

國立臺灣大學電機資訊學院電子工程學研究所

博士論文

Graduate Institute of Electronics Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Doctoral Dissertation

優化脈衝神經網絡推理：
架構設計與算法增強的協同方法

Optimizing Spiking Neural Network Inference:
A Synergistic Approach to Architecture Design and
Algorithmic Enhancement

魏旻良

Ming-Liang Wei

指導教授: 楊佳玲 博士

Advisor: Chia-Lin Yang Ph.D.

中華民國 113 年 2 月

February, 2024



國立臺灣大學博士學位論文

口試委員會審定書

DOCTORAL DISSERTATION ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

優化脈衝神經網絡推理：

架構設計與算法增強的協同方法

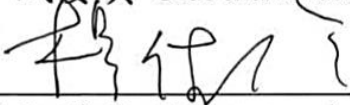

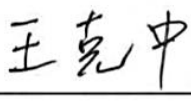
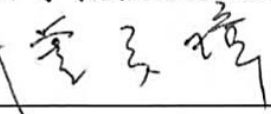
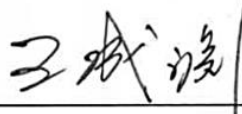
Optimizing Spiking Neural Network Inference:


A Synergistic Approach to Architecture Design and Algorithmic
Enhancement

本論文係 魏旻良 (D04943004) 在國立臺灣大學電子工程學研究所完成之博士學位論文，於民國 113 年 1 月 12 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Graduate Institute of Electronics Engineering on 12nd January 2024 have examined a Doctoral Dissertation entitled above presented by Ming-Liang Wei (D04943004) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

 (指導教授 Advisor)		
		

所 長 Director: 





致謝

感謝楊佳玲教授的細心指導、鞠躬盡瘁的栽培以及深刻的見解。在科學探索和個人發展的旅程中，她的支持讓我積極面對各種挑戰，並促成了我的思維模式的轉變與成長。感謝王克中技術總監、吳安宇教授、呂函庭處長、Hussam Amrouch 教授、鄭湘筠研究員在我攻讀博士的過程中，給我的幫助與指教。感謝王成淵協理、盧奕璋教授以及所有口試評審委員對我的論文進行審閱並給予寶貴的建議。感謝 Mikail Yayla 學術夥伴，在一同攻讀博士的過程中，給我許多的協助。感謝父母，在我成長過程給我的栽培與支持。感謝吳志宏教授、吳謂勝教授與各個學習階段的師長給我的鼓勵與教導。感謝又愷、名翔、元皓與實驗室同學的討論及協助。同時也感謝許多關照我的長輩，及公司同仁的照應與鼓勵。要感謝的人實在太多，不勝枚舉。我對在這段學術征程中遇到的每一位朋友、導師和同儕的感激之情無以名狀。

最終要向我的賢內助，葉珈華，表達最深的感謝。在我追求博士學位的艱難旅程中，她以無聲的支持與堅定的陪伴，讓我面對學術與生活挑戰時始終保持前進的勇氣和決心。她的理解與鼓勵，成為我在追尋學術卓越和個人成長道路上的重要力量。





摘要

作為人工智能領域的一項重大進展，脈衝神經網絡以其獨特且受大腦啟發的解題、推理和推斷方法，展現了巨大潛力。這種潛力促使眾多組織和企業投入於算法和硬件設計的開發，目標是應對並克服未來的技術挑戰。然而，現有的硬件和算法解決方案在仿真硬件的成本效益及深度神經模型的推理效率方面存在局限。本研究提出了一種旨在實現高成本效益和高效運作的硬件設計與算法改進。在硬體層面，該硬件使用計算型非揮發性記憶體，其特點是無需權重移動、低比特成本和提高的面積效率，這些特性共同提升了處理能力和成本效益。在算法層面，本研究開發了專門為脈衝神經網絡加速器設計的神經模型。為了實現此硬件與演算法的共同優化，該研究著重於三個部分：(1) 由於記憶體單元的不完美而導致的可靠性降低；(2) 非揮發性記憶體的大粒度與高延遲造成效能損失；以及 (3) 為實現準確的計算結果而需多次脈衝處理次數。針對圖像識別與解題應用，我們對各種記憶裝置進行了可靠性分析。此外，我提出了架構設計以減緩處理速度損失，並提出將神經模型轉化為二進制神經網絡，以微小的準確度的犧牲，換得顯著提升分類推理的能效。我們的研究結果指出，在校準具有小開關比和高信噪比的單元時，會有巨大的電容成本，為此應選用電晶體為主的記憶體。此外，我們的設計在 MAX-CUT、數獨和 LASSO 任務上明顯優於以往的數位型 SNN 處理器，分別實現了 3.1 倍、1.8 倍和 2.2 倍的加速。最後，與 4 位元 SNN 相比，我們整合的容錯二進制神經網絡不僅將電容大小減半，能源消耗減少了 57%，同時也大幅降低了兩個數量級的延遲，並保持了分類的準確性。

關鍵字：脈衝神經網路、快閃記憶體、記憶體內運算、神經態運算、可靠度分析





Abstract

As a significant advancement in the field of artificial intelligence, spiking neural networks showcase a unique, brain-inspired approach to computational problem-solving, reasoning, and inference. This notable potential drives various organizations and industries to dedicate efforts to the development of algorithms and hardware design, aiming to achieve a substantial leap in addressing and overcoming imminent technological challenges of the future. However, the existing hardware and algorithmic solutions demonstrate a lack of cost-effectiveness in terms of emulation hardware and inefficiency in the inference of deep neural models. This thesis presents a hardware design and algorithmic improvements aimed at achieving cost-effective and efficient operation. From a hardware perspective, the hardware utilizes computational non-volatile memory, characterized by its ability to operate without weight movement, lower bit costs, and improved area efficiency, which collectively enhances processing throughput at a reduced cost compared to prior architectures. On the algorithmic front, this study develops a neural model specially tailored for a spiking neural network accelerator. However, to fully implement this inference system, it is necessary to address three critical challenges: (1) the decline in reliability due to imperfections in memory cells, (2) the considerable size and extended latency associated with non-volatile memory macros, and (3) the requirement for multiple spiking processing cycles to achieve accurate computational outcomes. In response, we conduct a reliability analysis of various memory devices for image classification and optimization problem-solving. Additionally, we delve into architectural designs to counteract losses in processing throughput, especially when dealing with sparse inputs or a limited input degree of network model inference. We also propose transforming neural models into binary neural networks to substantially improve processing speed and energy efficiency with minor sacrifices in accuracy for image classification. Our results highlight the substantial capacitance expense incurred when calibrating cells with a minimal ON-OFF ratio and a high signal-to-noise ratio. Thus, transistor-based memory is suggested. The design we

propose significantly outperforms previous digital-based SNN processors, achieving processing speeds that are 3.1x, 1.8x, and 2.2x faster for MAX-CUT, SUDOKU, and LASSO tasks, respectively. In addition, when compared to 4-bit SNNs, our approach of integrating error-resilient binary neural networks (ER-BNN) into the probabilistic inference machine not only cuts the capacitor size by 50% and reduces energy consumption by 57%, but also significantly decreases latency, all while preserving the accuracy of classification.

Keywords: Spiking Neural Network, Flash Memory, Computing in Memory, Neuromorphic Computing, Reliability Analysis

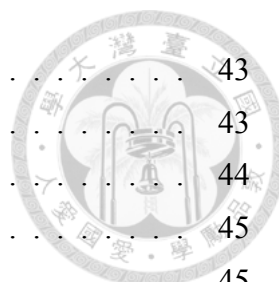


Contents

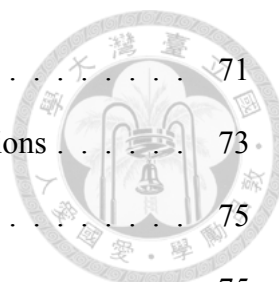
	Page
Doctoral Dissertation Acceptance Certificate	i
致謝	iii
摘要	v
Abstract	vii
Contents	ix
List of Figures	xv
List of Tables	xix
Denotation	xxi
Chapter 1 Introduction	1
1.1 Inference of Spiking Neural Network	1
1.2 Reliability Analysis for Non-volatile Synaptic Devices	2
1.3 Architecture Design for Efficient Inference	4
1.4 Algorithmic Enhancement for Image Recognition	6
1.5 Organization of the Dissertation	8
Chapter 2 Related Work	9
2.1 Reliability Analysis of Processing in Memory	9
2.2 Spiking Neural Network Architecture	10
2.3 Binarized Spiking Neural Network	11

Chapter 3	Impact of Non-volatile Memory Devices on Spiking Neural Network	13
3.1	Background and Motivation	13
3.1.1	Probabilistic Inference with Spiking Neurons	13
3.1.2	SNN Model and IF Circuit	14
3.1.3	Precision Modulation for Image Recognition	16
3.1.3.1	Neural Network with Quantized Input	16
3.1.3.2	Precision Modulation	17
3.1.4	Stochastic Annealing for Optimization Solver	18
3.1.4.1	Constraint Satisfaction Problem	18
3.1.4.2	SNN Annealing Machine	20
3.1.5	Unreliability Issues in SNNs	23
3.1.5.1	Impact of Memory Cells	24
3.1.5.2	Impact of Circuit Constraints	24
3.1.5.3	Overall Scope of Reliability Analysis	26
3.2	Design Tradeoff	27
3.2.0.1	Impact of Capacitance	28
3.2.0.2	Impact of Cell Current on C_{mem}	30
3.2.0.3	Current Variation	30
3.2.0.4	Leak Current from I_{OFF}	31
3.3	Reliability Evaluation for Image Classification	33
3.3.1	Evaluation Setup	33
3.3.1.1	Simulation Framework	33
3.3.1.2	Collection of Various Technologies	34
3.3.2	Experiment Result	37
3.3.2.1	ON-OFF Ratio	37
3.3.2.2	ON-state Current	38
3.3.2.3	Normalized Standard Deviation	40
3.3.3	Discussion	41
3.4	Reliability Evaluation for Annealing Purpose	43





3.4.1	Evaluation Setup	43
3.4.1.1	Collection of CSP Models	43
3.4.1.2	SNN Model Setup	44
3.4.1.3	Energy Model Setup	45
3.4.1.4	Collection of Memory Devices	45
3.4.2	Experiment Result	48
3.4.2.1	Impact of Current Variation	48
3.4.2.2	Cost of Calibrating Current Variation	49
3.4.2.3	Impact of Leak Current	50
3.4.2.4	Impact of Temperature	51
Chapter 4 A Neuromorphic Spiking Signal Processor with Advanced Mem- ory Technology		57
4.1	Background	57
4.1.1	SNN Annealing Machine	57
4.1.1.1	Recurrent-structured SNN	57
4.1.1.2	Formulation of Optimization Problems	59
4.1.1.3	Solve Optimization Problem by SNN	59
4.1.2	Spike Signal Processor	60
4.1.2.1	Tick-batching	60
4.1.2.2	Digital Synapse Operation	62
4.1.2.3	Synapse Operation in 3D-NOR Flash	62
4.2	Motivation	63
4.2.1	Opportunity of 3D-NOR Flash Synaptic Array for Solving Opti- mization Problems	64
4.2.1.1	Computing-dominated Case	64
4.2.1.2	Switching-dominated Case	64
4.2.2	Challenge of applying 3D-NOR Flash as Synaptic Array	65
4.2.2.1	Sparse Input Spike	66
4.2.2.2	Small Input Degree	66
4.2.2.3	Ineffective Pruning	68
4.3	Proposed Architecture-Neureka	69
4.3.1	Time Coarsening Unit	69

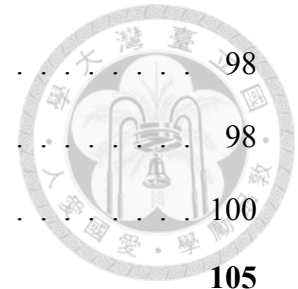


4.3.2	Reduction of Input Granularity of Synaptic Array	71
4.3.3	Skipping Polarized Neuron for Optimization Applications	73
4.4	Evaluation Setup	75
4.4.1	Latency Evaluation	75
4.4.2	Evaluation Across Diverse Applications	78
4.4.3	Implementation of Heuristic Methods	79
4.5	Evaluation Result	79
4.5.1	End to End Comparison	79
4.5.2	Effect of Skipping and Coarsening	81
4.5.3	Discussion of Neural Firing Rate Scaling	82
4.5.4	Comparison with CPU and GPU	82

Chapter 5 Efficient and Error-resilient Spiking Neural Networks through Binarization 85

5.1	Background and Motivation	85
5.1.1	Spiking Neural Networks (SNNs)	85
5.1.2	Existing challenges in SNNs	86
5.1.3	Binarized Neural Networks (BNNs)	86
5.1.4	Binarized Spiking Neural Networks (BSNNs)	87
5.2	System Model	87
5.2.1	Binarized Neural Networks (BNNs)	87
5.2.2	Spiking Neural Networks (SNNs)	89
5.3	Impacts of Binarization on SNNs	92
5.3.1	Impact of Binarization in SNNs on Membrane Capacitor	93
5.3.2	Impact of Binarization in SNNs on Latency	95
5.3.3	Impact of Binarization in SNNs on Energy	96
5.3.4	Errors by Cmem Reduction and Countermeasures	97

5.4	Evaluation	98
5.4.1	Experiment Setup	98
5.4.2	Experiment Results	100
Chapter 6	Conclusions	105
6.1	Memory Device Reliability Analysis	105
6.1.1	Reliability on Image Classification	105
6.1.2	Reliability on Solving Optimization Problems	106
6.2	Architecture Design Optimized for Solving Optimization Problems	107
6.3	Binary Spiking Neural Network	107
6.4	Future Works	108
	References	111

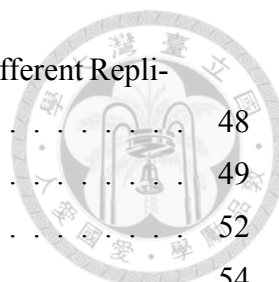






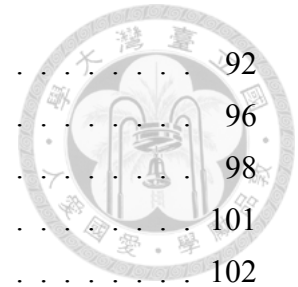
List of Figures

1.1	Overall Structure of the Dissertation	8
3.1	Implementation of Analog SNN Computation	14
3.2	Sampling of Input and Output Current	17
3.3	Definition of Constraint Satisfaction Problem (CSP)	18
3.4	The Overall Flow of SNN	19
3.5	Example of Flow for Solving CSPs	20
3.6	Annealing Machines	20
3.7	Comparison of Annealing Machines	22
3.8	Impact of Synapse Devices and Small Size of Membrane Capacitor on Classification	25
3.9	Impact of Synapse Devices and Small Size of Membrane Capacitor on Solving Optimization Problems	25
3.10	Impact of C_{mem} Capacitance	26
3.11	Overall Scope of Reliability Analysis	27
3.12	Replication of Memory Cells to Ease Computing Error	31
3.13	Implementation of the Signed Weight	31
3.14	Overview of Simulation Framework for Image Classification	34
3.15	CNN Model of VGG7	35
3.16	Current Characteristic of 2T NOR	35
3.17	Current Characteristics of FeFET	36
3.18	OFF-state Current Calibration and Case of Neglecting OFF-state Current	38
3.19	Effect of Low ON-OFF Ratio	39
3.20	Effects of ON-state Current and Variation	39
3.21	Current Replication Eases Current Variation	40
3.22	Comparison of Cycle Level Simulation and SNN Behavior Model	45
3.23	Temperature-induced Drift Curve	47
3.24	Impact of Cell Current Variation on Success Rate	47



3.25	Impact on the Requirement of Membrane Capacitance for Different Replication of Memory Cells	48
3.26	Id-Vg Curve of FeFET Devices	49
3.27	Impact of OFF-state Leakage on the Requirement of C_{mem}	52
3.28	Impact of Temperature on the Success Rate	54
3.29	ON-OFF Ratio and Robust Index for Each Memory Technology and Temperature	54
3.30	Predictability of Robust Index	55
3.31	Energy Consumption of an SNN Macro per Iteration	56
4.1	Switching Cost of a Spiking Processing Core	61
4.2	Prior Architectures of Spiking Processing Core	62
4.3	Structure of 3D-NOR Flash Synaptic Array	63
4.4	Overall Scope of Neureka	66
4.5	Impacts of Bus Contention on Digital-based Spiking Processor Cores under Different Tick-batching Sizes TW	67
4.6	Comparison of Area Efficiency	68
4.7	Computing Error from the Tick-batching Technique for Recurrent Network Structure	69
4.8	Impact of Tick-batching Technique on Solving Quality of Optimization Problems	70
4.9	Overall Architecture of Neureka	71
4.10	Design Concept of Time Coarsening Unit	72
4.11	Implementation of Time Coarsening Unit	73
4.12	Result of Coarsening Unit on 128 Input Spike Trains over 256 Cycles	74
4.13	Design Concept of Input Granularity Reduction	75
4.14	One-Dimensional Systolic Array	76
4.15	Architecture of 3D-NOR Flash Synaptic Array	77
4.16	Implementation of Runtime Soft Neuron-Pruning Algorithm	78
4.17	Processing Time Comparison amount Architectures	80
4.18	Effective of Proposed Design Throughout the Annealing Process	82
4.19	Impact of Maximal Firing Rate	83
4.20	Comparison with other Heuristic Method Processing on CPU and GPU	84
5.1	Overall Structure of Binarized SNN	88
5.2	XNOR Logic Gate for Binarized Multiplication	89
5.3	Typical AND Mode and BNN XNOR Mode	90

5.4	I_{ary} over Time	92
5.5	Description of Information Loss Metric	96
5.6	Waveform of SNN Circuit	98
5.7	Normalized Error from Information Loss	101
5.8	Effect of Sensing Frequency (SF)	102
5.9	Noise Tolerance with Sensing Frequency (SF) 2GHz	102
5.10	Comparisons of Guaranteed Response Time	103
5.11	Computing Efficiency for Each Model	103







List of Tables

3.1	Summary of Cell Characteristic for Evaluation	34
3.2	FeFET Parameters of Design Schemes	37
3.3	Constraint Satisfaction Problems	46
3.4	Parameters of Resistive Memory Technologies	46
3.5	Parameters of Transistor-based Memory Technologies	46
5.1	Energy Configurations of SNN Macro	100
5.2	Requirement of Cmem for 4-bit NN, BNN, ER-BNN	100





Denotation

BL	Bit-Lines
CIM	Computing in Memory
CNN	Convolutional Neural Network
CSP	Constraint Satisfaction Problem
CUA	Circuit under Array
DNN	Deep Neural Network
ER	Error-Resilient
ER-BNN	Error-Resilient Binary Neural Network
GCM	Gated Current Mirror
GRT	Gaurantee Response Time
HBM	High-Bandwidth Memory
MAC	Multiply-and-Accumulation
MF	Mean Field
MHL	Modified Hinge Loss

NEF	Neural Engineering Framework
NVM	Non-Volatile Memory
PTB	Parallel Time Batching
QUBO	Quadratic Unconstrained Binary Optimization
PR & EC	Remnant Polarization and Coercive Field
RRN	Recurrent Reasoning Network
SF	Sensing Frequency
SL	Source Line
SNN	Spiking Neural Network
FPAAs	Field-Programmable Analog Array
TSP	Traveling Salesman Problem
tRRD	Row-to-Row Delay
tWR	Write Recovery Time
tWTR	Write-to-Read Time
TW	Time Window
VMM	Vector Matrix Multiplication
WL	Word Lines
SPU	Spike Processing Unit



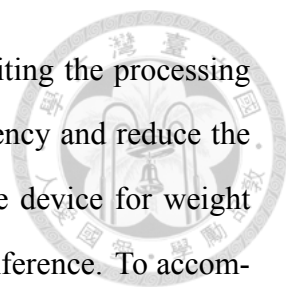


Chapter 1 Introduction

1.1 Inference of Spiking Neural Network

In the realm of artificial intelligence, the spiking neural network (SNN) emerges as a noteworthy innovation inspired by the brain's complex mechanisms. This sophisticated system operates through spiking neurons, which collaborate seamlessly, exchanging information through spike signals. This harmonious integration enables the network to perform inference tasks effectively, applying its unique capabilities to solve real-world challenges. The innovative approach to computing offers a potential method for efficient inference processing. This significant potential motivates organizations and industries to dedicate efforts to the development of algorithms and hardware design, aiming to achieve a strategic edge in meeting future technological challenges. Google has experimented with using temporally encoded spiking signals for recognition tasks to create an energy-efficient fundamental component [1]. Additionally, various organizations have unveiled their spiking neural accelerators, providing essential hardware to speed up spiking signal processing. For instance, IBM introduced TrueNorth [2], Stanford developed Neurogrid [3], and Intel launched Loihi [4].

