the proposed bounding boxes instead of the whole image frame. The EEDRP algorithm can realize near-memory computing by reading the image memory and processing locally. By scanning the whole memory array once, the RP computation happens only when the edge event (rising edge or falling edge) is detected. The EEDRP enable us deal with images even with noises and holes and it is also tolerant of fragmented objects because it will merge objects if the distance is less than the configured parameter. All the parameters can be programmable for different application scenarios. The EEDRP algorithm performs a high accuracy, high energy efficient and low latency region proposal than the traditional connected component labeling (CCL) algorithm. Simulated in 65 nm CMOS, this chip produces up to 15 region proposals per frame and achieves $\sim 580\times$ energy savings compared to the digitally implemented CCL algorithm and throughput of 2.6 $frames/msec$ at 200 MHz. We also proposed axes projection based RP (APBRP) to further reduce energy and time cost. It achieves $\sim 1767\times$ faster than the CCLRP implementation thanks to the parallel in memory computing technique. From measurement results, the in-memory computing based APBRP is $\sim 2700\times$ more energy efficient than

the near-memory based EEDRP. The weighted $F1$ scores of both EEDRP and APBRP achieve 2.55× and 1.7× better than the conventional HISTRP and CCLRP, respectively.

The image in these regions have to be next classified by neural networks and neuromorphic implementations that utilize analog or physical computing are promising and have been known to be energy efficient compared to digital baselines. Neuro-inspired spiking neural networks (SNN) have also gained popularity due to the promise of sparse activation leading to lower energy dissipation. In recent years, time-based computational circuits for DNN/SNN are gaining popularity due to the reduced power supply in scaled CMOS. An important building block in these designs is a digital delay cell. For example, it is used to create an oscillator that can convert analogue current to digital output (rate based neuron) or be used as an integrate and fire neuron with bio-plausible refractory period and spike frequency adaptation features.

This thesis also explored an energy- and area-efficient full differential CMOS current controlled ring oscillator (CCO) as a suitable and compact structure in neural network applications. The neuronal oscillator achieves higher frequency while consuming lower area and lower energy due to less transistors are utilized compared with the conventional structure. By eliminating the unnecessary transistors, the proposed structure is composed of a simplest dynamic positive feedback latch and differential pairs, saving 25% area in size. The CCO can be tuned by both input voltage and external variable resistor. The measurement results show our work achieves 11% frequency improvement and 13% energy-efficient without degrading the jitter and phase noise characteristics.

In summary, we presented a set of algorithms and hardware solutions for energy efficient neuromorphic circuits that use near/ in-memory computing techniques and time base computing approach. We have demonstrated the testing results and performance in the application of traffic monitoring.

Contents

List of Figures

List of Tables

Acronyms

ACC	Accumulator
ADC	Analog to Digital Converter
ADS	Automated Driving System
AER	Address Event Representation
AI	Artificial Intelligence
ALU	Arithmetic Logic Unit
ANN	Artificial Neural Network
APBRP	Axes Projection Based Region Proposal
APS	Active Pixel Sensor
ASIC	Application-Specific Integrated Circuit
ASP	Analog Signal Processing
ATIS	Asynchronous Time Based Image Sensor
AV	Autonomous Vehicles
BL(B)	Bit Line (Bar)
BNN	Binary Neural Network
BS	Bank Select
CAM	Content Addressable Memory
CCD	Charge Coupled Device
CCL	Connected Component Labeling
CCO	Current Controlled Oscillator
CMFB	Common Mode Feedback
CMOS	Complementary Metal Oxide Semiconductor
CMP	Comparator
CMRR	Common Mode Rejection Ratio
CNN	Convolutional Neural Network
CNT	Counter

CPU	Central Processing Unit
CRAM	Collocated SRAM and DRAM
CSRO	Current Starved Ring Oscillator
CV	Computer Vision
DAC	Digital to Analog Converter
DAVIS	Dynamic and Active Pixel Vision Sensor
DC	Delay Cell
DNN	Deep Neural Network
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processing
DVS	Dynamic Vision Sensor
EBBI	Event Based Binary Image
ED	Edge Detector
EEDC	Edge Event Driven Computing
EEDC	Edge Event-driven Computing
EEDRP	Edge Event Driven Region Proposal
FIFO	First In First Out
FOM	Figure of Merit
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
GT	Ground Truth
GWL	Global Word Line
HDC	Hysteresis Delay Cell
HISTRP	Histogram based Region Proposal
HMI	Human Machine Interface
I/O	Input/Output
IC	Integrated Circuit
IMC	In Memory Computing
IoT	Internet of Things
IoVT	Internet of Video Things
LP	Low Power
LWL	Local Word Line
MAC	Multiply and Accumulate
MC	Monte Carlo
MCU	Micro Controller Unit

ML	Machine Learning
MOM	Metal Oxide Metal
NN	Neural Network
NVM	Non-Volatile Memory
NVS	Neuromorphic Vision Sensor
PC	Personal Computer
PCM	Phase change memory
PD	Projection Detector
PI	Phase Interpolator
PISO	Parallel In Serial Out
PL	Projection Line
PSRR	Power Supply Rejection Ratio
PUD	Pull Down Network
PUN	Pull Up Network
(Q)VGA	(Quarter) Graphics Display Resolution
RO	Ring Oscillator
ROI	Region of Interest
RP	Region Proposal
RRAM	Resistive-RAM
SA	Sense Amplifier
SC	Switched-Capacitor
SNN	Spiking Neural Network
SRAM	Static Random Access Memory
STT-RAM	Spin-Transfer Torque RAM
TBC	Time Based Computing
TBC	Time-based Computing
TDC	Time to Digital Converter
TDNN	Time-domain Neural Network
TOPS	Tera Operations per Second
TPU	Tensor Processing Unit
UGS	Unattended Ground Sensor
VCO	Voltage Controlled Oscillator
VLSI	Very Large Scale Integration
WL	Word Line

Chapter 1

Introduction

1.1 Background

Artificial intelligence (AI) and machine learning (ML) have changed the way we interact with the world around us. Image recognition and classification make it possible to detect, locate and identify objects from a huge volume of image sets or video records more rapidly. Deep neural networks (DNNs) represent the state-of-the-art solution in virtually all relevant tasks in machine learning across an incredibly diverse variety of domains [1], [2]. The advantage deep networks provide over shallow ones is the ability to extract hierarchies of abstracted features from the underlying input data, giving rise to hierarchical data transformations which have been optimized for a specific task.

Convolutional neural networks (CNNs), one kind of deep neural networks, have provided significant leaps in accuracy on difficult computer vision benchmarks such as ImageNet [3], [4], currently reaching performance levels that rival humans [5], [6]. However, state-of-the-art CNNs require tens to hundreds of megabytes of parameters on billions of operations in a single interface pass, creating significant data movement (from on-chip and off-chip to support the computation) and intensive computation. These lead to not only huge energy consumption but also large latency.

Huge energy consumption makes it difficult to implement such CNNs in limited energy budget scenarios, such as edge devices and portable devices. By deploying machine intelligence to more and more devices around us, the Internet of Things

(IoT) and Internet of Video Things (IoVT) require "edge computing", i.e. computing data on these edge devices locally instead of processing on the "cloud". There are some considerations for this. First, "edge computing" enables the remote devices to make fast decisions locally, without having to wait for the "cloud". Second, it can significantly reduce the communication traffic to the "cloud", by only sending the critical and relevant information while filtering out the rest of 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" [7].

Another challenge caused by the massive data movement is higher latency, which is undesirable and unacceptable for cases that need real-time information, such as automated driving system (ADS), human–machine interface (HMI), and healthcare applications. To overcome these challenges, there is increasing interest in developing new brain-inspired hardware architectures termed "neuromorphic". A primary purpose of neuromorphic hardware research is to mimic extremely energy efficient neuro-biological architectures using very-large-scale integration (VLSI), and some recent technological breakthroughs make such vision to be possible [8].

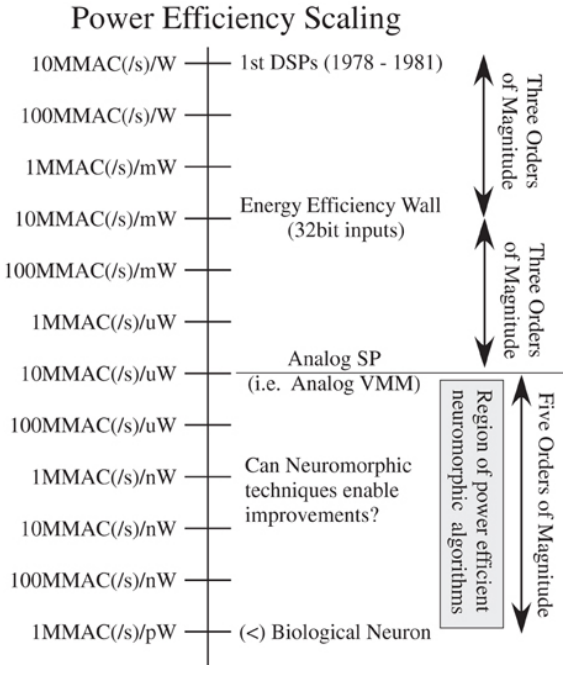


Figure 1.1: A spectrum showing the computational efficiency of various technologies, including digital signal processing (DSP), analog signal processing (ASP), as well as best estimate of biological neuron computation. Adopted from [8].

Fig. 1.1 shows the estimated peak computational energy efficiency for different techniques. As we can see, the digital signal processing (DSP) systems have the lowest power efficiency and there is an energy efficiency wall for digital systems. While analog signal processing (ASP) exhibits an ~ 6 orders higher power efficiency, neuromorphic techniques are possibly a more promising approach to further improve power efficiency. Computational power efficiency for biological systems is 8–9 orders of magnitude higher (better) than the power efficiency wall for digital computation [8]. Enormous potential for neuromorphic systems indicates the community has a lot of room for improvement, as well as potential directions on how to achieve these improvements.

1.2 The Rise, Fall and Rise of Neuromorphic Systems

Hardware implementation, especially dedicated hardware for specific applications, is a critical and promising part for the future development of artificial intelligence. State-of-the-art hardware for machine learning such as the central processing unit (CPU), graphics processing unit (GPU) and tensor processing unit (TPU) are built upon complementary metal oxide semiconductor (CMOS) transistors. Although superior computing capability has been demonstrated with such hardware, the end of transistor scaling and the separation of logic operation and memory units in the von Neumann architecture limit overall performance, in particular energy efficiency and execution time, especially for data intensive computing. Inspired by the extremely low power consumption of the human brain, neuromorphic hardware has been a promising research topic [9–12].

In the field of neuromorphic engineering, researchers study novel computing techniques based on the biological architecture and hope eventually to mimic human cognition and understand more about the human brain. Unlike digital signal systems where computers can execute many different applications by processing different software programs, analog circuits have traditionally been hardwired to address a single dedicated application. For example, energy-efficient analog circuits are widely used in portable devices for a number of specific functions, including sensing ambient environment signals [13], amplifying the sensed weak signals [14],

and controlling battery power [15], [16]. Since analog circuits do not have to process binary codes as digital computers, their computation performance can be both faster and require much less power.

Brain-inspired hardware has been investigated to some extent since the dawn of computing. However, the concept of neuromorphic systems, where the hardware directly emulates some aspects of biological systems, started in the 1980s with circuits designed to use analog CMOS circuits to emulate neuron dynamics [17]. Neuromorphic sensors processing artificial perceptive functions inspired by biological systems with analog electronic circuits have made the greatest strides in recent years. These circuits carry asynchronous timing information similar to the signed spikes in the nervous systems, and they respond in real time due to parallel computing. For example, silicon retina mimics human vision [18–20] and silicon cochlea models human hearing [21–23]. These spiking neuromorphic hardware aimed to get performance benefits primarily through analog processing, collocate memory with compute and lastly through event-driven communication. However, traditional analog circuits are less flexible and vulnerable to signal disturbance and noise interference issues. Moreover, the design automation process for digital circuits aided their rapid growth in the 90s and early 2000s. This resulted in a reduction in interest in neuromorphic systems around 2010 as seen from Google Trends [12].

However, because presently the CMOS technology scaling has saturated and is stalling, conventional digital computing techniques are reaching physical limits. Furthermore, with the advent of new computing loads related to deep neural networks at both server scale and edge nodes, there is a need to look for new computer architectures. Analog computing is making a comeback now under the name of "In-memory computing" (IMC) where analog processing is applied at the periphery of the memory array storing the weights of the neural network [7], [24]. These methods enable performing essential operations of the neural network without having to read out the weights, thus reducing latency and energy dissipation. Furthermore, principles of event-based computing are also being used with conventional digital circuits to reduce energy overheads and to implement new computational models of spiking neural networks, such as the SpiNNaker [25], TrueNorth [26] and LOIHI [27] neural chips. For short term benefits, these digital

event driven architectures hold promise [28] while for a longer term, novel memory materials may hold the key for improving density and energy efficiency [29–32].

Apart from solving engineering applications, these silicon systems are well suited as a tool to help understand neural computation. Recent innovative analog-based techniques could make it possible to build a practical neuromorphic computer [33], [34]. A core technological hurdle in this field involves the electrical power requirements of computing hardware. Although a human brain functions on a mere 20 watts of electrical energy [35], a digital computer that could approximate human cognitive abilities would require tens of thousands of integrated circuits (ICs) and a hundred thousand watts of electricity or more – levels that exceed practical limits.

1.3 Proposed System and Application

1.3.1 Application: Intelligent Traffic Surveillance and Unattended Ground Sensors

AI or ML is important and changing the world around us. Among its applications, the vision-related ones are gaining significant improvement and impacting visual object recognition, tracking, classification, etc. However, it is important to reduce energy dissipation in intelligent vision systems so that they can be deployed on mobile phones or battery powered remote devices.

The recent enormous advances in computing and networking technologies have enabled the wide deployment of Internet of Video Things (IoVT) systems in our environment, which is seen as the next generation IoT with visual sensors [36]. Visual sensors have been booming in IoT application due to their capability of providing richer and more versatile information. Surveillance cameras have been around every corner around the world. Visual data can serve as the foundation for smart cities, physical security, smart transportation, and many other types of applications. Installed on vehicles, cameras are utilized to monitor the environments surrounding the moving vehicles. While the traffic surveillance cameras as sensor nodes are deployed along the roadside to monitor the intersections and critical locations for traffic intelligence. These traffic monitoring sensors and real time data processing capability make the autonomous vehicles (AV) feasible. AV technology

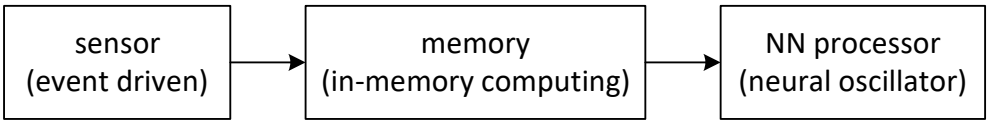


Figure 1.2: The blocks of the system proposed and studied in this thesis.

promises a more efficient transportation solution by optimising road usage and saving manpower.

Another critical application is the unattended ground sensors (UGS) used in defence and security scenarios. Intrusion detection and perimeter surveillance are key ingredients to successful defense of the homeland during the new era of international terrorism. Unattended ground sensors have proven to be invaluable in various military missions [37]. Neuromorphic vision sensors (NVSs) are suitable for monitoring troop movement and weapons of mass destruction in remote area. Merits of neuromorphic hardware in asynchronous triggering and parallel computing techniques are making larger scale applications feasible.

1.3.2 Proposed System

This thesis proposes to deal with the energy efficiency problem by using an event-driven sensor that reduces data at the source and novel computing paradigms (such as time based and in-memory computing) in order to reduce the energy consumption of the following processor. Near- and in-memory computing, as well as time-based computing techniques, are deployed for energy- and time-efficient applications.

This thesis proposed to develop novel low energy solutions for intelligent traffic monitoring systems for vehicle classification, achieved with innovations at both the sensor and the processor level. Fig.1.2 shows the block diagram of the proposed system. To reduce the invalid or redundant raw data and improve the dynamic range, the event-driven NVS is used to capture and translate the spatial and temporal view to a stream of digital signals. The raw image data are sent to our dedicated memory array, where the image denoising and filling are achieved by in-memory computing (IMC) with ultra low energy and latency. After IMC for image pre-processing, a better quality of image is obtained. Then our IMC-based region proposal (RP) approach proposes the region of interests (ROIs) for the following processing. For

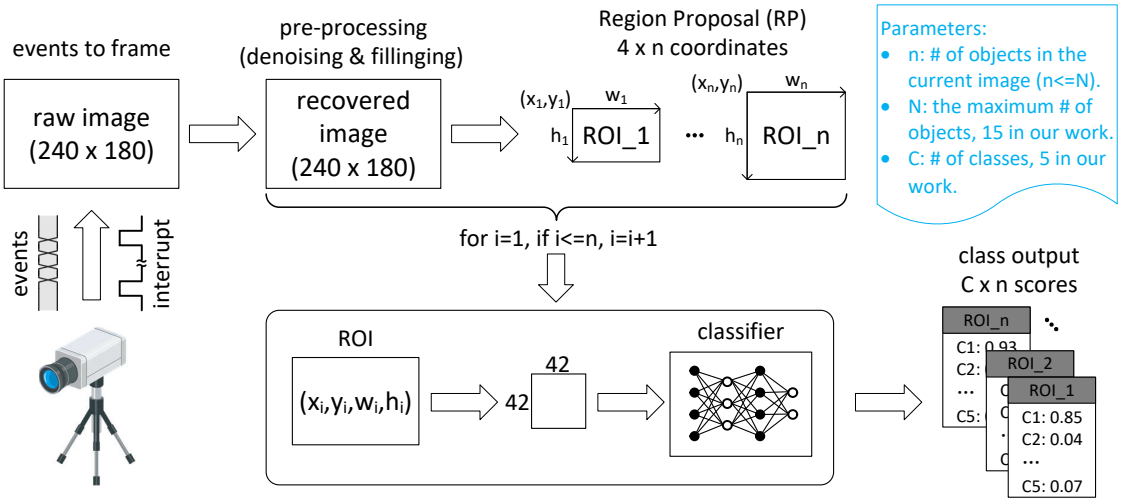


Figure 1.3: Detailed block architecture of the EBBI processing system. The events are converted to frame during a time interval t_F. The raw frame with resolution of 240 × 180 pixels is further restored by the pre-processing block where noise is removed and holes are filled. Based on the restored image, the RP algorithm groups pixel clusters to form a 'region' and provides 4 coordinates information of each RP boundary box. Finally, the regions are scaled to a fixed size of 42 × 42 pixels and then fed to a classifier to identify the class present.

the neural network (NN) processor, a novel neural oscillator architecture is used to benefit from the merits of time-based computing.

Fig.1.3 presents the detailed data flow and data format of the proposed system, from the event-based neuromorphic vision sensor (NVS) to the final classifier. For event-based cameras, "event" or "spike" means the pixel bright changes. If the bright change overcomes a threshold, an ON (going brighter) or OFF (going darker) event is generated [20]. The benefits of the event-based NVS are: very high temporal resolution and low latency (both in the order of microseconds), very high dynamic range (140 dB vs. 60 dB of standard cameras), and low power consumption [38]. Event-based binary image (EBBI) is created by a hybrid approach [39] during the time interval $t_F = 66ms$. The raw image captured by the NVS inherently contains lots of noise pixels characterized by abnormal firing rates because of a variety of factors such as thermal noise in transistors, shot noise and junction leakage current in some of the photodiodes [40–43]. Furthermore, as NVS pixel size shrinks, it becomes more susceptible to shot noise and junction leakage, which causes information-less noise events [43]. Besides, a unique disadvantage of event-based sensors is that they fragment objects in the captured images due to lack of events generated by smooth areas such as glass windows or due to occlusion by a stationary

object. This often happens in traffic monitoring application because big vehicles like a bus have a lot of plane surface on their sides that do not generate much events [44]. Hence, to leverage the benefit of event based sensor and improve the performance of the classifier, the raw image has to be restored by the pre-processing block. We propose a novel in-memory computing based image denoising and image filling to remove noise and fill holes [45]. The restored image can be then used for the subsequent processing. An edge event driven region proposal (EEDRP) is proposed to search valid region of interests (ROIs) within the whole image. To further reduce the energy consumption and execution time, we proposed the axes projection based region proposal (APBRP) approach leveraging the parallel in-memory computing technique. RP block groups pixel clusters to form a 'region' and provides 4 coordinates information (x, y, w, h) or $(x_{start}, y_{start}, x_{stop}, y_{stop})$ of each RP boundary box. Finally, the regions are scaled to a fixed size of 42×42 and then fed into a classifier to identify the class of each region. In the NN, we explored energy-efficient time based circuitry for robust neurocomputing [46]. Only the ROI with valid object is fed into the classifier, not only dramatically reducing the computes and energy consumption but also improving the accuracy of classifier.

The traffic monitoring system will be deployed in remote area for the scenario of surveillance without power and wired internet. This challenges the energy- and computing- efficiency of the overall system. Also, in most cases, the traffic is very sparse and therefore no valid data needs to be processed. So it is critical for the always-on block to consume very low energy. The computation block is woken up only when objects are detected to reduce power dissipation. In conclusion, neuromorphic computing inspired by the brain promises extreme energy efficiency for certain classes of learning tasks, such as classification and pattern recognition. This thesis explores the design and realization of such neuromorphic circuits, especially in-memory computing and time-based computing architectures, for energy efficient processing. We propose in memory computing approach for image preprocessing, such as image denoising and image filling. Next, both near-memory computing and in-memory computing based region proposal approaches are explored. Finally, we investigate time based computing techniques and propose a novel energy- and area-efficient neuron oscillator for robust neurocomputing applications. The following section describes the objectives and contributions of this thesis.

1.4 Thesis Objectives and Contributions

A great success in AI is largely attributed to the rapid development of ML algorithms and NN models. In real applications, however, a fundamental limitation of the capability of ML is the real-time performance of the hardware. The bottleneck of the energy efficiency of a computing system limits the capability of a powerful ML, in particular for battery-powered portable and edge devices. Hence, this thesis primarily focuses on exploring different energy-efficient algorithms and their circuit implementations. In-memory computing (IMC), edge event-driven computing (EEDC), and time-based computing (TBC) are presented in this thesis to reduce the energy consumption for neuromorphic applications. Next, a brief outline and contribution of each chapter will be provided.

Chapter 2 reviews the prior works previously reported in literature in the field of dynamic event-driven neuromorphic vision sensor (NVS), in-memory computing (IMC) and time-based computing (TBC) separately.

Chapter 3 presents an analog in-memory computing technique based on collocated SRAM and DRAM (CRAM) architecture for image/video preprocessing for traffic surveillance, including image denoising and image filling. In this chapter, a $9T$ bit-cell compact CRAM structure together with a 2-dimensional charge diffusion paths topology is used to store the input binary image data and compute locally. During in-memory computing mode, the CRAM cell is modeled as a RC cell. The 2-dimensional RC computing network performs both denoising for background noise and region filling for the fragmented objects. Compared to conventional in-memory computing implementations, the proposed prototype achieves better energy efficiency and throughput without any computational burden and bandwidth overhead, due to the natural in-memory analog computations. The computing is very fast and completes image denoising and filling in several clock cycles. The prototype structure exhibits error-free read and write operations with V_{DD} down to 400 mV and frequency up to 1 GHz. The energy efficiency is 233 TOPS/W at the normal operating frequency of 200 MHz and normal supply voltage of 1.2 V. The natural process of charge diffusion makes the process of IMC consuming $\sim 10000\times$ less energy on a chip area $5.8\times$ smaller than conventional digital implementations realized using the same process node.

Chapter 4 presents an energy efficient edge event-based region proposal (EEDRP) algorithm and hardware implementation that reduces data transfer and performs less computational operations by responding only to edge events (both rising and falling edges). The EEDRP algorithm can be realized using near-memory computing by reading the image memory and processing it locally. The proposed EEDRP approach achieves $\sim 2.9\times$ faster processing speed and occupies $\sim 5.6\times$ less computing memory resources than the conventional connected component labeling (CCL) implementation. Though it is both energy- and time-efficient compared to the CCLRP approach, it still takes significant processing time due to the intrinsic limitation of its serial operation mode. Hence, we went on to investigate the axes projection based region proposal (APBRP), which is an in-memory computing based parallel operation leveraging the CRAM structure. In a typical application, the whole processing only requires two or three projections, further reducing the energy dissipation and execution time. As a result, it is $\sim 1767\times$ faster than the CCLRP implementation for calculating the region of interests of 15 objects, thanks to the parallel computation approach. Both EEDRP and APBRP approaches can deal with not only the restored image but also the raw binary image directly with appropriate configuration. The in-memory computing based APBRP is $\sim 2700\times$ more energy efficient than the near-memory based EEDRP. The weighted $F1$ scores of both EEDRP and APBRP achieve $2.55\times$ and $1.7\times$ better than the conventional HISTRP and CCLRP, respectively.

Chapter 5 demonstrates a novel delay cell that can be used to implement many time-domain neural network (TDNN) functions. The proposed dynamic latch-based delay cell achieves higher energy efficiency by reducing the number of current paths. This chapter also presents a low-area and low-power consumption CMOS differential current controlled oscillator (CCO) for neuromorphic applications. The oscillation frequency is improved over the conventional one by reducing the number of MOS transistors, thus lowering the load capacitor in each stage. The analysis shows that for the same power consumption, the oscillation frequency can be increased about 11% compared with the conventional one without degrading the phase noise. Alternatively, the power consumption can be reduced 15% at the same frequency. The prototype structures are fabricated in a standard 65 nm CMOS technology and measurements demonstrate that the proposed CCO operates from 0.7-1.2 V supply with maximum frequencies of 80 MHz and average energy/cycle of 0.63 pJ over the tuning range. Further, system level simulations show that the nonlinearity

in current-frequency conversion by the CCO does not affect its use as a neuron in a Deep Neural Network if accounted for during training.

Finally, Chapter 6 draws conclusions obtained from the research performed in this work and shows some directions for future work that this thesis can lead to.

Chapter 2

Literature Review

In this chapter, existing related works of event-based image processing circuits and architectures will be reviewed. We will first describe the event-based neuromorphic vision sensor (NVS), a kind of bio-inspired cameras used to capture the vitality of a scene, significantly mitigating data redundancy and latency. Next, after a brief review of in-memory computation (IMC) structures for neural-network based energy-efficient machine learning applications, region proposal (RP) algorithms for computer vision applications are discussed. In the last section, different types of delay cells (DCs) and CMOS ring oscillators (ROs) that can be used for neurocomputing are presented.

2.1 Neuromorphic Vision Sensors

Bio-inspired neuromorphic sensing originates from the development of a "silicon retina" by Misha Mahowald [38]. Mimicking the human retina, the silicon retina reduces the bandwidth by subtracting average intensity levels from the image and reporting only spatial and temporal changes. Neuromorphic vision sensors (NVSs) or event cameras are asynchronous sensors that pose a paradigm shift in the way visual information is acquired. This is because they sample light based on the scene dynamics, rather than on a clock that has no relation to the viewed scene. Their advantages are: very high temporal resolution and low latency (both in the order of microseconds), very high dynamic range, and low power consumption. This

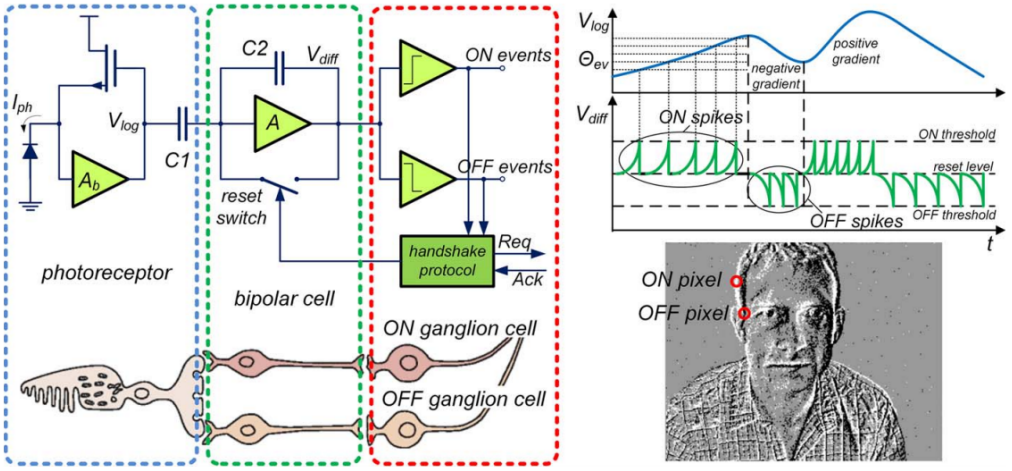


Figure 2.1: Three-layer model of a human retina and corresponding DVS pixel circuitry. Adopted from [47].

inspiration drives the concept behind the Dynamic Vision Sensor (DVS) and has led to the creation of a lot of startups in recent years.

Fig. 2.1 illustrates the three-layer circuity model of a human retina and the corresponding DVS pixel circuitry (left). It is composed of a logarithmic photodetector (photoreceptor), a differential amplifier (bipolar cells), and two detection units (ON/OFF ganglion cells), implementing the photoreceptor-bipolar-ganglion cell information flow [47]. For event-based cameras, "event" or "spike" means the pixel bright changes. If the bright change overcomes a threshold, an ON (going brighter) or OFF (going darker) event is generated [20]. The event-based sensor can simplify data by detecting the change pixels only, as shown in the right side of Fig. 2.1, only ON and OFF pixels are recorded. So this sensor can achieve very high dynamic range, lower power consumption and reduce data at the source.

The waveforms at the top right corner of Fig. 2.1 illustrate the operation principle of the pixel circuit. The upper trace represents a voltage waveform at the output node of first layer V_{log}, recording the photocurrent through the photoreceptor. The bipolar cell layer responds with spike events V_{diff} of different polarity to positive and negative changes of the photocurrent, while being monitored by the ganglion cell circuit that also transports the spikes to the next processing stage. The amount of log-intensity change is encoded in the number of events, the rate of change in interevent intervals. V_{diff} exceeding one threshold value (Θ_{eV}) results in an ON or OFF event's emission. Finally, the vision sensor outputs events following the address event representation (AER) protocol [48]. The output event e_i can be

expressed as

$$e_i = (x_i, y_i, t_i, p_i) \tag{2.1}$$

where (x_i, y_i) represents the event's pixel coordinate location on the sensor array, the 1 bit polarity $p_i = 1$ or -1 represents the ON (going brighter) or OFF (going darker) event, respectively, and t_i denotes the event's timestamp of the event generally at a microsecond resolution. Once read, the pixel is reset for the normal operation.

Event generation model is presented in [38]. Event based sensor has independent pixels that respond to changes in their brightness, i.e. log photocurrent $L(x_i, y_i, t_i) = log(I(x_i, y_i, t_i))$. An event $e_i = (x_i, y_i, t_i, p_i)$ is triggered at pixel $(x_i, y_i)^\intercal$ and at time t_i as soon as the brightness difference since the last event at the pixel reaches a temporal contrast threshold $\pm C$ (with $C > 0$), i.e.

$$\Delta L(x_i, y_i, t_i) = L(x_i, y_i, t_i) - L(x_i, y_i, t_i - \Delta t_i) = p_i C \tag{2.2}$$

where Δt_i is the time elapsed since the last event at the same pixel.

Contrary to traditional frame based sensor, the output of DVS is a sequence of asynchronous events. A new data processing method is required to deal with these event data. Although it increases latency, a simple and naive solution is to accumulate events occurred over a certain time interval Δt and adapt standard computer vision (CV) algorithms.

Surprisingly, many applications can be solved by only processing events (i.e., the brightness changes); however, to apply traditional computer vision algorithms, some form of static output (i.e., the "absolute" brightness) is also required [47]. A use case for this application is in tracking where the frame is used to extract key-points which are then tracked using the events in the interval between frames. To address this shortcoming, there have been several developments of event sensors that concurrently output dynamic and static information. The Asynchronous Time Based Image Sensor (ATIS) has pixels that contain a DVS subpixel that triggers another subpixel to read out the absolute intensity. However, the ATIS has the disadvantage that pixels have at least double the area of DVS pixels. Also, in dark scenes the time between the two intensity events can be long and the readout of intensity can be interrupted by new events.

Table 2.1: Specifications summary of event based sensors

Spec.	DVS	ATIS	DAVIS
Function	Asynchronous temporal contrast event detection	Asynchronous temporal contrast event detection + Intensity measurement for each event	Asynchronous temporal contrast event detection + Synchronous image capture
Fixed pattern noise	2.1%	0.25% intensity	3.5% (DVS) 0.5% (APS)
Array size	128 × 128	304 × 240	240 × 180
Pixel size (μm^2)	40 × 40	30 × 30	18.5 × 18.5
Latency (µs @ 1 klux)	15	4	3
Power consumption (mW)	24	175 (high activity) 50 (low activity)	14 (high activity) 5 (low activity)
Dynamic range (dB)	120	125	130 (DVS) 120 (APS)

The widely-used Dynamic and Active Pixel Vision Sensor (DAVIS) [38, 49] combines a conventional active pixel sensor (APS) in the same pixel with DVS. The advantage over ATIS is a much smaller pixel size since the photodiode is shared and the readout circuit only adds about 5 % to the DVS pixel area. Intensity (APS) frames can be triggered on demand, by analysis of DVS events, although this possibility is seldom exploited. However, the APS readout has limited dynamic range and, like a standard camera, it is redundant if the pixels do not change.

The specifications of different neuromorphic sensors mentioned above are summarized in Table 2.1.

2.2 In-memory Computing

Most computing systems nowadays are designed based on the von Neumann architecture where the stored data have to traffic to the processing unit. Huge amounts of data need to be shuttled back and forth between the processing unit and memory unit during the execution of each computational task. Hence, the process of computation incurs significant costs in both latency and energy. This is usually referred to as "memory wall" or "von Neumann bottleneck", challenging the big data centered machine learning application. As schematically illustrated in Fig. 2.2(a), a von Neumann architecture is implemented by exploiting in tandem the physical attributes of the memory devices, array-level organization, the peripheral circuitry as well as the digital control logic [50].

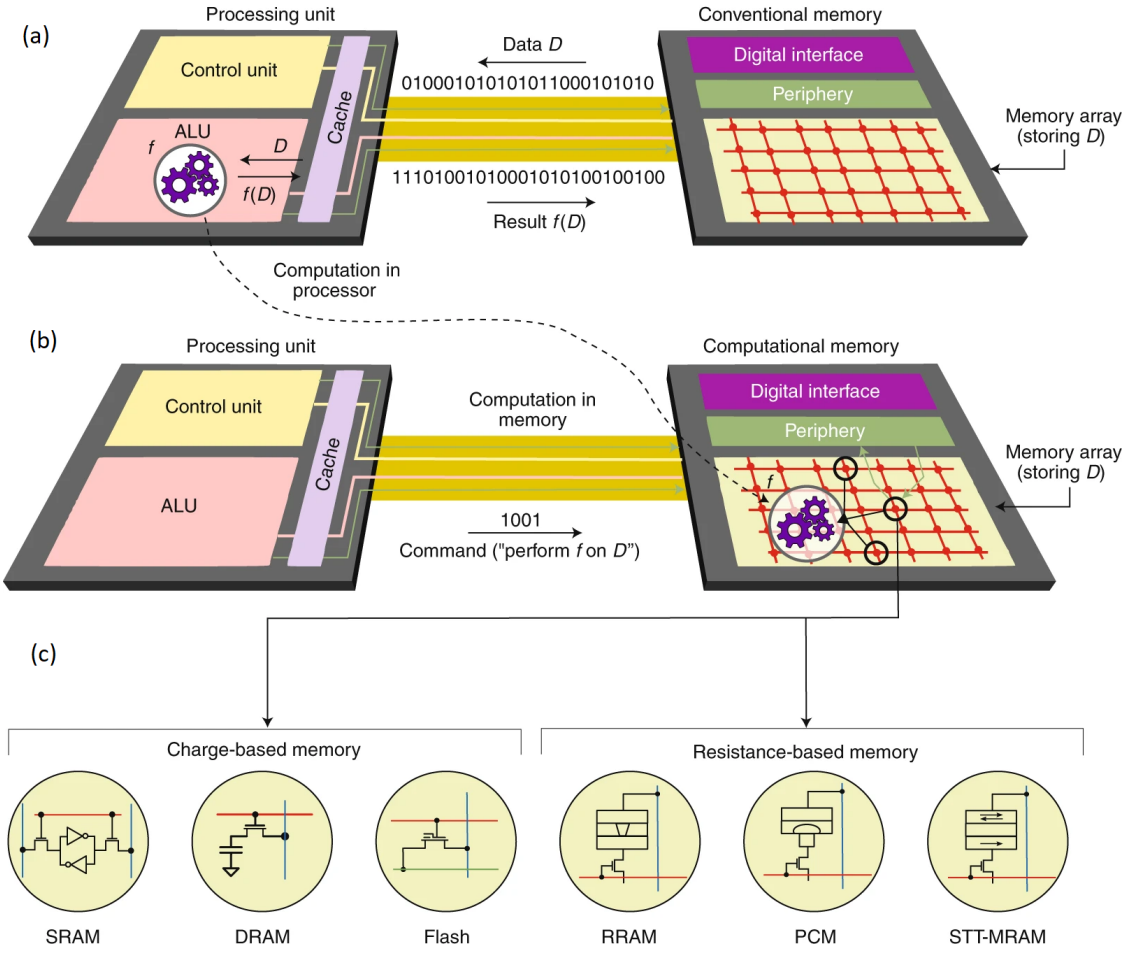


Figure 2.2: Von Neumann architecture and Non-von Neumann architecture. (a) In a conventional von Neumann computing system, when an operation f is performed on data D, D has to be moved into a processing unit, leading to significant costs in latency and energy. (b) In the case of in-memory computing, the result $f(D)$ is obtained within a computational memory unit by exploiting the physical attributes of the memory devices, thus obviating the need to move D to the processing unit. The computational tasks are performed within the confines of the memory array and its peripheral circuitry, albeit without deciphering the content of the individual memory elements. (c) Both charge-based memory technologies, such as SRAM, DRAM and flash memory, and resistance-based memory technologies, such as RRAM, PCM and STT-MRAM, can serve as elements of such a computational memory unit. Adopted from [51].

With the development of deep learning, Von Neumann architectures take unbearable energy and latency costs because of the separation of data storage device and computing unit. The increasing demands on memory storage capacity and computational capability make it challenging to deal with huge data on resource-limited platforms such as portable products and sensory devices. To overcome the bottleneck of the system performance, near-memory computing and in-memory

computing [52] were proposed to reduce or even remove the distance between memory device and processor unit.

Inspired by the highly energy-efficient human brain where memory and processing are deeply intertwined, we can break the barrier between processing unit and memory unit to improve in computational efficiency significantly. One prominent idea to overcome this barrier is to physically place monolithic computing units closer to a monolithic memory or even put the computation into the memory cell and mix them together. That is the concept called near-memory computing or in-memory computing. As shown in Fig. 2.2(b), the arithmetic logic unit (ALU) is integrated in the memory cell, eliminating the data transfer and reducing the associated cost of energy and time. However, this unavoidably poses some new challenges to the memory design. Fig. 2.2(c) depicts various devices used to implement in-memory computing. Conventional charge-based memory devices, such as static random access memory (SRAM), dynamic random access memory (DRAM) and flash memory, are widely used because of the compatibility with standard CMOS technology. Recently, new non-volatile memory (NVM) technologies appeared, such as resistive-RAM (RRAM), phase change memory (PCM) and spin-transfer torque RAM (STT-RAM), but which are not always fully CMOS compatible. This thesis will focus more on the standard charge-based memory, in particular the popular SRAM and DRAM for in-memory computing.

As one of the emerging non-von Neumann architectures, in-memory computing (IMC) structures are various and the vague concept needs to be defined first. A taxonomy for classifying IMCs was defined previously [50]. According to this taxonomy, the IMC approaches can be divided into four different categories, as represented in Fig. 2.3.

- Computation-near-Memory (CnM, Fig. 2.3-A): In most modern architectures, memory hierarchy usually consists of multiple levels of cache, the main memory, and storage. The traditional approach is to move data up to caches from the storage and then process it. In contrast, computation near-memory (CnM) aims at processing close to where the data resides. This data-centric approach places the computational units close to the data and seeks to minimize the expensive data movements [53].

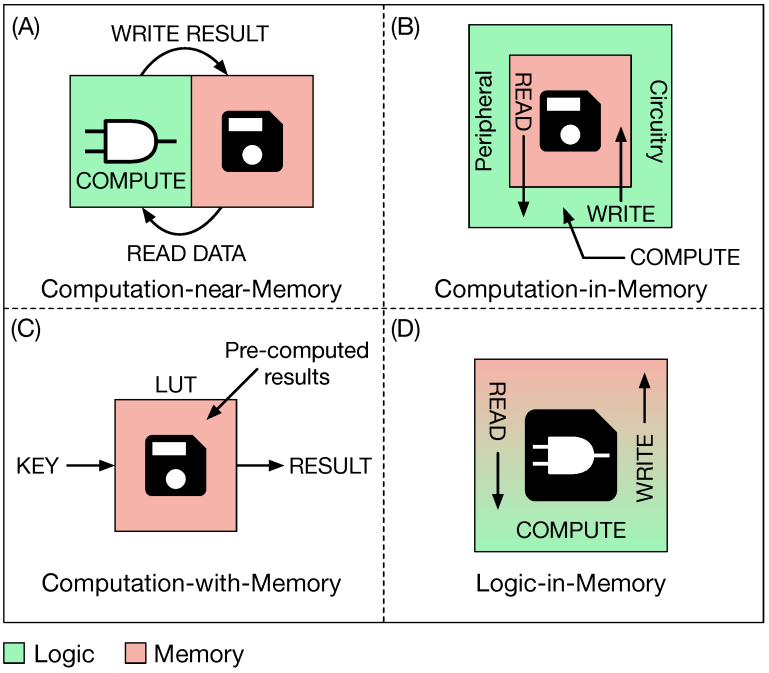


Figure 2.3: Depending on how the memory is used for computing data, four main in-memory computing (IMC) approaches can be defined. (A) Computation-near-Memory (CnM); (B) Computation-in-Memory (CiM); (C) Computation-with-Memory (CwM), and (D) Logic-in-Memory (LiM) [50].

- Computation-in-Memory (CiM, Fig. 2.3-B): To reduce the delay and energy associated with on-chip memory accesses, recent works have proposed memory-based CiM scheme, which performs computation on the bitline (BL) without reading out the each row of bitcells, demonstrating significant improvement in energy efficiency and throughput [54].

- Computation-with-Memory (CwM, Fig. 2.3-C): To provide the advantage of re-configuring hardware resources, CwM approach uses memory as a content addressable memory (CAM) to retrieve pre-computed results by means of a look-up table (LUT). The working principle of CwM is that Boolean functions can be encoded and stored in truth tables [50].

- Logic-in-Memory (LiM, Fig. 2.3-D): Unlike the other computation approaches, LiM computation is directly integrated inside the memory cell, where data are computed locally without the need to move them outside the array. Internal readings are performed in order to execute operations on data stored in different cells, by exploiting inter-cells connections. Internal writings are executed locally to save the result of the operation [50].

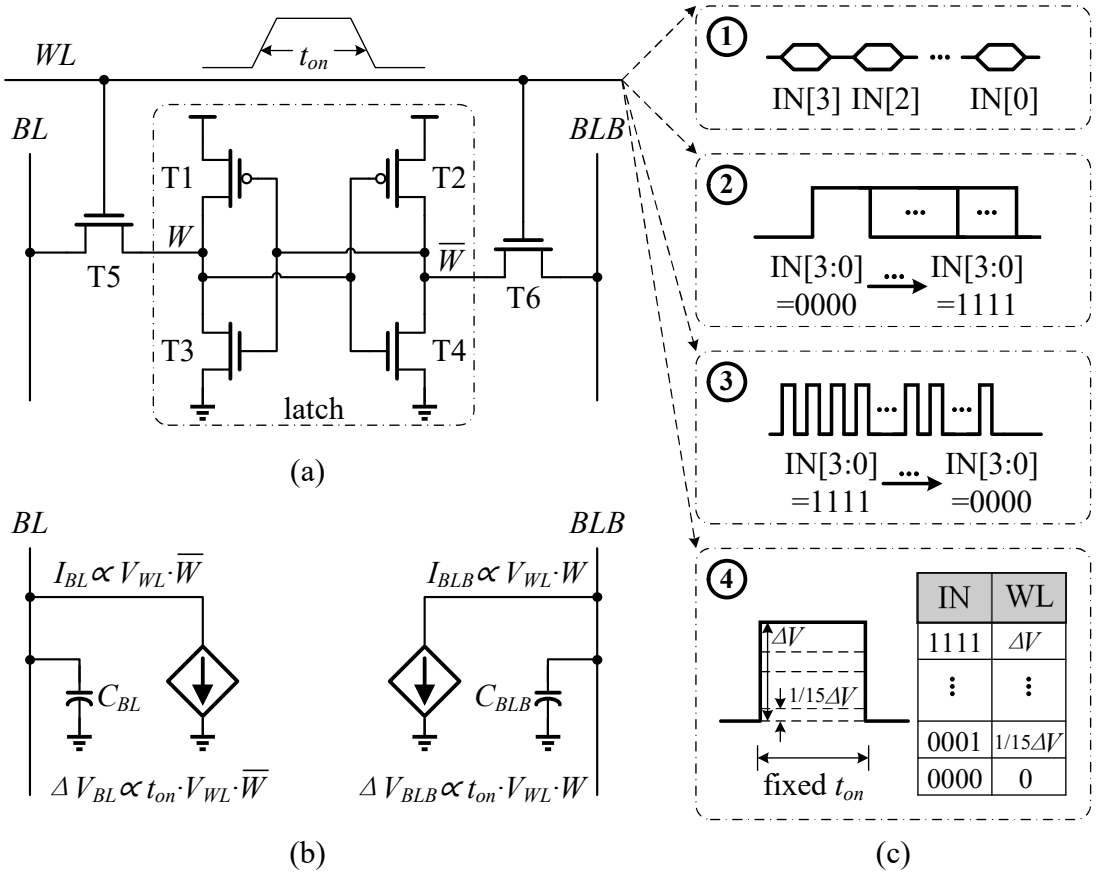


Figure 2.4: (a) Schematic of a basic 6T-SRAM bit-cell. (b) Equivalent circuit model for multiply and accumulate (MAC) operation. (c) WL modulation methods for multi-bit inputs IMC operation.

