UNIVERSITY OF CALIFORNIA SAN DIEGO

Energy-Efficient Event-Based Vision Sensors and Compute-in-Memory Architectures for Neuromorphic and Machine Learning Applications

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Medical Devices and Systems)

by

Rajkumar Chinnakonda Kubendran

Committee in charge:

Professor Gert Cauwenberghs, Chair
Professor Patrick Mercier, Co-Chair
Professor Henry D.I. Abarbanel
Professor Drew Hall
Professor Duygu Kuzum

2020

The dissertation of Rajkumar Chinnakonda Kubendran is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____
Co-Chair

_____
Chair

University of California San Diego

2020

DEDICATION

To my father Kubendran, mother Usha, step-father Hariharan,

wife Aishwarya, and son Krish:

who mean the world to me.

EPIGRAPH

Knowledge teaches humility, humility fosters character, character drives righteous action and righteous action brings true happiness.

—Hitopadesha

What I cannot create, I do not understand.

—Richard Feynman

If a machine is expected to be infallible, it cannot also be intelligent.

—Alan Turing

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

My heartfelt thanks and deepest gratitude goes to my advisor Dr. Gert Cauwenberghs, without whom this dissertation would not be possible. He has provided enormous support and guidance throughout my PhD research. I am constantly amazed by his incredible depth of knowledge in a wide variety of disciplines. His never-ending optimism and calm, positive attitude towards research provides me the energy to keep pushing forward even during rainy days. His admirable intellect, sincerity and generosity has been an inspiration for me to strive to be better both professionally and personally.

I would also like to express my appreciation to Prof. Patrick Mercier for his continuous support and numerous thought-provoking discussions. My thanks go to other members of my committee, Prof. Henry Abarbanel, Prof. Drew Hall, and Prof. Duygu Kuzum for their valuable inputs and guidance.

The members of the Integrated Systems Neuroengineering Lab - ISNL played a vital role throughout my PhD journey by being incredible pillars of support. It has been a privilege to work with talented researchers in the lab, whose warmth and encouragement even during hard times will always be remembered. I would like to thank Abraham Akinin, Steve Deiss, Prof. Sohmyung Ha, Soumil Jain, Prof. Siddharth Joshi, Prof. Chul Kim, Dr. Bruno Pedroni, Dr. Hesham Mostafa, Prof. Emre Neftci, Dr. Jongkil Park, Akshay Paul, Ritvik Sharma, Dr. Sadique Sheik, Pablo Tostado, Weier Wan, Dr. Jun Wang, Jiajia Wu, and other members from the Cauwenberghs Group at the Department of Bioengineering at UCSD.

A special thanks to the organization committee and participating members of the 2019 Annual Telluride Neuromorphic Workshop, in particular, Dr. Ryad Benosman, Dr. Sio-Hoi Ieng, Garrick Orchard, Gregor Lenz, Jonah Sengupta and Dr. Andreas Andreou.

I owe a debt of gratitude to those who financially supported my research and made this dissertation possible: Office of Naval Research (ONR), National Science Foundation (NSF) and UCSD ECE PhD Fellowship.

Last but not least, I would not be able to overcome all the challenges in my PhD journey without the incredible support from my family and friends in India, in San Diego, and around the world. More importantly, I would like to thank my loving wife, Aishwarya, my parents Usha and Hariharan, and my son Krish, for their unconditional love and unwavering support and inspiration throughout my ups and downs in my life. This dissertation is dedicated to them. My sincere thanks to my father-in-law Jayanthilal, mother-in-law Yamuna and brother-in-law Ashok for their constant encouragement and support. Finally, heartfelt gratitude to my best friend Balasubramaniam, for being a companion for life and someone I can always rely on.

Chapter 2 is in part a reprint of the material as it appears in Kubendran, R., Paul, A., and Cauwenberghs, G., "Query-driven multi-modal dynamic vision sensor with energy-efficient coding and streaming of visual information." (ready for submission to Nature Electronics), 2020. This chapter focuses on the hardware architecture design and measured results. The dissertation author was the primary investigator and author of this paper. This study was sponsored in part by the National Science Foundation (NSF EFRI-1137279), and the Office of Naval Research (ONR MURI N00014-13-1-0205). The views and the conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Chapter 3 is in part a reprint of the material as it appears in Kubendran, R., Paul, A., and Cauwenberghs, G., "Query-driven multi-modal dynamic vision sensor with energy-efficient coding and streaming of visual information." (ready for submission to Nature Electronics), 2020. This chapter focuses on the applications of the vision sensor described in Chapter 2. The dissertation author was the primary investigator and author of this paper. This study was sponsored in part by the National Science Foundation (NSF EFRI-1137279), and the Office of Naval Research (ONR MURI N00014-13-1-0205). The views and the conclusions contained in this document are those of the authors and should not be interpreted as representing the official

policies, either expressed or implied. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Chapter 4 is in part a reprint of the material as it appears in Kubendran, R., Park, J., Sharma, R., Kim, C., Joshi, S., Cauwenberghs, G., and Ha, S., "A 4.2-pJ/conv 10-B Asynchronous ADC with Hybrid Two-Tier Level-Crossing Event Coding." *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020. The dissertation author was the primary investigator and author of this paper.This study was sponsored in part by the National Science Foundation (NSF CCF-1317373), and the Office of Naval Research (ONR MURI N00014-13-1-0205). The views and the conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Chapter 5 is in part a reprint of the material as it appears in Wan, W., Kubendran, R., Eryilmaz, S.B., Zhang, W., Liao, Y., Wu, D., Deiss, S., Gao, B., Raina, P., Joshi, S., Wu, H., Cauwenberghs, G., and Wong, H.-S.P., "A 74 TOPS/W CMOS-ReRAM Neurosynaptic Core with Dynamically Reconfigurable Dataflow and In-situ Transposable Weights for Probabilistic Graphical Models." *IEEE International Solid State Circuits Conference (ISSCC)*, 2020, and in Kubendran, R., Wan, W., Joshi, S., Wong, H.-S.P., and Cauwenberghs, G., "A 1.52-pJ/Spike Reconfigurable Multimodal Integrate-and-Fire Neuron Array Transceiver." *ACM International Conference on Neuromorphic Systems (ICONS)*, 2020. The dissertation author was an equal contributing author of these papers. This study was sponsored in part by the National Science Foundation (NSF EFRI-1137279, CCF-1317560), and the Office of Naval Research (ONR MURI N00014-13-1-0205). The views and the conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Chapter 6 is in part a reprint of the material as it appears in Kubendran, R., Ieng, S.H., Orchard, G., Cauwenberghs, G., and Benosman, R., "An inverse STDP learning rule using synaptic delay kernel for spike-timing based temporal pattern recognition." (ready for submission to IEEE Transactions on Neural Networks and Learning Systems), 2020. The dissertation author was the primary investigator and author of this paper. This study was sponsored in part by the National Science Foundation (NSF CCF-1317373), and the Office of Naval Research (ONR MURI N00014-13-1-0205). The views and the conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

VITA

2008	Bachelor of Technology in Electronics and Communications Engineering National Institute of Technology, Tiruchirappalli, India
2012	Master of Science in Electrical and Computer Engineering Purdue University, USA
2020	Doctor of Philosophy in Electrical Engineering (Medical Devices and Systems) University of California San Diego, USA

PUBLICATIONS

Journal Articles

- **Kubendran, R.**, Paul, A., and Cauwenberghs, G. (ready for submission). "Query-driven Multi-modal Dynamic Vision Sensor with Energy-efficient Coding and Streaming of Visual Information." *Nature Electronics*.
- **Kubendran, R.**, Ieng, S.H., Orchard, G., Cauwenberghs, G., and Benosman, R. (ready for submission). "An Inverse STDP Learning Rule using Synaptic Delay Kernel for Spike-timing based Temporal Pattern Recognition." *IEEE Transactions on Neural Networks and Learning Systems*.
- Wan, W., **Kubendran, R.**, Eryilmaz, S.B., Zhang, W., Liao, Y., Wu, D., Deiss, S., Gao, B., Raina, P., Joshi, S., Wu, H., Wong, H.-S.P., and Cauwenberghs, G. (in preparation). "A CMOS-ReRAM neurosynaptic multi-core processor with 12K neurons and 3M synapses for neuromorphic and machine learning applications." *Nature*.
- Kim, C., Park, J., Ha, S., Akinin, A., **Kubendran, R.**, Mercier, P.P., and Cauwenberghs, G. (2019). "A 3 mm× 3 mm Fully Integrated Wireless Power Receiver and Neural Interface System-on-Chip." *IEEE Transactions on Biomedical Circuits and Systems*.
- **Kubendran, R.**, Lee, S., Mitra, S., and Yazicioglu, R.F. (2014). "Error Correction Algorithm for High Accuracy Bio-Impedance Measurement in Wearable Healthcare Applications." *IEEE Transactions on Biomedical Circuits and Systems*.

Conference Articles

- **Kubendran, R.**, Paul, A., and Cauwenberghs, G. (in preparation). "A 512x512 1000fps Query-driven Dynamic Vision Sensor for surveillance and autonomous drone navigation." *IEEE International Solid State Circuits Conference (ISSCC)*.

- **Kubendran, R.**, Wan, W., Joshi, S., Wong, H.-S.P., and Cauwenberghs, G. (2020). "A 1.52-pJ/Spike Reconfigurable Multimodal Integrate-and-Fire Neuron Array Transceiver." *ACM International Conference on Neuromorphic Systems (ICONS)*.
- Wan, W., **Kubendran, R.**, Gao, B., Joshi, S., Raina, P., Wu, H., Cauwenberghs, G., and Wong, H.-S.P. (2020). "A Voltage-Mode Sensing Scheme with Differential-Row Weight Mapping for Energy-Efficient RRAM-Based In-Memory Computing." *IEEE Symposia on VLSI Technology and Circuits*.
- **Kubendran, R.**, Park, J., Sharma, R., Kim, C., Joshi, S., Cauwenberghs, G., and Ha, S., (2020). "A 4.2-pJ/conv 10-B Asynchronous ADC with Hybrid Two-Tier Level-Crossing Event Coding." *IEEE International Symposium on Circuits and Systems (ISCAS)*.
- **Kubendran, R.**, Sengupta, J., Neftci, E., and Andreou, A. (2020). "High-Speed, Real-Time, Spike-Based Object Tracking and Path Prediction on Google Edge TPU." *IEEE Artificial Intelligence Circuits and Systems (AICAS)*.
- Wan, W., **Kubendran, R.**, Eryilmaz, S.B., Zhang, W., Liao, Y., Wu, D., Deiss, S., Gao, B., Raina, P., Joshi, S., Wu, H., Cauwenberghs, G., and Wong, H.-S.P. (2020). "A 74 TOPS/W CMOS-ReRAM Neurosynaptic Core with Dynamically Reconfigurable Dataflow and In-situ Transposable Weights for Probabilistic Graphical Models." *IEEE International Solid State Circuits Conference (ISSCC)*.
- **Kubendran, R.**, Paul, A., and Cauwenberghs, G. (2020). "Query-driven Dynamic Vision Sensor." *IEEE International Solid State Circuits Conference (ISSCC), Student Research Preview (SRP) Demo*.
- Kim, C., Ha, S., Akinin, A., Park, J., **Kubendran, R.**, Wang, H., Mercier, P.P., and Cauwenberghs, G. (2017). "Design of miniaturized wireless power receivers for mm-sized implants." *IEEE Custom Integrated Circuits Conference (CICC)*.
- Kim, C., Park, J., Ha, S., Akinin, A., **Kubendran, R.**, Wang, H., Mercier, P.P., and Cauwenberghs, G. (2016). "A fully integrated 144 MHz wireless-power-receiver-on-chip with an adaptive buck-boost regulating rectifier and low-loss h-tree signal distribution." *IEEE Symposia on VLSI Technology and Circuits*.
- **Kubendran, R.**, Kim, S., and Yazicioglu, R.F. (2016). "Error correction algorithm for high accuracy bio-impedance measurement in wearable healthcare applications." *IEEE International Symposium on Circuits and Systems (ISCAS)*.
- **Kubendran, R.** (2012). "Electromagnetic and Laplace domain analysis of memristance and associative learning using memristive synapses modeled in SPICE." *IEEE International Conference on Devices, Circuits and Systems (ICDCS)*.
- **Kubendran, R.**, Krishnan, H., Manola, B., John, S.W.M., Chappell, W.J., and Irazoqui, P.P. (2011). "A generic miniature multi-feature programmable wireless powering headstage ASIC for implantable biomedical systems." *IEEE International Conference of Engineering in Medicine and Biology Society (EMBS)*.

ABSTRACT OF THE DISSERTATION

Energy-Efficient Event-Based Vision Sensors and Compute-in-Memory Architectures for Neuromorphic and Machine Learning Applications

by

Rajkumar Chinnakonda Kubendran

Doctor of Philosophy in Electrical Engineering (Medical Devices and Systems)

University of California, San Diego, 2020

Professor Gert Cauwenberghs, Chair
Professor Patrick Mercier, Co-Chair

Neuromorphic engineering pursues the design of electronic systems emulating function and structural organization of biological neural systems in silicon integrated circuits that embody similar physical principles. The work in this dissertation presents advances in the field of neuromorphic engineering by demonstrating the design and applications of energy-efficient event-based sensors, compute-in-memory architectures, event-based learning algorithms and asynchronous data converters.

This dissertation focuses on neuromorphic very large scale integration (VLSI) archi-

tecture and algorithm design for the implementation of sensors and processors that are highly energy-efficient, emulating brain function through event-based sensory processing. In particular, we present three novel contributions towards achieving the goal of integrated visual cortical processing on silicon hardware. First, we present a novel hybrid approach to vision sensing called query-driven dynamic vision that achieves the best energy efficiency reported to-date and then show various applications enabled by such sensors with improved performance compared to conventional sensors. Second, we present an integrated compute-in-memory (CIM) architecture that combines an emerging device called resistive random access memory (RRAM) with complimentary metal oxide semiconductor (CMOS) technology. This design achieves the highest versatility in terms of reconfigurable dataflow, multiple modes of neuron activation using a single topology and the best energy-efficiency reported to-date for CMOS-RRAM CIM architectures. Third, we present a learning rule called the inverted synaptic time dependent plasticity (STDP) rule that can learn temporal patterns using only spike event timing information.

Combining these three advances, we are now a step closer to realizing the function and efficiency of biological vision in neuromorphic hardware, where the qDVS as artificial silicon retina provides the event-based visual stimulus to the primary visual cortex layers implemented on a CIM architecture using convolutional neural networks (CNN) to deploy event-based learning algorithms for temporal pattern recognition.

Chapter 1

Introduction

Understanding and augmenting brain function is one of the Grand Challenges of the 21st century, calling for concerted efforts beyond traditional disciplines. This includes many facets of brain research including, but not limited to, emulation of brain function using neuromorphic computing architectures, innovative neuro-technologies for brain interfacing, implementation of artificial neural networks (ANN) for artificial intelligence (AI) applications by drawing high-level inspiration from the organization of the brain [8].

Among these disciplines, rapid progress has been made in the field of AI during recent years with the advent of deep learning methods that has catapulted machine learning algorithms and architectures to the forefront [45]. Deep learning methods have accelerated the adoption of artificial neural networks (ANN) in many applications, spanning and impacting every field that requires automation [52, 27, 40]. This has resulted in a tremendous demand for hardware accelerators, primarily graphical processing units (GPU) [20] and recently field programmable gate array (FPGA) based systems [62] and custom hardware. There has been an escalated interest in both industry and academia, in developing robust, energy efficient and re-configurable hardware [18, 46, 48, 29]. Though deep learning has led a revolution in widespread adoption of AI but has its own limitations of poor energy-efficiency for large-scale networks and requirement of large

datasets for training.

On the other hand, neuromorphic systems have gained significant prominence recently, as it holds promise to extreme energy efficiency and low latency, ability to train with fewer samples, which is suitable for edge computing and Internet-of-Things (IoT) applications [22, 21, 28, 16]. Inspired by biophysical principles, the neuron design can use noise and mismatch variations to its advantage, while generating and processing spike-based events that can be efficiently implemented using switched-capacitor circuits and techniques for analog implementations [39, 36, 37] and advanced technology nodes for digital implementations [1, 12, 38]. However the major bottleneck preventing the wide adoption of neuromorphic systems is the lack of efficient and robust algorithms to train and validate such networks.

Significant advancements are still needed in the development of new class of AI architectures with emerging devices and algorithms that take advantage of distributed networks with sparse activity, taking inspiration from biology, neuroscience and machine learning. Recent developments in edge computing with AI, aims at advancing the engineering and applications of energy-efficient, adaptive and ubiquitous sensing and event-based processing of multimodal data for intelligent decision making, in highly resource-constrained environments including mobile platforms and wearables. Energy efficiency is the critical bottleneck for targeted applications such as fully autonomous drone navigation with lifelong learning, security surveillance, augmented/virtual reality (AR/VR). Neuromorphic systems with event-based sensing and compute-in-memory architectures and algorithms has great potential to usher in next-generation AI. Fig.1.1 presents an overview of neuromorphic VLSI systems engineering [8].

The outline of this dissertation is described next and follows closely the steps taken which eventually culminate towards an integrated event-based vision sensing and processing on silicon hardware.

Fig.1.2 presents the overall theme of this dissertation, which is to pursue novel approaches to realizing energy-efficient architectures on silicon hardware that can model visual sensing,

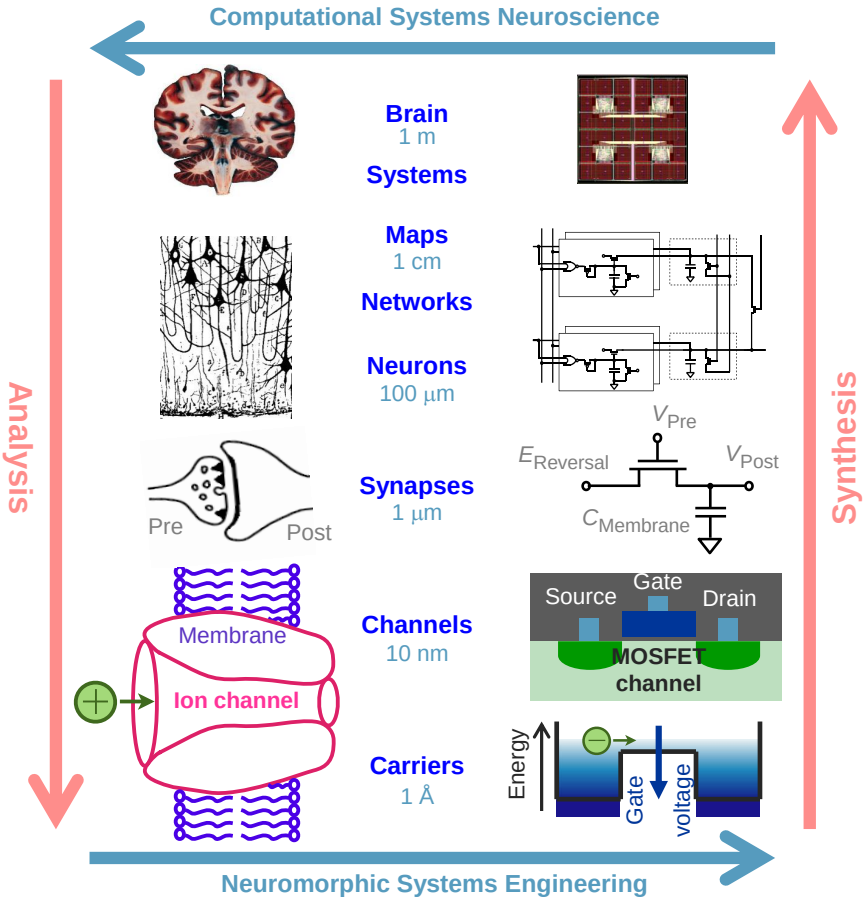


Figure 1.1: Neuromorphic VLSI systems engineering overview [8].

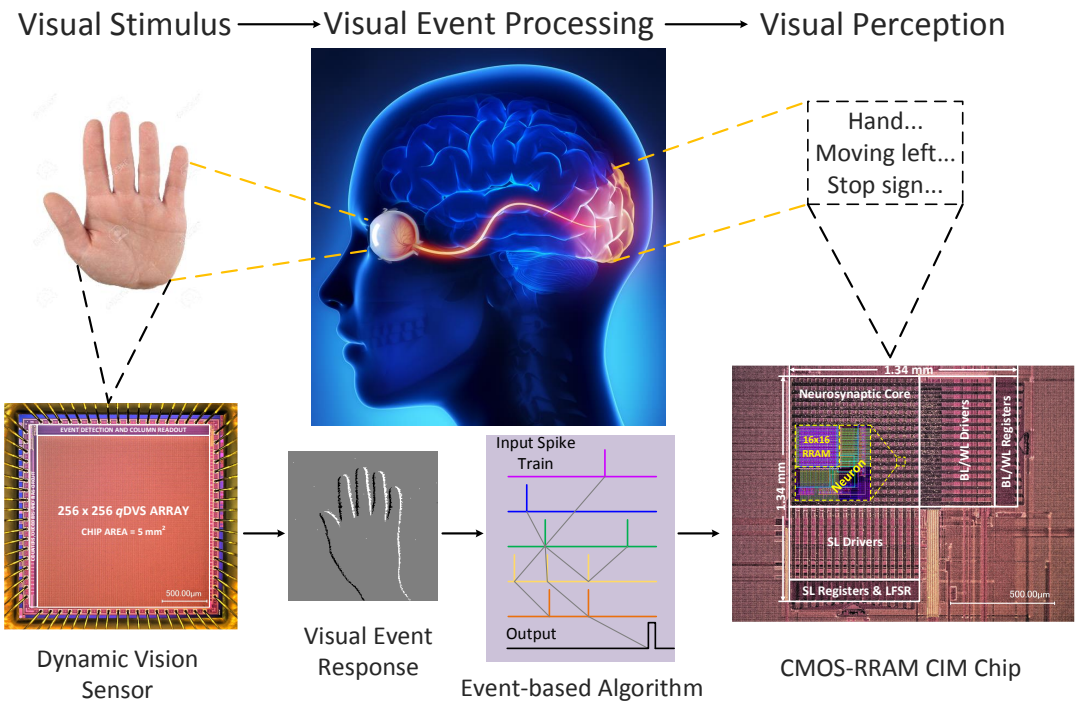


Figure 1.2: Visual sensing, processing and perception on silicon hardware.

processing and perception drawing significant inspiration from biology. Towards achieving that goal, this dissertation highlights three contributions - a query-driven dynamic vision sensor that provides temporal contrast event streams, a CMOS-RRAM computer-in-memory architecture that can implement event-based networks for processing input events and predicting/perceiving their state and an event-based learning algorithm that can learn timing information from a sequence of input events. These contributions are described in detail in the upcoming chapters.

Chapter 2 describes a query-driven dynamic vision sensing (*q*DVS) approach to computational imaging that substantially improves on the achievable pixel density and energy efficiency of visual event coding. The chip architecture combines complementary advantages of frame scanning active pixel sensors (APS) and event-driven DVS (*e*DVS) CMOS imagers. Event-based cameras are modeled based on the operation of the biological retina, as shown in Fig.1.3. Phototransduction occurs at the photoreceptor sites in the eye, the rods being sensitive to low light for dark vision and the cones being color sensitive to different wavelengths of light (red, green and

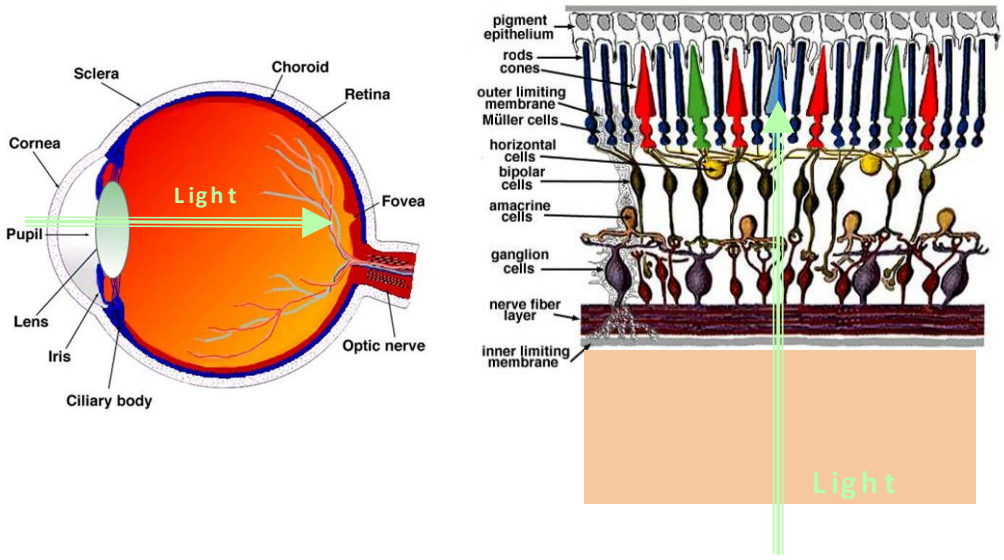


Figure 1.3: Biological retina overview.

blue). The horizontal and bipolar cells can then provide amplification and spatial highpass filtering, whereas the amacrine cells provide temporal highpass filtering. However, the most important part of retinal vision is the ganglion cells, that are responsible for spike rate and temporal event encoding to be transmitted through the optic nerve to the visual cortex in the brain. The custom pixel design with a tailored photodiode supports multiple modes of operation, including the proposed query-driven event streaming mode with 2-bit readout and a traditional frame scanning mode with 6-bit readout. Overall power consumption is drastically reduced by using circuit design optimizations to lower both static and dynamic current draw. A 256×256-pixel *q*DVS prototype chip was fabricated in a standard CMOS 180nm process technology, occupying an area of 5 mm^2 and pixel fill factor of 33%. This architecture achieves 10% temporal contrast sensitivity and consumes 0.5 mW from 1.2 V supply voltage at a peak event rate of 80 Meps, yielding 10 pJ/pixel-event energy efficiency.

Chapter 3 elaborates the applications of event-based vision sensors. The prototype described in Chapter 2 demonstrates superior performance in varied applications such as stereo

vision, video compression and visual attention. Combined with a custom computer-in-memory hardware accelerator for AI applications, the setup could potentially be the most energy-efficient system to-date for sensing and recognizing objects real-time.

In Chapter 4, an asynchronous continuous-time level-crossing analog-to-digital converter (LC-ADC) for high-throughput, high-resolution applications is presented. The proposed 10-bit ADC architecture comprises two stages of level-crossing ADCs, the first stage resolving for 5 MSBs and the second folded residue stage for 5 LSBs. Compared to uniform-sampling synchronous ADCs, LC-ADCs generate fewer samples for sparse signals, useful in many applications for biomedical signal acquisition, event-driven computer vision, etc. Unlike conventional LC-ADCs with a few comparators tuned for lower power consumption to acquire sparse signals, this two-tier LC-ADC is optimized for high-resolution tracking of continuous signals, like Electrocardiogram (ECG). Designed and fabricated in 0.18-μm CMOS technology, chip area of the proposed ADC is $1310 \times 125\ \mu m^2$. Operating at 1.8 V supply, the ADC consumes 160–426 μW for 1 Hz to 200 kHz input frequencies at full scale amplitude and achieves an energy efficiency figure-of-merit of 4.16-pJ/conv.

Chapter 5 describes a Compute-In-Memory architecture implemented in a 130-nm CMOS/RRAM process, that delivers the highest reported computational energy-efficiency of 74 TOPS/W for RRAM-based CIM architectures while simultaneously offering dataflow reconfigurability to address the limitations of previous designs. This is made possible through two key features: 1) a runtime reconfigurable dataflow with in-situ access to RRAM array and its transpose for efficient access to NN weights and 2) a voltage sensing stochastic integrate-and-fire analog neuron that is reused for correlated double sampling (CDS), stochastic voltage integration, and threshold comparison. The design features an array of 16×16 charge-mode mixed-signal reconfigurable neurons to implement various activation functions, including step, sigmoid and Rectified Linear Unit (ReLU), through a partial reset mechanism and additive stochastic noise by Linear Feedback Shift Register (LFSR) coupling. The neuron outputs spike-based sparse syn-

chronous events, which are either binary (event/no event) or ternary (positive/negative/no events). The reconfigurable energy-efficient design makes this architecture suitable for deep learning and neuromorphic applications including convolutional neural networks (CNN), restricted boltzmann machines (RBM) and general event-driven computing.

Chapter 6 presents a weight update learning rule based only on pre- and post-synaptic spike times for spike timing based pattern recognition. Following an inverse principle to that of the well known Synaptic Time Dependent Plasticity (STDP) rule for Hebbian learning, we name this rule as the ”Inverse STDP” rule. This rule can be applied to event-based networks with spatio-temporal features called time surfaces, that are generated using a synaptic delay kernel consisting of programmable weights with an exponential decay when activated. Time-encoding of spikes achieves better energy efficiency than rate coding of spikes and hence this rule can be applied to many applications where timing and efficiency are of critical importance. This rule allows to train a network of neurons and synapses to identify an unique sequence in the timing of input spike events. The proposed learning rule translates readily to neuromorphic hardware such as the Intel's Loihi platform. Implementation results are shown for both software and hardware implementations.

Finally, Chapter 7 summarizes the contributions and the findings of this dissertation. We then describe our current efforts toward building a real-time event-based energy-efficient neuromorphic vision system combining vision sensors with compute-in-memory hardware for visual event processing. We end with a discussion of the potential impact of such systems on applications in the realm of edge computing with AI.

Chapter 2

Query-driven Dynamic Vision Sensor with Energy-efficient Coding and Streaming of Visual Information

2.1 Introduction

Traditional CMOS imagers have high pixel density but use frame scanning, at a fixed clock rate, to continuously stream out the pixel intensity data, incurring constant data rate and power independent of information content. In contrast, DVS or "silicon retina" chips adopt an asynchronous detection and coding strategy to stream out visual events only when they occur, supporting variable output data rate that is intrinsically to the visual information rate [30, 58, 13, 14]. However, typical event-driven DVS implementations incur extra area and power overhead to continuously monitor for events and handle request and acknowledge handshaking within each pixel, thus resulting in low pixel density and incurring substantial static power despite the savings resulting from the efficiency of visual event coding [17].

Query-driven DVS overcomes such limitations by combining the best of both APS and

Table 2.1: Comparison between APS, *e*DVS and *q*DVS image sensor architectures.

Parameter	Active Pixel Sensors (APS)	Event-driven Dynamic Vision Sensors (*e*DVS)	Query-driven Dynamic Vision Sensors (*q*DVS)
Latency	High (upto 100fps $\approx$ 10ms)	Low (frameless $\approx 10\mu$s)	Medium and Variable (100-1000fps $\approx$ 1ms)
Bandwidth Utilization	High Activity independent (N bit high resolution high speed ADC)	Medium Activity dependent (Pixel address + 2 bit event + REQ/ACK)	Low Activity dependent (2 bit event)
Density	High (3T-5T per pixel)	Low (>10T per pixel)	Medium (7T-10T per pixel)
Power	High dynamic, medium static (100-500nW per pixel)	High dynamic, high static (500-800nW per pixel)	Low dynamic, low static (<10nW per pixel)

*e*DVS: using clocked time-division multiplexing to periodically scan the array, querying each pixel for threshold change events in pixel intensity. Fig. 2.1 illustrates the design and operation of APS, *e*DVS and *q*DVS imagers. This work extends on the *q*DVS concept initially introduced and demonstrated in [11], dramatically increasing its energy efficiency over APS and *e*DVS alternatives while retaining functional flexibility that subsumes both.

Eliminating any overhead within the pixel for detection, arbitration, and handshaking of events that are required to continuously monitor and rapidly route events in *e*DVS, the pixel area and power consumption are substantially reduced. Like *e*DVS, however, the activity-based event coding produces a sparse output data stream in which data is being transmitted only for active, information-bearing pixels, significantly reducing the data rate over APS. The ultra-low power operation and activity-based output streaming offer a versatile platform ideally suited for a myriad of applications in security surveillance, drone navigation, and other domains requiring rapid tracking and logging of visual events.