However, the low economic efficiency and the time domain encoding [1] hinder their extensive adoption in everyday life. The inefficiency in these architectures, especially when processing spiking neural networks, stems largely from the need to store network parameters, like weights, in SRAM or scratchpad memory. Additionally, the inherently stochastic nature of spiking neural networks necessitates multiple spike signal processing

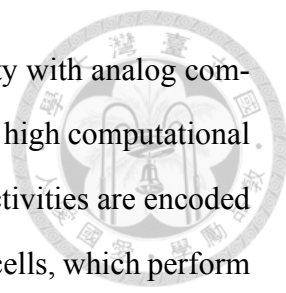


cycles to achieve accuracy competitive with traditional models, limiting the processing throughput. In response, this thesis seeks to enhance the cost-efficiency and reduce the latency of spiking neural network inference by employing a storage device for weight storage and proposing neural model enhancements for low-latency inference. To accomplish these objectives, we face three primary challenges. The non-ideal characteristics of non-volatile memory, such as current variation and leakage, pose risks to accuracy and necessitate an analysis of model reliability. The inherent long latency and large granularity of non-volatile memory present a significant challenge in maintaining processing throughput. The spiking neural model must be re-engineered to balance competitive accuracy with reduced inference latency. To address these challenges, this thesis is divided into three main components:

- A reliability analysis to understand the impact of cell nonideality on spiking neural network applications.
- The development of architecture and hardware designed to manage the challenges posed by the large granularity of cost-effective non-volatile memory devices.
- Training the model to be more compatible with spiking neural networks, aiming to achieve higher processing throughput than direct deployment of traditional neural models.

1.2 Reliability Analysis for Non-volatile Synaptic Devices

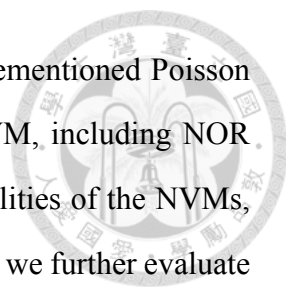
Organizations or companies have proposed a variety of specialized hardware accelerators to perform the computations of SNNs. Notable examples are BrainScaleS [5], Loihi [4], TrueNorth [2], SpiNNaker [6], and NeuralGrid [3]. While the prevalent method for buffering synaptic strength in prior works is SRAM, recent investigations have revealed the potential advantages of employing a non-volatile memory (NVM)-based synapse array for reducing cost, area, and energy consumption in comparison to SRAM, as discussed



in [7, 8]. Exceptional synergy is achieved by combining non-volatility with analog computations. NVM-based accelerators with analog computation provide high computational parallelism, increasing the chip area efficiency [7]. In SNNs, input activities are encoded as spike trains. These spikes are passed through synapses, i.e., NVM cells, which perform the analog weighting and return weighted currents. Then, the weighted currents from all synapses connected to the neuron circuit are summed by Kirchhoff's circuit law. This summed current is transmitted to the neuron circuit, charging a membrane capacitor. The neuron emits a spike when the voltage exceeds a predefined threshold voltage. However, the non-ideal nature of current from NVM cells in analog computing leads to reliability concerns. Therefore, choosing an appropriate device for synaptic operations to ensure reliable computation is crucial. Prior analytic works discussed the impact of focus on the impact of resistive memory on accuracy loss of image classification [9–11]. A comprehensive analysis of various memory devices is still not available. Moreover, the impact of NVM cells on the annealing function of SNNs remains largely unexplored. Most importantly, the selection of membrane capacitor, C_{mem} , needs to match the cell current characteristic to achieve the inference process with minimal time and energy consumption, but the requirements of C_{mem} for each memory devices are not yet discussed. Evaluating the influence of various NVM cells' nonidealities and circuit awarded evaluation on target applications, including image classification and optimization solving, is essential for designing brain-like computing systems under strict energy, space, and latency constraints.

Specifically, our contributions are as follows:

- We analyze the impact of NVM nonidealities on the cost of membrane capacitor, C_{mem} , on the accuracy of image classification and success rate of solving optimization problems. We also derive a Poisson model, which accurately describes the operation of the SNNs. Moreover, it connects the neuron circuit parameters (e.g., capacitance and currents) and the precision of the analog computations, enabling formal investigations of the nonideality effects.
- We evaluate the effect of NVM nonidealities on targeted applications in the ex-

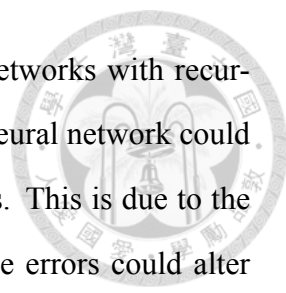


periments by controlling the circuit parameters using the aforementioned Poisson model. As the memory technologies, we use the data of NVM, including NOR Flash, FeFET, WOx ReRAM, and HfOx ReRAM. As nonidealities of the NVMs, we consider current variation, OFF-state leakage. Additionally, we further evaluate the effect of temperature-induced drift on the application of solving optimization problems.

1.3 Architecture Design for Efficient Inference

The enhancement of spike signal processing schemes has played a crucial role in the evolution of spiking neural network hardware, leading to significant advancements in emulating neural dynamics. Digital-based architectures, such as SpiNNaker [6], TrueNorth [2], Tianjic [12], and Loihi [4], have been developed to meet these requirements. On the other hand, the mixed-signal architecture, represented by Neurogrid [3], has also been introduced, aiming for maximized energy efficiency. Despite various proposals for spiking neural network accelerators, these architectures struggle to handle spiking neural networks that exceed the capacity of spiking neuron hardware. This is especially true when considering limited memory capacity and neuron circuit constraints. Recent designs for spiking neural networks, such as SpinalFlow [13] and Parallel Time Batching (PTB) [14], are engineered as spiking signal processors. They are primarily focused on handling spiking neural models buffered in DRAM. Nonetheless, the inevitable core switching required to process different sets of neurons introduces a significant delay in loading and storing neuron states and weights. This results in a substantial performance drop [14].

To address the critical challenge of switching overhead, the tick-batching technique, as proposed in [13–15], plays a pivotal role. This technique aims to capitalize on the reuse of loaded weights and neuron states to amortize latency when switching between neurons during processing. The tick-batching technique can be seamlessly applied to feedforward networks without compromising accuracy due to the independence between inputs and



outputs, as elaborated in [16]. However, it is not as effective for networks with recurrent structures. Applying the tick-batching technique to a recurrent neural network could lead to computing errors from unsynchronized input neural activities. This is due to the input and output dependency inherent in recurrent structures. These errors could alter the converged state, negatively impacting optimization problem-solving in diverse fields. Such fields include planning efficient routes for vehicles [17–23] in delivery systems [24–26], making decisions about product lines [27–32], analyzing genomics data [33–40], and managing investment portfolios [41–44], among others.

To ease the switching overhead with minor costs in neuron synchronization, processing at the place where data is located eases the performance drop from weight movement. Previous studies, such as RESPARC [45], Nebula [8], and NFCA [46], have demonstrated that non-volatile synaptic arrays can mimic neural synapse behavior in compliance with Kirchhoff’s current laws. Compared to SRAM, these non-volatile memory devices offer a lower bit cost for storing weights, while the analog synaptic device provides competitive computing throughput to digital alternatives. However, these 2D structures have limited capacity under area constraints. The solution to this problem is stacking these synapse arrays into 3D structures. The 3D NOR Flash memory is a novel 3D synaptic device providing lower bit cost and higher bit density than its 2D counterparts, with competitive processing throughput. Furthermore, 3D-NOR Flash inherits a substantial ON-OFF ratio, negligible leak-current, and small noise-ratio from NOR-Flash cells, making it a reliable device for computing purposes [47, 48]. These improvements reduce computational errors caused by cell leakage currents. Nonetheless, the extended setup time of 3D-NOR Flash requires a larger input granularity to maintain processing throughput. A spiking network characterized by neurons with limited input connections and high spatial sparsity in input spikes results in reduced throughput.

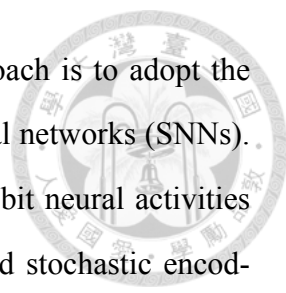
Conclusively, the main challenge lies in effectively harnessing the weight-moving free, high-bit density, and high ON-OFF ratio of 3D-NOR Flash technology while ensuring optimal utilization of its processing throughput. To mitigate the throughput reduc-

tion in 3D-NOR Flash synaptic arrays, the presented architecture employs a collaborative hardware-software design strategy. This method is tailored to augment processing efficiency while simultaneously diminishing negative impacts on the process of solving optimization problems. Our primary contributions to this research are as follows:

- We are the first spiking signal processor that optimizes for solving optimization problems. Moreover, the proposed architecture is synergy with 3D NOR-Flash memory, outperforms throughput of synapse operation, and eliminates the requirement of weight movement.
- We introduce three innovative designs to enhance spiking processing throughput using 3D NOR-Flash memory as a synaptic array. These include (a) a novel finite-disturbance input spike coarsening unit that reduces synapse operations while maintaining neuron firing time accuracy, (b) a synaptic array architecture designed to reduce input granularity, achieved through a 1D-systolic array derived from the circuit under array (CuA) technique, and (c) a run-time pruning technique that skips polarized neurons, significantly decreasing processing time.
- We benchmark our design against prior spiking signal processing units on optimization applications. The results show that our proposed design provides 3.1x, 1.8x, and 1.2x gains compared to the state-of-the-art SNN spiking processors for solving SUDOKU, MAX-CUT, and LASSO problems. Our proposed Neureka also achieves 6.6x, 1.8x, and 2.96x faster than other heuristic methods for processing on high-end CPUs and GPUs.

1.4 Algorithmic Enhancement for Image Recognition

The rise of deep learning has significantly improved classification accuracy across various fields like image recognition [49, 50], speech processing [51], and object detection [52, 53]. To process these complex models efficiently, numerous accelerators have



been developed for energy-efficient inference. An alternative approach is to adopt the brain's signal processing methods, specifically through spiking neural networks (SNNs). Existing research has explored various methods for encoding multi-bit neural activities in the time domain, including delay [13, 54, 55], rank [56, 57], and stochastic encoding [58, 59]. Among these, stochastic encoding aligns well with deep neural models. In this scheme, the weights and connections of the SNN can be derived from a pre-trained neural model, while input activities are translated into spike rates. This stochastic computing approach can maintain classification accuracy through multiple cycles of binary input operations.

However, stochastic encoding methods typically require numerous spike operations, which can slow down inference and increase energy consumption. To address this, network models with binary inputs have been developed to enable single-cycle computation. As the encoded probability is either one or zero, encoding can be completed in just one cycle. Additionally, binary neural networks (BNNs) can be further optimized through training schemes and network structure searches to reduce computational requirements without compromising accuracy [60, 61]. They can also be made error-resilient [62, 63], enhancing the noise margin for SNNs and improving area and energy efficiency [48]. This research is pioneering in leveraging the advantages of BNNs for SNNs, achieving significant improvements in area, latency, and power efficiency without sacrificing accuracy. Our novel contributions in this section are:

- We reveal the impact of deploying error-resilient BNNs to analog-based SNN accelerators on reducing the membrane capacitor size C_{mem} .
- We demonstrate how the resiliency of BNNs, especially when trained in the presence of bit errors, increases the robustness of SNN computation and enables significant C_{mem} reduction with marginal inference accuracy loss.
- We show that error-resilient BSNNs achieve 50% reduction in C_{mem} size, two magnitudes of improvement in latency, and 57% in energy compared to the baseline

(4-bit) SNN implementation under minimal accuracy cost of 3% same as BNN.



1.5 Organization of the Dissertation

This thesis aims to build a general-purpose probabilistic inference machine that can predict the output distribution from the sampling process. Three aspects are addressed to improve the processing speed to achieve this goal. Firstly, this thesis provides a reliability analysis while utilizing non-volatile memory to accelerate the spiking neural network. Unlike prior analysis, this thesis further discusses the relation between cell current characteristics and neuron circuit requirement. Secondly, the architectural design is proposed to speed up the evolving spiking neuron purpose. This thesis further designs the architecture for the annealing purpose and addresses the issues while utilizing computing-in-memory to accelerate synaptic operation. Thirdly, this thesis proposes a neural model conversion scheme for image classification to enable the probabilistic inference machine, dramatically speeding up image recognition tasks. The overall structure of the dissertation is as Fig. 1.1.

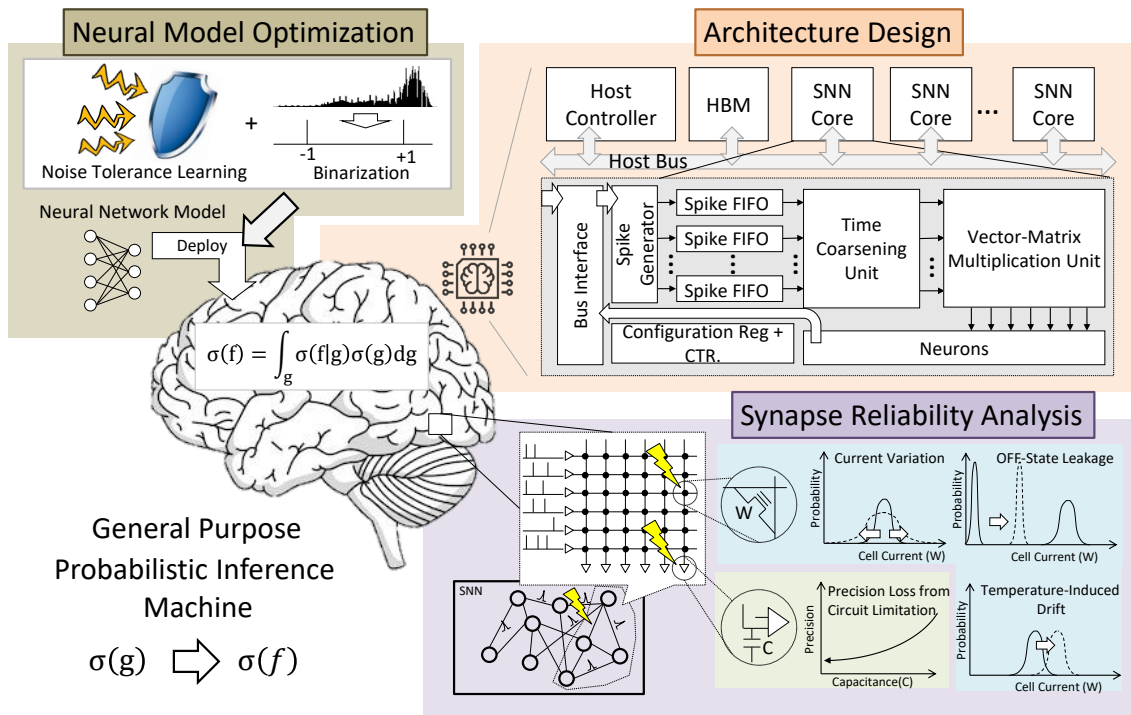


Figure 1.1: Overall Structure of the Dissertation



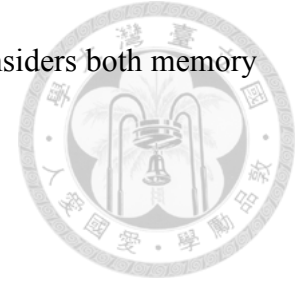
Chapter 2 Related Work

2.1 Reliability Analysis of Processing in Memory

The exploration of spiking neural networks (SNN) in recognition applications has addressed the reliability of stochastic-based computing schemes. Prior works have constructed error models for neurons, synapses, and routers for reliability analysis [64, 65]. Synapse errors, attributed to manufacturing defects or voltage-induced approximation errors, are analyzed through the raw bit error rate from bit-flipping and stuck-at-faults, impacting image classification tasks [66, 67]. Furthermore, the errors of spiking neurons, including dead neuron fault, stuck signal fault, and delay fault, have been explored for their effect on reliability [66]. Particularly, the work in [67] delves into a transistor-level model, providing a detailed view of how single transistor faults impact neuron behavior and overall image classification applications. The reliability analysis also extends to router faults and network parameter errors [64].

Despite these developments, it's crucial to note that all the above error models are built from the digital domain, which differs from the analog domain. An alternate approach for implementing synapse operation, i.e., the inner product, processes in memory by weighting input spikes and accumulating them to the output current, leading to direct accumulation of current noise. To assess reliability in this context, analog-based synapse error models have been introduced to evaluate the impact of current noise on applications like image classification [9, 68] and robot controlling [69]. However, as memory devices and neuron circuits are interconnected, the reliability of neuron circuit behavior is

inevitably influenced by the memory cell. This thesis, therefore, considers both memory cells and neuron circuits for a comprehensive reliability analysis.



2.2 Spiking Neural Network Architecture

Several organizations have proposed the spiking neural network (SNN) accelerator aiming to achieve high recognition capability and efficiency akin to the human brain. The University of Manchester has been at the forefront with its development of SpiN-Naker [6], a system engineered with an intricate array of microprocessors. This platform is particularly noted for its method of integrating microprocessors, which involves the transmission of inter-chip spikes through synchronized messages, facilitating complex neural network simulations. Building on these technological strides, Stanford University's Neurogrid [3] represents a significant advancement. This architecture operates on a mix-signal basis, specifically designed to emulate continuous-time neural dynamics, offering deeper insights into brain-like processes. In a concurrent technological advancement, BrainScaleS [70] has unveiled its wafer-scale analog spiking neural network hardware, integrating a fault-tolerance design. This design ensures that the hardware remains functional even when a few transmitters are dysfunctional. Such an innovation significantly reduces the necessity for inter-chip communication, thereby enhancing overall efficiency. Additionally, this approach effectively mitigates the issue of yield drop, which is commonly associated with wafer-scale chip production, ensuring more reliable and robust performance of the hardware. IBM's contribution to this field comes with the development of TrueNorth [2], which consists of synapse crossbars. This design innovatively transforms spike waveforms into digital pulses and is a complete application-specific integrated circuit, focusing on reducing power consumption and achieving a compact design. Complementing these advancements, Intel has unveiled Loihi [4], a digital-based spiking accelerator. Loihi is unique in its integration of brain-like learning functions, blending digital computing with neural network emulation. Tsinghua University's Tianjic [12] is designed to be compatible with both spiking neural networks and artificial neural networks, offering

a versatile approach to neuromorphic computing. Together, these developments mark significant milestones in neural network hardware, each contributing uniquely toward more efficient and accurate emulation of neural processes. However, these works cannot economically evolve the spiking neural model due to the significant SRAM or scratchpad memory required for weight storage. To address this challenge, we employ innovative, non-volatile computational memory. This approach accelerates synaptic operations, reducing the storage bit cost and increasing processing throughput per area.

2.3 Binarized Spiking Neural Network

In [71], SNNs are acquired by training BNNs with the methods proposed by Hubara et al. [72]. After converting the BNNs to SNNs, simple design and run-time explorations are employed to achieve inference latency improvements. In [73], a method for training SNNs with binarized weights is proposed without conversion from BNN to SNN. The role of the membrane capacitor in analog-based implementations of SNNs has been identified in [48]. The study focuses on how the non-ideal characteristics of various memory technologies affect the size requirement of the membrane capacitor. In [46], the authors evaluate SNNs on NOR flash computing array under input noise, relying on the inherent noise resiliency of NNs to sustain high accuracy. The capacitor size is set to a constant value for the LeNET-5 NN model. The study in [74] surveys the resiliency properties of SNNs trained with various state-of-the-art algorithms and proposes an approach to train SNNs for resiliency. Their main focus is on different types of synapses or neuron failures. The two studies in [75, 76] focus on fault analysis in hardware-implemented SNNs, assuming similar types of failures.

The above studies focus on the reliability issues that emerge from process variation, soft errors, and neuron and synapse faults. Methods for reducing membrane capacitor size have not been explored, although membrane capacitor size constitutes one of the major bottlenecks in analog SNNs, i.e., it determines energy, latency, area, and accuracy. To

the best of our knowledge, we are the first to investigate that. In this thesis, we evaluate the impact of deploying error-resilient BNNs on the analog implementation of SNN, and formally show that the membrane capacitor size and latency can be reduced significantly compared to multi-bit models for BNNs. We then conduct extensive experiments to support this.



Chapter 3

Impact of Non-volatile Memory Devices on Spiking Neural Network

3.1 Background and Motivation

3.1.1 Probabilistic Inference with Spiking Neurons

Humans have a remarkable capability to predict future events through incomplete or unclear stimuli from the environment. Prior research has indicated that the human brain's cognitive and reasoning functions are akin to probabilistic inference processes [77–80]. The general form of probabilistic inference is expressed as Eq. 3.1.

$$\sigma(f) = \int_g \sigma(f|g)\sigma(g)dg \quad (3.1)$$

The equation represents the marginal distribution of the predicted state, f , computed by the given distribution of the observed state vector, g . The integral of the $\int_g \sigma(f|g)\sigma(g)dg$ formulates the process of the probabilistic inference, which utilizes the uncertain state of g to predict the state distribution of f . For the recognition process, the g is the uncertain sensory input, and the f is the distribution of the representative vector. On the other hand, probabilistic inference can also apply to solving optimization problems. If the f and g are both in sequential relation, i.e., $g[t+1] = f[t]$, the discrete form of the Eq. 3.1 is also known

as a Markov chain. The eigenvector of the Markov chain's transition matrix represents the converged distribution. By properly designing the transition matrix, the state with the highest probability is considered the solution to the optimization problem. However, the computing complexity to calculate the margin distribution is $O(2^{N+M})$ where N and M are dimensions of g and f, which is impractical for finite-time inference.