There has been considerable progress in in-memory computing, exploiting the basic and various improved architectures of an SRAM cell [7], [24], [55]. A basic SRAM cell is a bi-stable transistor structure typically made of two CMOS inverters connected back to back, as illustrated in Fig. 2.4(a). The bit value W can be stored in the inverter-based latch, which is either "0" or "1". The positive feedback loop can retain its stored information as long as power is supplied. In order to write an SRAM cell, the bitline (BL) and bitline bar (BLB) are connected to the data and its complement, respectively. Subsequently, the wordline (WL) is activated, so BL and BLB can then drive to flip (if needed) the value stored previously in the SRAM bit-cell. On the other hand, to read a value from an SRAM cell, BL and BLB are pre-charged to the supply voltage V_{DD} first, and then deactivate the pre-charge signal. After the activation of WL, both BL and BLB are discharged at different rates that depend on the data stored in the bit cell. By sensing the differential voltage between V_{BL} and V_{BLB} with a sense amplifier (SA), the stored

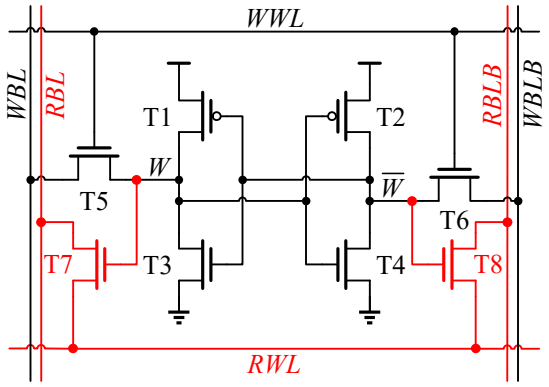


Figure 2.5: 8T-SRAM decoupled read and write path for MAC operation [24].

data can be readout.

Besides the normal read and write operations, SRAM is also capable of performing multiply and accumulate (MAC) operations. Typically, the bit lines BL and BLB are initially pre-charged to V_{DD} during IMC mode. Fig. 2.4(b) depicts an equivalent circuit model of the 6T-SRAM during multiplication operation, where V_{WL} or t_{on} can represent one of the operands (input feature), and another operand (weight $+1/-1$) is stored in the SRAM bit cell. The digital input features are translated by a digital to analog converter (DAC) into an analog representation, usually either a voltage level (V_{WL}) [56] or a pulse width (t_{on}) [57], that will then drives the wordline WL. The bit-cell currents I_{BL} and I_{BLB} are proportional to the gate-source voltage of the access NMOS devices $T5$ and $T6$. Within a column, the bit-cell currents add together to discharge the inherent parasitic capacitors of BL and BLB (i.e. C_{BL} and C_{BLB} shown in Fig. 2.4(b)), respectively. To increase the output precision, some works explored increasing the precision of inputs and weights. Clustering multi-bit cells together as one weight can realize multi-bit weighted computing and increase the precision. Multi-bit input features can be performed by modulating WL. Fig. 2.4(c) shows some common modulation methods for multi-bit (4-bit as illustrated) inputs IMC operation. During the IMC period, the access transistors ($T5$ and $T6$ in Fig. 2.4(a)) of SRAM cells within a column are enabled together, there may appear the possibility of unintentional bit flips (miswriting operation) if one of the lines (BL or BLB) discharges below the write trigger voltage of an SRAM cell.

To address this concern, an 8T-SRAM array with separated read and write paths by two additional nMOS devices (T7 and T8) was proposed for MAC operations [24],

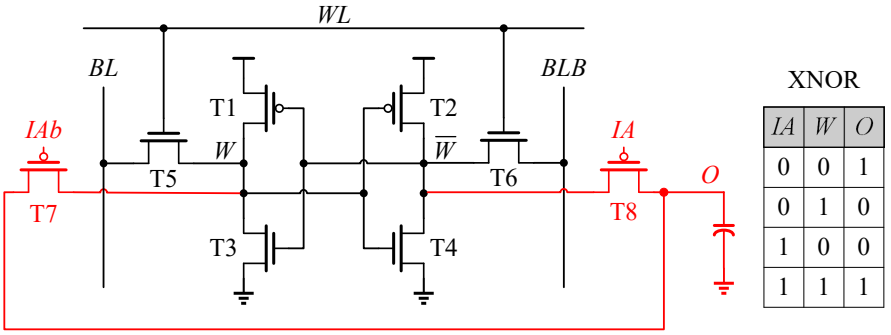


Figure 2.6: 8T1C-SRAM capable of of XNOR operation [55].

as shown in Fig. 2.5. Likewise, a 10T-SRAM array with decoupled read and write paths is presented in [7]. The proposed architecture leverages charge distribution for MAC operations. On a similar track, [55] exploits 8T1C SRAM bit-cell for charge domain computation, as shown in Fig. 2.6. Compared to a standard 6T bit cell, the 8T1C cell has two additional pMOS devices (T7 and T8) and an extra metal oxide metal (MOM) finger capacitor. It is worth mentioning that the above approaches only consider 1-bit weight in their implementations.

A summary of recent works on near/in-memory computing is captured in Table 2.2.

2.3 Region Proposal Algorithm

Raw video frames captured by a static visual sensor has redundant background pixels across frames since the background does not change much. Moreover, noise is fundamentally present in any image sensor. This leads to the necessity of region proposal (RP) algorithms and hardware to detect the region of interests (ROIs). To exploit spatial redundancy in the valid frames, the neural network triggers only when a valid region is detected and only ROIs instead of the whole image are fed into the subsequent stages to reduce the computation. [69]. RP algorithms aim to find out a bounding box (bbox) encapsulating objects in an image along with their location coordinates. The location and the shape of an object can be characterized by a 4-tuple in the form of (x, y, w, h), where (x, y) could be the spatial coordinate of the top-left corner (start point) or the center point (in this thesis, we will always consider it as the former), and w and h the the width and height of the bbox, respectively. We can also use the bottom-right corner (stop point) coordinates to replace w and h.

Table 2.2: Summary of recent works on near/in-memory computing

Work	SRAM Type	SRAM Capacity (kB)	Operation Mode	Input/Output Precision (bit)	Weight Precision (bit)	Energy Efficiency (TOPS/W)
Zhang '17 [56]	6T	2	Current summation	5/1	1	-
Si '19 [58]	Twin-8T	0.47	Current summation	$1-4/2-5$	$3-7$	$18.37-72.03$
Si '19 [59]	6T (split WL)	2	Current summation	1/1	1	$30.49-55.8$
Ando '17 [60]	6T	102	Digital XNOR	1/1	1	6.3
Kim '19 [61]	6T (split-WL)	2	Current summation	1/1	1	300
Biswas '18 [62]	10T	2	Charge sharing	6/6	1	51.3
Jiang '20 [63]	8T1C	2	Capacitive coupling	1/5	1	671.5
Valavi '19 [55]	8T1C	307.2	Charge sharing	1/1	1	866
Yin '20 [64]	12T	2	Voltage divider	1/3.46	1	403
Gonugondla '18 [57]	6T	16	Current summation & Charge sharing	8/4	8	3.125
Yu '20 [24]	8T	2	Current summation	$1/1-5$	1	490
Yang '19 [65]	8T	20	Pulse width modulation	8/8	1	119.7
Su '20 [66]	6T	8	Current summation & Digital accumulation	$2-8/10-20$	$4-8$	$7-61.1$
Dong '20 [67]	8T	0.5	Charge sharing	4/4	4	610.5
Si '20 [68]	6T	8	Current summation & Digital accumulation	$4-8/12-20$	$4-8$	$11.54-68.44$

For a binary image, the region proposal can be developed using a variant of the connecting component leveling (CCL) algorithm [70], [71], histogram based RP algorithm [44], and costly R-CNN [69], [72].

2.3.1 Connected Component Labeling (CCL)

Connected-component labeling (CCL) is a common and wide adopted algorithm. It assigns a unique label to all pixels of each connected component (i.e., each object) in a binary image [73]. The CCL algorithm scans an image and groups its pixels into components based on pixel connectivity, i.e. all pixels in a connected component share similar pixel intensity values and are in some way connected with each other. CCL is an important way to distinguish different objects, and a prerequisite for image analysis and object recognition in the image.

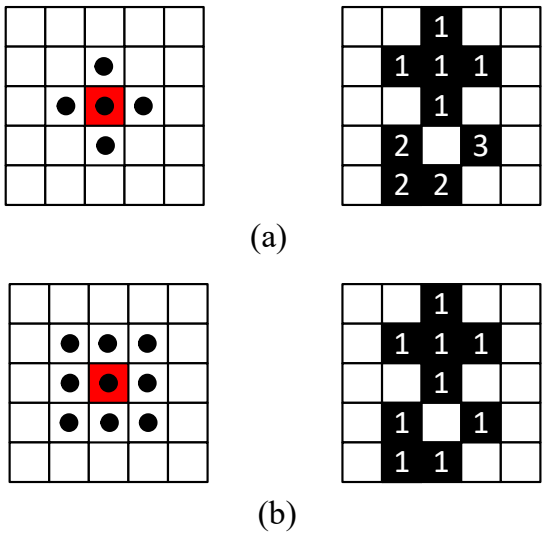


Figure 2.7: The definition of neighborhood of pixel $b(x, y)$ and the connected components example. (a) 4 connected neighborhood, (b) 8 connected neighborhood.

For pixel $b(x, y)$, the four pixels $b(x-1, y)$, $b(x, y-1)$, $b(x+1, y)$, and $b(x, y+1)$ are called the 4-neighbors of the pixel; the four-neighbors together with the four pixels $b(x-1, y-1)$, $b(x+1, y-1)$, $b(x-1, y+1)$, and $b(x+1, y+1)$ are called the 8-neighbors of the pixel. Fig. 2.7 illustrates the definition of 4 connected and 8 connected neighborhood separately and gives the corresponding examples of CCL results for a simple image. With different connected neighborhood used in the CCL algorithm, the object labeling results may vary. For the same image, Fig. 2.7(a) proposes three objects using 4 connected neighborhood while Fig. 2.7(b) proposes one object using 8 connected neighborhood. Obviously, CCL RP algorithm with 8 connected neighborhood is more accurate but more complicated than that with 4 connected neighborhood, because for each pixel, the northeast, north, northwest, and west pixel are checked when considering 8-connectivity while only the north and west pixel are checked for 4-connectivity.

There are different approaches to label the connected components. The simplest approach repeatedly scans the image to determine appropriate labeling until no further changes can be made to the assigned labels [74]. A label assigned to an object pixel is called a provisional label before the final assignment. If there is no object pixel in the scan mask, the current pixel receives a new provisional label. On the other hand, if there are object pixels in the scan mask, the provisional labels of the neighbors are considered equivalent, a representative label is selected

to represent all equivalent labels, and the current object pixel is assigned this representative label. One simple strategy for selecting a representative is to use the smallest label. There are many algorithms to label objects. Next, we will simply review some common CCL approaches.

A. Two-pass CCL

The two-pass CCL algorithms are divided into three steps and perform two image scans. The first scan (or first labeling) assigns a temporary label to each connected components and some label equivalences are built if needed. The second step solves the equivalence table by computing the transitive closure of the graphs associated to the label equivalences. The third step performs a second scan (or second labeling) that replaces temporary labels of each connected component (CC) by its final label, by doing a simple look up: $I(i, j) \leftarrow L[I(i, j)]$.

B. Seed-filling CCL

Seed filling is a fast and very simple approach to implement and understand. It is based on graph traversal methods in graph theory. In short, once the first pixel of a connected component is found, all the connected pixels of that connected component are labelled before going onto the next pixel in the image. So this algorithm is also called "one component at a time". It is efficient in time, but costly in space. The traversal can be done using a stack in depth-first order or a queue, in breadth-first order. The most efficient algorithm of this type is the Contour Tracing (CT) algorithm presented by [75].

However, CCL algorithms find region candidates by scanning the image row by row with raster scan fashion. Hence, longer searching time and higher computing energy are still the bottleneck due to the von Neumann architecture of the conventional processor, which involves enormous data movement between the storage and computing unit.

2.3.2 Region Proposal Network (RPN)

A Region-base Proposal Network (RPN) [76] takes an image as input and outputs a set of rectangular object proposals, each with an objectness score. The process is modeled by a fully convolutional network. RPN is a key block in a image processing

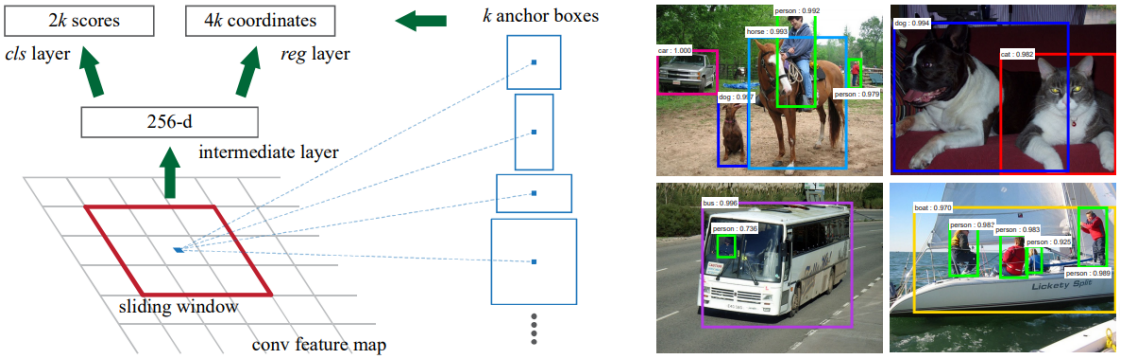


Figure 2.8: Left: Region Proposal Network (RPN). Right: Example detections using RPN proposals on PASCAL VOC 2007 test. Adopted from [72]

pipeline, like recognizing, detecting and tracking. The main idea is composed of two steps. First, a selective search is used to identify a manageable number of bounding-box object region candidates, i.e. region of interests (ROIs). Second, it then extracts CNN features from each region independently for classification. To make R-CNN faster, Fast R-CNN improved the training procedure by unifying three independent models into one jointly trained framework and increasing shared computation results [72].

Recent advances in object detection are driven by the success of region proposal methods and region-based convolutional neural networks (R-CNNs). But region-based CNNs were computationally expensive and usually operated on GPUs. Fig. 2.8 illustrates the RPN and example detections using RPN proposal [72]. To generate region proposals, a small network is slided over the conv feature map output by the last shared conv layer. This network is fully connected to an $n \times n$ spatial window of the input conv feature map. Each sliding window is mapped to a lower-dimensional vector. This vector is fed into two sibling fully-connected layers—a box-regression layer (reg) and a box-classification layer (cls).

These algorithms are very efficient for natural images but computation-intensive and not suitable for very low density images (very few pixels set to one) like EBBI generated from NVS.

2.3.3 Histogram-based RP (HIST RP)

In the application of traffic monitoring using a stationary camera, neuromorphic vision sensor (NVS) offers the advantage of almost perfect foreground background

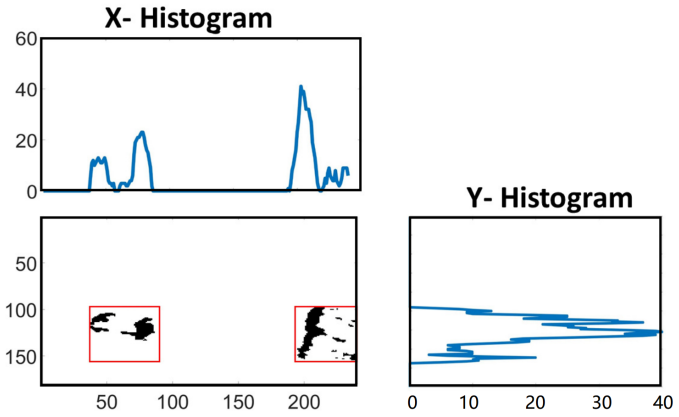


Figure 2.9: A sample EBBI with corresponding X and Y histogram based region proposals. Adopted from [44]

separation inherently since the pixels only respond to changes in contrast. Thus, background pixels will generate little or no events while moving objects will generate a significantly larger number of events. This allows us to locate valid regions without having to perform costly CNN operations.

To reduce the computation and memory cost, a histogram-based region proposal (HIST RP) was proposed [44], which only requires a side view of the ongoing traffic. X and Y histograms (H_X and H_Y) are created for the image and find regions in these two 1-D signals by finding consecutive entries that are higher than a threshold, as shown in Fig. 2.9. Thus, The actual 2-D region is obtained by finding intersections of the X and Y regions. To further reduce the computation and memory requirement, the histograms can be created from a scaled image [39].

Nevertheless, the histogram-based approach also suffers from false bounding boxes due to multiple regions on both horizontal and vertical directions. Besides, it does not provide exact coordinates of the boxes due to the overlap among the projections of the objects.

2.4 Time Based Circuits

Analogue signals are typically processed using physical quantities such as voltage, current, charge, frequency or time duration. Among these, voltage mode and current mode circuits are the most common examples, where the input and output signals are represented by a certain amount of voltage or current, like the basic operation

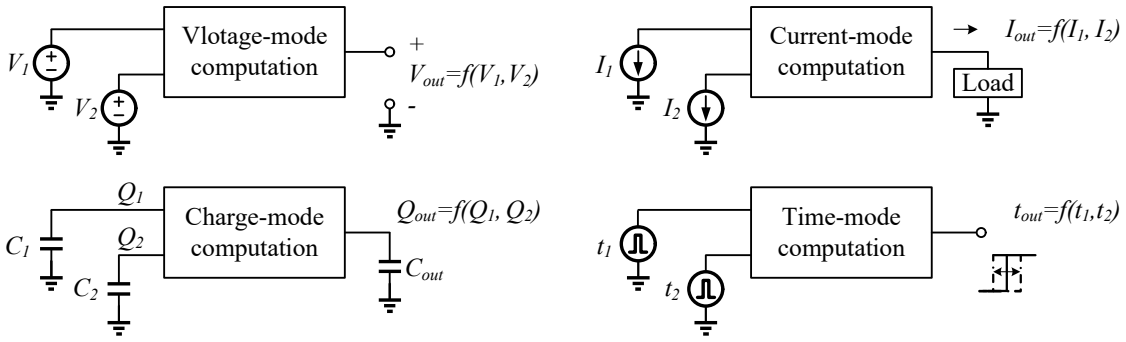


Figure 2.10: Different modes of computation used for signal processing.

amplifier, band-gap reference, and current mirror. Charge mode designs are used mostly in dynamic circuits, such as oscillator, switched-capacitor (SC), and charge coupled devices (CCDs). Fig. 2.10 depicts block diagrams for different modes of computation for signal processing. Signals are represented by temporal event or delay time in time-mode circuits. In reality, voltage, current, charge or time delay are there in every circuit and sometimes semantics and philosophy are debated when definitively categorizing these classes of circuits [77]. Voltage is associated with current by Ohm's law $V = IR$ and with charge by the definition of capacitance $C = Q/V$. On the other hand, all these are time dependent with the format of $Q(t) = \int I(t)dt$ and $I(t) = C\frac{dV(t)}{dt}$.

2.4.1 Variable Digital Delay Cell

Variable delay line elements can be realized by digital- or analog-controlled structures. Digitally controlled delay line elements are simple and easy to be cascaded to reach any desired number of delay elements. The number of elements in a chain determines the amount of the delay. Each delay element usually provides a fixed and quantized time delay. These kinds of delay line elements are suitable for coarse-grain delay variation in a wide range of regulation. Analog controlled delay line elements usually could be tuned by analog voltage or current, which are suitable for fine-grain delay variation. They are efficient in applications where small, accurate, and precise amount of delay is necessary to achieve.

There are three popular techniques for designing a variable delay element. These are known as: shunt capacitor technique [78], current-starved technique [79], and variable resistor technique [80].

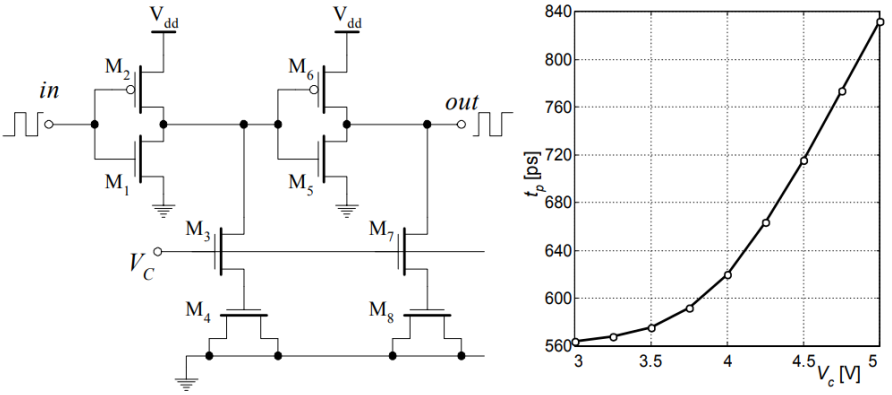


Figure 2.11: Shunt capacitor delay element scheme (left) and typical characteristic delay in term of control voltage (right) [78].

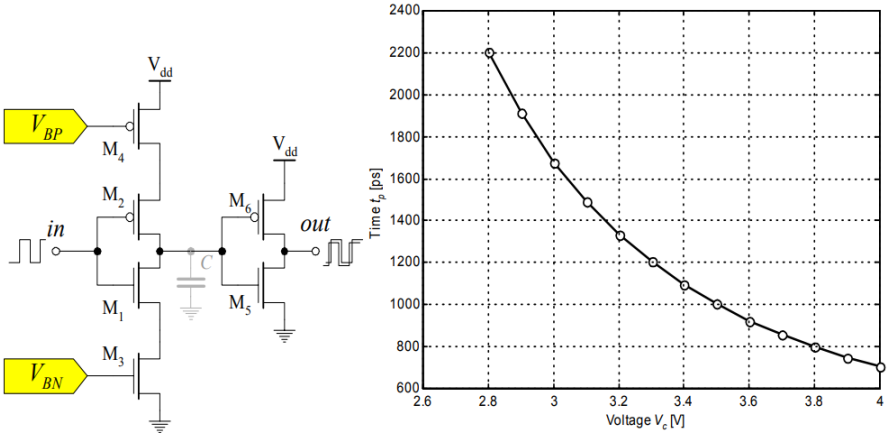


Figure 2.12: Current starved delay element scheme (left) and typical characteristic delay in term of control voltage (right) [79].

Fig. 2.11 shows the basic circuit of a two-stage delay cell using a shunt capacitor. The transistor M3 and M7 act as linear resistors controlled by gate voltage V_C and define the charging/discharging current of load MOS capacitors M4 and M8. The gate voltage of M3 and M7 modulates the delay of output signal indirectly. The drawbacks of this type of delay element are: a) the output MOS capacitor occupies a large silicon area; b) the MOS transistor is sensitive to process, voltage and temperature (PVT) variations; c) the amount of a delay and the active range of voltage regulation are small [78].

Fig. 2.12 illustrates the basic schematic of a current starved delay element. There are two stages (delay stage and buffer stage) between the input and output. The first stage is a current-starved delay element implemented by a CMOS inverter (M1 and M2) and current starving devices (transistors M4 and M3). The charging/discharging current of the output parasitic capacitor C is controlled by the bias

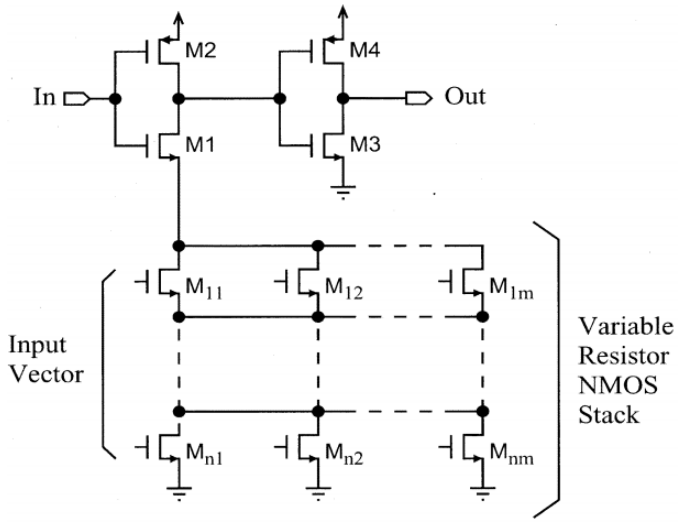


Figure 2.13: Delay element using variable resistor [80].

voltages (V_{BP} and V_{BN}), which can regulate the propagation delay of this element. In this delay element, the current starved transistors control both the rising and falling edges of the output signal. In some cases, only the control of rising (falling) edge is required, then only M3 (M4) need to be applied. The second stage inverter (composed of M5 and M6) is called buffer stage using shape inverter for improving the rise and fall edge of the circuit and increasing drive capacity of the output signal. Sometimes, multiple cascaded inverters are used for this purpose.

The last type of delay element is shown in Fig. 2.13, where a variable resistor is used to control the delay. The variable resistor is made by a stack of n rows by m columns of NMOS transistors array. This resistor controls the discharge current of the NMOS transistor M1 and subsequently controls the delay time. In this case, using NMOS transistors only the pull-down current is controlled by the input vector, hence only the rising edge of the output can be modulated. Similarly, to modulate the falling edge of the output signal, PMOS stack variable resistor is needed at the source of the PMOS transistor M2.

2.4.2 CMOS Ring Oscillator

The oscillator in a signal processing circuit should be tunable for a wide frequency range. However, high oscillation frequency consumes more energy because the power consumption is proportional to the oscillation frequency (dynamic power $P = CV^2f$). Oscillators are highly complex dynamical systems, and it is an

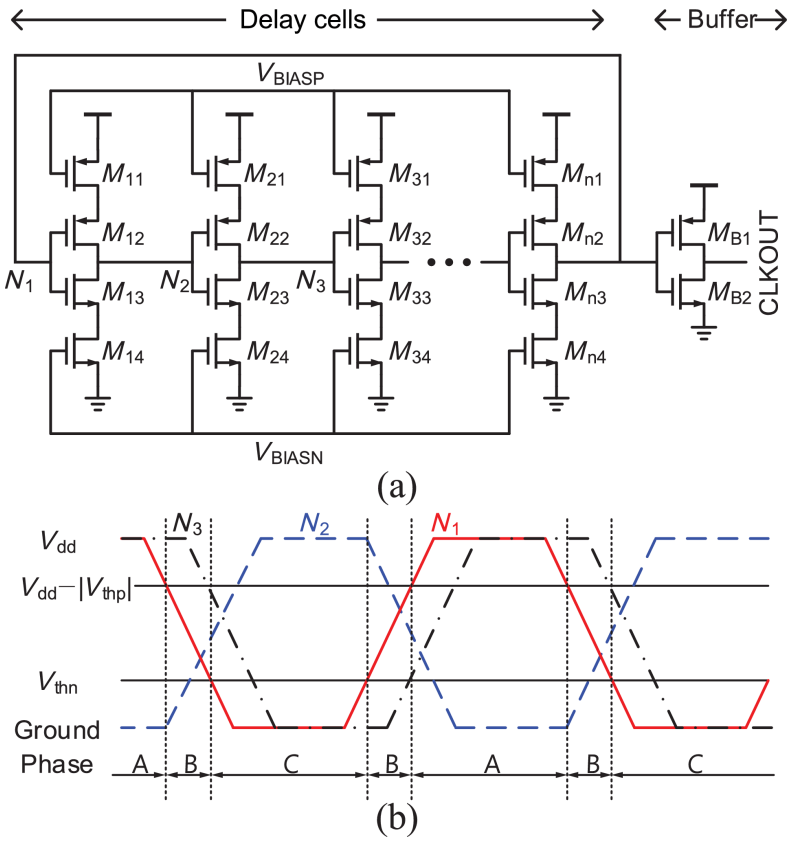


Figure 2.14: Current-starved ring oscillator (CSRO). (a) Circuit diagram. (b) Internal node waveforms [82]

intriguing concept to use oscillator dynamics for computation. The time domain computing is promising for lower supply voltage in advanced technology nodes. To a large extent, these efforts are motivated by biological observations – neural systems and mammalian brains, which seem to operate on oscillatory signals [81]. The CMOS ring oscillator is a good candidate due to its wide tuning range, small silicon area, and compact design.

Fig. 2.14(a) shows a conventional single-ended current starved ring oscillator (CSRO). The delay stage consists of a delay generator (M_{X2} and $M_{X3}, X = 1, ..., n$) and a current-starving circuit (M_{X1} and $M_{X4}, X = 1, ..., n$). The charge/discharge current depends on the source-to-drain resistance of the current-starving circuit, which is controlled by its gate voltage (V_{BIASP} and V_{BIASN}). The gate voltage control thus enables the oscillator to tune the output clock frequency. This is called voltage controlled oscillator (VCO). An internal signal (N_1) is connected to a logic inverter (M_{B1} and M_{B2}) to buffer the output. The simplified waveforms of internal node N_1, N_2 and N_3 in steady state are illustrated in Fig. 2.14(b). In Phases A

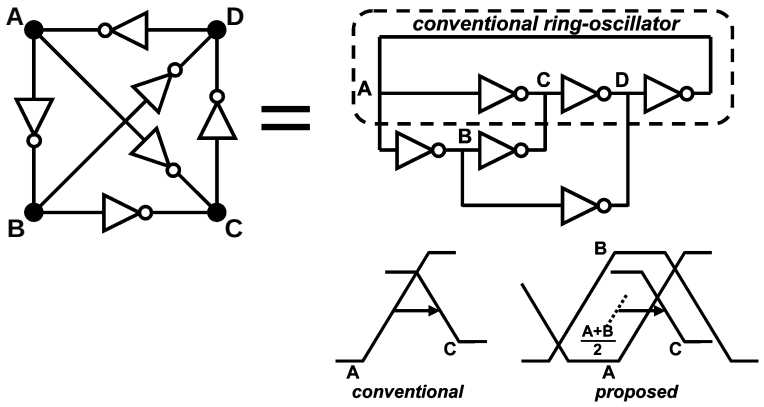


Figure 2.15: Oscillation principle of a hyper-ring oscillator. [83]

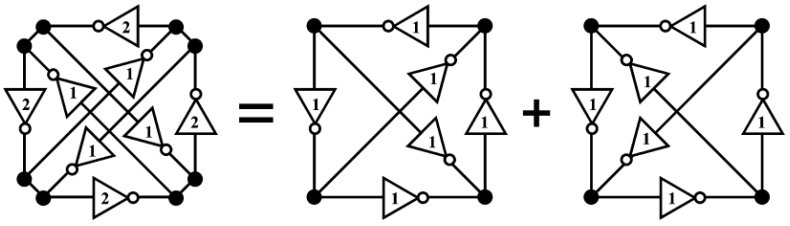


Figure 2.16: Decomposition of a tetrahedral oscillator into 2 identical hyper ring oscillators. [83]

and C, the first delay cell consumes current only for charging or discharging its load capacitance since either M_{12} or M_{13} is turned off and there is no short-circuit current. In Phase B, however, V_{N_1} lies between V_{thn} and $V_{dd} - |V_{thp}|$ and turns on both transistors, causing a short-circuit current. The waveform of N_1 is in opposition of phase with N_2 but with a time shift of Δt and has the same phase with N_3 but with a time shift of $2\Delta t$.

A hyper ring oscillator is shown in Fig. 2.15. It gives the brief explanation on how the hyper ring oscillator would have the faster frequency than the conventional one, since it has two interpolating nodes C and D. For note C, there exist two inputs, i.e. nodes A and B. For the conventional 3-stage ring oscillator as shown in the dash box, the rising of A leads to the falling of C. For the hyper-ring oscillator, however, both the rising of A and the rising of B lead to the falling of C. Therefore, the effective delay from A to C could be faster than the one inverter delay which would make the whole loop turnaround time to be shorter than normal 3 inverter gate delays. It is noted that the node labels in the original figure of [83] is incorrect.

Fig. 2.16 illustrates an interesting structure, which can be seen as a four-stage single-ended hyper ring oscillator or a two-stage differential-ended ring oscillator. It

is composed of four adjacent loops each of which is a 3-stage ring oscillator sharing one inverter between neighboring loops. This structure can act as an inherent quadrature phase generating oscillator since it has four nodes where all those nodes have exactly same environments, leading to 90 degree shift of each phase. It is composed of two identical hyper ring oscillators, as shown in Fig. 2.15. That explains how the oscillation frequency of 4-stages hyper ring oscillator could be faster than that of conventional 3-stage one.

Chapter 3

CRAM: In-memory Computing based Binary Image Denoising and Filling for Neuromorphic Vision Sensor Applications

3.1 Introduction

Recently, there has been a tremendous growth in the number of connected physical objects under the paradigm of the Internet of Things (IoT) spurred by the advent of 5G communication. Among such sensors, video cameras hold a special role due to their rich information content that is expected to reach up to 95% of total internet traffic by 2030 [84]. However, due to the huge volume of such data, it requires a special paradigm, internet of video things (IoVT), to process and handle the scalability of such networks [85]. Neuromorphic vision sensors (NVS) [86] hold promise for such applications due to their ability to reduce data at the source by optimal sampling [47]. By generating an interrupt at regular time intervals t_F=66 ms to collect the events accumulated since the last interrupt, [44] has shown a hybrid frame-event approach that processes event based binary images (EBBI) created out of events from a NVS allows for efficient denoising and region proposal (RP) operations. However, no hardware implementation was reported.

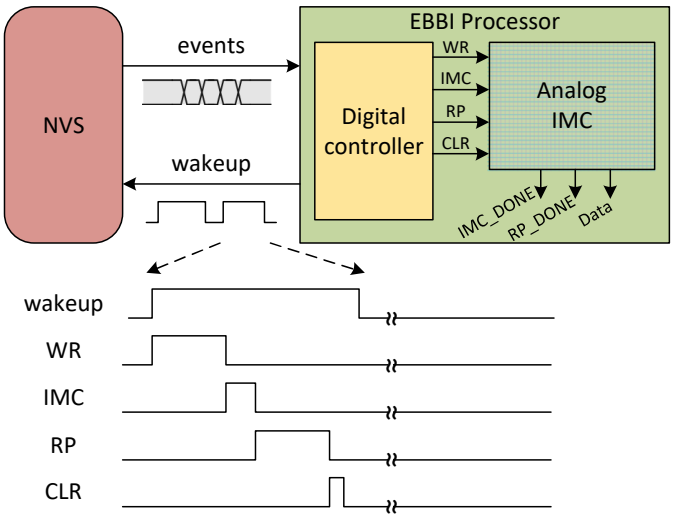


Figure 3.1: Overview of the system and timing diagram showing event driven operation of the NVS and the EBBI vision processor for low power application.

In this chapter, we propose an analog in-memory computing approach to achieve parallel computing for image denoising and filling locally within the memory array that is used to store the EBBI. A brief overview of the proposed system is shown in Fig. 3.1. The NVS feeds events signals to the EBBI vision processor when it receives a wake-up signal from it – this duty-cycled operation saves energy and is common in IoT nodes. When the wake-up signal is active, the image data will be written to the memory with an address event representation (AER) protocol. The NVS used in our work is DAVIS with the resolution of 240×180 setup at a traffic intersection. The system is very energy efficient leveraged by both the wakeup mechanism and the NVS inherent characteristic of only active pixels are recorded and passed to the EBBI processor. It takes only several clock cycles to perform in memory computing (IMC) for image denoising and filling. Subsequently, region proposal (RP) is done by scanning the memory once. Before the end of each frame, the memory is reset by a clear signal. The focus of our approach is to make the whole system less computationally complex leading to savings in energy/area and achieve a great acceleration. We focus on IMC approach the for denoising and filling only in this chapter and discuss RP algorithm and implementation in the next chapter.

Before we start the IMC, let's first take a look at the raw event based binary images (EBBI) created from NVS and explain how and where our inspiration of image denoising and filling came from. Fig. 3.2 shows a raw image where a bus is captured in the center of the frame. As we mentioned before, the NVS is a both time- and

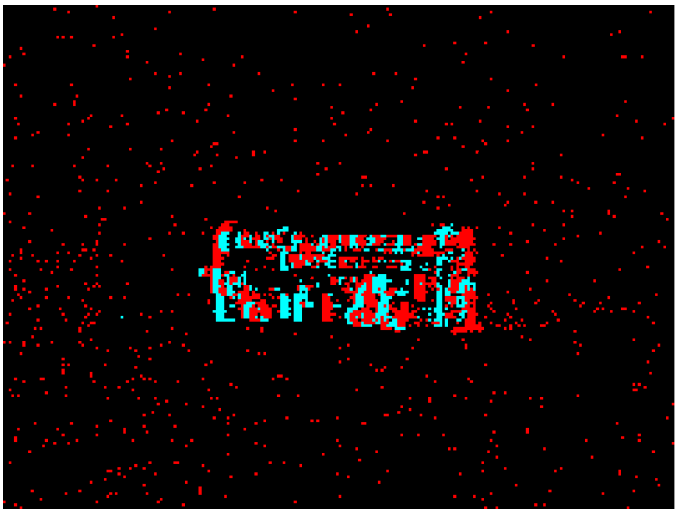


Figure 3.2: An example of event based binary images (EBBI) captured from the DAVIS240 camera where a bus is moving in the center of the frame.

data- efficient sensor in surveillance applications by capturing the changing pixels (contrast detection) only. The red pixels are the "ON" events while the blue pixels are the"OFF" events. As we can see, there are lots of unexpected noise pixels in the background and many unexpected missing pixels of the object (bus) itself. From a certain point of view, the noise and holes can be seen as a kind of pepper salt noise. The task in this chapter is to find an effective way to recover the raw image by filtering out the noise pixels and filling in the hole pixels.

Maybe you have seen the phenomenons of water ripple effect in a river and of ink diffusion on a paper. The inspiration of our IMC approach came from these basic physical phenomenons. The physical process of charge diffusion or charge redistribution is used in our designed CRAM array for image restoration. Considering that the "1" pixel stores a certain amount of electric charge while a "0" pixel stores zero electric charge, the electric charges across the 2D memory array could be manipulated to perform filtering and filling by charge redistribution. In a 2D cross-connected network, electric charges redistribution happens naturally due to the transport of charges occurring because of non-uniform amount of electric charges at each cell. This physical process leads to the homogenization, or uniform mixing, of the cell and its neighbours. Subsequently, the charge of a noise pixel will fade away and even die out because the majority of its neighbours are "0" pixels while the missing pixel will accumulate charges because the majority of its neighbours are "1" pixels. The whole process happens very quickly due to the globally parallel physical computing – electric charge diffusion. That is the basic

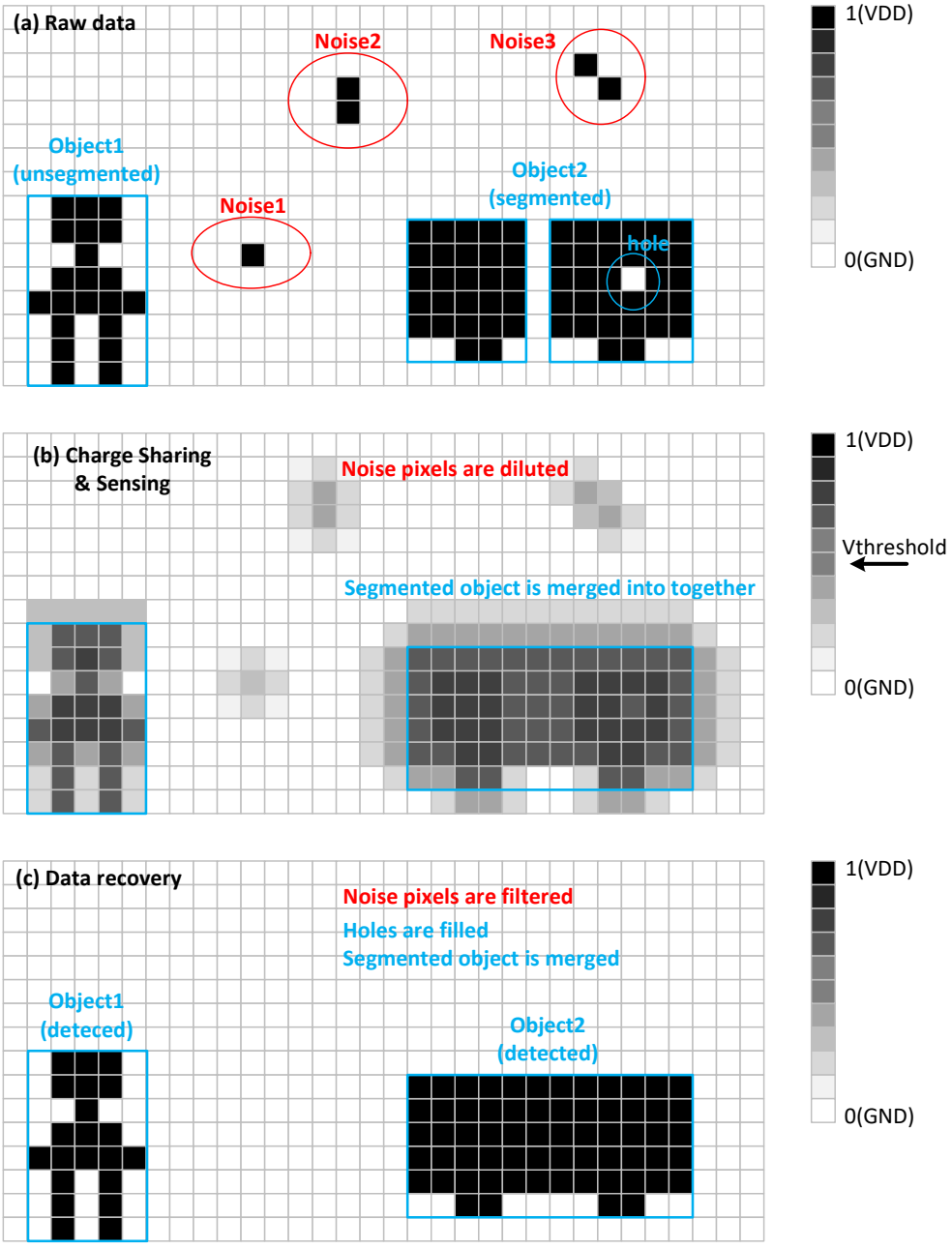


Figure 3.3: Demonstration of basic principle of the proposed IMC based image filtering and filling. The black dot represents a logic "1" pixel of an binary image. (a) Raw image data with some noise and two objects in which one is segmented. (b) Noise is diluted and holes are healed after charge sharing. (c) Objects are retained and healed while the noise is filtered and the holes are filling.

principle of our IMC approach. The rest challenges are how to provide the charge diffusion paths, how to control the extent of diffusion, and then how to sense to final results. A simple solution is MOS resistors can be used to connect each cell with its neighbours and form a 2D network, while a MOS capacitor can be used to store

the electric charges. The time that the charge is retained on the capacitor should be much longer than the charge diffusion duration. In our design, the retention time is 245 µs and the charge diffusion duration is around 50 ns.

A demonstration cartoon of the basic principle of the proposed IMC based image filtering and filling is illustrated in Fig. 3.3. The raw image data consists of one completed object, one segmented object, a hole of the object, and some background noise. After charge sharing and redistribution, the noise pixels are diluted while the holes are healed so that the segmented object is merged together. It is to be noted that the values stored in the cells are digital signal for the raw image data, i.e. either "0" or "1". But after charge diffusion, the values of each cell are analog, i.e. between "0" and "1". The right part of Fig. 3.3 shows the level of values from "0" (GND) to "1" (V_{DD}). When the charge diffusion is finished, the analog values of each cell are sensed and compared with an appropriate threshold voltage to recover the data. Another point worth highlighting is that the edge of an object will be dilated due to the charge diffusion, as shown in Fig. 3.3(b). But this could be recover by sensing and data recovery, as shown in Fig. 3.3(c).

3.2 Analog IMC Architecture Based on CRAM

3.2.1 CRAM Cell

In this section, we present a customized collocated SRAM and DRAM (CRAM) bit-cell for analog in memory computing. The schematic of the CRAM cell is shown in the left of Fig. 3.4, in which the storage component composed of an access transistor M_{access}, a storage transistor M_s serving as a capacitor and cross coupled inverters $inv1$ and $inv2$ selectively enabled by a PMOS switch M_{DR}. The access transistor M_{access}, and storage transistor M_s work as conventional $1T1C$ DRAM while the regenerative inverter chain $inv1$ and $inv2$ latches the storage data during write and read operation as a SRAM. Therefore, the proposed CRAM has the characters of both DRAM and SRAM. Except for the storage part, the vertical and horizontal transistors M_{DV} and M_{DH} are employed to perform the in memory computing (IMC) accross the whole array. The 2-dimensional (2D) transmission transistors (acting as resistors controlled by the programmable gate voltage) activate the storage charge diffusion simultaneously to enable the parallel

global computing. Generally, IMC is used in the memory in one mode, but in other mode it can also be used to store data like normal memories.

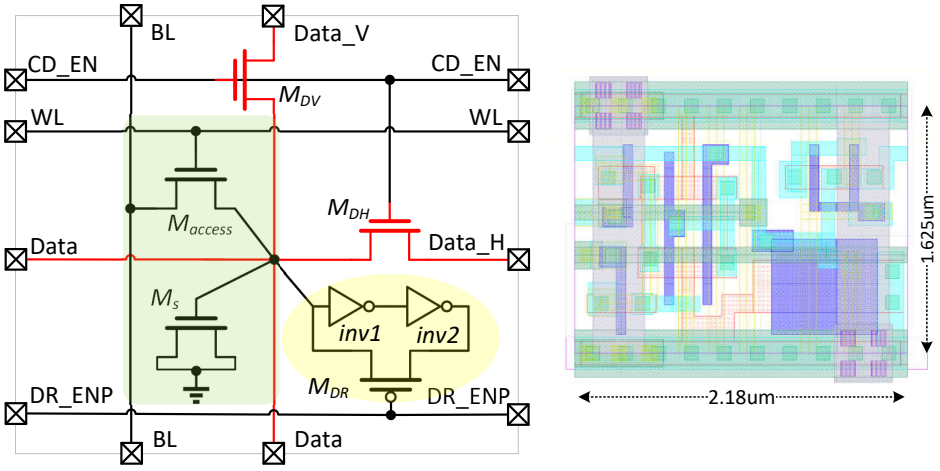


Figure 3.4: Schematic and layout of proposed CRAM cell. Line and block colours in the schematic are for clarity only.

The layout of the proposed $9T$ CRAM is shown in the right of Fig. 3.4 with size of 2.18 µm × 1.625 µm in 65 nm standard CMOS technology, including the two parallel power rails, as shown in the right of Fig. 3.4. The $9T$ CRAM cell occupies 1.21× larger area, compared with the $9T$ bit-cell area in [87] and 1.82× larger area, compared with the $9T$ bit-cell area in [88]. This is because of the additional MOS capacitor consuming 18% of the cell (0.64 µm^2) and the parallel integrated power rails in our work. However, as we show later, this overhead is less than the total logic area required to do the processing outside memory.

One big trade-off in area comes from the MOS capacitor – a large capacitor is better for matching between cells but at the cost of area overhead. Monte Carlo (CM) simulations with device mismatch is shown in Fig. 3.5. In our design, MOS capacitor size is selected as $W \times L = 0.25$ µm^2, resulting in a capacitor mean value of 2.9 fF and standard variation of 25 aF. The normalized variance of capacitor is $\Delta C/C = 0.86\%$. SPICE simulations show that is a good balance between performance and area.