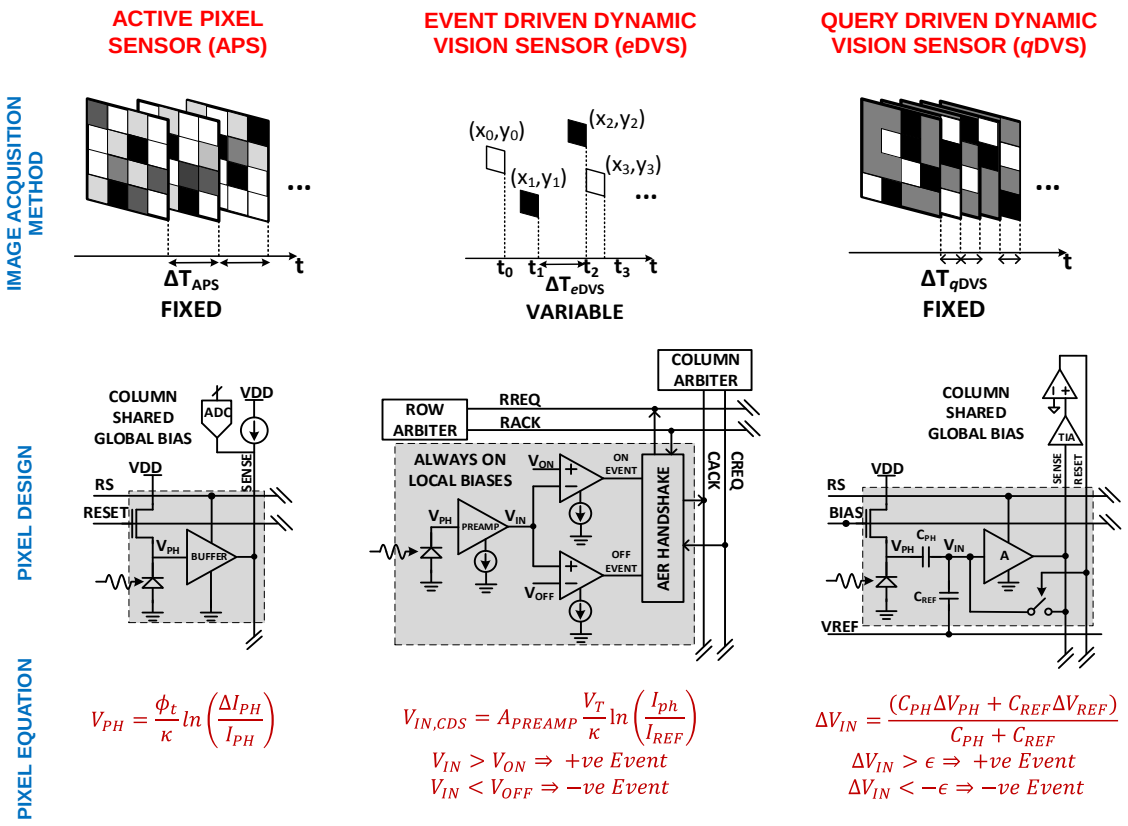


Figure 2.1: Comparison between APS, *e*DVS and *q*DVS image sensor architectures.

2.2 qDVS Architecture

Shown in Fig. 2.3, the *q*DVS consists of a 256×256 array of active pixel sensors augmented with temporal difference threshold detection capabilities served through configurable circuits at the array periphery. The *q*DVS performs a query-driven temporal change contrast event readout, which registers whether the intensity has risen or fallen by predetermined differentials from a previous intensity level. Row and column decoders support efficient serial scanning for efficient full field-of-view intensity or event readout, by virtue of Gray-coded addressing which avoids glitches during switching from one row/column to another, while additionally saving power by minimizing the number of transitions per complete scan. Digitally configurable functions of threshold detection and various forms of intensity coding are time-multiplexed at the periphery of the array, supporting large dynamic range while tracking fast transients in intensity through in-pixel temporal differencing. Periodic scanning is performed using three external clocks which trigger Gray counters whose output address codes are used to traverse the rows and columns and to read out pixel event or intensity data. The row decoder selects one row at a time, during which all the columns are queried in parallel.

The *q*DVS active pixel (Fig. 2.3) combines functions of phototransduction, temporal differencing, threshold detection, and reference reset, using as few as five metal-oxide-semiconductor (MOS) transistors in addition to two MOS capacitors. The photodetector as shown is a parasitic p-n junction photodiode that doubles as the source/drain-to-bulk junction of MOS transistors; other implementations of a photodiode or phototransistor may use a shallow or deep implant providing wider depletion region for greater quantum efficiency as desired, at the expense of silicon area.

The photodetector voltage encodes light intensity logarithmically where NBIAS is biased in the subthreshold MOS regime and the source voltage VPH settles to a steady state value logarithmic in photocurrent. The logarithmic voltage output VPH couples to the amplifier input

VIN through capacitance CPH, while a voltage reference signal VREF additionally couples to VIN through capacitance CREF. The relative strength of the signal vs reference coupling is set by the ratio CREF/CPH, and hence greater sensitivity is obtained by a larger choice of the ratio. The voltage on VREF thus provides fine control over the threshold for event detection, which is served inside the pixel by just a single-transistor common-source amplifier input stage.

$$I_{PH} = I_0 e^{\frac{\kappa V_{PH}}{kT/q}} \tag{2.1}$$

$$\frac{\Delta I}{I_{PH}} = e^{\frac{\kappa \Delta V_{PH}}{V_T}} - 1 \tag{2.2}$$

For x% temporal contrast sensitivity,

$$\Delta V_{PH} = \frac{V_T}{\kappa} \ln(1+x) \tag{2.3}$$

The MOS capacitors in the pixel combine using charge sharing to determine the input voltage, V_{IN} to the common source amplifier,

$$\Delta V_{IN} = \frac{C_{PH}\Delta V_{PH} + C_{REF}\Delta V_{REF}}{C_{PH} + C_{REF}} \tag{2.4}$$

The query for an entire row of pixels is initiated by pulsing RS active high, which activates the current output of the in-pixel amplifier input stage onto the column readout line SENSE. The current for each column is further amplified for event detection at the periphery and fed back with high trans-impedance gain over the FEEDBACK column feedback line to the pixel. Selective activation of RESET closes the high-gain feedback loop, discharging the coupling capacitors to reset the voltage on the VIN input node to the amplifier threshold. This selective reset samples the present intensity in the pixel as reference for subsequent change in intensity, providing temporal differencing in threshold change detection. Positive and negative thresholds for change in intensity are set by the amplitudes of negative and positive steps, respectively, in

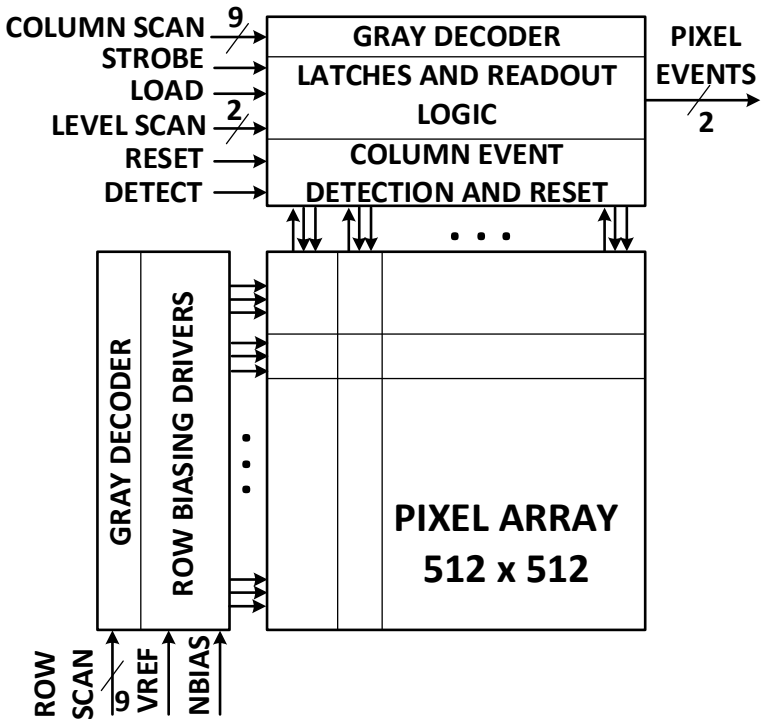


Figure 2.2: *q*DVS architecture block diagram.

the reference voltage VREF during event detection. By conditioning the selective pixel reset on detection of an event in the amplified output at the periphery (Fig. 2.5), the *q*DVS registers level crossings in intensity at adjustable incremental and decremental steps, providing an asynchronous form of delta coding in intensity.

The column drivers at the periphery of the array of active pixels (Fig. 2.5) serve three purposes: complementing and completing the amplification within the pixel to acquire the threshold detection signal from a selected row of pixels; feedback the high-gain amplified signal to reset the threshold detection reference level in the selected row of pixels; and encode the digital output from a series of threshold detection operations for row-parallel, column-serial scanned output. Temporal contrast change query provides 2 bit pixel events, depending on whether there was an increase, decrease or no change in pixel intensity. These bits are serially read out through a column scan which proceeds in pipelined fashion while the next row is acquired.

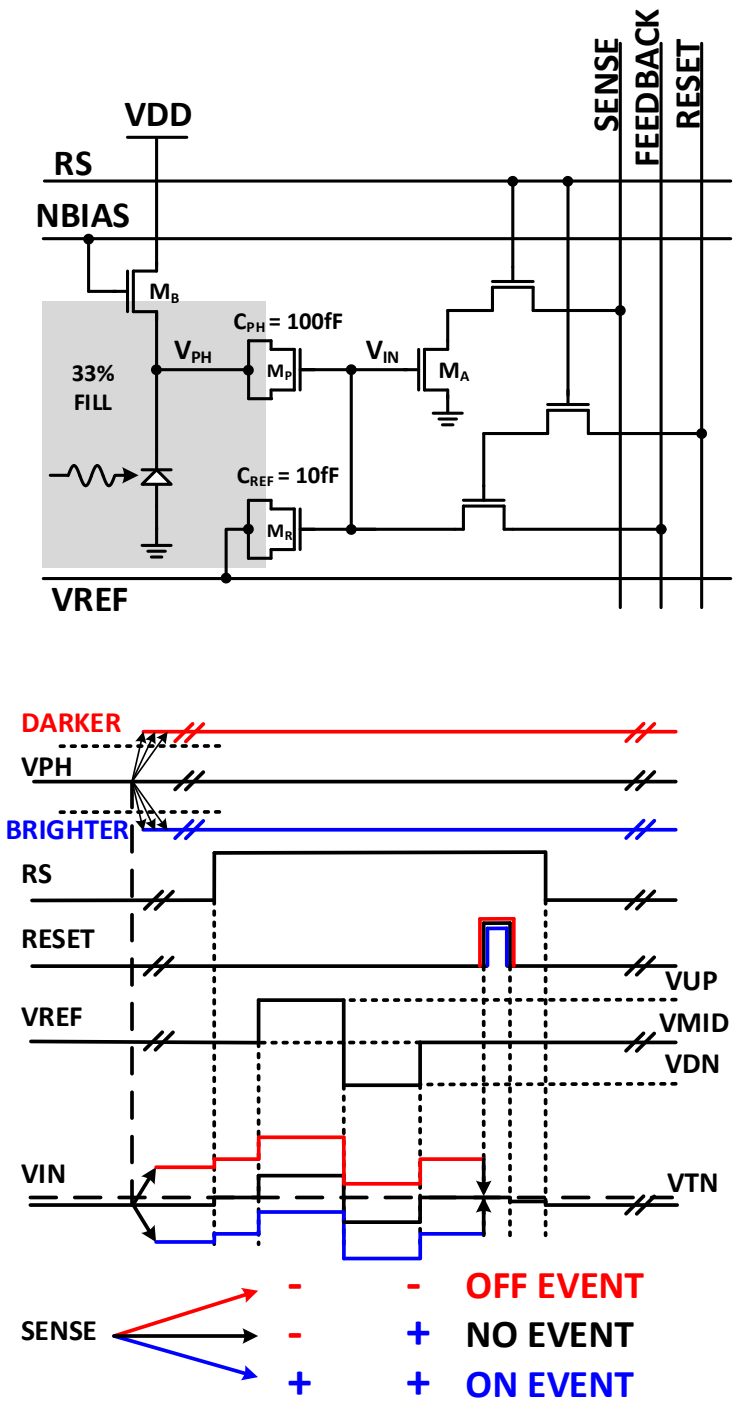


Figure 2.3: *q*DVS pixel schematic, layout and timing waveforms.

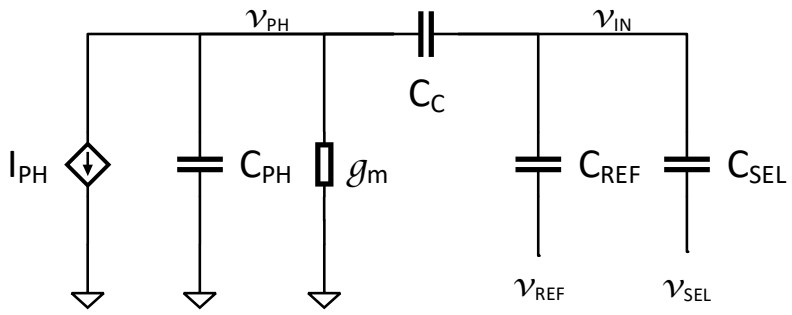


Figure 2.4: qDVS pixel small signal analysis.

2.3 Pixel Small Signal Analysis

Consider the small signal circuit of the pixel schematic shown in Fig. 2.4

$$g_m = \frac{I_{PH,DC}}{kT/q} \tag{2.5}$$

Using Kirchoff's current law, we obtain the following equations,

$$sC_c(v_{PH} - v_{IN}) = -[i_{PH} + (sC_{PH} + g_m)v_{PH}] \tag{2.6}$$

$$sC_c(v_{PH} - v_{IN}) = sC_{REF}(v_{IN} - v_{REF}) + sC_{SEL}(v_{IN} - v_{SEL}) \tag{2.7}$$

Rearranging equations 2.6 and 2.7,

$$v_{PH} = \frac{-i_{PH} + sC_c v_{IN}}{g_m + s(C_c + C_{PH})} \tag{2.8}$$

$$v_{IN} = \frac{C_c v_{PH} + C_{REF} v_{REF} + C_{SEL} v_{SEL}}{C_c + C_{REF} + C_{SEL}} \tag{2.9}$$

Solving for v_{IN},

$$v_{IN} = \frac{\frac{-C_c i_{PH}}{g_m(C_c+C_{REF}+C_{SEL})} + \frac{C_{REF}v_{REF}+C_{SEL}v_{SEL}}{C_c+C_{REF}+C_{SEL}}[1 + \frac{s(C_c+C_{PH}}{g_m}]}{1 + s\frac{(C_c+C_{PH})(C_c+C_{REF}+C_{SEL})-C_c^2}{g_m(C_c+C_{REF}+C_{SEL})}} \tag{2.10}$$

Equation 2.10 can be simplified into sum of three transfer functions for contributions from i_{PH},

v_{REF} and v_{SEL}.

$$v_{IN}(s) = R_{PH}(s)i_{PH}(s) + H_{REF}(s)v_{REF}(s) + H_{SEL}(s)v_{SEL}(s) \tag{2.11}$$

where R_{PH}, H_{REF}, H_{SEL}, τ_C and τ_{PH} are given by,

$$R_{PH}(s) = -\frac{C_c}{C_c + C_{REF} + C_{SEL}} \frac{1}{g_m} \frac{1}{1 + s\tau_{PH}} \tag{2.12}$$

$$H_{REF}(s) = \frac{C_{REF}}{C_c + C_{REF} + C_{SEL}} \frac{1 + s\tau_C}{1 + s\tau_{PH}} \tag{2.13}$$

$$H_{SEL}(s) = \frac{C_{SEL}}{C_c + C_{REF} + C_{SEL}} \frac{1 + s\tau_C}{1 + s\tau_{PH}} \tag{2.14}$$

$$\tau_{PH} = \frac{C_{PH}(C_c + C_{REF} + C_{SEL}) + C_c(C_{REF} + C_{SEL})}{g_m(C_c + C_{REF} + C_{SEL})} \approx \frac{C_{PH}}{g_m} \tag{2.15}$$

Assuming $C_c >> C_{REF}, C_{SEL}$, $$\tau_C = \frac{C_c + C_{PH}}{g_m} > \tau_{PH} \tag{2.16}$$

Equations 2.15 and 2.16 for time constants τ_{PH} and τ_c can be used to calculate the biphasic pulse response at v_{IN} for the pulse applied at v_{REF}. The response settling time can be determined when the residue voltage at v_{IN} is small enough to not be detected by the column readout circuit.

$$\frac{v_{residue}}{v_{REF}} = \frac{C_{REF}}{C_c + C_{REF} + C_{SEL}} \frac{\tau_c}{\tau_{PH}} (1 - e^{\frac{-\Delta T}{\tau_{PH}}})^2 e^{\frac{-(t-2\Delta T)}{\tau_{PH}}} \approx \frac{C_{REF}}{C_{PH}} e^{\frac{-2\Delta T}{\tau_{PH}}} \tag{2.17}$$

2.4 Imager Operation

Any pixel in the selected row can raise an event depending on whether the intensity of light shone on the pixel has decreased or increased, compared to the previous level of intensity at the pixel. An event is determined by using a two-level voltage test input as VREF (VUP and VDN). The pixel has been carefully designed to minimize area by using MOSCAP for high density capacitive input coupling and not using PMOS transistors (since it requires a NWELL

boundary), and avoiding source follower transistor to isolate the photodiode node, VPH. Also, a local reset for the pixel is generated by combining row select and column reset. A column readout circuit (Fig. 2.5) performs thresholding comparison of the pixel photodiode voltage to detect events. The pixel readout circuit consists of a Gm-boosted high gain cascode amplifier (Gain $>$ 90dB) along with a clamp to set input dc bias, and a dynamic comparator, with no dc bias current, to compare the amplifier output with a common-mode voltage, VCM. The power budget is carefully planned for each column. With a supply voltage of 1.2V and 120nA bias current per column, the power consumed per active column is only 144nW. The bias current per column includes a 20nA trickle current to guarantee proper biasing of the amplifier even if no pixel in the column rises an event.

An event readout circuit cascades the column readout circuit. This circuit generates a column-wise load signal to latch the event onto the strobe latch, column-wise reset signal to reset the pixel. The load signal is generated using SR latches and combinational logic with separate clock for output code readout. A global strobe signal synchronized with the readout clock is used to latch the event to the output bus, which is driven by tristate buffers. Event output data is loaded onto the readout bus through LOAD and STROBE signals (Fig. 2.5). The CODE INC clock is used by level scan counter to generate the event data loaded to DATA OUT bus. Event readout logic uses GLOAD and SELECT POLARITY to generate the LOAD signal to latch the event to the output bus (Fig. 2.5). RESET POLARITY and RESET SAMPLE are used to generate the PIXEL RESET signal to reset the pixel after readout. The CODE RST and EVENT RESET signals can be used to reset the logic and start a new cycle of event readout.

2.5 Experimental Validation

Fig. 2.8 shows the testbench setup for measurements and experiments with the *q*DVS prototype board. Fig. 2.7 characterizes temporal contrast sensitivity to relative changes in pixel

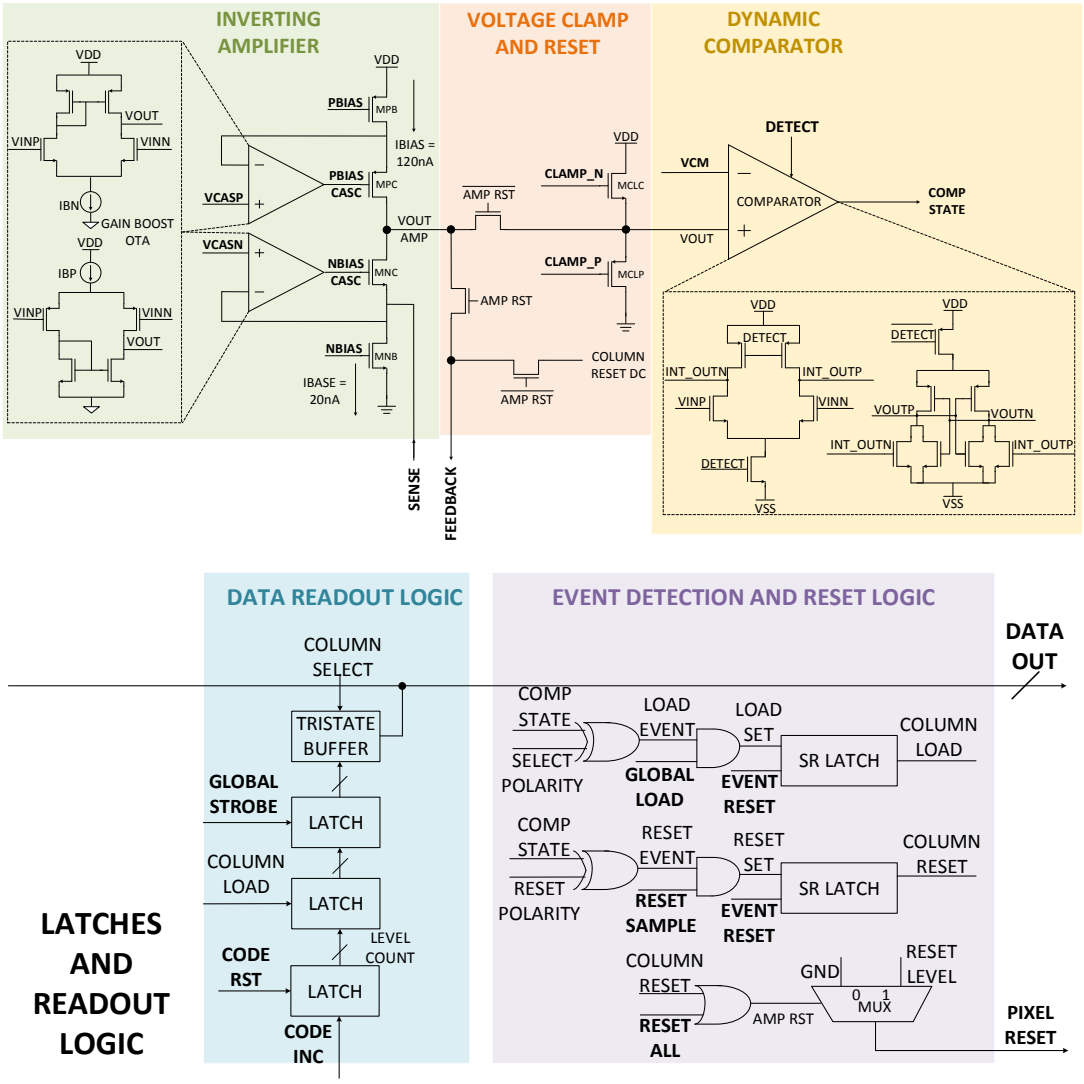


Figure 2.5: Column periphery event detect, reset and readout circuit implementation.

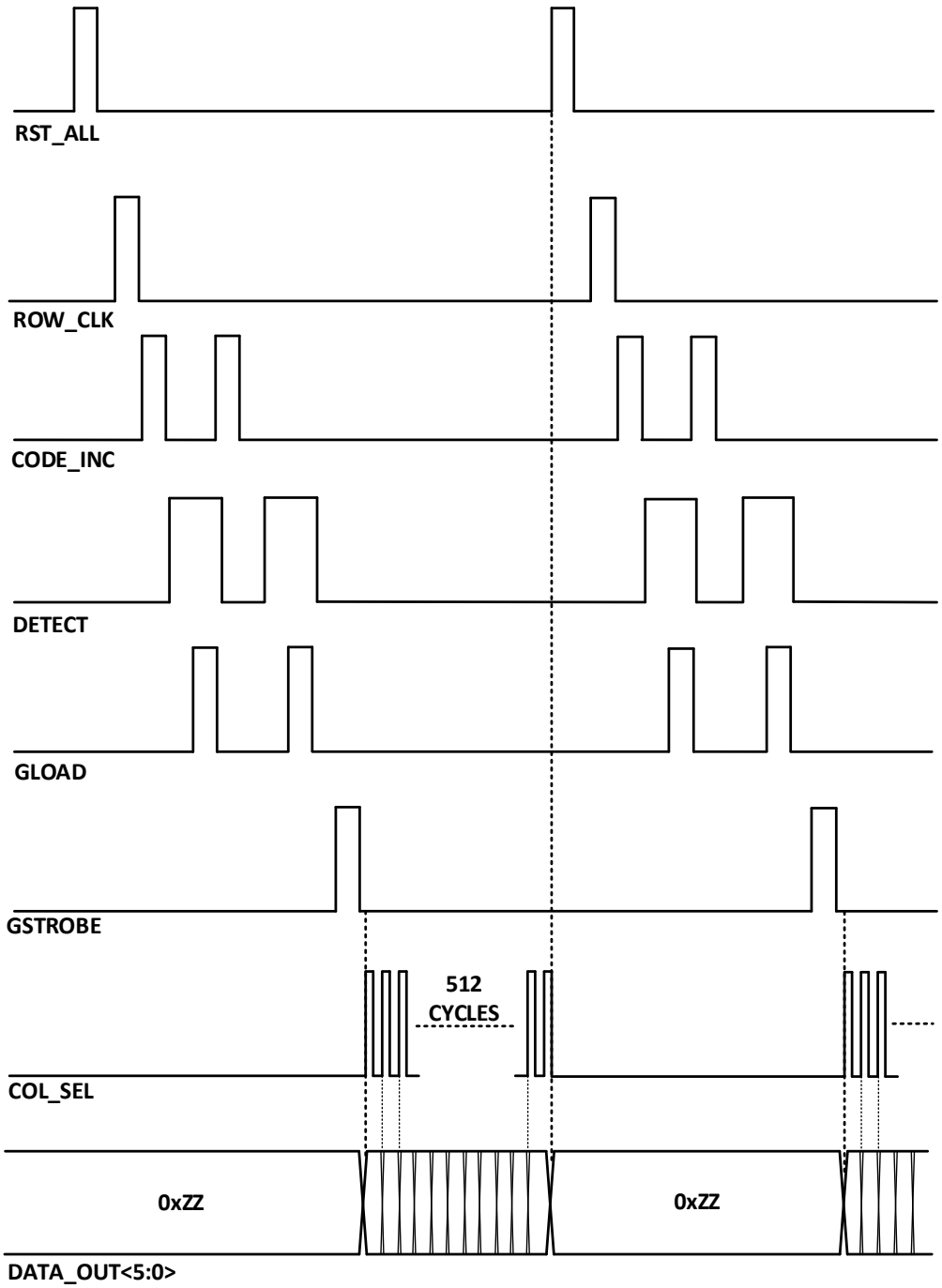


Figure 2.6: Column periphery event readout timing waveform.

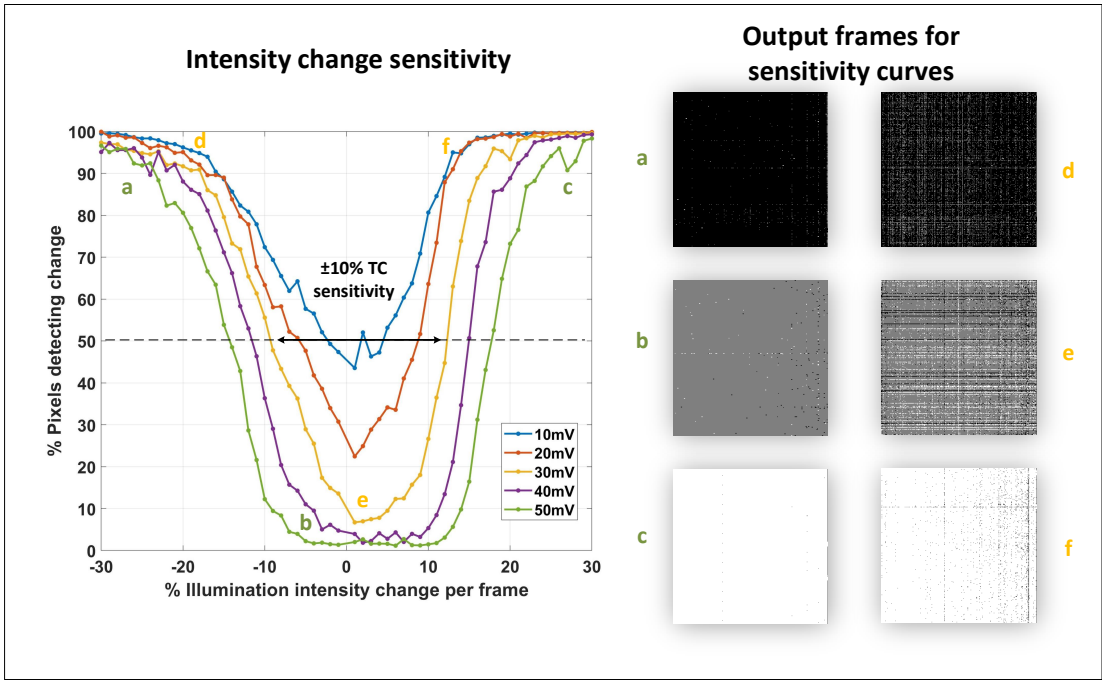


Figure 2.7: *q*DVS contrast sensitivity at various threshold levels. Lower case letters represent frames captured from specific illumination intensity deltas.

intensity, for two different contrast threshold levels set by VREF. Sample recorded frames a through f demonstrate the uniformity in pixel response to a pulsed intensity change, increasing with ΔVREF setting. Fig. 2.9 shows results from a pinwheel experiment contrasting the outputs of *q*DVS with a regular APS camera at different spin velocities, but at identical frame rate (40 Hz), revealing key differences in fundamental operation. *q*DVS visual events at 40fps for pinwheel rotating at different speeds, at 20, 80 and 100 rpm, are compared to APS camera output at 720p resolution and 30fps. Fig. 2.9 also shows snapshots of fast movement of hand left and then right across visual field. *q*DVS can also implement eye tracking during rapid left and right eye movements as seen in Fig. 2.9. Overall, *q*DVS achieves 2× greater pixel density than [49] and 20× greater energy efficiency (lower pJ/pixel-event) than state of the art [5].