The Spiking Neural Network (SNN) embodies a brain-inspired methodology that processes probabilistic inference via sampling, substantially reducing computational complexity to $O(MN)$. Within this brain-mimicking framework, spiking neurons engage through spike transmission and modulate their firing rates by weighting and integrating incoming spikes. This stochastic computing approach allows SNNs to balance computational precision with processing time effectively for image classification. The dynamic interaction consistently drives the spiking neurons to evolve towards a one-hot vector closely related to the input state for classification tasks and toward a state representing the optimization-approximated solution for combinatorial problems.

3.1.2 SNN Model and IF Circuit

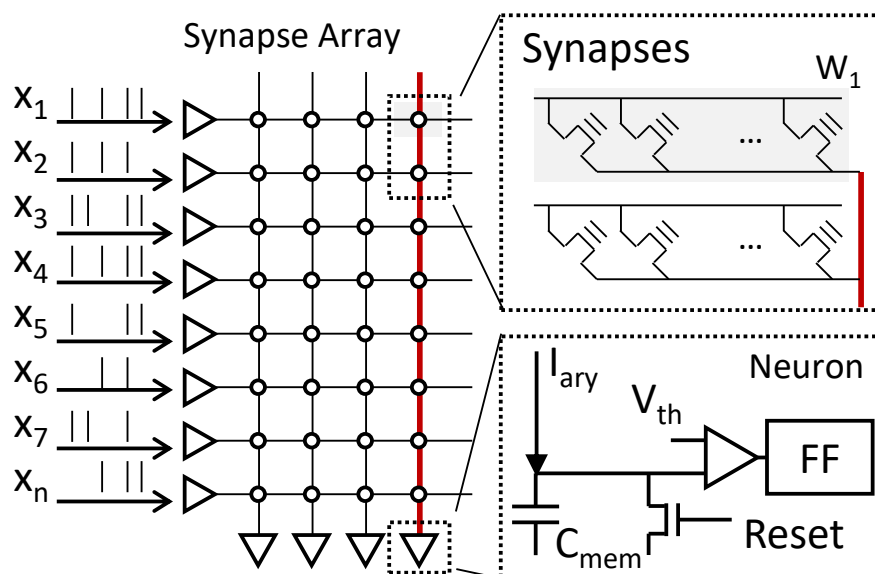


Figure 3.1: Implementation of Analog SNN Computation. We utilize a memory array to implement synapses and provide input spikes to the horizontal bit-line. Summed current is transmitted to integrated and fire (IF) neurons.

The spiking neural model used in this thesis is a self-sustained system that does not rely on external inputs to drive its states. It undergoes iterative updates of neural activities, allowing the model to evolve its state over time. The overall flow of the spiking neural model is illustrated in Fig. 3.4. In each iteration, the neural activities serve as inputs to the SNN macro, and the output of the macro is the firing time, which is subsequently converted into a gradient. This gradient is then used to update the neural activities for the next iteration. The SNN macro follows the classical current-based integrated-and-fire model, as described in references [4, 81]. The input spike is generated through a sampling process defined by Eq. 3.2. This process involves Bernoulli sampling, where the probability of generating an input spike is proportional to the input activity u_i relative to the maximum possible input value $MAX_i\{u_i\}$, denoted as Z . These input spikes generate currents that charge a membrane capacitor in an output neuron. Once the accumulated charges surpass a threshold value, the output neuron fires.

The dynamics of the spiking neural network system are governed by the differential equations presented in Eq. 3.3 to Eq. 3.5.

$$P(x_i = 1 | t) = \frac{u_i}{Z}, \quad (3.2)$$

$$v_{mem}[t + 1] = v_{mem}[t] + \gamma \sum_i w_i x_{qi}[t] - V_{th} \delta[t - T_{fire}], \quad (3.3)$$

$$T_{fire} = t + 1 \text{ if } v_{mem}[t] + \gamma \sum_i w_i x_{qi}[t] \geq V_{th}, \quad (3.4)$$

$$u_i[s + 1] = u_i[s] + \hat{\mathbb{E}}_{t \in \{0, \tau-1\}} \left[\sum_i w_i x_{qi}[\tau s + t] \right] \quad (3.5)$$

$$\hat{\mathbb{E}} \left[\sum x_{qi} w_i \right] = \sum \hat{\mathbb{E}}[x_{qi}] w_i = \frac{C_{mem} V_{th}}{T_{fire}} I_{cell} \quad (3.6)$$

The SNN system evolves the state variable of membrane voltage, V_{mem} , and neural activities, u_i . The fire rate of a neuron is proportional to u_i . The γ is a factor that converts the numerical inner product result into a physical variable, i.e., $\frac{I_{cell}T_{width}}{C_{mem}}$.

The evolution of V_{mem} in Eq. 3.3 is realized by an integrate-and-fire circuit with a synapse array as shown in Fig. 3.1. V_{mem} is the voltage across the membrane capacitor C_{mem} . The input spikes charge C_{mem} with the voltage change ΔV_{mem} across C_{mem} , $\Delta V_{mem} = \frac{T_{width}}{C_{mem}} \sum_i w_i x_i [t] I_{cell}$, in which the w , weights, is encoded by the number of cell in ON-state. After the neuron fires at the time at which $V_{mem} \geq V_{th}$, a reset signal triggers the discharge path, resetting V_{mem} to zero.

The neural activities u_i (defined in Eq. 3.2, which is proportional to the input spike probability) is also updated per time window of τ as Eq. 3.5. The value s is the number of time windows: $s = \lceil \frac{t}{\tau} \rceil$. Because of the random process to generate input spikes, the updated value of neural activities has deviations from the expected value of $\sum_i w_i x_i$ [82], i.e., $\hat{\mathbb{E}}[\sum_i w_i x_i] = \mathbb{E}[\sum_i w_i x_i] + \xi[t]$, where $\hat{\mathbb{E}}$ is the sample mean from the beginning of the time window until the neuron returns a spike signal.

To prevent the ADC from sensing the current at each cycle, the $\hat{\mathbb{E}}[\sum_i w_i x_i]$ is calculated when the neuron produces an output spike, i.e., at time T_{fire} , see Eq. 3.6. A reciprocal function [83] is required to convert the spike time of a firing event to frequency. The reciprocal can be used for multiple neurons without a bottleneck in throughput. Multiple neurons with the same firing time can share the reciprocal calculation. Moreover, the discrete and finite firing timing enables a lookup table to alternate the reciprocal unit.

3.1.3 Precision Modulation for Image Recognition

3.1.3.1 Neural Network with Quantized Input

The neural network models, including multiple-layer perception, convolutional neural networks, and transformers, are all constructed by layers of Perceptron. Each per-

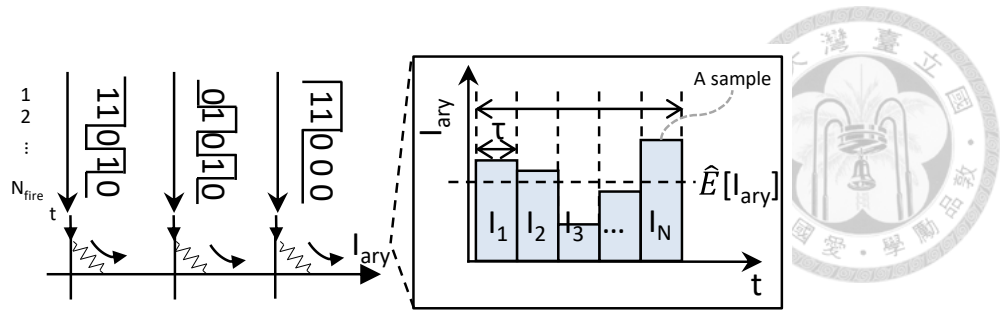


Figure 3.2: Sampling of Input and Output Current

ceptron processes the inner product of the input and weight vectors' inner product. The product results are converted by the nonlinear function and transmitted to the next layer as inputs. Recent network models are designed to be hardware-friendly without sacrificing classification accuracy dramatically. One widely adopted method quantizes the model into low-bit precision inputs that simplify the complexity multiplication circuits [60, 84–90]. The low-precision neural model also increases noise tolerance because the inner product result with small disturbance can be quantized into the same levels, performing as a noise filter. The noise-tolerable margin enables the spiking neural network, the stochastic computing-like process scheme, to achieve a higher inference energy efficiency and throughput from the noise margin.

3.1.3.2 Precision Modulation

Inputs of spiking neural networks are considered random variables. Thus, the dynamic of the spiking neural network is a random process. The relation among computing precision, circuit parameters, and array current is theoretically discussed based on the stochastic system.

In Figure 3.2, the binary inputs are resampled at each time step with a length of τ . The output current also changes at each time step. The resampling process is continued until the neural fire. Once the neuron fires, Equation (5) can be rewritten as Equation (12) in which the t_{fire} is the required duration to generate an output spike. The expected value of the sampled output current is defined as Equation (13), where E is called the observed array current. Once T is equal to t_{fire} , the expected value can be formulated as Equation(14)

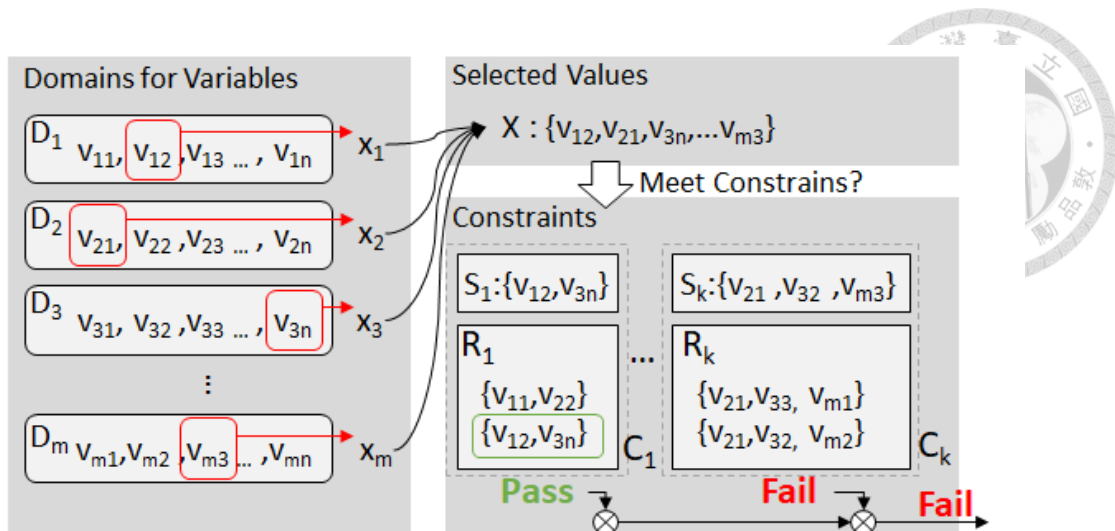


Figure 3.3: Definition of Constraint Satisfaction Problem (CSP). Each variable is a value selected from a value domain. The selected values meet a constraint if the selected value set is in the rule set. The constraint-fitted solution is a selection of values meeting all constraints.

by combining Equation (12) and (13). The expected current of the stochastic system is as Equation (15). Based on the above condition, the law of large numbers is formulated as Equation (16) and Equation (17). The N_{fire} in Equation (16) is a required time step number to fire the neuron where τN_{fire} is equal to t_{fire} . The law of large numbers shows that the observed array current will converge to the system's expected current if N_{fire} is increased. In other words, the precision of the observed current is correlated to the number of N_{fire} . Equation (18) shows the relation between circuit parameters, computing precision, and observed array current. N_{fire} indexes the precision of sum-of-product and can be modulated by the factor of

3.1.4 Stochastic Annealing for Optimization Solver

3.1.4.1 Constraint Satisfaction Problem

The constraint satisfaction problem (CSP) is to find out the value of variables that meet a set of objectives. All combinatorial optimization problems can be represented as CSP [91]. A CSP is described by a tuple $T(\mathbf{X}_{\text{csp}}, \mathbf{D}, \mathbf{C})$. $\mathbf{X}_{\text{csp}} = \{x_{c1}, x_{c2}, x_{c3}, \dots, x_{cn}\}$ is a set of variables, and $\mathbf{D} = \{D_1, D_2, D_3, \dots, D_n\}$ is the corresponding set of values for each

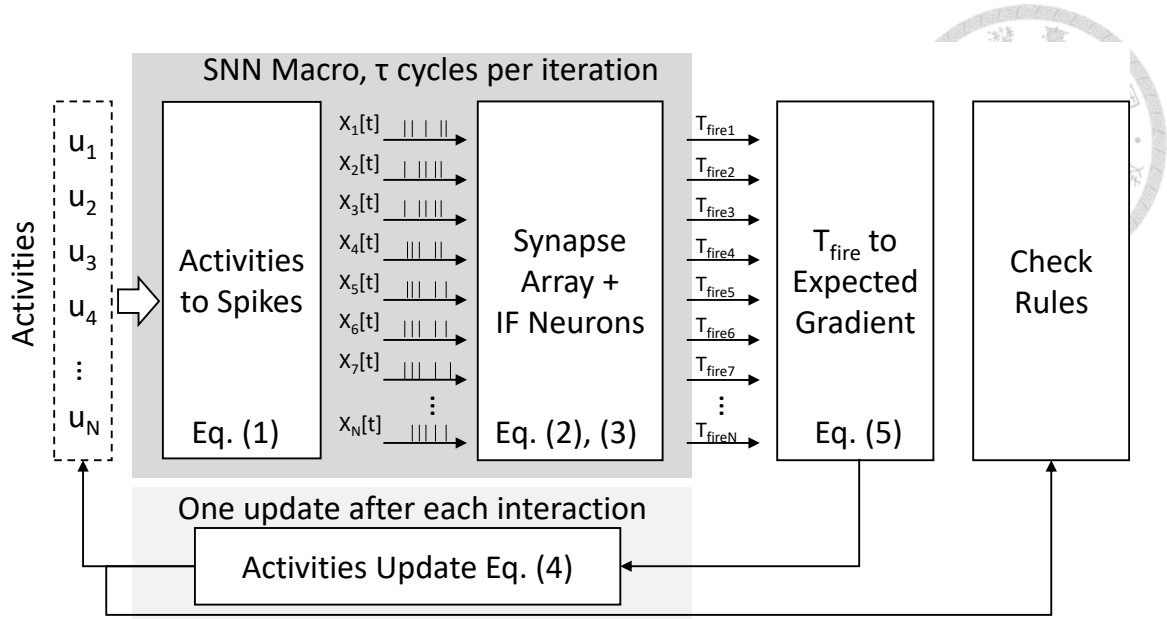


Figure 3.4: The Overall Flow of SNN. The internal variables of neural activities, \mathbf{U} , are the state of the spiking neurons. The SNN macro received the neural activities and took τ cycles to generate each output neuron's firing time, T_{fire} . We set $\tau = \text{MAX}_i \{u_i\} \frac{C_{mem} V_{th}}{T_{width} I_{cell}}$ of which the corresponding gradient is one over maximal neural activities, i.e., $\frac{1}{\text{MAX}_i \{u_i\}}$. The neurons that do not get to fire within τ cycles are considered not firing, i.e., $T_{fire} = \infty$. The model then converts the T_{fire} to the expected gradients, i.e., $\hat{\mathbb{E}}[\sum x_{qi} w_i]$, and updates the neural activities for the next iteration. In the meantime, the updated state is examined to determine whether the corresponding solution meets all the rules.

variable as shown in the Fig. 3.3. We have a value selection function f selecting a value from D_i for variable x_{ci} , i.e., $f : f(x_{ci}) \rightarrow v_i, v_i \in D_i$. $\mathbf{C} = \{C_1, C_2, C_3, \dots, C_m\}$ is a set of constrains. $C_i \in \mathbf{C}$ is a pair (S_i, R_i) where $\mathbf{S} \subseteq \mathbf{X}_{csp}$ and R_i is a set of allowed answer to check S_i , i.e., checking $\{f(z) | z \in S_i\} \in R_i$. The solution of a CSP is a selection of values from D_i for each value of x_i , and the selected values meet all the constraints of \mathbf{C} .

For example, in Sudoku, we have 81 variables in \mathbf{X}_c since we have a 9x9 matrix. Each valuable of \mathbf{D} is in $\{N | N \leq 9\}$, where $N \in \mathbb{N}$. The constraints are: Each N occurs only once in a row of 9 elements, once in a column of 9 elements, and once in every 3x3 sub-block. To solve the problem, the empty spaces in the grid need to be filled such that the constraints are satisfied.

The search space grows exponentially as the number of variables increases, which makes finding solutions a difficult challenge or outright infeasible. To get a solution to CSPs under affordable time cost, SNNs provide a brain-inspired methodology.

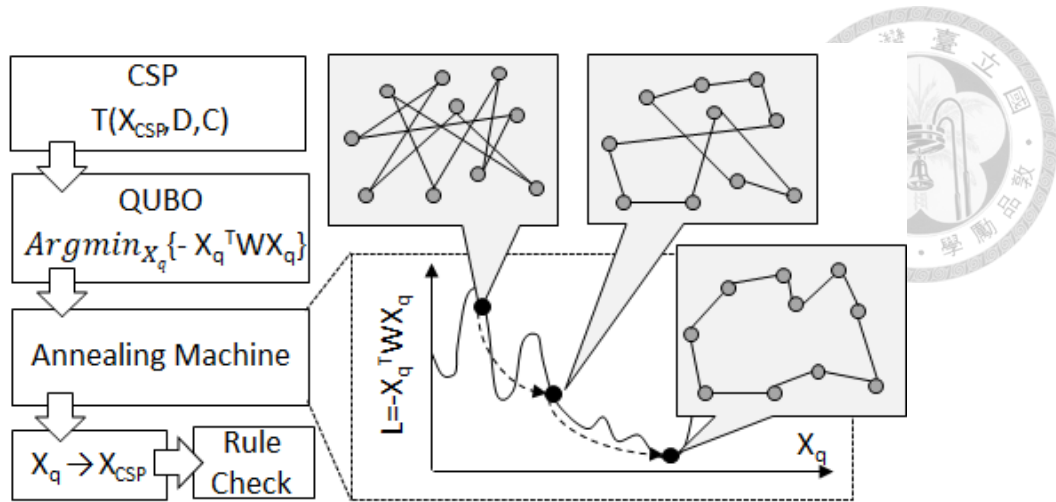


Figure 3.5: Example of Flow for Solving CSPs. The problem is converted into quadratic unconstrained binary optimization (QUBO) form and deployed to the SNN annealer. By evolving the state of SNN, the state vector \mathbf{X}_q converges to the state with less loss. The converged state (\mathbf{X}_q^*) is converted back to the value vector (\mathbf{X}_{CSP}), and it is checked whether all rules are met.

3.1.4.2 SNN Annealing Machine

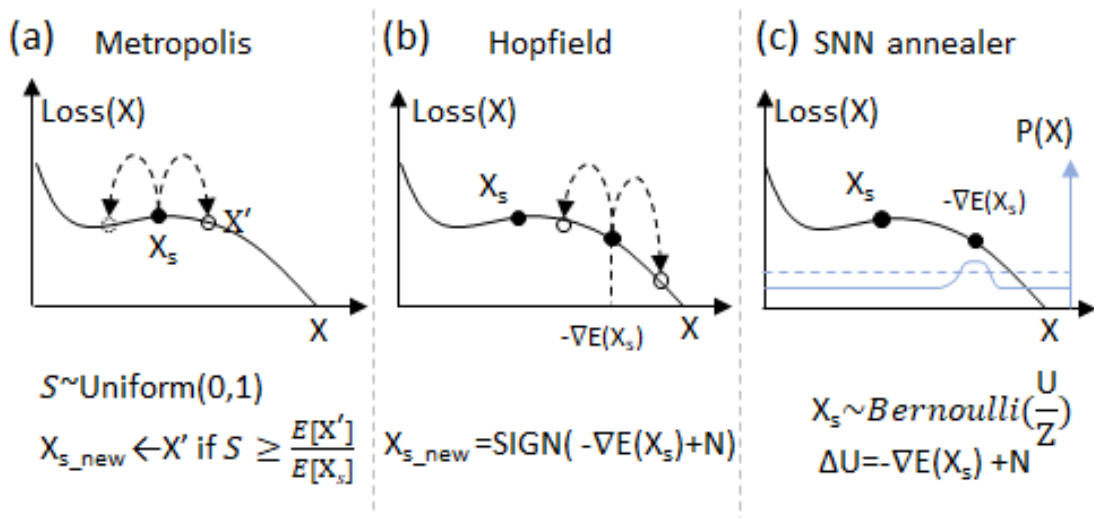


Figure 3.6: Annealing Machines. (a) Metropolis: Check the loss of re-sampled states. (b) Hopfield: Sample a new state from a gradient with noise. (c) SNN annealer: Converge the probability of the SNN state to the state with a smaller loss.