3.2.1.1 Write mode

In write mode, the deactivation of the charge diffusion enable signal $CD\_EN = 0$ disables the charge diffusion while the data recovery enable signal $DR\_ENP = 0$

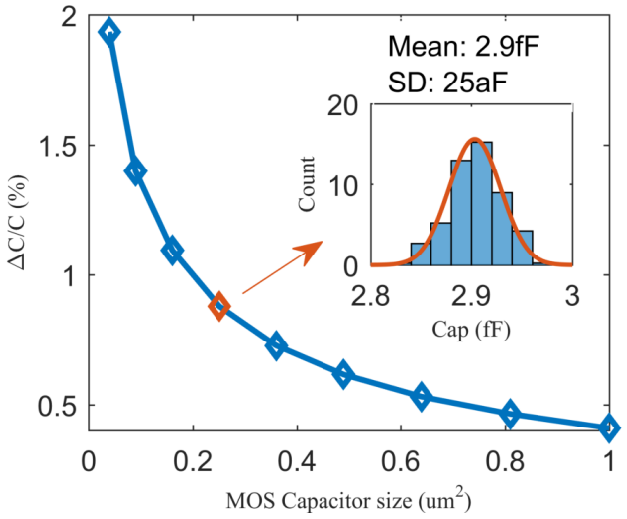


Figure 3.5: MOS capacitor mismatch for different capacitor sizes.

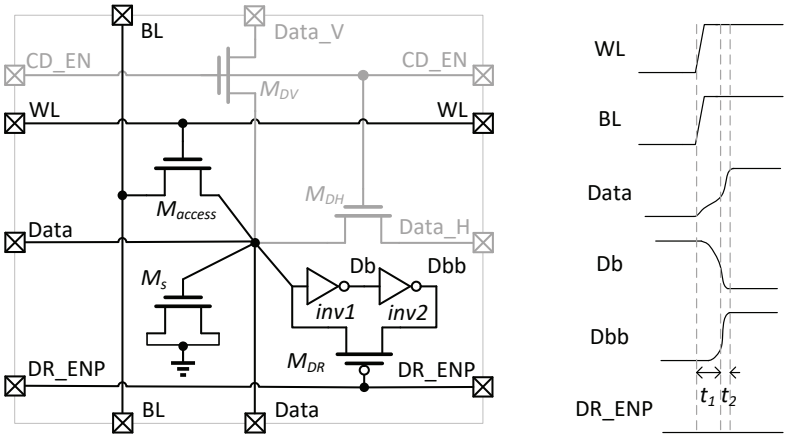


Figure 3.6: CRAM bit-cell operates in write mode and waveforms of writing bit "1" into the cell.

enables the latch. By activating word line WL and bit line BL, data will be stored and latched in the cell, as shown in Fig. 3.6. The charging process of the storage node is divided into two phases t_1 and t_2, where t_1 is the capacitor charging phase (the voltage of the capacitor V_{Data} is less than the threshold voltage of inverter $inv1$) taking a longer time, while t_2 is the latch phase and requires less time. Therefore the write speed of the proposed CRAM is faster than that of conventional DRAM but slightly slower than that of SRAM. The right of Fig. 3.6 depicts the voltage waveforms of each nodes when the case of "1" is wrote into the CRAM bit-cell.

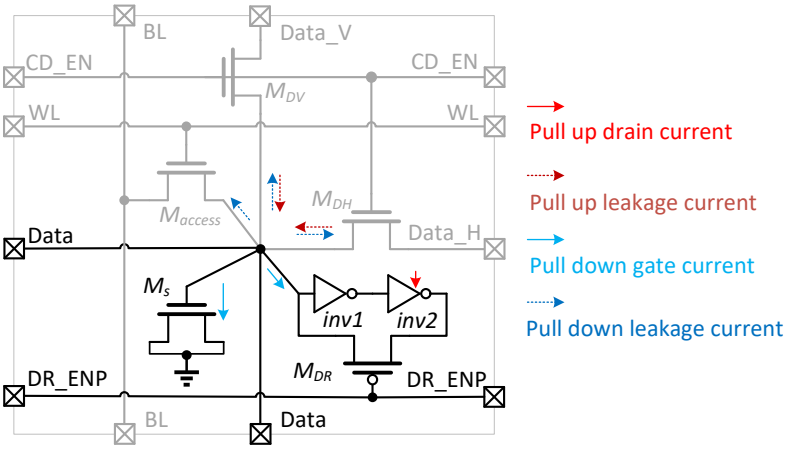


Figure 3.7: CRAM bit-cell operates in retention mode and the static currents flow.

3.2.1.2 Retention mode

Because the cross-coupled inverters latch when the switch transistor M_{DR} turns on, the SRAM-like cell latches and keeps the storage bit value, so the refresh circuit of the conventional DRAM design is not required here. The refresh-free CRAM makes the peripheral circuits simple. The static current through the storage node is illustrated in Fig. 3.7, where the directions of the horizontal and vertical sub-threshold leakage currents depend on the data stored in the corresponding cells.

3.2.1.3 IMC mode

After all the data are written into the CRAM array, the in-memory computing (IMC) will be activated if the configuration bit IMC_EN is enabled. In IMC mode, both word line WL and bit line BL are inactivated to disable access to the cell while the data recovery switch M_{DR} is turned OFF to release the stored charge, as shown in Fig. 3.8. The horizontal and vertical located transistors M_{DH} and M_{DV} operate as resistors whose resistance is determined by the gate voltage CD_EN programmable by a 4-bit DAC controlled by $diffusion\_r\_sel < 3:0 >$ – thus, the 2-dimensional charge diffusion RC network is formed. The RC model of a single bit-cell is illustrated in the right of Fig. 3.8. The diffusion time depends on the pulse width of CD_EN signal programmable by $diffusion\_t\_sel$. The detailed analysis will be discussed later.

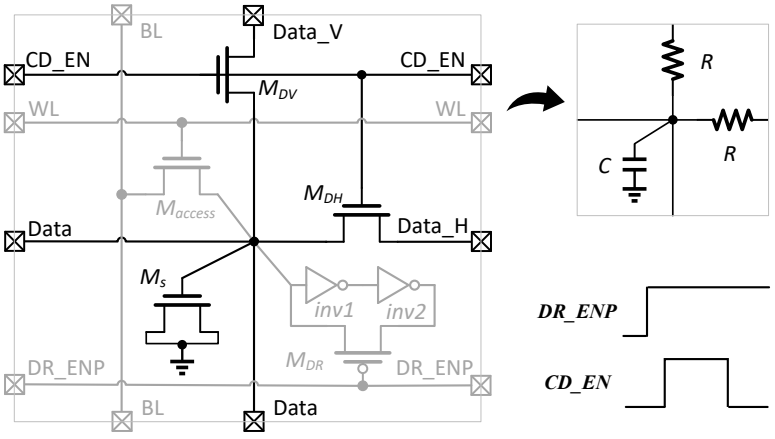


Figure 3.8: CRAM bit-cell operated in IMC mode and its equivalent RC model.

3.2.1.4 Sensing and data recovery mode

When IMC is done, all the data stored on the MOS capacitors are analog voltages between V_{DD} and GND. The inverter $inv1$ is carefully designed and the trip voltage is used as a reference to sense the redistributed charge information. The data recovery enable signal DR_ENP is set to GND to enable the inverters couple to a latch. The first inverter $inv1$ also acts as a comparator using its well-designed trip voltage. If the voltage of one particular cell is greater than the trip voltage of $inv1$, a bit "1" is detected and stored in the inverter latch, otherwise, a bit "0" is stored in the cell. The sensing and recovery of the redistributed data only takes one clock cycle due to the globally parallel computing, leading to high time efficiency.

3.2.1.5 Read mode

The read mode is identical with that of the conventional single-ended SRAM where data are read out by activating both word line WL and bit line BL. As shown in Fig. 3.9, the read operation is divided into three cycles, named precharge, access and sense. In our approach, the entire row data are read out to the latch first and then output one by one to reduce the WL power dissipation.

3.2.1.6 Clear mode

Since a NVS is used in our application, only a logic "1" is written into the memory randomly and hence memory clearing is necessary every frame cycle. In clear mode,

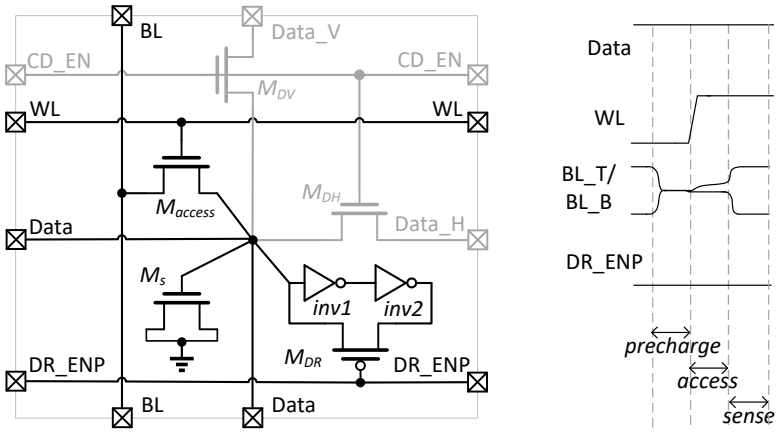


Figure 3.9: CRAM bit-cell operates in read mode and the waveforms of reading bit "1" from the cell.

all rows are selected and all bit lines are shorted to GND. Clear mode is actually to write "0" to all the bit-cells. It is to be pointed out that there is a balance to clear the memory array. Enabling a single WL in every clock cycle stretches the execution time (240 clock cycles in our case). In contrast, asserting all WLs high leads to a higher surge current at the BL, which may further trigger electromigration-induced reliability issues. Furthermore, the strength and area of BL drivers need to be very high to clear all 240 SRAM cells at once (worst case scenario). To address these concerns, we enable 12 WLs in every clock cycle involving $240/12 = 20$ clock cycles to erase the memory.

3.2.2 CRAM Array

A 320×240 CRAM array was designed for neuromorphic vision applications that would process quarter graphics display resolution (QVGA) or lower resolution images. The architecture of the CRAM array is designed by bi-level bit line construction as shown in Fig. 3.10, where the memory array is separated by sub-array0 and sub-array1. In each column, two bit line cross point sections are placed side by side where one is the true bit line while the other is designated as complementary bit line. The two parallel bit lines of top sub-array BL_T and bottom sub-array BL_B in each column improve common mode noise rejection, are sensed and compared by the consequential differential sense amplifier (SA). The architecture can be divided into separate analog and digital blocks.

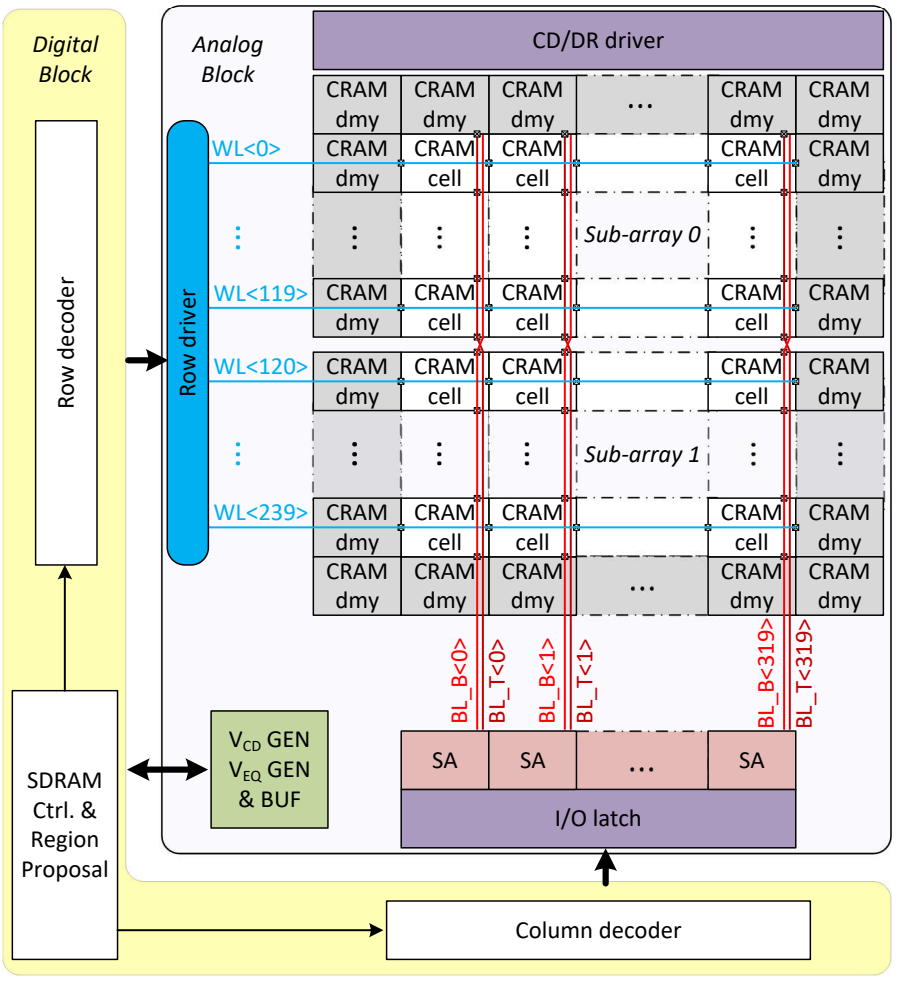


Figure 3.10: CRAM array architecture with two sub-arrays for bi-level bit line sensing.

The proposed IMC approach employs the basic physical principle of charge diffusion occurring because of the nonuniform distribution of the charge stored in the memory array. The motion of charge depends heavily on its surroundings, including the distribution of other charges in its nearest cells and the transport paths it can move through. To make each cell match well and offer the same transport surroundings for charges across the whole array, a dummy cell ring is added, as the grey part shown in Fig. 3.10. Although these dummy cells are utilized only to create matching surroundings for charge diffusion and are not used to compute, they can still be used as storage cells in normal memory mode.

The peripheral circuit of the CRAM array is as simple as SRAM controller, while higher word line voltage and refresh circuits in DRAM are not necessary. However, in order to implement the IMC function, a lower voltage pulse CD_EN is needed to

control the performance of computing by charge diffusion through the RC network.

3.2.3 Charge Diffusion based IMC

The proposed IMC approach can realize both image denoising and region filling by the physical phenomenon of charge dilution in lower density pixels and charge merging in higher density pixels. In other words, denoising refers to removal of a spurious '1' in a sea of '0' pixels while region filling refers to converting a spurious '0' in a sea of '1' pixels. Interestingly, other event-based filtering approaches, e.g. the nearest neighbour filter (NN-filt) [89], cannot perform the second operation.

Pixel denoising is performed by discharging, while pixel filling is achieved by charging – hence, denoising and filling are two opposite processes. A balance has to be achieved to optimize the performance of both. Both denoising and filling of a particular pixel depend on:

- **Charge diffusion time:** A longer diffusion time benefits denoising but degrades filling.
- **The resistance values of the resistor in RC network:** Because the storage capacitor is area-limited, the resistor equivalent MOS transistors work in linear region to provide a larger resistance.
- **The trip voltage of the inverter latch:** A higher trip voltage leads to a severe denoising effect while a lower trip voltage leads to over-filling.
- **The influence of the neighbourhood on charge distribution:** Discharging happens faster in areas of lower density of '1'pixels while filling happens faster in areas of higher density of '1' pixels.

A larger resistance of the equivalent resistor takes longer charging and discharging time to cross the trip voltage of the sensing inverter. In our design, the diffusion time is controlled by the configuration bit $diffusion\_t\_sel$, ranging from one clock cycle to two clock cycles. The resistance of the MOS resistor is programmable by the configuration bit $diffusion\_r\_sel < 3:0 >$, ranging from KΩ to MΩ order. The simulation results show that the overlapping range of voltages for charging and discharging is $0.1 - 0.5$ V at supply voltage of $V_{DD} = 1.2$ V within a clock cycle.

It indicates that a trip voltage lower than half V_{DD} should be designed when using in low-noise applications to get better filling performance, and vice versa.

Here, we are concerned about the transient state because the diffusion process will be terminated at a certain moment (controlled by the pulse width of CD_EN) before the steady state. Due to charge conservation, $\sum_{i=1}^{M\times N} C_i(t) \cdot v_i(t) = Constant$ for a $M \times N$ array, where $v_i(t)$ is the transient voltage of the i^{th} node of the array, C_i is the capacitor of the i^{th} node which is the total capacitor of the storage capacitor and the parasitic capacitor of the node. Ideally, C_i is equal for each cell. So, $\sum_{i=1}^{M\times N} v_i(t) = Constant$, the value of which depends on how many "1"s are stored in the memory array. For a certain cell i, the cell voltage $v_i(t)$ is sensed and compared by trip voltage V_{TR} of the inverter $inv1$ after the diffusion time t_{diff}. The compared result is then stored in the cell as the final bit.

To theoretically analyze the charge diffusion over 2-D RC network, we make the approximation that major diffusion currents of a pixel only flow to or from its four nearest neighbours and separately write equations for each case as follows:

3.2.3.1 Denoising

One high pixel in a sea of zeros is a noise and it should be filtered after diffusion. Hence, the voltage after the diffusion period should be lower than $V_{TR,min}$, where, $V_{TR,min}$ is the minimum value of the trip voltage V_{TR} due to devices mismatch. The voltage variation ΔV meets the condition of $\Delta V > V_{DD} - V_{TR,min}$. From the point of view of discharging, it can be represented as

$$t_{diff} > \frac{C(V_{DD} - V_{TR,min})}{4i_{discharge}} \tag{3.1}$$

where $i_{discharge}$ is the discharge current of the high pixel going to one direction.

3.2.3.2 Filling

To accomplish image filling, our approach looks for the nearest neighbours of the holes and borrow the information by charge sharing for the filling process. One zero pixel in sea of highs is a missing pixel by sensor and it should be filled in after diffusion. Hence, the voltage after the diffusion period should be greater than

$V_{TR,max}$, where $V_{TR,max}$ is the maximum value of trip voltage V_{TR} due to devices mismatch. The voltage variation ΔV meets the condition of $\Delta V > V_{TR,max}$. From the point of view of charging, it can be represented as

$$t_{diff} > \frac{CV_{TR,max}}{4i_{charge}} \tag{3.2}$$

where i_{charge} is the charging current of the zero pixel coming from one direction.

3.2.3.3 Dilation and Erosion

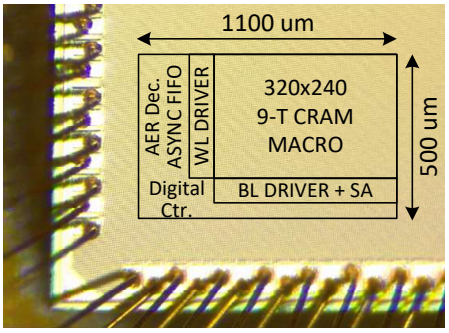
The boundary of a valid object should not be dilated or eroded by too much charge diffusion so as to distort the original object. Pixels at the edge diffuse in only one outside direction while the corners diffuse in two directions and suffer more erosion. Since the erosion of corner pixels are acceptable, we consider the edge pixels in the following analysis. In case of erosion of the edge pixels,

$$t_{diff} < \frac{C(V_{DD} - V_{TR,max})}{i_{discharge}} \tag{3.3}$$

In case of dilation of the edge pixels,

$$t_{diff} < \frac{CV_{TR,min}}{i_{charge}} \tag{3.4}$$

To achieve satisfactory performance of image restoration, all equations 3.1 - 3.4 should be fulfilled. $i_{discharge}$ and i_{charge} are relatively large at the beginning of diffusion and decrease afterwards because of the increasing resistance of the equivalent resistor caused by the decreasing of gate-source voltage v_{gs} and drain-source voltage v_{ds}. From the above analysis, it results that a longer diffusion time is beneficial for image restoration of both denoising and filling, but poses a risk to the valid pixels which csn be altered by both dilation and erosion. Practically, our choice of $1-2$ clock cycles prevents dilation or erosion, as was indicated in our simulations. Since charge diffusion happens naturally and does not need any power supply, there is minimal energy loss in our IMC approach as shown next.



Technology	65nm LP
Memory type	9T-CRAM
Mem. capacity	90.75Kb
Active area	0.55mm^2
Algorithm	Image restoration
Design methodology	Mixed-signal
Supply voltage	0.7 – 1.2 V
Avg. computing energy	22 fJ/bit @1.2V

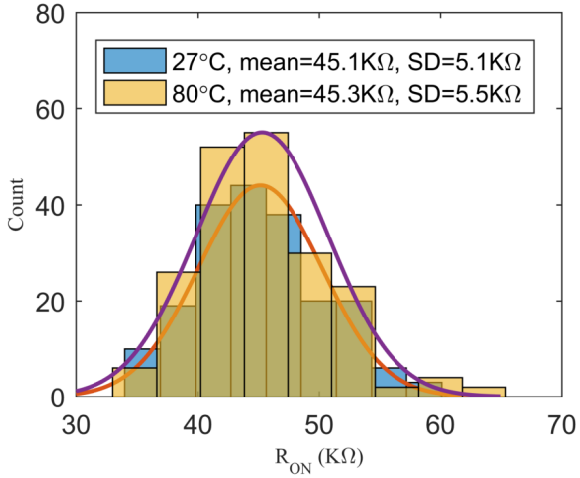
Figure 3.11: Die photograph and performance summary.

3.3 Results and Discussion

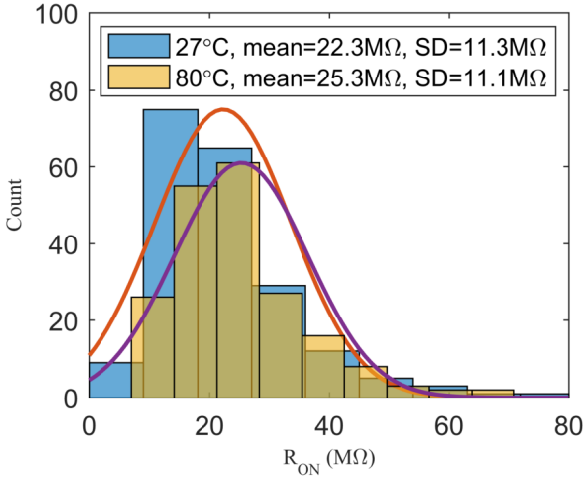
This section presents the simulation and test results from the proposed EBBI processor designed in the TSMC 65 nm CMOS process. Fig. 3.11 shows the die photograph of the proposed circuit and performance summary. The silicon area occupies only 110×500 µm^2 due to the compact layout.

Fig. 3.12 depicts the variation in resistance due to mismatch and temperature variations for the lowest and highest settings of the diffusion resistor. It can be seen that variation due to temperature is quite minimal. However, the mismatch induced variation at the high resistance setting is significantly larger than at the low resistance setting – this may lead to non-uniform filtering and will be studied in detail in future. We next demonstrate the denoise operation on an image from traffic video recording [44] in Fig. 3.13 for two different settings of the diffusion resistor. The traffic recordings were done using the DAVIS 240 neuromorphic vision sensor [86] at two different locations and captured vehicles of various sizes with velocities ranging from sub-pixel to 6 pixels/frame. It can be seen that the noise in the captured binary image is effectively removed by the diffusion and latch process in our proposed CRAM based denoise operation.

Due to the structure of collocated of SRAM and DRAM, the proposed CRAM can work in a wide supply voltage range. At sub-threshold voltage, the storage node is charged as a DRAM first and then latched as a SRAM. The simulated energy consumption per cell of writing "1" is 4.6 fJ, where 3.2 fJ is used to charge the DRAM capacitor at supply voltage of 1.2 V and operation frequency of 100 MHz. The inherent property of charge diffusion realizes globally fully parallel computing and reduces the execution time to about 10 clock cycles, the exact value depending



(a) Low resistor



(b) High resistor

Figure 3.12: Resistance variation of the MOS transistor used as equivalent diffusion resistor at different temperatures.

on the diffusion configurations. In contract, previously reported works of denoising have to scan the memory and compute intensively row by row [90] or multiple row [91].

Table 3.1 compares the performance of the proposed analog IMC based natural filter with a spatio-temporal [92] and a fully digital median filter that is synthesized in the same process for fair comparison. The former operated on the continuous events from the NVS, whereas the proposed approach and fully-digital implementation process event-based binary image following [44]. The IMC latency and energy are estimated at the configuration diffusion period equal to 2 clocks cycles. The average computing energy is 22 fJ/bit for noise and edge pixels that are discharged from

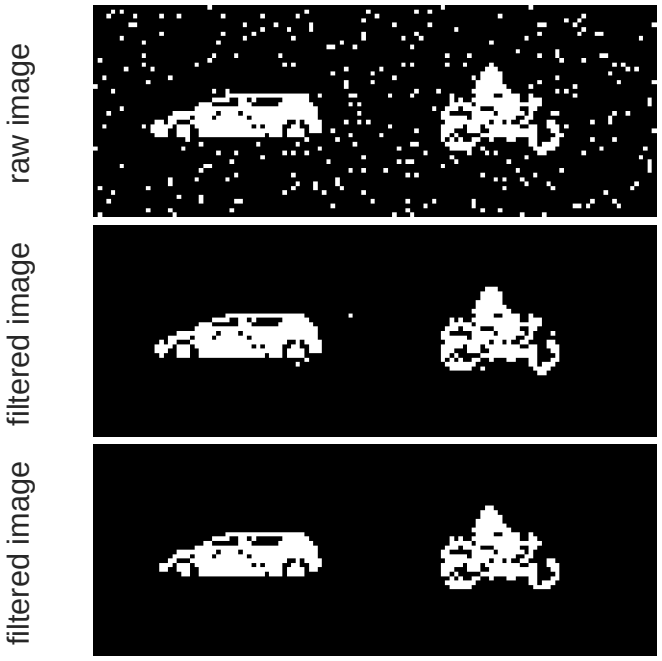


Figure 3.13: Performance of proposed IMC filter with different configurations of diffusion resistance (lower equivalent resistance in the bottom panel).

Table 3.1: Comparison with Different Filter Implementations

Type	Process	Area/Cell (μm^2)	Latency (ns/bit)	Energy (pJ/bit)
Spatio-temporal Filter [92]	180 nm	400	10	20
Median Filter	65 nm	4.89	95	228
Proposed IMC-based Analog Natural Filter	65 nm	3.54	0.11	0.02/ 0.001

"1" to "0" and only 0.5 fJ/bit for valid and null pixels that keep "1" or "0" during diffusion. The total energy consumption is proportional to the noise density. The estimated energy for filtering the 320 × 240 image is 170 pJ assuming the noise density is 10%.

We compared our work with state-of-the-art analog/digital IMC structures as listed in Table 3.2. M. Kang et al. presented a multi-functional deep in-memory architecture accessing multiple rows of a standard 6T SRAM array per precharge, leading to energy-efficiency [91]. However, pulse width modulated (PWM) wordline enabling signals make the controller complex. In a different approach, a $10T$ SRAM was used to store 1-bit filter weights but DACs were needed to convert digital

Table 3.2: Comparison with Prior Work on IMC Hardware Implementations

Metric	This work	JSSC'18 MF [91]	ISSCC'18 Conv [93]	ISSCC'19 T8T [58]
Technology	65nm	65nm	65nm	55nm
Topology	CRAM, 9T	SRAM, 6T	SRAM, 10T	SRAM, T8T
Computation mode	in memory, analog	in memory, digital	in memory, analog	in memory, digital
Algorithm	filter & filling	matched filter	CNN	CNN
Memory size (kb)	90.75*	128	16	3.75
Core area (mm^2)	0.315*	0.254**	0.067	0.045**
Throughput (GOPS)	9293	10.2	10.7	-
Energy Efficiency (TOPS/W)	233	1.94	28.1	18.37-72.1

* include dummy memory rings
** obtained from area of bit-cell times memory size

inputs for analog convolution computing [93]. Finally, digital in-memory multiply-and-accumulate operations were realized by twin-8T SRAM macro but led to large area overhead per cell [58]. In comparison, the proposed approach exhibits a huge improvement in throughput and energy efficiency due to the global fully parallel processing provided by the natural process of charge diffusion.

3.4 Conclusion

In this chapter, a novel CRAM-based analog IMC architecture is presented to fulfill image denoising and region filling tasks. The proposed approach was tested with the binary image frames from a DAVIS sensor setup and consumed 10000× less energy cost than the conventional non-IMC approach realized in the same process. The primary boost in energy efficiency comes from a 864× reduction in latency. We discussed the operation principles of the charge diffusion in the CRAM array architecture and illustrated key performances for NVS applications. The fully parallel natural diffusion architecture reduces the processing time at least to 20 ns and the average power consumption to 170 pJ per frame, resulting in a large throughput and energy efficiency.

The filtered image is acquired by the in-memory computing technique discussed in this chapter. In the next chapter, we will further explore the region proposal approaches to find where the objects are located in the image frame.

Chapter 4

Near-/In-Memory Computing based Region Proposal Approaches

4.1 Introduction

The previous chapter explored the in-memory computing based binary image denoising and image filling approach for neuromorphic vision sensor applications. The raw image is recovered and ready for the subsequent processing. In real applications, however, valid objects occupy only a small part of the frame size. It is a waste of computational energy if we feed the whole frame or a blank frame into the DNN classifier. For efficient denoising and region proposal (RP) operations, [44] has shown a hybrid frame-event approach that processes event based binary images (EBBI) created out of events from a neuromorphic vision sensors (NVS) by generating an interrupt at regular time intervals t_F=66 ms to collect the events accumulated since the last interrupt. We analyzed the same event-based binary image (EBBI) data set used in [39], and found that a valid object occupies only $\sim 1\%$ (bike) up to 15% (bus) of full-frame size when the resolution of the image is 240×180. This implies that there is significant spatial redundancy in active frames, as summarized in Table 4.1. Besides that, the blank frames occupy a large fraction of the video frames in the application of traffic surveillance especially in the remote areas. Fig. 4.1 shows the percentage of total frames having 0 (blank frame), 1,

Table 4.1: Mean object sizes (height×width) with values expressed in pixels at different recording locations [39].

Recording Site	Car/Van	Bus	Bike	Truck
Site 1	16 × 42	31 × 94	15 × 21	22 × 50
Site 2	25 × 47	52 × 107	17 × 22	35 × 61
Site 3	34 × 82	64 × 180	26 × 44	50 × 104

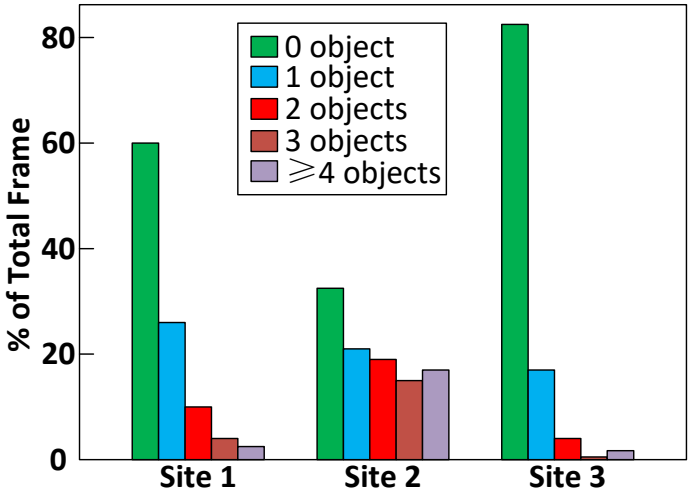


Figure 4.1: Percentage of total frames having 0 (blank frame) 1 ,2, 3, and ≥ 4 objects per frame at three different recording sites. The total number of frames at Site 1, Site 2, and Site 3 are 104389, 124798, and 55153, respectively.

2, 3, and ≥ 4 objects per frame at three different locations. The EBBI data sets were acquired from a DAVIS240 [49] that had been set up at three different traffic junctions to capture moving entities in the scene. The typical objects of interest include cars, vans, trucks, buses, and bikes. Five sample recordings of varying duration were obtained at different lens settings. The total number of frames at Location 1, Location 2, and Location 3 are 104389, 124798, and 55153, respectively[1]. Following these, there is a need for energy-efficient, dedicated hardware – Application Specific Integrated Circuit (ASIC) – to detect the region of interests (ROIs) to exploit spatial redundancy in the valid frames, which triggers the DNN only when a valid region is detected and hence reduces the computation in the subsequent stages confining computing in the ROIs [69].

Apart from the in-memory computing image denoising and filling as described in Chapter 3, we would like to extend to the next stage of image processing – a high accuracy, high energy efficient and low latency region proposal (RP) architecture.

[1]Dataset: https://zenodo.org/record/3839231

In this chapter, we present two different RP algorithms and the dedicated hardware implementations. The first one is edge-event driven RP (EEDRP) approach that is triggered only when the edge of event is detected during scanning the image. This RP algorithm can be realized using near-memory computing by reading the image memory one time and processing locally. The second one is axes projection based region proposal (APBRP) approach by searching the x-axis (horizontal) projection and y-axis (vertical) projection of the image. This RP algorithm can be easily implemented by parallel in memory computing based on our CRAM structure.

4.2 Edge Event Driven Region Proposal (EEDRP)

4.2.1 EEDRP Algorithm

After finishing the pre-processing (denoising and filling) of the raw image presented in Chapter 3, the restored image should be read-out to evaluate the performance of denoising and filling. An interesting question is: can we propose regions simultaneously during the process of image readout? The Edge Event Driven Region Proposal (EEDRP) approach is presented based on an affirmative answer to this question. The event means the active pixel ("1") and the edge event means the rising and falling edge when reading the image. This RP algorithm leverages near-memory computing by reading the image memory serially and processing locally. Compared with the conventional Connecting Component Labeling (CCL) algorithm, the proposed EEDRP only scans the image one time and proposes all the regions of interests (ROIs) with fault-tolerant considerations.

The flow chart of the proposed EEDRP is illustrated in Fig. 4.2, where both the rising edge and falling edge are detected when scanning the image memory array. There are two different processing loops for the two different edges - the green loop represents rising edge and the red one represents falling edge. We tentatively presume the rising edge means a start of a new object. At the falling edge, we consider all the possibilities – is it a new object or a part of previous object or a fault object (noise). In this algorithm, we also considered the case of incomplete objects located at the right and left edges of the memory array. This can happen when the falling edge and the previous rising edge are not in the same row. For normal objects in the middle of the memory array, the rising edge and the corresponding

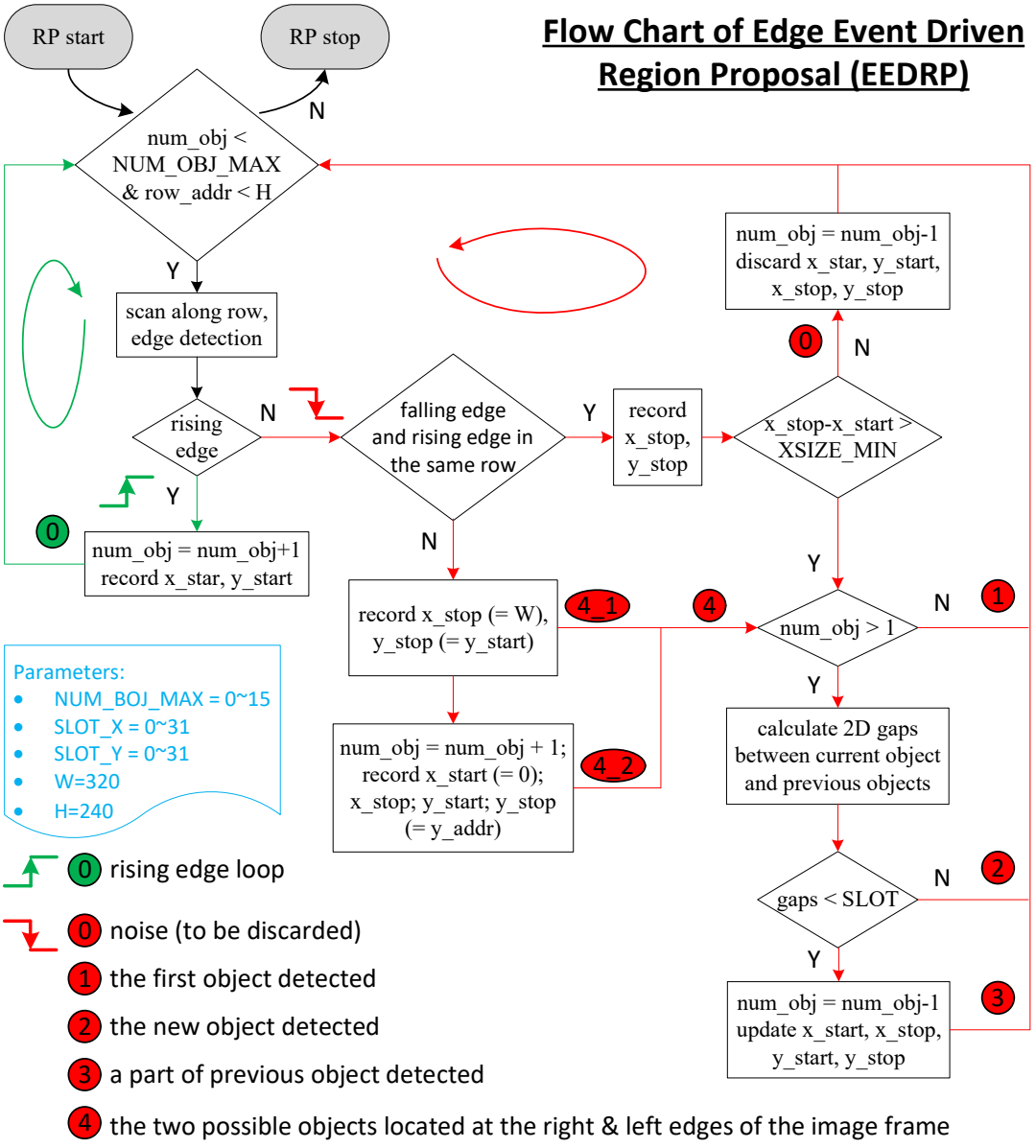


Figure 4.2: The whole flow diagram of the edge event driven region proposal (EEDRP) approach with rising edge loop and falling edge loops.

falling edge are always at the same row. We calibrate the current object by: 1) check the previous assumption and recognize if it's really a new object, or 2) merge it with the previous object if it's a part of it, and 3) discard it if it's just a noise. It is noteworthy that our EEDRP can not only process the recovered image but also the raw image with noise and fragmentation thanks to the fault-tolerance parameters built into the algorithm. These parameters are programmable and flexible to meet various application scenarios. This algorithm is expected to be more energy efficient since computations are performed only at the edge of the objects and are more

rapid compared with those of the conventional CCL region proposal algorithm since the computing finishes after scanning the image only one time.

Next, we will analyse the flow chart of EEDRP algorithm in detail. The RP process starts by scanning the the image memory array and stops when the whole array is scanned out or the number of objects detected is greater than the maximum number of objects NUM_OBJ_MAX. During the image readout, the memory readout data is monitored and the EEDRP algorithm is signalled to wake-up (activate) only when the edge (rising edge or falling edge) is detected. That is the reason we call it as edge event driven region proposal. The detailed processing flow of the EEDRP algorithm is explained as follows:

A. Rising Edge Detected

We tentatively assume that a new object is detected if a rising edge is detected even though it might be false and we will calibrate this assumption at the following falling edge. When a rising edge comes, the process flow follows the green loop labeled as ⓪ shown in the left part of Fig. 4.2. The number of detected objects n, is reset to 0 initially. When the current number of objects (the n^{th}) is larger than the maximum number of objects NUM_OBJ_MAX (which is user programmable), i.e. $n > NUM\_OBJ\_MAX$, the region proposal is finished in advance. Otherwise, it is assumed as a new object and the number of object is accumulated by $n = n+1$. Meanwhile, the coordinate position (i.e. read address) of current rising edge is recorded as x_start and y_start of that object. Then, the algorithm waits for the next falling edge. It is to be noted that the rising edge and the falling edge interleaving appears during the memory readout period.

B. Falling Edge Detected

When the falling edge is detected, we first check whether the falling edge and the last rising edge are in the same row. If they are in the same row, it means all the pixels are the same object from the last rising edge until the current falling edge. The coordinate position (i.e. read address) of current falling edge is recorded as x_stop and y_stop of that object. We then calculate the size of the detected object ("strap") by subtracting the stop and start x-address and compare it with the minimum of size of a valid object along the x-axis. If $x\_stop - x\_start > XSIZE\_MIN$, it is a valid object, otherwise it is a noise and needs to be discarded, as the red loop labeled as ⓪ shows at the top right corner of Fig. 4.2. If the current number of objects is $n = 1$, then the current detected object is (part of) the first object in this

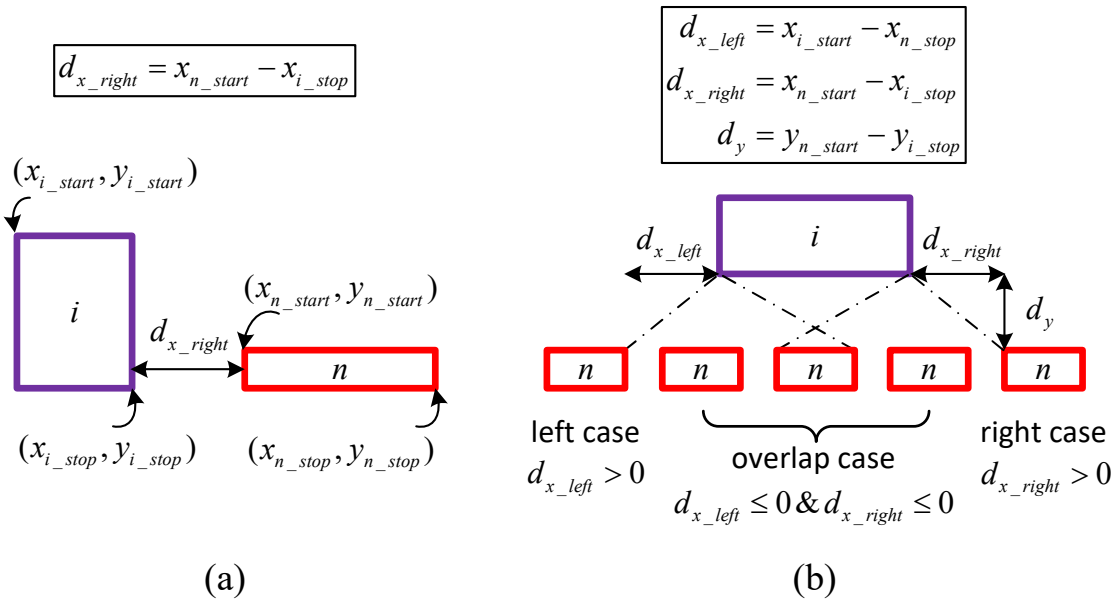


Figure 4.3: Calculation of the distances of detected object (red colored) and one of the previously detected objects (purple colored). (a) When they are on the same row, only the lateral distance between them needs to be calculated; (b) When they are on different rows, both lateral and vertical distances have to be calculated.

frame, as the red loop labeled as ① shows in the right side of Fig. 4.2. Because there is no previous detected object, the first object does not need to be compared and updated.

Keep in mind that the event based binary image (EBBI) suffers from noise (due to the leakage current of the pixel's photodiode) and fragmentation (due to a lack of contrast, e.g. the plane surface or occluded by stationary object). Our EEDRP is designed to tackle these issues. To avoid a single object being proposed as several fragmented objects, we introduce the user programmable parameters $SLOT\_X$ and $SLOT\_Y$ to tolerate a gap in the object body and merge the fragmented object in two dimensions (both vertical and horizontal). When the number of detected objects at the current moment is $n > 1$, the detected object could be a new object or part of one of the previous objects. In order to determine which of those alternatives is true, we calculate the distances between the current object and previous objects in two dimensions. Fig. 4.3 illustrates the distance calculation of the current detected object (the n^{th}, red colored) and the i^{th} previous object (purple colored), where $i \in [1, n]$. Because EEDRP uses parallel computing, the previous objects must be located at the left or top of the current object.

When the previous object is located on the same row, the current object is at the right of previous object, so only need calculate the lateral distance as shown in Fig. 4.3(a). When the previous object is located at the top of current object, there are three cases as listed in Fig. 4.3(b). The lateral relation of the current object and the previous object could be:

(1) The current object is at the left side of the previous object if $d_{x\_left} > 0$.

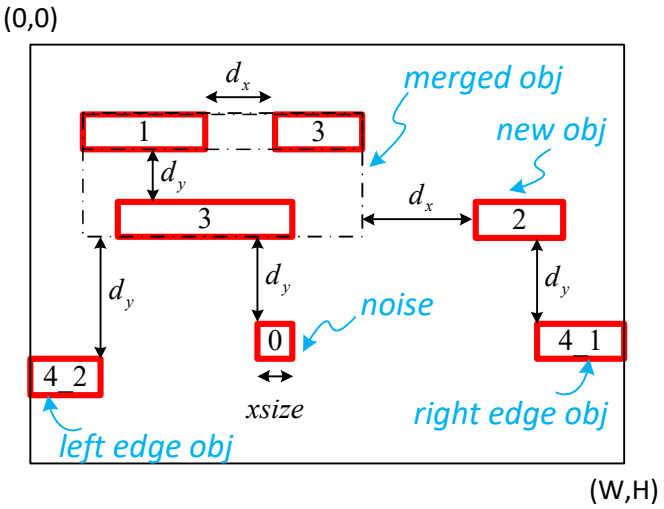
(2) The current object is at the right side of the previous object if $d_{x\_right} > 0$.

(3) The current object is (partially or totally) overlapped with the previous object if $d_{x\_left} \leq 0$ **and** $d_{x\_right} \leq 0$.

If both of the distances are greater than the user programmable parameter, i.e. $min(d_{x\_left}, d_{x\_right}) > SLOT\_X$ **and** $d_y > SLOT\_Y$, the current object is recognized as a new object, as the red loop labeled as ② shown in the right of Fig. 4.2. Otherwise, if one of the distance is less than the user programmable parameter, i.e. $min(d_{x\_left}, d_{x\_right}) < SLOT\_X$ **or** $d_y < SLOT\_Y$, the current object is recognized as a part of the previous object. This means the earlier assumption (we previously assume that a new object is detected if a rising edge is detected) is wrong, and so we withdraw this assumption by reducing the number of objects (i.e. $n = n - 1$) and updating the bounding box (x_start, x_stop, y_start, and y_stop). This case is illustrated as the red loop labeled as ③ shown in the right of Fig. 4.2.

Another important case is that when two objects appear simultaneously at opposite sides (left edge and right edge of the image frame). In this case, the falling edge and the last rising edge appear at different rows, depicted as the red loop labeled as ④ shown in Fig. 4.2. We first record the stop position of the right edge object, i.e. $x\_stop = W$, and $y\_stop = y\_start$, where W is the width of the image. And then the number of objects is increased by 1 (i.e. $n = n + 1$), so we can record the coordinate position (i.e. read address) of the second object at the left edge, i.e. $x\_start = 0$, $x\_stop = x\_addr$, $y\_start = y\_addr$, and $y\_stop = y\_addr$.

All cases mentioned above can be illustrated in a single image frame, as shown in Fig. 4.4, which depicts the following cases: ⓪ noise; ① the first object; ② new object; ③ parts of previous object; ④ objects located at the right edge and left edge of the image frame. The numbers labeled in the objects correspond to the



No.	comparison	result	update
0	$xsize < XSIZE\_MIN$	noise	discard
1	$xsize > XSIZE\_MIN$	1st object	wait for the next edge
2	$d_x > SLOT\_X$	new object	wait for the next edge
3	$d_x < SLOT\_X$ $d_y < SLOT\_Y$	parts of previous object	$x/y_{n\_start} = \min(x/y_{i\_start})$ $x/y_{n\_stop} = \max(x/y_{i\_stop})$
4_1	$d_y > SLOT\_Y$	right edge object	$x_{n\_stop} = W, y_{n\_stop} = y\_addr$
4_2	$d_y > SLOT\_Y$	left edge object	$n = n+1$ $x_{n\_start} = 0, x_{n\_stop} = x\_addr$ $y_{n\_start/stop} = y\_addr$

Figure 4.4: Object locations for all scenarios and the corresponding comparison and update at the falling edge. The numbers labeled correspond to the red loops in Fig. 4.2.

red loops in Fig. 4.2. The table below lists the comparison and update for each scenario.