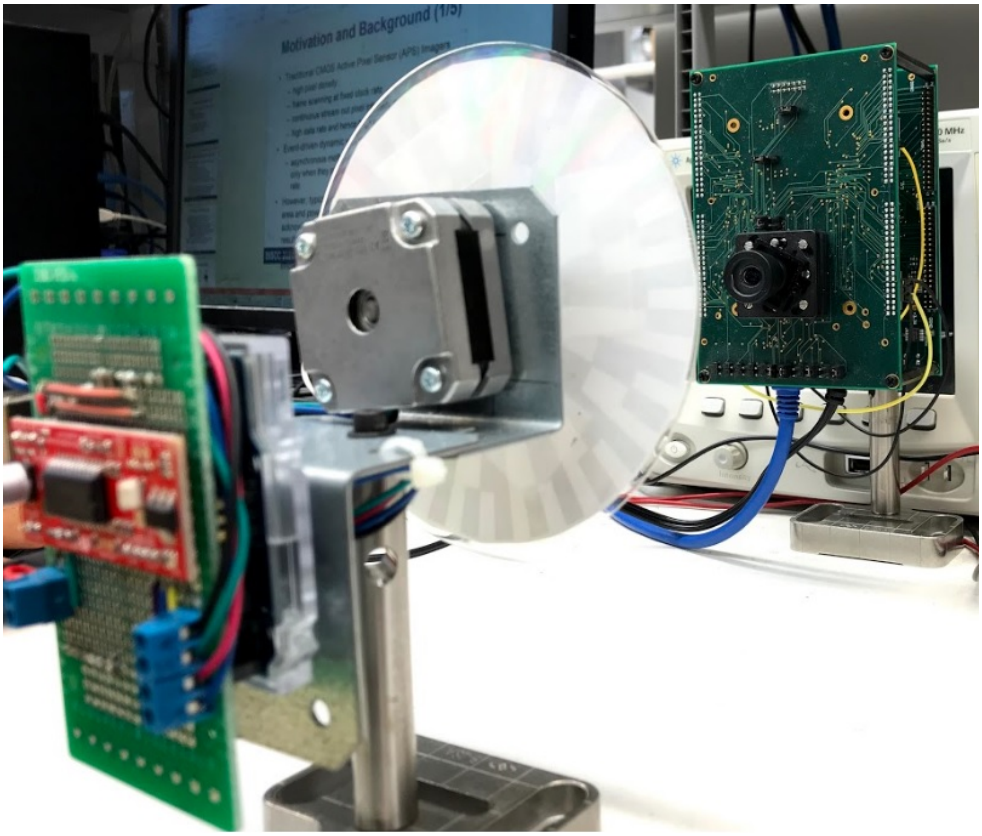


Figure 2.8: *q*DVS testbench setup.

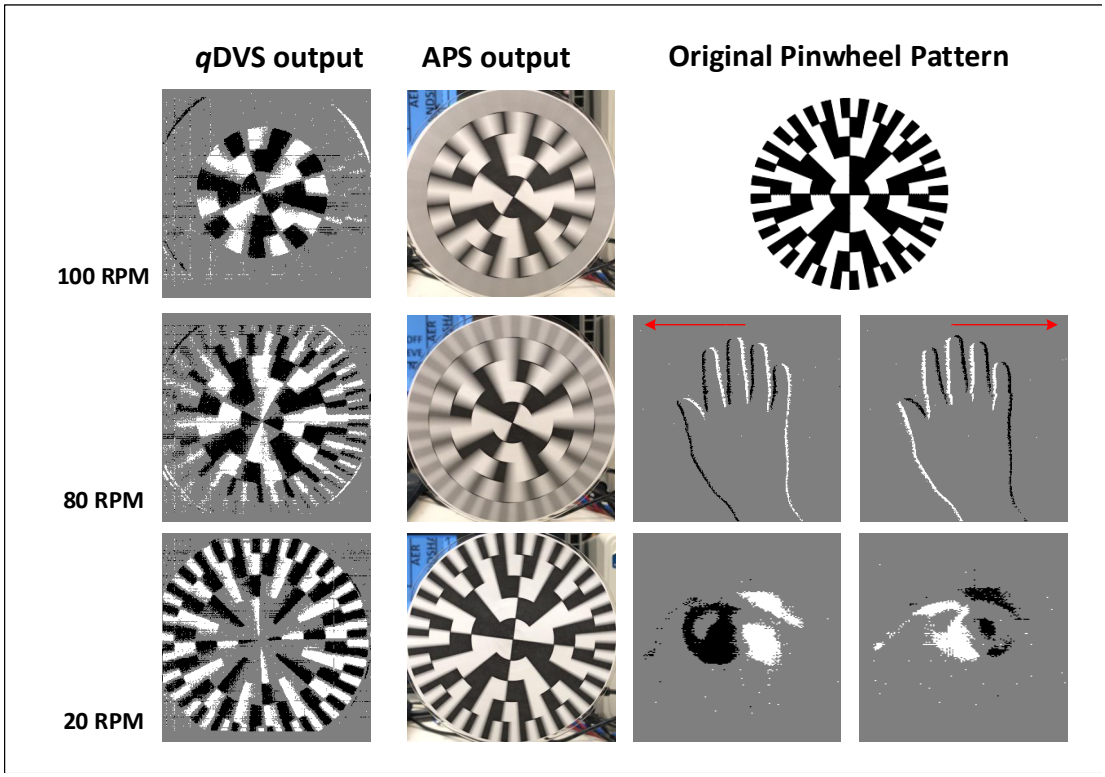


Figure 2.9: Pinwheel experiment comparing qDVS and APS camera visual events at different speeds. Snapshot of qDVS images showing handwave and eye tracking motion.

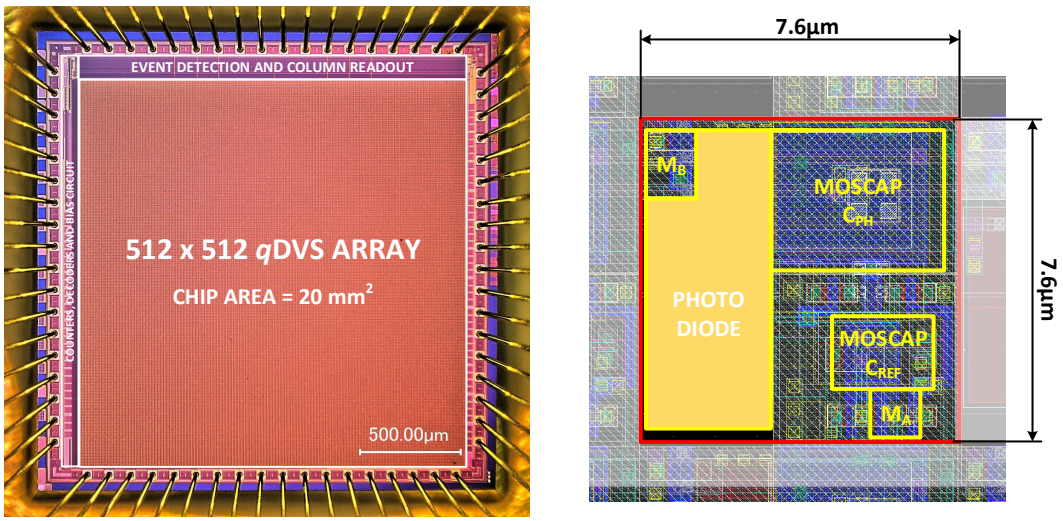


Figure 2.10: qDVS micrograph and pixel layout fabricated in 180nm CMOS technology.

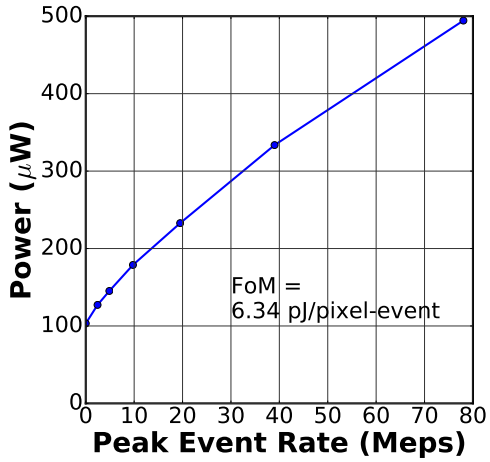


Figure 2.11: qDVS energy efficiency measured from power vs throughput plot.

Table 2.2: DVS Performance Comparison Table

Parameter	Prophesee ISSCC'20 [15]	Samsung ISSCC'17 [49]	iniVation JSSC'14 [5]	CelePixel CVPR'19 [9]	Insightness [23]	iniLabs JSSC'15 [59]	UCSD ISN JSSC'07 [11]	This Work
Technology	90nm BI CIS + 40nm CMOS	90nm 1P5M	0.18μm 1P6M	0.18μm 1P6M	65nm CIS	0.18μm 1P6M CIS	0.5μm 2P4M	0.18μm 1P6M
Resolution	1280x720	640x480	240x180	60x30	1280x800	320x262	90x90	256x256
Chip Area (mm^2)	4.86x4.86	8x5.8	5x5	3.2x1.6	14.3x11.6	5.3x5.3	3x3	2.236x2.236
Pixel Size (μ^2)	4.86x4.86	9x9	18.5x18.5	31.2x31.2	9.8x9.8	13x13	25.2x25.2	7.6x7.6
Fill Factor (%)	77	NA	22	10.3	8	22	17	33.2
Supply Voltage (V)	1.1/2.5	1.2/2.8	1.8/3.3	1.8	1.2/2.5	1.8/3.3	1.2	1.2
Power Consumption (mW) (at 100kEPS)	32/73	27	5	0.72	400	70	4.2	0.103
Energy Efficiency FoM (pJ/pixel-event)	137	340	140	7200	2857	3500	42000	6.34
Min. Contrast Sensitivity (%)	11	9	11	1	10	15	2.1	10
Dynamic Range (dB)	124	80	130	130	120	100	51	68
Latency (μs)	1.2	NA	3	NA	8	125	366-33000	5.2-1331

2.6 Discussion and Conclusions

The query-driven dynamic vision sensor (qDVS) offers lowest reported power consumption of visual event coding and streaming, normalized for process technology and pixel array resolution, compared to to other state-of-the-art prototypes from industry including Samsung, Prophesee, iniVation, CelePixel and Insightness. Table2.2 provides detailed comparison of performance metrics. Activity based event streaming to monitor pixel activity and corresponding modulation of clock frequency can reduce output rate and power even further.

2.7 Acknowledgments

Chapter 2 is in part a reprint of the material as it appears in Kubendran, R., Paul, A., and Cauwenberghs, G., "Query-driven multi-modal dynamic vision sensor with energy-efficient coding and streaming of visual information." *arXiv preprint arXiv:1601.04187*, (under review with Nature Electronics), 2020. This chapter focuses on the hardware architecture design and measured results. The dissertation author was the primary investigator and author of this paper. This study was sponsored in part by the National Science Foundation (NSF EFRI-1137279), and the Office of Naval Research (ONR MURI N00014-13-1-0205). The views and the conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Chapter 3

Event-driven Visual Cognition using Query-driven Dynamic Vision Sensor

This chapter elaborates on various applications of the query-driven dynamic vision sensor (qDVS) described in Chapter 2. The hybrid approach combining the best features of conventional active pixel sensors and event-driven dynamic vision sensors, lends itself to unique advantages in using qDVS for computer vision applications. DVS cameras have huge potential to usher in rapid adoption in edge computing AI applications, with limited resources of power and compute. This chapter is divided into sections describing different applications such as, gesture recognition, motion detection, multiple object tracking for traffic monitoring, visual attention on hardware, stereo vision for 3D reconstruction and video compression using absolute intensity and temporal contrast readout modes.

3.1 Gesture Recognition

Gesture recognition is a common and useful application in the field of computer vision. Always-on gesture detection can provide a highly interactive user experience for smart homes,

virtual or augmented reality based gaming, at shopping outlets for real-time customer tryouts and interactive monitors at other public places [33]. However, gesture recognition is usually a data intensive and compute expensive process. Using traditional frame scanning imagers, a large amount of data has to be collected and processed real-time. This involves many steps that have to be executed in the pre-processing stage to prepare the data for classification [55]. This includes background removal, edge detection and contour extraction. In particular, background removal is a hard problem involving complex algorithms to distinguish moving objects of interest amidst a static background in each frame. The major disadvantage of using high-definition cameras for gesture recognition is that background constitutes most of the sensor data which is by large redundant and unused leading to highly inefficient usage of bandwidth and power.

In order to recognize and distinguish different gestures, after pre-processing the data, the image frames are usually passed through a convolutional neural network (CNN) with many layers of convolution kernels, pooling and fully connected output stages. The number of parameters to set up such a network, including the network connection topology, bias and weights of each connection, total up to a few million parameters, hence taking large memory and bandwidth in a graphical processing unit (GPU). On the other hand, with event-driven DVS (eDVS) cameras, the throughput is significantly reduced due to transmission of event data only from the pixels generating events. But there is no efficient method to learn and classify gestures from an event stream, though there has been recent research in spiking CNN built on custom hardware accelerators to perform classification [47]. Another approach is to convert the event streams back to frames or voxel grids, which are then fed to a traditional CNN implemented on a GPU. This adds compute cost and additional steps in the pipeline [17].

In contrast, query-driven DVS (qDVS) provides event streams in ordered form, similar to frame scanning readout. Hence it is straightforward to construct the frames with sparse content and pass it to a CNN implemented with far less number of parameters. As a proof of concept, a CNN with just 6000 parameters was trained using Keras and Tensorflow in Python to achieve

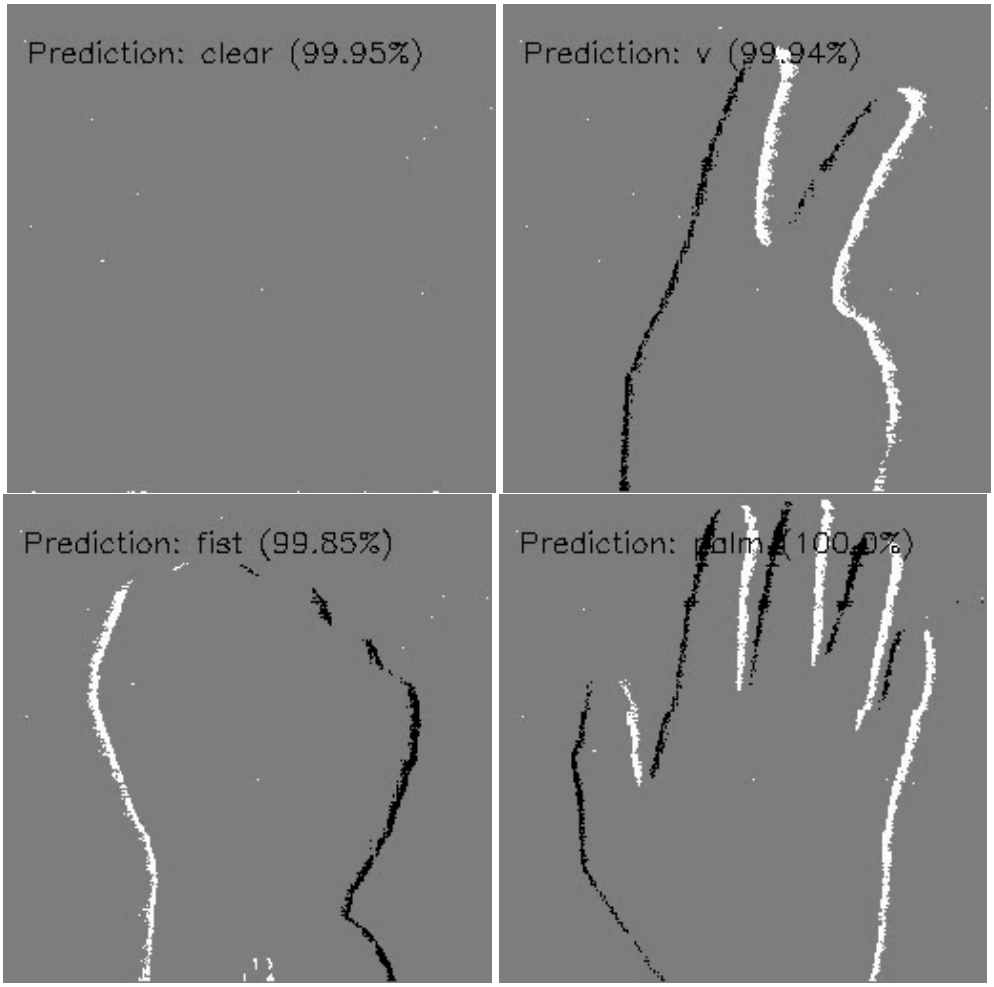


Figure 3.1: Gesture recognition using *q*DVS and software trained CNN. Prediction result and score shown on top left corner of each image.

¿99% accuracy in recognizing simple hand gestures such as palm, fist, V and okay. Training data for gestures palm, fist and V were captured using *q*DVS and later evaluated real-time with CNN prediction scores labelled on top left corner of each image. The same setup was used to demonstrate real-time inference on gesture recognition with live prediction scores as shown in Fig. 3.1.

3.2 Motion Detection

Motion detection is an important feature required in many computer vision applications. This includes predicting direction of motion of individual objects in the scene and also estimation of velocity of each object in motion. In conventional active pixel sensor (APS) cameras, the individual frames are static and do not convey any information regarding motion of the objects. Hence significant amount of post processing is required on hardware to implement real-time motion detection. This process usually involves extracting motion vectors i.e. identifying object boundaries or contours in individual frames and measuring the distance an direction moved by each contour spatially between frames. Given a fixed frame rate, it is then possible to estimate velocity of each motion vector [4].

With event-driven DVS (*e*DVS), initially a clustering algorithm is used to identify group of events that belong to each object in the scene. The clustering algorithm needs to dynamically allocate memory for each new cluster being formed and keep track of the position of all the clusters in every frame. Each asynchronous incoming event is assigned to its closest cluster (in terms of euclidean distance) in the image plane. The clustering algorithm is described in detail in [13]. Again, the event stream is converted to frames to identify object boundaries. However, it is possible to predict the direction of motion using a single frame. Since the events are positive or negative depending on detection of increase or decrease in light intensity at each pixel, the leading edges of the objects produce negative events and trailing edges of the objects produce positive events. By denoting positive events as white pixels and negative events as black pixels, the black edges in an image frame show leading edges and hence the direction of motion. In order to estimate velocity, more than one frame is still required in order to calculate the distance moved by the object, hence an asynchronous event stream provided by (*e*DVS) has to be converted to a fixed frame rate output.

In contrast, *q*DVS camera provides frames of events which are readily translated to

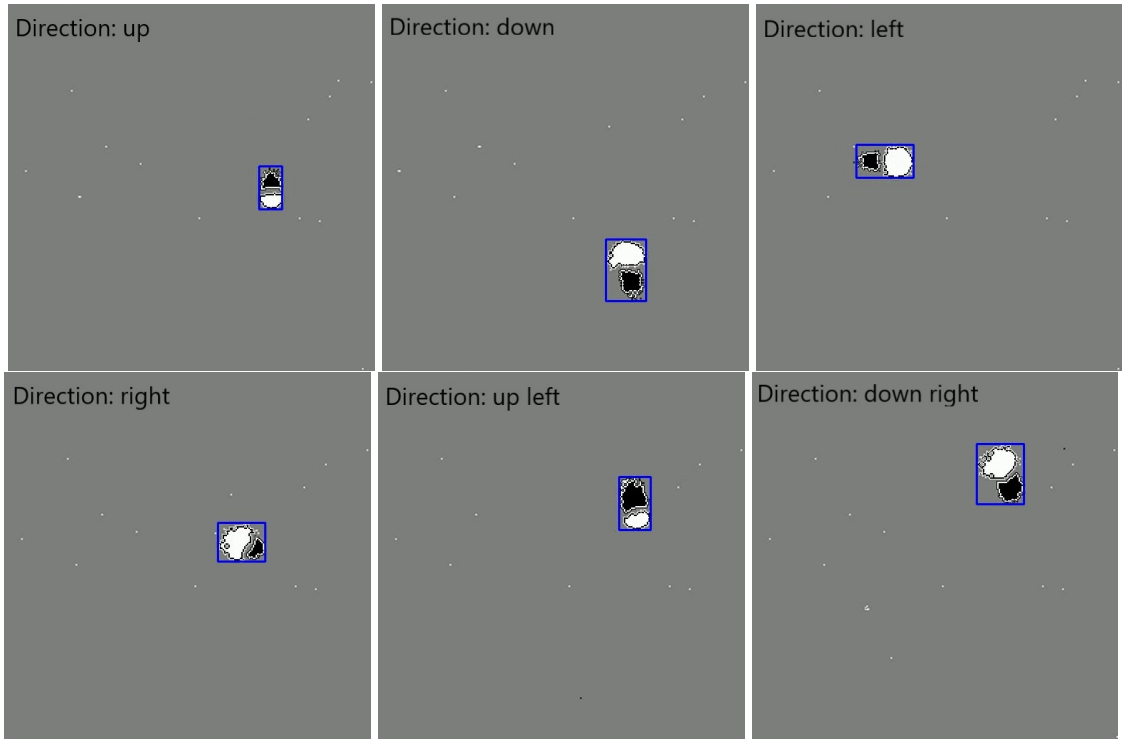


Figure 3.2: Motion detection and direction prediction using *q*DVS and software algorithm. Direction of motion is predicted and labelled on top left corner of each image.

predicting direction of motion and velocity estimation. There is no need for a clustering algorithm to identify objects. The events occurring in each frame are extracted as a sparse activity matrix in Python, which provides the row and column addresses of pixels producing events. These addresses can then be sorted and grouped into individual objects based on the assumption that each object is separated by a minimum distance (can be set as a parameter) and that its leading and trailing edges are available in consecutive rows and columns. Fig. 3.2 shows a snapshot of *q*DVS output with real-time prediction of direction of motion of the object in the scene. The algorithm can predict eight directions up, down, right, left and diagonally up_left, up_right, down_left and down_right.

3.3 Multiple Object Tracking

Tracking objects in a scene should be a straightforward extension of the previously described methods for object detection and motion estimation. Nevertheless, tracking multiple objects at the same time becomes more computationally expensive requiring more memory resources and compute time. Modern deep learning methods provide promising results in implementation with the huge compute resources and unlimited power source available at its disposal. With high-definition cameras generating terabytes of data everyday that could be processed in the cloud with unlimited wireless connectivity, even though it is achievable with today's cutting edge technology like 5G and big data server farms, it is an extremely inefficient approach that cannot scale to cater to huge markets. DVS cameras provides a much better alternative, especially as an augmented device to the existing high-definition imaging cameras available on drones, cars.

Consider the task of traffic monitoring, a classical example of multiple object tracking. In the case of APS imagers, several algorithms have to work in parallel to remove static background, extract object contours/edges using complex networks like CNN and identify individual objects by separating non-overlapping contours [60]. Using *e*DVS cameras, the task becomes more efficient, as there is no static background by design and a good clustering algorithm can both extract object boundaries and track them individually. Nonetheless, the most efficient form of traffic monitoring, and in general multiple object tracking, can be achieved using *q*DVS imagers. Fig. 3.3 presents screenshots of traffic monitoring on the Interstate-5 highway in San Diego with multiple cars identified and tracked real-time. Top left image shows a single car passing through the Interstate-5 highway in San Diego. Top right image shows two cars passing at the same time and identified separately by the algorithm. Bottom left image shows a car and a bike passing at the same time. Bottom right image shows three cars passing at the same time.

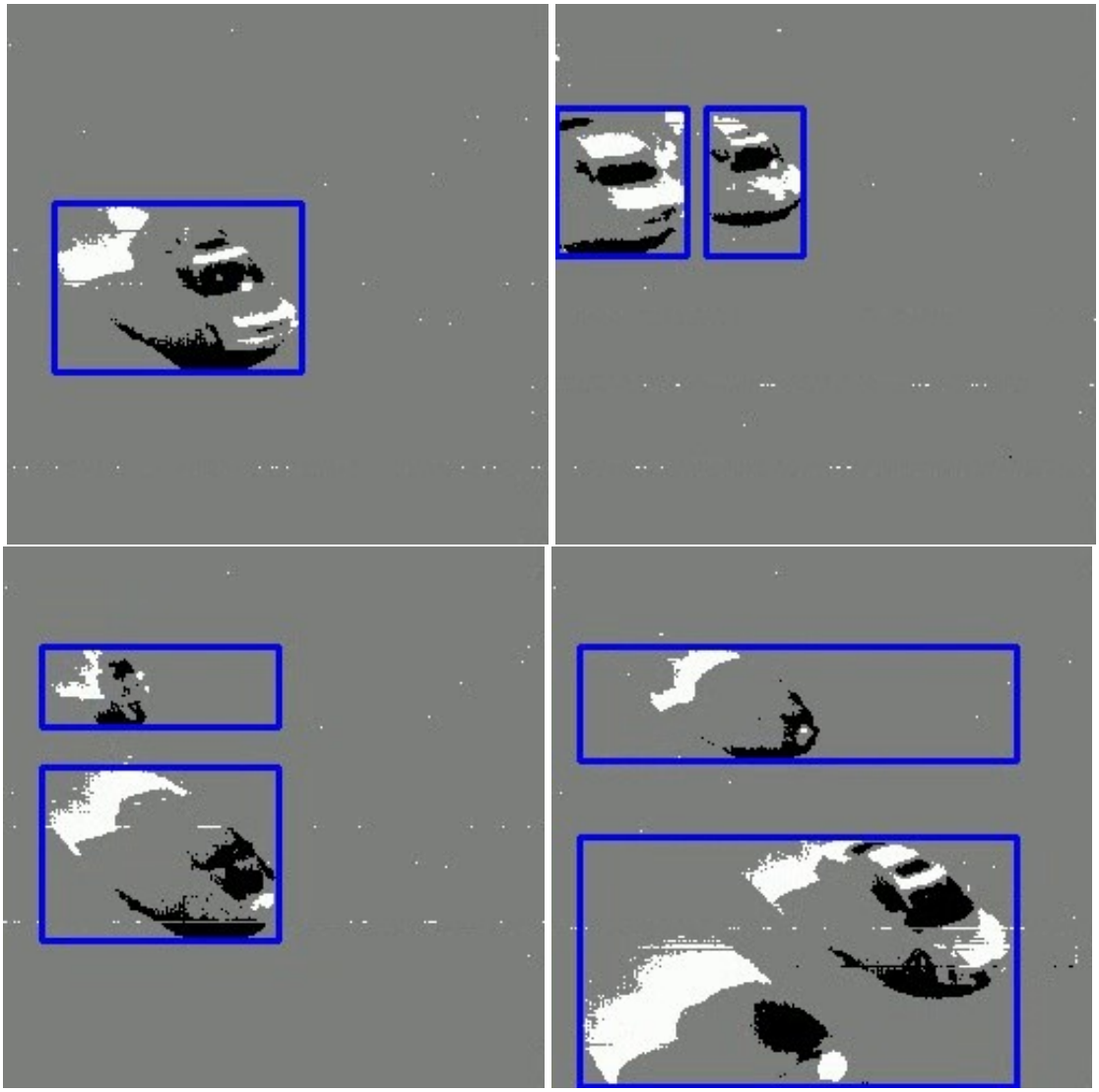


Figure 3.3: Traffic monitoring with multiple object tracking using *q*DVS and software algorithm.

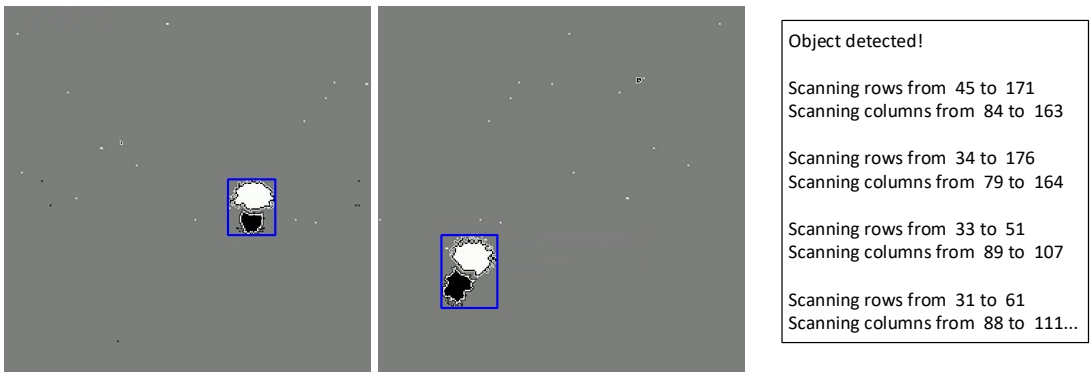


Figure 3.4: Visual attention on hardware using *q*DVS chip in the loop. FPGA program performs selective row and column scanning based on the object boundaries in the scene. Scan log shown on right.

3.4 Visual Attention

Visual attention is one of the most powerful features of biological vision, wherein it is possible to focus on a subsection or region of the visual receptive field, which can be single or multiple objects, even if there is a lot of activity in the background or the rest of the visual field. This is a remarkable feat of nature achieved by tight coupling between sensing, processing and feedback pathways of the entire retinal and visual cortical signaling pipeline. There has been a lot of interest in academia and industry to achieve visual attention in software using computer vision algorithms [56]. However, many real-world applications like autonomous drone navigation and self-driving cars would need visual attention implemented in a highly resource constrained environment with limited power supply and latencies in the order of sub-milliseconds (sub-ms) for fast reaction times. Even small margins of error can prove to be fatal. Asynchronous temporal event streams from *e*DVS do not readily tend to selective visual attention since the spatial context is completely missing and has to be created in software. Though the overall data rate is significantly reduced, there is still lot of post-processing to be done to select an object of interest. After that, subsequent events can be filtered based on proximity to the location of the object of interest, thus emulating visual attention. In contrast, *q*DVS offers a straightforward approach for visual attention using selective row and column scanning. In normal operation, the

*q*DVS sensors scans all rows one by one and reads out the columns sequentially. The finite state machine (FSM) implemented on FPGA hardware provides the clocks for on-chip counters and decoders to select particular rows and columns. For implementing visual attention completely on hardware, with chip in the loop, the FSM can be modified to select a sub-set of rows and columns based on the object boundaries derived by motion detection algorithms described in the previous sections in this chapter. Hence, only a subset of pixels in the frame are queried for event detection, reducing the output data rate and power consumption even further. Fig. 3.4 presents screenshots of object tracking with visual attention turned off (left) and on (right). In this particular case, the power consumption and data rate were reduced by 15% and 20% respectively. In general, the savings in power and throughput depends on the activity in the scene. The sparser the activity, higher the savings.

3.5 Stereo Vision

Stereo vision is the ability to construct an unified view of an object or scene, with two different vision sensors/cameras [41]. Based on the binocular disparity between the left and right images of an object, it is possible to estimate the depth of the object. This requires identifying and matching features in the left and right images of the same point in the visual scene that helps to derive information about how far away objects are, based solely on the relative positions of the object in the two sensors [41]. A simplified form of stereo vision can be implemented using the OpenCV library with Python that takes in the individual image frames generated by the left and right cameras (can be APS, *e*DVS or *q*DVS), and produces an output disparity map frame that can be used to esimate relative depth of objects in the scene. Fig. 3.5 shows an example of stereo vision with left and right input frames from *q*DVS sensors and output disparity map shown on the right. Further study is planned to implement 3D image reconstruction using DVS cameras for simultaneous localization and mapping (SLAM). This is an active area of research pursued by

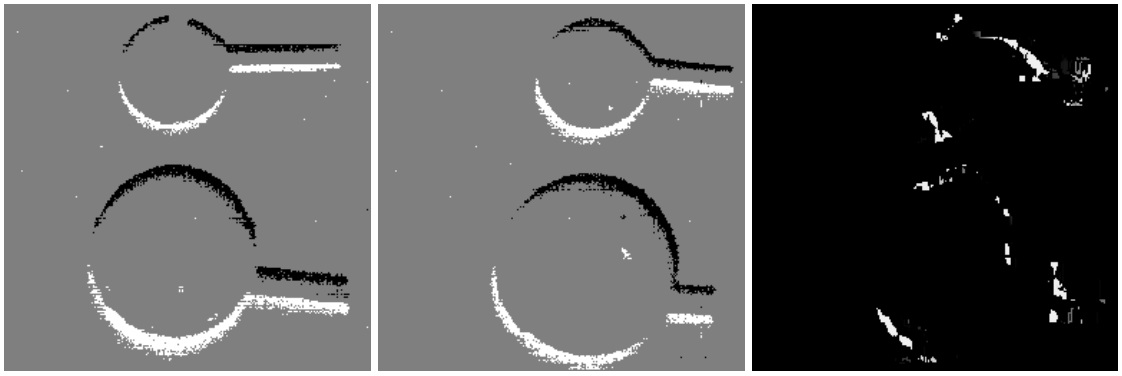


Figure 3.5: Stereo vision using two *q*DVS cameras and OpenCV software algorithm to create disparity map. Images captured by the left and right *q*DVS cameras are shown followed by the disparity map.

industry and academia [59, 2].