The overall flow to solve the CSP is shown in Fig. 3.5. The CSP is a deterministic problem that checks whether the provided answer of \mathbf{X}_{csp} meets all the constraints. Yet, this format does not fit into the annealing machine.

To solve combinational optimization problems by annealing machines, the CSP is converted into the quadratic unconstrained binary optimization (QUBO) form [92, 93].

The QUBO describes the problem of finding an $\mathbf{X}_q \in \mathbb{B}^n$ for a given \mathbf{W} such that the loss function $-\mathbf{X}_q^T \mathbf{W} \mathbf{X}_q$ is minimized, i.e., find $\mathbf{X}_q^* = \arg \min_{\mathbf{X}_q} -\mathbf{X}_q^T \mathbf{W} \mathbf{X}_q$. To apply QUBO, the \mathbf{X}_{csp} and \mathbf{C} in CSP need to be converted into \mathbf{X}_q and \mathbf{W} . Each element of x_q is a binary number that represents a selection of a possible value of \mathbf{X}_{csp} , i.e., $x_q[i, j] = 1 \rightarrow x_{csp}[i] = v_j$. The \mathbf{W} is directly derived from \mathbf{C} , which can be achieved by the procedure described in [92].

The time needed to ensure finding the \mathbf{X}_q^* grows exponentially as the number of variables increases. To solve this issue, several annealing algorithms are proposed to get acceptable solutions in an affordable search time. These annealing algorithms iteratively switch the state of \mathbf{X}_q to approach the state with the lowest loss, considered as the solution, as in Fig. 3.6. One of the classic annealing algorithms is Metropolis [94, 95], which switches the current state to a sampled state by the probability of their loss ratio. However, the sampling process is inefficient because the state switches to a lower loss state only when the state is sampled. Recent works utilize the Hopfield network to accelerate the annealing process [96–99]. Due to the gradient of the loss function, the new state tends to have a smaller loss, which anneals more efficiently than Metropolis sampling. However, due to the gradient-based minimization, the samples of new states are close to old states, which not only limits the search space but also leads to local minima. As a result, the solution quality is highly related to initial states [99, 100].

To overcome the issue of Metropolis sampling and Hopfield networks, the brain inspired annealing machines realized by SNNs [101–103] offer higher probability to find rule-fitted solutions. The SNN-based annealing machine is capable of finding solutions with a smaller loss compared to Metropolis sampling and Hopfield networks, as shown in Fig. 3.7. Different from prior annealing machines, the SNN annealing machine does not iteratively switch the state vector of \mathbf{X}_q . Instead, the SNN annealing machine considers \mathbf{X}_q as the distribution and aims to converge the distribution to the state vector with the smallest loss, i.e., $P(\mathbf{X}_q = \arg \min_{\mathbf{X}_q} \{-\mathbf{X}_q^T \mathbf{W} \mathbf{X}_q\}) = 1$. To achieve this, we define the tendency matrix of $\mathbf{A} = (\mathbf{U}_1, \mathbf{U}_2, \dots, \mathbf{U}_n)$, where \mathbf{U}_i represents the selected potential of

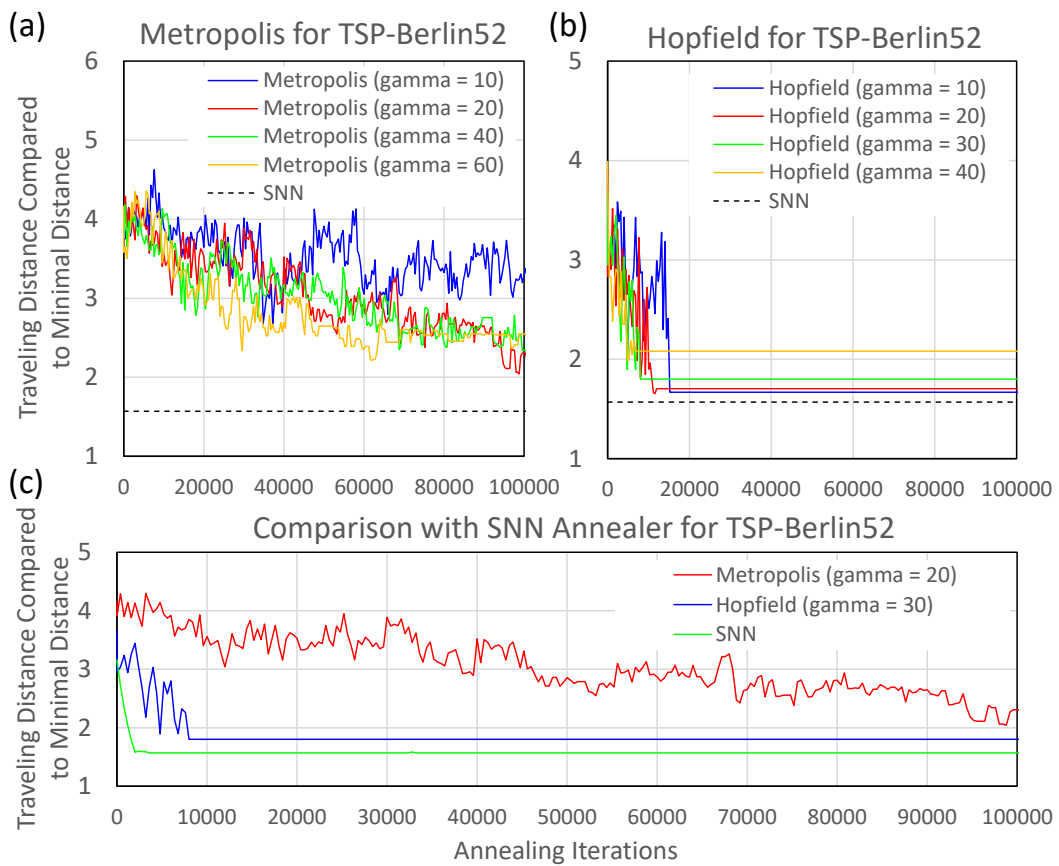


Figure 3.7: Comparison of Annealing Machines. We optimize the annealing speed for Metropolis sampling and Hopfield networks. Further, we compare them with the SNN annealer.

each variable for a x_{csp} . The \mathbf{U}_i is a vector with the same length as the number of possible values of x_{csp} . We further define \mathbf{H}_i as a binary vector of which each element represents the selection of a value of x_{csp} . The concatenation of all individual \mathbf{H}_i forms a state vector of \mathbf{X}_q . During the annealing process, H_i is a sampled vector following the distribution of $\frac{U_i}{Z}$. After the annealing process, \mathbf{H}_i is a one-hot vector of U_i , which has corresponding elements with maximal potential set to 1 and others to 0.

The procedure for updating the potential is described in Eq. 3.9.

$$\nabla \mathcal{L}(\mathbf{X}) = \left\langle \frac{\partial \mathcal{L}}{\partial \mathbf{x}_{q1}}, \frac{\partial \mathcal{L}}{\partial \mathbf{x}_{q2}}, \dots, \frac{\partial \mathcal{L}}{\partial \mathbf{x}_{qn}} \right\rangle \quad (3.7)$$

$$\frac{\partial \mathcal{L}}{\partial x_{qk}} = - \sum_{i \neq k} x_{qi} (w_{ik} + w_{ki}) - 2 \sum x_{qk} w_{kk} \quad (3.8)$$

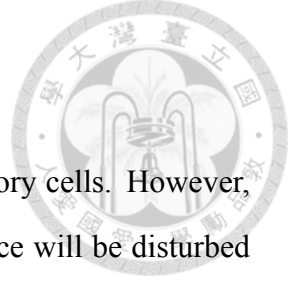
$$\Delta b f U = \left\langle \sum x_{qj} w_{1j}, \sum x_{qj} w_{2j}, \dots, \sum x_{qj} w_{nj} \right\rangle \quad (3.9)$$

The loss function of the QUBO is $\mathcal{L}(\mathbf{X}) = -\mathbf{X}_q^T \mathbf{W} \mathbf{X}_q$.

Due to the symmetry of the weight matrix in QUBO, the partial derivative of Eq. 3.8 is $\frac{\partial \mathcal{L}}{\partial x_{qk}} = -2 \sum x_{qi} w_{ik}$. The negative gradient is the vector pointing to the state with a smaller loss. With the goal of converging the state distribution to the state with the smallest loss, we update the state by $\Delta U = -\nabla \mathcal{L}(\mathbf{X})$ as shown in Eq. 3.9.

3.1.5 Unreliability Issues in SNNs

To leverage the high parallelism from analog computing [7] and to minimize the ADC process [8], synapse operation on NVM-memory crossbar and integrate and fire in the analog domain is utilized. However, in analog computing, both non-ideal memory cell current and circuit constraint of neuron circuits induce errors in applying SNN to classification and solving optimization problems.



3.1.5.1 Impact of Memory Cells

In this thesis, the \mathbf{W} is represented by the conductance of memory cells. However, when we program the \mathbf{W} into memory cells, the practical conductance will be disturbed by cell current variation, OFF-state leak, and temperature-induced drift. The disturbance causes the programmed \mathbf{W} to be not the same as the expected \mathbf{W} .

In the application of classification, the noise added to the weights leads to the change of inner product result \mathbf{WX} . The example is in Fig. 3.8(a). The error injected on the weight changes the predicted item with the maximal inner product result that causes the prediction error.

The noise injected into the weights also negatively impacts the quality of solving optimization problems. To solve CSP, we convert the constraints C into a \mathbf{W} matrix [92], and evolve the state of SNNs to find \mathbf{X}_q such that the loss $-\mathbf{X}_q^T \mathbf{W} \mathbf{X}_q$ is minimized. Once the \mathbf{W} is disturbed, the energy function is changed as the blue curve in Fig. 3.9(a). As a result, the state solved from the disturbed \mathbf{W} is not the same as the expected state. The change will cause the solution not to meet the constraints of CSPs.

3.1.5.2 Impact of Circuit Constraints

The precision of the multiply-and-accumulation (MAC) is reduced by the restricted size of C_{mem} as shown in Fig. 3.10 while processing the inner product through a spiking neural computing scheme, which the MAC result is proportional to $\frac{1}{T_{fire}}$ as Eq. 3.6. Once the capacitance of C_{mem} is too small, the firing signal of the output neuron is easily mis-triggered only by a few random spikes. As a result, the firing time exhibits significant variation, leading to low MAC precision, and the firing time may be shorter than the sensing time resolution. We denote such noises from the insufficient size of capacitance as $Noise_{C_{mem}}$.

In the application of classification, the limited size of C_{mem} induces the $Noise_{C_{mem}}$

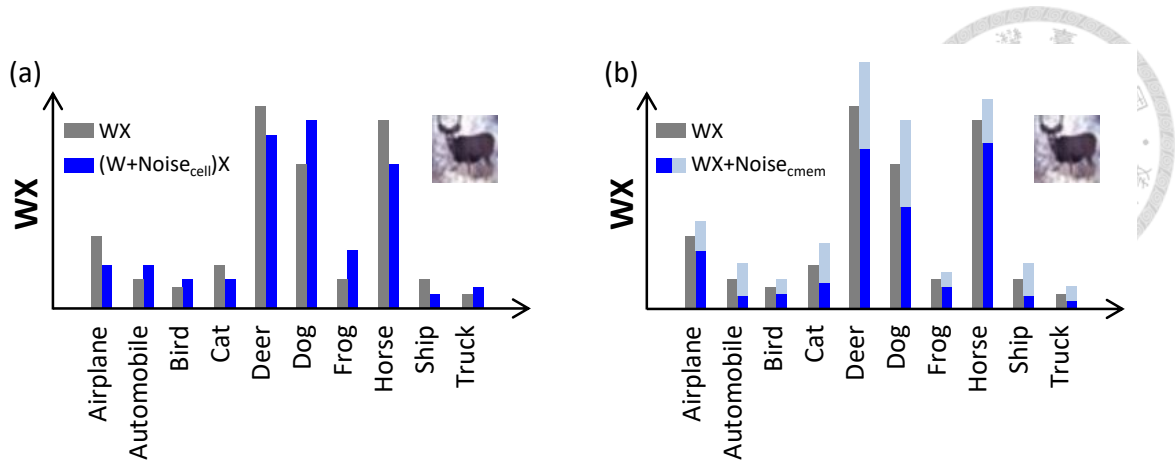


Figure 3.8: Impact of Synapse Devices and Small Size of Membrane Capacitor on Classification. (a) The noise on weight changes the inner product result, causing classification errors. (b) Low precision computing (through small capacitor size) causes uncertainty in determining classification results.

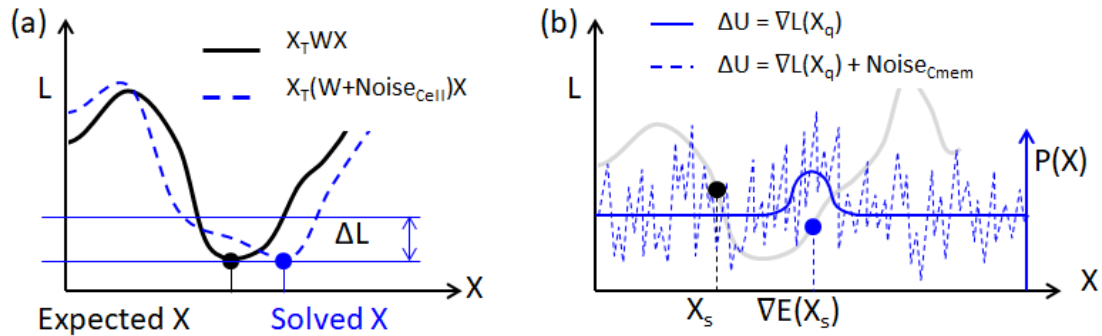


Figure 3.9: Impact of Synapse Devices and Small Size of Membrane Capacitor on Solving Optimization Problems. (a) The energy function disturbed by cell variation causes wrong solutions. (b) Low precision computing (through small capacitor size) causes noise in the update of the state distribution.

on the inner-product result, causing uncertainty as shown in Fig 3.8 (b). The additional noise also increases the hazard of inducing classification errors.

In the application of solving optimization problems. The annealing process of SNN iteratively updates the distribution to converge the probability to the state with the least energy. The updated probability of each iteration is vector-matrix multiplication (VMM), the result of \mathbf{WX} of which the primitive operation is multiply-and-accumulation (MAC). The noise from limited C_{mem} disturbs the distribution update. When the size of C_{mem} is too small, the $Noise_{C_{mem}}$ will dominate the gradient, i.e., $|Noise_{C_{mem}}| \gg |\nabla E(\phi)| \rightarrow Noise_{C_{mem}} \cong \Delta U$. As a result, the annealing process is dominated by noise and causes the distribution hardly to converge to a low energy state.

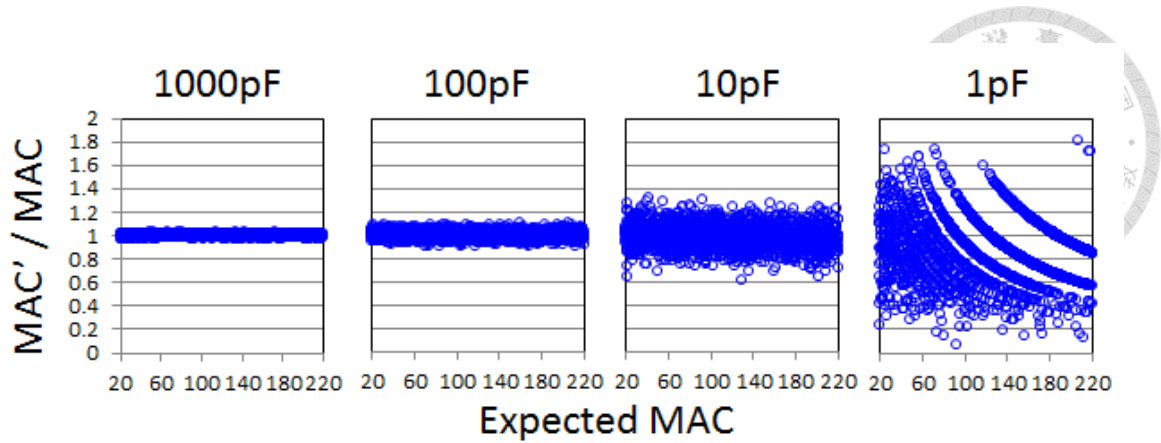


Figure 3.10: Impact of C_{mem} Capacitance. $\frac{MAC'}{MAC}$ is the ratio between MAC calculated from a spiking neuron and its expected value. One over variation of the ratio is considered as computing precision. The capacitance reduction decays the computing precision and becomes sensitive to limited sensing frequency, as shown in arcs of 1pF.

3.1.5.3 Overall Scope of Reliability Analysis

The prior reliability analytic work [9] has revealed the impact of random telegraph noise (RTN) and stuck-at-faults (SAFs) on image classification. However, the relationship between circuit constraints and memory cell characteristics is not explored. Moreover, the prior analysis only focuses on image classification, and the reliability analysis on solving optimization problems is not discussed. In this thesis, both circuit and cell parameters are considered to evaluate the reliability of image classification and solving optimization problems. The overall scope of the reliability analysis is shown in Fig.3.11. In image classification, the primitive operations of the convolution and fully connected layers, i.e., the inner product, are implemented through the spiking processing scheme. The noise model is built considering the circuit and memory cell current for efficient simulation, while the memory cell is characterized by ON-state current, normalized standard deviation, and ON-OFF ratio. Then, the model is inserted into the simulator for reliability analysis. On the other hand, the optimization problems are transformed into QUBO form and deployed into spiking neural network annealing matching. The noise model is also inserted to analyze the impact of physical parameters on the success rate for solving optimization problems.

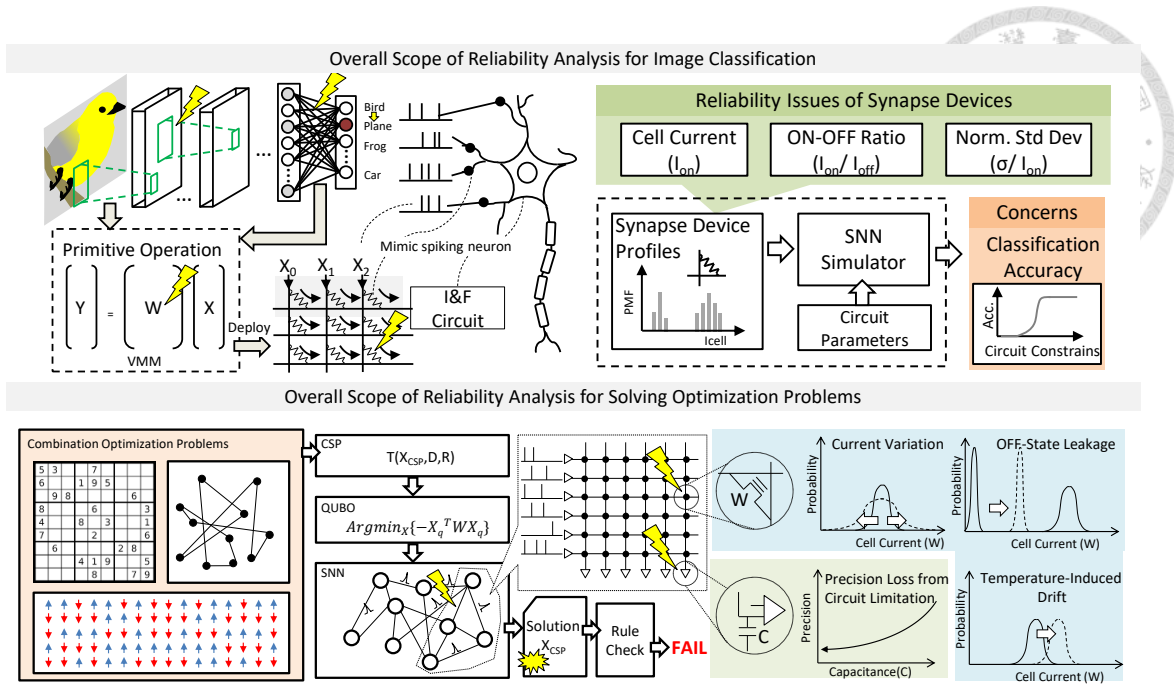


Figure 3.11: Overall Scope of Reliability Analysis for Image Classification and Solving Optimization Problems.

3.2 Design Tradeoff

Increasing the capacitance of C_{mem} also increases computing precision. However, the capacitance is a major bottleneck in the circuit design of IF-SNNs, as it leads to high energy, area, and latency cost [46, 48, 104]. To analyze the impact of the capacitance, variations, and leak currents on the success rate of CSP-solving SNNs, we derive a model to connect the neuron circuit parameters and the precision of the analog computations.