Finally, for all the four cases mentioned above, the region proposal is considered to be finished if the row address reaches to H, where H is the height of the image. It is to be noted that the row address starts from 0 in hardware. If the row address is less than H, it waits for the next rising edge.

The pseudocode of the proposed EEDRP algorithm is listed in Algorithm 1. It is to be noted that this pseudocode is to explain the algorithm clearly but not exactly

Algorithm 1 EEDRP: Edge Event Driven Region Proposal

Input: x_addr, y_addr: memory read address
$dout$: memory readout data

Output: num_obj: number of objects in the image frame
rp_x: x coordinate positions of all proposed regions, the format is $rp\_x = [x_{1\_start}, x_{1\_stop}, ..., x_{n\_start}, x_{n\_stop}]$
rp_y: y coordinate positions of all proposed regions, the format is $rp\_y = [y_{1\_start}, y_{1\_stop}, ..., y_{n\_start}, y_{n\_stop}]$

/* $(x_{i\_start}, y_{i\_start})$ and (x_{i_stop}, y_{i_stop}) stands for the start and stop positions of the i^{th} object, respectively, where $i \in [1, num\_obj]$. */

initialize $n \leftarrow 0,\ num\_obj \leftarrow 0,\ rp\_x \leftarrow 0,\ rp\_y \leftarrow 0$
/* read the memory array and monitor signal $dout$ */
if $n < NUM\_OBJ\_MAX$ **then**
  for $y\_addr \leftarrow 1$ **to** H **step** 1 **do**
    for $x\_addr \leftarrow 1$ **to** W **step** 1 **do**
      if $dout$ **and** *(***not** $dout\_last$*)* **then** // rising edge detected
        /* hypothesize it is a new object and record it down */
        $n \leftarrow n + 1$
        $x_{n\_start} \leftarrow x\_addr$
        $y_{n\_start} \leftarrow y\_addr$
      else if *(***not** $dout$*)* **and** $dout\_last$ **then** // falling edge detected
        if $y\_addr = y_{n\_start}$ **then** // at the same row
          $x_{n\_stop} \leftarrow x\_addr$
          $y_{n\_stop} \leftarrow y\_addr$
          do $EEDRP\_UPDATE$ // compare and update
        else // at the different row
          /* record the object at the left edge of the frame */
          $x_{n\_stop} \leftarrow W$
          $y_{n\_stop} \leftarrow y\_addr$
          do $EEDRP\_UPDATE$ // compare and update
          /* record the object at the right edge of the frame */
          $n \leftarrow n + 1$
          $x_{n\_start} \leftarrow 1$
          $x_{n\_stop} \leftarrow x\_addr$
          $y_{n\_start} \leftarrow y\_addr$
          $y_{n\_stop} \leftarrow y\_addr$
          do $EEDRP\_UPDATE$ // compare and update
        end
      end
    end
  end
  $num\_obj \leftarrow n$
  break ; // end of RP
else
  break ; // end of RP
end

Algorithm 2 EEDRP UPDATE

Function `EEDRP_UPDATE`$(n, rp\_x, rp\_y, SLOT\_X, SLOT\_Y, XSIZE\_MIN)$: //
  Input: $n, rp\_x, rp\_y, SLOT\_X, SLOT\_Y, XSIZE\_MIN$
  Output: n, rp_x, rp_y `/* updated values */`
1 **if** $x_{n\_stop} - x_{n\_start} < XSIZE\_MIN$ **then** `// noise`
2 $n \leftarrow n - 1$ `// discard`
3 break
4 **else**
5 **if** $n > 1$ **then** `// check and update if needed`
6 $i \leftarrow 1$
7 **while** $i < n$ **do**
8 $d_{x\_left} \leftarrow x_{i\_start} - x_{n\_stop}$
9 $d_{x\_right} \leftarrow x_{n\_start} - x_{i\_stop}$
10 $d_y \leftarrow y_{n\_start} - y_{i\_stop}$
11 **if** $(d_{x\_left} > 0$ **and** $d_{x\_left} < SLOT\_X)$ **or** $(d_{x\_right} > 0$ **and** $d_{x\_right} < SLOT\_X)$ **or** $(d_{x\_left} \leq 0$ **and** $d_{x\_right} \leq 0$ **and** $space\_top < SLOT\_Y)$ **then** `// merge`
12 $x_{i\_start} \leftarrow \mathbf{min}(x_{i\_start}, x_{n\_start})$
13 $x_{i\_stop} \leftarrow \mathbf{max}(x_{i\_stop}, x_{n\_stop})$
14 $y_{i\_stop} \leftarrow \mathbf{max}(y_{n\_stop}, y_{n\_stop})$
15 $n \leftarrow n - 1$
16 $i \leftarrow i - 1$
17 **else** `// keep`
18 $i \leftarrow i + 1$
19 **end**
20 **end**
21 **else** `// the first object does not need to be updated`
22 break
23 **end**
24 **end**
25 **End Function**

what is implemented in hardware, e.g. address and index starts from 0 in real hardware design. Once the rising edge is detected, it is seen as a new object and the start location (x_start, y_start) is recorded. Once the falling edge is detected, the stop location (x_stop, y_stop) is recorded. Then the current object ("strap") is compared with previously detected objects and updated if necessary, according to the previously detected objects.

The details of how to update the coordinate positions are illustrated in Algorithm 2. It first calculates the horizontal distance (size along x-axis) of the current object and check if it is a noise or not. Here we introduce a customer programmable parameter $XSIZE\_MIN$. If the horizontal size is less than the parameter $XSIZE\_MIN$, the current object is a noise and will be discarded. Otherwise, it calculates the

spatial distances at left, right, and top between the current object and each of the previously detected objects. Here, again, two other customer programmable parameters, $SLOT\_X$ and $SLOT\_Y$ were introduced. If the spatial distance is less than both of the $SLOT$ parameters, the current object is recognized as a part of the previous one and merged with it, resulting in a larger bounding box covering both of them. Otherwise, the current object is recognized as a new object.

4.2.2 EEDRP Hardware Architecture

4.2.2.1 Top Level

In this section, we will discuss the hardware architecture of the proposed edge event driven region proposal (EEDRP). Fig. 4.5 shows the top-level view of the event based binary image (EBBI) processor. The commercially available DAVIS sensor [49] is used as the neuromorphic vision sensor (NVS) with resolution of 240×180 pixels. Active (ON or OFF) events are captured as temporal sequences and then encoded by an address event representation (AER) encoder. The AER signals are received and decoded by the AER decoder synchronously in our EBBI processor. The decoder module follows the AER protocol with a 10-bit data line AER_Data that includes address and polarity of an event, as well as two handshake signals – AER_nreq and AER_nack – used to coordinate both-way communication. The decoded event information $xAddr$ and $yAddr$ are than temporarily stored in a 128×32 bit first-in-first-out (FIFO) working as an asynchronous buffer.

The EBBI processor has two clock domains: AER_CLK for processing raw events from the NVS, and sys_clk for other blocks. The AER communication protocol takes several AER_CLK cycles to complete the transfer of an event address, and for this reason, the frequency of AER_CLK is allowed to be higher than the core operation frequency sys_clk. The two clocks act as the write clock and read clock of the FIFO separately, determining the depth of the FIFO block.

The row and column bus signals from the CRAM controller are 240-bit and 320-bit respectively, following the dimensions of the CRAM macro ($W = 320, H = 240$). The memory size is chosen to be compatible with quarter video graphics array (QVGA) resolution, as well as lower resolution images from NVS [86], [94]. The region proposal logic is implemented by a finite state machine (FSM). The 320×240

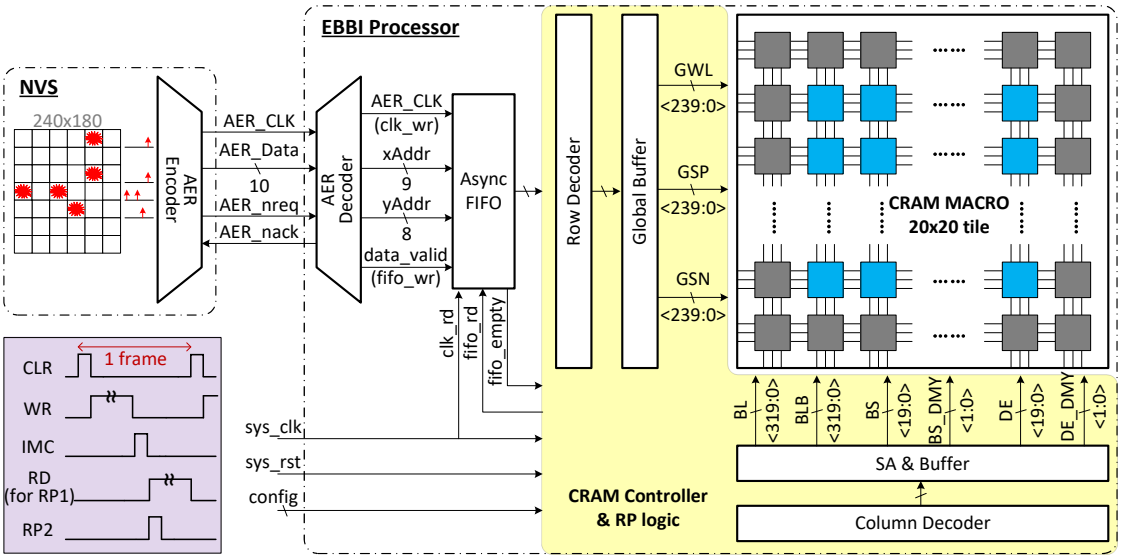


Figure 4.5: Top level of the EBBI processor. The chip consists of an address event representation (AER) decoder module, a 128 ×32 bit asynchronous first-in first-out (FIFO) buffer, a digital controller, and a 320 × 240 CRAM macro partitioned as 20 × 20 tiles. The waveform at the bottom left of the image shows the different operation modes sequentially performed by the processor.

memory array is partitioned into 20 × 20 tiles and each tile is composed of a mini array with 16 × 12 bit cells. As the grey part shows in Fig. 4.5, a dummy tile ring is added to make each cell in the array match well and offer the identical surroundings across the whole array. Although these dummy cells are used to create matching surroundings for charge diffusion and are not used to compute in our work, they can be still used as storage cells in normal memory mode because they are addressed by the decoder as well. To minimize the dynamic bit-line power dissipation [95] in the write mode, the CRAM macro is divided into $N_{bank} = 320/16 = 20$ banks. Only one bank select (BS) signal is active to enable the selected memory bank while writing. All the signals are generated or controlled either in parallel or independently to perform operations.

As shown in the simple waveforms illustrated in the bottom left of Fig. 4.5, the EBBI processor performs five primary operation modes sequentially: (1) clear mode to reset the whole memory array, (2) write mode to receive the raw binary image, (3) in-memory computing mode to preprocess the image, such as denoising and filling, (4) read mode to readout the image and perform near-memory computing based region proposal simultaneously, and (5) in-memory computing mode for region proposal (this will be discussed in section 4.3).

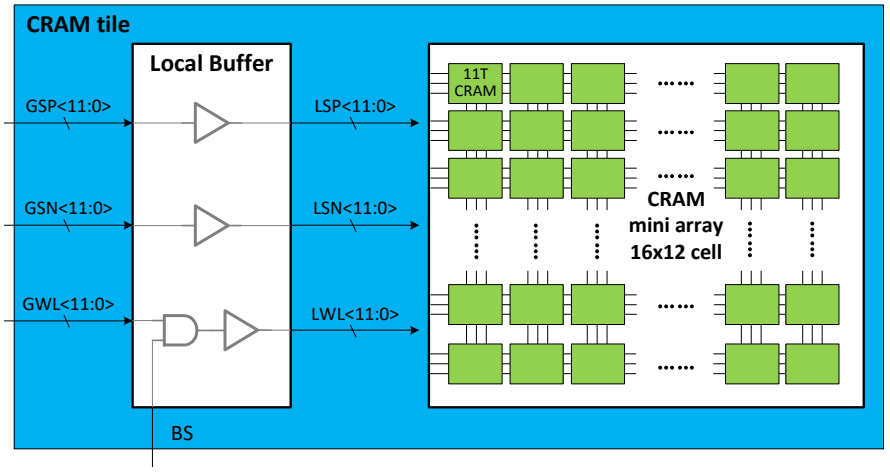


Figure 4.6: The CRAM tile consists of local buffers and a CRAM mini array with 16×12 bit-cells. Each CRAM tile is enabled by activating the bank select signal BS.

The details of the CRAM tile are depicted in Fig. 4.6. The local buffers drive the local control signals for the CRAM mini-array itself composed of 16×12 bit cells. GWL, GSP and GSN are the global control signals for all the CRAM cells across the array (we will discuss the operation principle in detail later). If we connected all the nodes together directly, an extra big global buffer would be necessary to drive the resulting significant parasitic capacitance due to both wire capacitors and gate capacitors that would, therefore and suffer from severe propagation delay. In our design, the global signals only drive the local buffer of each tile to reduce the parasitic capacitor. The mini-array is driven by the local control signals LSP and LSN, which are driven by the local buffers. The global word line (GWL) drives the buffer of each tile only instead of all the cells in the row. The access of the mini array is controlled by the local word line (LWL), which is activated only when the bank select (BS) signal is selected. The local buffer approach effectively reduces not only the power consumption of the drivers but also the time latency of the signal propagation. The bank partitioning enables writing (or reading) only a small part of the macro array at one time and the other major part is in standby mode. This remarkably decreases the dynamic power dissipation, which is the dominant power dissipation in these CMOS circuits.

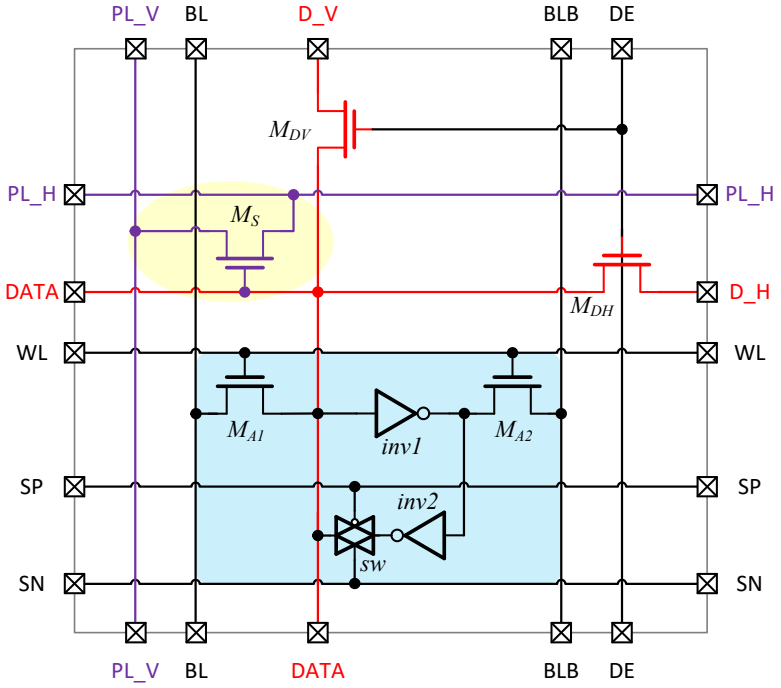


Figure 4.7: The revised 11T CRAM bit-cell. It consists a 6T SRAM with a CMOS switch, two orthogonal NMOS transistors M_{DH} *and* M_{DV} for charge diffusion, and a versatile NMOS transistor M_S. Line and block colours in the schematic are for clarity only.

4.2.2.2 11T CRAM Cell

Fig. 4.7 illustrates the schematic of the revised CRAM bit cell, consisting of 11 transistors (11T). It can be divided into three parts: (1) a 6T SRAM ($inv1, inv2, M_{A1}$, and M_{A2}) with a transmission gate (TG) switch sw inserted after the output of inverter $inv2$ and before the storage node, as shown in the light blue area, (2) an NMOS transistor M_S works together with access transistor M_{A1} as a conventional $1T1C$ DRAM, besides that, M_S also acts as storage capacitor used for in memory computing, as shown in the light yellow area, and (3) the two red coloured vertical and horizontal NMOS M_{DV} and M_{DH} are employed to connect the cell to its 2-dimensional neighbours when in memory computing mode.

This CRAM bit cell is the revision of the previous version [45] presented in Fig. 3.4 in Chapter 3. In the previous version, we reduced the overhead cell area excessively and damaged the stability and reliability of the memory. The PMOS switch is now replaced with a CMOS switch to improve the data transmission and, hence, increase the output load capacity of the latch in the positive feedback loop. We use two-ended access approach (for writing and reading), just like a standard 6T SRAM

instead of single-ended access to improve the write ability and read stability. Both methods make the CRAM cell more robust and reliable. Another change is that the NMOS M_S is not only used as storage capacitor for in memory computing based image denoising and filling (as described in Chapter 3), but also as the projection device for in-memory computing based region proposal. We will discuss this in section 4.3.2. The bit cell of the proposed 11T CRAM occupies 2.18×1.82 um^2. The area overhead is 1.12× larger than the previous 9T CRAM structure discussed in Chapter 3.

It is to be noted that transistor M_S has three different roles in the CRAM bit cell. Firstly, M_S together with the access transistor M_{A1} form a conventional $1T1C$ DRAM structure. Secondly, M_S is used as MOS capacitor (by configuring $PL\_H = 0$, and $PL\_V = 0$) for in memory based image filtering and filling. The above two roles of M_S are the same as our previous design described in Chapter 3. Finally, it also works as a projection device to pass the stored data to the horizontal projection line (PL_H) or the vertical projection line (PL_V) for in memory computing based region proposal. We refer to it as working in projection mode and will discuss this in detail later in section 4.3. The primary principles of the proposed CRAM in different operation modes (excluding projection mode) are analysed as follows. It is largely similar to the CRAM presented in Chapter 3, but with some differences as pointed out below.

A. Clear Mode

Since the NVS only asynchronously reports pixels with value "1", which are generally sparse, the whole memory array needs to be fully cleared or rested before the writing mode, otherwise all the "1" pixels of the last image frame will be superimposed on the current image. The bit-line (BL) and its complement (BLB) are driven to 0 and V_{DD}, respectively, to clear the CRAM macro. Once a word line (WL) is activated, all bit cells in that particular row get cleared. Enabling only one WL in every clock cycle extends the execution time (240 rows take 240 clock cycles). In contrast, setting all WLs as 'high' simultaneously leads to a higher surge current at the bit-line pair (BL/BLB) because each bit-line pair will write "0" for all the 240 cells in a column. The surge current coupled with small wire size due to layout constraints may further trigger electromigration-induced reliability issues. Furthermore, the output load capacity and area of both the BL and BLB drivers need to be very high to clear all 240 CRAM bit cells at once (worst case scenario).

To address these concerns, we enable only 12 WLs in every clock cycle, hence $240/12 = 20$ clock cycles are required to erase the whole memory array. The clear mode can be seen as a special case of write mode where the written data are all "0".

B. Write Mode

Since the addresses of the events are random and non-contiguous by nature, the memory is under the risk of cross-talk during the write mode, which may overwrite the half-selected cells. Consequently, we implemented a special writing mechanism enabling only one single particular WL and bit-line pair (BL to V_{DD} and BLB to 0) at a time following the address signal. The CRAM macro shown in Fig. 4.5 is divided into 20 separate banks to minimize the dynamic power dissipation of the word-line during writing. In each clock cycle during write mode, only one particular bank is enabled. The local word line (LWL) is enabled by both the global word line (GWL) and bank select (BS) signals. The column decoder and buffer drive the data ("1") and its complement ("0") on the bit-line pair BL and BLB, respectively. The rest of the bit-line pairs are precharged to V_{DD} to suppress the read disturb issue of the half-selected (HS) cells in the selected bank.
In write mode, we do not care about the vertical and horizontal projection lines PL_V and PL_H shown in Fig. 4.7. We set both of them to "0" just for the subsequent IMC mode. The control signal pair of CMOS switch sw is set as $SP = 0$ and $SN = 1$ to enable the latch so the written data would be latched in the bit cell. Deactivation of the charge diffusion enable signal $DE = 0$ to cut off the charge diffusion path.

C. IMC Mode

As we mentioned before, the event based binary image (EBBI) suffers from noise (due to the leakage current of photo-diode) and fragmentation (due to a lack of contrast, e.g. the plane surface or occluded by other objects). In general, the background comprises a majority of "0" valued pixels that surround some noisy "1" pixels, which should be removed by a filtering operation. On the contrary, an object comprises a majority of "1" valued pixels that surround some 'holes' of "0" pixels, which should be filled by a filling operation. The CRAM is a dedicated design for both image filtering and filling leveraging parallel in-memory computing.

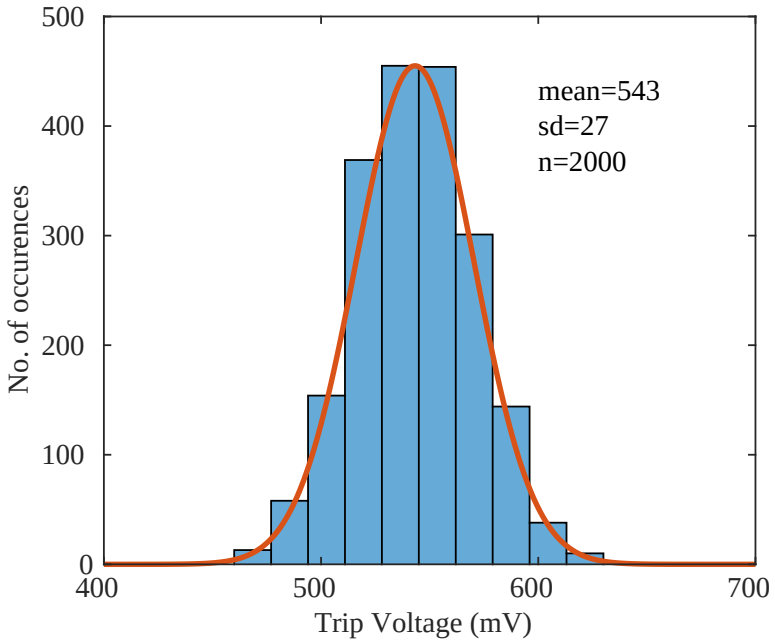


Figure 4.8: Distribution of the trip voltage of $inv1$, as resulted from Monte Carlo simulations generated in 2000 runs. Notice that the standard deviation sd is only 5% of the average.

To implement the parallel image filtering and filling across the whole array, the CRAM controller inactivates all the word lines (WLs) to disable the access of bit cells. To release the stored charge, all the CMOS switches sw are also disabled by setting $SP = 1$ and $SN = 0$. The vertical and horizontal located transistors M_DV and M_DH will then operate as resistors whose resistance is determined by the gate voltage DE – thus, the 2-dimensional charge diffusion RC network is formed. This voltage is programmable by a 4-bit DAC controlled by $diffusion\_r\_sel < 3 : 0 >$. The diffusion time depends on the pulse width of DE signal programmed by $diffusion\_t\_sel < 1 : 0 >$. Both the projection lines PL_V and PL_H are connected to GND, so the NMOS M_S acts as a capacitor that is charged to V_{DD} or GND depending on the values stored in the bit cell. Initially, the values stored in bit cells are digital signals – either "0" or "1". As time goes on, the charges of each cell are redistributed and the voltages are analog signals – between "0" and "1". Hence, a sense amplifier (SA) is required to sense these analog signals and convert them back to digital signals and then store each value into a bit cell again for future processing.

D. Sensing and Recovering mode

When IMC is done, all the data stored on the MOS capacitors are analog voltages between V_{DD} and GND. The diffusion-enable signal DE is deactivated to cut off the diffusion paths. The vertical and horizontal projection lines PL_V and PL_H do not matter in this mode and are reset to "0". The inverter $inv1$ is carefully designed and the trip voltage is used as a reference to sense the redistributed charge information. The Monte Carlo (MC) simulation results for the trip voltage of $inv1$ are shown in Fig. 4.8. The mean value is 543 mV and the statistical distribution was found to be largely insensitive to mismatch and process variations, with a standard deviation of 27 mV. In this mode, the transmission gate control signals SP and SN are set to GND and V_{DD} separately to enable the back-to-back inverter latch. The first inverter $inv1$ also acts as a comparator (i.e. SA) using its well-designed trip voltage. If the voltage of one particular cell is greater than the trip voltage of $inv1$, a bit "1" is detected and stored in the cell, otherwise, a bit "0" is stored in the cell. The sensing and recovering of the redistributed data only takes one clock cycle due to the globally parallel computing, leading to high time efficiency.

E. RP Mode

The edge event-driven region proposal (EEDRP) is performed while reading the memory; hence, the RP mode is also the read mode of the memory. In this mode, the CRAM is configured as a common SRAM by keeping the transmission gate enabled. Unlike the single-ended accessing in the last version, both BL and BLB are first pre-charged to V_{DD} and then kept floating when WL is asserted. As a consequence, one of the bit lines (either BL or BLB) gets discharged faster than the other one according to the values stored in the cell. The difference between the bit line voltages can be then easily sensed by a differential sense amplifier. The read operation is divided into three cycles, namely precharge, access and sense. In our approach, the entire row data are read out to the latch first and then output one by one to reduce the WL power dissipation. The pre-processed image can be readout by scanning the whole memory array (320×240 clock cycles). Meanwhile, all the regions are proposed by detecting the edge of readout signal, as discussed in section 4.2.1.

In summary, the EBBI processor discussed in this chapter is the revision and extension of the design presented in Chapter 3. Major changes are summarized as follows: (1) the CRAM cell is redesigned and optimized to improve the stability (both writability and readability) of the memory during write and read operations,

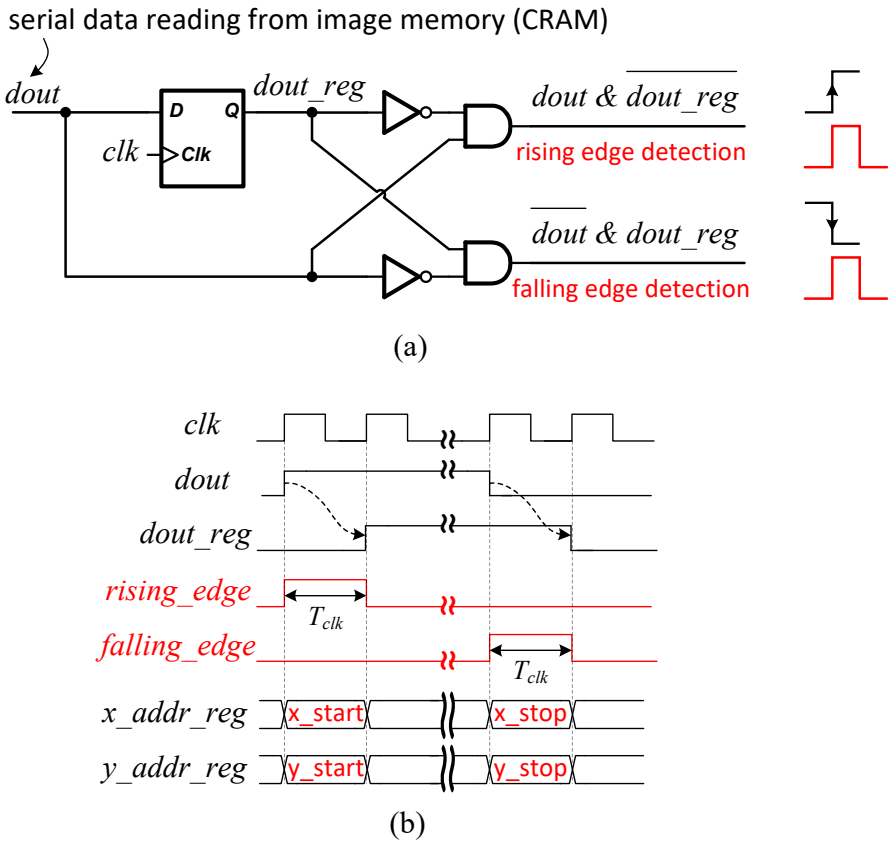


Figure 4.9: (a) Edge detect circuit implemented by basic logic gates (DFF, NOT, AND). (b) Waveforms of the input signal *dout*, its delayed signal by one clock cycle with a DFF register, and the final output signals *rising_edge* and *falling_edge*, as well as the coordinate positions (reading address).

(2) the 320 × 240 memory array is partitioned into 20 × 20 tiles and each tile is composed of a mini array with 16 × 12 bit cells and local buffers, (3) we designed the dedicated edge event-driven region proposal approach leveraging near memory computing, and (4) in-memory computing histogram based region proposal is realized by exploiting the CRAM cell.

4.2.2.3 Edge Detector

The computation of our proposed EEDRP is driven by the event edge, so the first part of the proposed RP approach is to detect the event edge. By reading the memory array row by row, a data out signal *dout* is acquired and fed into an edge detector (ED). The gate level ED circuit is depicted in Fig. 4.9(a) consisting of an edge-triggered D flip-flop (DFF), two NOT gates, and two AND gates. Fig. 4.9 (b) illustrates the typical waveforms of the key nodes of ED circuit. The signal

dout_reg is the register output signal of DFF, which is the signal *dout* delayed with one clock cycle. The rising edge detection signal *rising_edge* pulses when $dout = 1$ and $dout\_reg = 0$ while the falling edge detection signal $falling\_edge$ pulses when $dout = 0$ and $dout\_reg = 1$. The pulse width of the edge detected signals is one clock cycle T_{clk}, triggering the EEDRP computation. The memory readout and edge detection are synchronous with the clock signal *clk*, so there is one clock cycle delay between the rising/falling edge and the corresponding read address. Consequently, we take the one clock delayed address (x_addr_reg, y_addr_reg) as the coordinates positions (x_start, y_start, x_stop, y_stop) when edge is detected, as shown in the bottom of Fig. 4.9(b).

The controller of the EEDRP algorithm is firstly modeled by RTL HDL and synthesized to obtain the gate level design which is then combined with the analog CRAM block as a part of our mixed signal IC.

4.2.3 Demonstration

An illustration of the proposed edge event-driven region proposal (EEDRP) is shown in Fig. 4.10 for a typical event-based binary image (EBBI) in 3 cases: 1) multiple objects along both axes, 2) fragmented object, and 3) noise pixel. It is to be noted that our EEDRP can deal with both the raw image or the filtered image with different parameter configurations. Fig. 4.10(b) shows the corresponding sequential waveforms of the output signal *dout* when reading the image. A "1" pixel outputs a high voltage level and a "0" pixel outputs a low voltage level. The EEDRP acts only when rising or falling edges are detected. Fig. 4.10(c) explains the whole process of how the EEDRP algorithm finds the region, discards noise, and merges the fragments of a object. The coordinate positions are recorded in the format of start and stop coordinates on x-axis and y-axis, respectively. The top-left most position is considered as the origin $(0, 0)$ of the 15×15 image array. The EEDRP begins with the first "1" pixel located at (7, 0), and records the completed coordinate position (7, 7, 0, 0) as the first object. Then, the width of this object ($width = 7 - 7 + 1 = 1$) is compared with the minimum size along the x-axis ($XSIZE\_MIN = 4$). Because $width < XSIZE\_MIN$, it is recognized as a noise and discarded finally. There is no any action during reading the first row because no edge is detected. The next action happens at the second row and records the

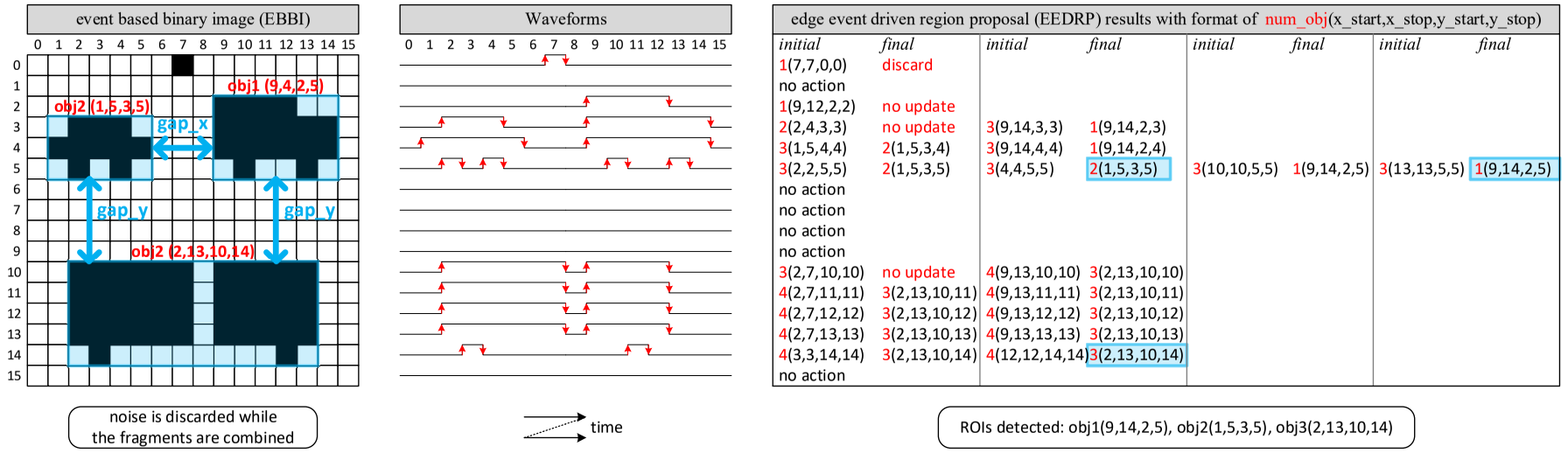


Figure 4.10: Demonstration of the proposed edge event driven region proposal (EEDRP) for a typical event base binary image (EBBI). (a) A complex EBBI applied for 1) multiple objects along both axes, 2) fragmented object, and 3) noise pixel. (b) The corresponding sequential waveforms of the output signal *dout* when reading the image. The waveforms are drawn with time in terms of left to right and top to bottom. All the rising and falling edges are highlighted. (c) The process of the EEDRP algorithm and the final proposed region of interests (ROIs).

object (9, 12, 2, 2) as the first object. Since the width is larger than $XSIZE\_MIN$, this object is retained. The next action happens at the third row and records the object (2, 4, 3, 3) as the second object. In the same row, the third object (9, 14, 3, 3) is detected. This object is then combined into the first object and updated as (9, 14, 2, 3) because the vertical gap is less then the slot of objects $SLOT\_Y = 4$.

The algorithm works like this until the last edge event, and finally proposes three region of interests (ROIs) – object 1 (9, 14, 2, 5), object 2 (1, 5, 3, 5), and object 3 (2, 13, 10, 14), as the light blue regions shown in Fig. 4.10(a). As we can see, the noise pixel is discarded and the fragmented object is reconstituted as a complete one. The proposed EEDRP proposes very accurate ROIs due to its detection of the edge of objects. By appropriate parameter configuration, it is suitable for various application scenarios – high/low noise, more/less fragmentation etc.

4.3 Axes Projection Based Region Proposal (APBRP)

Traditionally, a sliding window and its improved version of selective search technique have been used for object detection; however, both of them are exhaustive search methods and are computationally expensive [96], [97]. Recently, histogram-based RP (HIST-RP) [44], [39] were explored in software to reduce the computation cost. It calculates the X and Y histograms (H_X and H_Y) for the image and finds regions in these two 1-D signals by finding consecutive entries that are higher than a threshold value. Nevertheless, the HIST-RP approach suffers from false bounding boxes when multiple objects occur due to interactions between the projections onto horizontal or vertical axes. Besides, there is no dedicated hardware implementation reported.

In this section, we propose an advanced approach that is superior to HIST-RP from both algorithm and hardware perspectives. HIST-RP needs a counter for each row and each column to accumulate the "1" pixels of each row and column. From the hardware point of view, it needs $W + H$ accumulators (ACC) and comparators (CMP). Our proposed axes projection-based region proposal (APBRP) is similar to HIST-RP but much simpler and it is in-memory computing-based so that the

accumulator is not needed, efficiently reducing the hardware overhead of area, energy, and latency.

We have described the different operation modes of a CRAM cell (clear, write, IMC for image pre-processing, and so on). In this section, the CRAM works in a new operation mode – projection mode. As the purple lines shown in Fig. 4.7, the data stored in the bit cell can be projected to the horizontal projection line PL_H and vertical projection line PL_V, respectively, by the projection device M_S. Next, we will discuss the APBRP software algorithm and its hardware implementation separately.

4.3.1 APBRP Algorithm

The algorithm of the proposed APBRP is similar to HIST-RP but much simpler and overcomes the drawbacks of HIST-RP. The APBRP can smartly deal with multiple objects and propose the exact bounding boxes and coordinate positions. To acquire the region coordinates, it first calculates X and Y histograms (H_X and H_Y) for the image by projecting the image onto the horizontal and vertical axes, respectively, and then searches regions in these two 1-D signals by finding consecutive entries whose individual value is higher than a threshold. The actual 2D region is obtained by finding the intersections of the X and Y regions discussed earlier (Fig. 2.9). However, the conventional histogram based region proposal (HISTRP) approach cannot propose an accurate bounding box when projections along one direction overlap for multiple objects case, as illustrated in Fig. 4.11. The purple- and blue-colored bounding boxes represent HISTRP and APBRP, respectively. Because projections of the two objects along the y-axis overlap each other, the conventional HISTRP approach proposes inaccurate vertical positions, while the APBRP approach overcomes this shortcoming and proposes correct bounding boxes. For the case of multiple objects in a image, the projection overlap issue of conventional HISTRP is fixed by projecting and searching locally in our APBRP approach. The fragmentation issue is overcame by using programmable threshold to merge the adjacent fragments together as one object. The calculation of histograms is parallelly processed by the analog in-memory computing architecture. We will discuss the details of hardware implementation in section 4.3.2.

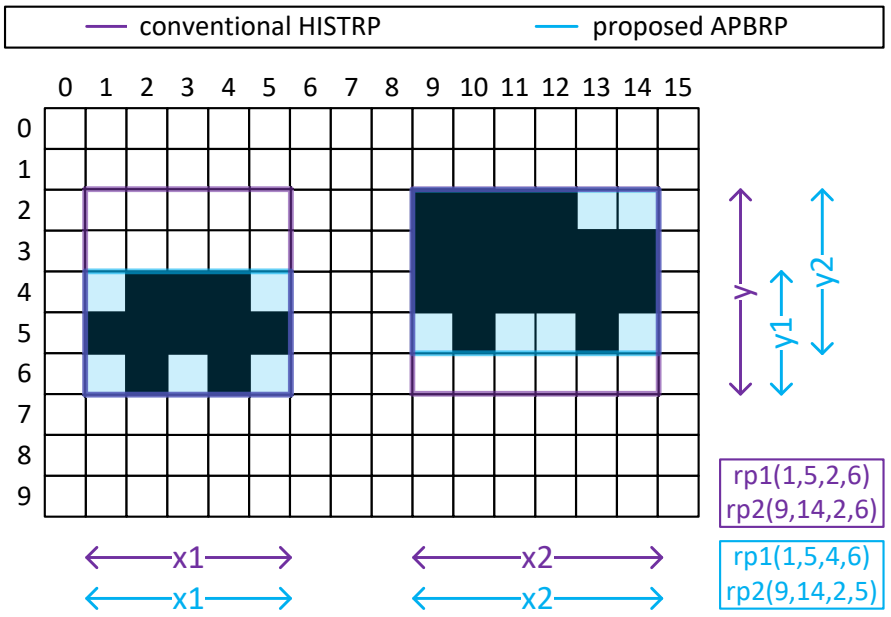


Figure 4.11: Demonstration of the conventional histogram-based region proposal (HISTRP) and the proposed axes projection-based region proposal (APBRP). The HISTRP proposes inaccurate bounding boxes when the projections overlap, while the APBRP overcomes this shortcoming by using a local projection. The region proposal results listed at the right bottom corner are given in the format $(x\_start, x\_stop, y\_start, y\_stop)$.

Fig. 4.12 illustrates the operation principle of APBRP for multiple objects. Fig. 4.12(a) shows a typical case of side view application where all the objects are in a single traffic flow, and the vertical projections of different objects interact (overlap). Fig. 4.12(b) shows a typical case of top view application where objects are in different traffic flows (two in this demonstration) and the projections of different objects interact (overlap) in both directions. This is a challenge for conventional HIST-RP. We will explain how the proposed APBRP fixes the overlapping issue in Algorithm 3. It is to be noted that the application scenario of side view is simple because all objects look like in a single traffic flow but it will make the region proposal difficult because the objects may be overlap in the frame. If two or more objects overlap, most region proposal approaches recognize them as one object. This issue can be solved by placing the camera sensor overhead to capture the top view of the ongoing traffic or by using a velocity based tracker after the RP. The top view can separate all objects well and make region proposal easier. Our APBRP supports both cases mentioned above – side view application and top view application.

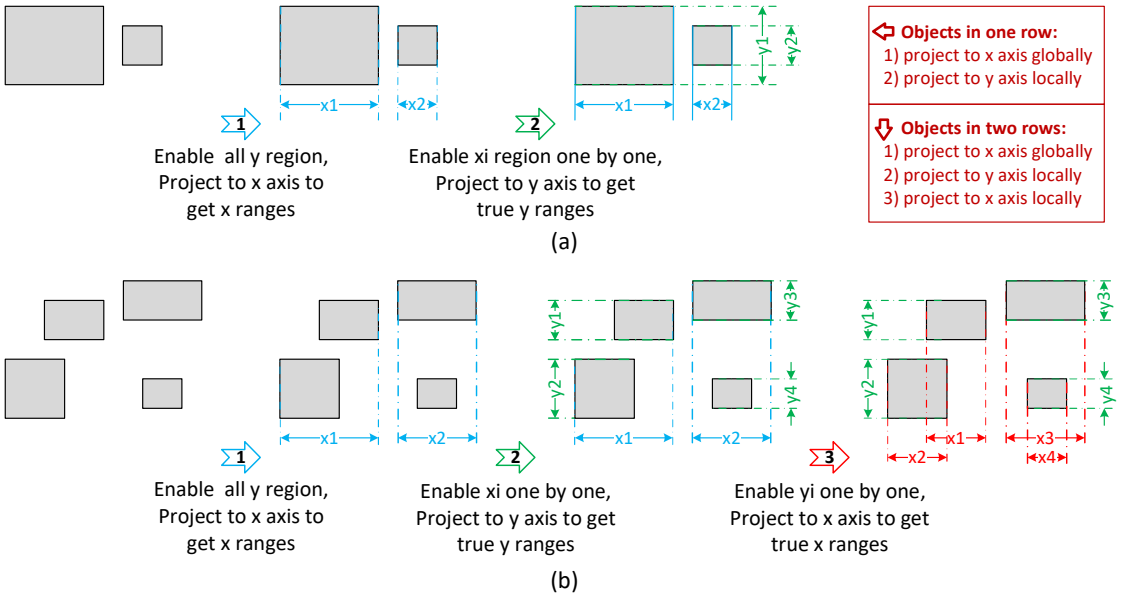


Figure 4.12: Demonstration of the proposed axes projection based region proposal (APBRP) for an image frame having multiple objects. (a) Side view application where multiple objects are in the same traffic flow and overlapped projection onto the vertical direction. To propose the exact region of interests (ROIs), the image is projected onto the x-axis globally first and then projected onto the y-axis locally. (b) Top view application where objects are in different traffic flow and overlapped projection may happen on both directions. To propose the exact ROIs, another local projection onto the x-axis is required.

Algorithm 3 describes the pseudocode of the execution flow of the proposed APBRP approach. All the symbols used are listed and explained in Table 4.2. $X_i = [x_{i\_start}, x_{i\_stop}], i \in [1, n_p]$, where n_p is the temporal number of objects detected by the p^{th} projection. The current n_p may be not the ground truth (GT) for the case of multiple traffic flows, but it will be calibrated by the next projection. In this algorithm, we use both global and local projections to find all objects with exact coordinate positions. The global projection onto the x-axis (y-axis) means projection of all rows (columns) onto the horizontal (vertical) direction. The local/partial projection onto the x-axis (y-axis) means that only specific rows (columns) that were previously selected are projected onto the horizontal (vertical) direction.