3.6 Video Compression

Raw video frames generated by any camera has to be processed and encoded into a compressed data stream using a video encoder. There are several standards available for video encoding, Moving Picture Experts Group (MPEG2 or MPEG4), H.264 to name a few [42]. A common approach to video compression is to convert the raw frames into sparse I-frames (Intra-coded) consisting of the raw pixel data, interpolated with many P-frames (Predicted), consisting of only the pixel value differences. This removes encoding redundant background every frame. Motion vector estimation is used to further reduce the data rate by object tracking based on their estimated direction and velocities. Video compression techniques require large computational resources including dedicated hardware such as micro-controllers to control timing logic for raw video capture and digital signal processors (DSP) to implement convolutional filters to compress and encode the visual information. *e*DVS imagers that can operate in absolute intensity readout mode, such as DAVIS or ATIS [3] can provide the raw data to generate I frames and P frames required for the video encoder to compress the data stream. However, additional processing is involved to accumulate events, with variable latency, within a time window, based on the

Table 3.1: Comparison of Implementation steps for Event-based vs Frame-based Computer Vision Applications

Application	Active Pixel Sensors (APS)	Event-driven Dynamic Vision Sensors (*e*DVS)	Query-driven Dynamic Vision Sensors (*q*DVS)
Gesture Recognition	Background removal Contour detection Large CNN parameters	Frame reconstruction Contour detection Small CNN parameters	Contour detection Small CNN parameters
Motion Detection	Object detection Motion vector estimation Predict direction based on vectors	Frame reconstruction Object detection in single frame Predict direction based on edge contrast	Object detection in single frame Predict direction based on edge contrast
Object Tracking	Background removal Object detection every frame Track contours	Event accumulation in memory Dynamic clustering Track cluster centers	Object detection every frame Track contours
Visual Attention	Background removal Object detection every frame Label objects of interest Attention using selective region scanning	Event accumulation in memory Dynamic clustering Label clusters of interest Attention using cluster center tracking	Object detection every frame Label objects of interest Attention using selective region scanning
Stereo Vision	Large data and compute from two sensors Left and right images of object Disparity map generation Depth estimation	Small data and compute from two sensors Left and right images of object Frame reconstruction Disparity map generation Depth estimation	Small data and compute from two sensors Left and right images of object Disparity map generation Depth estimation
Video Compression	Generate I-frames and P-frames using frame scanning and temporal differencing Motion vector estimation Encode compressed data stream	Reconstruct I-frames and P-frames using output event stream Motion vector estimation Encode compressed data stream	Generate I-frames using intensity readout and P-frames using event readout modes Motion vector estimation Encode compressed data stream

target frame rate. On the other hand, *q*DVS imager can use its two scanning modes alternatively, 6bit frame scanning mode described in Chapter 2 to directly provide the I frames and the event scanning mode to directly generate the P frames. This streamlines the pipeline, at a fixed frame rate, without any additional processing bottleneck. Future work is to implement a video encoding standard, such as the MPEG4, using the *q*DVS imager.

3.7 Conclusion

Table3.1 provides a summary of steps involved in event-based and frame-based computer vision applications described in this chapter. Dynamic vision sensors offer a clear advantage over traditional frame-based cameras for all of these applications, where relative temporal contrast changes in pixel intensity matter more than the actual pixel content. In particular, *q*DVS can achieve further gains, compared to *e*DVS, in pixel density and push the envelope of energy efficiency even lower, while able to provide sufficiently fast response times in most cases. There is a significant reduction in compute memory, processing and power and hence overall energy efficiency, making DVS suitable for low power edge computing and Internet of Things (IoT) applications. ———————————————————————————————

3.8 Acknowledgements

Chapter 3 is in part a reprint of the material as it appears in Kubendran, R., Paul, A., and Cauwenberghs, G., "Query-driven multi-modal dynamic vision sensor with energy-efficient coding and streaming of visual information." *arXiv preprint arXiv:1601.04187*, (under review with Nature Electronics), 2020. This chapter focuses on the applications of the vision sensor described in Chapter 2. The dissertation author was the primary investigator and author of this paper. This study was sponsored in part by the National Science Foundation (NSF EFRI-1137279), and the Office of Naval Research (ONR MURI N00014-13-1-0205). The views and the conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Chapter 4

Asynchronous Analog-to-Digital Converter with Hybrid Two-Tier Level-Crossing Event Coding

4.1 Introduction

Data converters have been a backbone hardware behind the digital revolution, interfacing natural continuous-time systems seamlessly with discrete-time digital systems. Digital systems have inevitably been based on a synchronizing clock to uniformly sample and process signals with discrete uniform time stamps. However, most natural systems are asynchronous and continuous time with no explicit clock to govern the timing. Recently, there has been increased interests in understanding and modeling such asynchronous systems in the digital domain because of many inherent advantages like improved energy efficiency, lower latency and input activity dependent throughput [32]. Towards this end, there has been exploration of several asynchronous data converter architectures [61, 44, 50, 54].

Many biological signals have sparse information with mostly uneventful periods and

short period of event bursts with high information content, such as electrocardiogram and neural spikes. Uniform sampling of such signals results in a highly inefficient use of resources, by converting and storing unnecessary data as well as losing information during high frequency events. Asynchronous ADCs are more suitable to acquire such signals where the number of output samples are proportional to the input frequency and hence the information. In particular, level-crossing ADCs have an additional advantage of using the same energy per level conversion of both low-frequency large-amplitude signals and high-frequency low-amplitude signals.

Asynchronous ADCs used for single channel biomedical signal acquisition, like electrocardiography (ECG), is optimized for ultra-low power operation since the signals are sparse. However, for signals with dense activity, like electroencephalography (EEG), or in event driven computer vision to read out pixel intensities in a serial frame scanning method, there is a need to multiplex and stream high throughput data continuously. The proposed ADC architecture is designed to cater to these applications.

The Asynchronous level-crossing ADC presented in this paper was a part of an asynchronous imager chip with temporal contrast threshold detection and simultaneous random-access digital readout using the ADC [37]. The proposed ADC is completely clock-free, unlike other pseudo-asynchronous ADC architectures reported in recent literature [19, 51], which use a global clock and event-driven clocking techniques to update the output depending on the sparsity of the input signal. Also, by using a hybrid two-tier design, the proposed ADC has a higher resolution than other level crossing ADCs reported previously [61, 44, 50, 54]. The paper is organized as follows. Section 4.2 presents an overview of the ADC architecture to explain its operation. In Section 4.3, the circuit implementation of the chip is described in detail. The measurement results of the ADC are presented in Section 4.4, and Section 6.5 concludes the paper with a comprehensive summary.

4.2 ADC Architecture

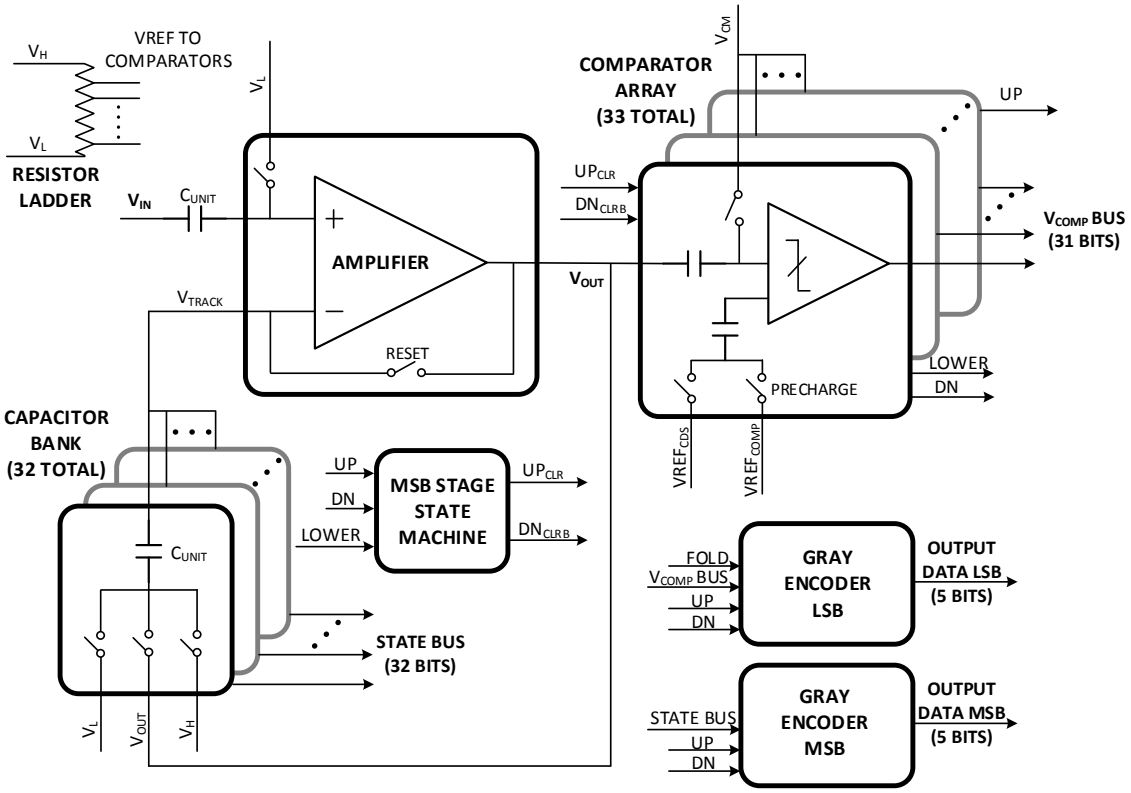


Figure 4.1: Asynchronous ADC Block Diagram

Fig. 4.1 shows an architectural block diagram of the proposed two-tier hybrid ADC. The input analog voltage, V_{IN}, is fed to the positive input of the amplifier. The negative input terminal is connected to a tracking voltage, V_{TRACK}, generated by a 32-level capacitor bank. V_{TRACK} tracks V_{IN} using the feedback from the amplifier output through the capacitor bank. The amplifier output, V_{OUT}, is settled to make sure that V_{TRACK} and V_{IN} are equal, and is compared with 33 comparators, which generate 5-bit LSB level-crossing ADC outputs. The 5-bit MSBs are determined by the state of the capacitor bank. The overall 10-bit ADC outputs are converted to gray-coded outputs by Gray Encoders.

4.2.1 MSB Stage

The 5-bit MSBs are determined by the connectivity of the 32 capacitors in the bank. The capacitors are arranged in parallel, and each of them is made of the Metal-Insulator-Metal (MiM) capacitor with size of 20 fF. Their top plates are all connected together, and the bottom plate of each capacitor is connected to a 3-way switch to connect it to one of 3 different voltages: V_H, V_L and V_{OUT}. V_H and V_L are the high and low reference voltages that set the input range of the ADC.

Initially, all the capacitors are reset to zero. After the reset, the bottom plate of the first capacitor is connected to V_{OUT} and the rest are connected to V_L. When V_{IN} increases, V_{OUT} also increases with a 32-times gain. As V_{OUT} reaches V_H, which is the upper boundary, the first capacitor is connected to V_H and the second capacitor is now connected to V_{OUT}. As V_{IN} increases further, more capacitors are connected to V_H

When M number of the capacitor units are connected to V_H, the $(M+1)^{th}$ capacitor is connected to V_{OUT} and the rest to V_L. The resulting V_{TRACK} can be represented as follows:

$$V_{TRACK} = \frac{M}{32}(V_H - V_L) + \frac{1}{32}(V_{OUT} - V_L) + V_L. \tag{4.1}$$

When V_{OUT} crosses V_H, then the $(M+1)^{th}$ capacitor, which was connected to V_{OUT}, is now connected to V_H and the next capacitor ($(M+2)^{th}$) is newly connected to V_{OUT}. When V_{OUT} crosses the low boundary, the number of capacitors connected to V_H is decreased by 1 and the capacitor that is connected to V_{OUT} is moved to left. By doing so, V_{TRACK} tracks the ADC input seamlessly. Here, the number of capacitors connected to V_H is consequently the thermometer code of the 5-bit MSBs.

The switch signals of the capacitor bank are generated by a digital logic based on a 32-bit shift registers. Its output is shifted to the left or right by *UP* and *DN* signals, respectively. *UP* and *DN* signals, which indicate boundary crossing of V_{OUT} over V_H and V_L, respectively, are generated by the two boundary comparators, which compares V_{OUT} with V_H and V_L at the LSB

stage.

4.2.2 LSB Stage

As described above, V_{OUT} is generated by the feedback of the amplifier and the capacitor bank, and represented by:

$$(V_{OUT} - V_L) = 2^5 \left[(V_{TRACK} - V_L) - \frac{M}{2^5}(V_H - V_L) \right] \tag{4.2}$$

where V_{TRACK} is equivalent to the ADC input voltage V_{IN} and M is the thermometer code of the MSBs ranged from 0 to 2^5. Hence, V_{OUT}, referenced to V_L, is the residue of the input voltage after subtraction of the MSBs. At the output of the amplifier, V_{OUT} is continuously quantized to the 5-bit LSBs by 33 continuous-time comparators, which compare V_{OUT} with 33 reference voltages between V_H and V_L, generated by a resistive ladder. The comparator outputs are the corresponding thermometer code for the 5-bit LSBs of the ADC.

The two comparators at both boundaries compare V_{OUT} with V_H and V_L, respectively, to check that V_{OUT} is within the range. When V_{OUT} gets out of the range, UP or DN signals are raised and a left or right shift of the capacitor bank takes place according to the signals. Whenever a level crossing occurs, the UP_{CLR} and DN_{CLR} signals are generated to reset V_{OUT}.

Continuous Time Flash ADC has 32 comparator latches to compare the output of the first stage to 32 reference voltage generated using a resistive ladder. The Latch is initially precharged to perform a Correlated Double Sample (CDS) where in the input voltage is capacitively coupled to a CDS Reference voltage followed by Resistive ladder reference voltage. The output of the latch then generates the COMP output state along with UP and DN state variables. The state variables are cleared by handshaking with the first stage.

4.2.3 Output Stage

The 32-bit thermometer codes from the capacitor bank and the 32-bit thermometer codes from the comparators are converted by a gray encoder with bubble corrections to the 10-bit outputs of the ADC. The UP_{CLR} and DN_{CLR} signals are used to generate the *FOLD* variable which flips the output bit 4, the highest bit of the LSB stage, in order to seamlessly transition the input tracking between the MSB and LSB stages. STATE variable from the first stage and the 32 bit COMP variable from the second stage are used by the Gray Encoder with Bubble Correction to generate the output 10 bits of the ADC.

4.2.4 Overall Operation

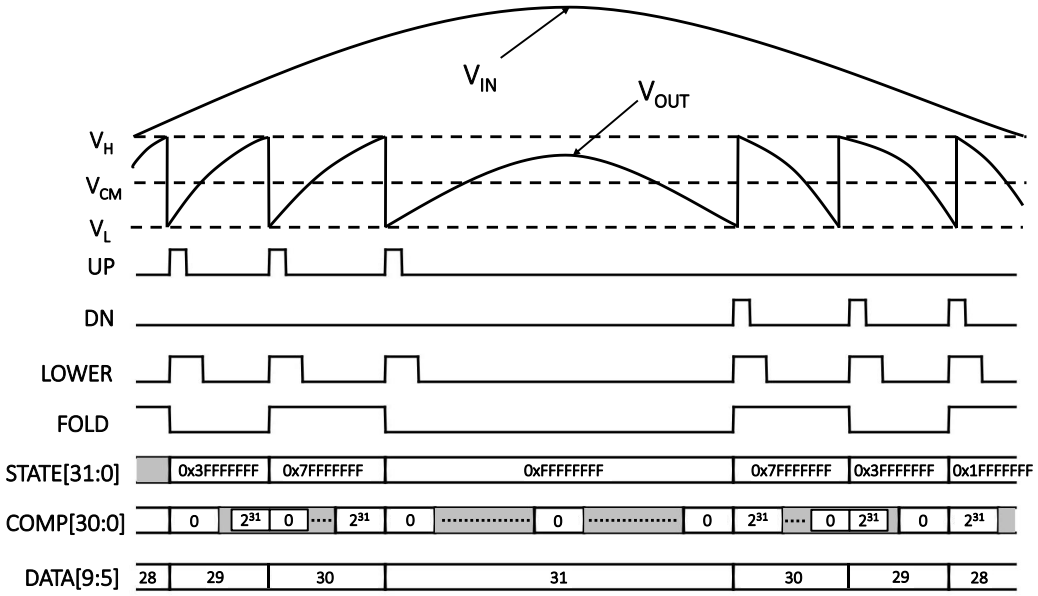


Figure 4.2: Timing diagram of the two-tier asynchronous ADC operation

Fig. 4.2 shows timing waveforms of the main analog voltages and internal state variables of the ADC. As V_{OUT} reaches V_H, *UP* is raised and *STATE* and DATA[9:5] changes to the next number. When V_{IN} decreases and V_{OUT} crosses V_L, it raises *DN* to lower down *STATE*. *LOWER* variable triggers the controller of the MSB stage, indicating the timing to increase or decrease

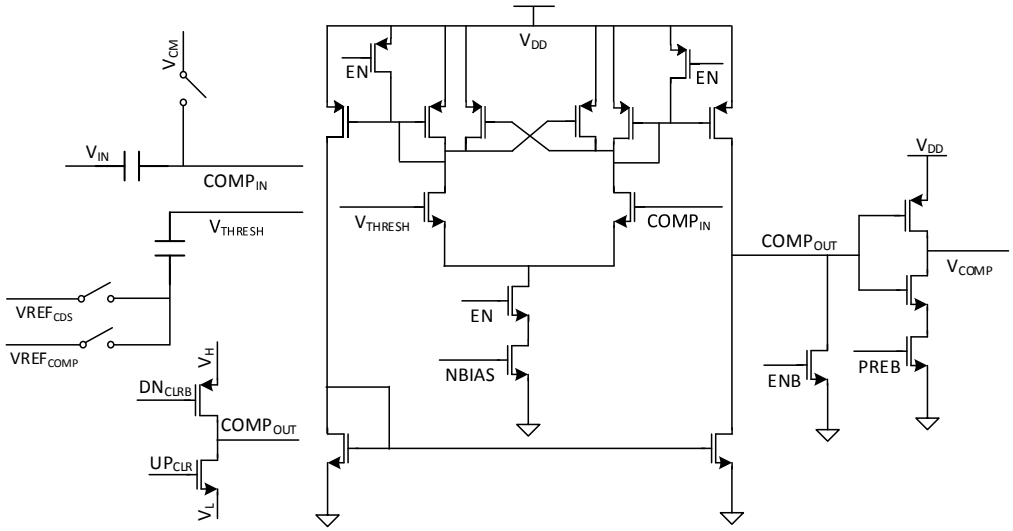


Figure 4.3: Circuit diagram of comparator of the LSB Stage

STATE value. As shown in the figure, the *DATA* bus reaches maximum near the peak swing of the input signal V_{IN} where the *STATE* and *COMP* variables have also reached their maximum value.

4.3 Circuit Implementation

The amplifier in the MSB stage is made up of a folded cascode op-amp with internal bias circuitry to achieve high gain and bandwidth. The reset switch shorts its single-ended output, V_{OUT}, to the negative input terminal, V_{TRACK}, during the reset phase. During reset, V_{IN} is also set to V_L, thus setting the DC baseline voltage of V_{IN} as well as V_{TRACK}.

A circuit diagram of the the comparators in the LSB stage are shown in Fig. 4.3. The inputs of the comparators are capacitively coupled, with one input connected to the output of the MSB stage and the other input to a reference voltage. Each of the comparators is initially precharged to perform Correlated Double Sampling (CDS). The CDS technique is used to compensate for transistor mismatches and temperature variations in the architecture. During the reset phase, $COMP_{in}$ is connected to V_{CM} to save any mismatches in the input, which is coupled to V_{OUT} as shown in Fig. 4.1. The other node that is coupled to V_{THRESH} connects to a CDS reference voltage,

$VREF_{CDS}$. When the comparison gets started, a resistive ladder reference voltage, $VREF_{COMP}$ is applied to the comparator. The output of the comparators then generates a *COMP* output state along with *UP* and *DN* state variables. The state variables are cleared by handshaking with the MSB stage.

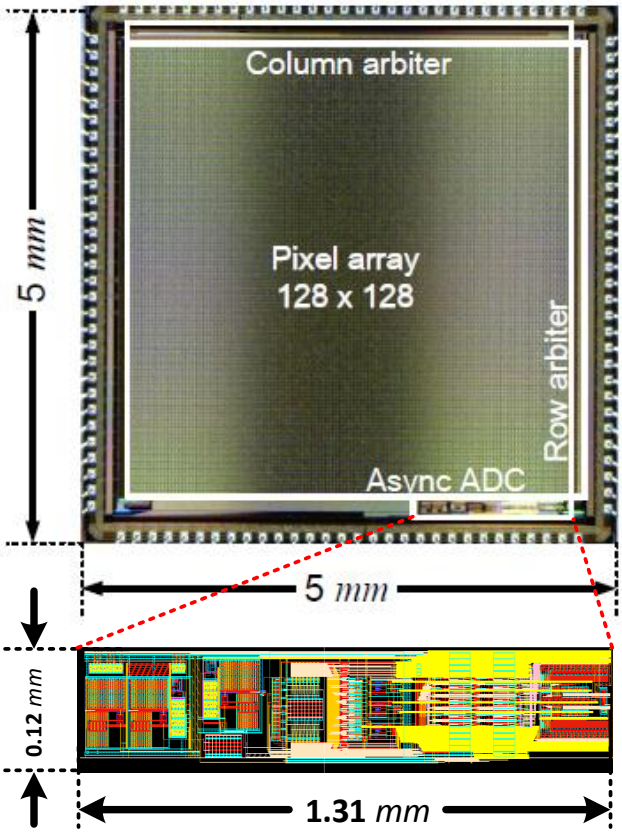


Figure 4.4: Chip micrograph and layout of the asynchronous LC-ADC.

4.4 Measurement Results

The two-tier level-crossing asynchronous ADC was designed and fabricated in 180-nm CMOS technology, as part of an asynchronous event-driven imager. Fig. 4.4 shows the chip micrograph along with a zoom-in view of the ADC layout. Active area of the ADC is approximately 0.126 mm^2 including input SPI and probe circuitry, with the capacitor bank and

comparator array dominating the area. The ADC operates at 1.8V supply for both digital and analog blocks.

Fig. 4.5 and Fig. 4.6 show measured waveforms of the ADC for an input voltage ramp and sine wave signal at 10kHz. The oscilloscope traces show buffered input voltage V_{IN}, output of the 5-bit folded residue stage V_{OUT}, and the digital output bits of the ADC that are gray encoded. The asymmetry of V_{OUT} at the peaks and valleys of V_{IN} can be attributed to the non-linearity of the residue amplification stage for large input swing and mismatch in the latency of transition between MSB and LSB stages. The ADC was configured through a Xilinx FPGA board to program bias voltages and currents, send commands to the chip, and receive data from the chip.

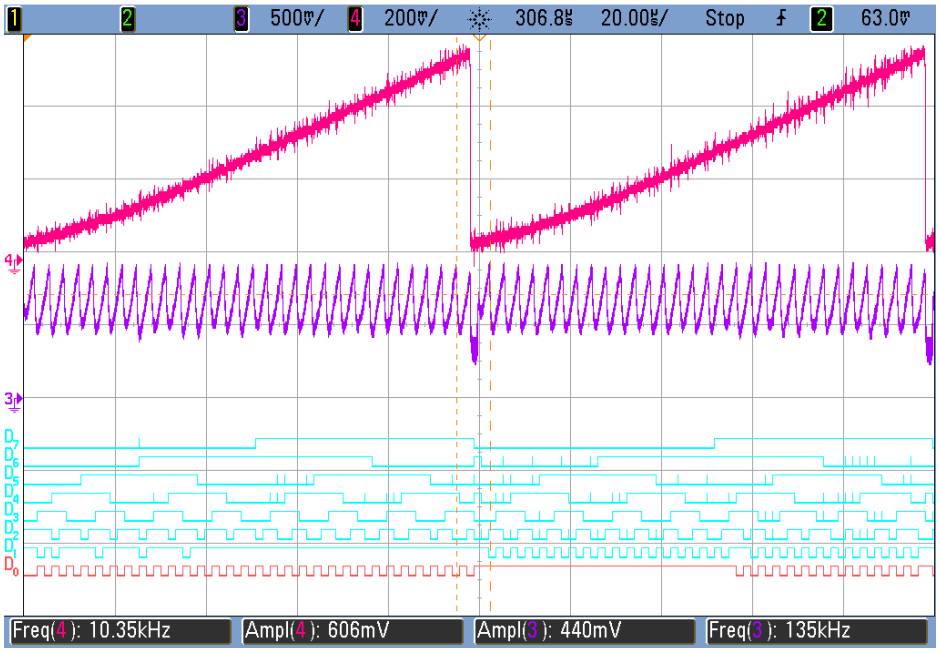


Figure 4.5: Oscilloscope recorded analog waveforms V_{IN} (red) and V_{OUT} (purple), and gray-coded digital ADC outputs (green) for (a) a linear ramp input, and (b) a sinusoidal input V_{IN}.

The ADC output data were sent through USB to a PC where the waveform was reconstructed and the frequency spectrum analysis was performed. Fig. 4.7 shows the FFT spectrum of the reconstructed signal of the ADC for a 20 kHz 0.4 Vpp sinusoidal input. Since the USB readout is synchronous, the ADC output is sampled at a fixed clock rate, which contributes to

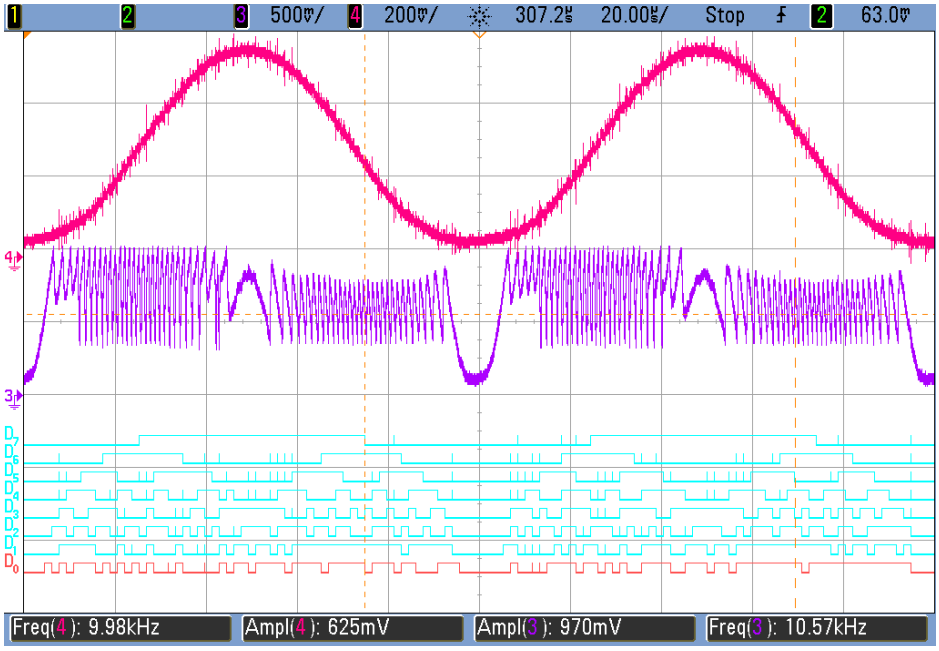


Figure 4.6: Oscilloscope recorded analog waveforms V_{IN} (red) and V_{OUT} (purple), and gray-coded digital ADC outputs (green) for (a) a linear ramp input, and (b) a sinusoidal input V_{IN}.

the noise floor and jitter seen in the spectrum. This spectrum was used to compute the Signal-to-Noise-Distortion-Ratio (SNDR) of the ADC and the Effective-Number-of-Bits (ENOB). The ENOB is then used to estimate the Figure-of-Merit (FoM) of the asynchronous ADC:

$$FoM = \frac{E_{conv}}{2^{ENOB}} \tag{4.3}$$

where E_{conv} is the energy consumed per conversion, for each registered level-crossing event. This definition is consistent with the usual FoM for a Nyquist-rate ADC, with a conversion energy given as the ratio of power over sampling frequency, $E_{conv} = P \,/\, f_{sample}$. The performance metrics of the proposed ADC are summarized in Table 4.1 in comparison with other asynchronous ADCs reported in the literature.

The use of the hybrid two-tier level crossing scheme offers a greater FoM than an equivalent flat hierarchy, which for 10-b conversion would have required 1,024 level-crossing

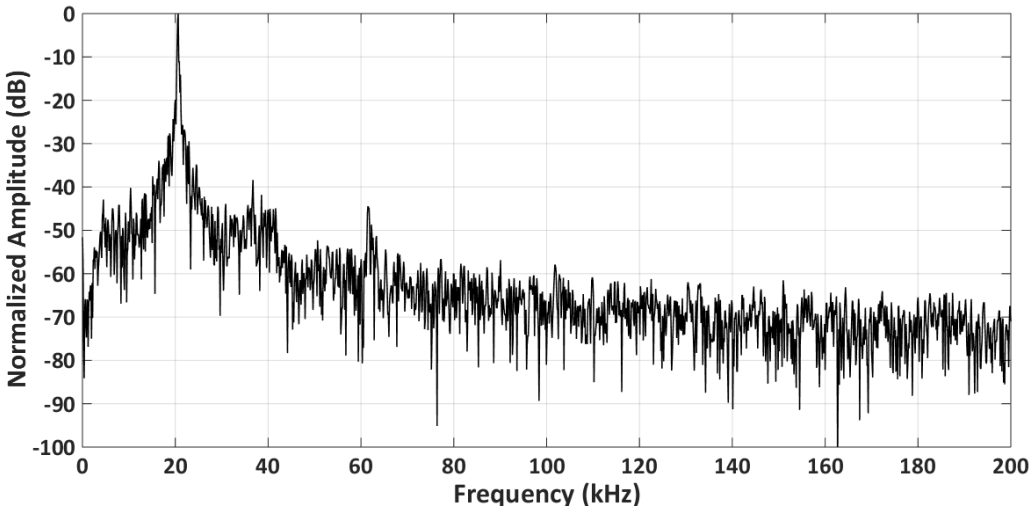


Figure 4.7: FFT spectrum of the ADC output for a 0.4 Vpp 20 kHz sine wave input.

Table 4.1: Performance Comparison of Asynchronous ADC architectures

Parameter	[61]	[44]	[50]	[54]	This Work
Technology	180nm CMOS	90nm CMOS	180nm CMOS	130nm CMOS	180nm CMOS
Supply Voltage	0.8 V	1 V	0.7 V	0.8 V	1.8 V
Resolution	6 bits	8 bits	4 - 8 bits	4 - 8 bits	10 bits
SNDR	40 - 49 dB	47 - 62 dB	43.2 dB	47 - 54 dB	49 - 57 dB
Input Bandwidth	5 Hz - 3.3 kHz	200 Hz - 4 kHz	1 Hz - 1.1 kHz	20 Hz - 20 kHz	1 Hz - 200 kHz
Full Scale Input Voltage	1.6 Vpp	0.5 Vpp	1.4 Vpp	0.72 Vpp	0.6 Vpp
Power Consumption	0.58 μ W	40 μW	25 μW	7.4 μ W	160 - 426 μW
FoM	565 fJ/conv.	27.3 pJ/conv.	106 pJ/conv.	880 fJ/conv.	1.57-4.16 pJ/conv.
Active Area	0.045 mm^2	0.06 mm^2	0.96 mm^2	0.36 mm^2	0.126 mm^2

detectors, at substantially greater energy than the 32 comparators used here, despite the power overhead of the folding residue amplification input stage.

Fig. 4.8 shows input and output waveforms of the MSB and LSB stages of the ADC for a sample ECG signal. As shown in the figure, the number of samples are significantly higher during the QRS complex compared to the rest of the ECG signal, including the P-Wave and T-Wave.