In this section, we show how to utilize the Poisson process to assess the precision of inner product results directly from the circuit parameters and show the validation of the stochastic model from circuit simulations (Section 3.2.0.1). Moreover, we discuss the impact of current variation and leakage current on the requirement of capacitance to achieve high success rates of the CSP-solving SNNs.(Section 3.2.0.2)

3.2.0.1 Impact of Capacitance



Reducing the capacitance reduces the precision of the MAC result. We show this based on experimental data in Fig. 3.10. This effect can be theoretically deduced from two ingredients: (1) The statistics of the current that is charging C_{mem} and (2) the model of the SNN spiking behavior by the Poisson process [9, 48, 105].

$$\mathbb{E}[I_c] = \sum \mathbb{E}[x_{qi}]w_i I_{cell} \quad (3.10)$$

$$\mathbb{E}[I_c] \equiv \frac{Events}{T_{fire}} * Charge\ per\ Event \quad (3.11)$$

$$Events \sim Posi(\lambda t) \quad (3.12)$$

The $\mathbb{E}[I_c]$ is the expected current charging C_{mem} . $\mathbb{E}[I_c]$ is equal to the weighted sum of input spikes multiplied by the cell currents I_{cell} . $\mathbb{E}[I_c]$ can also be modeled by the Poisson process. The Poisson process describes the probability of the total number of events occurring based on a given expected event count. In the Poisson process, λ is the arrival rate, and t is the considered time duration. λt is the expected number of events that occur within the considered time duration. A Poisson process function is modeled with a certain variance $\sigma^2 = N_{cp}$ and mean $\mu = N_{cp}$. We use the Poisson process to model the random process of the spike occurrence charging membrane capacitors of neurons in SNNs. In SNNs, λt is the expected number of spike events. The distribution of the number of spike arrivals can then be defined in Eq. 3.12. Since the current is charge divided by time, Eq. 3.11 holds. Now, we connect the Poisson process with the circuit parameters of SNNs. λt can also be expressed in terms of C_{mem} , V_{th} , T_{width} and I_{cell} . $C_{mem}V_{th}$ is the charge in the capacitor at spike time, T_{fire} . T_{width} is the duration of an input spike, and I_{cell} is the current of one memory cell. Based on this, the number of spike events

$\lambda t = N_{cp}$ in Eq. 3.13 can be calculated, establishing the connection between the Poisson process and the circuit parameters. By multiplying Eq. 3.10 by T_{fire} , it results in CV_{th} , which is inserted into Eq. 3.14. As a result, we acquire the distribution of the charging current in Eq. 3.14.

$$N_{cp} = \frac{C_{mem}V_{th}}{T_{width}I_{cell}} \quad (3.13)$$

$$\sum_i \mathbb{E}[x_{qi}]w_i \frac{T_{fire}}{T_{width}} \sim Posi(N_{cp}) \quad (3.14)$$

$$C_{mem} = \frac{T_{width}I_{cell}}{(\frac{\sigma}{\mu})^2V_{th}} \quad (3.15)$$

With the properties of the Poisson process, we can obtain the normalized standard deviation ($\frac{\sigma}{\mu} = \frac{\sqrt{\lambda t}}{\lambda t} = \frac{\sqrt{N_{cp}}}{N_{cp}}$), which is $N_{cp}^{-1/2}$ in a Poisson process.

The value of N_{cp} in Eq. 3.13 is influenced by the parameters C_{mem} , V_{th} , and T_{width} . To decrease N_{cp} , we can increase C_{mem} and V_{th} or decrease T_{width} . However, when the objective is to maintain N_{cp} while using different memory devices, C_{mem} becomes a critical factor. It enables scalability for a broad range of cell currents, spanning from microamps (uA) to nanoamps (nA). Unlike V_{th} and T_{width} , which are subject to limitations imposed by electronic physics, the scalability of C_{mem} is achieved by sacrificing area. Although enlarging the capacitance may not be cost-effective, it is a viable option.

Increasing C_{mem} alleviates the normalized standard deviation but increases the required energy for generating an output spike. The energy of the synapse array is defined by Eq. 3.16. In this equation, the term $\int_0^{T_{fire}} I_{cell} \sum x_i(t)w_i dt$ represents the charge stored in the capacitance while the neuron fires. To establish the connection to C_{mem} , we substitute this term with $V_{th}C_{mem}$, resulting in Eq. 3.17. This modified equation shows that the energy required to generate an output spike is directly proportional to C_{mem} .



$$Energy = \int_0^{T_{fire}} V_d I_{cell} \sum x_i(t) w_i dt \quad (3.16)$$

$$Energy = V_d V_{th} C_{mem} \quad (3.17)$$

In conclusion, enlarging C_{mem} can reduce the normalized standard deviation but with the cost of energy. The following subsection further introduces the impact of memory cell properties on the capacitance of C_{mem} .

3.2.0.2 Impact of Cell Current on C_{mem}

The non-ideality of cell current includes the current variation and the OFF-state leak current. Although these current properties can be controlled to some extent, they are unavoidable in analog computations.

3.2.0.3 Current Variation

Due to inherent variations occurring in analog computing, it is unavoidable that the programmed weights have variations. Therefore, the currents suffer from variations as well. To alleviate the effects of current variations, in this thesis, we propose to duplicate memory cells.

By replicating memory cells, the normalized current distribution is narrowed [48, 106]. When we connect cells' output into one, the expected output current grows faster than its standard deviation. As a result, the normalized standard deviation is reduced, which is illustrated in Fig. 3.12.

We reduce the normalized standard deviation to $n^{-1/2}$ of the memory cell by copying the conductance of a memory cell to cells sharing the same inputs. However, the cost is that the mean current is increased by n times. The increase of I_{cell} enlarges $N_{cp}^{-1/2}$ (see Eq. 3.14). To reduce the normalized standard deviation again, a larger membrane

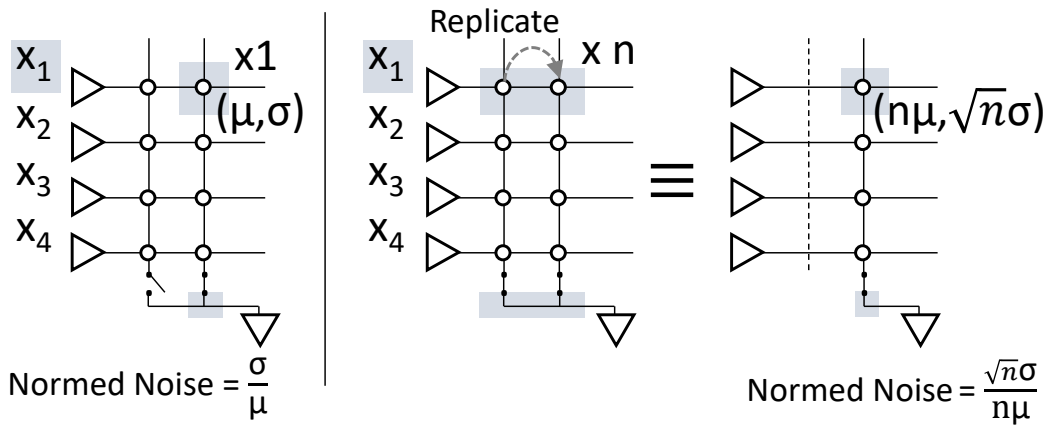


Figure 3.12: Replication of Memory Cells to Ease Computing Error. Replication of memory cells reduces the normalized standard deviation but increases the expected current. In the case of a cell with an expected current of μ and a standard deviation of σ , when we replicate the cell into n columns and connect their output nodes together, these cells can be considered as one cell with an expected current of $n\mu$ and a standard deviation of $\sqrt{n}\sigma$. Through replication, the normalized noise can be reduced from $\frac{\sigma}{\mu}$ to $\frac{1}{\sqrt{n}}\frac{\sigma}{\mu}$.

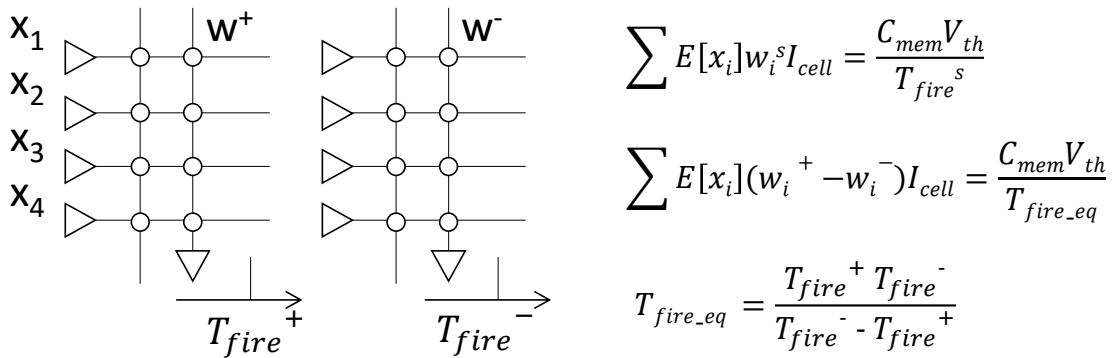


Figure 3.13: Implementation of the Signed Weight on a Pair of the Memory Array. The equivalent-and-signed firing time is calculated from the firing time of two neurons.

capacitance is required.

3.2.0.4 Leak Current from I_{OFF}

When a cell is at the OFF state, in ideal scenarios, the conductance is assumed to be zero and, therefore, the resistance to be infinite. However, in practice, the resistance of a non-volatile memory cell is always finite. When the input spike is provided to the neuron, the OFF-state current contributes to the charging of the membrane capacitor. As a result, these leak currents cause earlier firing times (compared to the ideal firing times) and, therefore, lead to low computing precision. As can be seen in Eq. 3.18 and Eq. 3.19, we

can acquire the firing time by dividing the total charge of $C_{mem}V_{th}$ by the expected current. These two equations show that the T_{fire} will be larger than the firing time with leakage T_{fire_l} when the I_{leak} in the denominator contributes current. The factor of $W_{MAX} - W_i$ is the number of cells in the OFF state. Thus, as can be seen in Eq. 3.20, the I_{leak} is positive and increased when I_{OFF} increases, causing earlier firing times.

The earlier firing time can cause computing errors. Due to the differential neuron pair for the negative weight (see Fig. 3.13), the firing time can be scaled back by the factor of $\frac{I_{cell} - I_{OFF}}{I_{ON}}$, as can be seen in Eq. 3.22. The scaling factor can be deduced from setting $w_i = w_i^+ - w_i^-$, and alternating the $w_i^s I_{cell}$ into the one in Eq. 3.19., where the sign s is denoted by + or -.

$$T_{Fire}^s = \frac{C_{mem}V_{th}}{\sum \mathbb{E}[x_{qi}]w_i^s I_{cell}} \quad (3.18)$$

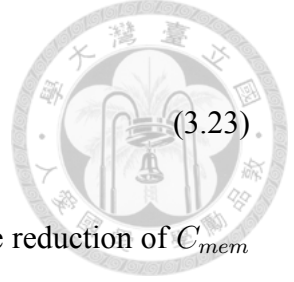
$$T_{Fire_l}^s = \frac{C_{mem}V_{th}}{\sum \mathbb{E}[x_{qi}](w_i^s I_{cell} + I_{leak}^s)} \quad (3.19)$$

$$I_{leak}^s = (w_{MAX} - w_i^s)I_{OFF} \quad (3.20)$$

$$\sum \mathbb{E}[x_{qi}]w_i I_{cell} = \frac{C_{mem}V_{th}}{T_{fire_eq}} \quad (3.21)$$

$$T_{fire_eq} = \frac{T_{fire_l}^+ T_{fire_l}^-}{T_{fire_l}^- - T_{fire_l}^+} \frac{I_{cell} - I_{OFF}}{I_{cell}} \quad (3.22)$$

Although the impact of I_{OFF} can be eliminated, the leak current effectively contributes charges to C_{mem} . These charges from leak current occupy charge capacity of $C_{mem}V_{th}$, and the remaining capacity available for operation is decreased. We get the effective capacitor size C_{mem_eff} from C_{mem} minus the occupied capacitance in Eq. 3.23.



$$C_{mem_eff}(T_{fire}) = C_{mem} - \frac{\sum \mathbb{E}[x_{qi}] I_{leak} T_{fire}}{V_{th}} \quad (3.23)$$

The C_{mem_eff} is decreased by increasing of I_{leak} . Moreover, the reduction of C_{mem} for operation reduces the charge capacity available for operation and causes low computing precision as shown in Eq. 3.13 and Eq. 3.14. As a result, an increase of I_{leak} induces a reduction of C_{mem_eff} , and a larger C_{mem} is required to preserve computing precision.

3.3 Reliability Evaluation for Image Classification

3.3.1 Evaluation Setup

3.3.1.1 Simulation Framework

To simulate the effect of cell current characteristics on the requirement of circuit cost, the simulation framework is built to evaluate classification accuracy under circuit parameters. The simulation framework architecture is shown in Fig. 3.14. A convolutional neural model is provided to the Inference Engine. The engine utilizes the SNN Simulation Kernel to decompose both convolutional layers and fully-connected layers in convolutional neural network (CNN) into vector-matrix multiplication (VMM) and deploys these VMM operations into the SNN macro of Fig. 3.1. To explore the effects of cell current characteristics on classification accuracy, the Noise Injector adds noise to weights based on cell current distributions of collected memory technologies. Finally, the SNN dynamic model of Eq. 3.3 is provided to the SNN Simulation Kernel to evaluate spiking neural dynamics. In this thesis, the size of a memory crossbar is set to 128x128. The current transporter gain κ is 0.01. The widely used workload of 4-bit CNN7 on the CIFAR10 dataset shown in Fig. 3.15 is adopted to analyze the effect of cell current characteristics on classification accuracy.

	1T-NOR [46]	WOx [108]	HfOx [109]
ON-OFF Ratio	>1000	40	>1000
Vd, Vg	1.5V, 5V	0.6V,-	0.2V,-
ON State Current (I_{ON})	10uA	30uA	130uA
σ/I_{ON}	< 5%	< 5%	<30%
Accuracy (0.1pF,1pF,10pF)	10%,90%,91%	10%,10%,91%	10%,80%,83%

2T-NOR [110]	FeFET-Low [111]	FeFET-Normal [111]	FeFET-High [111]
>1000	40	570	>1000
0.2V,0	0.2V, 0.6V	0.2V, 0.6V	0.2V, 0.6V
1.5uA	6uA	10uA	10uA
5%	30%	15%	7%
91%,91%,91%	10%,10%,79%	10%,87%,88%	10%,89%,91%

Table 3.1: Summary of Cell Characteristic for Accuracy Evaluations. The evaluations are under the condition of membrane capacitors of 0.1 pF, 1 pF, and 10 pF.

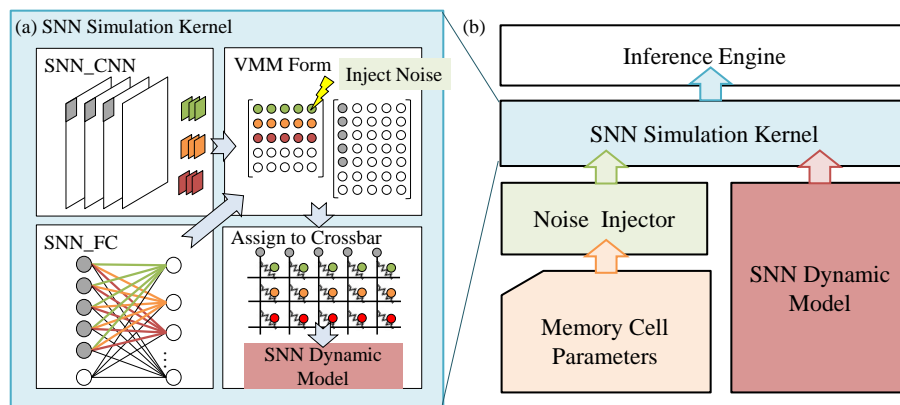


Figure 3.14: Overview of Simulation Framework

3.3.1.2 Collection of Various Technologies

To analyze how cell current characteristics affect classification under limited circuit cost using existing memory techniques, various non-volatile memory technologies are collected for simulation. The memory properties, including cell current characteristics, operation conditions, and classification accuracies under different sizes of membrane capacitors, are listed in Table 3.1. These collected technologies include NOR flash, WOx ReRAM, HfOx ReRAM, 2TNOR, and FeFET. The cell characteristics of the WOx ReRAM are based on [107], which shows WOx ReRAM has a small ON-OFF ratio but with a tight distribution.

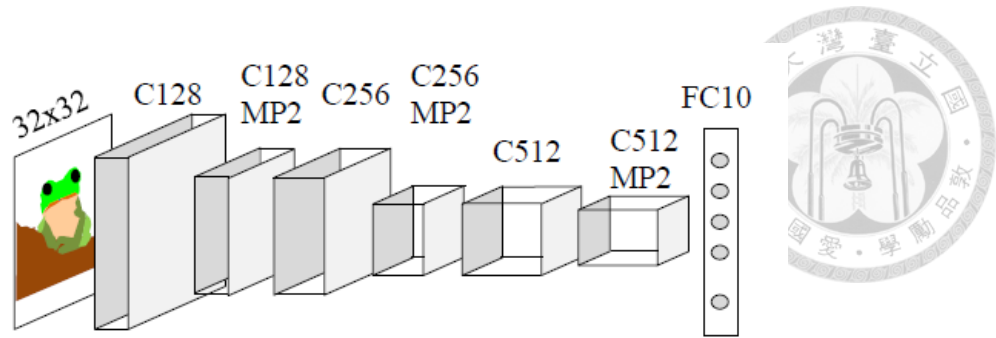


Figure 3.15: The CNN Model of VGG7. “CN ” means a convolutional layer with N output channels, “MPN ” represents the max-pooling of N by N, and ”FC10” means a fully-connected layer with the output node number of 10.

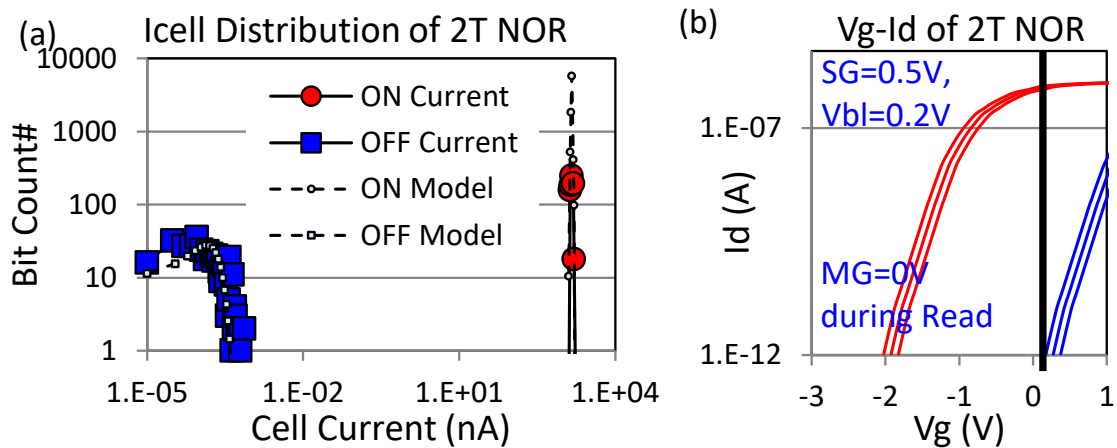


Figure 3.16: Current Distribution and Vg-Id Curves of 2T NOR.

Conversely, HfOx in [108] has a large ON-OFF ratio but with a large normalized standard deviation (σ/I_{on}). NOR flash technologies are also selected. The 2T-NOR [110], different from general 1T-NOR [46], has an additional selecting transistor beside a flash cell. The cell current can be trimmed by selecting the transistor without operating flash cells on the near-threshold region with large noises. Thus, 2T-NOR can get a small current without large noises, as in Fig. 3.15. Furthermore, three design scenarios of FeFET [111] are included: high, normal, low remnant polarization and coercive field (PR & EC) representing the boundary conditions in the P-V loop of FE caused by process variation. The high PR & EC FeFET results in a larger memory window but with higher V_t variation than the other two FeFETs. The FeFET current distribution and statistics of V_t from TCAD simulations are shown in Fig. 3.17 (after careful calibration with measurement data for both the underlying 14nm FinFET device and the upper FE layer included in the gate stack [111]).