The algorithm begins with the global projection onto the x-axis and searches consecutive ranges where the histogram is larger than the threshold value. The number of objects n_1 and the coordinate positions along x-axis X_i are exact in the case of a side view application (Fig. 4.12(a)). Then, the image is locally projected onto the y-axis for each X_i region separately. By the same searching method,

Algorithm 3 APBRP: Axes Projection Based Region Proposal

Input: event based binary image (EBBI) with resolution of $W \times H$

Output: n: number of objects proposed in the image frame
rp_x: x coordinate positions of all proposed regions, the format is $rp\_x = [x_{1\_start}, x_{1\_stop}, ..., x_{n\_start}, x_{n\_stop}]$
rp_y: y coordinate positions of all proposed regions, the format is $rp\_y = [y_{1\_start}, y_{1\_stop}, ..., y_{n\_start}, y_{n\_stop}]$

/* $(x_{i\_start}, y_{i\_start})$ and (x_{i_stop}, y_{i_stop}) stands for the start and stop coordinate positions of the i^{th} object, respectively, where $i \in [1, n]$. */

Initialize $n \leftarrow 0,\ rp\_x \leftarrow 0,\ rp\_y \leftarrow 0,\ n_1 \leftarrow 0,\ n_2 \leftarrow 0,\ n_3 \leftarrow 0$
for $h \in [1, H]$ **do** // h is the enabled vertical range. global projection (onto x-axis)
  Calculate $H_X^w, w \in [1, W]$ // w is the enabled horizontal range
  if $(H_X^m > TH_X, m \in X_i$ // search consecutive ranges
   and $X_i > XSIZE\_MIN)$ **then** // otherwise filtering
    $n_1 \leftarrow n_1 + 1$ // n_1 is the number of objects detected by 1^{st} projection
    $X_i \leftarrow [x_{i\_start}, x_{i\_stop}], i \in [1, n_1]$ // X_i is the x range of i^{th} object
  else // no object detected
    break // end of RP
  end
end
for $i = 1 : n_1$ **do** // for each object detected by H_X
  for $w \in X_i$ **do** // w is the enabled horizontal range. local projection (onto y-axis)
    Calculate $H_Y^h, h \in [1, H]$ // calculate H_Y of regions where $w \in X_i$
    if $(H_Y^k > TH_Y, k \in Y_j$ // search consecutive ranges
     and $Y_j > YSIZE\_MIN)$ **then** // otherwise filtering
      $n_2 \leftarrow n_2 + 1$ // n_2 is the number of objects detected by 2^{nd} projection
      $Y_j \leftarrow [y_{j\_start}, y_{j\_stop}], j \in [1, n_2]$ // Y_j is the y range of j^{th} object
    end
  end
end
$n \leftarrow n_2$ // update the number of objects
case *side view application* **do** // single traffic flow (refer to Fig. 4.12(a))
  break // end of RP
end
case *top view application* **do** // multiple traffic flows (refer to Fig. 4.12(b))
  goto next step // need local projection once more
end
for $i = 1 : n_1$ **do** // for each object detected by H_X
  for $j = 1 : n_2$ **do** // for each object detected by H_Y
    Calculate $H_X^m, m \in X_i$ // calculate H_X of regions where $h \in Y_j$
    for $h \in Y_j$ **do** // h is the enabled vertical range. local projection (onto x-axis)
      if $(H_X^m > TH_X, m \in X_l \cap X_i$ // local search consecutive ranges ***belong to*** $\boldsymbol{X_i}$
       and $X_l > XSIZE\_MIN)$ **then** // otherwise filtering
        $n_3 \leftarrow n_3 + 1$ // n_3 is the number of objects detected by 3^{rd} projection
        $X_l \leftarrow [x_{l\_start}, x_{l\_stop}], l \in [1, n_3]$ // X_l is the x range of l^{th} object
      end
    end
  end
end
$n \leftarrow n_3$ // update the number of objects
break // end of RP

Table 4.2: Symbols used in the APBRP algorithm

W	width of the image size ($W = 320$)
w	the enable range along horizontal direction
H	depth of the image size ($H = 240$)
h	the enable range along vertical direction
m	index along horizontal direction
k	index along vertical direction
H_X^m	X histogram for the specific m^{th} column ($m \in [1, W]$)
H_Y^k	Y histogram for the specific k^{th} row ($k \in [1, H]$)
TH_X	threshold of X histogram
TH_Y	threshold of Y histogram
$XSIZE\_MIN$	the minimum X size of valid object (programmable parameter)
$YSIZE\_MIN$	the minimum Y size of valid object (programmable parameter)
i (or l)	index of the object detected by H_X
j	index of the object detected by H_Y
X_i (or X_l)	the x range of i^{th} (or l^{th}) object detected by projection onto x-axis
Y_j	the y range of j^{th} object detected by projection onto y-axis
$n_1,\ n_2,\ n_3$	the number of objects detected by the 1^{st}, 2^{nd}, and 3^{rd} projection
n	the number of objects detected (finally)

the number of objects n_2 and the coordinate positions along y-axis Y_j are exactly acquired. That is all for the side view case and $n_1 = n_2$ in this case. However, the number of objects n_1 and the coordinate positions along the x-axis X_i may be wrong in the case of top view application (Fig. 4.12(b)). In this case it is necessary to do another local projection onto the x-axis for each Y_j region separately to find the true value of total number of objects n_3 and the coordinate positions along the x-axis X_l, as the step 3 shown in Fig. 4.12(b). For example, by enabling rows in $y1$, projecting the image onto the x-axis, and then searching x coordinate in the region of previous $x1$ (blue labeled), the true x coordinate position of this object can be proposed as the new $x1$ (red labeled).

4.3.2 APBRP Hardware Architecture

The proposed APBRP is similar to HIST-RP but much simpler and easier for hardware implementation by the CRAM structure thanks to the parallel in-memory computing. Fig. 4.13(a) shows the CRAM bit cell circuit working in projection mode for in-memory computing base region proposal. The diffusion enable signal DE is deactivated to cut off the charge diffusion path. The switch is enabled to latch the data while the world line WL is kept low to disable the memory access.

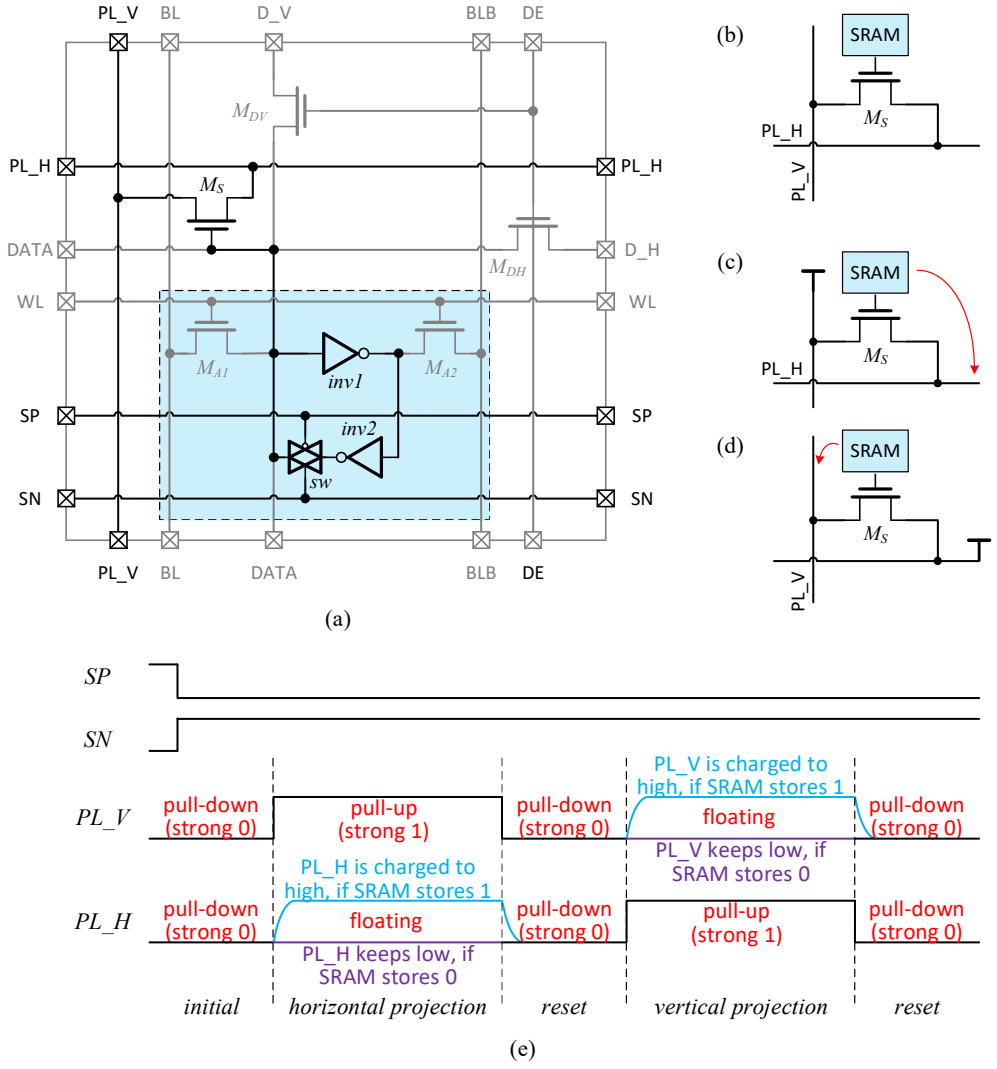


Figure 4.13: (a) CRAM bit cell working in projection mode for in-memory computing based region proposal. The SRAM as shown in the light blue colour works in retention mode to hold the stored value. MOSFET M_S acts as the projection device for 2-dimension separately, by which the value stored in SRAM can be readout along rows and columns. (b) The equivalent CRAM bit cell circuit working in projection mode can be modelled as a MOS transistor gate-controlled by an SRAM. (c) Project the bit cell value to the horizontal projection line PL_H by pulling up the vertical projection line PL_V to V_{DD}. (d) Project the bit cell value to the vertical projection line PL_V by pulling up the horizontal projection line PL_H to V_{DD}. (e) The conceptual timing diagram of projection onto horizontal and vertical directions.

The CRAM cell in projection mode can be simplified as shown in Fig. 4.13(b). The horizontal projection line PL_H and vertical projection line PL_V are connected to the source and drain of the projection device M_S respectively, while the gate of the projection device M_S is controlled by a SRAM. The data stored in the SRAM

determines whether one of the projection lines can pass its value to the other. Fig. 4.13(c) and (d) illustrate how the stored data are sensed and projected to the two orthogonal directions (x-axis and y-axis). To project the data to one projection line (PL), both the orthogonal PLs are pulled down to GND initially and then the PL is floating while the other PL is pulled up to V_{DD}. If the data stored in the cell is "1", the PL will be charged and can be sensed out. If the data in the cell is "0", the PL will keep floating. The conceptual timing diagram is shown in Fig. 4.13(e).

In previous modes, the projection lines PL_V and PL_H of CRAM are pulled down to GND so the NMOS transistor M_S works as a capacitor. This is the initial state of the projection mode and then the projection lines are released as floating. All the projection lines will be maintained at "0" no matter what the data stored in the cells because both the source and drain of M_S are reset to "0" initially. Next, we vertically project the image onto the x-coordinate by pulling up all horizontal projection lines PL_H, then sense the vertical projection lines PL_V to find the x-coordinates of the positions of each object, $X_i,\ i \in [1, n_1]$ (n_1 is the temporally total number of objects in this image frame). After acquiring the x-coordinates, all PLs are pulled down again for the next projection. In the next step, we successively pull up the specific vertical projection lines that belong to the range of X_i to find the y-coordinates of the positions of each object, $Y_j,\ j \in [1, n_2]$ (n_2 is the current detected number of objects). It is possible that the detected x-coordinates and the number of objects may not be correct if there are multiple objects in the image and their projections overlap. However, the coordinates and the number of objects will be calibrated by local projection once more, as illustrated in Fig. 4.12.

For each row and column, a projection detector (PD) is used to configure and detect the projection line. The PD block is composed of a pull up network (PUN), a pull down network (PDN), and a sense amplifier (SA). Before projection, the projection lines need to either be reset to "0" by enabling the PDN, or be precharged to V_{DD} by enabling the PUN. Fig. 4.14 shows a 320×240 CRAM macro in projection mode together with the PDs of the i^{th} row to detect the horizontal projection line $PL\_H\langle i \rangle$ and of the j^{th} column to detect the vertical projection line $PL\_V\langle j \rangle$. All the control signals for PUN and PDN are generated from the digital controller block. The reference voltage V_{ref} is an analog voltage programmable by a 4-bit DAC. This voltage is used to compare the projection line voltage and generate the 1-bit detection result. The bottom right corner of Fig. 4.14 lists the three different

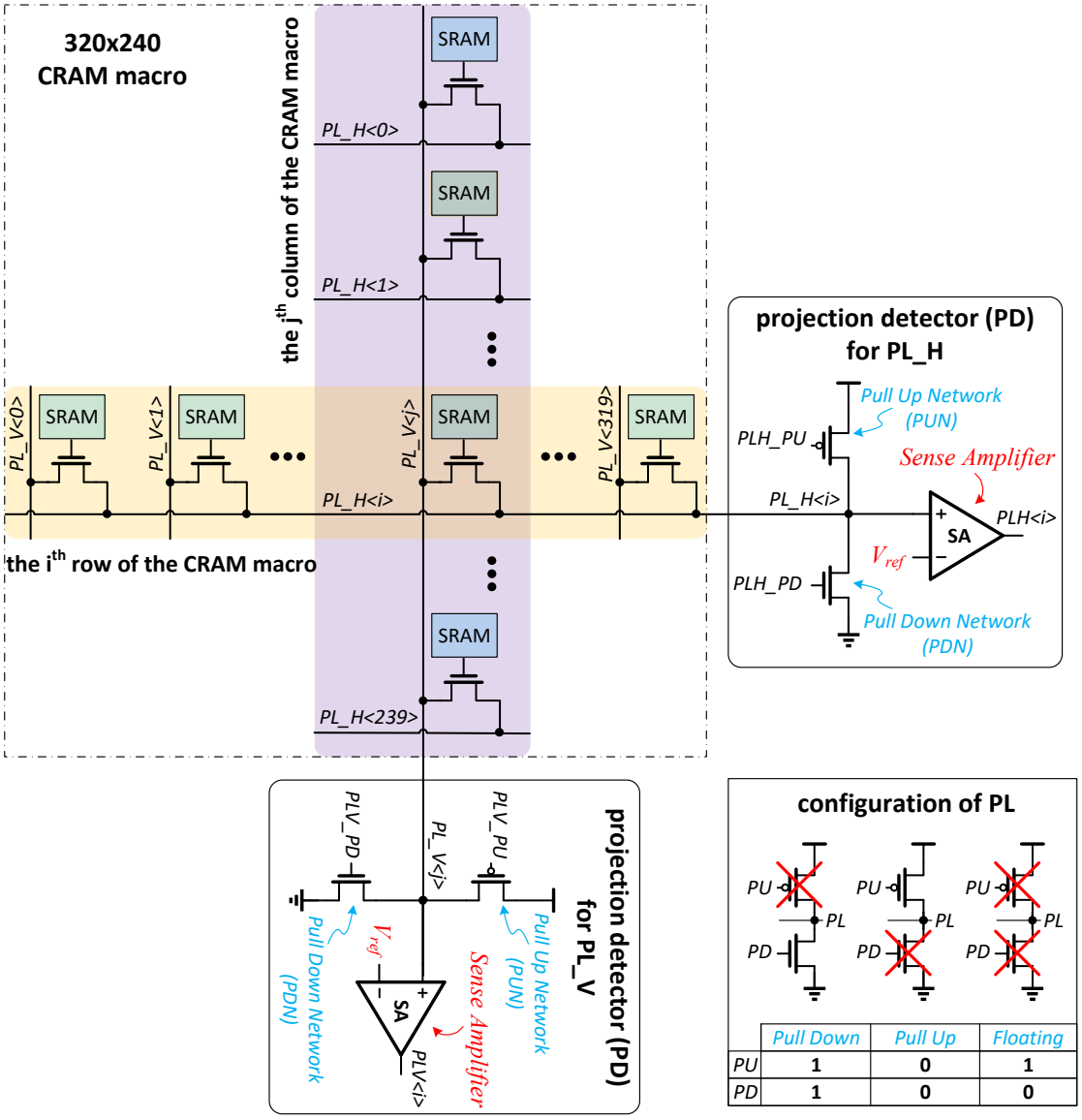


Figure 4.14: Horizontal and vertical 1-D projection detection for the x- and y-coordinates, respectively. The projection detector (PD) includes a pull up network (PUN), a pull down network (PDN), and a sense amplifier (SA). For clarity, only the PDs of the i^{th} row and the j^{th} column are shown.

configurations of projection line (PL) and the corresponding truth table of gate control signals.

A basic inverter can act as a simple SA but it takes longer time to wait the projection line charged over the trip voltage and consumes higher power consumption. In our work, a differential sense amplifier is used to overcome the above drawbacks. A small 4×4 array is illustrated in Fig. 4.15 to demonstrate how the proposed APBRP works. Before projection, both the horizontal and vertical lines are pulled

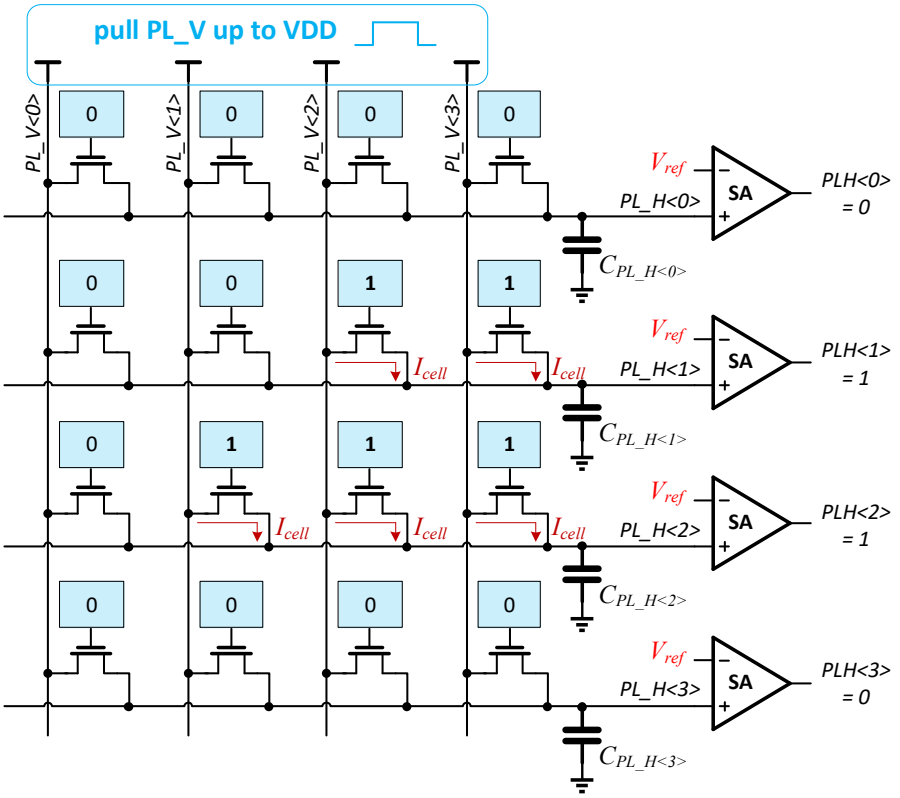


Figure 4.15: The horizontal projection and detection of a small 4×4 array. The detection result of horizontal projection line (PLH) is "1" if there is a "1" bit stored in any cell of this row.

down to GND. During projection mode, the vertical lines are pulled up to V_{DD} while horizontal lines are released as floating. For the projection NMOS device, the drain is connected to V_{DD} and the source is weak zero (floating). If the cell stores a "1", there is a current I_{cell} through NMOS to charge its source node, however, if the cell stores a "0", the source node keeps weak zero. As shown in the figure, the second and third rows store "1" data in some of the cells, so the corresponding parasitic capacitors are charged. When the line voltage is charged over the reference voltage V_{ref}, this line is detected by the SA and hence the coordinate positions can be proposed. In this figure, we only show and explain the axis projection and region proposal of the horizontal direction, but the implementation is identical for the vertical projection.

An equivalent circuit model of projection detector (PD) for a horizontal projection line (PL_H) is illustrated in Fig. 4.16 to explain the principle of projection detection. The process of detection is actually the charging process of the parasitic capacitor of the projection line. The pulled-up vertical projection lines and the cells across

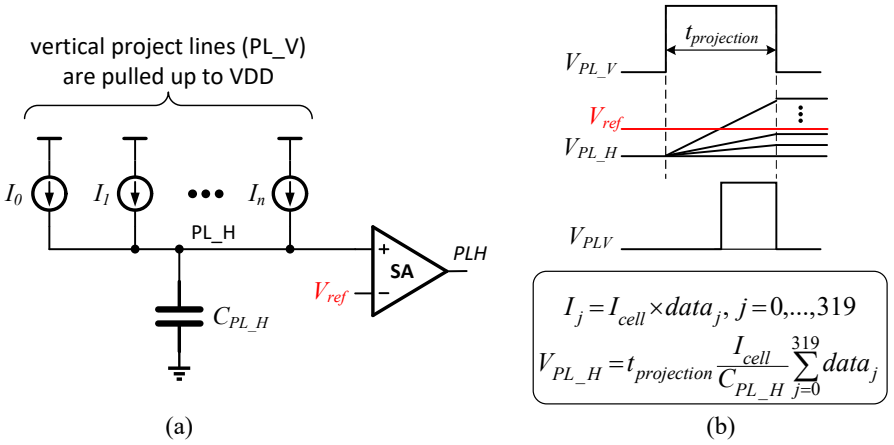


Figure 4.16: The principle of projection detection. (a) The data in a row are projected to the horizontal projection line (PL_H) by the current accumulation. (b) The voltage of horizontal projection line V_{PL_H} is proportional to the total current and then compared with a reference voltage V_{ref} to sense the projection value and find the region coordinates.

the PL_H are modeled as current sources (though the currents flowing through them are not constant). The i^{th} horizontal projection line current contributed by the j^{th} vertical projection line $PL\_V\langle j\rangle$ is given as

$$I_{i,j} = I_{cell} \times data_{i,j} \tag{4.1}$$

where I_{cell} is the cell current when "1" is stored, and $data_{i,j}$ is the data stored in the cell of i^{th} row and j^{th} column. The amplitude of the cell current I_{cell} depends on the total number of "1"s stored in this column. All the currents charge the parasitic capacitor C_{PL_H}, so the voltage of the horizontal projection line (PL_H) is expressed as

$$V_{PL\_H<i>} = t_{projection} \frac{I_{cell}}{C_{PL\_H}} \sum_{j} data_{i,j} \tag{4.2}$$

where j is the range of projection on the x-axis, $t_{projection}$ is the projection time or charging time. It depends on the amplitude of reference voltage V_{ref} – a higher V_{ref} requires a longer projection time $t_{projection}$. It is to be noted that noise in the image can be ignored by setting V_{ref} and $t_{projection}$ appropriately. Hence, the proposed APBRP can also propose accurate regions by processing the raw image directly.

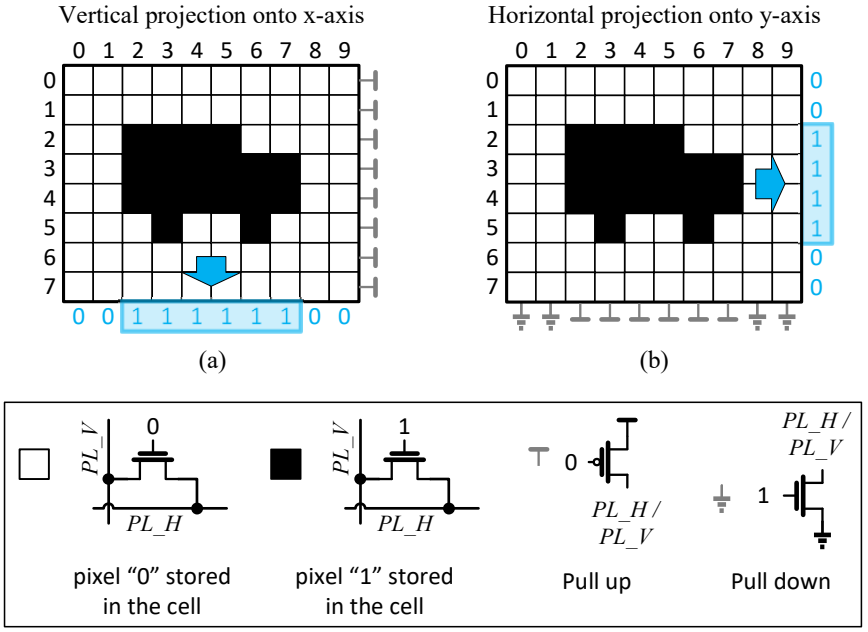


Figure 4.17: Demonstration of axes projection based region proposal (APBRP) for a single object. (a) Projection of the object onto the x-axis by pull-up all rows, i.e horizontal projection lines (PL_H), so that the object position along the x-axis is acquired ($x_{start} = 2$ and $x_{stop} = 7$). (b) Projection of the object onto the y-axis by pull-up the detected columns, i.e vertical projection lines (PL_V), so that the position along the y-axis is acquired ($y_{start} = 2$ and $y_{stop} = 5$). The legends of the cartoon are listed at the bottom.

4.3.3 Demonstration

In this section, we will show some demonstrations of the proposed axes projection based region proposal (APBRP) and explain how to configure the hardware implementation.

An illustration of APBRP for binary image with a single object is shown in Fig. 4.17. The blank dot represents a zero pixel where "0" is stored in the bit cell, so it has no effect on the projection lines PL_H or PL_V. The black dot represents a non-zero pixel where "1" is stored in the bit cell, so the data can be projected to the PL_H or PL_V by pulling up the other one to VDD. The power source symbol represents the line of this row (or column) is pulled up to V_{DD} by a "ON" PMOS while the ground symbol represents the line of this row (or column) is pulled down to GND by a "ON" NMOS. Applying the region proposal for a single object in the image is simple. To propose the region of object, we only need to first project the image vertically onto the horizontal axis to find the x-coordinates of

its boundaries and then project the object horizontally onto the vertical axis to find the y-coordinates of its boundaries. Fig. 4.17(a) depicts the configuration and projection results of vertical projection. Keep in mind that both the projection lines PL_V and PL_H of all rows and columns are initially connected to GND by a pull down network (PDN), i.e. the NMOS transistors controlled by PLH_PD and PLV_PD shown in Fig. 4.14. In the projection mode, we first release the PDN of all vertical projection lines PL_V to keep them floating and project the object onto the vertical projection lines by pulling up all horizontal projection lines PL_H. The object position along the x-axis is acquired (the start and stop positions of the object on the x-axis are $x_{start} = 2$ and $x_{stop} = 7$ respectively), as shown in Fig. 4.17(a). To get the position along the y-axis, we pull up the vertical projection lines belonging to the detected object while pulling down the rest of the projection lines, i.e. $PL\_V\langle i\rangle = V_{DD},\ i \in [2, 7];\ PL\_V\langle i\rangle = 0,\ i \notin [2, 7]$. As shown in Fig. 4.17(b), the object position along the y-coordinate is acquired (the start and stop positions of the object on the y-axis are $y_{start} = 2$ and $y_{stop} = 5$). So, for a single object, the region of interest (ROI) can be detected by projecting the image vertically and horizontally once each. The bounding-box of the ROI in this case is proposed as $(x_{start}, x_{stop}, y_{start}, y_{stop}) = (2, 7, 2, 5)$.

4.4 Comparison with CCL Approach

In this section, we will compare the performances of the proposed near-memory computing-based edge event-driven region proposal (EEDRP) approach and of the in-memory computing-based axis projection based region proposal (APBRP) approach with that of the conventional connecting component labeling (CCL) approach [97], [98], in terms of memory requirement, execution time, and energy consumption. We define first a list of symbols that will be used throughout this section.

W	Width of the binary image
H	Height of the binary image
$\overline{W_{obj}}$	Average object size along the horizontal direction
$\overline{H_{obj}}$	Average object size along the vertical direction
N	Maximum number of objects in the image frame
M_{EEDRP}	Memory requirement of the proposed EEDRP approach
M_{APBRP}	Memory requirement of the proposed APBRP approach
M_{CCLRP}	Memory requirement of the CCL RP algorithm
CC_{EEDRP}	Number of clock cycles for RP following the EEDRP approach
CC_{APBRP}	Number of clock cycles for RP following the APBRP approach
CC_{CCLRP}	Number of clock cycles for RP following the CCL algorithm
E_{EEDRP}	Total energy dissipation of the proposed EEDRP approach
E_{APBRP}	Total energy dissipation of the proposed APBRP approach
E_{CCLRP}	Total energy dissipation of the CCL RP algorithm
EPC_{EEDRP}	Energy per clock cycle of the proposed EEDRP approach
EPC_{APBRP}	Energy per clock cycle of the proposed APBRP approach
EPC_{CCLRP}	Energy per clock cycle of the CCL controller

4.4.1 Memory Requirement

Memory requirement is of great importance for hardware system processing images or video streams. This subsection calculates the amount of memory required in order to label a binary image frame with size of H rows and W columns. The near-memory computing based EEDRP approach uses serial operation by reading the memory array once. So it needs less memory requirement for storing but takes longer execution time. To detect the edge event, one register is required to delay (shift) the readout data. Since the start and stop coordinate positions of each object need to be stored, it occupies $2(N+1)\lceil \log_2 W \rceil$ bits and $2(N+1)\lceil \log_2 H \rceil$ bits of memories for the x- and y-coordinates, respectively, where $\lceil \cdot \rceil$ is the ceiling operator, which rounds a number to its nearest upper integer. The equation has $N+1$ instead of N because the EEDRP temporarily regards any rising edge as a new object and records its coordinates. Besides, the number of objects and the iteration index used for updating require $\lceil \log_2 N \rceil$ bits memory for each. Hence,

the total memory required for the EEDRP approach is estimated as:

$$M_{EEDRP} \approx WH + 2(N+1)\lceil \log_2 W \rceil + 2(N+1)\lceil \log_2 H \rceil + 2\lceil \log_2 N \rceil \tag{4.3}$$

The in-memory computing based APBRP approach uses parallel operation by projecting the image globally or locally onto the orthogonal directions. It requires a temporary memory with size of $Max\{2N\lceil \log_2 W \rceil,\ 2N\lceil \log_2 H \rceil\}$ for storing the local projection where $Max\{\cdot\}$ calculates the maximum values of its arguments. The total memory required for the APBRP approach in a side view application is estimated as:

$$\begin{aligned} M_{APBRP} \approx & WH + 2N\lceil \log_2 W \rceil + 2N\lceil \log_2 H \rceil + 2\lceil \log_2 N \rceil + \\ & Max\{2N\lceil \log_2 W \rceil,\ 2N\lceil \log_2 H \rceil\} \end{aligned} \tag{4.4}$$

The conventional CCL algorithm labels all pixels (both object pixels and background pixels) in the image frame. It needs $\lceil \log_2(N+1) \rceil$ bits to store the label of each pixel, where the +1 comes from the fact that 0 is a preoccupied label [99]. However, the recently modified CCL (single pass) algorithm scans the image in a raster scan for calculating the coordinates of the objects, which requires to keep the labels for two consecutive rows (present row and previous row). The amount of operating memory for label merging and data handling is small enough and is ignored. Therefore, the memory requirements for the CCL algorithm can be expressed as:

$$M_{CCLRP} \approx WH + 2N\lceil \log_2 W \rceil + 2N\lceil \log_2 H \rceil + 2W\lceil \log_2(N+1) \rceil \tag{4.5}$$

The WH number of memory cells required to store the binary image constitutes the main memory block and in our design this role is performed by the dedicated CRAM array. That is the dominant term of each approach. Other memories for x-coordinates, y-coordinates, number of objects, and temporary memory are implemented via register files. It can be seen from Equations 4.3, 4.4, and 4.5 that the CCL algorithm occupies the most number of register files while the proposed EEDRP approach occupies the least registers, considering the fact that $W(H) >> N$. In our design, for $W = 320$, $H = 240$, and $N = 15$, the required memory resources of the proposed EEDRP and APBRP are $\sim 5.6\times$ and $\sim 3.9\times$ less

than that of the conventional CCL algorithm, respectively, excepting the memory array for image storage.

4.4.2 Execution Time

The execution time T_{exe} is proportional to the clock cycle period T_{cycle} and the number of clock cycles CC for a specific task. It can be calculated as:

$$T_{exe} = T_{cycle}CC = \frac{CC}{f_{cycle}} \tag{4.6}$$

For a giving operation frequency f_{cycle}, the execution time depends only on the total number of clock cycles CC. In this subsection, we discuss the number of clock cycles required per frame to perform region proposal task. The serial processing of the near-memory based EEDRP approach takes WH clock cycles to scan the whole memory array. Theoretically, after detecting the last edge event, it still needs several cycles to compare and update if necessary. But in reality, we find no objects located at the right corner of the last row. So the clock cycle count of $CC_{EEDRP} = WH$ is enough for the EEDRP approach to propose all region of interests (ROIs) in the frame.

The in-memory computing based APBRP approach takes less clock cycles to finish the region proposal thanks to its parallel operation. For side view applications where there are no overlapping projections along the x-axis, it takes $8N$ clock cycles to search and record x- and y-coordinates. Besides, ech of the precharging, projection, sensing, and edge detection steps requires 2 clock cycles for two-direction calculations. Subsequently, the total clock cycle count is $CC_{APBRP} = 8N + 8$ for the APBRP approach to perform region proposal.

The conventional CCL-based region proposal [44] requires $CC_{CCLRP} \approx 2WH + 6N\overline{W_{obj}H_{obj}}$ clock cycles to calculate the ROIs, where $\overline{W_{obj}}$ and $\overline{H_{obj}}$ represent the average object size along the horizontal and vertical direction, respectively. The first term of CC_{CCLRP}, i.e. $2WH$ clock cycles, denotes the calculation involves reading and checking the value of each pixel. If the pixel belongs to an object, the state machine goes ahead to read its forward 4-neighborhood labels (left, top-left, top, and top-right), updates the bounding box of the object and the label of the

pixel. This is approximately represented by the second term. On the other hand, if the pixel belongs to the background, the CCLRP approach reads the next pixel.

In our application, $W = 320$, $H = 240$, $N = 15$, $\overline{W_{obj}} \approx 0.1W$ and $\overline{H_{obj}} \approx 0.1H$, the proposed EEDRP and APBRP approaches take 76800 and 126 clock cycles, respectively, for region proposal. Whereas, the clock cycle count for CCLRP is around 222720. Therefore, the proposed EEDRP and APBRP approaches are $\sim 2.9\times$ and $\sim 1767\times$ faster than the CCLRP implementation, respectively.

4.4.3 Energy Dissipation

The total energy dissipation to propose region of interests (ROIs) includes the energy consumed in memory for data access or computation and the energy consumed by the control logic. In modern technology, the energy consumption associated with the memory is much larger than that of the controller, especially for large-scale memory arrays. For this reason, in the following part we only consider the main part of energy dissipation associated with memory operations.

The proposed near-memory EEDRP approach proposes ROIs by reading the memory once, so the energy is mainly consumed for the memory read. To read a pixel, we need to precharge bit lines. The total capacitance of each bit line is HC_{unit}, where C_{unit} represents the unit (per CRAM cell) capacitance associated with bitline. To read all pixels in the array, the energy consumed on the bitlines is $WH \times HC_{unit}V_{DD}^2$, where V_{DD} is the system operating supply voltage. Because of the local buffer used in our design, the word line only drives WN_{bank} cells (in a mini array) for each read operation. Besides, all data in the same row and same mini array are captured at the same time by the efficient read strategy. Hence, in order to read all pixels in the array, the energy consumed on the word lines is $H \times WC_{unit}V_{DD}^2$. Consequently, the overall energy consumption for memory access can be expressed as

$$E_{EEDRP\_mem} = (H+1)WHC_{unit}V_{DD}^2 \tag{4.7}$$

Table 4.3 summarize the number of lines required to be charged or discharged while proposing regions for the APBRP approach. For a large size array, each line has a significant parasitic capacitor and it consumes a dominant part of energy to charge or discharge the line for dedicated projection. For the first projection onto the

Table 4.3: Number of different lines required to pull down (PD) and pull up (PU) for projection and the number of lines to be charged (CH) while projection (N_{obj} is the number of objects in the image frame with dimension of $W \times H$, and $\overline{W_{obj}}$ and $\overline{H_{obj}}$ denote the average object size along the horizontal and vertical direction).

projection mode	**PD**	**PU**	**CH**
1^{st} projection (onto x-axis, globally)	0	H	$\leq N_{obj}\overline{W_{obj}}$
2^{nd} projection (onto y-axis, locally)	$H + (N_{obj}-1)\overline{W_{obj}}$ + $(N_{obj}-1)(\overline{H_{obj}}+\overline{W_{obj}})$	0 + $(N_{obj}-1)\overline{W_{obj}}$	$N_{obj}\overline{H_{obj}}$
3^{rd} projection (onto x-axis, locally)	$\overline{W_{obj}}$ + $(N_{obj}-1)(\overline{W_{obj}}+\overline{H_{obj}})$	0 + $(N_{obj}-1)\overline{H_{obj}}$	$N_{obj}\overline{W_{obj}}$

x-axis, no line need to pull down (PD) because all lines are initially connected to GND, while all the vertical lines need to pull up (PU) for the global projection. During projection, the number of vertical lines to be charged is $N_{obj}\overline{W_{obj}}$ if all projections along the x-axis are non-overlapped. If some projections are overlapped (top view case), the number will be less. For the second projection onto the y-axis, we should pull down all the vertical lines (i.e. H) and all objects detected along the x-axis except the first object (i.e. $(N_{obj}-1)\overline{W_{obj}}$), so we can detect the y-coordinates of this object. No extra pull up along the x-axis is required because we keep the region of first object as high. For this local projection, only the y-coordinate of one object is detected, so the number of horizontal lines to be charged is $\overline{H_{obj}}$. For each of the next $(N-1)$ objects detected from the x-axis, we need to pull down the previous $\overline{H_{obj}}$ horizontal lines and $\overline{W_{obj}}$ vertical lines, while pulling up the current $\overline{W_{obj}}$ vertical lines. Each time, the number of horizontal lines to be charged is $\overline{H_{obj}}$. It is to be noted that in the side view case, the first two projection are enough to capture the exact bounding box for all objects. For the top view application, a third projection is necessary, as listed in the last row of Table 4.3. In our real testing, we deployed the camera at road junctions to detect vehicles. Subsequently, the total energy consumed by the memory of the in memory computing based APBRP can be expressed as

$$E_{APBRP\_mem} = \{[H + (2N-1)\overline{H_{obj}}]WC_{unit} + [(4N-3)\overline{W_{obj}}]HC_{unit}\}V_{DD}^2 \quad (4.8)$$

where C_{unit} is the unit (per CRAM cell) capacitance associated with the vertical and horizontal projection lines. V_{DD} is the system operating supply voltage. In this

equation, we consider the first two projections.

The CCLRP algorithm reads the binary pixel and assigns a label to that pixel. Since the controller needs to store the labels of two consecutive rows, it needs $2W\lceil\log_2(N+1)\rceil$ bit memory location, which can be implemented using register files. Hence, in the total energy calculation of the CCL controller, we consider only the SRAM read energy. Therefore, the total energy of the controller can be estimated as

$$E_{CCLRP\_mem} \approx WH(W+H)C_{unit}V_{DD}^2 \tag{4.9}$$

where C_{unit} is the unit capacitance associated with an SRAM cell bit line and word line. Therefore, to read a bit, the total capacitance needs to be charged is $(W+H)C_{unit}$.

The energy consumed by the controller can be expressed as $E_{cycle}CC$, where E_{cycle} denotes the energy per cycle and the CC presents the number of clock cycles to perform the region proposal as discussed in section 4.4.2. Hence, the total energy consumption for each region proposal approach is estimated as

$$E_{RP} = E_{mem} + E_{cycle}CC \tag{4.10}$$

For a fair comparison and simplicity, we assume the energy per cycle for each RP controller is equal and estimate it as 10 pJ/cycle, according to [100]. The post-simulation indicates the unit capacitance C_{unit}=2.2 fF. At $W = 320$, $H = 240$, $N = 15$, $\overline{W_{obj}} \approx 0.1W$ and $\overline{H_{obj}} \approx 0.1H$, the energy consumed of the conventional CCL region proposal is 2.34 µJ, while the proposed EEDRP and APBRP are 2.83× and 780× more energy efficient, respectively. At the same time, the in-memory computing based APBRP is ∼ 276× more energy efficient than the near-memory based EEDRP.

4.5 Results and Discussion

The proposed region proposal (RP) techniques were fabricated in a 65 nm low power (LP) CMOS process. Fig. 4.18(a) shows the layout of CRAM bit cell composed of 11 transistors (11T). The customized bit cell is designed elaborately with size of 1.82 µm × 2.18 µm, including the power strips. The die photo of the prototype

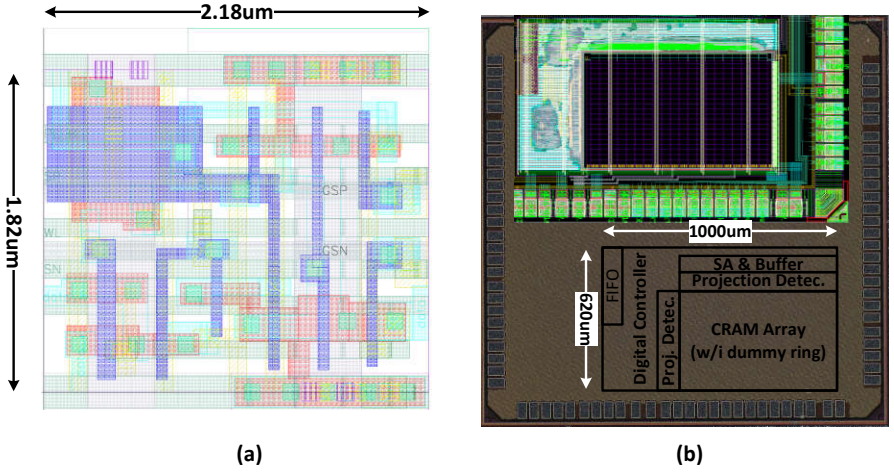


Figure 4.18: (a) Layout of the CRAM bit cell. (b) Chip micrograph with layout overlay of the fabricated ASIC. The total active area is 0.62 mm^2.

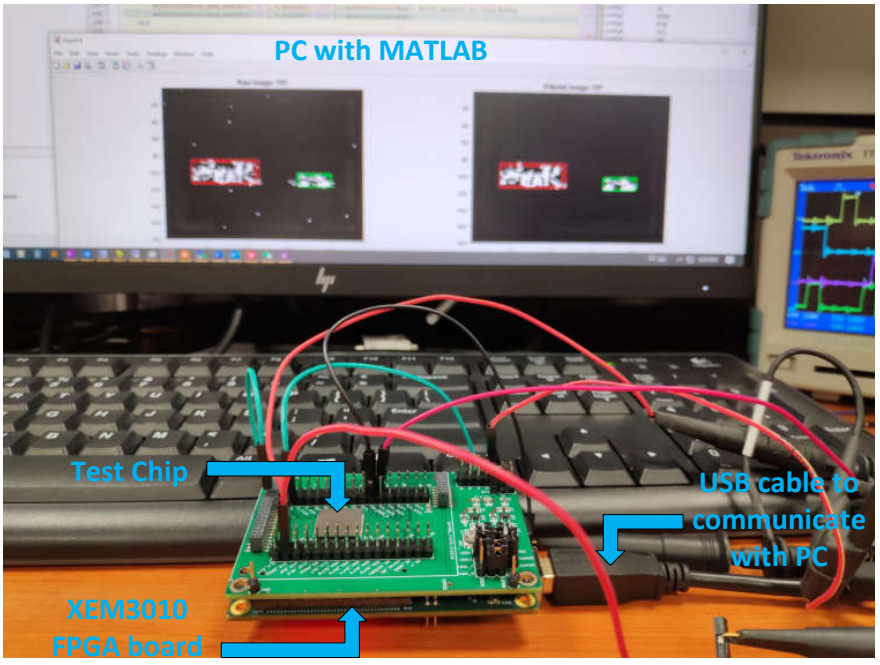


Figure 4.19: Testing setup for the chip. The testing board stacks with Opal Kelly XEM3010 FPGA for transferring data from and to the PC.

chip is shown in Fig. 4.18(b), covering an active area of 0.62 mm^2. The analog block occupies 820 µm × 500 µm area overhead, where the 84 Kb CRAM macro (including the dummy array ring) size is 800 µm × 450 µm. The 128 × 32 bit FIFO block occupies 300 µm × 70 µm and the rest of the area is occupied by a digital controller with utilization of 78%. The measurement setup is shown in Fig 4.19, where the testing board is hooked with a FPGA board. The FPGA works as an input/output (I/O) module for testing. The raw video from the traffic dataset

[39] with noise and holes are fed into the chip by a FPGA board, and the output results from the chip are gathered by the FPGA and then send to MATLAB for visualization and further data analysis. In the next part of this section, we will present results of characterization of the designed chip in detail and compare the performance of the proposed region proposal approaches with the state of the art techniques.

4.5.1 Testing Results

Fig. 4.20 demonstrates the effect of image denoising and region proposal of our design for a traffic surveillance data set captured by DAVIS240C camera. Fig. 4.20(a) shows a random frame of event based binary image (EBBI), with two objects (a car and a bike) in it. We also demonstrate the image denoising and filling operation with different diffusion levels by setting the equivalent resistors in the 2D RC network. In Fig. 4.20(b), a higher equivalent resistance (source-drain resistance R_{DS} of the diffusion transistors M_{DV} and M_{DH} shown in Fig. 4.7) is set by configuring $diffusion\_r\_sel\langle 3:0\rangle = 4'b0010$. It can be seen that most noise pixels are filtered but still some residues remain in the places where the initial noise density was higher. A lower equivalent resistance is set by configuring $diffusion\_r\_sel\langle 3:0\rangle = 4'b0100$ in Fig. 4.20(c) to filter more noise and a further lower equivalent resistance is set by configuring $diffusion\_r\_sel\langle 3:0\rangle = 4'b0101$ in Fig. 4.20(d). The supply voltage is set to $V_{DD} = 1.2\ V$ and the diffusion time is set to $diffusion\_t\_sel\langle 1:0\rangle = 2'b10$ for all the three filtered images.

The bounding boxes according to the region proposal results are plotted in Fig. 4.20 as well. The solid line bounding box represents the edge event driven region proposal (EEDRP) while the dash dot line represents the axes projection based region proposal (APBRP). The bounding boxes are colored for different objects just for clarity. As we can see from the figure, both EEDRP and APBRP propose correct region of interests (ROI) with only slight difference of the coordinate positions. Both the proposed EEDRP and APBRP approaches work well even if there still is some noise in the image. We will evaluate the performance of the region proposal approaches with the weighted $F1 - score$ in the later part of this section.

We also manipulated different degrees of denoising and checked the robustness of the proposed region proposal approaches. Fig. 4.21(a) - (c) illustrates three different

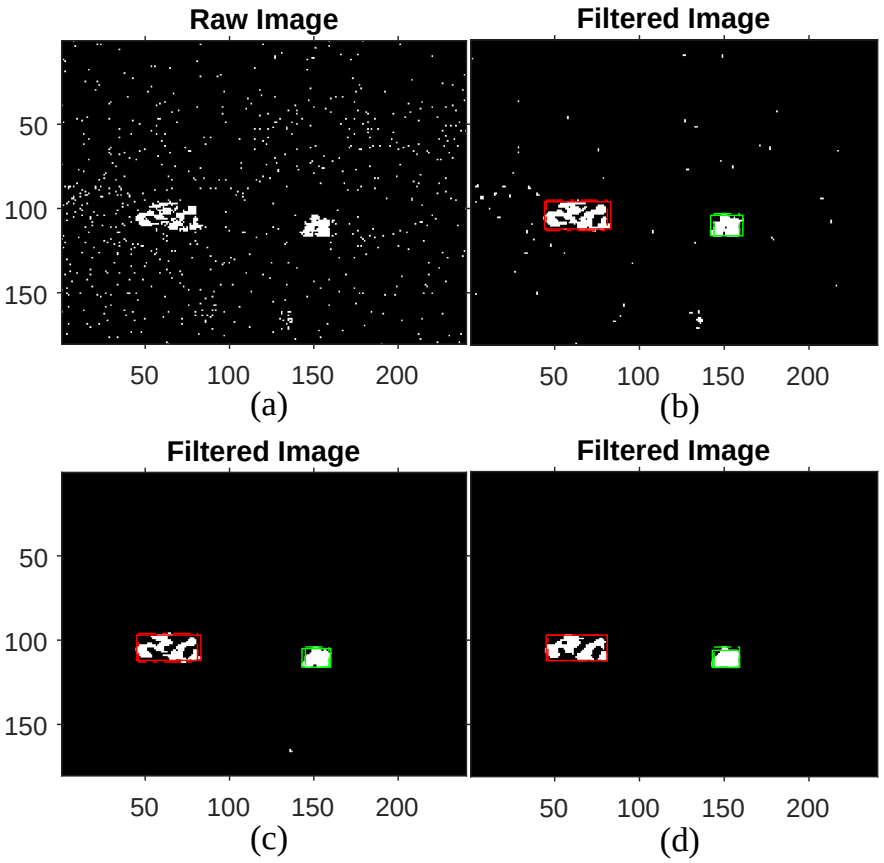


Figure 4.20: Results of denoising and region proposal for an image with two objects. The solid line bounding box represents the edge event driven region proposal (EEDRP) while the dash dot line represents the axes projection based region proposal (APBRP). The bounding boxes are colored for different objects. (a) Raw binary image captured by the DAVIS. (b) - (d) Different degrees of filtered image with different diffusion levels.

levels of denoising (of incresed strengthen gradually from left to right) for a image with two objects – a bus and a car. Region proposal results are correct for both (a) and (b) cases, while case (c) gives three objects because the stringent denoising fragments the car into two parts. Even so, the correct region proposal can still be detected by increasing the value of the parameter $SLOT\_X$ (see Algorithm 2) to merge the fragmented regions. In Fig. 4.21(d), the degree of denoising is the same as that of Fig. 4.21(c), but the region proposal result is correct because of the different parameters configuration. In Fig. 4.21(a) - (c), the RP parameters are set to $SLOT\_X = 4$, $SLOT\_Y = 4$, while in Fig. 4.21(d), $SLOT\_X = 12$, $SLOT\_Y = 4$.