Figure 4.8: Sample ECG waveform recorded on oscilloscope: V_{IN} (red) and V_{OUT} (purple), and gray-coded digital ADC outputs (green).

4.5 Acknowledgments

Chapter 4 is in part a reprint of the material as it appears in Kubendran, R., Park, J., Sharma, R., Kim, C., Joshi, S., Cauwenberghs, G., and Ha, S., "A 4.2-pJ/conv 10-B Asynchronous ADC with Hybrid Two-Tier Level-Crossing Event Coding." *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020. The dissertation author was the primary investigator and author of this paper.This study was sponsored in part by the National Science Foundation (NSF CCF-1317373), and the Office of Naval Research (ONR MURI N00014-13-1-0205). The views and the conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Chapter 5

CMOS-RRAM Compute-In-Memory Architecture with Multimodal Integrate-and-Fire Neurons and Dynamically Reconfigurable Synapses

5.1 Introduction

Deep learning methods have accelerated the adoption of artificial neural networks (ANN) in many applications, spanning and impacting every field that requires automation. This has resulted in a tremendous demand for hardware accelerators, primarily graphical processing units (GPU) and recently field programmable gate array (FPGA) based systems and custom hardware. There has been an escalated interest in both industry and academia, in developing robust, energy efficient and re-configurable hardware.

On the other hand, neuromorphic systems have gained significant prominence recently, as it holds promise to extreme energy efficiency and low latency, which is suitable for edge

computing and Internet-of-Things (IoT) applications[22]. Inspired by biophysical principles, the neuron design can use noise and mismatch variations to its advantage, while generating and processing spike-based events that can be efficiently implemented using switched-capacitor circuits and techniques for analog implementations [39, 36, 37] and advanced technology nodes for digital implementations [1, 12, 38]. In conventional von-Neumann architectures, there is a

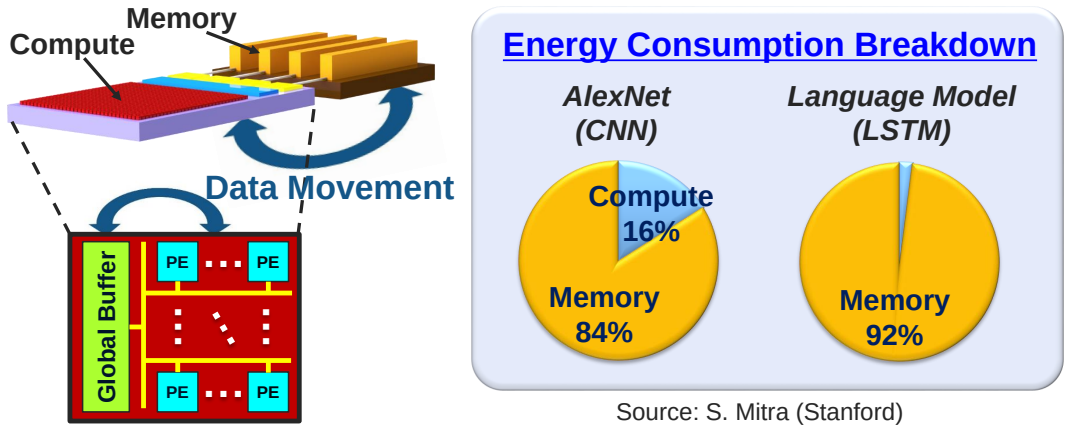


Figure 5.1: Memory wall in conventional von-Neumann architectures.

memory wall bottleneck that prevents energy-efficient data movement, since the memory and computer are physically separated and connected through a high-speed bus interface as shown in Fig.5.1. Hence, there has been great interest to develop compute-in-memory architectures that can bring memory closer to compute through tight physical integration using innovative fabrication technologies. [6] A major challenge here is that the memory has to be of higher density than the compute area, especially for high dimensional deep learning networks or high fan-in neuromorphic topologies. Traditional memory unit cells like static random access memory (SRAM), dynamic random access memory (DRAM) constitute many transistors to store 1 digital bit and hence are not scalable for computer-in-memory architectures. Many emerging and beyond CMOS devices are proposed as candidates for high-density memory design such as, resistive random access memory (RRAM), spin-transfer torque magnetic RAM (STT-MRAM)

and other devices with memristive properties, i.e. the resistance is dependent on the history of the voltage applied across the device. Among these candidates, CMOS-RRAM integrated

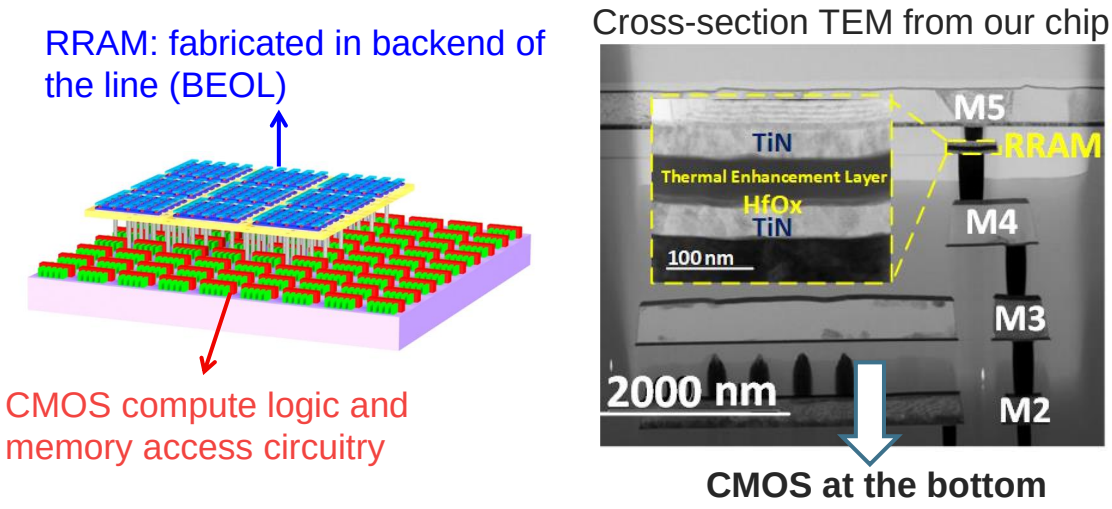


Figure 5.2: Compute-In-Memory illustration with emerging device.

compute-in-memory architecture has proven to be practically feasible, reliable and scalable to large networks. Fig.5.2 shows a depiction of the RRAM crossbar array fabricated on top of existing CMOS compute logic, including a cross-section of transmission electron microscope (TEM) image.

Many powerful neural network (NN) models such as probabilistic graphical models (PGMs) and recurrent neural networks (RNNs) require flexibility in dataflow and weight access patterns as shown in Fig. 5.3. Typically, CIM designs do not implement such dataflows or do so by replicating circuits at the memory periphery such as ADCs/neurons along both the rows and columns of the memory array, leading to an overhead in operation. This chapter describes a CIM architecture implemented in a 130-nm CMOS/RRAM process, that delivers the highest reported computational energy-efficiency of 74 TOPS/W for RRAM-based CIM architectures while simultaneously offering dataflow reconfigurability to address the limitations

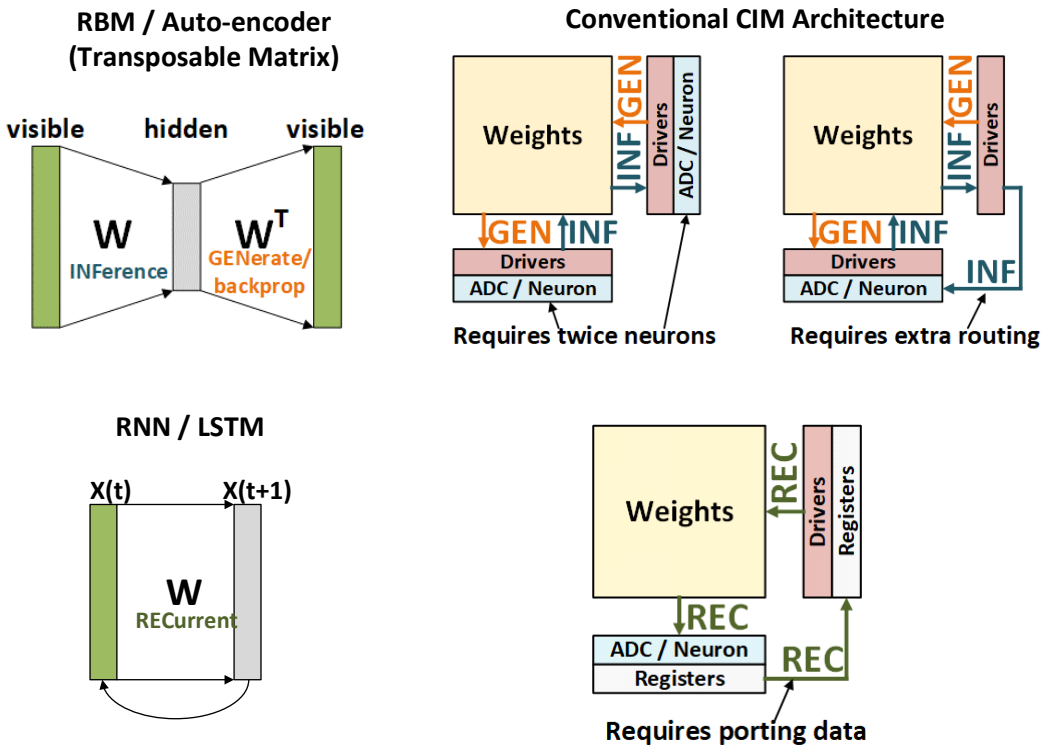
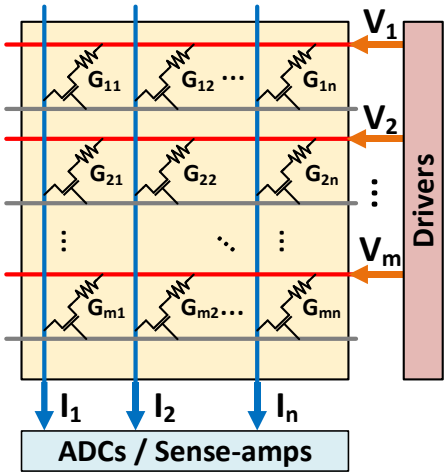


Figure 5.3: Conventional Computer-In-Memory architectures.

of previous designs [10, 34, 57, 7]. This is made possible through two key features: 1) a runtime reconfigurable dataflow with in-situ access to RRAM array and its transpose for efficient access to NN weights and 2) a voltage sensing stochastic integrate-and-fire analog neuron (I&F) that is reused for correlated double sampling (CDS), stochastic voltage integration, and threshold comparison. Fig.5.4 shows a simplified block diagram depicting matrix vector multiplication performed using a crossbar array with voltage inputs provided by drivers and the resulting output currents sensed using amplifiers and analog-to-digital converters.

This architecture consists of 256 neurons and supporting peripheral drivers and circuits, that can be programmed and configured to implement a variety of features tailored for different ANNs. A voltage sensing stochastic integrate-and-fire (I&F) analog neuron forms the core component, which can be reused for correlated double sampling (CDS), deterministic or stochastic



$$(V_1 \quad \cdots \quad V_m)\begin{pmatrix} G_{11} & \cdots & G_{1n} \\ \vdots & \ddots & \vdots \\ G_{m1} & \cdots & G_{mn} \end{pmatrix} = (I_1 \quad \cdots \quad I_n)$$

Figure 5.4: Matrix Vector Multiplication with RRAM crossboar array.

voltage integration, and binary or ternary level output from comparison of integrated input with a threshold window. The neuron design supports a variety of activation functions, that can cater to deep learning and neuromorphic applications. At only a fraction of the energy efficiency compared to the state-of-the-art implementations, this architecture can be easily adopted and scaled for large networks [53].

This chapter is organized as follows. Section 4.2 presents an overview of the chip architecture and its component blocks. The neuron design stages, including sampling, integrator, comparator, latch and readout, are described in detail in this section. In Section 5.3, the different modes of operation of the neuron are elaborated, illustrating how the same design can be configured to implement various activation functions. The measurement results of the chip are presented in Section 4.4, and Section 6.5 concludes the paper with a summary of the design features and possible applications.

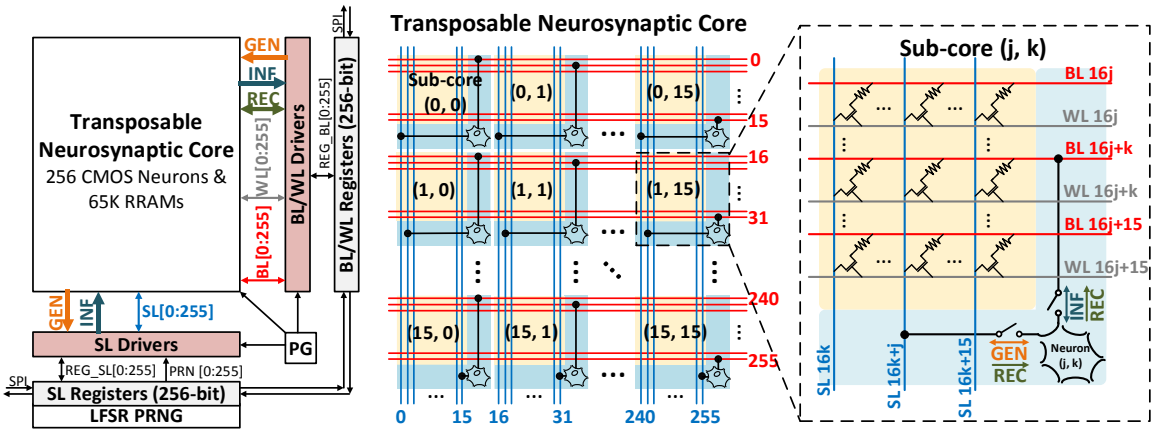
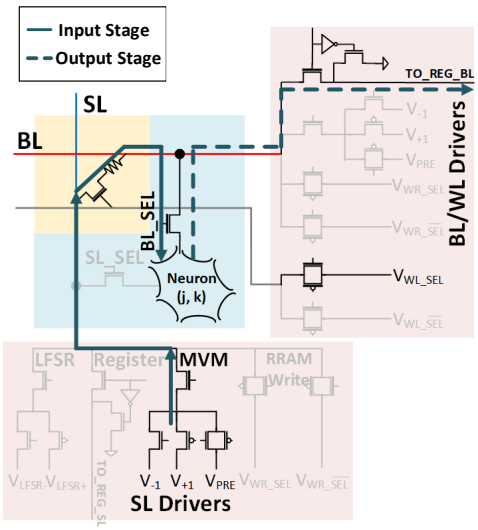


Figure 5.5: Block diagram of CIM architecture with neurosynaptic core. 16×16 I&F neuron array with 16×16 RRAM crossbar array for each neuron. Peripheral drivers, biasing, LFSR and SPI for I/O are also shown.

5.2 Chip Architecture

Fig. 5.5 shows the top-level architecture of the neurosynaptic core (NSC) composed of 16x16 sub-cores with shared bit-lines (BLs), word-lines (WLs), and select-lines (SLs). BLs connect to RRAM top electrodes, WLs to access transistors and SLs to RRAM bottom electrodes. Each sub-core contains 16x16 RRAM synapses, which store the NN weights, and one I&F neuron. The neuron in the (j, k)th sub-core in the 16x16 grid connects to the (16j+k)th BL and (16k+j)th SL for in-situ transposability in RRAM array access. The periphery consists of BL and WL drivers along the rows, SL drivers along the columns, and a delay-line-based tunable pulse generator (PG) to generate pulses for implementing RRAM read. Linear-feedback shift register (LFSR) chains enable stochastic sampling required for PGMs and 256-bit BL and SL registers along the periphery provide I/O. Configuring the SL/BL switches connects the I&F neuron to its respective SL and BL to realize multiple types of dataflows such as forward (INFer), as shown in Fig. 5.6 and reverse (GENerate), as shown in Fig. 5.7 matrix-vector multiplication (MVM), and recurrent connections (REC), as shown in Fig. 5.8. It is also possible to add linear feedback shift register (LFSR) noise as shown in Fig. 5.9 or send input pulses directly to the neuron, as shown in

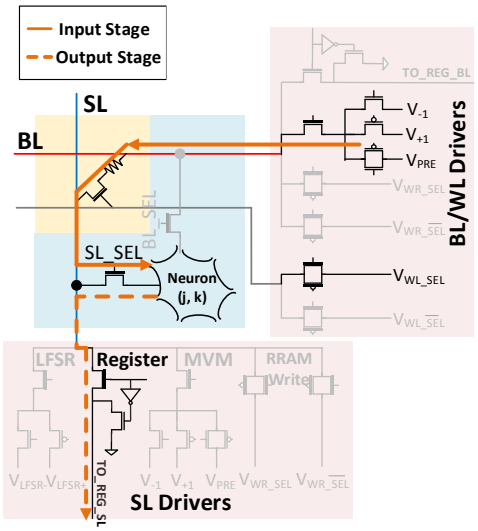


pulse input	neuron input switch	neuron output switch	Operation
SL	BL	BL	Forward (SL-BL) MVM
BL	SL	SL	Backward (BL-SL) MVM
BL	SL	BL	BL Recurrent MVM
SL	BL	SL	SL Recurrent MVM
BL	BL		Directly driving neurons
SL	SL		
LFSR	SL		Pseudo-random-noise injection into neurons

Figure 5.6: Reconfigurable dataflow to perform forward MVM.

Fig. 5.10. During GEN, input pulses are applied at the BLs, weighed by the RRAM, and sampled by the I&F through the SL switch. The I&F outputs are written into SL registers through the SL switches. During INF, pulses are applied at the SLs, weighed by the RRAMs, and sampled by the I&F through the BL switches. Fig. 2.1 shows the block diagram of the neuron array transceiver architecture composed of 16x16 I&F neurons with shared rows and columns. The peripheral circuitry consists of row and column drivers and biasing circuits. Linear feedback shift register (LFSR) chains add stochasticity to input samples in order to shape the neuron activation function. The 256-bit row and column registers along the periphery provide I/O communication through a serial peripheral interface (SPI). Configuring the row/column-select switches connects the I&F neuron to its respective row and column. The input pulses can be applied at the rows (or columns), and sampled by the neuron through the column-select (or row-select) switches. The neuron outputs are written into column (or row) registers through the column-select (or row-select) switches.

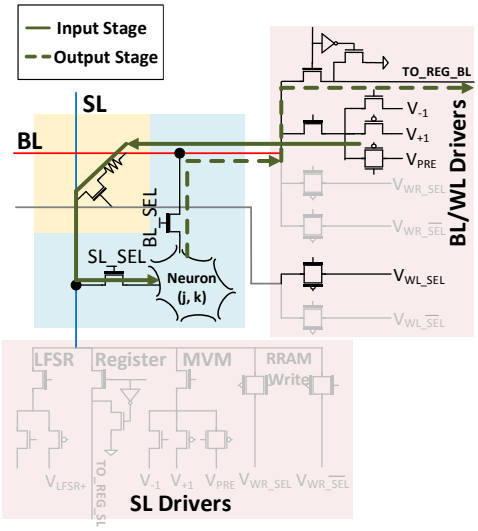
Fig. 5.11 shows the design of the I&F neuron which is composed of a single high-gain



pulse input	neuron input switch	neuron output switch	Operation
SL	BL	BL	Forward (SL-BL) MVM
BL	SL	SL	Backward (BL-SL) MVM
BL	SL	BL	BL Recurrent MVM
SL	BL	SL	SL Recurrent MVM
BL	BL		Directly driving neurons
SL	SL		
LFSR	SL		Pseudo-random-noise injection into neurons

Figure 5.7: Reconfigurable dataflow to perform backward MVM.

cascode amplifier, a latch, and switches to reconfigure the amplifier feedback loop between multiple modes of operation. Periodic offset cancellation through CDS mitigates circuit variation across the array and establishes a DC operating point for the capacitively coupled amplifier. During CDS offset compensation, first, a self-bias phase establishes the DC operating point (Vop) by shorting the input of the amplifier with the output. Simultaneously, a known reference (Vref) is sampled onto the capacitor Csample; together these sample the unknown input offset onto CCDS for subsequent cancellation. The I&F supports integration of input pulse sequences over time using multiple sample-integrate (SI) cycles to enable stochastic sampling and accumulation in PGMs. Fig. 5.12 shows two such SI cycles. During the sample phase, the input voltage is sampled onto Csample, followed by the integration phase during which the charge on Csample transfers onto Cinteg. We implement the neuron activation function, by breaking the amplifier feedback loop while simultaneously driving the bottom plate of Cinteg by Vth. The open loop amplifier functions as a comparator, generating a binary decision by comparing Vth+Vinteg and Vref. The output is then latched and written to the BL/SL registers through the corresponding switches. Each neuron occupies an area of 1200 µm2 and measurements show operation with 63 nW static power drawn from a 1.8 V supply. The total static power consumption for one NSC (256 I&F



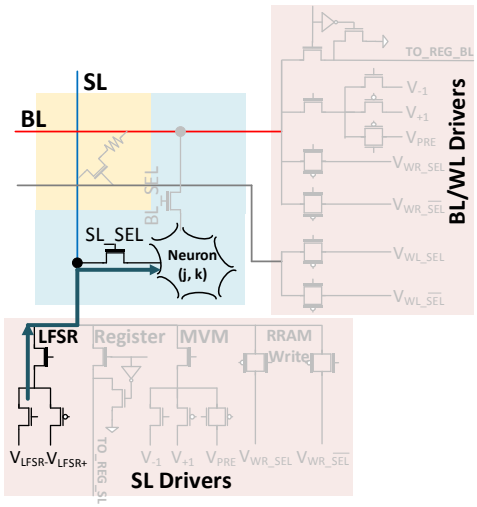
pulse input	neuron input switch	neuron output switch	Operation
SL	BL	BL	Forward (SL-BL) MVM
BL	SL	SL	Backward (BL-SL) MVM
BL	SL	BL	BL Recurrent MVM
SL	BL	SL	SL Recurrent MVM
BL	BL		Directly driving neurons
SL	SL		
LFSR	SL		Pseudo-random-noise injection into neurons

Figure 5.8: Reconfigurable dataflow to perform recurrent MVM.

neurons and biasing) is 17 µW. Energy efficiency is in part derived from voltage sensing in the I&F which drives the output of the RRAM array to high impedance during MVM, thus avoiding the static current draw in current sensing configurations. Fig. 5.11 shows the design of the I&F neuron which is composed of a single high-gain operational trans-conductance amplifier (OTA), switches and output latch to reconfigure the amplifier feedback loop and enable multiple modes of operation. The CDS technique provides periodic offset cancellation, mitigating systematic variations in the circuit and also establishes a DC operating point for the capacitively coupled OTA.

5.2.1 Input Stage: CDS, Sampling and Integration

The DC operating point (V_{OP}) of the amplifier is determined by a self biasing phase, where the input and output of the OTA are shorted together. At the same time, a known reference voltage (V_{REF}) is sampled onto the capacitor C_{SAMPLE}. Thus, the unknown input offset is sampled onto C_{CDS} for subsequent cancellation. In the Sampling phase, the input pulse delivers a charge onto C_{SAMPLE} whenever the *SAMPLE* switch is enabled. The neuron supports integration of input pulse



pulse input	neuron input switch	neuron output switch	Operation
SL	BL	BL	Forward (SL-BL) MVM
BL	SL	SL	Backward (BL-SL) MVM
BL	SL	BL	BL Recurrent MVM
SL	BL	SL	SL Recurrent MVM
BL	BL		Directly driving neurons
SL	SL		
LFSR	SL		Pseudo-random-noise injection into neurons

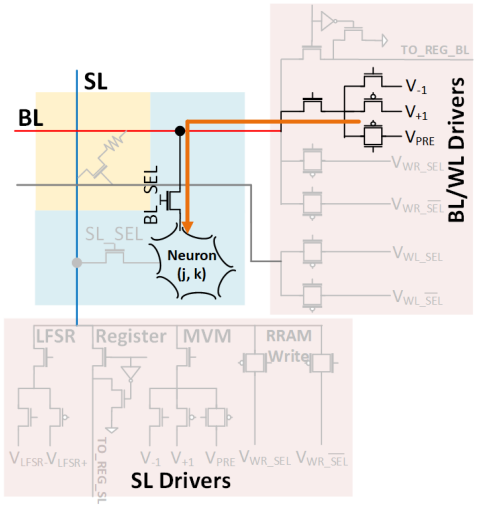
Figure 5.9: Reconfigurable dataflow to send inputs through LFSR to neuron.

sequences over time using multiple sample-integrate (SI) cycles. During the integration phase, the charge on C_{SAMPLE} is transferred onto C_{INTEG}, by enabling the $INTEG$ switch. Using LFSR to generate input pulses, enables stochastic sampling and accumulation in the neuron. Sizing C_{INTEG} to be 6x C_{SAMPLE} extends the linear accumulation range of the neuron. Fig. 5.12 shows the detailed timing waveform of all the input control signals (analog and digital) that configure the neuron to operate in various phases, starting with CDS, followed by multiple Sampling and Integration cycles. The compare phase then does binary/ternary comparison to generate a 1bit/2bit output that is written to an output register during Readout phase. For partial reset phase, the neuron output is sampled by the neuron input through feedback to increase/decrease V_{SAMPLE} based on the current neuron state. The OTA based integrator gain is given by,

$$\Delta V_{OUT} = \frac{C_{SAMPLE}}{C_{INTEG}} \Delta V_{SAMPLE} \tag{5.1}$$

5.2.2 Output Stage: Comparison, Register Write and/or Partial Reset

The neuron activation function is implemented by breaking the amplifier feedback loop while simultaneously driving the bottom plate of C_{INTEG} by V_{COMP_TH}. The open loop amplifier



pulse input	neuron input switch	neuron output switch	Operation
SL	BL	BL	Forward (SL-BL) MVM
BL	SL	SL	Backward (BL-SL) MVM
BL	SL	BL	BL Recurrent MVM
SL	BL	SL	SL Recurrent MVM
BL	BL		Directly driving neurons
SL	SL		
LFSR	SL		Pseudo-random-noise injection into neurons

Figure 5.10: Reconfigurable dataflow to send inputs directly to neuron.

functions as a comparator, generating a binary decision by comparing V_{COMP_TH}+V_{INTEG} and V_{REF}. The output is then latched and written to the row/column registers through the corresponding switches. In order to generate a ternary output, the comparison is done in two phases. In phase one, a decision similar to the binary output, by comparing V_{INTEG}+V_{PLUS_TH} and V_{REF}, is made to resolve the neuron state to be ”0” (no event) or ”+1” (positive event). In phase two, another binary decision is made by comparing V_{INTEG}-V_{MINUS_TH} and V_{REF}, to resolve the neuron state to be ”-1” (negative event) or ”0” (no event).

Fig. 5.13 illustrates how sampling, integration, compare and readout phases affect the analog voltages V_{SAMPLE} and V_{OUT} and digital output of the neuron, $V_{OUT\_DIGITAL}$. Fig. 5.14 characterizes the linearity of the SI operation over multiple input pulses. Sizing Cinteg to be 6x Csample extends the linear accumulation range of the I&F. Fig. 3 shows the measured linear dependence between the I&F threshold and 1) the number of input pulses (top left), and 2) the pulse amplitude (top right). Spatially uncorrelated, controlled Bernoulli PRNs are required for PGM operation; these are generated through two counterpropagating LFSRs whose outputs are modulated to be Vref+Vn and Vref-Vn and applied to the I&F via SLs. The accumulation of multiple noise pulses smoothens the sharp decision point of the comparator to a more sigmoidal

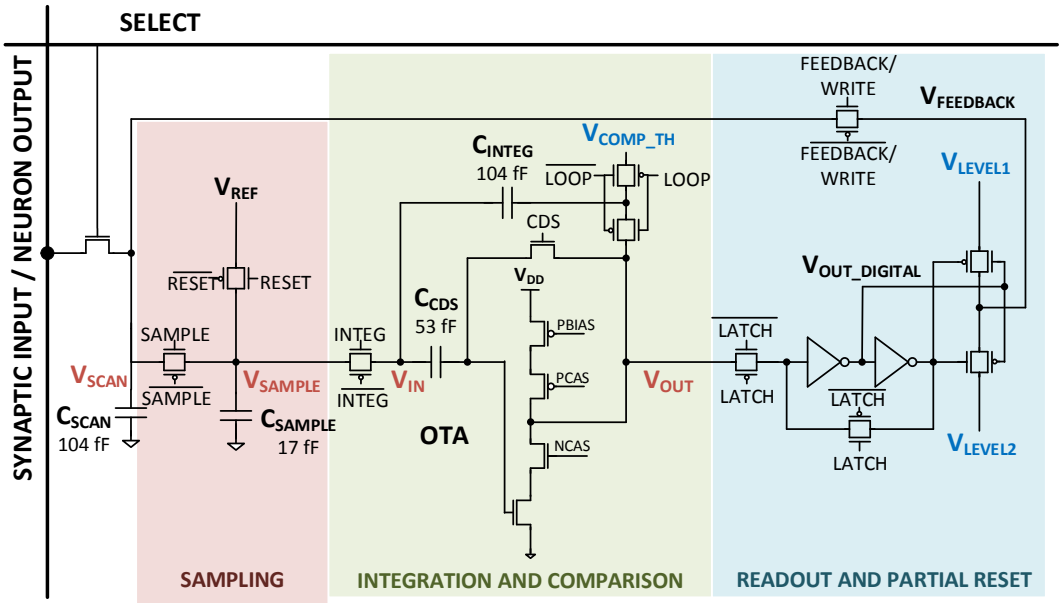


Figure 5.11: Charge-mode mixed-signal circuit implementation of the neuron (left) with peripheral biasing (right).

function as measurements show in Fig. 5.15. Fig. 5.15 shows the measured relationship between the number of stochastic pulses and the sigmoidal characteristic of the I&F.

5.3 Neuron Activation Modes

5.3.1 Step activation function: binary or ternary states

Step activation is realized with a tunable V_{COMP_TH} voltage that sets the stepping threshold when the neuron flips from ”-1” (negative) to ”0” (no event) state, or from ”0” (no event) to ”+1” (positive event) state. If only one threshold voltage is applied for comparison, the neuron output will be binary. If two different threshold voltages, V_{PLUS_TH} and V_{MINUS_TH}, are applied for comparison in two phases, the neuron output will be ternary. Since the CDS phase eliminates any mismatch or offset in the neurons, the 256 neurons in the chip show minimum variation in the activation function, as can be seen in Fig. 5.17 (top).

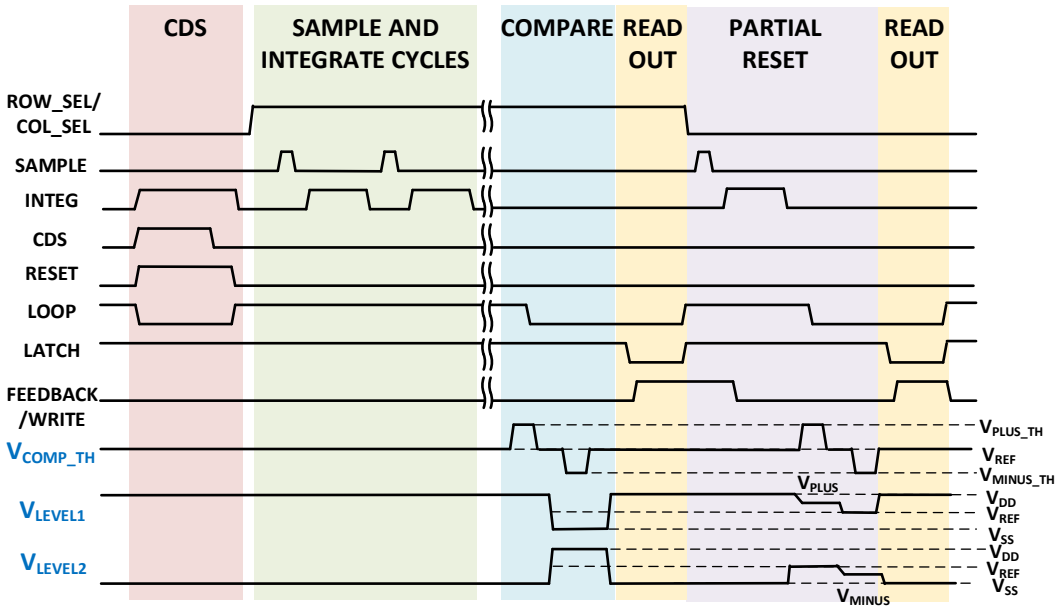


Figure 5.12: Timing diagram of neuron operation. Digital and analog input control signals.