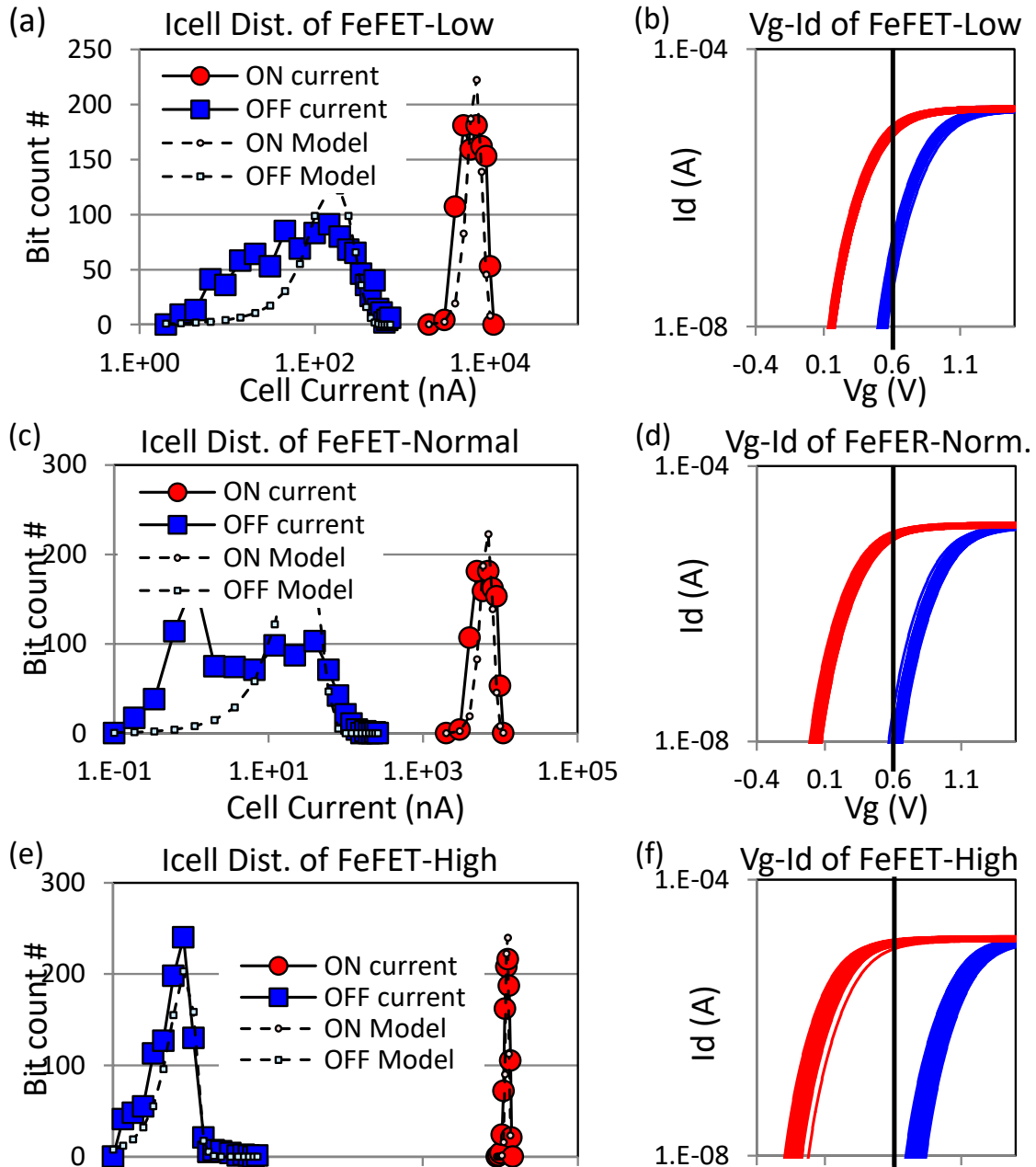


Figure 3.17: Current Distributions and Vg-Id Curves of FeFET. Three design scenarios of FeFET with high, normal, and low remnant polarization and coercive field (PR & EC) are included in the accuracy evaluation.



	Low	Normal	High
μ of Low V_t	0.266	0.138	-0.028
σ of Low V_t	0.01	0.017	0.026
μ of High V_t	0.629	0.732	0.874
σ of High V_t	0.011	0.017	0.029
Capacity ($\Delta\mu/\Sigma\sigma$)	17.3	17.5	16.4

Table 3.2: FeFET Parameters of Design Schemes

3.3.2 Experiment Result

3.3.2.1 ON-OFF Ratio

The ideal OFF-state current is 0. However, the value of resistance hardly reset to infinite. The effect of these OFF-state currents should be considered. As shown in Fig. 3.18(a), the red bars are the accuracies dropped by the finite ON-OFF ratio of cell currents. The reason for the accuracy loss is that the accumulation of these OFF-state currents shifts output current while the shifting current from OFF-state cells is expected to be 0. A method to eliminate the effect of OFF-state current is inspired from [112]. Thanks to encoding weight by differential pair, the effect of OFF-state current can be eliminated by rescaling the parameter of β in Eq. 3.24.

$$\Sigma E[X_i]W_i = \alpha\beta' C_{mem} V_{th}/t_{fire}/\kappa \quad (3.24)$$

$$\beta' = \beta \frac{I_{ON}}{I_{ON} - I_{OFF}} \quad (3.25)$$

The accuracy after calibration is as shown in blue bars in Fig. 3.18 where the accuracies of low ON-OFF ratio are recovered. However, these OFF-state currents still contribute currents to array output currents. As the conclusion of Eq. 3.14, the increasing array current will reduce the precision of the computing result. The simulation result shows that both FeFET-Low in Fig. 3.19(a) and WOx ReRAM in Fig. 3.19(b) with the

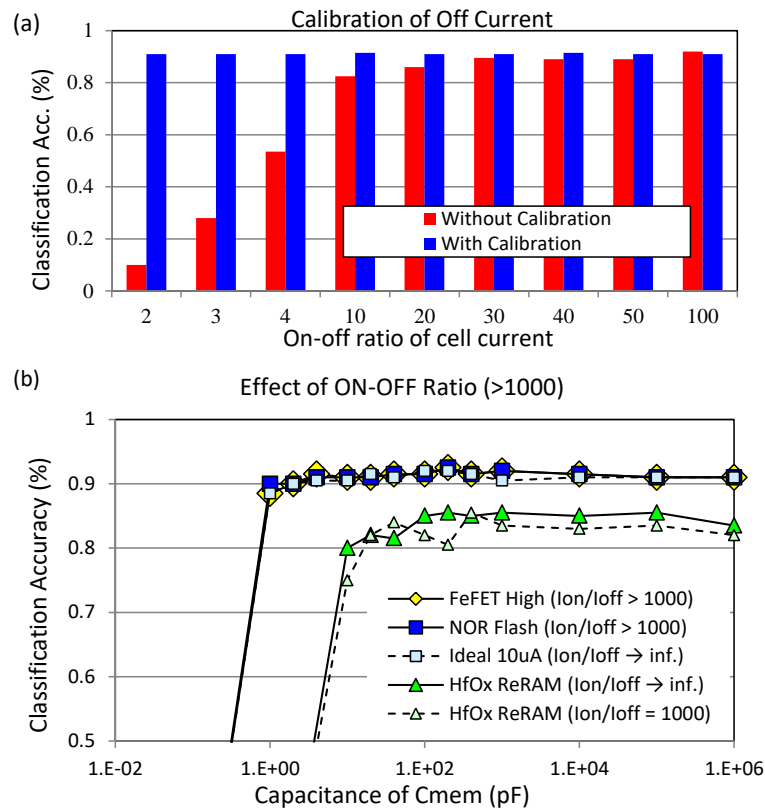


Figure 3.18: Accuracy Evaluation Considering OFF-state Current (a) Accuracy with and without OFF-current calibration without membrane capacitor size constraints. (b) Requirement of membrane capacitor for memory technologies with ON-OFF ratio >1000 including NOR Flash memory, HfOx ReRAM, and FeFET-High. When the ON-OFF ratio exceeds 1000, the accuracy is equivalent to zero OFF current.

ON-OFF ratio of 40 require 10x the size of the membrane capacitor to preserve the accuracy compared to the case of no OFF-state current. On the contrary, the requirement of membrane capacitors for memory technologies with an ON-OFF ratio >1000 is the same as OFF-current free, as shown in Fig. 3.19(b).

3.3.2.2 ON-state Current

Memory cells with large ON current also require a large size of the membrane capacitor. The large cell currents charge the membrane capacitor easily, and the large membrane capacitor is required to prevent neuron firing by few input spikes. As shown in Eq. 3.14, the large size of the cell current directly decreases the number of N_{cp} , which reduces output precision. The theoretical conclusion is also shown in Fig. 3.20(a) that the required size of the membrane capacitor is proportional to the size of the ON-state

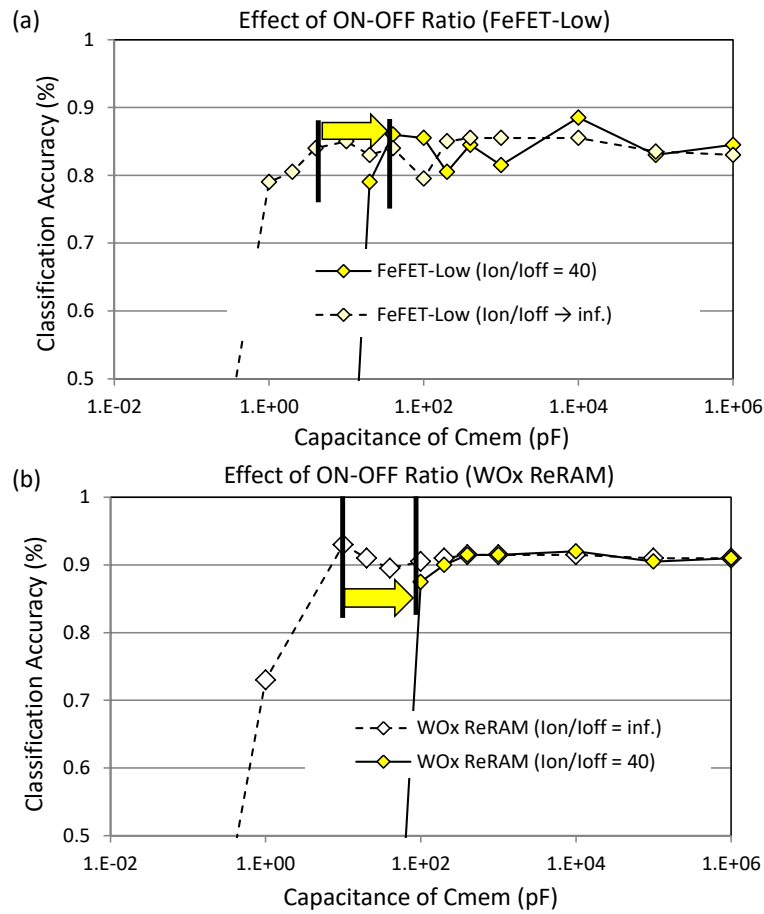


Figure 3.19: Effect of Low ON-OFF Ratio. Classification accuracy of (a) FeFET-Low with ON-OFF ratio =40, and (b) WOx ReRAM with ON-OFF ratio = 40.

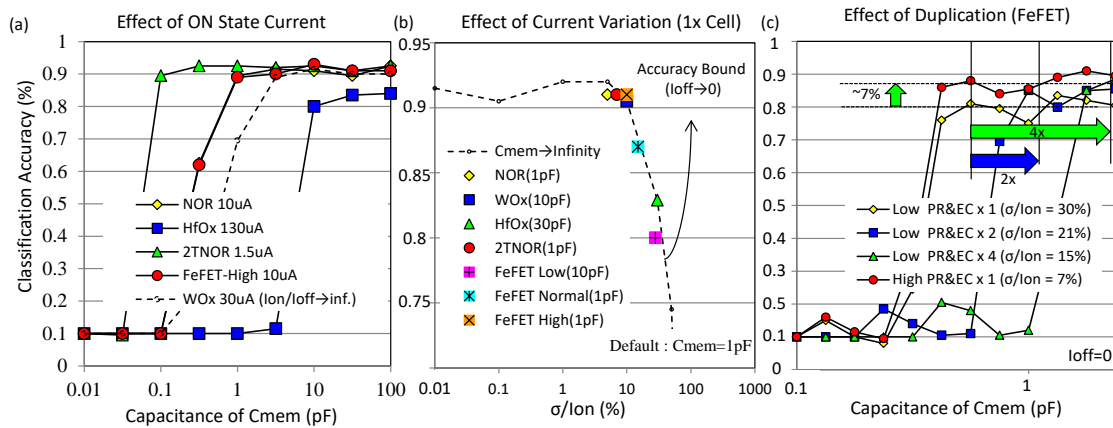


Figure 3.20: Effects of ON-state Current and Variation. (a) Requirement of membrane capacitor size for various memory technologies without considering ON-OFF ratio. (b) Accuracy loss from normalized standard deviation without duplication of weights. (c) Duplication of weights increases accuracy, but a larger membrane capacitor is required.

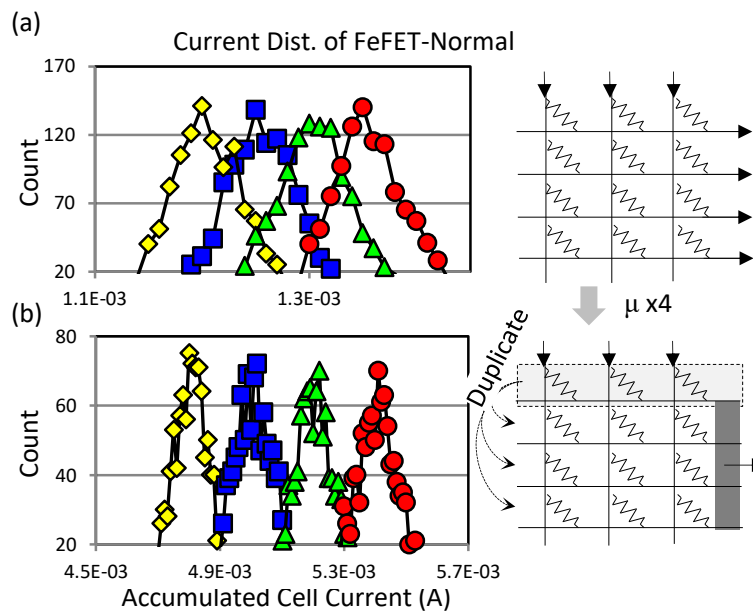


Figure 3.21: Current Replication Eases Current Variation. (a) Current distributions accumulated from 120, 125, 130, 135 FeFET-Normal cells and (b) Those distributions with four copies of duplicating weight arrays.

current. ReRAM HfOx [109] enlarges the ON-OFF ratio by increasing ON-state current to 130uA. The required size of a membrane capacitor is up to 10pF, which is ten times larger than the required size of a 1T-NOR flash. Moreover, both FeFET and 1T-NOR Flash are designed for low-latency and error-free reading. Thus, the output currents are not well trimmed for spiking neuron purposes. With the ON-state current up to 10uA, 1pF of membrane capacitor is still required. Although the design scenario of Low RP&EC FeFET reduces ON-state current, the normalized standard deviation is increased, limiting the classification accuracy as Fig. 3.20(b). On the contrary, the 2T-NOR trims its current down to 1.5uA with a controllable current variation. The required size of the membrane capacitor 2T-NOR is only one-tenth of the size of 1T-NOR Flash.

3.3.2.3 Normalized Standard Deviation

Normalized standard deviation is an index of noise-to-signal ratio and is defined as σ/I_{ON} . To purely analyze the accuracy loss from the normalized standard deviation, the membrane capacitor is enlarged for those technologies with large ON-state current and small ON-OFF ratio. Fig. 3.20(b) shows that the accuracy drops while the normalized

standard deviation of the cell current is >10 . The effect of normalized standard deviation can be alleviated by duplication of weights by combining output currents as in Fig. 3.21. With duplication, the growth of signal (window) is faster than noise (standard deviation) as described as Eq. 3.26 and 3.27.

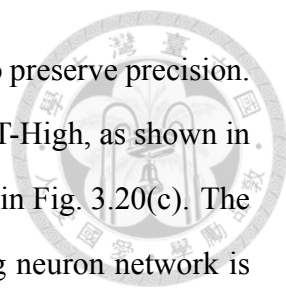
$$\mathbb{E}[\sum_n U_i] = n(\mathbb{E}[U_i] - \mathbb{E}[V_i]) \quad (3.26)$$

$$DEV(\sum_n U_i - \sum_n V_i) = \sqrt{n}DEV(U_i - V_i) \quad (3.27)$$

The U and V are two random variables of sum-of-product results, and the $\sum_n U_i$ and $\sum_n V_i$ are the output current from the weights duplicated by n times. The simulation result of Fig.3.21 shows that the duplication of weights can reduce overlapping areas of output current distribution. However, the output current is also increased by weight duplication. Based on the conclusion of Eq.3.14, the duplication methodology increases output current and requires a large membrane capacitor to preserve classification. In other words, the duplication of weight can alleviate the effect of cell current variation, but the cost is to enlarge the membrane capacitor, increasing computing energy. Taking FeFET-Normal Low as an example, the normalized standard deviation of a single cell is up to 30%, and the accuracy drops to 80% without weight duplication as in Fig. 3.20(b). After duplication of weights by 4x, the maxima accuracy can be recovered to 87%, but four times the size of the membrane capacitor is required as in Fig. 3.20(c).

3.3.3 Discussion

The non-volatile memory specifications for storage are not the same as those for spiking neural networks. For storage purposes, both high ON-OFF and cell capacity, indexed by signal-to-noise ratio as shown in Table 3.2, are considered to reduce sensing latency and raw bit error rate. The memory technologies designed for SNN place greater emphasis on ON-state cell current and normalized standard deviation. ReRAM HfOx has a high ON-



OFF ratio, but its large current requires a large membrane capacitor to preserve precision. On the other hand, FeFET-Normal is with higher capacity than FeFET-High, as shown in Table 3.2, but two times of membrane capacitor is required as shown in Fig. 3.20(c). The reason the storage capacity of v_t is not a feasible index for a spiking neuron network is that the spiking neural network evaluates its dynamic by charging the membrane capacitor with cell currents directly without filtering current noise, while the conventional read operation is to compare v_t with reference voltage regardless noisy current beyond reference current. Moreover, high precision is not enough to minimize the requirement of a membrane capacitor. The ON-OFF ratio also needs to be considered at the same time. For example, ReRAM WO_x with low normalized standard deviation still has low classification accuracy under 1 pF of membrane capacitor because of its low ON-OFF ratio. Ten times of the membrane capacitor is required to reach the classification accuracy of ReRAM WO_x without IOFF current. Transistor-based memory technologies are good candidates because of their high ON-OFF ratio and low standard deviation. However, the memory cell currents must be tuned to small without increasing current variation. The 2T-NOR device is an example that meets the requirement. The cell current of 2T-NOR is trimmed by an additional selecting transistor beside the NOR flash cell, which the variation is more easily controlled by the process than the NOR-Flash cell. In this case, 2T-NOR can be trimmed to a small current with a controllable current variation. However, the 2T-NOR costs more than conventional flash-based memory cells. FeFET is a candidate that can be trimmed to a small current and ease the current variation with the proper setting of operating conditions.

3.4 Reliability Evaluation for Annealing Purpose



3.4.1 Evaluation Setup

3.4.1.1 Collection of CSP Models

The CSPs for SNN annealer include a 2D-square-lattice Ising Model, Sudoku, and Traveling Salesman Problem (TSP) as listed in Table 3.3. These CSPs have been used in previous works [101, 102]. The metric for reliability analysis is the success rate of the solution of the SNN annealer meeting the constraints.

The 2D-square-lattice Ising model depicts a spin class system. Each spin is coupled with its neighbor's spins. The coupling of J represents the interaction between spins. $J_{ij}(V_j)$ is the coupled state of spin i given the state of spin j . The rule is to check whether the expected state from neighbor spins is the same as the current state of the spin i . The weight of J_{ji} is 1 for positive interaction, -1 for negative interaction, and 0 otherwise. The size of the 2D lattice square is 64x64.

Sudoku is a puzzle game in which the numbers in a 9x9 matrix meet the following constraint: Each number only occurs once in the same column, row, and 3x3 submatrix. The weights are 1 when they represent the relationship between different numbers in the same column, row, and 3x3 block. Negative weights represent the relationship between the same numbers (under the same setting).

The TSP is one of the most famous CSPs. A salesman visits all cities on a map where the traveling distance is smaller than a given criterion of K , and each city can only be visited once. The TSP is based on data of Berlin52 from TSPLIB95 [113] with 52 cities. We convert the TSP into the QUBO form by using the method in [101], as described in Sec. 3.1.4.2. In the TSP problem, the distances are normalized by 0.5x the maximal distance and quantized to 4 bits. The weights representing the selection of multiple cities at the same time or the re-selection of the same city are set to the negative average distance

between a certain city and its neighbors. Otherwise, the weights representing the selection that meets the constraints are set to 15 minus distance.



3.4.1.2 SNN Model Setup

The dynamic process of IF circuits is approximated by the modified Poisson process (MPP), as shown in Eq. 3.28. By the MPP, the term $\mathbb{E}[\sum_i w_i x_i]$ in Eq. 3.5 is sampled from a Gaussian distribution, where only positive y is sampled. The Poisson process can be closely approximated by a Gaussian distribution, which we employ for efficient sampling on GPUs. We deduce the MPP from Eq. 3.2, 3.14, 3.21.

$$\sum \mathbb{E}[x_{qi}]w_i = \frac{C_{mem}V_{th}}{\lceil \frac{N_{cp}Z}{y} \rceil T_{width}}, \quad (3.28)$$

$$\text{where } y \sim G(x_{act}, \frac{K_c x_{act}}{\sqrt{N_{cp}}})$$

The x_{act} is a MAC value, i.e., $\sum u_i w_i$. K_c is the fitness constant set to 1.6. The comparison of MAC results generated from a dynamical process and a Poisson-like sampling is shown in Fig. 3.22. The selection of proper temporal resolution is important to provide efficient simulation while preserving accurate reliability analysis. In the stochastic model, the temporal resolution is delineated by the pulse width of the input spike. The specific voltage waveform present during this pulse does not affect the neuron's firing cycle. Rather, the emphasis is on the total charge accumulated over the entire duration of the pulse width. Therefore, a detailed representation of the waveform within the pulse width is not essential for the simulation. The comparison shows that the sampling process meets the trend of results from evolving the dynamic system.