Both the proposed near-memory computing based EEDRP and in-memory computing based APBRP can deal directly with a raw image with noise and holes

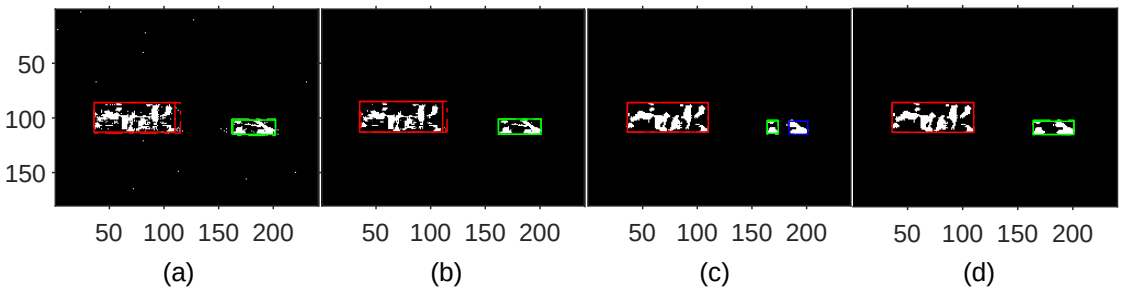


Figure 4.21: Results of denoising and region proposal for different diffusion levels. (a) Lower degree of denoising with residual noise pixels. (b) Better degree of denoising where noise pixels are filtered and the objects are unfragmented. (c) Higher degree of denoising where the object is fragmented. (d) The same degree of denoising as (c) but with modified RP parameters.

by setting the appropriate configurations. They are robust enough for different application scenarios from high noise density to low noise density. However, it must be highlighted that both the region proposal approaches consumes more energy to calculate the ROIs when noises present in the image. The EEDRP approach is driven by edge events – hence, the computation is triggered whenever noise is encountered because edges are detected of each noisy pixel. This increases the amount of unnecessary computations and, therefore, the relevant power dissipation. For APBRP, the noise pixel will charge its column or row line during projection, which also increases the unnecessary power dissipation. Another trade-off we would like to mention is that increasing the value of the RP parameters $SLOT\_X$ or $SLOT\_Y$ can merge the fragmented object, like the car shown in Fig. 4.21(d), but this will lead to an incorrect region proposal when two objects are very close. When the distances of multiple objects are less than the RP parameters $SLOT\_X$ or $SLOT\_Y$, the RP approaches will propose only one object with a big bounding box covering the multiple objects, as discussed in section 4.2.1.

4.5.2 Evaluation Metrics

To evaluate the effect of the proposed region proposal approaches, we compare the bounding boxes proposed by our region proposal (RP) approaches with the manually annotated ground truth (GT) bounding boxes and calculate the intersection-over-union, IoU.

$$IoU = \frac{A_{Intersection}}{A_{Union}} = \frac{A_{GT} \cap A_{RP}}{A_{GT} \cup A_{RP}} \tag{4.11}$$

where $A_{Intersection}$ is the area of intersection and A_{Union} is the area of union of GT and RP bounding boxes. A_{GT} and A_{RP} denote the area of GT and RP boxes, respectively. IoU is an effective metric to evaluate the detection accuracy of a region proposal method [101]. If the IoU of a proposed region is greater than a threshold (IoU_{th}), the region is assumed to be a true positive region (correct region detection), otherwise a false positive region (incorrect detection). Therefore, we sweep the threshold IoU_{th} in the range $0.1 - 0.9$ with a step of 0.1 to find out the precision and the recall averaged over the entire duration of the recording. Precision (P) is the ratio between the true positive regions and the total positive regions, and recall (R) is the ratio of the true positive regions to the total ground truth regions. Both of these metrics are commonly used to evaluate performance of trackers [102]. $F1 - score$ is the harmonic mean of the precision and the recall. Using the harmonic mean has the effect that a good $F1 - score$ requires both a good precision and a good recall. We calculate the $F1 - score$ and the weighted score as follows [102, 103]

$$P^i_{th} = \left.\frac{true\ positive\ regions}{total\ proposed\ regions}\right|_{IoU=IoU_{th}} \tag{4.12}$$

$$R^i_{th} = \left.\frac{true\ positive\ regions}{total\ ground\ truth\ regions}\right|_{IoU=IoU_{th}} \tag{4.13}$$

$$F1^i_{th} = 2\frac{P^i_{th} \times R^i_{th}}{P^i_{th} + R^i_{th}} \tag{4.14}$$

$$F1^{wgt}_{th} = \frac{\sum_{i=1}^{K} (N^i_{tracks} \times F1^i_{th})}{\sum_{i=1}^{K} N^i_{tracks}} \tag{4.15}$$

Here, P^i_{th} and R^i_{th} are precision and recall for the recording i at $IoU = IoU_{th}$. N^i_{tracks} represents the number of tracks in recording i. $F1^i_{th}$ and $F1^{wgt}_{th}$ are $F1 - score$ and the weighted score for all K recordings, $i = 1, ..., K$.

In our testing, $\sim$ 70k raw binary images [104] in K recordings ($K = 5$) were fed into the chip to evaluate the proposed EEDRP and APBRP approaches. Fig. 4.22 shows the weighted average of precision and recall across the K recordings, where the weights correspond to the number of ground truth (GT) tracks present in a given recording. EEDRP and APBRP approaches show similar and stable precision and recall values for varying thresholds.

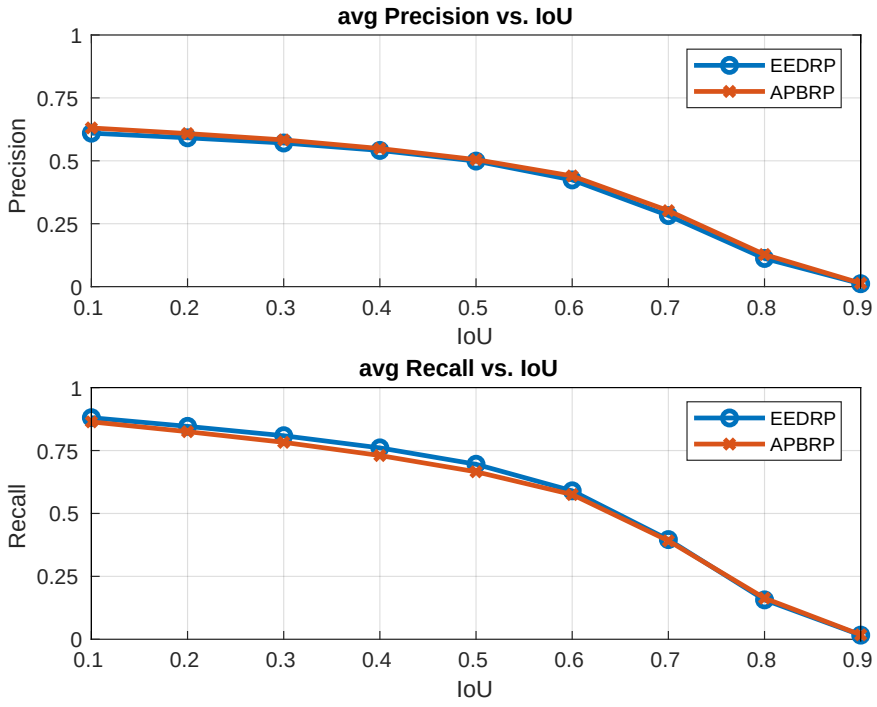


Figure 4.22: Average precision and recall for different IoU thresholds.

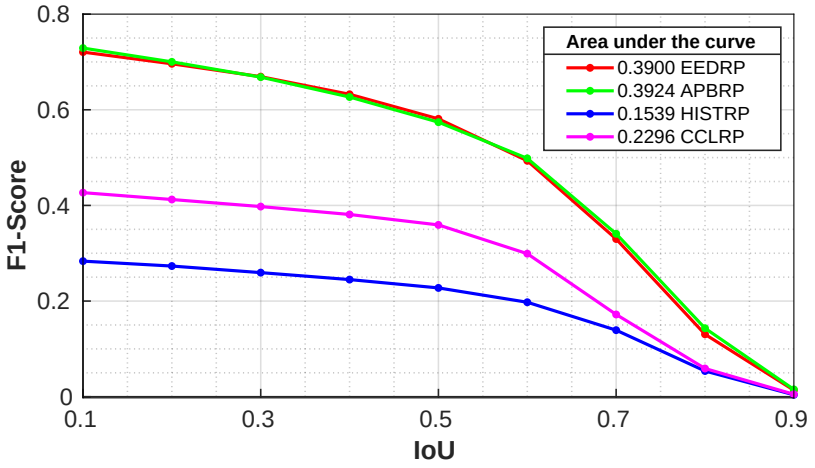


Figure 4.23: Comparison of weighted $F1-score$ for different IoU thresholds.

As a comparison, we also evaluated the conventional connected component labeling (CCL) algorithm and histogram based (HIST) algorithm with the same data recordings. To evaluate the performance of region proposal (RP) over a range of threshold, Fig. 4.23 shows the weighted $F1-score$ at various IoU_{th} values for the test recording. The performance is determined by looking at the area under the curve (AUC). AUC is a good metric to compare robustness and performance of all the methods. It can be seen that the proposed RP methods outperform

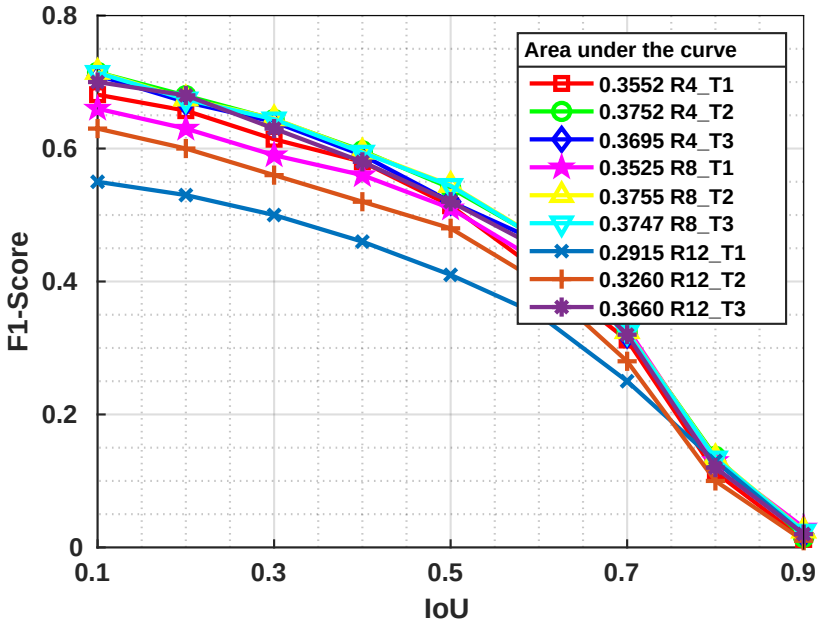


Figure 4.24: AUC with different resistance and diffusion time of the CRAM based APBRP.

the other previous RP methods for binary images, namely HISTRP and CCLRP. The values of the parameters for the proposed EEDRP and APBRP approaches are set as follows: $diffusion\_r\_sel = 8$, $diffusion\_n\_sel = 2$, $diffusion\_t\_sel = 1$, $XSIZE\_MIN = YSIZE\_MIN = 8$, $SLOT\_X = SLOT\_Y = 16, HIST\_X = HIST\_Y = 1$. The scale parameter and histogram threshold for the HISTRP [39] are set as 3 and 1 respectively.

Fig. 4.24 shows the weighted $F1 - score$ curves and AUC values with different settings of resistance and diffusion time. The high accuracy is robust across multiple settings of resistor and diffusion times. Higher resistance takes longer diffusion time to achieve better image denoising and region proposal performance. This shows the flexibility supported by the proposed new CRAM unit cell.

4.5.3 Power and Frequency

The proposed CRAM array performs denoising and region proposal tasks leveraging near- or in-memory computing techniques. It works in different operation modes for different tasks. We will present the energy consumed for normal memory access, charge diffusion in denoising mode, and region proposal, separately. While not stated otherwise, we assume that the following results are measured under the

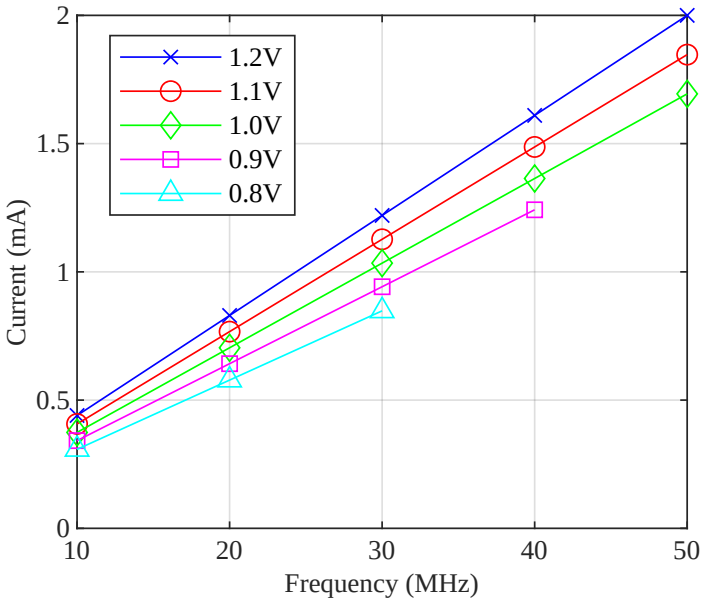


Figure 4.25: Measured current consumption of APBRP vs frequency at different supply voltages.

typical measurement conditions, where power supply voltage is 1.2 V, and operation frequency is 50 MHz.

It is to be noted that the power consumption varies with the content of the input raw images. The power dissipation for diffusion depends on the noise density and distribution while the power dissipation for region proposal approaches depend on the number of objects and the shape (size) of the objects. To evaluate the energy consumption, a typical event based binary image (EBBI), where the objects occupy $\sim 10\%$ of the image frame is fed into the hardware. The standby current is only 20 µA at 1.2 V supply voltage. Working in the EEDRP mode, the EBBI processor draws a peak current of 2.38 mA when edge event occurs and an average 1.9 mA read current otherwise because it only responses to the edge events and computes for region proposal when edge is detected. The measured average current of the APBRP mode is 2 mA at V_{DD}=1.2 V and f=50 MHz. Fig. 4.25 shows the average current consumption of APBRP across different operation frequencies as well as supply voltages. As shown in the Fig. 4.25, the prototype consumes lower current at lower supply voltage, but the operation time increases because the operation frequency is limited by the voltage V_{DD}.

The minimum power is measured under 0.8 V supply voltage and 30 MHz operation frequency. The prototype consumes 1.13 mA average current for diffusion and

0.85 mA current for region proposal at 0.8 V and 30 MHz. Considering that the operation time for diffusion and APBRP are 2 clock cycle and 16 clock cycles, respectively, it consumes 1.39 fJ and 8.33 fJ per pixel respectively for a single object region proposal within a 240 × 180 frame size.

4.5.4 Comparison

Table 4.4 compares the performance of the proposed EBBI processor with the prior works published recently. Both this work and [98] are for event based image processing, while this work and the rest of the references listed in the table leverage the parallel in-memory computing techniques. For the IMC-based image restoration, we define 5 operations per pixel – average of four neighbours plus threshold comparison. Since the CRAM array (320 × 240) is processed simultaneously in a clock cycle during diffusion mode, the throughput of image denosing is 19200 GOPS at 50 MHz. The edge event driven region proposal (EEDRP) is implemented by near-memory computing, reducing the energy consumption but the throughput depends heavily on the operation frequency. Image denoising achieves the highest throughput due to the globally parallel computing, while EEDRP achieves lowest throughput because of its serial operation. The axes projection region proposal (APBRP) achieves a competitive throughput because it takes only 16 clock cycles to perform region proposal for single object. It is to be noted that the throughput depends on the operation frequency, image size, and the level of parallelism. This work demonstrates IMC-based global parallel diffusion and column/row-wise projection to achieve a maximal energy efficiency of 1220 TOPS/W for image restoration and 915 TOPS/W when combined with region proposal at 0.8 V and 30 MHz.

The high energy efficiencies of denoising and APBRP are achieved by the in-memory computing based parallel operation. Although the near-memory computing based EEDRP approach is not competitive in terms of both throughput and energy efficiency compared with the in-memory computing techniques, this novel algorithm is still more efficient than the conventional CCL-based RP because of its unique feature of responding to edge events only.

Table 4.4: Performance Comparison of Prior Works

	This Work	SoVC'20 [98]	ISSCC'19 [58]	TCAS-I'19 [59]	JSSC'18 [60]	SoVC'19 [61]	JSSC'19 [62]	JSSC'20 [54]
Technology (nm)	65	10	55	65	65	28	65	65
Cell Type	11T	6T	Twin-8T	Split-6T	6T	Split-6T	10T	12T
Memory Capacity (kB)	10.4[a]	-	0.468	4	102	2	2	2
Chip Area (mm^2)	0.62	2.6	0.047	-	3.9	0.03	0.063	0.11
Task	Image denoising, Region proposal	Event generation, Region proposal	CNN	BNN	Versatile DNN	BNN	BWN	DNN
Operation Mode	Charge distribution, Current summation	Digital	Current summation	Current summation	Digital XNOR	Current summation	Charge distribution	Voltage divider
Supply Volatge (V)	$0.7-1.2$	$0.65-0.8$	1	1	$0.55-1$	1	$0.8-1$	$0.6-1$
Throughput (GOPS)	Denoising: 19200 Denoising+RP: 1018	-[b]	-	278.2	1380	615	8	5461
Energy Efficiency (TOPS/W)	Denoising: 1220 Denoising+RP: 915	-	$18.37-72.03$ $[13.15-51.5]$[c]	$30.49-55.8$	6.3	300 [55.6][c]	40.3	403

[a] Including dummy cell ring.
[b] Reported as 70fps@1Meps.
[c] Assuming energy efficiency for a 65 nm implementation according to the relation energy efficiency $\propto \frac{1}{(Tech.node)^2}$.

4.6 Conclusion

In this chapter, we proposed a near-memory based edge event driven region proposal (EEDRP) approach and an in-memory computing based axes projection based region proposal (APBRP) approach for binary image. Both are implemented based on the novel 11T CRAM structure and verified by a 320×240 CRAM array. Although the proposed EEDRP approach takes 76800 clock cycles to propose all region of interests due to its serial operation mode, it still achieves $\sim 2.9\times$ faster than the conventional CCL algorithm. Because it only responds to edge events, it occupies $\sim 5.6\times$ less memory resources and $\sim 2.83\times$ less energy for computing regions compared with the conventional CCL algorithm. However, the proposed APBRP approach only takes about tens to hundred clock cycles to perform region proposal and achieves $\sim 1767\times$ faster than CCL algorithm thanks to the parallel in-memory computing strategy. The in-memory computing based APBRP is $\sim 2700\times$ more energy efficient than the near-memory based EEDRP. The weighted $F1$ scores of both EEDRP and APBRP achieve $2.55\times$ and $1.7\times$ better than these of the conventional HISTRP and CCLRP, respectively.

We have discussed the in memory computing based image denoising and image filling to recover the raw event based binary image (EBBI) in Chapter 3. In this chapter, we presented both near-memory and in-memory computing based region proposal approaches to find the region of interests (ROIs) in the image. In the next chapter, we will explore the use of time-based computing for energy efficient implementation of neural networks to classify the content of the ROI.

Chapter 5

Time-based Computing with Neuron Oscillator

5.1 Introduction

The last two chapters presented in memory computing techniques for energy efficient neuromorphic applications. This chapter will explore another efficient technique – time based computing for creating neural networks that can process/classify the content of the image in the region of interests (ROIs) detected by the earlier circuits. While there has been a large number of recent works exploring the acceleration of the matrix multiplication in DNNs, relatively few works have explored efficient implementation of the neuronal nonlinearity. Moreover, an emerging area of spiking neural networks (SNN) is being explored where sparse firing activity of bio-inspired neurons lead to better energy efficiency than their conventional artificial neural network (ANN) counterparts. The time-based computing circuits introduced in this chapter can be used for both these cases.

As discussed in section 1.3.2, the last block in the pipeline of our work is a neural network (NN) for classification. The pipeline of object classification using dynamic vision sensor (DVS) is illustrated in Fig. 5.1. The events are captured as event-based binary image (EBBI), where energy- and time-efficient denoising and filling are implemented by in memory computing technique based on globally parallel charge diffusion. The preprocessed image is then used to propose regions in the frame. There are two region proposal (RP) approaches - one is the edge event

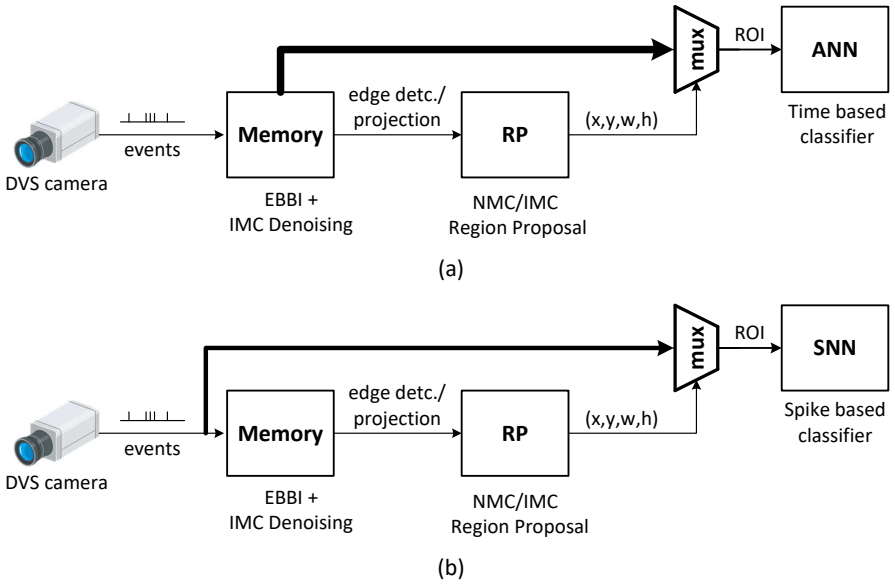


Figure 5.1: Pipeline of object classification for dynamic vision sensor based traffic surveillance with: (a) artificial neural network (ANN), and (b) spike neural network (SNN).

driven region proposal (EEDRP) using near-memory computing technique and another one is the axes projection-based region proposal (APBRP) using in-memory computing technique. According to the boundary box (x, y, w, h), only region of interests (ROIs) are fed into the NN block. For both ANN and SNN, there are two important parts – neuron and synaptic array. NN are gaining popularity recently in many applications, such as face recognition [105], speech recognition [106], natural language processing [107], etc. due to improved performance compared to other machine learning algorithms. Deploying these networks at the edge in the Internet of Things (IoT) is important for scalability and fast response by reducing data transmission to the cloud [108, 109]. Because of the extremely attractive performance, NNs have been explored in-depth at both algorithm and hardware level. A lot of work has focused on implementing vector-matrix multiplication for in-memory synaptic array with various devices, like SRAM [110–113], RRAM [114–116], and floating gate memory [117–119]. The vector-matrix multiplication is mostly accelerated, however, the neuron gains less focus. Hence, we focus on implementing neuron itself and designing a novel time domain low-energy neuron oscillator.

Neuromorphic implementations that utilize analog or physical computing have shown the benefits in terms of energy efficient compared to digital baselines when

the required precision is low [120, 121]. Neuro-inspired spiking neural networks (SNN) have also gained popularity due to the promise of sparse activation leading to lower energy dissipation [122]. Time domain operation is a good strategy to improve the energy efficiency due to the reduced power supply in scaled CMOS. In recent years, time-based computational circuits for DNN/SNN are gaining popularity [123, 124]. An important building block in these designs is a digital delay cell. For example, it is used to create an oscillator that can either convert an analogue current to digital output (rate-based neuron) in [125–127] or be used as an integrate-and-fire neuron [123] with bio-plausible refractory period and spike frequency adaptation features. The major requirements for an oscillator to function as a neuron are:

(1) Low area requirement since a large number of them are needed per chip.

(2) Low energy/cycle to reduce operational energy.

(3) Since these circuits typically co-exist with a large amount of digital circuits, they have to be robust against power supply noise and other interference.

Interestingly, for NN implementations, the linearity of the tuning curve of the oscillator-based neuron is not important since learning can be used to correct for it.

Given these requirements, a CMOS ring oscillator (RO) based structure seems like a good choice due to its low-area and power requirements. To address the third point of robustness, differential ring oscillators should be the preferred topology [128]. In this chapter, we propose a new differential delay cell that has reduced number of transistors compared to the conventional one, thus reducing the energy/cycle due to lower capacitance. Given the popularity of rate-based SNN converted from pre-trained ANN [129], [130], we show results from a fabricated chip in 65 nm CMOS where the proposed delay cell is used in a differential ring oscillator to convert input analog currents to a frequency or rate. Nevertheless, the proposed delay cell can be used in conventional mixed signal applications as well as differential delay line-based neural networks [124], [131], [132]. Further, other neuromorphic computations using coupled oscillators are also gaining in popularity [133], [134]–the proposed oscillators can be used for such computations as well.

The delay element is a fundamental cell for time domain operation, and we proposed a new compact structure that can be used in many time-based NN. In this chapter, we will first introduce the background of time-based computation and digital delay

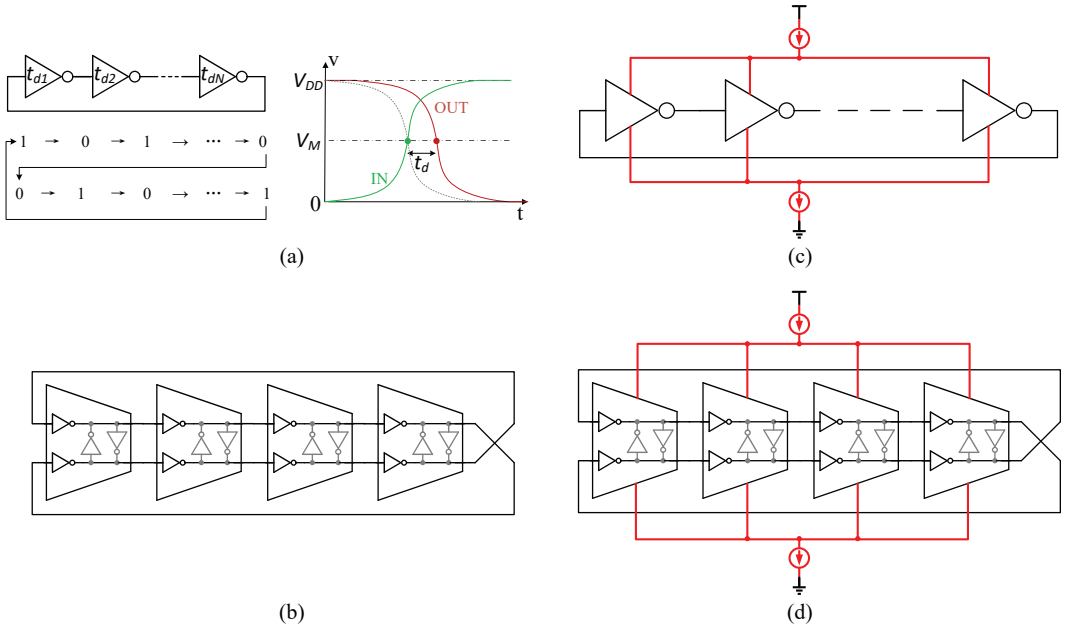


Figure 5.2: (a) Single-ended RO with odd number of stages and delay definition of each stage. (b) Pseudo-differential RO with $N = 4$ stages. (c) Current-starved single ended RO. (d) Current-starved fully differential RO.

circuit in NN applications followed by the conventional CMOS ring oscillator (RO) based current controlled oscillator (CCO) structures. Then, we propose a novel CCO structure and analysed same key parameters interested. After that, we illustrate the measurement results and compared with existing structures. Finally, we will discuss the NN simulation with the proposed CCO neuron model.

5.2 Conventional RO-CCO Implementations

A RO is made from gain stages, or delay stages, in feedback. ROs can be built in any standard CMOS process with a number of delay stages connected in a feedback loop. A single-ended RO is comprised of an odd number stages of inverters cascaded in self-feedback (Fig. 5.2(a)) with capability of rail-to-rail output. The most common way to derive an equation for the oscillation frequency of an N-stage ring oscillator is to assume each stage provides a delay of t_d. For a stable oscillator, the signal must go through each of the delay stages twice to provide one period of oscillation, as shown in Fig. 5.2(a). Therefore, the period is $2Nt_d$, resulting in the oscillation

frequency equation

$$f = \frac{1}{2Nt_d} = \frac{I_{ss}}{2NC_L V_{sw}} \propto \frac{I_{ss}}{C_L}. \tag{5.1}$$

where I_{ss} is the tail current of the a single delay stage, C_L is the total load capacitance of a single stage, V_{sw} is the amplitude of the voltage swing. The delay of each stage t_d is defined as the time interval of output signal and input signal of the same stage at half V_{DD}, illustrated in Fig. 5.2(a). The propagation delay t_d is the most important parameter of this type of oscillator because it directly determines the oscillation frequency, the number of stages and, hence, determines the number of output phases and the power consumption. Moreover, the stability of the delay time t_d also reflects the jitter characteristics in the time domain and the phase noise in the frequency domain.

By eliminating the restriction of odd number of stages in single ended RO, the pseudo-differential RO is more flexible since both odd and even stages are allowed. A 4-stage pseudo differential RO [128] is shown in Fig. 5.2(b). A single delay stage is composed of four inverters, in which two inverters work as a pseudo differential structure, while another two grey inverters constitute cross-coupled regeneration to generate a hysteresis phase shift. The differential RO can offer more output phases compared with single-ended RO with the same number of stages. This is a very important criteria in time domain processing circuits especially where multiple phases are necessary for more accurate operations.

Although single ended configurations consume less area, they are more susceptible to power supply variations and common-mode noise. In modern machine learning accelerators with analog computing neurons co-existing with digital memory and processing, it is extremely important to ensure good noise rejection properties for the analog blocks. A pseudo differential RO can suppress common mode noise and have a better common mode rejection ratio (CMRR) characteristic [128].

To further reduce the sensitivity to power supply variation and noise disturbance, current starved structures, as shown in Fig. 5.2(c), have been introduced [135, 136]. Since the current source and current sink transistors offer a negative feedback in the bias, the current variation is much less than that non-current-starved structures. Another important advantage is that the current-starved structures control the

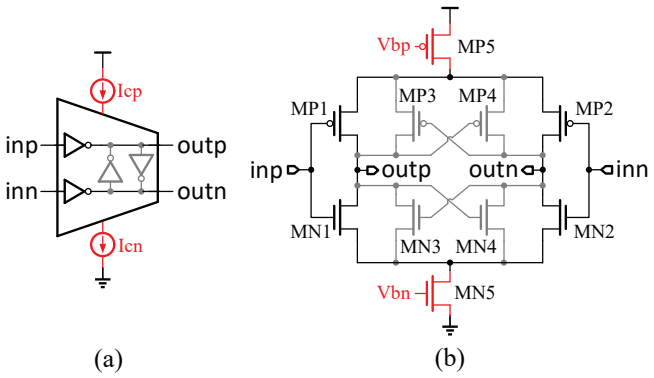


Figure 5.3: Delay stage of a fully differential RO showing the (a) Symbol and (b) Transistor level circuit diagram comprising 8 transistors excluding the shared current sources on top.

output frequency by controlling the bias current, resulting in the name of current controlled oscillator (CCO).

Finally, combining current starving and differential architecture, current starved fully differential RO (Fig. 5.2(d)) has better power supply rejection ratio (PSRR) and CMRR characteristics than other types of RO (shown later in Fig. 5.8). The power supply noise and substrate noise are isolated by the current source and current sink, resulting in a better PSRR. Common mode disturbances, such as temperature variations, can also be suppressed by the differential structure.

Although in practice, current starving transistors are shared by all stages, it is easier to understand the operation of a unit delay cell by assigning a current source and a current sink to it. Fig. 5.3 shows the single delay stage of a current-starved fully differential RO. MN1, MP1 and MN2, MP2 constitute the differential input and differential output pairs. MN3, MP3 and MN4, MP4 constitute the positive feedback latch to offer hysterestic phase shift. The controllable current source and current sink, MP5 and MN5, transform the delay stage from a pseudo differential structure to a fully differential one. The core stage is composed of a differential input-output path and a hysteresis delay cells (HDC) composed of two cross-coupled inverters based positive feedback latch.

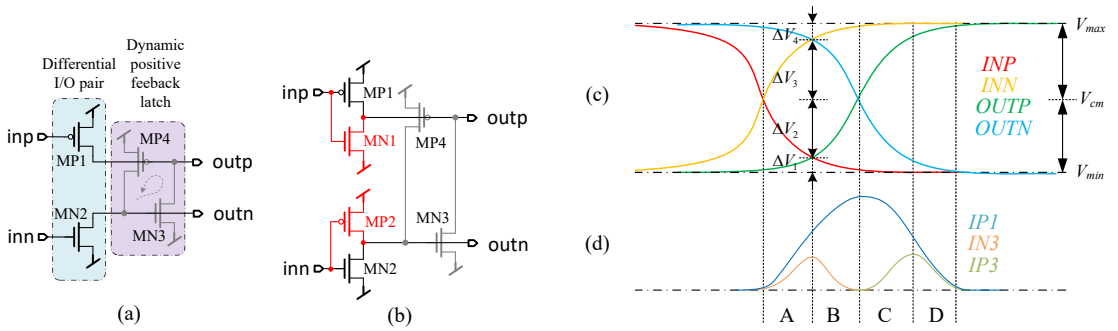


Figure 5.4: Proposed differential stage of RO: (a) The simplest positive feedback element removing 4 transistors from the one in Fig. 5.3 (b). (b) Including 2 more transistors for the start-up circuit. (Note that the current-starved transistors are not shown in (a) and (b) for ease of understanding; the sources of PMOS and NMOS are connected to the soft rail voltages V_{max} and V_{min}, respectively.) (c) Voltage waveform of the different nodes at the input and output of a delay stage. (d) Charging and discharging currents corresponding to the charging node *outp* ($IP3$ is the current of MP3 in Fig. 5.3 (b).)

5.3 Proposed Structure and Theoretic Analysis

As mentioned in Section 5.1, two important metrics for oscillators used for neuro-computing are: (a) high resistance to common-mode noise and (b) low energy/cycle. While the former was addressed through a current starved fully differential architecture as mentioned in the previous section, we address the latter in this section. However, lowering the consumed energy is often associated with an increase in thermal noise induced jitter in the output signal. We propose a new oscillator structure that is better than the conventional structure in terms of energy/cycle while not adversely affecting the jitter performance.

5.3.1 Proposed Oscillator

From both the analysis of the conventional ring oscillators mentioned in section 5.2 and results of simulations shown later in this section (Fig. 5.8), we deduced that the current-starved fully differential structure has both better PSRR and CMRR. To offer differential inputs, outputs and a positive feedback hysteresis, each stage of the conventional differential RO uses four inverters (i.e. 8 transistors) except for the current-starved transistors (Fig. 5.3). Our novelty stems from the observation that only four transistors are necessary to realize oscillation, as shown in Fig. 5.4(a). MP1 and MN2 are the pull-up and pull-down differential I/O pair while MN3

and MP4 serve as their loads, respectively. At the same time, MN3 and MP4 constitute the simplest dynamic positive feedback latch. Another way to interpret this proposed structure is as follows: MP1 and MN3 constitute a dynamic inverter since *outn* and *inp* are almost the same (with a certain phase shift). Similarly, MP4 and MN2 constitute another dynamic inverter since *outp* and *inn* are almost the same (with a certain phase shift). Likewise, the two dynamic inverters compose the regeneration cross pair to generate positive feedback and offer extra hysteresis phase shift. Every transistor acts as an active device and is a load device for another transistor. The push-pull nature and transistor reuse make the cell compact and efficient.

However, one problem is that *inp* and *inn* are strictly inverse only when oscillations have been sustained, while *outn* and *inn* are inverse with a phase delay of $180°/N$. If for some reason, the initial condition of *inp* and *inn* (Fig. 5.4(a)) are same at say 0 V, then *outp* is guaranteed to be pulled up to V_{DD} but the condition of *outn* stays at the initial condition since both MN2 and MP4 are OFF. If initial condition of *outn* was V_{DD}, then the outputs of this stage are both at high voltage and the oscillator gets locked in this state and cannot start oscillating. So a start-up circuit is necessary to maintain robust oscillations. Only a NMOS or PMOS or both (Fig. 5.4(b)) can eliminate the possible stable state and work as start-up circuit. In the above example, when both *inn* and *inp* are 0 V, now with added MP2, it will force *outn* to V_{DD} forcing the outputs of this stage to be opposite in polarity thus breaking the lock state earlier. From SPICE simulations, the structure with start-up of both NMOS and PMOS as shown as Fig. 5.4(b), gives the best frequency performance (see Section 5.3.2) and is adopted in the rest of this chapter.

5.3.2 Startup Circuit

In the last subsection, we proposed a novel structure of CCO and explained the necessity of a start-up circuit. A robust oscillation can be ensured by a start-up circuit with only a PMOS transistor or a NMOS transistor or both. As shown in Fig. 5.5(a), a start-up circuit is equipped at the differential output nodes to eliminate the uncertainty state. The start-up circuit would be a single PMOS acting as a pull-up device to eliminate the uncertainty of *outn*, as shown in Fig. 5.5(b), or a single NMOS acting as a pull-down device to eliminate the uncertainty of *outp*, as

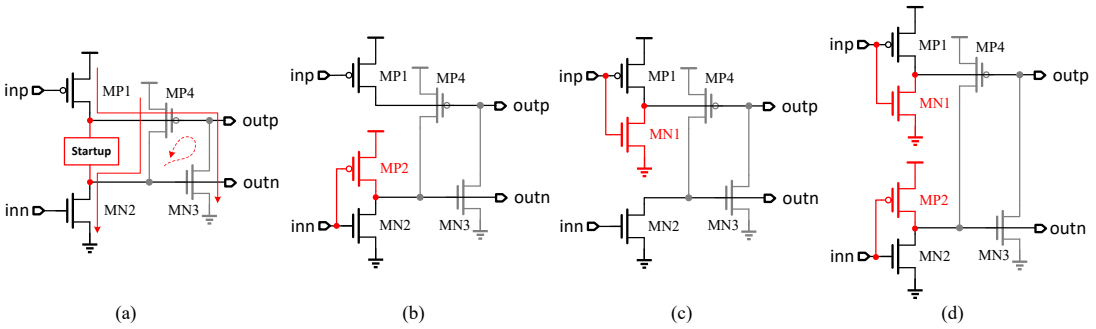


Figure 5.5: Diagram of a single delay stage with (a) start-up circuit. (b) a PMOS start-up. (c) a NMOS start-up. (d) both PMOS and NMOS start-up.

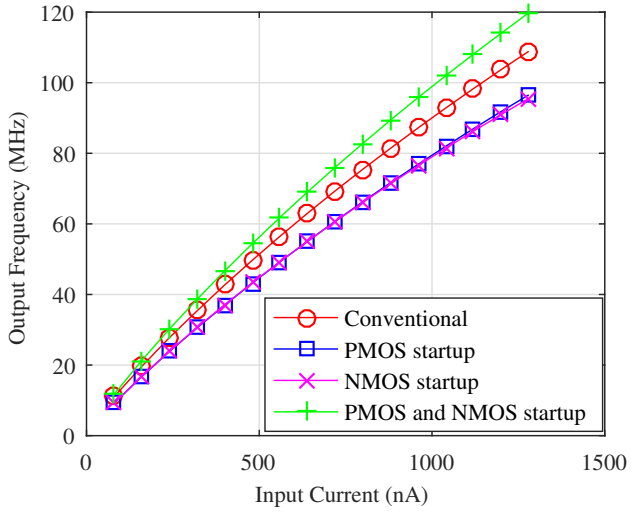


Figure 5.6: Comparison between the frequency of CCOs with different start-up circuits.

shown in Fig. 5.5(c), or both a PMOS and a NMOS to eliminate the uncertainty of both the differential outputs, as shown in Fig. 5.5(d).

The different start-up circuits give different oscillation frequencies. Fig. 5.6 compares the simulated output frequencies of conventional and proposed structure with three different start-up circuits. The CCO with both PMOS and NMOS start-up performs $\times 11\%$ higher frequency than the conventional one while the CCOs with a single PMOS or NMOS start-up degrade the frequencies around $\times 13\%$ compared to the conventional one. Hence, we only use the startup with both PMOS and NMOS in the proposed structure.

5.3.3 Frequency and Energy Dissipation

The novelty in our design of reducing the number of transistors should lead to an increased oscillation frequency per unit current. However, the charging currents also change and hence the combined effect is not obvious. Intuitively, the proposed delay stage offers less charging and discharging current by removing MP3 and MN4, which cuts down on power consumption. To precisely compare the charging/discharging current and the effect on the output frequency, we analyze the load capacitance and charging current model of a single stage in more detail to present a theoretical prediction.

First, the load capacitor of each stage for the conventional structure (Fig. 5.3) is given by:

$$\begin{aligned} C_{L\_conv} &= C_{gdP1} + C_{gdN1} + C_{gdP3} + C_{gdN3} \\ &\quad + C_{gP4} + C_{gN4} + 2C_{g\_NextStage} \\ &= 4C_{gd} + 4C_g \\ &= 8C_{gd} + 4C_{gs}. \end{aligned} \tag{5.2}$$

In comparison, for the proposed structure (Fig. 5.4 (b)), it reduces to

$$\begin{aligned} C_{L\_prop} &= C_{gdP1} + C_{gdN1} + C_{gdN3} \\ &\quad + C_{gP4} + 2C_{g\_NextStage} \\ &= 3C_{gd} + 3C_g \\ &= 6C_{gd} + 3C_{gs}. \end{aligned} \tag{5.3}$$

Thus, $C_{L\_prop} = 3/4C_{L\_conv}$. It is therefore reasonable to expect that the load capacitor decreases by 25% since the number of transistors of each stage decreases by the same amount.

Next, we analyze the charging process of the node of *outp* in Fig. 5.4(b), while the discharging process is same as the node of *outn*. Fig. 5.4(c) illustrates the charging period, which is divided into four phases (A,B,C, and D) according to the operation of the transistors. V_{max} and V_{min} are the swing range of the oscillation, which is lower than rail-to-rail because of the current starved structure. In our design, $V_{max} \approx 900$ mV, $V_{min} \approx 300$ mV, and $V_{cm} \approx 600$ mV. For simplicity, we

assume that $V_{TN} = |V_{TP}| = V_T$, $\Delta V_1 = \Delta V_4 = 50$ mV and $\Delta V_2 = \Delta V_3 = 250$ mV such that $\sum_{i=1}^{4} \Delta V_i = V_{max} - V_{min} = 600$ mV.

$$t_{d\_conv} = \frac{C_{L\_conv}\Delta V_1}{I_{1\_conv}} + \frac{C_{L\_conv}\Delta V_2}{I_{2\_conv}} + \frac{C_{L\_conv}\Delta V_3}{I_{3\_conv}} + \frac{C_{L\_conv}\Delta V_4}{I_{4\_conv}} \tag{5.4}$$

$$t_{d_prop} = \frac{C_{L\_prop}\Delta V_1}{I_{1\_prop}} + \frac{C_{L\_prop}\Delta V_2}{I_{2\_prop}} + \frac{C_{L\_prop}\Delta V_3}{I_{3\_prop}} + \frac{C_{L\_prop}\Delta V_4}{I_{4\_prop}} \tag{5.5}$$

where, I_{i_conv} and I_{i_prop} represent charging current in four different phases of conventional and proposed stage respectively, which are given by

$$I_{i\_conv} = I_{MP1} + I_{MP3} - I_{MN3}, \quad i = 1, 2, 3, 4. \tag{5.6}$$

$$I_{i\_prop} = I_{MP1} - I_{MN3}, \quad i = 1, 2, 3, 4. \tag{5.7}$$

The equations and approximated average values of I_{MP1}, I_{MP3} and I_{MN3} in all phases are listed in Table 5.1, where the charging process is divided into the four phases A, B, C and D as mentioned earlier. The regions of operation of the MOSFET are referred to as 0, 1 and 2 for cut-off, linear and saturation respectively.

The drain current of short-channel MOSFETs is assumed to follow the widely used alpha power law [137], [138]. The carrier velocity saturation coefficient α is between 1.2 to 1.5 for sub-micron CMOS technology. Then, according to the above equations and the estimated values in the table, we can get the propagation delay relationship between the proposed and the conventional ring oscillator (RO) as $t_{d\_prop} = 88.4\% t_{d\_conv}$. Here, t_{d_prop} and t_{d_conv} denote the delay time for the proposed and conventional RO, respectively. Thus, the relation between the oscillation frequencies are:

$$\frac{f_{prop}}{f_{conv}} = 1.156. \tag{5.8}$$

This means that the frequency of proposed RO increases 15.6% compared with conventional structure for the same input current. From equation 5.1, the average current relationship is expressed as:

$$\frac{I_{prop}}{I_{conv}} = \frac{f_{prop}C_{L\_prop}}{f_{conv}C_{L\_conv}} = 0.867. \tag{5.9}$$

Table 5.1: Comparison of charging and discharging current of each contributory transistors for a charging node in four phases, where $\beta = \mu_0 C_{ox} W/L$. β of NMOS and PMOS are the same assuming proper sizing to nominally maximize noise margin. Region 0, 1 and 2 refer to cut-off, linear and saturation regimes of operation of the MOSFET.