5.3.2 Sigmoidal activation function: binary or ternary states

This activation mode is similar to the step activation, but in addition to the input pulses, LFSR pulses are added that represent noise. Uncorrelated pseudo-random noise is generated through two counter propagating LFSRs whose outputs are modulated to be V_{REF}+V_N and V_{REF}-V_N and applied to the neurons via column lines. The accumulation of multiple noise pulses smoothens the step transition of the comparator to a sigmoidal function as measurements show in Fig. 5.17 (bottom). Similar to the step activation mode, this mode can also generate binary/ternary output based on the comparison phase threshold voltages.

5.3.3 ReLU activation function

ReLU activation is based on rate coding of spikes, unlike the previous modes which are based on individual spike timing. This activation function can be generated by using the partial reset mechanism, where the digital output of the neuron is sampled through feedback to increase

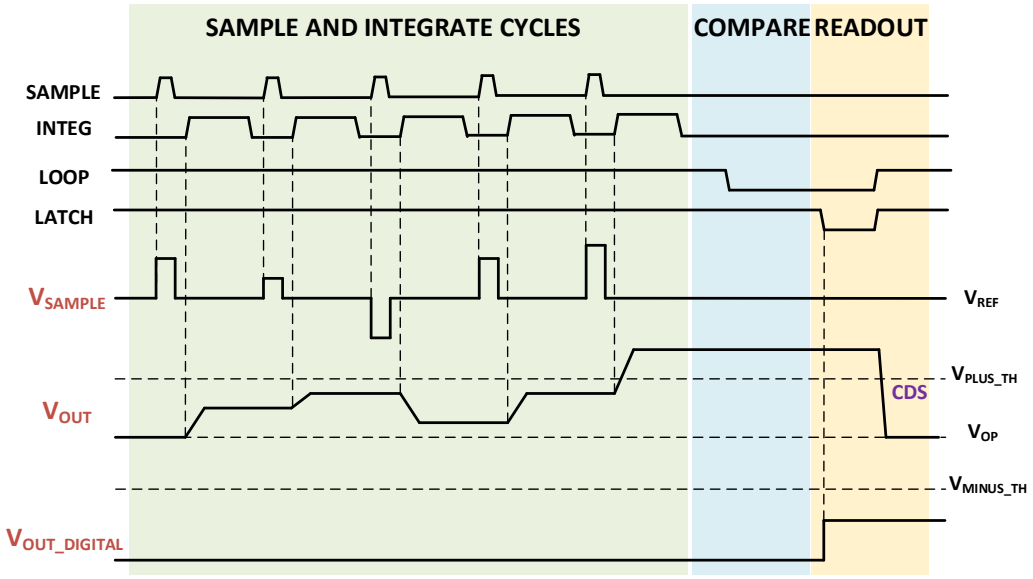


Figure 5.13: Timing diagram of neuron operation. Illustration of sampling, integration and comparison phases.

or decrease from previous level depending on the current neuron output level. If the current output level is a positive event ("+1"), the partial reset would decrease V_{SCAN}, and vice versa, such that when the output is readout next time, has an increasing probability of not firing (no event).

Fig. 5.18 (left) shows the implementation of the ReLU activation for different partial reset threshold voltages. The transfer curve is the average of the 256 neuron output firing probabilities. The number of pulses sent as input to the neuron is fixed to 30. When the input amplitude is negative, the neuron does not fire at all, hence the output probability is zero. When the input pulse amplitude increases above zero, the output firing probability of the neuron increases linearly until it reaches the partial reset threshold voltage where the probability of neuron firing plateaus close to 1. Fig. 5.18 (right) shows that sigmoid activation function can also be generated by adding LFSR noise to the ReLU neuron.

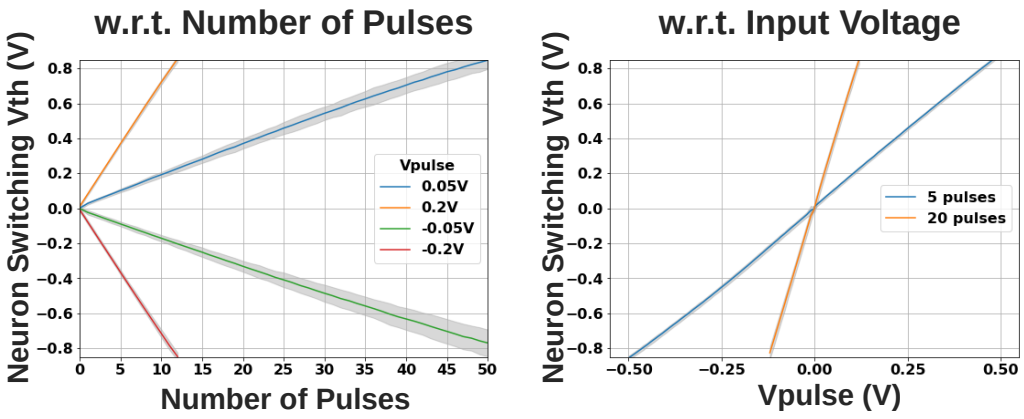


Figure 5.14: Linearity of Sample and Integration using the proposed neuron circuit.

5.4 Measurement Results

The I&F neuron array transceiver was designed and fabricated in 130-nm CMOS technology [53]. Fig. 5.19 shows the chip micrograph. Active area of the chip is approximately 1.796 mm^2 including input SPI and peripheral circuitry, with the row/column drivers, registers and LFSR dominating the area. Individual I&F neuron has an area of $1200\mu\ m^2$. The chip operates at 1.8V supply for both digital and analog blocks. Measurements show each neuron operation with 63 nW static power drawn from a 1.8 V supply. The total power (static+dynamic) consumption of the chip (256 I&F neurons, biasing and peripherals) is 140.6 μW for data throughput of 92.5 MSpikes/s.

We characterize the MVM performance of the system by programming analog-valued weights in each RRAM device (device TEM shown in Fig. 5.2) using a write-verify scheme similar to [63]. The RRAMs are programmed to ±2 µS of their targeted conductance (program success rate = 99.4%). Fig. 5.16 shows the RRAM conductance distribution measured immediately after programming and 1 day after programming. We apply random input vectors to collect the output distribution of each neuron at different nominal MVM output values, testing MVM operation. The sharp transition between -1 to +1 in the neuron output in Fig. 5.16 demonstrates robustness

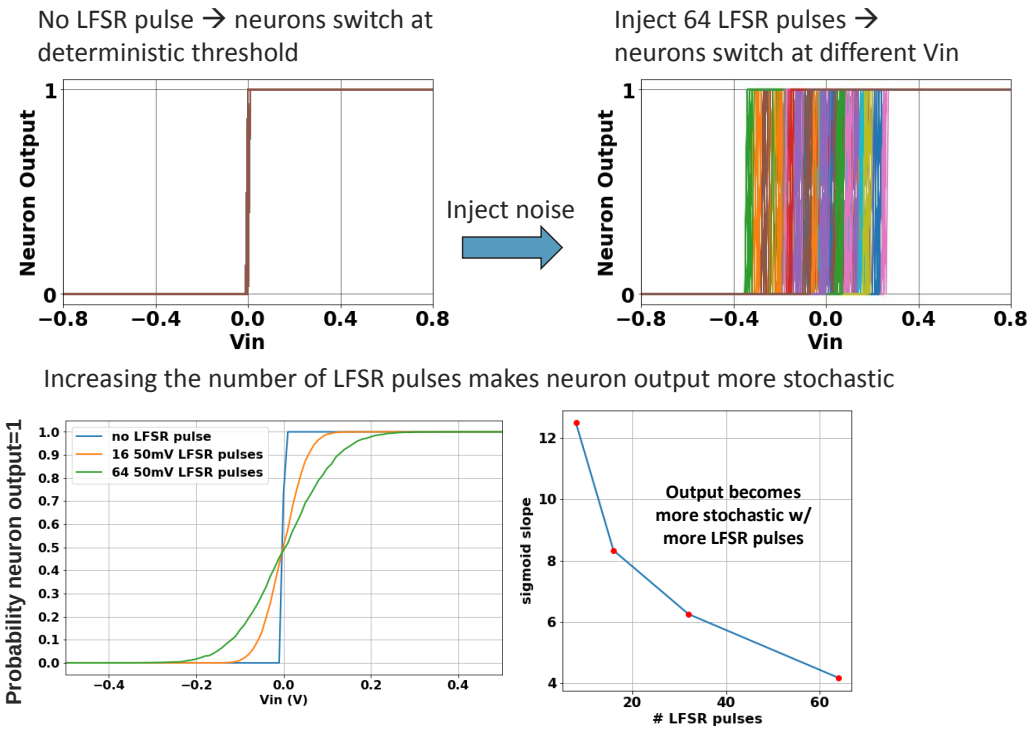


Figure 5.15: Adding LFSR pulses to neuron to enable stochastic sampling.

to RRAM resistance drift over time. To illustrate MVM capabilities in both forward and reverse directions, as well as temporal accumulation and probabilistic sampling in the neuron, we use a generative RBM to reconstruct MNIST digit images, shown in Fig. 5.20. RBMs are PGMs consisting of a set of fully-connected visible neurons and hidden neurons. During inference, binary inputs sampled from the gray-level pixels are presented to the visible neurons. MVM and Gibbs sampling are repeated back-and-forth between visible and hidden neurons. We subsample MNIST digit images to 15x15 gray images and implement an RBM using 225 visible neurons (15x15 pixels) and 60 hidden neurons. We use 4 RRAM cells (2-row by 2-column) per weight for differential encoding in both horizontal and vertical directions to implement positive/negative weights in both INF and GEN. Fig. 5.20 (bottom) shows the reconstruction performed on all 10 digits with a mean square reconstruction error of 1.91 per image, comparable to software implementations with 7 level quantized weights. The NSC along with the periphery measures

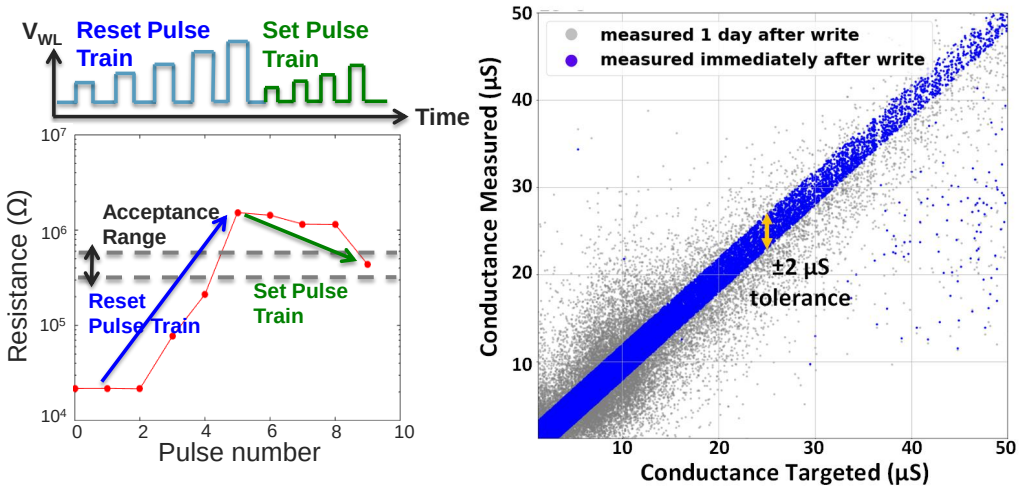


Figure 5.16: Programming RRAM device using SET and RESET pulse trains.

1.79 mm2 in 130-nm CMOS with a single sub-core occupying 1849µm2. Energy breakdown for the NSC operation is in Fig. 5.22, with energy dominated by WL switching. Scaling clocking frequency of the I&F linearly scales throughput and power (from 50 uW to 1.5 mW), with 74 TOPS/W energy efficiency over that range. A comparison of the key metrics with state of the art RRAM-based CIM designs is given in Fig. 5.23, showing that our design achieves the highest reported computational energy-efficiency among RRAM-based CIM implementations.

Fig. 5.17 (a) - (d) show measured waveforms of the activation functions of the neuron characterized for binary or ternary levels. Step activation is realized with a tunable V_{TH} whereas sigmoid activation can be realized by either adding LFSR pulse for noise shaping or using the partial reset mechanism. The chip was configured through a Xilinx FPGA development board to program bias voltages and currents, send commands to the chip, and receive the data from the chip. The output data from the chip were sent through USB to a PC where data analysis and post processing was performed to generate the transfer curves and extract the neuron activation function.

The performance metrics of the proposed Neuron array transceiver are summarized in

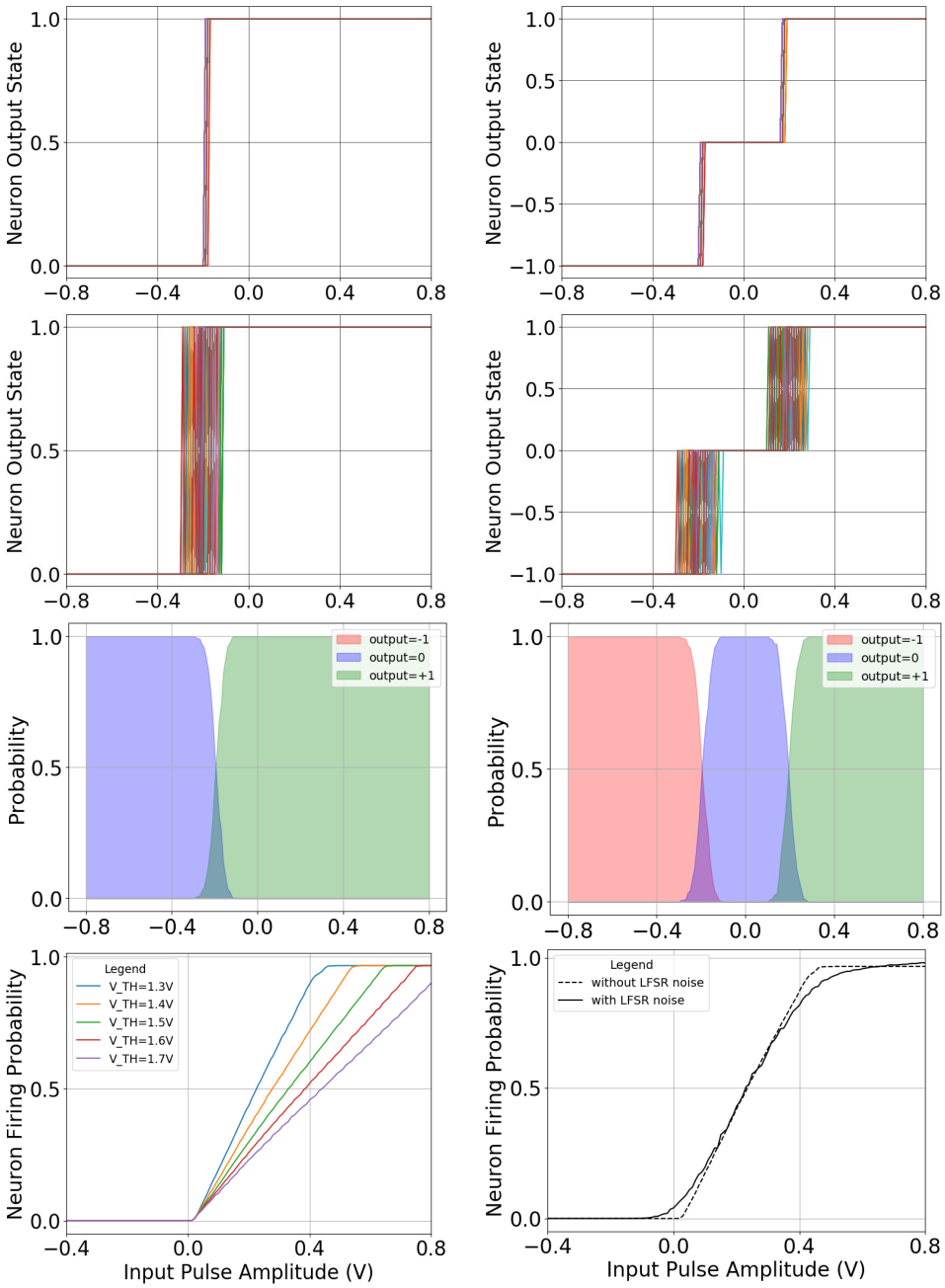


Figure 5.17: Implementation of step, sigmoid and ReLU activation functions. Dual thresholds are $\pm$0.2V for both step and sigmoid functions. ReLU activations are shown for different threshold voltage, V_{TH}. Sigmoid activations are generated by coupling LFSR noise to the neuron.

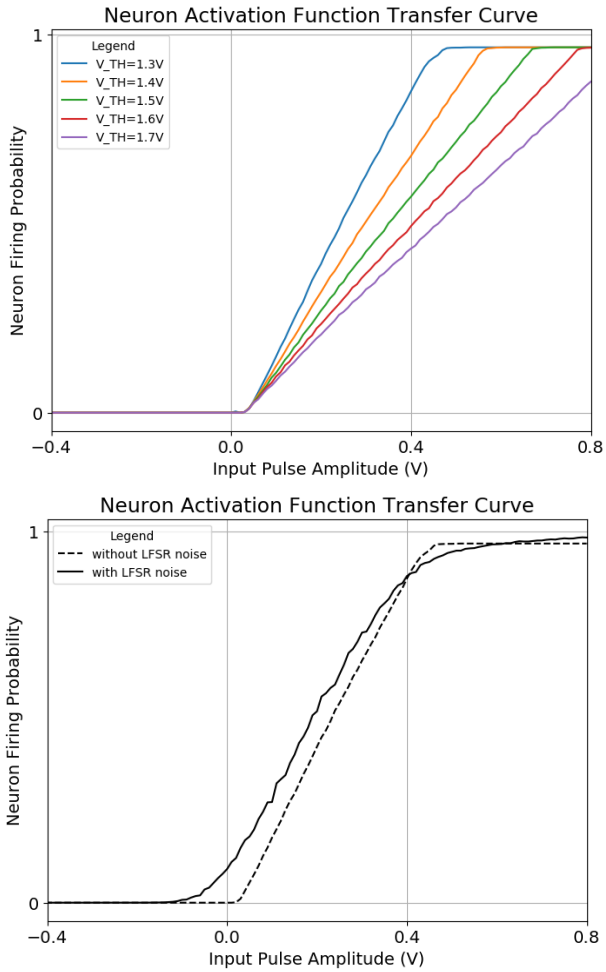


Figure 5.18: Implementation of ReLU Activation using partial reset technique. (a) Characterization curves shown for different threshold voltage, V_{TH}. (b) Sigmoid activation function generated by coupling LFSR noise to the neuron.

Table 5.2 in comparison with other architectures reported in the literature. Since each architecture has different number of neurons, that have different complexity of implementation and operating at different supply voltages, power consumption varies significantly. However, energy efficiency is an effective Figure-of-Merit (FoM) for comparison of these architectures, given by, FoM = $E_{op}\,/\,N_{op}$, where E_{op} is the energy consumed for synaptic operations N_{op}, performed. The proposed architecture achieves 1.52 pJ/Spike., where each pre-synaptic input event constitutes one operation, which is better energy efficiency than other analog implementations [39, 37].

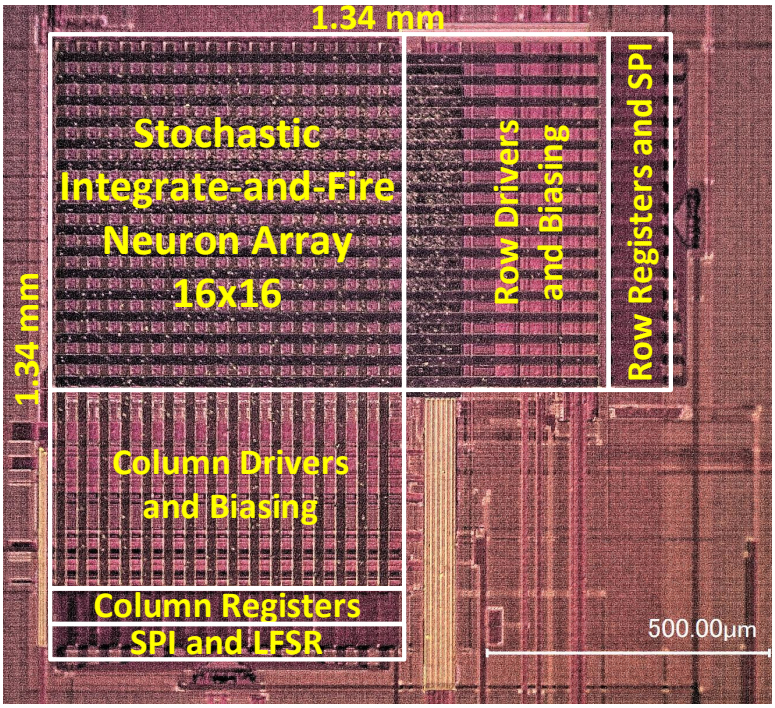


Figure 5.19: Chip micrograph.

5.5 Conclusions

In this chapter, we presented the first system-level demonstration of a high-efficiency, fully-integrated CMOS-RRAM CIM architecture with core-level support for a diverse set of NN architectures and dataflows including probabilistic graphical models. Multi-core extensions of this work open the door to advances in artificial intelligence at a large scale with extreme energy-efficiency and integration density. An ultra-low power neuron array transceiver with a multi-modal integrate-and-fire neuron architecture was presented. A variety of activation functions were realized by using additional pseudo-random noise or partial reset mechanism that makes this chip versatile to cater to different ANN architectures. The highly reconfigurable and energy efficient design makes this transceiver suitable for deep learning applications such as RBMs, CNNs using neurons with ReLU or sigmoidal activation. Neurons with step or sigmoidal activation, with binary or ternary output levels, can be used for both rate coding or spike timing

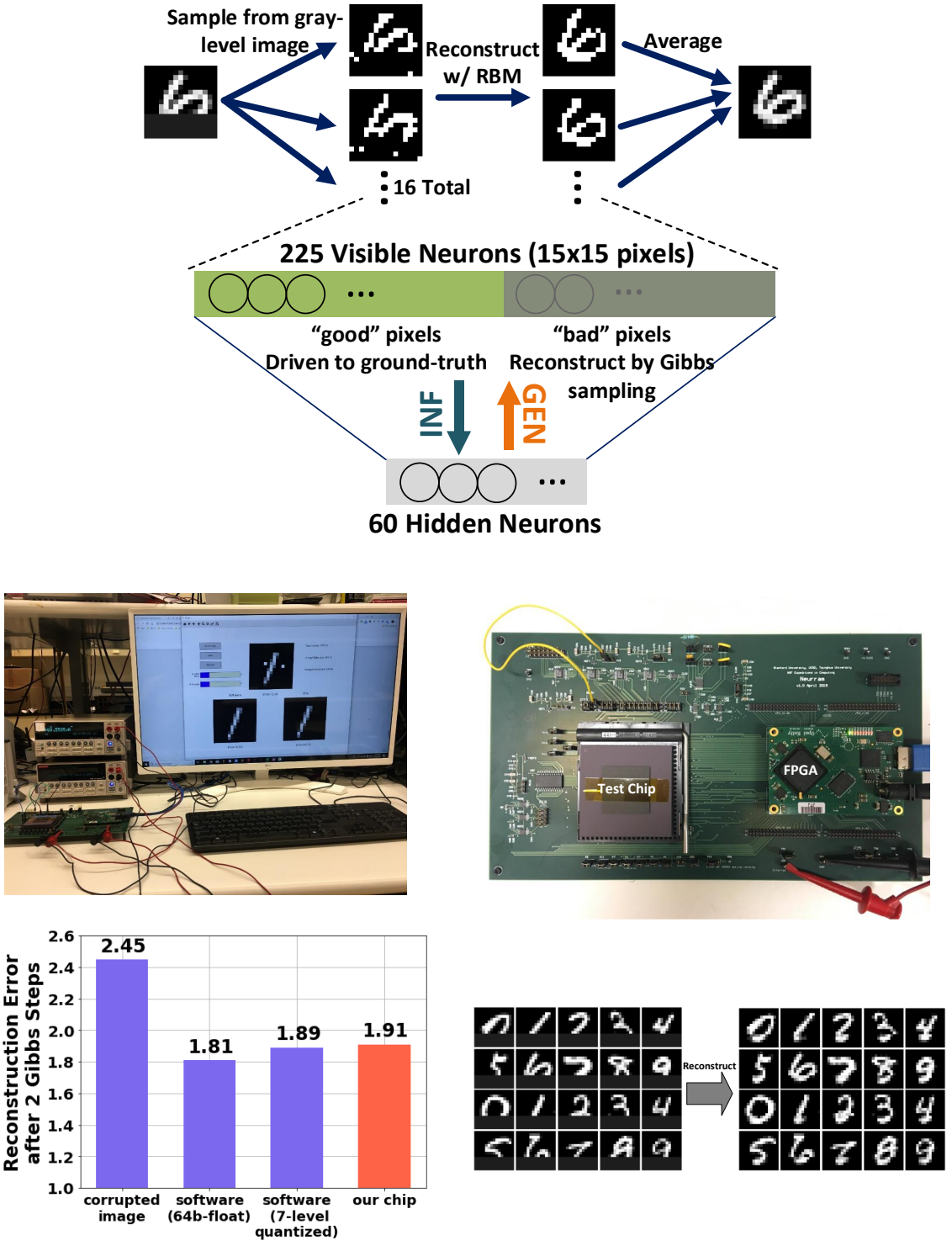
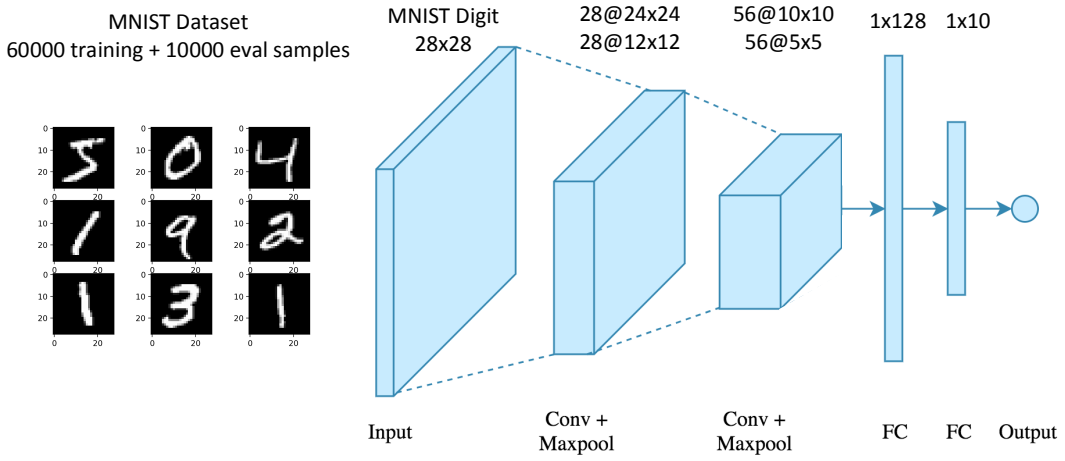


Figure 5.20: Implemented RBM architecture (top) and evaluation results on MNIST in software vs hardware (bottom).



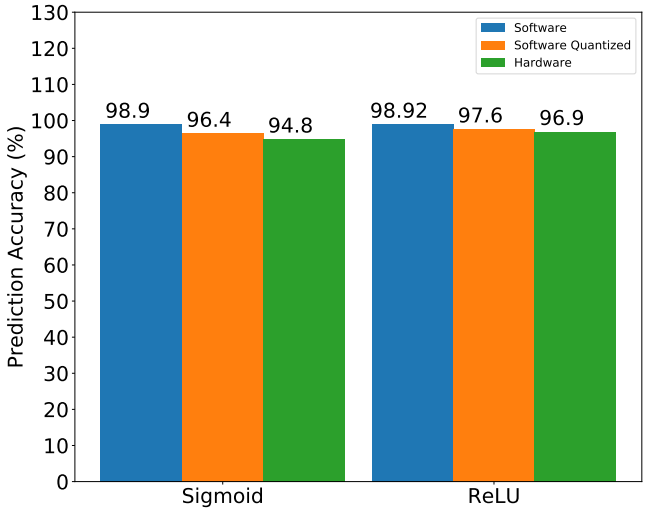


Figure 5.21: Implemented CNN architecture (top) and evaluation results on MNIST in software vs hardware (bottom).

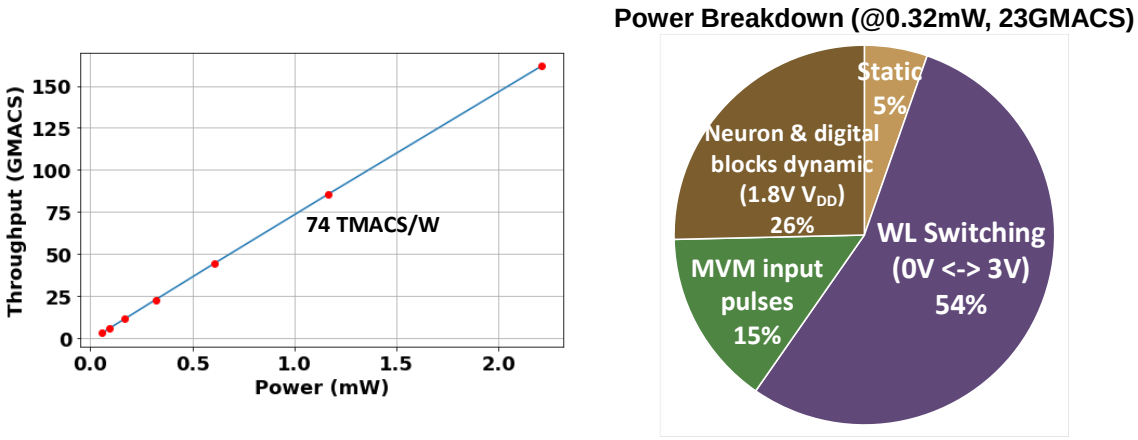


Figure 5.22: Power consumption pie chart and throughput plot.

based neuromorphic applications.