Moreover, in order to illustrate that the success rate of solving the CSP is constrained by the capacitor size, we investigate the impact of varying capacitance sizes. We use capacitor sizes ranging from sizes that preserve the success rate to sizes that lead to solving failures.

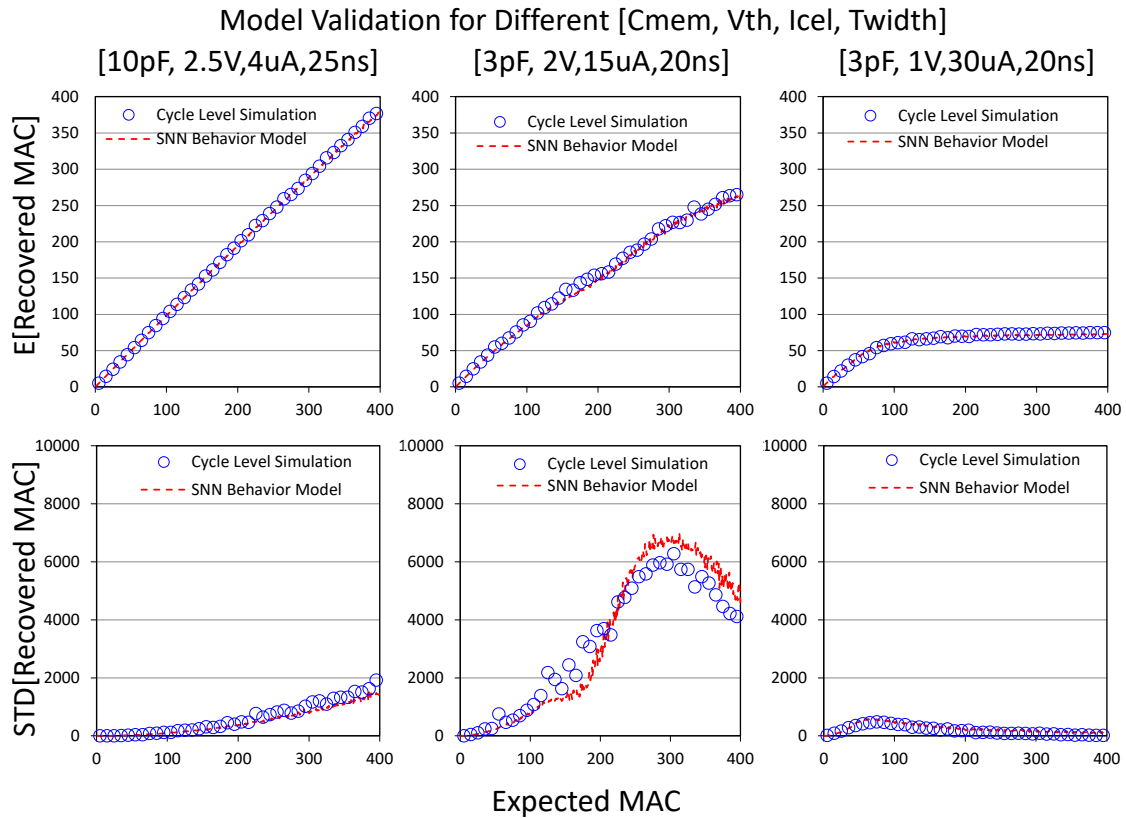


Figure 3.22: Comparison of Cycle Level Simulation and SNN Behavior Model. The cycle level simulation is the process of sampling input spikes and converting the output firing time into MAC value cycle by cycle. The SNN behavior model directly samples the MAC result through its expected value. The SNN behavior model fits the average and standard deviation from the cycle-level simulation.

3.4.1.3 Energy Model Setup

The energy model of the SNN hardware is based on [114]. Its energy consumption comprises the analog spike process and the algorithmic circuit at the 65 nm technology node. The algorithmic part is independent of the memory devices. We select a time window, τ , to 800 cycles per iteration for our energy evaluation. The energy of the analog spike process considers the current mirror. Thus, the energy of the spike process is two times the energy in Eq. 3.17.

3.4.1.4 Collection of Memory Devices

We collect device data from four types of memory technologies, including resistive memory (ReRAM) with WOx and HfOx and field-effect transistor (FET) based memory

with 2T-NOR Flash and FeFET. The memory device parameters are shown in Table 3.4 and 3.5. The HfOx-based ReRAM includes two architectures: Pt/HfOx/Ti/Al and TiN/HfO2/Ti/TiN. They have different ON-OFF ratios. The 2T-NOR device is NOR Flash memory with one additional selection transistor, which optionally can reduce the cell current variation.

We use three design settings of the FeFET device, with low, normal, and high programming voltage [111, 115].

We collect temperature models for each memory (except RRAM with Pt/HfOx/Ti/Al). We also build up the current drift model of WOx ReRAM, TiN/HfO2/Ti/TiN ReRAM, 2T-NOR, and FeFET for a temperature range from 273K to 358K as seen in Fig. 3.23.

Table 3.3: Constraint Satisfaction Problems

	Ising	Sudoku	TSP
Variables	States in 2D lattice-square	Values in 9x9 matrix	Selected city at each time
Value Domain	[up,down]	[1 to 9]	Set of City
Subset of t	$\{(x_i, x_j) s_i, s_j \text{ binded}\}$	$\{(x_1, \dots, x_9) x \text{ in same col, same row, or same } 3 \times 3 \text{ block}\}$	$\{(x_1, \dots, x_9) \forall X\}$
Rules	$(V_i, V_j) = t_k$ $V_i = J_{ij}(V_j)$	$\forall (V_i, V_j) \subset t_k$ $V_i \neq V_j$	$\forall (V_i, V_j) \subset t_i$ $V_i \neq V_j \wedge$ Distance $\leq K$

Table 3.4: Parameters of Resistive Memory Technologies

	WOx [108]	Pt/HfOx/Ti/Al [109]	TiN/HfO2/Ti/TiN [116]
I_{ON} (uA)	10.4	179	53.0
I_{ON}/I_{OFF}	21.6	> 1000	3.6
σ/u	0.036	0.3	-

Table 3.5: Parameters of Transistor-based Memory Technologies

	2T-NOR [117]	FeFET [111, 115]		
		Low	Normal	High
I_{ON}	0.146	6.00	10.0	10.0
I_{ON}/I_{OFF}	> 1000	40	570	> 1000
σ/u	0.01	0.3	0.15	0.075

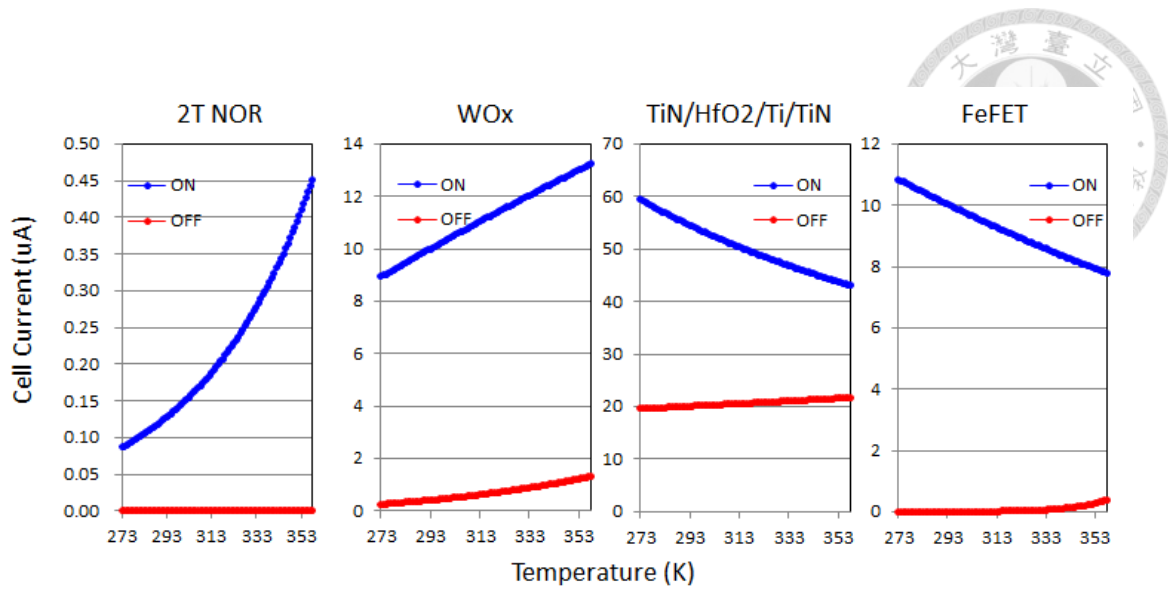


Figure 3.23: Temperature-induced Drift Curve of 2T NOR Flash, WOx, TiN/HfOx/Ti/TiN, and FeFET.

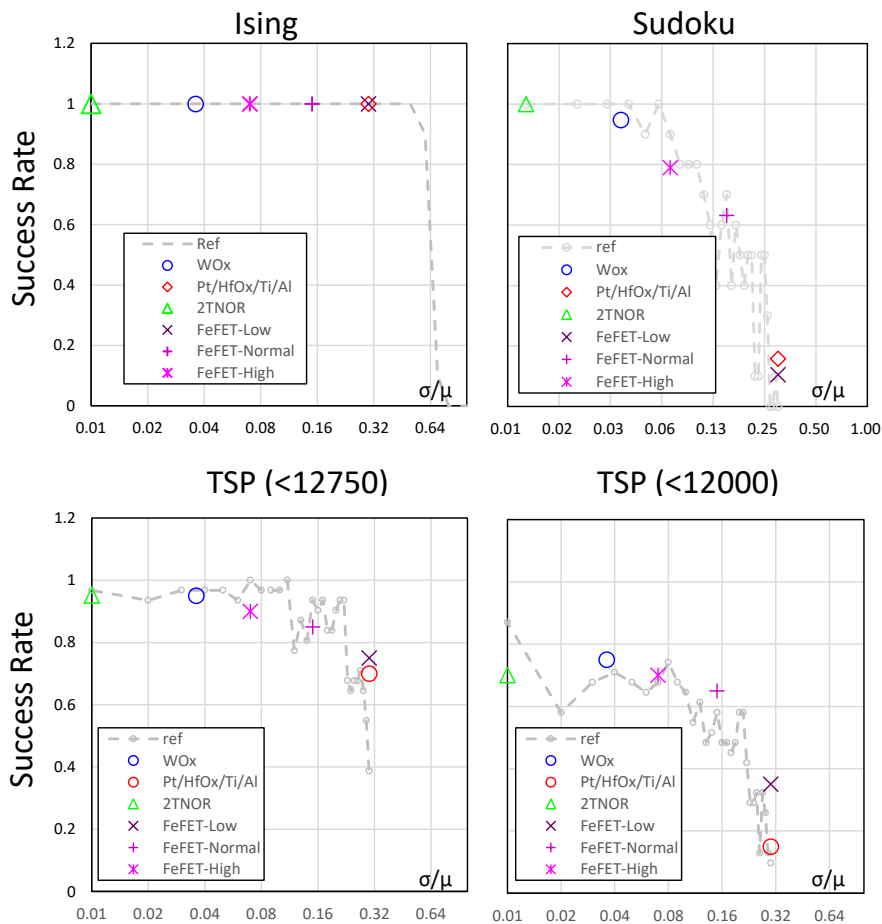


Figure 3.24: Impact of Cell Current Variation on Success Rate. The normalized standard deviation $> 5\%$ and $> 15\%$ decays the success rate of solving Sudoku and TSP, respectively. The process of solving the Ising model tolerates all the cell variation.

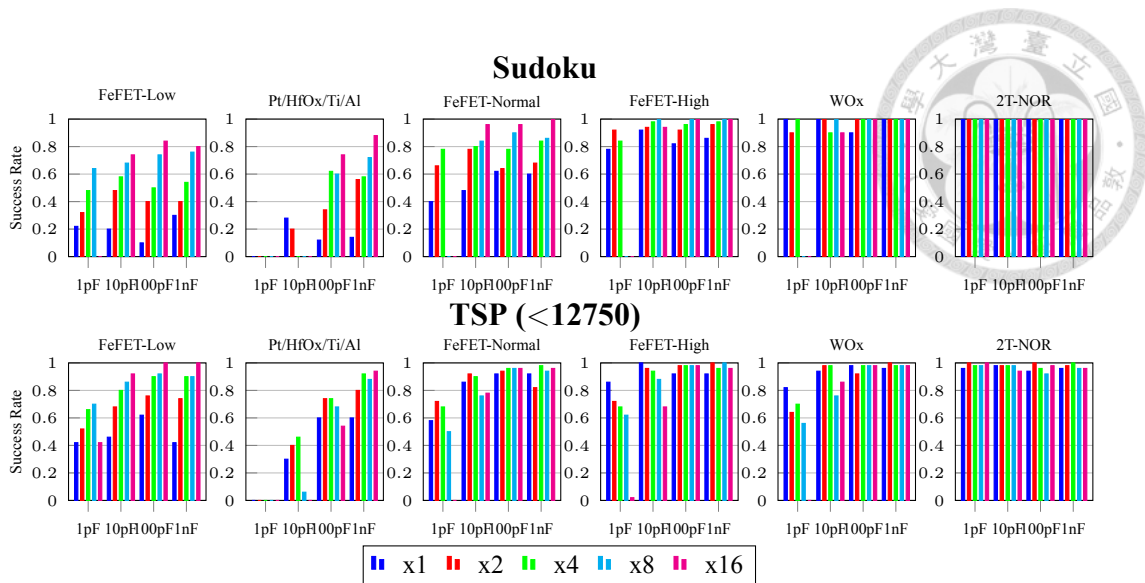


Figure 3.25: Impact on the Requirement of Membrane Capacitance for Different Replication of Memory Cells. Increasing the number of replications increases the success rate, but a higher capacitance is required.

3.4.2 Experiment Result

3.4.2.1 Impact of Current Variation

The simulation results in Fig. 3.24 show the impact of current variation, where the OFF-state leakage current is set to zero. The application of the Ising model can tolerate the cell normalized standard deviation up to 50%. Due to the high tolerance, all the memory technologies can preserve the success rate.

On the contrary, solving Sudoku requires a normalized standard deviation $<5\%$ to sustain the success rate, such as WOx ReRAM and 2T-NOR. The success rate drops to 80% when the normalized standard deviation increases to 7% as FeFET-High. It fails to solve the Sudoku while the normalized standard deviation is more than 25%. The success rate of solving TSP is preserved until the normalized standard deviation increases to 15%. Thus, memory devices such as WOx, 2T-NOR, FeFET-Low, and FeFET-Normal with normalized standard deviation $<10\%$ can apply to solve TSP with an ignorable success rate drop. On the contrary, FeFET-Low and Pt/HfOx/Ti/Al ReRAM, with a normalized standard deviation of around 30%, dramatically drop the success rate.

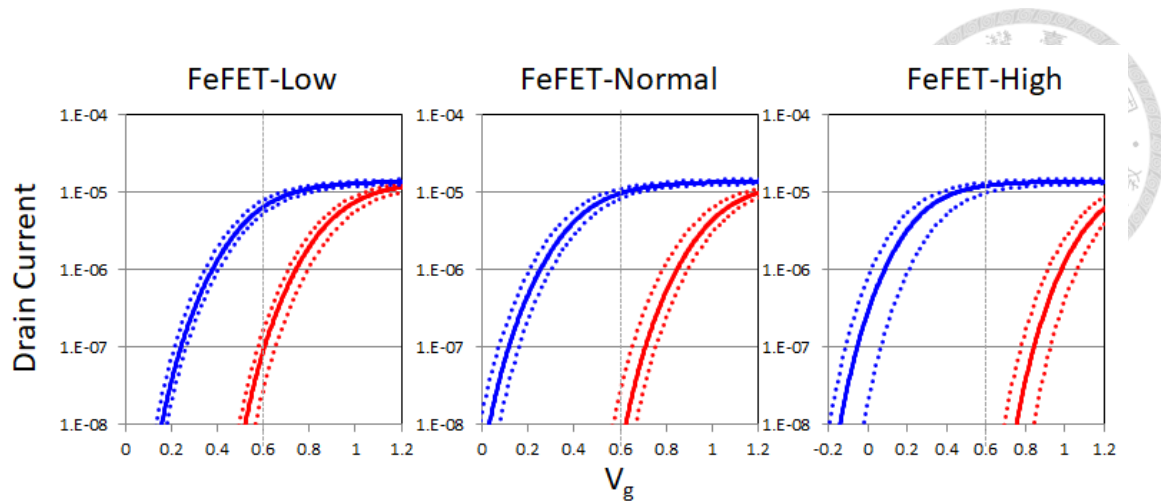


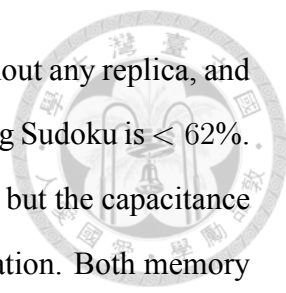
Figure 3.26: I_d - V_g curve of FeFET Devices with Low, Normal, and High Programming Voltage.

FeFET-Normal and FeFET-Low operate in the sub-saturation region with large variation, as shown in Fig. 3.26. On the other hand, FeFET-High operates in the saturation region, and the current variation is reduced to $< 5\%$.

WO_x ReRAM and 2T-NOR also have a normalized standard deviation of $< 5\%$. The WO_x ReRAM can separate the currents into eight non-overlapping levels, while the 2T-NOR eases current variation by a selection transistor, which represses the normalized standard deviation. Among all the simulations in Fig. 3.24, we observe that the cell normalized current variation needs to be smaller than 5% to achieve success rates without significant drops.

3.4.2.2 Cost of Calibrating Current Variation

Fig. 3.25 shows the requirement of the membrane capacitance for different numbers of replication. Because the Ising model can tolerate the current variation of all memory technologies without any drop in the success rate, we only show the results for TSP and Sudoku. In the simulation, the leak current is set to zero. The memory technique of 2T-NOR, WO_x, and FeFET-High with normalized standard deviation $< 10\%$ hold success rates of $> 80\%$ without replication of memory cells for solving both TSP($K < 12750$) and Sudoku. The FeFET-Normal with normalized standard deviation $< 15\%$ can also perform



with a success rate of 90% for the application of TSP ($K < 12750$) without any replica, and it only requires 1pF of capacitance. However, its success rate of solving Sudoku is $< 62\%$. With weight cells replicated by 16, the success rate increases to 96%, but the capacitance requirement is $10pF$, which is 10 times the requirement of no replication. Both memory techniques, Pt/HfOx/Ti/Al and FeFET-Low, have success rates $< 65\%$ without replica. To increase the success rate of solving Sudoku, cells are replicated 16 times to reduce the normalized standard deviation to 8%. As a result, the success rate rises to 80%, but the required capacitance is 100pF and 1nF for FeFET-Low and Pt/HfOx/Ti/Al, respectively. Those required capacitance sizes are > 10 times the required capacitance of no replica.

These simulation results show the trend that those memory technologies with a normalized standard deviation of $> 30\%$ limit the success rate. They require more replicas to increase the success rate, but they also require a sufficient size of membrane capacitance.

Due to the law of large numbers, the replication can reduce the normalized standard deviation. However, this is always at the cost of higher capacitance, described in Eq. 3.14. The replication increases the cell currents flowing into the membrane capacitor, which reduces the precision of the MAC result. Consequently, the capacitance needs to be increased to reduce the precision loss from duplicated cell currents. If we consider the limitation of the membrane capacitor, the cell currents need to have a small normalized standard deviation to preserve the success rate. 2T-NOR, WOx, and FeFET-High having normalized standard deviations of $< 7\%$ are required for solving CSP with high success rates.

3.4.2.3 Impact of Leak Current

The impact of the leak current is shown in Fig. 3.27. In this simulation, the memory cell is replicated to reduce the normalized standard deviation of $< 15\%$ and of $< 5\%$ for TSP and Sudoku, respectively. Moreover, a set of membrane capacitances is chosen to make N_{cp} equal to 5, 50, 500, or 5000. The success rate when considering the OFF-state leakage current of 2T-NOR, FeFET-High, and Pt/HfOx/Ti/Al is similar to the success

rate without leak current. In the case of $N_{cp} = 50$, the success rate of the three memory technologies with ON-OFF ratio > 1000 reach success rates of $> 60\%$. FeFET Normal with ON-OFF ratio = 570 requires N_{cp} of 500 to prevent success rate drop while solving TSP. Moreover, FeFET-Low and WOx with ON-OFF ratio < 100 fail to solve TSP and Sudoku even though the N_{cp} is increased to 500.