Phase	**MP1**	**MP3**	**MN3**
A	region=2 $v_{sg} \uparrow$: $V_{max} - V_{cm} \rightarrow$ $V_{max} - V_{min} - 50mV$ $v_{sd} \downarrow$: $V_{max} - V_{min} \rightarrow$ $V_{max} - V_{min} - 50mV$ $I = 1/2 \times \beta(v_{sg} - V_T)^{\alpha} \uparrow$ $\overline{I} \approx 0.02\beta$	region=0	region=1 $v_{gs} \downarrow$: $V_{max} - V_{min} \rightarrow$ $V_{max} - V_{min} - 50mV$ $v_{ds} \uparrow$: $0 \rightarrow$ $50mV$ $I = \beta(v_{gs} - V_T)^{\alpha/2} v_{ds} \uparrow$ $\overline{I} \approx 0.01\beta$
B	region=2 $v_{sg} \uparrow$: $V_{max} - V_{cm} - 50mV \rightarrow$ $V_{max} - V_{min}$ $v_{sd} \downarrow$: $V_{max} - V_{min} - 50mV \rightarrow$ $V_{max} - V_{min} - V_{cm}$ $I = 1/2 \times \beta(v_{sg} - V_T)^{\alpha} \uparrow$ $\overline{I} \approx 0.07\beta$	region=0	region=1 $\rightarrow$ 3 $v_{gs} \downarrow$: $V_{max} - V_{min} - 50mV \rightarrow$ $V_{max} - V_{cm}$ $v_{ds} \uparrow$: $50mV \rightarrow$ $V_{min} - V_{cm}$ $I = \beta(v_{gs} - V_T)^{\alpha/2} v_{ds} \uparrow$ $\overline{I} \approx 0.01\beta$
C	region=2 $\rightarrow$ 1 $v_{sg}(max) : V_{max} - V_{min}$ $v_{sd} \downarrow$: $V_{max} - V_{cm} \rightarrow$ $50mV$ $I = 1/2 \times \beta(v_{sg} - V_T)^{\alpha} \uparrow$ $\overline{I} \approx 0.07\beta$	region=2 $v_{sg} \uparrow$: $V_{max} - V_{cm} \rightarrow$ $V_{max} - V_{min} - 50mV$ $v_{ds} \uparrow$: $V_{max} - V_{cm} \rightarrow$ $50mV$ $I = 1/2 \times \beta(v_{sg} - V_T)^{\alpha} \uparrow$ $\overline{I} \approx 0.02\beta$	region=0
D	region=1 $v_{sg}(max) : V_{max} - V_{min}$ $v_{sd} \downarrow$: $V_{max} - V_{cm} \rightarrow$ $50mV$ $I = \beta(v_{sg} - V_T)^{\alpha/2} v_{sd} \uparrow$ $\overline{I} \approx 0.02\beta$	region=1 $v_{sg} \uparrow$: $V_{max} - V_{cm} \rightarrow$ $V_{max} - V_{min} - 50mV$ $v_{ds} \uparrow$: $V_{max} - V_{cm} \rightarrow$ $50mV$ $I = \beta(v_{sg} - V_T)^{\alpha/2} v_{sd} \uparrow$ $\overline{I} \approx 0.02\beta$	region=0

This implies the average current of the proposed RO decreases by 13.3% compared to the conventional structure.

To verify these theoretical models, we conducted the current-frequency (I-F) transfer curve simulations of the conventional and proposed 4 stage RO using 65 nm CMOS models in SPICE. Fig. 5.7 shows the simulation results and the theoretical prediction results of the output frequency. As predicted, the proposed oscillator indeed produces higher frequency than the conventional one for the same input current. However, the theoretical values are slightly greater than that of simulation because of the inaccuracy in estimation of the current of the transistor MP3.

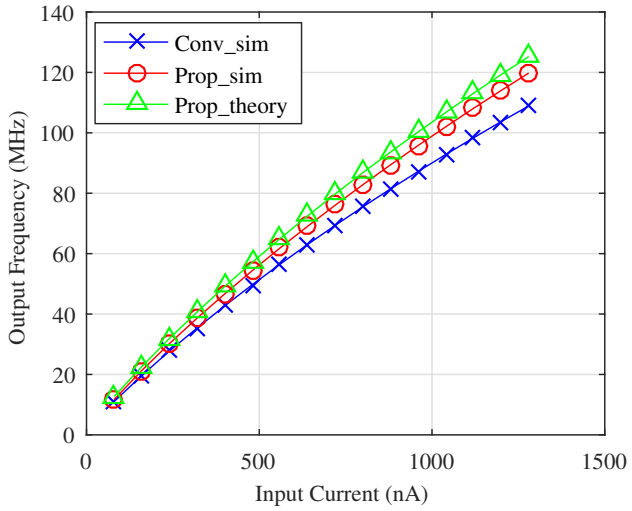


Figure 5.7: Simulation results of oscillation frequency of both conventional and proposed 4 stage fully differential RO, along with the theoretical prediction for the proposed structure. As expected, the proposed RO has higher frequency for the same current.

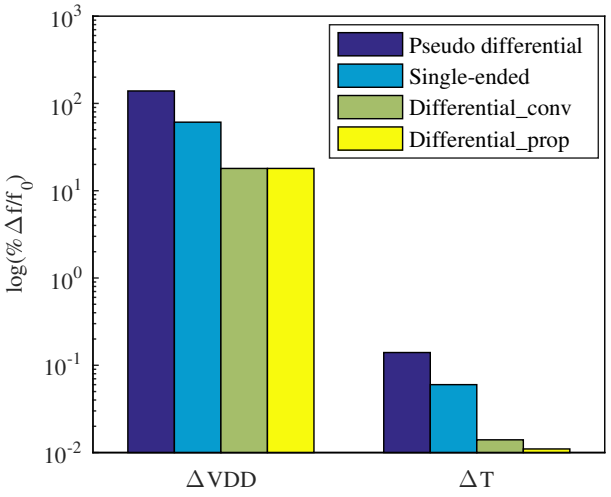


Figure 5.8: Comparison of the sensitivity of four different types of CCO topology with power supply and temperature: pseudo differntial, current starved single-ended, conventional and proposed current starved differential CCO. The proposed and conventional current starved differential designs have similar sensitivity that is much less than pseudo-differential or current starved single ended architectures.

5.3.4 Robustness and Jitter

While the increase in oscillation frequency is good, it is not useful if the new oscillator has a degradation in other key metrics such as robustness and jitter. We first evaluate the robustness of the proposed oscillator in comparison to the

conventional one through simulations. The simulation results of line sensitivity ($\%/V$) and temperature sensitivity ($\%/T$) of four types of RO-CCOs is shown in Fig. 5.8. As expected, the characteristics of differential current starved structures, both proposed and conventional, are better ($\approx 18\%$ for line sensitivity and $< 0.014\%$ for temperature sensitivity) than those for the non current starved pseudo differential RO ($\approx 139\%$ for line sensitivity and $\approx 0.14\%$ for temperature sensitivity). For current starved structures, the differential structures are better than single ended ones ($\approx 61\%$ for line sensitivity and $\approx 0.06\%$ for temperature sensitivity), as expected. It can be seen that the line sensitivity and temperature sensitivity of the proposed differential structure is not degraded compared to the conventional differential structure. The relatively high value of sensitivity to power supply is traced back to the tail current sources coming out of saturation at power supply voltages lower than 0.9 V – this can be solved by reducing the current range. Confined to $1 - 1.2$ V for power supply, the sensitivity is only around 1.5%.

Following [139], the variance of period jitter for a RO can be expressed as:

$$\sigma_\tau^2 = \frac{KT}{If_0}[\frac{2}{V_{DD} - V_t}(\gamma_N + \gamma_P) + \frac{2}{V_{DD}}]. \tag{5.10}$$

where

$$f_0 \approx \frac{I/C}{MV_{DD}}. \tag{5.11}$$

M is the number of delay stage. γ_N and γ_P are technology-dependent noise factors for NMOS and PMOS respectively. Thus,

$$\sigma_\tau^2 \propto \frac{C}{I^2}. \tag{5.12}$$

In our implementation, the load capacitor and average current decrease by 25% and 13.3% respectively. Hence,

$$\frac{\sigma_{Prop}^2}{\sigma_{Conv}^2} = \frac{(1 - 0.25)}{(1 - 0.133)^2} \approx 1. \tag{5.13}$$

Consequently, the jitter of the proposed CCO is expected to be approximately the same as that of the conventional structure. This has been confirmed in simulations and measurement–we present these results in the next section.

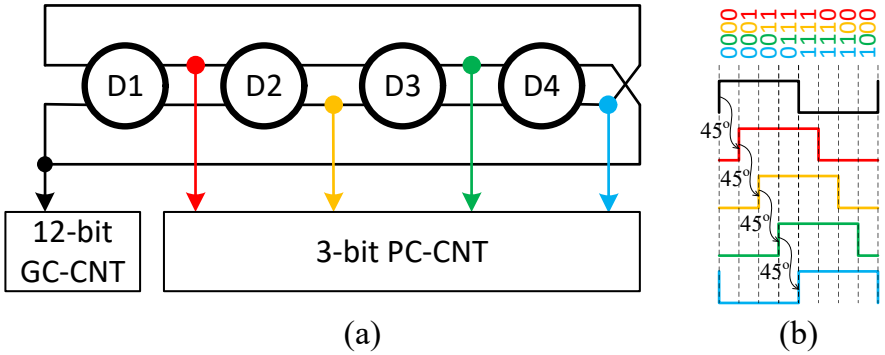


Figure 5.9: (a) Time to digital converter (TDC) composed of a four delay stages differential CCO structure and two different counters for coarse and fine conversion separately. (b) The sequence diagrams for Gray code counter and the phase code counter with 45° phase shift of each other as well as the corresponding 3-bit complementary phase code for one oscillation cycle.

5.3.5 Frequency to Digital Conversion

For usage within a neural network system, the raw frequencies of the CCO need to be often converted to a digital word. The easiest method to do this is to take the CCO output as a clock signal and pass it to a counter [126]. However, this method incurs a large conversion time ($\propto 2^N$, N is the number of stage) and concomitantly large conversion energy. A different approach, following [125] is used in our work as described next.

Fig. 5.9 shows the diagram of a time-to-digital converter (TDC) with a four delay stage differential CCO and the sequential counters. Although the four stage differential CCO has eight outputs totally, only half are unique with separate phase information while the rest are exactly the inverse of the unique phases and offer the same phase information. So only four phases with 45° phase shift of each other, as shown in Fig. 5.9(a), are utilized to generate the phase code counter (PC-CNT). Only $360°/45° = 8$ codes are generated by the four stages differential CCO and hence only 3-bits are available in the PC-CNT acting as fine counter. The sequence diagrams of the four choice outputs and the corresponding complementary symmetry phase codes are illustrated in Fig. 5.9(b). The opposite phase of the last phase of PC-CNT is chosen to clock a Gray code counter (GC-CNT) serving as coarse counter. The number of bits of GC-CNT depends on the input current range while the number of bits in PC-CNT determines the accuracy of the converter. A Gray code counter is selected for higher reliability in neural networks with tightly

packed layout. To enable wide dynamic range testing, a 12-bit GC-CNT is used in our design but the bit width can be optimized in neural network applications.

Both the phase code counter and the Gray code counter possess the merits of energy efficiency and reliability since only one bit trips during state transition. Regarding the choice of number of stages in the CCO, we note that more stages of the CCO structure ($\times M$) offer more valid output phases ($\times M$) and thus more bits in PC-CNT, but at a lower frequency (under the the energy dissipation) and thus less bits in GC-CNT. Overall, the total number of bits is constant but more CCO stages consume more area overhead ($\times M$). Assuming the same input current for both cases, the tail current for CCO core is constant and, hence, the energy per conversion due to CCO is constant. Energy dissipated in the counter is less for more CCO stages, but this is much smaller than the CCO energy dissipation. Hence, due to the need of small footprint of the CCO, a 4-stage CCO core is designed in our work.

5.4 Measurement Results and Discussion

The prototype chip has been fabricated in a 65 nm standard CMOS technology and includes a conventional four-stage CCO and the proposed CCO with output inverter buffers and peripheral circuits. The die micrograph and major sub-blocks are shown in Fig.5.10(a). The entire block is 110×90 μm^2 in size, including BIAS, common mode feedback (CMFB), phase interpolator (PI), counter (CNT), parallel-in-serial-out (PISO) interface, and controller block.

The testing board is shown in Fig.5.10(b), where the configuration switches, testing points and power supply are on the front side of the testing board while an FPGA board is hooked to the rear side. This section presents the measurement results and explains the testing conditions and methodology.

The testing setup is illustrated in Fig. 5.11. The bias current of the CCO can be controlled in coarse steps using an external resistor R_{ext} on the board and in fine steps by a voltage input $VDAC$ from an on-board Digital to Analog Converter. The combination of resistor and voltage are converted into a reference current by a rail-to-rail operation amplifier based V-I converter and then copied as the starved current for the CCO core by current mirrors. The range and amplitude of the

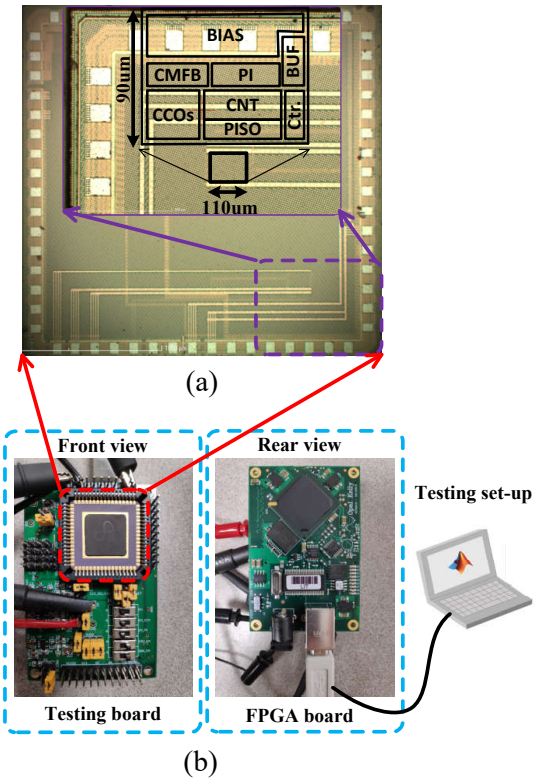


Figure 5.10: (a) Die photo and (b) testing set-up photo of the fabricated IC in 65 nm CMOS. The testing board stacks with an FPGA board in charge of data transfer to and from the PC.

current is also programmable by the configuration bit $ISEL$. CCO cores selection is configured by CCO_SEL. The output of the oscillator is converted to a digital code by the coarse/fine digital quantizer as described earlier. The bit stream is serialized and sent to the FPGA for storage and analysis. The frequency of oscillation can be inferred from the digital code without having to measure a high frequency signal directly.

Note that the V-I converter is used just for convenience of testing. In an actual neural network, the input current I_{in} to the CCO will come from a synaptic array. Fig. 5.12 shows an example how the CCO can be integrated into a neural network system with resistive synapses [140]. Here the CCO is used as the $i-th$ neuron in the $(l+1)-th$ layer and the count output from one neuron feeds a digital-analog converter (DAC) in the next layer to create voltage inputs. The custom activation function can be realized by the counter (CNT) following the oscillator neuron. By setting the counter window T_{EN}, the output of counter can mimic the shifted Rectified Linear Unit (ReLU), as illustrated in Fig. 5.12(b). When the clock period

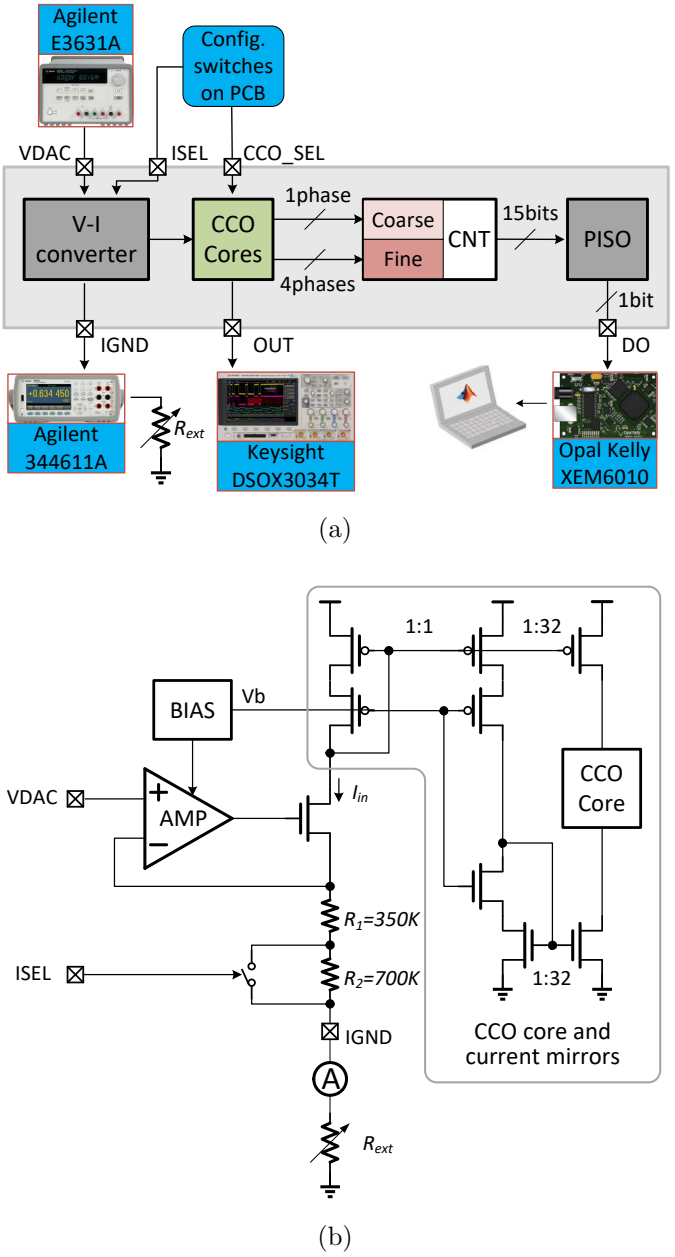


Figure 5.11: (a) Block diagram of the fabricated IC and experimental setup for measurement. A tunable V-I converter provides input current to the CCO while a counter based on-chip digitizer is used to estimate the frequency of oscillation. (b) Circuit details of the V-I converter. ISEL is a digital bit used to enable or disable an on-chip resistor for coarse control of input current.

of the oscillator neuron T_{osc} is less than T_{EN}, the out of counter is proportional to the frequency of the oscillator neuron f_{osc}, otherwise the out of counter keeps zero. Note that this is just an example implementation just to clarify that the V-I converter is not a part of the neuron.

Measured frequency transfer curves of the conventional and proposed CCOs are

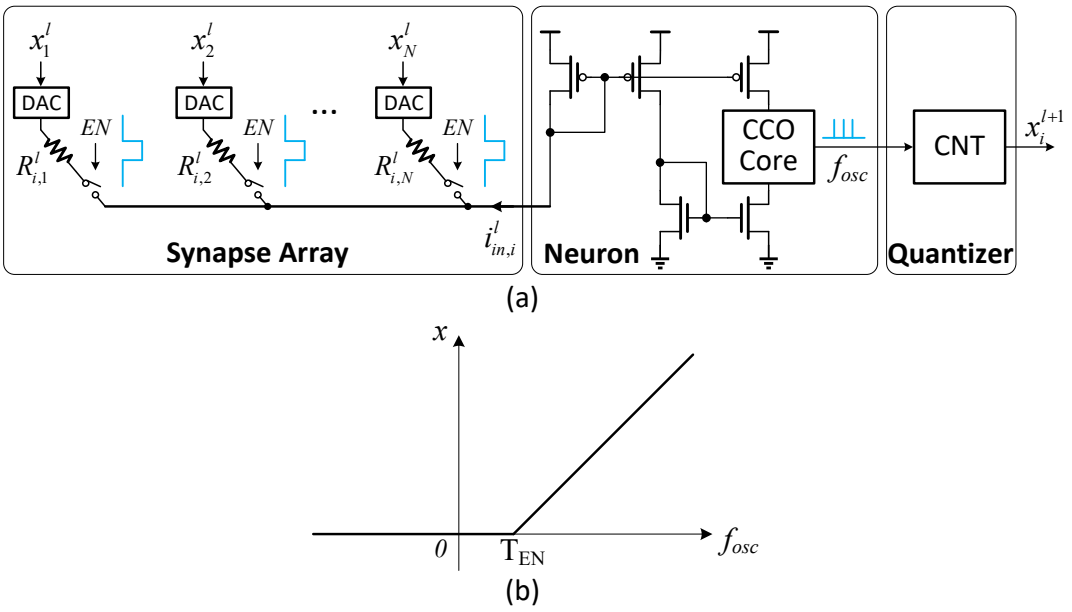


Figure 5.12: An example of neuron oscillator integrated neural network system. (a) The input current i_{in} to the CCO comes from a synapse array in an actual neural network. (b) The shifted ReLU function

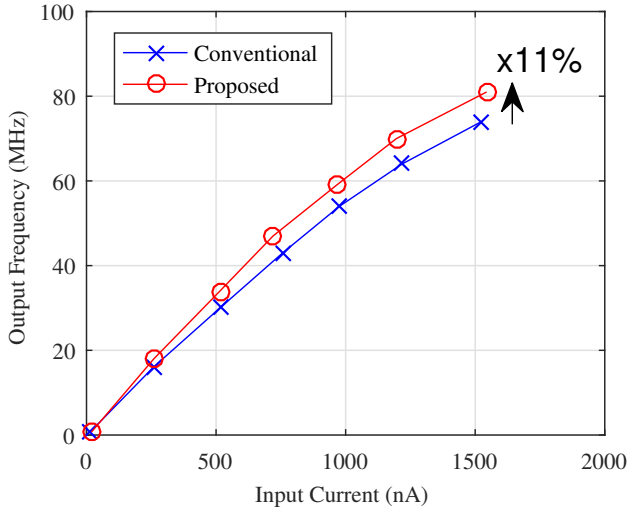


Figure 5.13: Measured frequency for different input current.

plotted in Fig.5.13 at different input current obtained by varying the DAC voltage over a wide range. The measured frequency of proposed CCO indeed achieves ≈ 11% improvement compared with that of the conventional one for the same current input, ranging approximately from 0 to 1.5 μA, close to the simulation based result in Section 5.3. Fig. 5.14 shows the measured power consumption at different oscillation frequencies. At the same oscillation frequency, the proposed CCO consumes 13% less energy due to the less number of transistors and the corresponding capacitors.

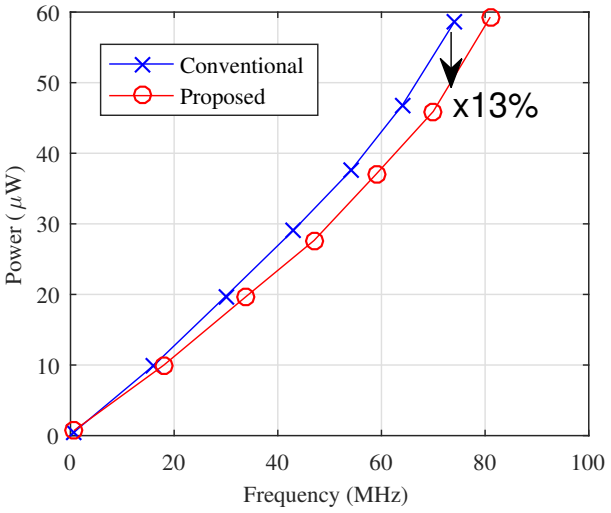


Figure 5.14: Measured power consumption at different frequencies.

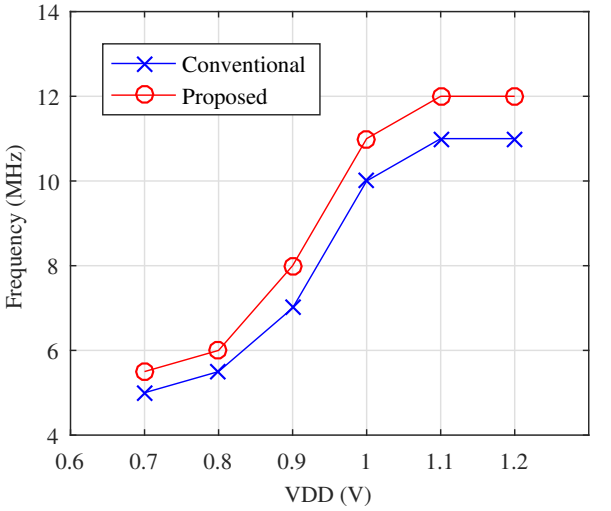


Figure 5.15: Measured frequency for different V_{DD} under the condition of $VDAC = 0.2$ V and $ISEL = 0$.

The lowest energy/conversion step is 0.56 pJ/cycle at lower frequency while it increases to 0.63 pJ/cycle at higher frequencies. Equivalently this energy efficiency can be quoted as 0.63 μW/MHz. Finally, the dependence of the oscillation frequency on power supply voltage is tested. The experimentally obtained frequencies of the conventional and the proposed CCOs with different power supply V_{DD} are shown in Fig.5.15. The testing is under the condition of $VDAC = 0.2$ V, $R_{ext} = 0\ \Omega$ and $ISEL$ is set to low (total resistor is 1.05 $M\Omega$ in this case). The result shows a sharp drop in frequencies for V_{DD} less than 1 V and is traced back to the reference circuit not functioning properly at these voltages resulting in a change in the starving

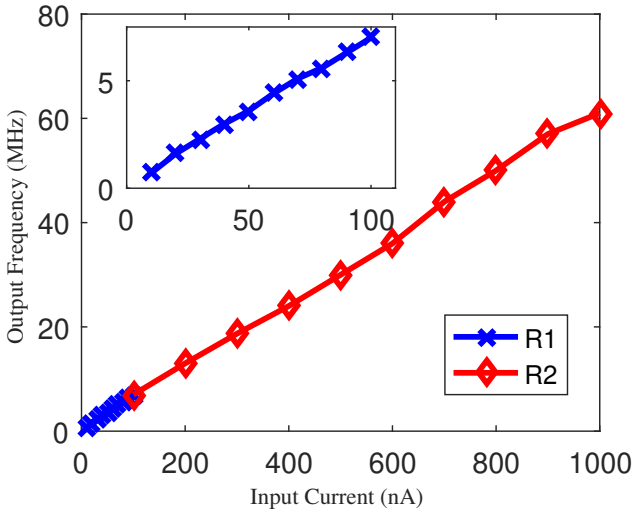
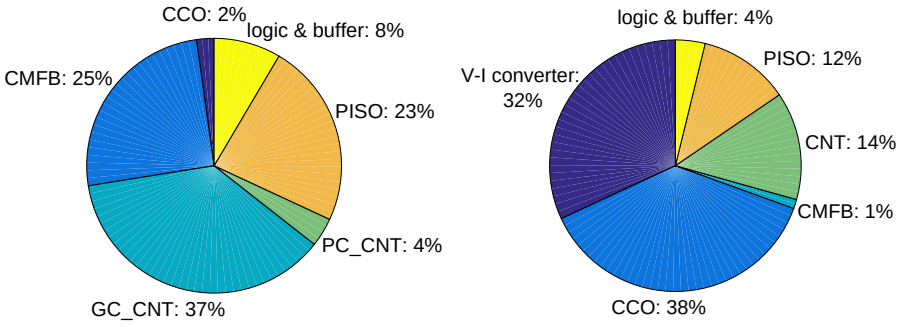


Figure 5.16: Measured frequency with different current by adjusting external resistor and input voltage respectively.



(a) Area (b) Power

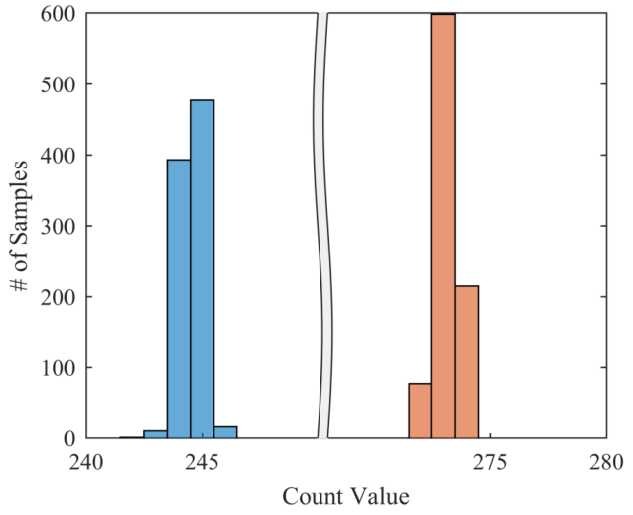
Figure 5.17: (a) Area and (b) Power contributions (simulation) from each sub-circuit in the system.

current. However, the proposed oscillator produce higher frequencies than the conventional design for all values of V_{DD}.

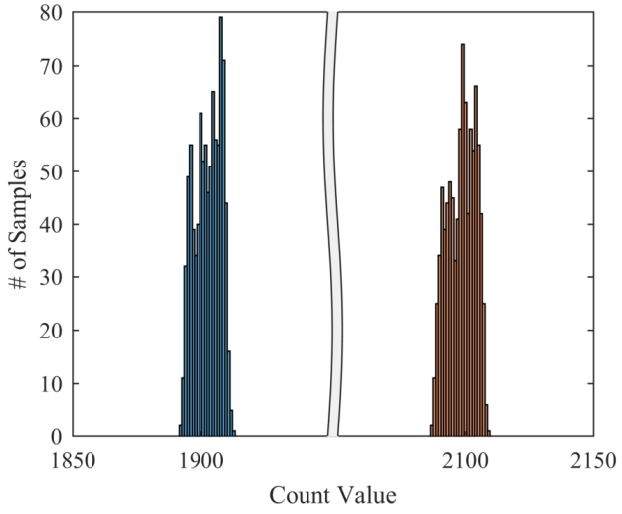
As mentioned before, the frequency of CCO can be tunable by both input voltage $VDAC$ and the external resistor R_{ext}. Fig.5.16 shows the transfer curves, where $VDAC$ sweeps from 100 mV to 1 V. First, the external resistor R_{ext} is fixed to 8.95 MΩ and the configuration bit $ISEL$ is set to low, making the total resistance $R_{tot} = 10$ MΩ. Tuning input voltage $VDAC$ from 100 mV to 1 V covers the input current range from 10 nA to 100 nA, as shown in the blue line (R1).Then, the external resistor R_{ext} is fixed to 650 KΩ and $ISEL$ is set to high, leading the total resistance $R_{tot} = 1$ MΩ.Tuning input voltage $VDAC$ from 100 mV to 1 V covers the input current from 100 nA to 1000 nA, as shown in the red line (R2). The

testing points show the two kinds of tuning matches well and one can act as coarse tuning while another as fine tuning.

Fig. 5.17 shows the relative area utilization and the simulated power breakdown. The 4-stage CCO core contributes a small area of 44 μm^2. It may be noted that although the V-I converter consumes a significant system power in our testing, this block is not an intrinsic part of our design but is there only for testing as mentioned earlier.



(a)



(b)

Figure 5.18: Jitter performance comparison. The variation of the counter output for a specified time interval. (a) low input current, jitters of conventional structure (left) and proposed structure (right) are 0.23% and 0.20% respectively, (b) high input current, jitters of conventional structure (left) and proposed structure (right) are 0.26% and 0.25% respectively.

To test the jitter performance, the outputs of CCOs are connected to a time-to-digital converter (TDC) in the chip, as shown in Fig. 5.11(a) and described in the earlier section. A parallel in serial out (PISO) circuit is used to reduce the number of output pads. The 15-bit parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control. In our testing, the serial output data is samples by the FPGA (Opal Kelly XEM6010) and then passed to PC to decode. Fig. 5.18(a) and (b) display the measured jitter performance for the conventional and proposed structure at a low and high frequency respectively. It can be seen that the two structures perform approximately the same with around 0.2% jitter at lower frequency and around 0.25% jitter at higher frequency.

Table 5.2: Performance Summary and Comparison of Ring Oscillator

Reference	Technology (nm)	Topology	Power (μW/MHz)	Frequency (MHz)
This work	65	Differential, 4-stage	0.63	80
Lee 2018 [141]	2000 (SOI)	Single-ended, 3-stage	360	-
Yi 2019 [125]	65	Differential, 4-stage	0.71	70
Yao 2017 [126]	350	Single-ended, 3-stage	0.2	4
Basu 2010 [142]	350	Single-ended	17.4	100
Sahoo 2017 [143]	65	Single-ended	-	1.5

Table 5.2 compares characteristics and performance of various oscillators used in neuromorphic applications. The proposed CCO has one of the highest energy efficiencies reported so far (except [126]). However, [126] presents a single ended design with a much lower frequency of operation. Further, compared with the other differential design in [125], the proposed work uses 25% less area.

Further, a detailed performance comparison is summarized in Table 5.3 for the recent time based ADC architectures. To compare with other ADCs, the center frequency of CCO is set to be 60 MHz in the prototype ADC. With the input analog bandwidth of 500 kHz, running at 5 MS/s, it consumes an average of 72 μW from 1.2 V at full input signal swing, out of which 40 μW are drawn by the digital circuits and the buffer, while 32 μW are consumed by the analog circuits (including CCO core and bias). The proposed novel CCO structure consumes less

Table 5.3: Performance Summary and Comparison with State of the Art

Reference	Technology (nm)	Architecture	Supply (V)	Power (mW)	Area (mm^2)	Bandwidth (MHz)	FOM (fJ/conv.)	Application Comments
This work	65	Novel CCO + TDC	1.2	0.02	0.004	0.5	79[1]	silicon neuron in neuromorphic applications
Leene 2018 [144]	65	VCO + $\Delta\Sigma$	0.5	1.3	0.006	0.011	175	electrode bio-potential recordings
Li 2017 [145]	130	VCO + $\Delta\Sigma$	1.2	1.05	0.13	0.4	118	continuous-time delta-sigma modulator
Young 2014 [146]	65	OTA + VCO	1.2	38	0.49	50	294	continuous-time delta-sigma modulator
Talor 2013 [147]	65	VCO	0.9	11.5	0.075	5.08	246	digital wireless receiver, CT delta-sigma modulator
Kim 2014 [148]	130	delay cell + 3-D Vernier	1.5	0.33	0.28	0.5	400	time-of-flight (ToF) application
Tu 2017 [149]	40	PWM + $\Delta\Sigma$	1.2	0.02	0.015	0.005	1643	CMOS Image Sensor, X-ray detector
Jayaraj 2019 [150]	65	VCO + $\Delta\Sigma$	1.2	1	0.06	2.5	151	continuous-time (CT) ADC
Jayaraj 2019 [151]	65	VCO + $\Delta\Sigma$	1	0.1	0.06	2.3	8.6	continuous-time (CT) ADC
Zhong 2018 [152]	40	VCO + $\Delta\Sigma$	1.1	0.91	0.086	5.2	-	continuous-time (CT) ADC

[1] FOM is based on simulation since V-I converter does not support high frequency input.

area and lower energy overhead by using less transistors. Besides, the Gray code counter and phase code counter consume less dynamic energy due to their merit of minimum change code. These lead to high energy efficiency of the time based ADC quantified by the figure of merit (FOM) calculated as:

$$FOM = \frac{P}{2^{ENOB} * 2 * BW} \tag{5.14}$$

Where, P is the total power consumed by the CCO-based ADC. The ADC achieves an FOM in the range of $79 - 102$ fJ/conv-step for different input frequencies. Note that [151] achieves much lower FOM by using 2^{nd} order noise shaping loop; however, this benefit comes mostly due to reduced quantization noise and is orthogonal to the improvements in oscillator structure we report. Also, this improvement comes with a $> 10\times$ area penalty which may not be suited for neuron designs.

5.5 Neuron Oscillator

5.5.1 Neural Network Simulation

In this section, we explore the potential application of the proposed CCO as a neuron in neural network (NN) implementations. The custom activation function of the proposed neuron is modelled using the previous experimental results of the CCO (see Fig. 5.13 – 5.15 and Fig. 5.18). The I-F transfer curve of CCO has a high sensitivity to process variations and hence the neurons on a chip are expected to have mismatch. Fig. 5.19 shows the mismatch of CCO through Monte Carlo (MC) simulation. As seen from the figure, the standard variation is 1.8 MHz at the mean frequency of 39.7 MHz, leading to a coefficient of variation of $\approx 5\%$. The mismatch of CCO was obtained by performing MC runs while sweeping input currents. The variability in the slope of this I-F curve is included in the NN simulation as shown next. Note that synaptic non-idealities are a separate issue and has to be considered for the full system as well. This is beyond the scope of this work but many strategies to handle synaptic non-idealities such as mismatch, low-precision and write non-linearity have been published [126], [153–156].

At first, the neural networks are trained and tested using ReLu activation function to establish a benchmark for further evaluation of proposed activation function.

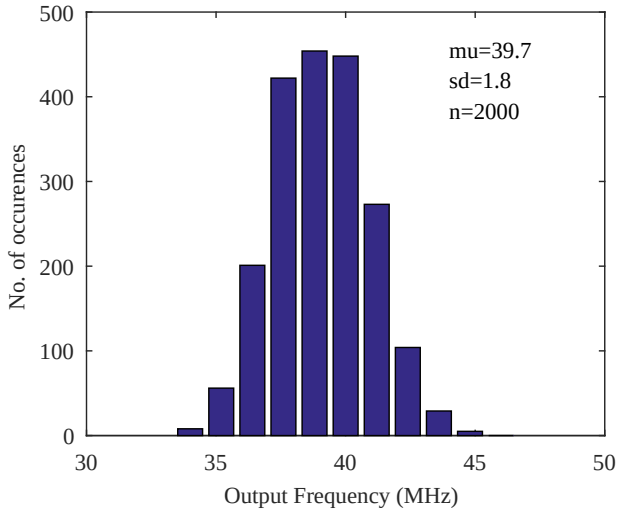


Figure 5.19: Results of Monte Carlo simulations of the proposed CCO generated from 2000 runs.

Algorithm 4 Hardware Simulation

1: Fit a straight line for each set of readings (i-f curves) and get a distribution of slopes. Calculate mean slope $\tilde{m}$.
2: Divide the curves in two regions and fit a straight line for each region. Optimum boundary between regions $\mathcal{B}$ is determined based on minimising MSE over all curves.
3: For both the regions, obtain the slope distributions and model them using Gaussian distributions: $\mathcal{N}_1(\mu_1, \sigma_1^2)$ and $\mathcal{N}_2(\mu_2, \sigma_2^2)$
4: Noise is modelled as Gaussian noise $n \sim \mathcal{N}(\mu = 0, \sigma = jitter)$
5: Finally the custom activation function is given by: Custom Activation(x) = $ReLu(m_1 x[x < \mathcal{B}] + m_2 x[x > \mathcal{B}]) + (m_1 x[x < \mathcal{B}] + m_2 x[x > \mathcal{B}]) \times n$, where $m_1 \sim \mathcal{N}_1$ and $m_2 \sim \mathcal{N}_2$
6: **For hardware simulation (inference):**
training activation: $ReLu(\tilde{m}x)$
inference activation: Custom Activation(x)
7: **For hardware simulation (training + inference):**
training activation: Custom Activation(x)
inference activation: Custom Activation(x)

For hardware simulation, we adopted two strategies. In the first one, the networks are trained using ReLu activation and the custom activation function is introduced only during inference. In the second strategy, the custom activation is used both during training and inference. The simulation details are described in detail in the algorithm 4.

We used three different datasets (MNIST [157], fashion MNIST [158], CIFAR10 [159])

Table 5.4: Neural Network Simulation

Datasets		MNIST		Fashion MNIST		CIFAR10	
Network		ANN	CNN	ANN	CNN	ANN	CNN
Simulation Accuracy	Software	0.982	0.9926	0.8919	0.9174	0.5284	0.7478
	Hardware (inference)	0.9812	0.9915	0.8846	0.9088	0.5014	0.6974
	Hardware (training+inference)	0.9829	0.9931	0.8937	0.9135	0.5257	0.7435

and two network topologies (one ANN and one CNN) to evaluate the performance of the proposed neuron model. Since the primary aim of these simulations is to verify the viability of the CCO based neuron, instead of experimenting with different network architectures, we used the same ANN and CNN architectures for all the datasets and compared the performance of the hardware simulations with their pure software counterpart. The ANN comprises of two fully connected hidden layers with 800 and 300 neurons respectively. The CNN network architecture used can be described as: $A \times B \times C - 32c5 - m2 - 64c5 - m2 - 128c1 - m2 - fc512 - 0.5d - fc10$, where the input dimension is $A \times B \times C$, XcY represents a convolution layer with X convolution filters with $Y \times Y$ size, mZ represents a $Z \times Z$ 2D max-pooling layer, Pd represent a dropout layer with dropout rate P and fcL represent a fully connected dense layer with L neurons. All models are trained and tested through google colaboratory on a server with Intel Xeon CPU and NVIDIA Tesla K80 GPU. All the models are trained with categorical crossentropy loss and Adam optimizer and the testing accuracies for all the datasets and network topologies are averaged over 3 trials.

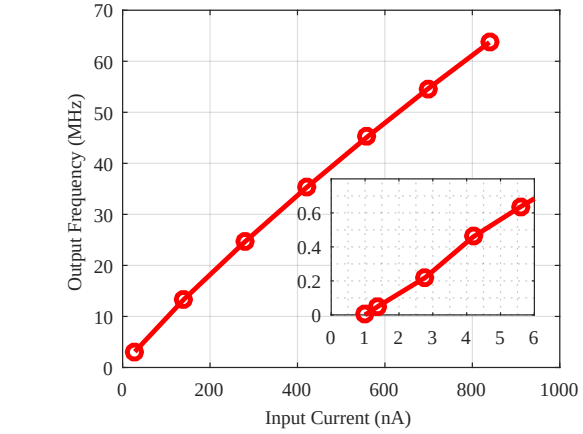
The results (mean across 3 trials) are shown in table 5.4. The key observations from the results are as follows: firstly, for simple datasets like MNIST, there is almost no loss of accuracy even if the custom activation function is introduced only during inference. For a relatively more complex fashion MNIST dataset, there is a small loss ($\sim 1\%$) in testing accuracy, while for more complex CIFAR10 dataset, the loss of accuracy becomes significant ($\sim 2.5 - 4\%$). Secondly, the loss in accuracy can be almost completely recovered even for complex datasets like CIFAR-10, if the custom activation is introduced during training i.e. weight learning through back-propagation is able to correct for any non-ideality resulting from the custom activation. Thirdly, in some cases, when the custom activation is introduced during training, the testing accuracies seem to surpass the software accuracy. This can be attributed to Gaussian Noise injection in the custom activation acting as noise

regularization and improving overall network performance [160]. For all the datasets and both hardware simulation settings, the loss of accuracy is similar for both ANN and CNN network topologies. Since the ANN uses only two hidden layers while the CNN architecture consists of large number of layers, this observation establishes the scalability of the proposed neuron model. Finally, we evaluated in detail the variability in the performance of the classifiers caused by random variation of the custom activation function described in step 5 of Algorithm 4. For MNIST classification task with ANN and CNN, we evaluated the classification accuracies over 20 trials where the slopes of the custom activation are randomly generated for each trial. The standard deviation of classification accuracy for ANN and CNN are 0.15% and 0.10% respectively. This goes to show that the performance variation introduced by the slope variability is very small and therefore, further proves the viability of the proposed neuron model.

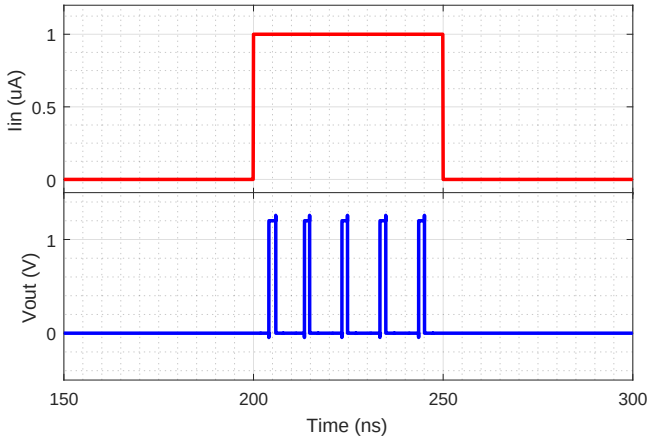
Changes in power supply voltage and temperature are ideally common mode perturbations and affect the CCO less due to differential structure. Moreover, due to sharing of power supply, if all neurons are affected by the same factor, RELU based neural networks will not be ideally affected. However, due to variations between neurons, there will be some effect in real implementations. To capture this effect, Monte-Carlo simulations were done to find distributions of supply voltage sensitivity (temperature sensitivity is almost negligible as shown in Section 5.3). The resulting sensitivity distribution had a mean of -0.78% and standard deviation of 1.74% when V_{DD} varied in the range of $1 - 1.2$ V. Including this as an additional perturbation for the neuron outputs during inference, the results for CIFAR-10 were re-evaluated. We observed negligible reduction in accuracy of $< 0.5\%$ confirming our hypothesis that the network is relatively robust against supply variations.

5.5.2 CCO as Spiking Neuron

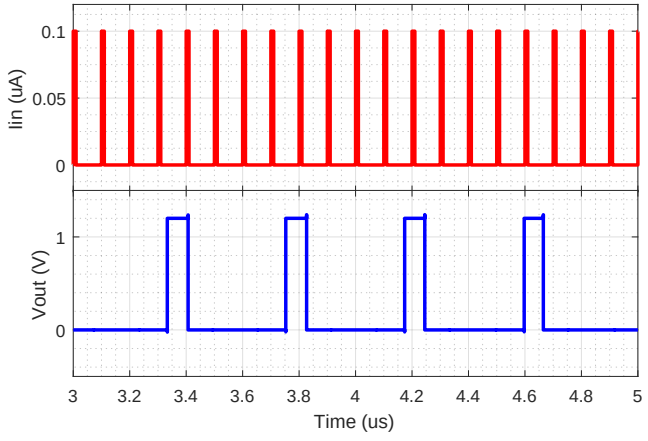
The earlier sections showed the usage of the CCO as a rate encoded neuron similar to its use in [125, 126]. However, it may also be used as a spiking neuron with precisely timed output spikes as needed in spiking neural networks (SNN). While one way to use the CCO as a spiking neuron is shown in [123], we show next how some of the critical features can be included in the CCO much easily. We identify



(a)



(b)



(c)

Figure 5.20: Usage of the CCO in an SNN based on SPICE simulations: (a) With leak current of 1 nA added, the CCO does not oscillate for input current less than I_{leak}. (b) For a step current input, the neuron only oscillates during the step input. (c) For pulse current inputs, the charge is integrated on a current mirror's capacitor and outputs spike when the CCO reaches a desired phase.

one of the phases of the CCO as the spike phase and the corresponding output of the fine counter is plotted as spike output.

First, we show in SPICE simulations that adding a leak current source [161] next to the PMOS input current mirror in Fig. 5.11(b) can act to suppress spurious neuron spikes at low input currents. The resulting I-F curve plotted in Fig. 5.20(a) shows that indeed valid neuron firing starts for input currents larger than I_{leak}=1 nA. Next, the response of the CCO neuron to a step of magnitude I_{in}=1 µA is shown in Fig. 5.20(b). As expected, the neuron fires spikes only when the step is applied. Lastly, to see how to integrate such a neuron in a network, we show the response of the CCO neuron to spike inputs that may come from other neurons in the network. Plotted in Fig. 5.20(c), it shows that for input synaptic current pulses of magnitude 100 nA, the neuron integrates the charge on the PMOS current mirror and produces an output spike for every 4 input spikes.

5.6 Conclusion

In this chapter, an energy- and area-efficient full differential CMOS current controlled ring oscillator (CCO) is presented as a suitable and compact structure in neurocomputing applications. The CCO achieves higher frequency while consuming lower area and lower energy due to less transistors being utilized compared with the conventional structure. By eliminating the unnecessary transistors, the proposed structure is composed of a simplest dynamic positive feedback latch and differential pairs, saving $\approx$ 25% in size. The CCO can be tuned by both input voltage and external variable resistor. The measurement results show our work achieves 11% frequency improvement and 13% energy-efficient without degrading the jitter and phase noise characteristics. Simulations of both ANN and CNN with the proposed CCO characteristics as the neuron transfer curve show no degradation in accuracy if the CCO nonlinearity is included in training.