5.6 Acknowledgments

Chapter 5 is in part a reprint of the material as it appears in Wan, W., Kubendran, R., Eryilmaz, S.B., Zhang, W., Laio, Y., Wu, D., Deiss, S., Gao, B., Raina, P., Joshi, S., Wu, H., Cauwenberghs, G., and Wong, H.-S.P., "A 74 TOPS/W CMOS-ReRAM Neurosynaptic Core with Dynamically Reconfigurable Dataflow and In-situ Transposable Weights for Probabilistic Graphical Models." *IEEE International Solid State Circuits Conference (ISSCC)*, 2020, and in Kubendran, R., Wan, W., Joshi, S., Wong, H.-S.P., and Cauwenberghs, G., "A 1.52-pJ/Spike Reconfigurable Multimodal Integrate-and-Fire Neuron Array Transceiver." *ACM International Conference on Neuromorphic Systems (ICONS)*, 2020. The dissertation author was an equal contributing author of these papers. This study was sponsored in part by the National Science Foundation (NSF EFRI-1137279, CCF-1317560), and the Office of Naval Research (ONR MURI N00014-13-1-0205). The views and the conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or

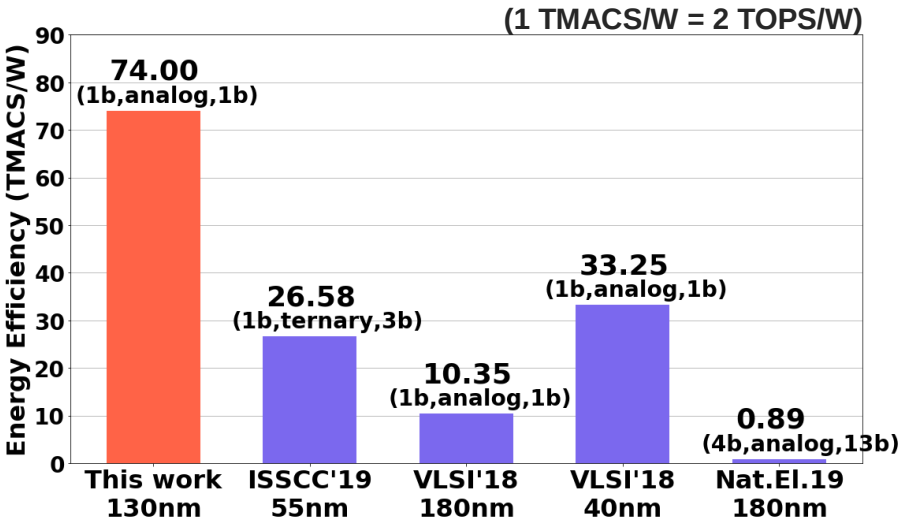


Figure 5.23: Throughput comparison of CMOS-RRAM Compute-In-Memory architectures.

implied. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Table 5.1: Performance Comparison of CMOS-RRAM Compute-In-Memory architectures.

Parameter	ISSCC'18 [10]	VLSI'18 [34]	ISSCC'19 [57]	Nature Elec. 19 [7]	This Work
CMOS Technology	65nm	40nm	55nm CMOS	180nm	130nm
Synaptic Device	Unipolar RRAM	TaOx RRAM	RRAM	WOx RRAM	HfOx/TaOx RRAM
Connectivity (dataflow direction)	Forward	Forward	Forward	Forward, Reverse	Forward, Reverse, Recurrent
Supported Neural Network Architecture	CNN/MLP	MLP	CNN/MLP	MLP, Sparse Coding	ANNs: RNN, MLP, CNN PGMs: RBM, Bayesian
Neuron/ADC Type	3-b sense-amp	1-b sense-amp	4-b sense-amp	13-b ADC	I&F spiking, deterministic/stochastic
Synaptic Array Dimension	512 x 256	-	256 x 512	54 x 108	256 x 256
Synapses/mm^2	-	1.48M	-	0.04M	0.59M
Power (mW)	-	9.9	-	64.4	0.05-2.2
Precision (input, weight, output)	(1b, ternary, 3b)	(1b, analog, 1b)	(1b, 3b, 3b)	(4b, analog, 13b)	(1b, analog, 1b)
Energy efficiency (TOPS/W)	-	66.5	53.17	0.89	75

Table 5.2: Neuron Architecture and Performance Comparison

Parameter	ROLLS Processor [39]	Braindrop [36]	IFAT [37]	This Work
Technology	180nm CMOS	28nm CMOS FDSOI	90nm CMOS	130nm CMOS
Supply Voltage	1.8 V	1 V	1.2 V	1.8 V
Neuron Count	256	4096	65536	256
Activation Function	Step, Sigmoid	Step	Step	Step, Sigmoid, ReLU
Output Levels	Binary	Binary	Binary	Binary, Ternary
Power Consumption	4 mW	NA	1.572 mW	140.6 μW
FoM	NA	381 fJ/Syn.Op.	22 pJ/Spike.	1.52 pJ/Spike.
Active Area	51.44 mm^2	0.65 mm^2	16 mm^2	1.796 mm^2

Chapter 6

Inverted STDP Learning Rule for Temporal Pattern Recognition

6.1 Introduction

The last decade has seen significant development in the field of event-based sensing and computing. Several event-based cameras such as the Dynamic Vision Sensor (DVS) and the Asynchronous Time-based Image Sensor (ATIS) set the stage for spike-based computation, inciting the development of a new class of algorithms and machine learning architectures. These algorithms are able to operate on the timing of events rather than creating frames to recycle decades of conventional image processing. Recently, an algorithm called Hierarchy of Time Surfaces (HOTS)[26] introduced the concept of unsupervised learning using time surfaces that builds on increasingly deep temporal layers to bind events on several time scales to create a hierarchy of deep temporal features. The principle behind the method is to convert the relative timing between events occurring at different lines of information into normalized features that are invariant to the actual timing and only emphasize the temporal interval between past events and the current event.

This method is data driven, so when no events are being recorded, nothing is computed. HOTS uses clustering to extract the most representative temporal surfaces at each layer. In this paper we will show how this approach can be applied to spiking neural networks. We will provide an alternative to clustering by introducing the concept of inverted STDP that builds on the dynamic decaying kernels at the core of temporal surfaces.

Spiking Neural Networks (SNN) have the potential to transform edge learning for AI applications, due to their potential for inherent energy efficiency when using asynchronous hardware resources. SNN are also more suitable for temporal pattern recognition, as conventional Artificial Neural Networks (ANN) using backpropagation algorithm are not intuitively and seamlessly compatible. ANN use expensive and ineffective techniques such as network unfolding, as used in Backpropagation-through-time (BPTT)[35] in order to learn temporal patterns. On the other hand, one of the major bottlenecks in the widespread adoption of SNN is the lack of effective learning algorithms. Backpropagation of error signals is not directly compatible with SNN due to the spike function being non-differentiable. The few learning algorithms available for SNN are mostly based on rate coding of spikes and rarely work exclusively with spike timing[47, 25].

Previous work on spiking neural networks based on temporal coding of spikes for learning use a variety of synaptic delay kernels, such as the Synaptic Kernel Inverse Method (SKIM) [24] or fully digital programmable delay for a fixed kernel, such as Synapto-dendritic Kernel Adapting Neuron (SKAN) model [43]. In both these methods, there is no synaptic weight update i.e. the weight is fixed and only the decay function (alpha, exponential, damped resonant synapse) is used to encode timing information.

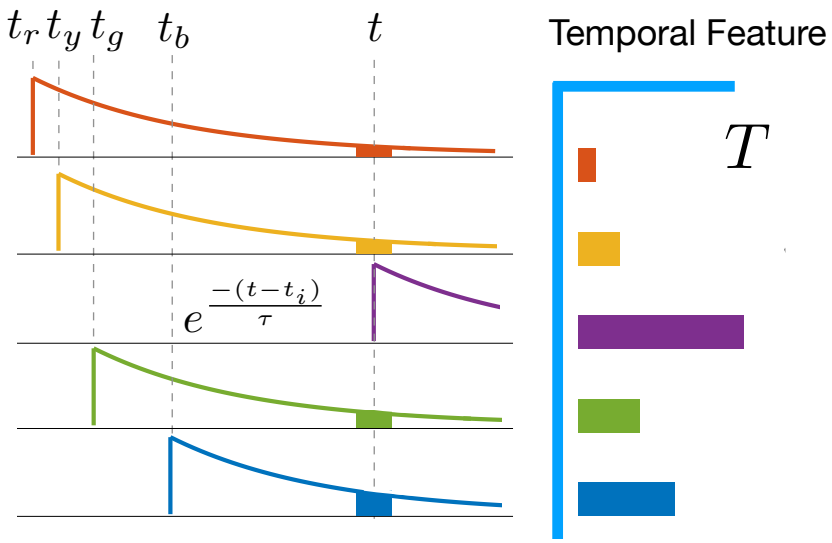


Figure 6.1: Principle of Temporal Context Representation. Five lines on information conveyed temporal events at different time locations tr, ty, ..., t_b. A temporal context $\mathcal{T}$ is computed for each incoming event (here at time t) as a vector expressing temporal delays between events as a value between 0 and 1.

6.2 Background

6.2.1 Deep temporal Networks: Hierarchy of Time Surfaces

The principle of time surface is to convert the delay between the current event time tt and other events shown with different colors and appearing at times t_r, t_y,...t_b, into a normalized value between 0 and 1 using an exponentially decaying function with a constant τ such that the timing of the events that are closer to the reference spike give values close to 1 while those occurring earlier tend to get closer to 0 as shown in inset Fig. 6.1. Thus for each incoming event, we can define a temporal context vector T of dimension 5 where 5 is the current number of temporal information lines of this example computed as : $T_j = exp(\frac{-|t-t_j|}{\tau})$.

The parameter τ maybe considered as the history parameter that decides how far in time is an event compared to the current incoming one that sets the reference timing. The exponential kernel, unlike a linear kernel, places more weight to closer spikes in time. Unlike a Gaussian kernel it is asymmetric and ignores future spikes.

The principle of Deep Temporal Learning is shown in Fig.6.2. Events are fed into the system, (a) for each incoming event a temporal feature is computed based on a temporal integration time τ_1, (f) a clustering algorithm is used to select the most "canonical" n_1 temporal features $T_1^{\tau_1}$,...,$T_{n1}^{\tau_1}$ that constitute the first layer of deep temporal architecture shown in (b). Once this layer is stable incoming temporal features will match a set of existing canonical features. If a canonical feature matches the current input, it emits a new event to the second layer that will then integrates information on a larger time scale $\tau_2 > \tau_1$. The same process is applied to the second layer that, once stabilized, will emit new events to the third layer that will integrate events to form even more complex temporal combination of features of features on a larger time scale τ_3. As the network gets deeper, integration time increases, leading to increased sparsity and lower numbers of canonical temporal vectors. The main idea is to build on temporal combinations of more and more complex features. The system is also able to operate in a continuous learning manner as, once the process of determining the canonical temporal vectors is stabilized, if a new feature appears and does not match an existing T^{τ_1} it updates existing base temporal features by triggering a new clustering computation.

Fig. 6.3 shows an illustration of the target problem of temporal pattern recognition. In this particular case, consider a simple network consisting of five synaptic inputs connected to a single somatic neuron. There can be multiple input spike events arriving through each pre-synaptic dendritic connection. These events generate synaptic currents that are summed with a weighted average that is represented by its membrane voltage, V_{mem} and compared to a threshold voltage, V_{thresh}, by the post-synaptic neuron. The neuron will generate a post-synaptic spike on the axonal output if its membrane voltage exceeds the threshold and resets itself back to the reset voltage, V_{reset}. The task of the network is to adjust the weights such that the neuron spikes if and only if the particular timing order and timing difference is maintained by the input spike train. This can be achieved using dynamic synapses with delay kernels, in which case the neuron membrane voltage itself acts as the time surface histogram that has a unique signature based on the spatio-temporal

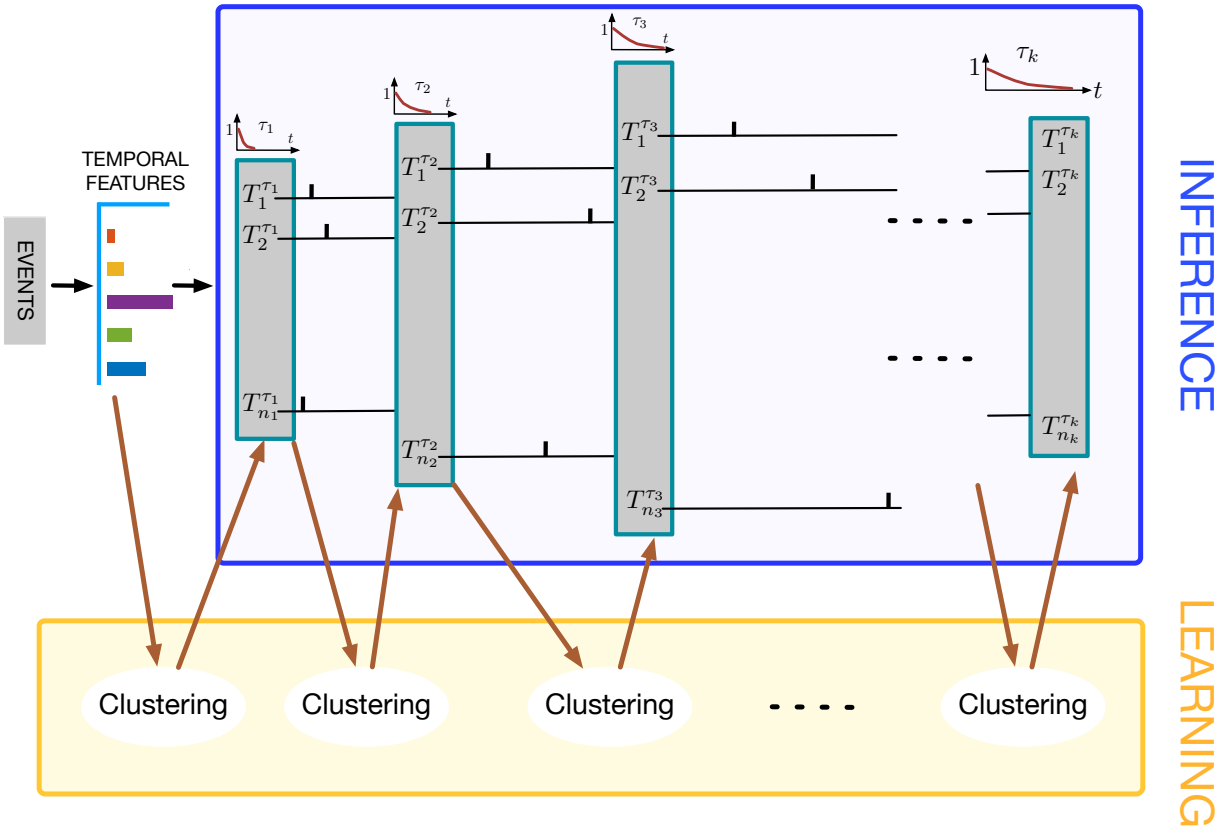


Figure 6.2: Principle of Deep Temporal Networks based on Hierarchy of Time Surfaces: temporal features are computed at each incoming event. The first layer will learn the most representative temporal features T^{τ_1} by using clustering. The clustering allows to learn the elementary features for a specific integration time τ_1. Once the first layer is stable each incoming temporal feature will elicit the activity of the closest base temporal feature that will in turn emit a new spike to the second layer that will perform a similar operation but using a larger decay τ_2 to compute deep temporal features T^{τ_2} that is a temporal combination of elementary base features T^{τ_1} from layer 1.

information provided by the input spike times.

Consider the simple SNN implemented in Fig. 6.3, with one neuron that receives inputs through synapses from an input spike train and generates an output spike train based on threshold crossings of its membrane voltage. For a LIF neuron, the membrane voltage V_{mem} ramps up/down proportional to the total synaptic current I_{syn_total} on the membrane capacitance C_{mem}.

$$\frac{\Delta V_{mem}}{\Delta t} = \frac{I_{syn\_total}}{C_{mem}} \tag{6.1}$$

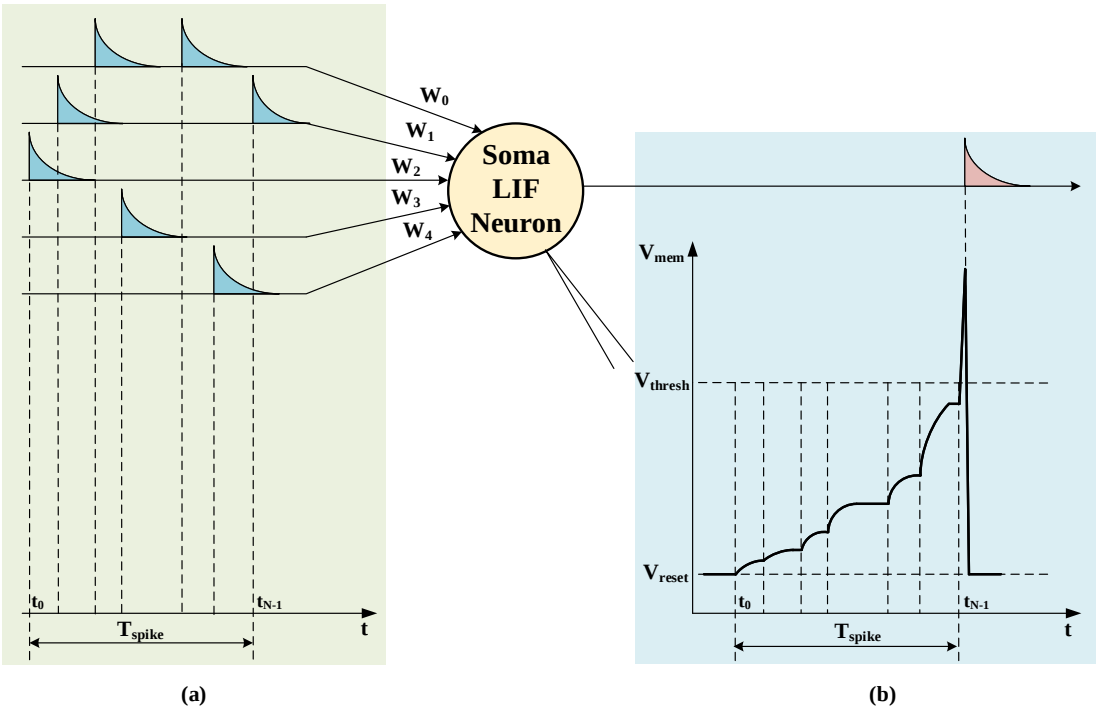


Figure 6.3: Illustration of neural network for temporal pattern recognition. (a) Input spike train applied to synaptic weights with delay. (b) Output neuron membrane voltage and spike generation.

The total synaptic current depends on the change in the individual synaptic conductance G_{syn} or the weight of the synapse in the neural network. The change in the synaptic conductance can have an exponential decay factor that helps to maintain history of the input spikes and thus provides the basis for generation of time surfaces and spike-timing based pattern recognition.

$$I_{syn\_total} = \sum_{j=1}^{N_{syn}} I_{syn_j} \tag{6.2}$$

$$I_{syn} = G_{syn} e^{\frac{-\Delta t}{\tau_{decay}}} (E_{rev} - V_{mem}) \tag{6.3}$$

6.2.2 STDP Learning Rule

STDP, the plasticity rule for synaptic weight updates based on spike times, was originally proposed in 1995 [31]. The STDP rule is temporally asymmetric form of Hebbian learning induced by tight temporal correlations between the spikes of pre- and postsynaptic neurons. In a neural network based on the STDP learning rule, neurons that fire together wire together i.e. neurons that fire around the same time increase their corresponding synaptic weight. The weight update equation for the standard STDP learning rule is given by,

$$\Delta G_{syn} = \lambda sgn(\Delta t) e^{\frac{-sgn(\Delta t)(\Delta t)}{\tau_{learn}}} \tag{6.4}$$

where $\Delta t = t_{pre} - t_{post}$, λ is the learning rate parameter. Fig. 6.4 shows the weight update plot for the STDP rule. As seen in the plot, the rule depreciates weights of pre-synaptic spikes that arrived just before the neuron fired a post-synaptic spike and vice versa, increases the weights of pre-synaptic spike that arrive after the post-synaptic spike. This method of learning with STDP is useful to form associative memories, a present event can indicate a higher expectation for a future event. This is illustrated in the classic example of a dog trained to expect food immediately after the bell ring. On the other hand, the STDP rule tends to train the network to forget immediate past history of events. Hence it is not suitable for temporal pattern recognition.

6.2.3 Loihi Architecture

Intel's Loihi chip is a platform to implement neuromorphic algorithms. Loihi is a fully asynchronous digital neuromorphic system with extensively programmable dynamics of networks of neurons and synapses [12]. Continuous functions of membrane potentials are approximated in discrete time steps whereby all neurons maintain a timestamp synchronized throughout the entire chip. Leveraging its highly configurable neurons compared with the asynchronous nature of computation, Loihi provides an ideal platform to exploit the concepts of neural computation.

Membrane potential and input current for neuron compartments on Loihi are calculated anew at each time step. Compartments will accumulate current from attached synapses and decay it using the following model:

$$i(t) = i(t-1)(2^{12} - \delta)2^{-12} + 2^{6+\eta}\sum_j \omega_j s_j(t) \tag{6.5}$$

where $\delta \in [0..2^{12}]$ is a current decay factor and $\eta \in [-8..7]$ is a weight exponent scaling factor. $\omega \in [-255..254]$ is the weight and $s \in \{0,1\}$ the spike indicator for a connected synapse j. The compartment current is then integrated into the compartment's membrane potential, calculated as follows:

$$V(t) = V(t-1)(2^{12} - \lambda)2^{-12} + i(t) \tag{6.6}$$

where $\lambda \in [0..2^{12}]$ is a voltage decay factor. As soon as $V(t)$ reaches or exceeds the membrane voltage threshold $V_{spike} = V_{th} \times 2^6$, the compartment emits a spike and V is reset to zero. V_{th} can range from 1 to 2^{16}.

6.3 Inverse STDP Learning Rule

For temporal pattern recognition, the network has to maintain history of past events in order to identify a particular sequence of events. An intuitive deduction can thus be made that a learning rule inverse to that of the STDP rule can satisfy this requirement. In contrary to STDP rule, the proposed rule has to increase weights of those pre-synaptic spikes that arrive before the post-synaptic spike. Hence the proposed Inverse STDP rule. The parameters of the SNN implementing the proposed learning rule include, the number of synapses N_{syn}, the maximum number spikes per synapse, N_{spike_max}, synaptic exponential delay time constant τ_{syn}, learning rule time constant, τ_{learn}, neuron threshold voltage V_{thresh}, neuron membrane capacitance C_{mem}, reversal potential, E_{rev}, learning rate parameter λ, time step δt and the spike-timing of the input

pulse train to each synapse, t_{pre}.

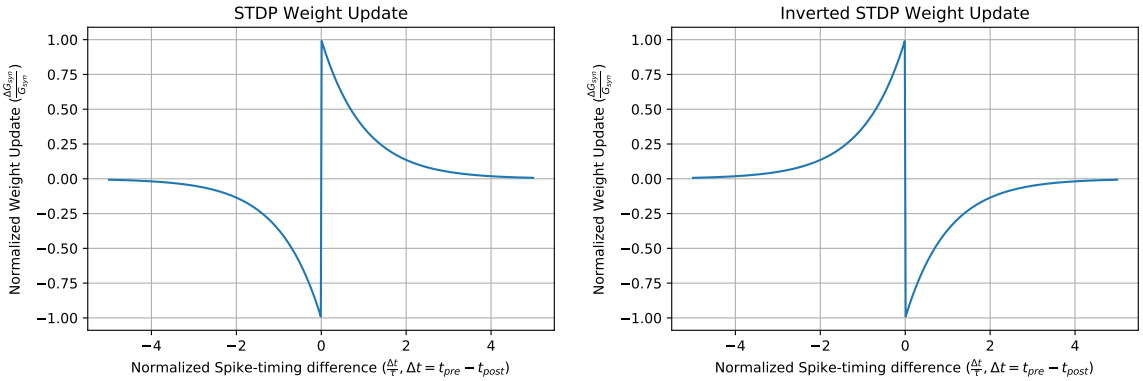


Figure 6.4: Weight update comparison of STDP vs proposed iSTDP learning rules. λ=1, α=0

The learning rule described by the weight update equation is given by,

$$\Delta G_{syn} = G_{syn\_new} - G_{syn\_prev} \tag{6.7}$$

$$\Delta G_{syn} = \alpha f(t_{post}) G_{syn\_prev} + \lambda g(t_{post} - t_{pre}) \tag{6.8}$$

where,

$$f(t) = \sum_{t_{spike} \in t_{pre}} (t > t_{spike}) - \sum_{t_{spike} \in t_{pre}} (t <= t_{spike}) \tag{6.9}$$

$$g(\Delta t) = sgn(\Delta t) e^{\frac{-sgn(\Delta t)(\Delta t)}{\tau_{learn}}} \tag{6.10}$$

where $\Delta t = t_{post} - t_{pre}$ is the inverse of the STDP rule, α is the weight scaling factor, λ is the learning rate parameter. Fig. 6.4 shows the weight update plot for the proposed inverse STDP rule. The plot clearly shows an inverted characteristic compared to that of the standard STDP rule. Fig. 6.5 shows the effect of parameter tuning of the learning rate, λ and the scaling factor, α. Tuning λ varies the maximum change in synaptic weights (ΔG), hence affecting the learning rate. Tuning α varies the residual change in synaptic weight, when the spike timing difference is large ($\Delta G \longrightarrow \pm\alpha$ as $\Delta t \longrightarrow \pm 4\tau_{learn}$), thus enables scaling of weights.

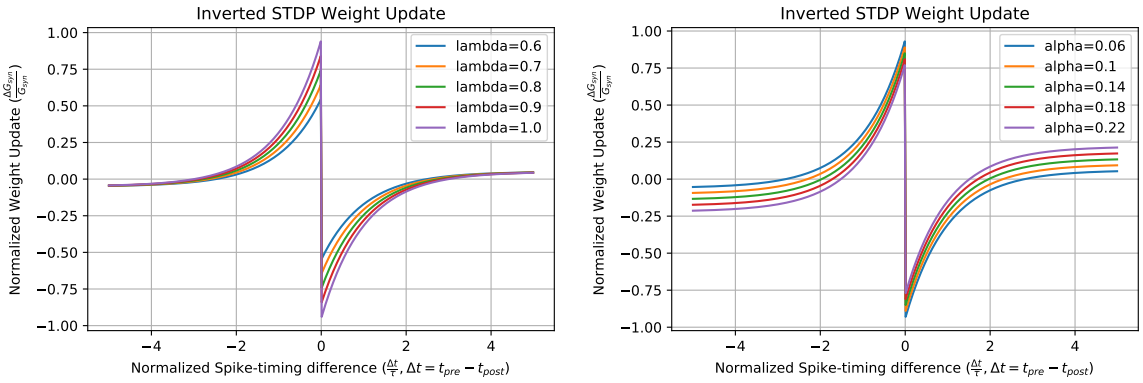


Figure 6.5: Effect of lambda and alpha parameters on weight update for proposed iSTDP learning rule.

In order to learn and recognize patterns in a limited temporal window, a network should also be able to forget its outdated history so that it does not give false output events. If the network permanently retains its history, any order of input spikes will eventually produce an output spike, since the membrane voltage keeps increasing monotonically. A simple solution to this problem is to introduce a leakage current component that keeps draining V_{mem} at a rate proportional to the leak conductance, G_{leak}.

$$\frac{\Delta V_{mem}}{\Delta t} = \frac{I_{syn\_total} - I_{leak}}{C_{mem}} \tag{6.11}$$

$$I_{leak} = G_{leak}(E_{leak} - V_{mem}) \tag{6.12}$$

The proposed inverted STDP rule is easily translatable to analog or digital neuromorphic hardware, which supports a network of integrate-and-fire neurons with programmable pre-synaptic input weights with tunable delay and a leakage component.

6.4 Implementation

The inverted STDP rule was evaluated using different networks implemented in three stages in software and hardware. Intel Loihi chip was the hardware platform chosen to port the trained networks and run inference to validate the results and extract performance metrics.

In each stage, a neural network with a pre-defined structure is chosen to implement the proposed learning rule. The synaptic weights are trained in epochs in order to converge to a reasonable range of final values, which uniquely identify the input spike-timing pattern(s). Once the synaptic weights are trained, the final weights and input spike pattern are used to run inference. During the inference stage, we monitor the membrane potential of the output neuron, V_{mem} which varies proportional to the arrival of the pre-synaptic input spikes, such that it crosses the threshold voltage, V_{thresh} definitely but only after the last input spike arrives.

6.4.1 Stage one: Inference with single neuron network

In stage one, a single neuron network is chosen as described in Fig. 6.3. This network is trained to identify one unique input spike pattern. Fig. 6.6 shows a training example depicting the evolution of the membrane voltage with each epoch. Eventually, in this case, the voltage crosses threshold and the neuron generates an output spike. In other cases, the neuron might spike too early, in which case the learning rule adjusts the weights to reduce the membrane voltage, allowing the neuron to spike later. Fig. 6.7 shows the simulation results of a trained network, where the membrane voltage normalized to the threshold voltage i.e. the neuron spikes when V_{mem_norm} exceeds 1. The figure also shows a test case where the input spikes have added jitter and time shift. The output neuron tolerates the variation and registers a spike when the expected input sequence has arrived.

In order to evaluate the specificity of the neuron, a long sequence of input spikes are presented with the trained pattern presented first followed by multiple sequences of scrambled patterns, as shown in Fig. 6.8. It can be seen the membrane voltage, V_{mem} crosses threshold only once for the correct input pattern and for all scrambled patterns, it never reaches threshold. All training done in software (Python) and inference measured in both software and hardware.

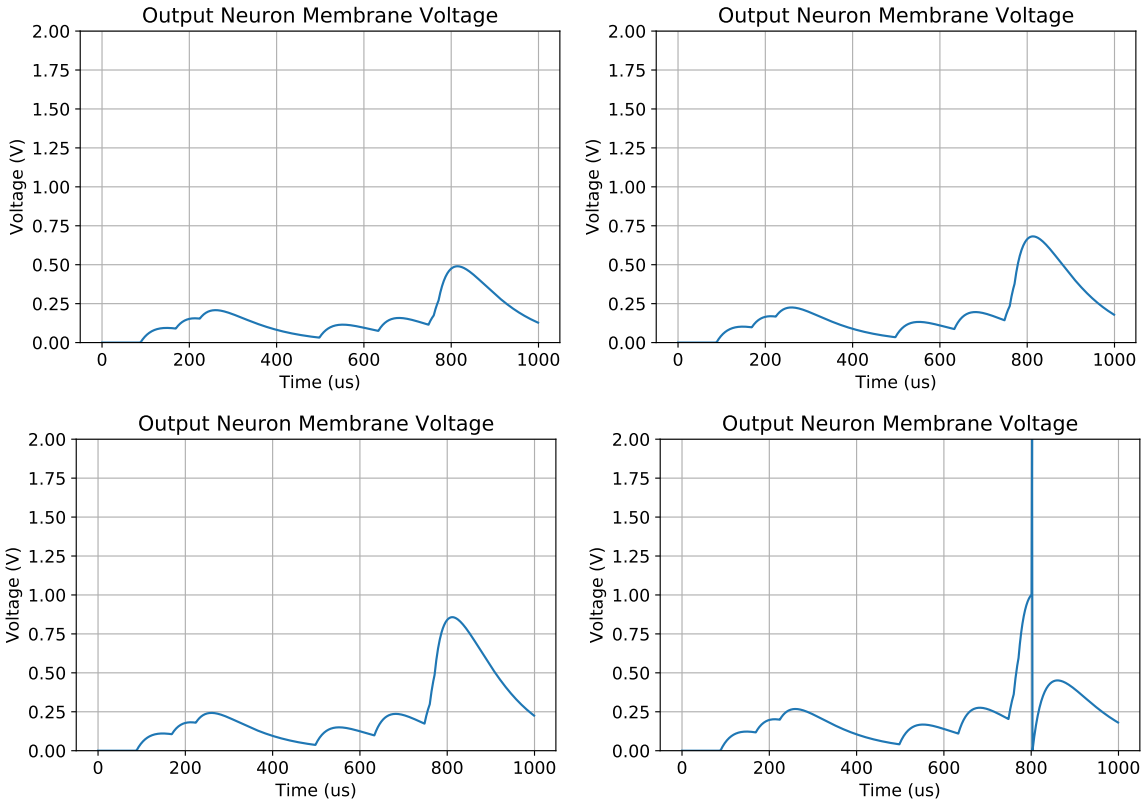


Figure 6.6: Training example illustrating evolution of membrane voltage of output neuron at each epoch.

6.4.2 Stage two: Inference with multi-layer neuron network

In stage two, a multi layer network of neurons and synapses are trained to identify distinct patterns of input spikes. A simplified version of the mus silicium was chosen as the target problem, as described in [24]. This network is suitable for implementing keyword spotting, where timing is critical to ensure reliable and low power solution. The network consists of an input layer of 40 channels that produce one spike per channel for each distinct input pattern. 5 labelled patterns are chosen, each label constituting a 40 channel input vector. The input layer connects to a hidden layer of 5 neurons, with each neuron having 40 input synaptic connections. The hidden layer is trained to identify each label using a single neuron only. The output layer is a simple Winner Take All (WTA), where the first hidden layer neuron to spike will be considered the winner and the output predicted to the corresponding label.

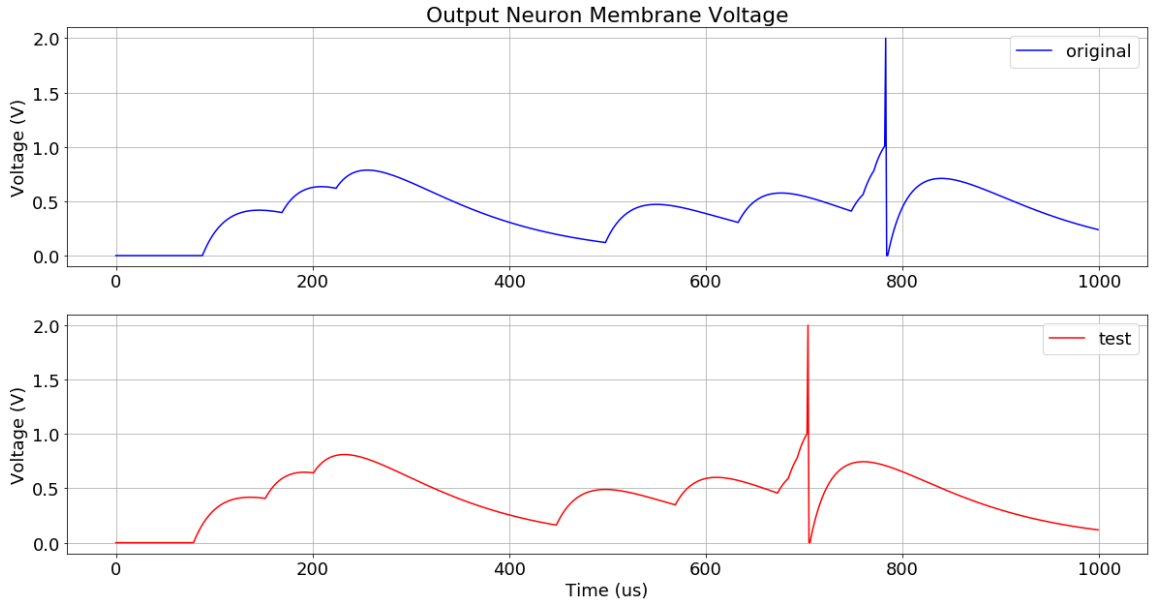


Figure 6.7: Simulation results of a network with 10 pre-synaptic input neurons and 1 post-synaptic output neuron. Original trained network output is shown in blue and test input with shifted spike times is shown in red. Output neuron spikes when the relative order of input spikes is maintained.

Once training is completed, the network is tested with various combinations of input spike patterns to check if the output label predicted matches the actual input label. A cross-validation matrix is generated to estimate the number of true positives, true negatives, false positives and false negatives. Fig.6.9 presents the results of stage two, with input spike patterns for each label are presented sequentially and the output pattern matches correctly. All training done in software (Python) and inference measured in both software and hardware.

The trained network is ported on Loihi, which is able to leverage the full potential of asynchronous computation. A single run to feed the spike trains for all 5 labels through the whole network takes 133ns and uses 200nJ of energy. These numbers were averaged across 20 runs.

6.4.3 Stage three: Training on Loihi

It is possible to implement the proposed learning rule directly on Loihi, since it depends only on the pre and post-synaptic spike timing. Implementing learning rules on Loihi uses pre-

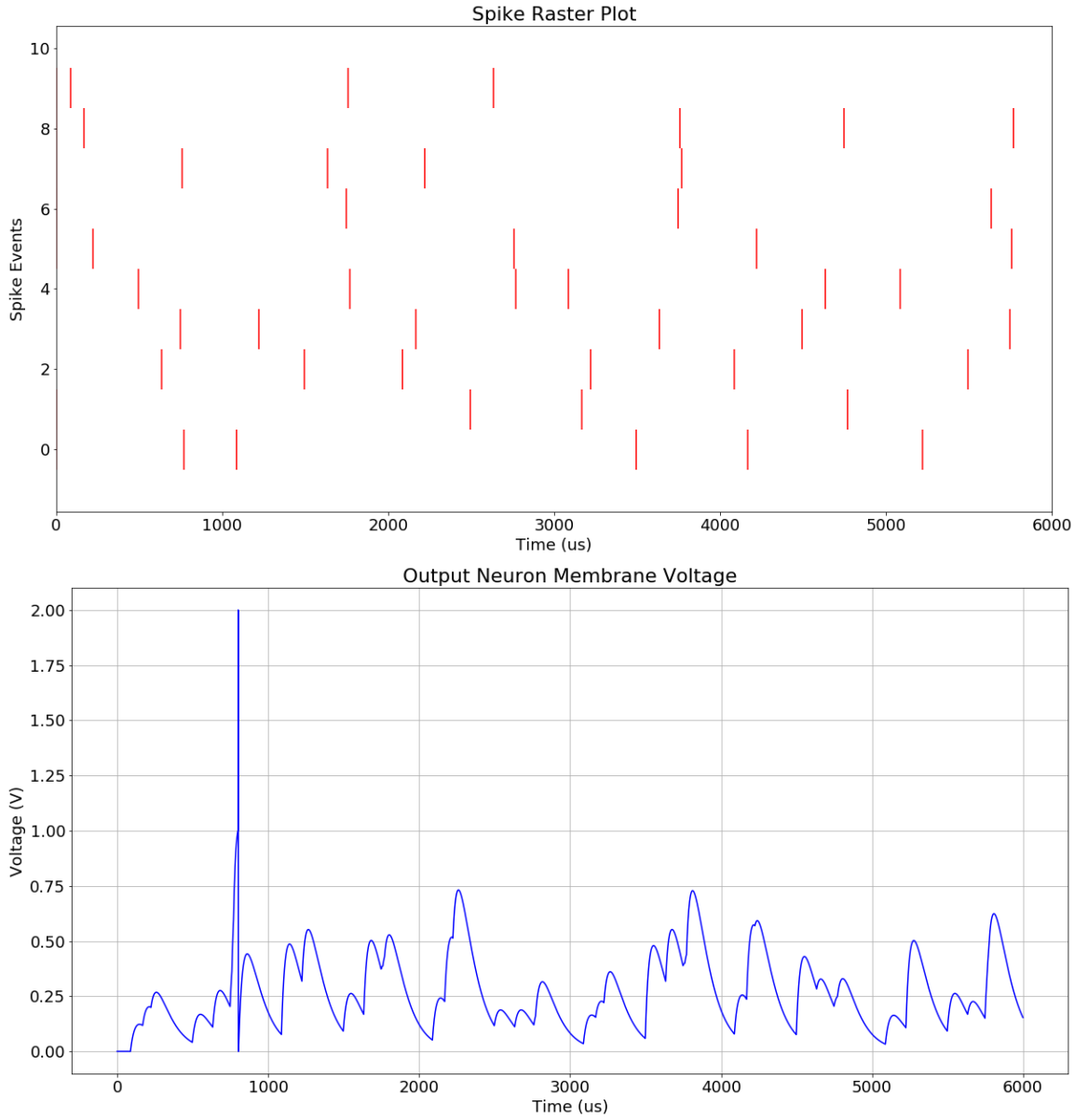


Figure 6.8: Output neuron membrane voltage for a long sequence of input events with scrambled patterns. Neuron fires after the first sequence which is the trained pattern. For the other scrambled patterns, the neuron does not fire.

and postsynaptic traces that are automatically generated. Fig. 6.10 shows the weight changes of the learning-enabled synapse as a function of the difference between post and pre synaptic spikes that preceed the weight change. The tuning curve shows the expected behavior for iSTDP learning. Small anti-causal spike events give rise to a large increase of the weight which decreases

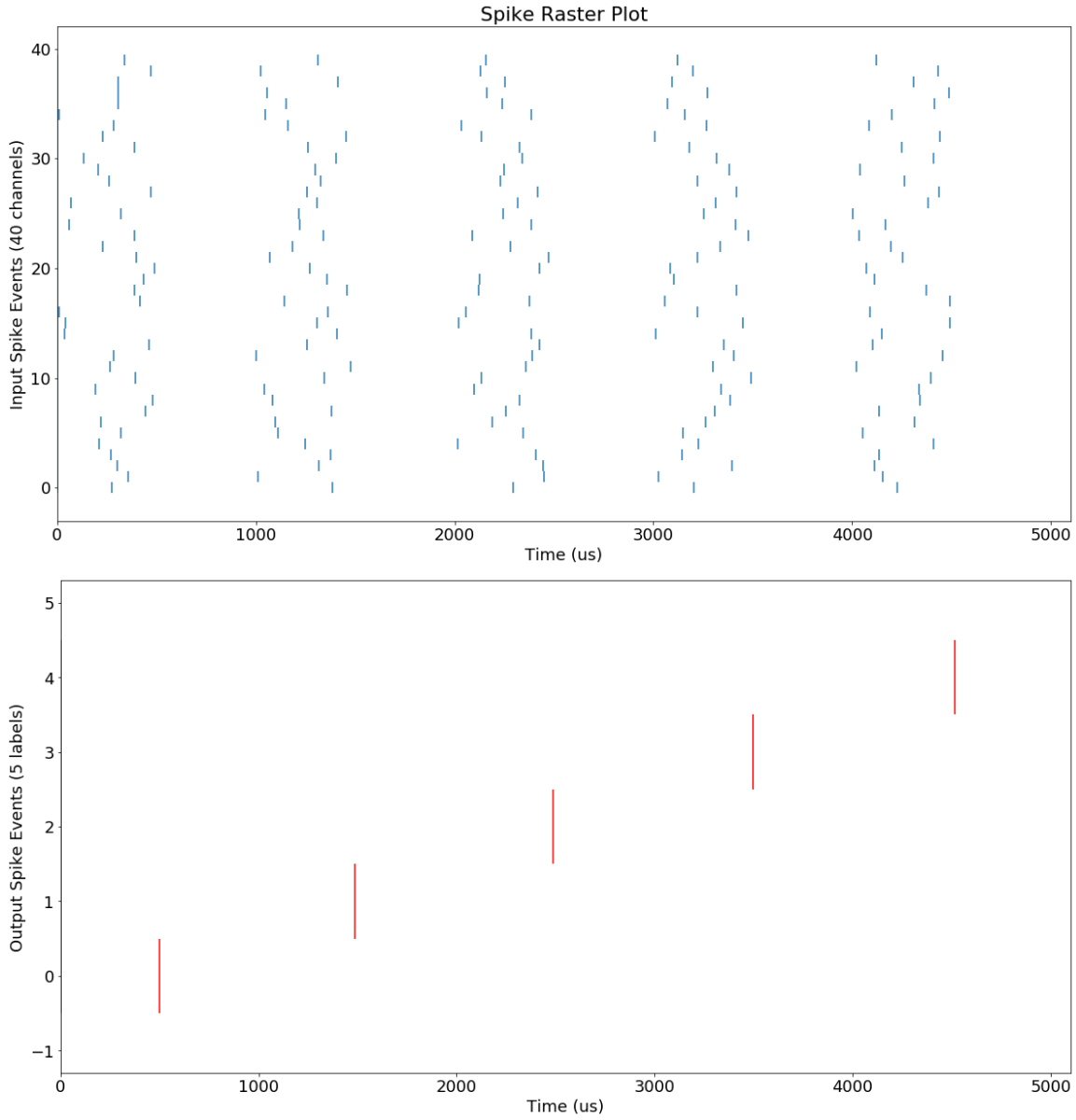


Figure 6.9: Loihi implementation of a 2 layer network to identify 5 distinct output labels using input spike patterns from 40 channels.

as the time between pre and post synaptic spike times decreases. Also, small causal spike events give rise to large decreases of the weight which increases as the time between post and pre synaptic spike times increases.

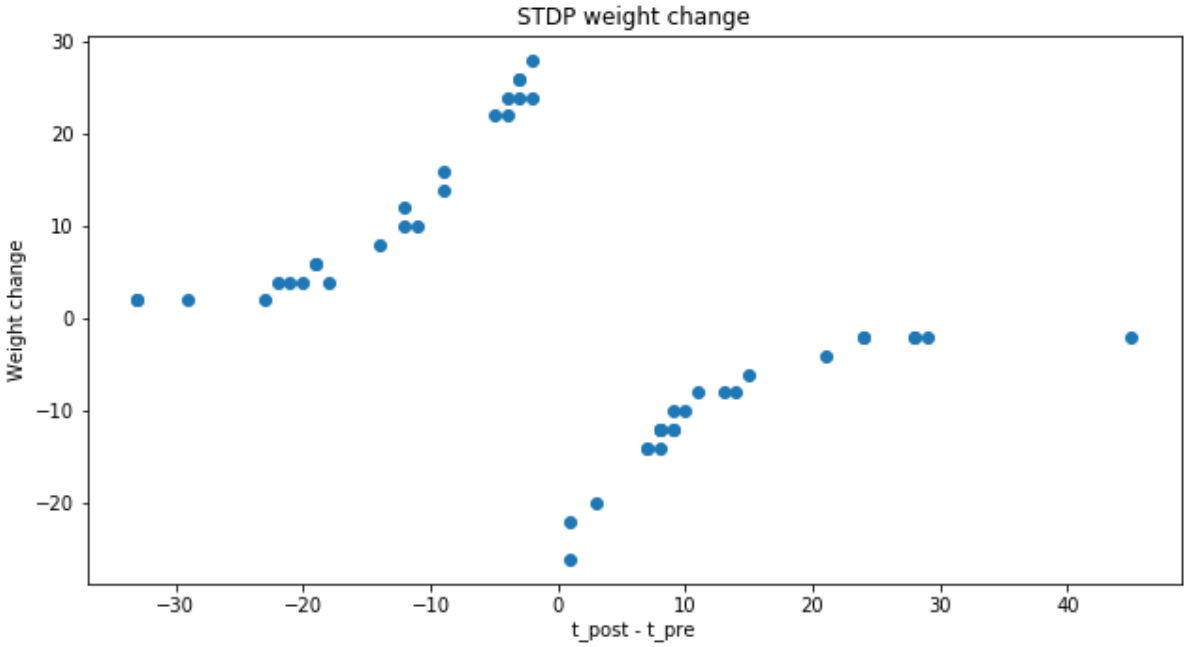


Figure 6.10: Loihi learning rule updates.

Table 6.1: Pattern Recognition Error Comparison

Network	Error
Wills, 2001	0.253
SKIM, Delay plus Gaussian, 2013	0.169
Hopfield and Brody, 2013	0.15
This work	0.111

6.5 Conclusion

The error in detecting input patterns for stage two implementation was calculated as follows,

$$error = \frac{false\ negatives}{true\ positives} + \frac{false\ positives}{true\ negatives} \tag{6.13}$$

Table 6.1 shows the error numbers in comparison to previous implementations.

In order to guarantee convergence during training, hyper-parameter tuning is essential. For the proposed rule implemented on the Loihi platform, the following considerations proved to be highly useful during training.

Loihi parameters

- Initialize all weights, G_{syn}, to a small value. Random initialization takes a lot longer to converge or does not converge at all.
- Keep δ and number of time steps, t_{steps} as small as possible, to ensure the decay kernels overlap and time surface is properly generated. This is especially useful with higher fan-in i.e. larger number of input synapses.
- Keep η lower to allow final weights to be in reasonable range. If training updates result in weights either too low or too high, they are clipped using lower and upper bounds to make them realizable on hardware. However, extreme weights tend to suppress many valid input spikes or highlight a few important ones that result in poor performance and generalization.

Inverted STDP parameters

- Keep τ_{learn} small and λ large enough to learn faster
- Tune α optimally so that weights adjust fast enough, especially when V_{mem} spikes too early

This chapter presents a learning algorithm called Inverted STDP rule to learn input spike patterns based on their spike timing, using time surfaces generated by synaptic decay kernels. The theory and equations for the learning rule are described in detail. The effects of tuning parameters of the weight update rule are analysed and discussed. The learning algorithm was used to train various networks implemented in both software and hardware, demonstrating the ease of porting to hardware. The algorithm can be used in many applications where timing is critical, such as keyword spotting, robotic control and visual tracking.

6.6 Acknowledgments

Chapter 6 is in part a reprint of the material as it appears in Kubendran, R., Ieng, S.H., Orchard, G., Cauwenberghs, G., and Benosman, R., “An inverse STDP learning rule using

synaptic delay kernel for spike-timing based temporal pattern recognition." *arXiv preprint arXiv:1601.04187* (under review with IEEE Transactions on Neural Networks and Learning Systems), 2020. The dissertation author was the primary investigator and author of this paper. This study was sponsored in part by the National Science Foundation (NSF CCF-1317373), and the Office of Naval Research (ONR MURI N00014-13-1-0205). The views and the conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Chapter 7

Conclusions and Future Work

In this dissertation, we presented the query-driven dynamic vision sensor (*q*DVS) which offers lowest reported power consumption of visual event coding and streaming, normalized for process technology and pixel array resolution. Activity based event streaming to monitor pixel activity and corresponding modulation of clock frequency can reduce output rate and power even further. The applications of event-based vision sensors was elaborated. The prototype developed during this thesis demonstrates superior performance in varied applications such as stereo vision, video compression and visual attention.

A two-tier hybrid asynchronous ADC was presented as an efficient hierarchical alternative to a flat architecture, avoiding the excessive energy and area demands for large numbers of level-crossing detectors at high resolution. The hybrid architecture reduces overall power and area for a 10-b design by resolving 5-b MSBs with a 32-thermometer-coded capacitive voltage DAC, and the remaining 5-b LSBs for the folded residue with an array of level-crossing comparators. Energy efficiency of synchronous ADCs depend on the clock frequency, which can be reduced by not overclocking the system depending on its bandwidth requirements, whereas for clock-free asynchronous ADCs, the energy efficiency is dependent on the input signal sparsity. This topology achieves appreciably improved FoM by keeping the energy per level conversion low particularly

for high resolution. The main limitation of the two-tier scheme is the need for accurate and high-bandwidth residue amplification, requiring power overhead and incurring latency at major transitions. Gray coding of the digital output minimizes impact of system-level latency, as the effect of jitter in event registration amounts to errors bound to the LSB level. The chip was experimentally validated for tracking of continuous input signals, including a ramp, sinusoidal and ECG signals.

We then presented the first system-level demonstration of a high-efficiency, fully integrated CMOS-RRAM CIM architecture with core-level support for a diverse set of NN architectures and dataflows including probabilistic graphical models. Multi-core extensions of this work open the door to advances in artificial intelligence at a large scale with extreme energy-efficiency and integration density. An ultra-low power neuron array transceiver with a multi-modal integrate-and-fire neuron architecture was presented. A variety of activation functions were realized by using additional pseudo-random noise or partial reset mechanism that makes this chip versatile to cater to different ANN architectures. The highly reconfigurable and energy efficient design makes this transceiver suitable for deep learning applications such as RBMs, CNNs using neurons with ReLU or sigmoidal activation. Neurons with step or sigmoidal activation, with binary or ternary output levels, can be used for both rate coding or spike timing based neuromorphic applications. Combining *q*DVS with the custom computer-in-memory hardware accelerator for AI applications, this setup has the potential to be the most energy-efficient system to-date for sensing and recognizing objects real-time.

We also presented a weight update learning rule based only on pre- and post-synaptic spike times for spike timing based pattern recognition. This rule allows to train a network of neurons and synapses to identify an unique sequence in the timing of input spike events. The proposed learning rule translates readily to neuromorphic hardware such as the Intel's Loihi platform. Implementation results were shown for both software and hardware implementations.

Our future work will continue to improve the design of dynamic vision sensors to include

more features for spatio-temporal event generation and processing and provide high impact applications in various fields involving computer vision like always-on surveillance, autonomous drone navigation and augmented reality. Another promising frontier is to fully integrate the sensor and processor into one architecture using advanced 3D integration technologies to bring us much closer towards efficient, robust, resilient, and autonomous bio-inspired vision.

Bibliography

[1] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam, et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(10):1537–1557, 2015.

[2] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi. Event-based visual flow. *IEEE transactions on neural networks and learning systems*, 25(2):407–417, 2013.

[3] R. Benosman, S. Kang, and O. Faugeras. *Panoramic vision*. Springer Verlag, 2000.

[4] A. Borst and M. Egelhaaf. Principles of visual motion detection. *Trends in neurosciences*, 12(8):297–306, 1989.

[5] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck. A 240$\times$ 180 130 db 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014.

[6] G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola, et al. Neuromorphic computing using non-volatile memory. *Advances in Physics: X*, 2(1):89–124, 2017.

[7] F. Cai, J. M. Correll, S. H. Lee, Y. Lim, V. Bothra, Z. Zhang, M. P. Flynn, and W. D. Lu. A fully integrated reprogrammable memristor–cmos system for efficient multiply–accumulate operations. *Nature Electronics*, 2(7):290–299, 2019.

[8] G. Cauwenberghs. Reverse engineering the cognitive brain. *Proceedings of the national academy of sciences*, 110(39):15512–15513, 2013.

[9] S. Chen and M. Guo. Live demonstration: Celex-v: A 1m pixel multi-mode event-based sensor. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1682–1683, 2019.

[10] W.-H. Chen, K.-X. Li, W.-Y. Lin, K.-H. Hsu, P.-Y. Li, C.-H. Yang, C.-X. Xue, E.-Y. Yang, Y.-K. Chen, Y.-S. Chang, et al. A 65nm 1mb nonvolatile computing-in-memory reram macro with sub-16ns multiply-and-accumulate for binary dnn ai edge processors. In *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 494–496. IEEE, 2018.

[11] Y. Chi, U. Mallik, M. Clapp, E. Choi, G. Cauwenberghs, and R. Etienne Cummings. Cmos camera with in-pixel temporal change detection and adc. *IEEE Journal of Solid State Circuits*, 42(10):2187–2196, 2007.

[12] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[13] T. Delbruck. Frame-free dynamic digital vision. *Proceedings of the International Symposium of Secure-Life Electronics*, 1:21–26, 2008.

[14] T. Delbruck, B. Linares-Barranco, E. Culurciello, and C. Posch. Activity-driven, event-based vision sensors. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 2426–2429. IEEE, 2010.

[15] T. Finateu, A. Niwa, D. Matolin, K. Tsuchimoto, A. Mascheroni, E. Reynaud, P. Mostafalu, F. Brady, L. Chotard, F. LeGoff, H. Takahashi, H. Wakabayashi, Y. Oike, and C. 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.

[16] S. Furber. Large-scale neuromorphic computing systems. *Journal of neural engineering*, 13(5):051001, 2016.

[17] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza. Event-based vision: A survey. *CoRR*, abs/1904.08405, 2019.

[18] K. Hazelwood, S. Bird, D. Brooks, S. Chintala, U. Diril, D. Dzhulgakov, M. Fawzy, B. Jia, Y. Jia, A. Kalro, et al. Applied machine learning at facebook: A datacenter infrastructure perspective. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 620–629. IEEE, 2018.

[19] B. Hershberg, B. van Liempd, N. Markulic, J. Lagos, E. Martens, D. Dermit, and J. Craninckx. 3.6 a 6-to-600ms/s fully dynamic ringamp pipelined adc with asynchronous event-driven clocking in 16nm. In *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 68–70. IEEE, 2019.

[20] L. N. Huynh, Y. Lee, and R. K. Balan. Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 82–95, 2017.

[21] G. Indiveri and R. Douglas. Neuromorphic vision sensors. *Science*, 288(5469):1189–1190, 2000.

[22] G. Indiveri, B. Linares-Barranco, T. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. SAÏGHI, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen. Neuromorphic silicon neuron circuits. *Frontiers in Neuroscience*, 5:73, 2011.

[23] Insightness. Insightness event-based sensor modules. *Insightness Product Manual*, 2020.

[24] T. Jonathan, C. Greg, A. Saeed, S. Klaus, B. Yossi, H. Tara, and van Schaik André. Synthesis of neural networks for spatio-temporal spike pattern recognition and processing. *Front. Neurosci.*, 7:153, Aug. 2013.

[25] J. Kaiser, H. Mostafa, and E. Neftci. Synaptic plasticity dynamics for deep continuous local learning (decolle). *arXiv preprint arXiv:1811.10766*, 2018.

[26] X. Lagorce, G. Orchard, F. Gallupi, B. E. Shi, and R. Benosman. HOTS: A hierarchy of event-based time-surfaces for pattern recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(7):1346–1359, July 2017.

[27] F. Lanusse, Q. Ma, N. Li, T. E. Collett, C.-L. Li, S. Ravanbakhsh, R. Mandelbaum, and B. Póczos. Cmu deeplens: deep learning for automatic image-based galaxy–galaxy strong lens finding. *Monthly Notices of the Royal Astronomical Society*, 473(3):3895–3906, 2018.

[28] H. E. Lee, J. H. Park, T. J. Kim, D. Im, J. H. Shin, D. H. Kim, B. Mohammad, I.-S. Kang, and K. J. Lee. Novel electronics for flexible and neuromorphic computing. *Advanced Functional Materials*, 28(32):1801690, 2018.

[29] C.-K. Lin, A. Wild, G. N. Chinya, Y. Cao, M. Davies, D. M. Lavery, and H. Wang. Programming spiking neural networks on intel's loihi. *Computer*, 51(3):52–61, 2018.

[30] M. Mahowald. *An analog VLSI system for stereoscopic vision*, volume 265. Springer Science & Business Media, 1994.

[31] H. Markram, W. Gerstner, and P. J. Sjöström. Spike-timing-dependent plasticity: a comprehensive overview. *Frontiers in synaptic neuroscience*, 4:2, 2012.

[32] A. J. Martin and M. Nystrom. Asynchronous techniques for system-on-chip design. *Proceedings of the IEEE*, 94(6):1089–1120, 2006.

[33] S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324, 2007.

[34] R. Mochida, K. Kouno, Y. Hayata, M. Nakayama, T. Ono, H. Suwa, R. Yasuhara, K. Katayama, T. Mikawa, and Y. Gohou. A 4m synapses integrated analog reram based 66.5 tops/w neural-network processor with cell current controlled writing and flexible network architecture. In *2018 IEEE Symposium on VLSI Technology*, pages 175–176. IEEE, 2018.

[35] M. Mozer. A focused backpropagation algorithm for temporal pattern recognition. *Complex Systems*, 3, 01 1995.

[36] A. Neckar, S. Fok, B. V. Benjamin, T. C. Stewart, N. N. Oza, A. R. Voelker, C. Eliasmith, R. Manohar, and K. Boahen. Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model. *Proceedings of the IEEE*, 107(1):144–164, 2019.

[37] J. Park, S. Ha, T. Yu, E. Neftci, and G. Cauwenberghs. A 65k-neuron 73-Mevents/s 22-pJ/event asynchronous micro-pipelined integrate-and-fire array transceiver. In *Proceedings of the IEEE Biomedical Circuits and Systems Conference*, pages 675–678, Oct 2014.

[38] J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He, et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

[39] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in Neuroscience*, 9, Apr. 2015.

[40] Q. Rao and J. Frtunikj. Deep learning for self-driving cars: chances and challenges. In *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*, pages 35–38, 2018.

[41] J. C. Read. Stereo vision and strabismus. *Eye*, 29(2):214–224, 2015.

[42] I. E. Richardson. *H. 264 and MPEG-4 video compression: video coding for next-generation multimedia*. John Wiley & Sons, 2004.

[43] A. Saeed, G. Libin, T. Jonathan, van Schaik André, and T. J. Hamilton. Racing to learn: statistical inference and learning in a single spiking neuron with adaptive kernels. *Front. Neurosci.*, 8:377, Nov. 2014.

[44] B. Schell and Y. Tsividis. A continuous-time ADC/DSP/DAC system with no clock and with activity-dependent power dissipation. *IEEE Journal of Solid-State Circuits*, 43(11):2472–2481, 2008.

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

[46] F. Seide and A. Agarwal. Cntk: Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2135–2135, 2016.

[47] S. B. Shreshtha and G. Orchard. Slayer: Spike layer error reassignment in time. *32nd Conference on Neural Information Processing Systems*, 2018.

[48] R. Socher, Y. Bengio, and C. D. Manning. Deep learning for nlp (without magic). In *Tutorial Abstracts of ACL 2012*, pages 5–5. ACM, 2012.

[49] B. Son, Y. Suh, S. Kim, H. Jung, J.-S. Kim, C. Shin, K. Park, K. Lee, J. Park, J. Woo, et al. A 640× 480 dynamic vision sensor with a 9μm pixel and 300meps address-event representation. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 66–67. IEEE, 2017.

[50] M. Trakimas and S. R. Sonkusale. An adaptive resolution asynchronous ADC architecture for data compression in energy constrained sensing applications. *IEEE Transactions on Circuits and Systems. I, Reg. Papers*, 58(5):921–934, 2011.

[51] B. Vaz, A. Lynam, B. Verbruggen, A. Laraba, C. Mesadri, A. Boumaalif, J. Mcgrath, U. Kamath, R. De Le Torre, A. Manlapat, D. Breathnach, C. Erdmann, and B. Farley. 16.1 a 13b 4gs/s digitally assisted dynamic 3-stage asynchronous pipelined-sar adc. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 276–277, 2017.

[52] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[53] W. Wan, R. Kubendran, S. B. Eryilmaz, W. Zhang, Y. Liao, D. Wu, S. Deiss, B. Gao, P. Raina, S. Joshi, et al. A 74 tmacs/w cmos-rram neurosynaptic core with dynamically reconfigurable dataflow and in-situ transposable weights for probabilistic graphical models. In *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 498–500. IEEE, 2020.

[54] C. Weltin-Wu and Y. Tsividis. An event-driven, alias-free ADC with signal-dependent resolution. *Proc. IEEE Symp. VLSI Technology and Circuits*, pages 28–29, 2012.

[55] Y. Wu and T. S. Huang. Vision-based gesture recognition: A review. In *International Gesture Workshop*, pages 103–115. Springer, 1999.

[56] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418, 2013.

[57] C.-X. Xue, W.-H. Chen, J.-S. Liu, J.-F. Li, W.-Y. Lin, W.-E. Lin, J.-H. Wang, W.-C. Wei, T.-W. Chang, T.-C. Chang, et al. A 1mb multibit reram computing-in-memory macro with 14.6 ns parallel mac computing time for cnn based ai edge processors. In *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 388–390. IEEE, 2019.

[58] O. Yadid Pecht and R. Etienne Cummings. *CMOS Imagers: From Phototransduction to Image Processing*. Springer, 2004.

[59] M. Yang, S.-C. Liu, and T. Delbruck. A dynamic vision sensor with 1% temporal contrast sensitivity and in-pixel asynchronous delta modulator for event encoding. *IEEE Journal of Solid-State Circuits*, 50(9):2149–2160, 2015.

[60] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13–es, 2006.

[61] L. Yongjia, Z. Duan, and A. S. Wouter. A Sub-Microwatt Asynchronous Level-Crossing ADC for Biomedical Applications. *IEEE Transactions on Biomedical Circuits and Systems*, 7(2):149–157, 2013.

[62] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays*, pages 161–170, 2015.

[63] X. Zheng, R. Zarcone, D. Paiton, J. Sohn, W. Wan, B. Olshausen, and H.-S. P. Wong. Error-resilient analog image storage and compression with analog-valued rram arrays: An adaptive joint source-channel coding approach. In *2018 IEEE International Electron Devices Meeting (IEDM)*, pages 3–5. IEEE, 2018.