Chapter 6

Conclusions and Future work

Inspired by the human brain, neuromorphic computing implemented by very-large-scale integration (VLSI) system aims to mimic extremely energy efficient neuro-biological architectures. In recent years, it has emerged as an exciting and promising research area, primarily owing to the paradigm shift from conventional computing architectures to asynchronous data-driven, cognitive computing. Various novel and interesting algorithm and hardware researches have demonstrated its power in energy efficient applications. Considering the growing data volumes and increasing requirement for intelligent versatile computing, neuromorphic architecture will further unleash huge potential by breaking the limitations of the traditional von Neumann architecture.

This dissertation has presented different neuromorphic circuits and algorithmic techniques, namely, near- and in-memory computing as well as time-based computing for energy efficient artificial intelligence applications. We first presented the CRAM based in-memory computing technique to perform global parallel image processing, like image filtering and image filling. Next, we proposed two algorithms and the corresponding hardware implementations for region proposal. One is the serial operation based EEDRP approach, achieving low energy consumption by the strategies of near memory computing and response only on edge events. Another is the parallel operation based APBRP approach, leveraging in-memory computing to achieve significant energy and latency reduction. In the end, we explored a time-based computing technique and proposed a novel energy- and area-efficient neuron oscillator for robust neurocomputing application. This chapter presents a

conclusion to the thesis, a summary of our work is listed below and followed by an outlook on potential future directions.

6.1 Summary

6.1.1 In-memory Computing based Binary Image Denoising and Filling for Neuromorphic Vision Sensor Applications

In this work, we presented a charge mode in-memory computing (IMC) technique based on collocated SRAM and DRAM (CRAM) architecture for image/video preprocessing in the application of traffic surveillance. Bio-inspired neuromorphic vision sensors capture the vitality of a scene, mitigating data redundancy and latency, but they suffer from noise and holes, making the image rough. Noise and holes in the image not only increase the complexity of computation but also pose a challenge to object detection and classification. This work is the first exploration of global parallel computation for image pre-processing, including image filtering and image filling. A $9T$ bit-cell compact CRAM structure together with a 2-dimensional charge diffusion paths topology is proposed to store the input binary image data and compute locally. During the in memory computing mode for image filtering and region filling, the CRAM array is modeled as a 2-dimensional RC network, where the charge stored in each cell can be diffused and redistributed freely according to Ohm's law. The process of charge sharing is configured by some programmable parameters, such as equivalent resistance, diffusion level and duration, to make it adapt for various application scenarios. The implementation of circuit was fabricated using a 65 nm CMOS technology, which have successfully demonstrated the in-memory computing based image filtering and filling with a 320×240 CRAM array. The prototype structure exhibits error-free read and write operations with V_{DD} down to 400 mV and frequency up to 1 GHz. The energy efficiency is 233 TOPS/W at the normal operating frequency of 200 MHz and normal supply voltage of 1.2 V. The proposed in-memory parallel approach of image filtering achieves $\sim 10000\times$ energy savings and is $864\times$ faster compared to the digital counterpart of conventional non-IMC approach in the same process.

6.1.2 Near-/In-Memory Computing based Region Proposal Approaches

This work explores further of the CRAM structure to implement region proposal, which finds out all the objects in the frame and covers each of them with a bounding box. Region proposal is vital for object detection and classification to reduce the computation of the neural network (NN) hardware by computing regions of interests only instead of the whole image. We propose two different region proposal algorithms and the corresponding hardware implementations. The first one is the edge event driven region proposal (EDDRP), which is energy efficient by responding only to edge event during image reading. The proposed EEDRP approach achieves $\sim 2.9\times$ and faster and occupies $\sim 5.6\times$ less computing memory resources than the conventional connected component labeling (CCL) implementation. Though it is both energy and time efficient compared to the CCLRP approach, it still takes significant processing time due to the serial operation. Hence, we went on to propose the axes projection based region proposal (APBRP), which is in-memory computing based parallel operation leveraging the CRAM structure. In a typical application, the whole processing only requires two or three projections, further reducing the energy dissipation and execution time. It achieves $\sim 1767\times$ faster than the CCLRP implementation for calculating the region of interests of 15 objects, thanks to the parallel computation approach.

6.1.3 Time based Computing with Neuron Oscillator

In this work, a time based computing technique is explored, to be specific, a novel energy- and area-efficient neuron oscillator is proposed for robust neurocomputing application. By eliminating the unnecessary transistors, the proposed structure is composed of a simplest dynamic positive feedback latch and differential pairs, saving $\sim 25\%$ in size. The current controlled oscillator (CCO) can be tuned by both input voltage and external variable resistor. The measurement results show it achieves 11% frequency improvement and 13% energy-efficient without degrading the jitter and phase noise characteristics. Simulations of both ANN and CNN with the proposed CCO characteristics as the neuron transfer curve show no degradation in accuracy if the CCO nonlinearity is included in training.

6.2 Future Work

6.2.1 Improvement of Image Denoising and Filling

In Chapter 3, we presented a CRAM based in-memory image processing to perform image denoising and filling. The CRAM array is modeled as a 2-dimensional RC network for charge sharing and redistribution. After that, a sensor is required to recover the analog voltage of each cell to the digital. In our current design, we directly reuse the first inverter of the CMOS latch as the sensor. This makes the bit cell compact and area efficient, but causes the trade-off between denoising effect and filling effect. As we mentioned, the trip voltage of the inverter affect the overall effect – higher trip voltage leads to severe denoising effect and will remove some useful object pixels while lower trip voltage leads to over-filling effect and may leave some remnant of noise. Though we presented a solution with multiple diffusion-recover strategy, another possible and better solution is to use the dynamic threshold voltage for improvement both denoising and filling effect. A high threshold voltage could be used in the first diffusion-recover phase to remove noise, and then a low threshold voltage could be set during the second diffusion-recover phase to fill up holes.

6.2.2 In Memory Computing based MAC implementation

In this work, CRAM cell is used for in memory computing based image denoising and image filling in Chapter 3 and in memory computing based region proposal in Chapter 4. The proposed CRAM can also perform the current mode multiply and accumulate (MAC) operation, similar to [162], [163], [164]. The binary weights can be stored in the CRAM bit cells, and a binary input can be applied through either direction of the projection lines, finally the other direction of projection lines will calculate the dot-product of inputs and weights. MAC is a common operation that computes the product of two numbers and then accumulates the product. This can be used in energy efficient mixed signal processors for data-intensive application.

6.2.3 Hardware Implementation of Binary Neural Networks (BNNs)

In the application of event based binary image, we have proposed CRAM based parallel operation for image denoising and region proposal, which reduces the number of computations in classification task. After proposing regions of interests, these regions are fed into a neural network (NN) to execute the subsequent processing, like feature extraction, object classification, etc. However, the implementation of object classification in a microcontroller unit (MCU) or field-programmable gate array (FPGA) dissipates the majority of the energy of the system. There has been considerable progress in implementing BNN, exploiting SRAM cells [55], [93], which reduces the data movement and energy dissipation. Therefore, future work can explore the hardware implementation of BNN or DNN by manipulating the CRAM cells.

6.2.4 Self-adaptive Design

The current design includes many programmable parameters to configure the charge diffusion and region proposal approaches, which provide more flexibility and make the prototype suitable for different application scenarios. Future work can focus on how to adjust these parameters automatically by introducing feedback loops or dedicated algorithms. For example, the effect of denoising heavily depends on the diffusion resistance, which varies with temperature. The gate voltage of diffusion resistance could be controlled by a negative feedback control circuit to make the resistance independent of temperature to a first order. Besides, a dedicated algorithm can be designed to tune the region proposal parameters $SLOT\_X$ and $SLOT\_Y$ according to the details of the captured scene.

List of Author's Publications

Journal Articles

- **X. Zhang**, J. Acharya and A. Basu, "A 0.11–0.38 pJ/cycle Differential Ring Oscillator in 65 nm CMOS for Robust Neurocomputing," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 2, pp. 617-630, Feb 2021.

- **X. Zhang**, V. Mohan and A. Basu, "CRAM: Collocated SRAM and DRAM With In-Memory Computing-Based Denoising and Filling for Neuromorphic Vision Sensors in 65 nm CMOS," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 5, pp. 816-820, May 2020.

Conference Proceedings

- **X. Zhang**, V. Mohan and A. Basu, "CRAM: Collocated SRAM and DRAM With In-Memory Computing-Based Denoising and Filling for Neuromorphic Vision Sensors in 65 nm CMOS," 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Sevilla, 2020. *(Invited to TCAS-II as one of the best papers in ISCAS 2020)*

- Q. Chen, C. C. Boon, **X. Zhang**, C. Li, Y. Liang, Z. Liu, and T. Guo, "Multi-Channel FSK Inter/Intra-Chip Communication by Exploiting Field-Confined Slow-Wave Transmission Line," 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Sevilla, 2020, pp. 1-5.

- **X. Zhang**, and A. Basu, "A 915-1220 TOPS/W Hybrid In-Memory Computing based Image Restoration and Region Proposal Integrated Circuit for

Neuromorphic Vision Sensors in 65nm CMOS", 2022 IEEE Custom Integrated Circuits Conference (CICC) *submitted*

- A. Basu, C. Frenkel, L. Deng, and **X. Zhang**, "Spiking Neural Network Integrated Circuits: A Review of Trends and Future Directions", 2022 IEEE Custom Integrated Circuits Conference (CICC) *submitted*

Bibliography

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 1

[2] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015. 1

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 1

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[7] Avishek Biswas and Anantha P Chandrakasan. CONV-SRAM: An Energy-Efficient SRAM With In-Memory Dot-Product Computation for Low-Power Convolutional Neural Networks. *IEEE Journal of Solid-State Circuits*, 54(1): 217–230, 2018. 2, 4, 19, 21

[8] Jennifer Hasler and Harry Bo Marr. Finding a roadmap to achieve large neuromorphic hardware systems. *Frontiers in neuroscience*, 7:118, 2013. 2, 3

[9] Karl Berggren, Qiangfei Xia, Konstantin K Likharev, Dmitri B Strukov, Hao Jiang, Thomas Mikolajick, Damien Querlioz, Martin Salinga, John R Erickson, Shuang Pi, et al. Roadmap on emerging hardware and technology for machine learning. *Nanotechnology*, 32(1):012002, 2020. 3

[10] Youhui Zhang, Peng Qu, Yu Ji, Weihao Zhang, Guangrong Gao, Guanrui Wang, Sen Song, Guoqi Li, Wenguang Chen, Weimin Zheng, et al. A system hierarchy for brain-inspired computing. *Nature*, 586(7829):378–384, 2020.

[11] Oliver Rhodes. Brain-inspired computing boosted by new concept of completeness, 2020.

[12] Sumon Kumar Bose, Jyotibdha Acharya, and Arindam Basu. Is my Neural Network Neuromorphic? Taxonomy, Recent Trends and Future Directions in Neuromorphic Engineering. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 1522–1527, 2019. doi: 10.1109/IEEECONF44664.2019.9048891. 3, 4

[13] H. Morimura, S. Shigematsu, and K. Machida. A novel sensor cell architecture and sensing circuit scheme for capacitive fingerprint sensors. *IEEE Journal of Solid-State Circuits*, 35(5):724–731, 2000. doi: 10.1109/4.841500. 3

[14] Kian Ann Ng and Yong Ping Xu. A Low-Power, High CMRR Neural Amplifier System Employing CMOS Inverter-Based OTAs With CMFB Through Supply Rails. *IEEE Journal of Solid-State Circuits*, 51(3):724–737, 2016. doi: 10.1109/JSSC.2015.2512935. 3

[15] Hui Wang, Xiaoyang Wang, Abbas Barfidokht, Jiwoong Park, Joseph Wang, and Patrick P. Mercier. A Battery-Powered Wireless Ion Sensing System Consuming 5.5 nW of Average Power. *IEEE Journal of Solid-State Circuits*, 53(7):2043–2053, 2018. doi: 10.1109/JSSC.2018.2815657. 4

[16] Inhee Lee, Yoonmyung Lee, Dennis Sylvester, and David Blaauw. Battery Voltage Supervisors for Miniature IoT Systems. *IEEE Journal of Solid-State Circuits*, 51(11):2743–2756, 2016. doi: 10.1109/JSSC.2016.2600565. 4

[17] Carver Mead. How we created neuromorphic engineering. *Nature Electronics*, 3(7):434–435, 2020. 4

[18] Giacomo Indiveri and Rodney Douglas. Neuromorphic vision sensors. *Science*, 288(5469):1189–1190, 2000. 4

[19] Anup Vanarse, Adam Osseiran, and Alexander Rassau. A review of current neuromorphic approaches for vision, auditory, and olfactory sensors. *Frontiers in neuroscience*, 10:115, 2016.

[20] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128× 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, 2008. doi: 10.1109/JSSC.2007.914337. 4, 7, 13

[21] Shih-Chii Liu and Tobi Delbruck. Neuromorphic sensory systems. *Current opinion in neurobiology*, 20(3):288–295, 2010. 4

[22] Soumyajit Mandal, Serhii M. Zhak, and Rahul Sarpeshkar. A Bio-Inspired Active Radio-Frequency Silicon Cochlea. *IEEE Journal of Solid-State Circuits*, 44(6):1814–1828, 2009. doi: 10.1109/JSSC.2009.2020465.

[23] Minhao Yang, Chen-Han Chien, Tobi Delbruck, and Shih-Chii Liu. A 0.5 V 55 μW 64 $\times$ 2 Channel Binaural Silicon Cochlea for Event-Driven Stereo-Audio Sensing. *IEEE Journal of Solid-State Circuits*, 51(11):2554–2569, 2016. doi: 10.1109/JSSC.2016.2604285. 4

[24] C. Yu, T. Yoo, T. T. Kim, K. C. Tshun Chuan, and B. Kim. A 16K Current-Based 8T SRAM Compute-In-Memory Macro with Decoupled Read/Write and 1-5bit Column ADC. In *2020 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–4, 2020. 4, 19, 20, 22

[25] Steve B Furber, Francesco Galluppi, Steve Temple, and Luis A Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014. 4

[26] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014. 4

[27] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018. 4

[28] James B Aimone. A Roadmap for Reaching the Potential of Brain-Derived Computing. *Advanced Intelligent Systems*, 3(1):2000191, 2021. 5

[29] Thomas F Schranghamer, Aaryan Oberoi, and Saptarshi Das. Graphene memristive synapses for high precision neuromorphic computing. *Nature communications*, 11(1):1–11, 2020. 5

[30] Irem Boybat, Manuel Le Gallo, SR Nandakumar, Timoleon Moraitis, Thomas Parnell, Tomas Tuma, Bipin Rajendran, Yusuf Leblebici, Abu Sebastian, and Evangelos Eleftheriou. Neuromorphic computing with multi-memristive synapses. *Nature communications*, 9(1):1–12, 2018.

[31] Buyun Chen, Hao Yang, Boxiang Song, Deming Meng, Xiaodong Yan, Yuanrui Li, Yunxiang Wang, Pan Hu, Tse-Hsien Ou, Mark Barnell, et al. A memristor-based hybrid analog-digital computing platform for mobile robotics. *Science Robotics*, 5(47), 2020.

[32] Elliot J Fuller, Scott T Keene, Armantas Melianas, Zhongrui Wang, Sapan Agarwal, Yiyang Li, Yaakov Tuchman, Conrad D James, Matthew J Marinella, J Joshua Yang, et al. Parallel programming of an ionic floating-gate memory array for scalable neuromorphic computing. *Science*, 364(6440):570–574, 2019. 5

[33] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784): 607–617, 2019. 5

[34] Yves Frégnac. Big data and the industrialization of neuroscience: A safe roadmap for understanding the brain? *Science*, 358(6362):470–477, 2017. 5

[35] Anders Sandberg. Energetics of the Brain and AI. *arXiv preprint arXiv:1602.04019*, 2016. 5

[36] Chang Wen Chen. Internet of Video Things: Next-Generation IoT With Visual Sensors. *IEEE Internet of Things Journal*, 7(8):6676–6685, 2020. 5

[37] Thomas J Bruns, Raymond J O'Connell, Jeffrey S Wells, and Mark Dapper. Unattended ground sensor (UGS) systems for homeland defense. In *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Defense and Law Enforcement II*, volume 5071, pages 280–288. International Society for Optics and Photonics, 2003. 6

[38] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *arXiv preprint arXiv:1904.08405*, 2019. 7, 12, 14, 15

[39] Deepak Singla, Vivek Mohan, Tarun Pulluri, Andres Ussa, Bharath Ramesh, and Arindam Basu. EBBINNOT: A Hardware Efficient Hybrid Event-Frame Tracker for Stationary Neuromorphic Vision Sensors. *arXiv preprint arXiv:2006.00422*, 2020. 7, 26, 52, 53, 73, 93, 98

[40] Yuji Nozaki and Tobi Delbruck. Temperature and Parasitic Photocurrent Effects in Dynamic Vision Sensors. *IEEE Transactions on Electron Devices*, 64(8):3239–3245, 2017. doi: 10.1109/TED.2017.2717848. 7

[41] Thomas Finateu, Atsumi Niwa, Daniel Matolin, Koya Tsuchimoto, Andrea Mascheroni, Etienne Reynaud, Pooria Mostafalu, Frederick Brady, Ludovic Chotard, Florian LeGoff, Hirotsugu Takahashi, Hayato Wakabayashi, Yusuke Oike, and Christoph Posch. 5.10 A 1280×720 Back-Illuminated Stacked Temporal Contrast Event-Based Vision Sensor with 4.86µm Pixels, 1.066GEPS Readout, Programmable Event-Rate Controller and Compressive Data-Formatting Pipeline. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, pages 112–114, 2020. doi: 10.1109/ISSCC19947.2020.9063149.

[42] Tobi Delbruck, Yuhuang Hu, and Zhe He. V2E: From video frames to realistic DVS event camera streams. *arXiv e-prints*, pages arXiv–2006, 2020.

[43] Chenghan Li, Luca Longinotti, Federico Corradi, and Tobi Delbruck. A 132 by 104 10m-Pixel 250W 1kefps Dynamic Vision Sensor with Pixel-Parallel Noise and Spatial Redundancy Suppression. In *2019 Symposium on VLSI Circuits*, pages C216–C217, 2019. doi: 10.23919/VLSIC.2019.8778050. 7

[44] Jyotibdha Acharya, Andres Ussa Caycedo, Vandana Reddy Padala, Rishi Raj Singh Sidhu, Garrick Orchard, Bharath Ramesh, and Arindam Basu. EBBIOT: A Low-complexity Tracking Algorithm for Surveillance in IoVT

Using Stationary Neuromorphic Vision Sensors. In *2019 32nd IEEE International System-on-Chip Conference (SOCC)*, pages 318–323. IEEE, 2019. 8, 22, 26, 33, 47, 48, 52, 73, 88

[45] Xueyong Zhang, Vivek Mohan, and Arindam Basu. CRAM: Collocated SRAM and DRAM With In-Memory Computing-Based Denoising and Filling for Neuromorphic Vision Sensors in 65 nm CMOS. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(5):816–820, 2020. 8, 65

[46] Xueyong Zhang, Jyotibdha Acharya, and Arindam Basu. A 0.11-0.38 pJ/cycle Differential Ring Oscillator in 65 nm CMOS for Robust Neurocomputing. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2020. 8

[47] Christoph Posch, Teresa Serrano-Gotarredona, Bernabe Linares-Barranco, and Tobi Delbruck. Retinomorphic event-based vision sensors: bioinspired cameras with spiking output. *Proceedings of the IEEE*, 102(10):1470–1484, 2014. 13, 14, 33

[48] John Lazzaro and John Wawrzynek. A multi-sender asynchronous extension to the AER protocol. In *Proceedings Sixteenth Conference on Advanced Research in VLSI*, pages 158–169. IEEE, 1995. 13

[49] User Guide - DAVIS240, inivation. https://inivation.github.io/inivation-docs/Hardware%20user%20guidesUser_guide_-_DAVIS240.html. 15, 53, 62

[50] Giulia Santoro, Giovanna Turvani, and Mariagrazia Graziano. New logic-in-memory paradigms: An architectural and technological perspective. *Micromachines*, 10(6):368, 2019. 15, 17, 18

[51] Abu Sebastian, Manuel Le Gallo, Riduan Khaddam-Aljameh, and Evangelos Eleftheriou. Memory devices and applications for in-memory computing. *Nature nanotechnology*, 15(7):529–544, 2020. 16

[52] A. Basu, J. Acharya, T. Karnik, and et. al. Low-Power, Adaptive Neuromorphic Systems: Recent Progress and Future Directions. *IEEE Journal on Emerging Topics in Circuits and Systems*, 8(1):6–27, 2018. 17

[53] Gagandeep Singh, Lorenzo Chelini, Stefano Corda, Ahsan Javed Awan, Sander Stuijk, Roel Jordans, Henk Corporaal, and Albert-Jan Boonstra. Near-memory computing: Past, present, and future. *Microprocessors and Microsystems*, 71: 102868, 2019. 17

[54] Shihui Yin, Zhewei Jiang, Jae-Sun Seo, and Mingoo Seok. XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks. *IEEE Journal of Solid-State Circuits*, 2020. 18, 101

[55] Hossein Valavi, Peter J Ramadge, Eric Nestler, and Naveen Verma. A 64-tile 2.4-Mb in-memory-computing CNN accelerator employing charge-domain compute. *IEEE Journal of Solid-State Circuits*, 54(6):1789–1799, 2019. 19, 21, 22, 137

[56] Jintao Zhang, Zhuo Wang, and Naveen Verma. In-memory computation of a machine-learning classifier in a standard 6T SRAM array. *IEEE Journal of Solid-State Circuits*, 52(4):915–924, 2017. 20, 22

[57] Sujan Kumar Gonugondla, Mingu Kang, and Naresh Shanbhag. A 42pJ/decision 3.12 TOPS/W robust in-memory machine learning classifier with on-chip training. In *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 490–492. IEEE, 2018. 20, 22

[58] Xin Si, Jia-Jing Chen, Yung-Ning Tu, Wei-Hsing Huang, Jing-Hong Wang, Yen-Cheng Chiu, Wei-Chen Wei, Ssu-Yen Wu, Xiaoyu Sun, Rui Liu, et al. A twin-8T SRAM computation-in-memory macro for multiple-bit CNN-based machine learning. In *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 396–398. IEEE, 2019. 22, 50, 101

[59] X. Si, W. Khwa, J. Chen, J. Li, X. Sun, R. Liu, S. Yu, H. Yamauchi, Q. Li, and M. Chang. A Dual-Split 6T SRAM-Based Computing-in-Memory Unit-Macro With Fully Parallel Product-Sum Operation for Binarized DNN Edge Processors. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(11):4172–4185, 2019. 22, 101

[60] K. Ando, K. Ueyoshi, K. Orimo, H. Yonekawa, S. Sato, H. Nakahara, S. Takamaeda-Yamazaki, M. Ikebe, T. Asai, T. Kuroda, and M. Motomura. 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*, 53(4):983–994, 2018. 22, 101

[61] Jinseok Kim, Jongeun Koo, Taesu Kim, Yulhwa Kim, Hyungjun Kim, Seunghyun Yoo, and Jae-Joon Kim. Area-efficient and variation-tolerant in-memory BNN computing using 6T SRAM array. In *2019 Symposium on VLSI Circuits*, pages C118–C119. IEEE, 2019. 22, 101

[62] A. Biswas and A. P. Chandrakasan. CONV-SRAM: An Energy-Efficient SRAM With In-Memory Dot-Product Computation for Low-Power Convolutional Neural Networks. *IEEE Journal of Solid-State Circuits*, 54(1):217–230, 2019. 22, 101

[63] Z. Jiang, S. Yin, J. Seo, and M. Seok. C3SRAM: In-Memory-Computing SRAM Macro Based on Capacitive-Coupling Computing. *IEEE Solid-State Circuits Letters*, 2(9):131–134, 2019. 22

[64] S. Yin, Z. Jiang, J. Seo, and M. Seok. XNOR-SRAM: In-Memory Computing SRAM Macro for Binary/Ternary Deep Neural Networks. *IEEE Journal of Solid-State Circuits*, 55(6):1733–1743, 2020. 22

[65] J. Yang, Y. Kong, Z. Wang, Y. Liu, B. Wang, S. Yin, and L. Shi. Sandwich-RAM: An Energy-Efficient In-Memory BWN Architecture with Pulse-Width Modulation. In *2019 IEEE International Solid- State Circuits Conference -(ISSCC)*, pages 394–396, 2019. 22

[66] J. Su, X. Si, Y. Chou, T. Chang, W. Huang, Y. Tu, R. Liu, P. Lu, T. Liu, J. Wang, Z. Zhang, H. Jiang, S. Huang, C. Lo, R. Liu, C. Hsieh, K. Tang, S. Sheu, S. Li, H. Lee, S. Chang, S. Yu, and M. Chang. 15.2 A 28nm 64Kb Inference-Training Two-Way Transpose Multibit 6T SRAM Compute-in-Memory Macro for AI Edge Chips. In *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, pages 240–242, 2020. 22

[67] Q. Dong, M. E. Sinangil, B. Erbagci, D. Sun, W. Khwa, H. Liao, Y. Wang, and J. Chang. 15.3 A 351TOPS/W and 372.4GOPS Compute-in-Memory SRAM Macro in 7nm FinFET CMOS for Machine-Learning Applications. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, pages 242–244, 2020. 22

[68] X. Si, Y. Tu, W. Huang, J. Su, P. Lu, J. Wang, T. Liu, S. Wu, R. Liu, Y. Chou, Z. Zhang, S. Sie, W. Wei, Y. Lo, T. Wen, T. Hsu, Y. Chen, W. Shih, C. Lo, R. Liu, C. Hsieh, K. Tang, N. Lien, W. Shih, Y. He, Q. Li, and M. Chang. 15.5 A 28nm 64Kb 6T SRAM Computing-in-Memory Macro with 8b MAC Operation for AI Edge Chips. In *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, pages 246–248, 2020. 22

[69] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 21, 22, 53

[70] Miroslaw Jablonski and Marek Gorgon. Handel-C implementation of classical component labelling algorithm. In *Euromicro Symposium on Digital System Design, 2004. DSD 2004.*, pages 387–393. IEEE, 2004. 22

[71] Hugo Hedberg, Fredrik Kristensen, and Viktor Owall. Implementation of a labeling algorithm based on contour tracing with feature extraction. In *2007 IEEE International Symposium on Circuits and Systems*, pages 1101–1104. IEEE, 2007. 22

[72] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016. 22, 25

[73] Lifeng He, Xiwei Ren, Qihang Gao, Xiao Zhao, Bin Yao, and Yuyan Chao. The connected-component labeling problem: A review of state-of-the-art algorithms. *Pattern Recognition*, 70:25–43, 2017. 22

[74] Kesheng Wu, Ekow Otoo, and Kenji Suzuki. Two strategies to speed up connected component labeling algorithms. Technical report, Ernest Orlando Lawrence Berkeley National Laboratory, Berkeley, CA (US), 2005. 23

[75] Fu Chang, Chun-Jen Chen, and Chi-Jen Lu. A linear-time component-labeling algorithm using contour tracing technique. *computer vision and image understanding*, 93(2):206–220, 2004. 24

[76] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 24

[77] Hanspeter Schmid. Why 'current mode' does not guarantee good performance. *Analog Integrated Circuits and Signal Processing*, 35(1):79–90, 2003. 27

[78] Mohammad Maymandi-Nejad and Manoj Sachdev. A digitally programmable delay element: design and analysis. *IEEE transactions on very large scale integration (VLSI) systems*, 11(5):871–878, 2003. 27, 28

[79] Yongsam Moon, Jongsang Choi, Kyeongho Lee, Deog-Kyoon Jeong, and Min-Kyu Kim. An all-analog multiphase delay-locked loop using a replica delay line for wide-range operation and low-jitter performance. *IEEE Journal of Solid-State Circuits*, 35(3):377–384, 2000. 27, 28

[80] Martin Saint-Laurent and Madhavan Swaminathan. A digitally adjustable resistor for path delay characterization in high-frequency microprocessors. In *2001 Southwest Symposium on Mixed-Signal Design (Cat. No. 01EX475)*, pages 61–64. IEEE, 2001. 27, 29

[81] Gyorgy Csaba and Wolfgang Porod. Coupled oscillators for computing: A review and perspective. *Applied Physics Reviews*, 7(1):011302, 2020. 30

[82] Inhee Lee, Dennis Sylvester, and David Blaauw. A constant energy-per-cycle ring oscillator over a wide frequency range for wireless sensor nodes. *IEEE journal of solid-state circuits*, 51(3):697–711, 2016. 30

[83] Kyu-hyoun Kim, Young-Soo Sohn, Chan-Kyoung Kim, Moonsook Park, Dong-Jin Lee, Woo-Seop Kim, and Changhyun Kim. A 20-Gb/s 256-Mb DRAM with an inductorless quadrature PLL and a cascaded pre-emphasis transmitter. *IEEE Journal of solid-state circuits*, 41(1):127–134, 2005. 31

[84] Jr Thomas Barnett, Shruti Jain, Usha Andra, and Taru Khurana. Cisco Visual Networking Index (VNI) Complete Forecast Update, 2017-2022. [online] Available: https://www.cisco.com/c/dam/m/en_us/network-intelligence/service-provider/digital-transformation/knowledge-network-webinars/pdfs/1211_BUSINESS_SERVICES_CKN_PDF.pdf, 2018. 33

[85] Anup Mohan, Kent Gauen, Yung-Hsiang Lu, Wei Wayne Li, and Xuemin Chen. Internet of video things in 2030: A world with many cameras. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2017. 33

[86] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240× 180 130 db 3 μs latency global shutter spatio temporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014. 33, 47, 62

[87] Ming-Hsien Tu, Jihi-Yu Lin, Ming-Chien Tsai, Chien-Yu Lu, Yuh-Jiun Lin, Meng-Hsueh Wang, Huan-Shun Huang, Kuen-Di Lee, Wei-Chiang Shih, Shyh-Jye Jou, et al. A single-ended disturb-free 9T subthreshold SRAM with cross-point data-aware write word-line structure, negative bit-line, and adaptive read operation timing tracing. *IEEE Journal of Solid-State Circuits*, 47(6): 1469–1482, 2012. 38

[88] Ming-Hung Chang, Yi-Te Chiu, and Wei Hwang. Design and Iso-Area V_{min} Analysis of 9T Subthreshold SRAM with Bit-Interleaving Scheme in 65-nm CMOS. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 59 (7):429–433, 2012. 38

[89] V. R. Padala, A. Basu, and G. Orchard. A Noise Filtering Algorithm for Event-Based Asynchronous Change detection Image Sensors and its Implementation on TrueNorth. *Frontiers in Neuroscience*, 12:118, 2018. 44

[90] Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 52(1):127–138, 2016. 48

[91] Mingu Kang, Sujan K Gonugondla, Ameya Patil, and Naresh R Shanbhag. A multi-functional in-memory inference processor using a standard 6T SRAM array. *IEEE Journal of Solid-State Circuits*, 53(2):642–655, 2018. 48, 49, 50

[92] Hongjie Liu, Christian Brandli, Chenghan Li, Shih-Chii Liu, and Tobi Delbruck. Design of a spatio temporal correlation filter for event-based sensors. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 722–725. IEEE, 2015. 48, 49

[93] Avishek Biswas and Anantha 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)*, pages 488–490. IEEE, 2018. 50, 137

[94] Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS. *IEEE Journal of Solid-State Circuits*, 46 (1):259–275, 2010. 62

[95] Koichiro Ishibashi and Kenichi Osada. *Low power and reliable SRAM memory cell and array design*, volume 31. Springer Science & Business Media, 2011. 63

[96] Christoph H Lampert, Matthew B Blaschko, and Thomas Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *IEEE transactions on pattern analysis and machine intelligence*, 31(12):2129–2142, 2009. 73

[97] Rafael C Gonzalez, Richard E Woods, et al. *Digital Image Processing*. Prentice hall Upper Saddle River, NJ, 2002. 73, 85

[98] S Paul, T Majumder, C Augustine, AF Malavasi, S Usirikayala, R Kumar, J Kollikunnel, S Chhabra, S Yada, ML Barajas, et al. A 0.05 pJ/Pixel 70fps FHD 1Meps Event-Driven Visual Data Processing Unit. In *2020 IEEE Symposium on VLSI Circuits*, pages 1–2. IEEE, 2020. 85, 100, 101

[99] Robert Walczyk, Alistair Armitage, and T David Binnie. Comparative study on connected component labeling algorithms for embedded video processing systems. *IPCV*, 10:176, 2010. 87

[100] Longyang Lin, Saurabh Jain, and Massimo Alioto. A 595pW 14pJ/cycle microcontroller with dual-mode standard cells and self-startup for battery-indifferent distributed sensing. In *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 44–46. IEEE, 2018. 91

[101] Rafael Padilla, Wesley L Passos, Thadeu LB Dias, Sergio L Netto, and Eduardo AB da Silva. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, 10(3):279, 2021. 96

[102] David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020. 96

[103] Cyril Goutte and Eric Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *European conference on information retrieval*, pages 345–359. Springer, 2005. 96

[104] Dataset: "https://zenodo.org/record/3839231". 96

[105] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. DeepFace: Closing the gap to human-level performance in face verification. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1701–08, 2014. 104

[106] L. Deng and et. al. Recent advances in deep learning for speech research at Microsoft. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8604–08, 2013. 104

[107] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *J. Machine Learning Research*, 12:2493–2537, 2011. 104

[108] V. M. Suresh and et. al. Powering the IoT through embedded machine learning and LoRa. In *IEEE World Forum on IoT (WF-IOT)*, pages 349–354, 2018. 104

[109] A. Basu, Y. Chen, and E. Yao. Big Data Management in Neural Implants: The Neuromorphic Approach. In Chattopadhyay A. and Chang C.and Yu H, editors, *Emerging Technology and Architecture for Big-data Analytics*. Springer, Cham, 2017. 104

[110] Xin Si, Yung-Ning Tu, Wei-Hsing Huang, Jian-Wei Su, Pei-Jung Lu, Jing-Hong Wang, Ta-Wei Liu, Ssu-Yen Wu, Ruhui Liu, Yen-Chi Chou, et al. A Local Computing Cell and 6T SRAM-Based Computing-in-Memory Macro With 8-b MAC Operation for Edge AI Chips. *IEEE Journal of Solid-State Circuits*, 2021. 104

[111] Jian-Wei Su, Yen-Chi Chou, Ruhui Liu, Ta-Wei Liu, Pei-Jung Lu, Ping-Chun Wu, Yen-Lin Chung, Li-Yang Hung, Jin-Sheng Ren, Tianlong Pan, et al. A 28nm 384kb 6T-SRAM Computation-in-Memory Macro with 8b Precision for AI Edge Chips. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 64, pages 250–252. IEEE, 2021.

[112] Jingcheng Wang, Xiaowei Wang, Charles Eckert, Arun Subramaniyan, Reetuparna Das, David Blaauw, and Dennis Sylvester. A 28-nm compute SRAM with bit-serial logic/arithmetic operations for programmable in-memory vector computing. *IEEE Journal of Solid-State Circuits*, 55(1):76–86, 2019.

[113] Mustafa Ali, Akhilesh Jaiswal, Sangamesh Kodge, Amogh Agrawal, Indranil Chakraborty, and Kaushik Roy. IMAC: In-memory multi-bit multiplication and ACcumulation in 6T SRAM array. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(8):2521–2531, 2020. 104

[114] Cheng-Xin Xue, Yen-Cheng Chiu, Ta-Wei Liu, Tsung-Yuan Huang, Je-Syu Liu, Ting-Wei Chang, Hui-Yao Kao, Jing-Hong Wang, Shih-Ying Wei, Chun-Ying Lee, et al. A CMOS-integrated compute-in-memory macro based on resistive random-access memory for AI edge devices. *Nature Electronics*, 4(1): 81–90, 2021. 104

[115] Cheng-Xin Xue, Je-Min Hung, Hui-Yao Kao, Yen-Hsiang Huang, Sheng-Po Huang, Fu-Chun Chang, Peng Chen, Ta-Wei Liu, Chuan-Jia Jhang, Chin-I Su, et al. A 22nm 4Mb 8b-Precision ReRAM Computing-in-Memory Macro with 11.91 to 195.7 TOPS/W for Tiny AI Edge Devices. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 64, pages 245–247. IEEE, 2021.

[116] F Merrikh Bayat, Mirko Prezioso, Bhaswar Chakrabarti, H Nili, I Kataeva, and D Strukov. Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits. *Nature communications*, 9(1): 1–7, 2018. 104

[117] Farnood Merrikh-Bayat, Xinjie Guo, Michael Klachko, Mirko Prezioso, Konstantin K Likharev, and Dmitri B Strukov. High-performance mixed-signal neurocomputing with nanoscale floating-gate memory cell arrays. *IEEE transactions on neural networks and learning systems*, 29(10):4782–4790, 2017. 104

[118] M Reza Mahmoodi and Dmitri Strukov. An ultra-low energy internally analog, externally digital vector-matrix multiplier based on NOR flash memory

technology. In *Proceedings of the 55th Annual Design Automation Conference*, pages 1–6, 2018.

[119] L Danial, V Gupta, E Pikhay, Y Roizin, and S Kvatinsky. Modeling a floating-gate memristive device for computer aided design of neuromorphic computing. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 472–477. IEEE, 2020. 104

[120] A. Basu and et. al. Low-Power, Adaptive Neuromorphic Systems: Recent Progress and Future Directions. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, 8(1):6–27, 2018. 105

[121] R. Sarpeshkar. Analog versus digital: extrapolating from electronics to neurobiology. *Neural Computation*, 10(7):1601–38, 1998. 105

[122] M. Pfeiffer and T. Pfeil. Deep learning with spiking neurons: opportunities and challenges. *Frontiers in Neuroscience*, 12, 2018. 105

[123] B. Sahoo. Ring Oscillator Based Sub-1V Leaky Integrate-and-Fire Neuron Circuit. In *IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, 2017. 105, 130

[124] L. Everson, M. Liu, N. Pande, and C. H. Kim. A 104.8TOPS/W One-Shot Time-Based Neuromorphic Chip Employing Dynamic Threshold Error Correction in 65nm. In *IEEE Asian Solid-State Circuits Conference (ASSCC)*, 2018. 105

[125] Y. Chen, Z. Wang, A. Patil, and A. Basu. A 2.86-TOPS/W Current Mirror Cross-Bar Based Machine-Learning and Physical Unclonable Function Engine for Internet-of-Things Applications. *IEEE Trans. on Circuits and Systems-I*, 66(6):2240–52, 2019. 105, 117, 125, 130

[126] E. Yao and A. Basu. VLSI Extreme Learning Machine: A Design Space Exploration. *IEEE Trans. on VLSI*, 25(1):60–74, 2017. 117, 125, 127, 130

[127] A. Patil, S. Shen, E. Yao, and A. Basu. Hardware Architecture for Large Parallel Array of Random Feature Extractors applied to Image Recognition. *Neurocomputing*, 261:193–203, 2017. 105

[128] W. Bae, H. Ju, K. Park, S.-Y. Cho, and D.-K. Jeong. A 7.6 mW 414 fs RMS-jitter 10 GHz phase-locked loop for a 40 Gb/s serial link transmitter based on a two-stage ring oscillator in 65 nm CMOS. *IEEE J. Solid-State Circuits*, 51(10):2357–2367, 2016. 105, 107

[129] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going Deeper in Spiking Neural Networks: VGG and Residual Architectures. *Frontiers in neuroscience*, 13, 2019. 105

[130] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11: 682, 2017. 105

[131] Luke R Everson, Muqing Liu, Nakul Pande, and Chris H Kim. An energy-efficient one-shot time-based neural network accelerator employing dynamic threshold error correction in 65 nm. *IEEE Journal of Solid-State Circuits*, 54 (10):2777–2785, 2019. 105

[132] Daisuke Miyashita, Shouhei Kousai, Tomoya Suzuki, and Jun Deguchi. A neuromorphic chip optimized for deep learning and cmos technology with time-domain analog and digital mixed-signal processing. *IEEE Journal of Solid-State Circuits*, 52(10):2679–2689, 2017. 105

[133] A. Raychowdhury, A. Parihar, and et .al. Computing With Networks of Oscillatory Dynamical Systems. *Proc. of IEEE*, 107(1):73–89, 2019. 105

[134] S. Dutta, A. Parihar, and et .al. Programmable coupled oscillators for synchronized locomotion. *Nature Communications*, 10(3299), 2019. 105

[135] Ali Hajimiri, Sotirios Limotyrakis, and Thomas H Lee. Jitter and phase noise in ring oscillators. *IEEE Journal of Solid-state circuits*, 34(6):790–804, 1999. 107

[136] Piotr Dudek, Stanislaw Szczepanski, and John V Hatfield. A high-resolution CMOS time-to-digital converter utilizing a Vernier delay line. *IEEE Journal of Solid-State Circuits*, 35(2):240–247, 2000. 107

[137] A.R. Newton T. Sakurai. Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas. *IEEE J. Solid-State Circuits*, 25, 1990. 113

[138] Keith A Bowman, Blanca L Austin, John C Eble, Xinghai Tang, and James D Meindl. A physical alpha-power law MOSFET model. *IEEE Journal of Solid-State Circuits*, 34(10):1410–1414, 1999. 113

[139] A. A. Abidi. Phase noise and jitter in CMOS ring oscillators. *IEEE J. Solid-State Circuits*, 41(8):1803–1816, 2006. 116

[140] P. Yao, H. Wu, and et. al. Fully hardware-implemented memristor convolutional neural network. *Nature*, 577, 2020. 119

[141] Jeong-Jun Lee, Jungjin Park, Min-Woo Kwon, Sungmin Hwang, Hyungjin Kim, and Byung-Gook Park. Integrated neuron circuit for implementing neuromorphic system with synaptic device. *Solid-State Electronics*, 140:34–40, 2018. 125

[142] Arindam Basu and Paul E Hasler. Nullcline-based design of a silicon neuron. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57(11):2938–2947, 2010. 125

[143] Bibhu Datta Sahoo. Ring oscillator based sub-1V leaky integrate-and-fire neuron circuit. In *IEEE Symp. Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2017. 125

[144] Lieuwe B Leene and Timothy G Constandinou. A 0.006 mm^2 1.2 μW Analog-to-Time Converter for Asynchronous Bio-Sensors. *IEEE Journal of Solid-State Circuits*, 53(9):2604–2613, 2018. 126

[145] Shaolan Li, Abhishek Mukherjee, and Nan Sun. A 174.3-dB FoM VCO-Based CT $\Delta\Sigma$ Modulator With a Fully-Digital Phase Extended Quantizer and Tri-Level Resistor DAC in 130-nm CMOS. *IEEE Journal of Solid-State Circuits*, 52(7):1940–1952, 2017. 126

[146] Brian Young, Karthik Reddy, Sachin Rao, Amr Elshazly, Tejasvi Anand, and Pavan Kumar Hanumolu. A 75dB DR 50MHz BW 3^{rd} order CT-$\Delta\Sigma$ modulator using VCO-based integrators. In *2014 Symposium on VLSI Circuits Digest of Technical Papers*, pages 1–2. IEEE, 2014. 126

[147] Gerry Taylor and Ian Galton. A reconfigurable mostly-digital delta-sigma ADC with a worst-case FOM of 160 dB. *IEEE Journal of Solid-State Circuits*, 48(4):983–995, 2013. 126

[148] Yeomyung Kim and Tae Wook Kim. An 11 b 7 ps resolution two-step time-to-digital converter with 3-D Vernier space. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(8):2326–2336, 2014. 126

[149] Chih-Chan Tu, Yu-Kai Wang, and Tsung-Hsien Lin. A low-noise area-efficient chopped VCO-based CTDSM for sensor applications in 40-nm CMOS. *IEEE Journal of Solid-State Circuits*, 52(10):2523–2532, 2017. 126

[150] Akshay Jayaraj, Mohammadhadi Danesh, Sanjeev Tannirkulam Chandrasekaran, and Arindam Sanyal. Highly Digital Second-Order $\Delta\Sigma$ VCO ADC. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(7): 2415–2425, 2019. 126

[151] A. Jayaraj, A. Das, S. Arcot, and A. Sanyal. 8.6fJ/step VCO-Based CT 2nd-Order $\Delta\Sigma$ ADC. In *2019 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pages 197–200, 2019. 126, 127

[152] Y. Zhong, S. Li, A. Sanyal, X. Tang, L. Shen, S. Wu, and N. Sun. A Second-Order Purely VCO-Based CT $\Delta\Sigma$ ADC Using a Modified DPLL in 40-nm CMOS. In *2018 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pages 93–94, 2018. 126

[153] R. A. John and et. al. Optogenetics inspired transition metal dichalcogenide neuristors for in-memory deep recurrent neural networks. *Nature Communications*, 11(1):1–9, 2020. 127

[154] A. Bhaduri and et. al. Spiking neural classifier with lumped dendritic nonlinearity and binary synapses: a current mode VLSI implementation and analysis. *Neural Computation*, 30(3):723–60, 2018.

[155] S. Roy and A. Basu. An online unsupervised structural plasticity algorithm for spiking neural networks. *IEEE Trans. on Neural Networks and Learning Systems*, 28(4):900–910, 2016.

[156] D. Querlioz and et. al. Immunity to device variations in a spiking neural network with memristive nanodevices. *IEEE Trans. on Nanotechnology*, 12 (3):288–95, 2013. 127

[157] Y. LeCun, C. Cortes, and C. Burges. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`, 1998. 128

[158] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 128

[159] Alex Krizhevsky, Geoffrey Hinton, et al. *Learning Multiple Layers of Features from Tiny Images*. Citeseer, 2009. 128

[160] Hyeonwoo Noh, Tackgeun You, Jonghwan Mun, and Bohyung Han. Regularizing deep neural networks by noise: Its interpretation and optimization. In *Advances in Neural Information Processing Systems*, pages 5109–5118, 2017. 130

[161] G. Indiveri and et. al. Neuromorphic silicon neuron circuits. *Frontiers in Neuroscience*, 5, 2011. 132

[162] Chengshuo Yu, Taegeun Yoo, Tony Tae-Hyoung Kim, Kevin Chai Tshun Chuan, and Bongjin Kim. A 16K current-based 8T SRAM compute-in-memory macro with decoupled read/write and 1-5bit column ADC. In *2020 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–4. IEEE, 2020. 136

[163] Q. Dong, S. Jeloka, M. Saligane, Y. Kim, M. Kawaminami, A. Harada, S. Miyoshi, M. Yasuda, D. Blaauw, and D. Sylvester. A 4 + 2T SRAM for Searching and In-Memory Computing With 0.3-V V_{DDmin}. *IEEE Journal of Solid-State Circuits*, 53(4):1006–1015, 2018. doi: 10.1109/JSSC.2017.2776309. 136

[164] Amogh Agrawal, Akhilesh Jaiswal, Chankyu Lee, and Kaushik Roy. X-SRAM: Enabling in-memory Boolean computations in CMOS static random access memories. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65 (12):4219–4232, 2018. 136