The normalized standard deviation of the MAC result is proportional to $N_{cp}^{-1/2}$ as shown in Eq. 3.14. The leak current causes the reduction of effective capacitance as described in Eq. 3.23. The reduction of effective capacitance reduces N_{cp} and causes an increase in normalized standard deviation. We observe that the effect of leakage current can be neglected when the ON-OFF ratio > 1000 . On the contrary, the leakages of those memory technologies with the ON-OFF ratio of < 100 decay the N_{cp} dramatically. Consequently, the normalized standard deviation increases when N_{cp} is reduced, which drops the success rate. The simulation results show that more than 100x of N_{cp} is required to preserve the success rate compared to the case with no leakage current when the ON-OFF ratio < 100 .

3.4.2.4 Impact of Temperature

We select the size of the membrane capacitance such that the success rate is $> 80\%$ at the temperature of 300K. The impact of temperature from 273K to 358K for each memory technology is shown in Fig. 3.28. For both memory technologies, WOx and FeFET, the success rate drops dramatically with an increased temperature of $> 338K$. For 2T-NOR devices, the success rate stays above 60% while solving the Ising model and TSP. However, it fails to solve Sudoku when the temperature increases above 338K.

The TiN/HfO2/Ti/TiN devices can reach a success rate of $> 66\%$ for all the problems except the TSP at 358K. The memory technologies WOx and FeFET-High both drop the success rate and the ON-OFF ratio with increasing temperature, as shown in Fig. 3.29.

In Fig. 3.27, the ON-OFF ratio < 1000 requires sufficient capacitance to preserve the

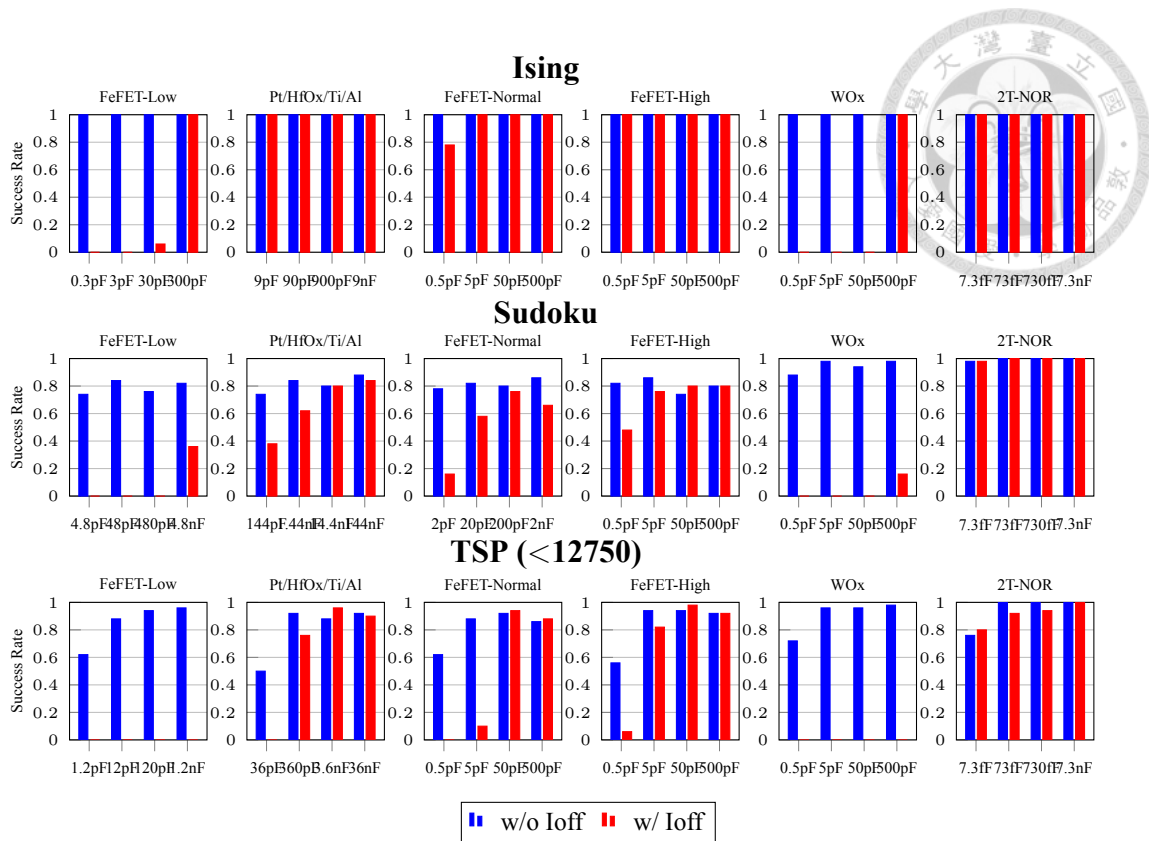
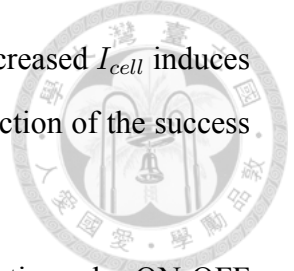


Figure 3.27: Impact of OFF-state Leakage on the Requirement of C_{mem} . The capacitance is selected such that $N_{CP} = [5, 50, 500, 5000]$. The impact of OFF-state leakage can be neglected in Pt/HfOx/Ti/Al, FeFET-High, and 2T NOR Flash with an ON-OFF ratio > 1000 .

success rate. However, when the temperature increases from 300K to 338K, the ON-OFF ratio of FeFET-High drops from 1.6k to 90. In selecting the required capacitance for 300K, the capacitance is not enough for 338K, and the success rate drops dramatically. Similar phenomena can also be observed for WOx. With the temperature rising from 300K to 338K, the ON-OFF ratio drops from 22 to 13, which in turn drops the success rate from 80% to 60%. On the contrary, in the case of 2T-NOR Flash with an ON-OFF ratio above 10^5 , OFF-state leakage is insignificant. The success rate reaches above 80% even though the ON-OFF ratio of 273K drops to 0.5 times of 300K. Moreover, TiN/HfOx/Ti/TiN with an ON-OFF ratio changing smaller than 17% with temperature $< 338K$ preserves the success rate $> 76\%$.

The ON-state current drift also decreases the success rate. The ON-state current of 2T-NOR increases dramatically, as shown in Fig. 3.23. The ON-state current of 338K and

358K increases to 2x and 3x compared to the current at 300K. The increased I_{cell} induces low computing precision as shown in Eq. 3.14, and causes 20% reduction of the success rate compared to 300K.



Based on the observation that both the reduction of ON-OFF ratio under ON-OFF ratio < 1000 and ON-state current drift decay the succeed rate of solving TSP, we design a *Robust Index* to indicate the reliability under different temperatures, shown in Eq. 3.29.

$$Robust\ Index(T) = \frac{\frac{R(T)}{R(300)} - 1}{\frac{R(T)}{1000} + 1} + \log\left(\sqrt{\frac{I_{ON}(300)}{I_{ON}(T)}}\right) \quad (3.29)$$

$$R(T) = \frac{I_{ON}(T)}{I_{OFF}(T)} \quad (3.30)$$

The Robust Index is calculated using three components: $\frac{R(T)}{R(300)}$, $\frac{R(T)}{1000}$, and $\frac{I_{ON}(300)}{I_{ON}(T)}$. If the ON-OFF ratio of a certain T is smaller than the default temperature of 300K, the value $\frac{R(T)}{R(300)} - 1$ becomes negative. However, this term can be decreased by $\frac{R(T)}{1000} + 1$. When the ON-OFF ratio of T and $R(T)$ is $\gg 1000$, the term of $(\frac{R(T)}{R(300)} - 1) / (\frac{R(T)}{1000} + 1)$ converges to 0. Another factor is precision loss from increased cell current. Because the normalized standard deviation is increased by $\sqrt{I_{cell}}$ as shown in Eq. 3.14, we utilize the $\sqrt{\frac{I_{ON}(300)}{I_{ON}(T)}}$, representing the computing precision. The cell current at the temperature of T is larger than the current at the default T , i.e., 300K, which increases the normalized standard deviation, and the term of $\frac{I_{ON}(300)}{I_{ON}(T)} < 1$ which is negative after application of the logarithm. The *Robust Index* of each memory technology under different temperature states and their predicative are shown in Fig. 3.29. The result shows that when the *Robust Index* has a score > 0 , success rates of 80% are achieved. The result also shows that the *Robust Index* score < -0.14 indicates a success rate drop to $< 76\%$.

Among all results, the memory cell with a large ON-OFF ratio can tolerate the ON-OFF ratio drop caused by temperature changes. A stable ON-state current is also required to preserve the success rate. A temperature that can be tolerated satisfies the condition $Robust\ Index(T) > 0$. 2T-NOR provides a high ON-OFF ratio from 273K to 358K, but

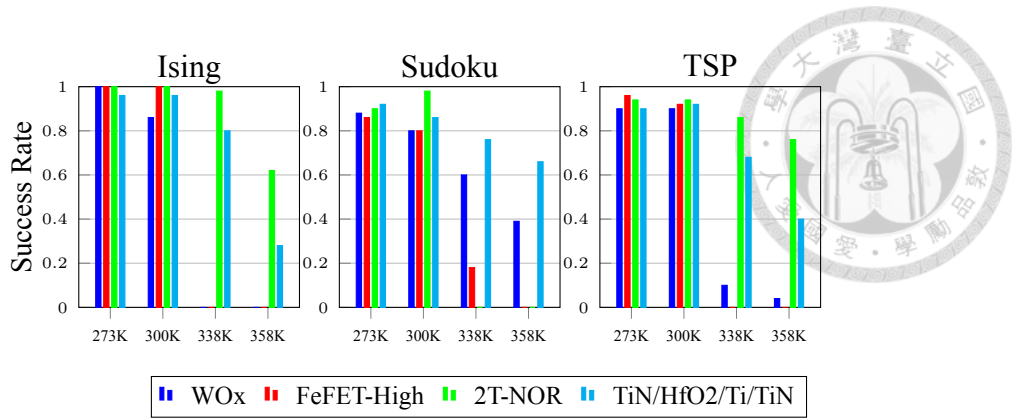


Figure 3.28: Impact of Temperature on the Success Rate of Ising, Sudoku, and TSP. WOx and FeFET-High have success rate drops as temperature increases, while 2R-NOR and TiN/HfO2/Ti/TiN are not affected much.

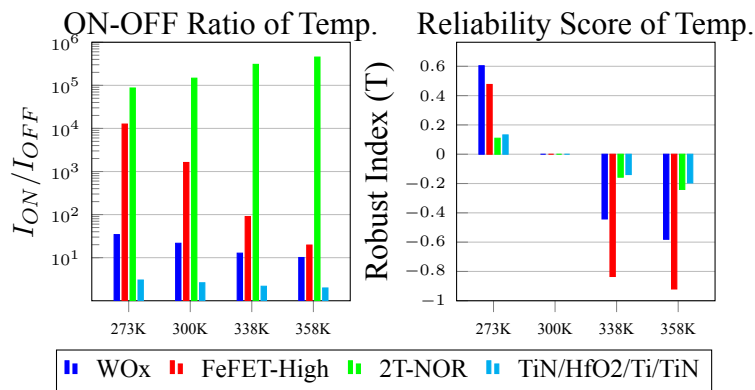


Figure 3.29: ON-OFF Ratio and Robust Index for Each Memory Technology and Temperature. The ON-OFF ratio reduction drops the robust index of WOx and FeFET-High, while the cell current drift drops the index of 2T-NOR.

the drift of the ON-state current drops the success rate. For WOx and FeFET, the success rate drops when the ON-OFF ratio dramatically drops below 1000. The TiN/HfO2/Ti/TiN provides a stable ON-OFF ratio and ON-state current, but its low ON-OFF ratio requires large capacitance inducing high energy costs. We conclude that the memory technologies examined here cannot tolerate temperature changes under reasonable capacitor size.

We select the size of membrane capacitance to maintain a success rate of over 80% for each memory device, and the comparison of their energy consumption is shown in Fig. 3.31. The results indicate that the energy consumption from processing input spikes through 2T-NOR, FeFET-Normal, and FeFET-High synapses array with IF neural circuit are less than 5% of the energy consumed by the algorithmic circuit. The energy con-

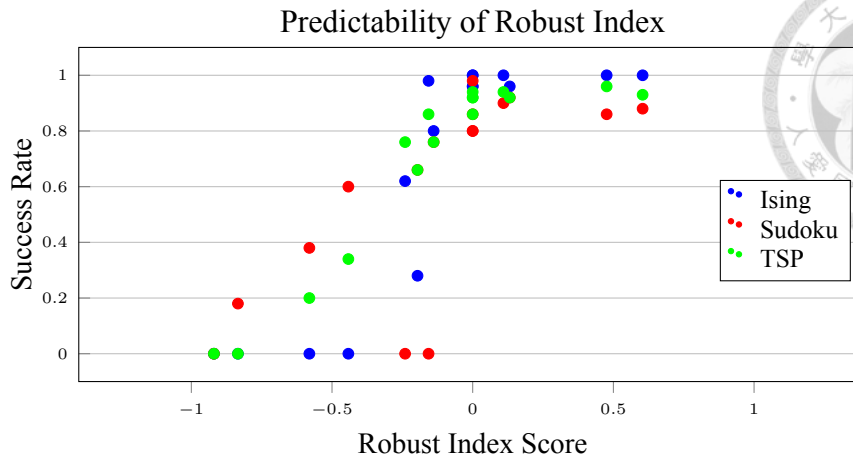


Figure 3.30: Predictability of Robust Index for Each Problem. The robust index score's positive value can sustain the success rate while the index score < -0.2 drops the success rate to ≤ 0.6 .

sumption of WO_x and FeFET-Low is comparable to that of the algorithmic circuit. The energy consumption of the TiN/HfO₂/Ti/TiN synapse array dominates the overall energy consumption. The 2T-NOR and FeFET-High devices exhibit tolerable current noise and ON-OFF ratios > 1000 , resulting in lower requirements for membrane capacitance and less energy consumption than others. The FeFET-Normal, with a finite ON-OFF ratio, requires twice the membrane capacitance than FeFET-High despite having a similar ON-state current. Additionally, for the application of solving Sudoku, four times replication is required to mitigate the impact of current variation compared to solving the Traveling Salesman Problem (TSP) when using FeFET-Normal and Pt/HfO_x/Ti/Al as the synapse device. Consequently, the energy consumption required to solve Sudoku is four times higher than that for solving TSP. WO_x and FeFET-Low, with an ON-OFF ratio smaller than 100, dramatically increase the requirement for membrane capacitance, which is fifty times larger than FeFET-Normal. The TiN/HfO₂/Ti/TiN, with an ON-OFF ratio of less than 5, also significantly increases the size requirement of the membrane capacitance. As a result, the energy consumption of the synapse array and integrate-and-fire neuron dominate the overall energy consumption. In conclusion, the results demonstrate that replicating memory cells increases the required energy to solve the problem linearly. However, a finite ON-OFF ratio leads to a dramatic increase in energy requirements. An ON-OFF

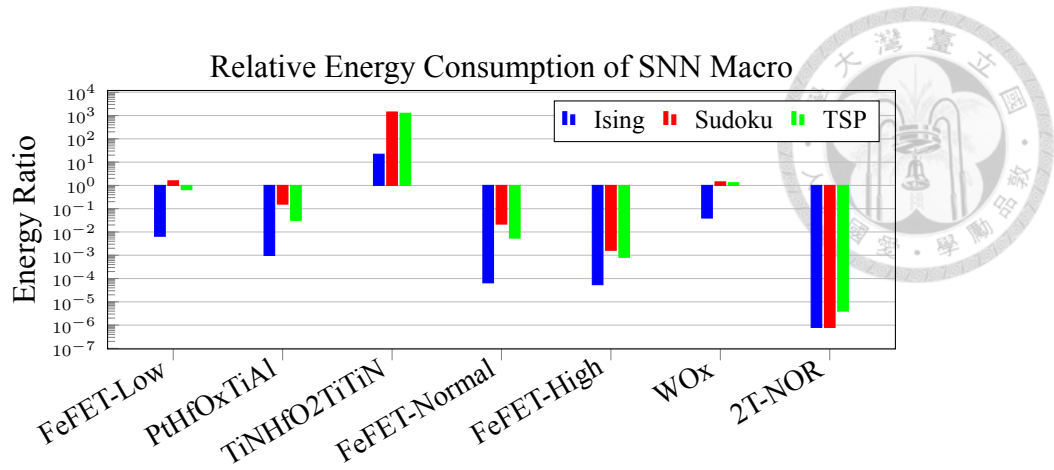


Figure 3.31: Energy Consumption of an SNN Macro per Iteration. The y-axis is the energy ratio of the synapse array and the peripheral algorithmic circuit. The energy consumption of the algorithmic circuit for the SNN macro is independent of the memory device, which is fixed to 1.37mJ per crossbar per iteration.

ratio > 1000 is recommended to minimize energy costs from leakage current. Devices with tolerable noise can prevent additional energy losses.



Chapter 4

A Neuromorphic Spiking Signal Processor with Advanced Memory Technology

4.1 Background

4.1.1 SNN Annealing Machine

The optimization problems are transformed into a loss function, and the annealing machine, or Ising machine, is a device to find the state of the variables to make the output of the function approach to the minimal loss score [47, 118, 119]. The SNN annealing machine leverages the spiking neural network's embedded stochastic behavior and convergence characteristic to find the optimization-approaching solution for NP-hard optimization problems [47]. The following sections will introduce the dynamical system of a spiking neural network, the formulation of an optimization problem, and how to utilize the spiking neural model to solve the optimization problem.

4.1.1.1 Recurrent-structured SNN

The recurrent-structured SNN is constructed by spiking neurons connected by the closed-loop network topology. This structure creates a dynamical system in which the

neural activity of each neuron is iteratively updated following the rule that mimics neuron cells [120–125]. The neural stochastic dynamics is described as eq. (4.1) to (4.5).

$$V_j^{ch}[t + 1] = V_j^{ch}[t] + \sum_i X_i[t] W_{i,j}^{ch} - V_{th} \delta[1 - E_j^{ch}[t]] \quad (4.1)$$

$$E_j^{ch}[t + 1] = \begin{cases} 1, & \text{if } V_j^{ch}[t] + \sum_i X_i[t] W_{i,j}^{ch} \geq V_{th} \\ 0, & \text{if } V_j^{ch}[t] + \sum_i X_i[t] W_{i,j}^{ch} < V_{th} \end{cases} \quad (4.2)$$

$$\Delta P_j[t] = \eta \delta[1 - E_j^e[t]] - \eta \delta[1 - E_j^h[t]] \quad (4.3)$$

$$P_j[t + 1] = \text{Rec.}(P_j[t] + \Delta P_j[t]) \quad (4.4)$$

$$X_j[t] \sim \text{Bernoulli}(P_j[t]) \quad (4.5)$$

Each neuron- j contains the internal variable of membrane voltage (V_j^{ch}), which contains the channel of excitation (V_j^e) and inhibition (V_j^h). These membrane voltages are charged by the current of $\sum_i X_i[t] W_{i,j}^{ch}$, which is the sum of weighted input spikes. The $\sum_i X_i[t] W_{i,j}^s$ is known as a synapse operation that can be interpreted as vector-matrix multiplication with binary input. Once the membrane voltage exceeds the threshold voltage, V_{th} , the membrane voltage resets to 0 and updates the neural activities, P_j , of the neuron- j . The excitation and the inhibition of the neural activities are based on the channel of the membrane voltage as eq. (4.4) in which the *Rec.* is the rectifier unit to ensure the updated neural activities are normalized between 0 and 1. At each cycle, the spike rates of the neuron- j transmitted to others are based on the neural activities as eq. (4.5). The network structure is a direct graph in which the connection strength from neuron- i to neuron- j is represented by $W_{i,j}^s$. The network to solve the optimization problem is recurrent structure [126–129]. This recurrence implies that the future input spikes of a neuron depend on

its present state. Evolving neural states without using up-to-date neurons will negatively impact solving optimization problems as introduced in Sec. 4.2.1.2.

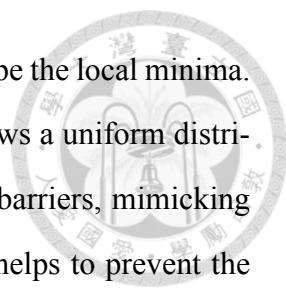


4.1.1.2 Formulation of Optimization Problems

The optimization problems involve finding the state vector \mathbf{X} that minimizes the loss function. The general form for most of the optimization problems is formulated as $Loss(\mathbf{X}) = \alpha \mathbf{X}^T \mathbf{W}_s \mathbf{X} + \beta \mathbf{X}^T \mathbf{S} + \gamma |\mathbf{X}|_1$ where \mathbf{W}_s and \mathbf{S} are related to the applications. One example is the combination optimization problem, which aims to find variable options that satisfy specific rules. These problems can be formulated as quadratic unconstrained binary optimization (QUBO) [130–132], where the objective is to find \mathbf{X} that minimizes $-\mathbf{X}^T \mathbf{W}_s \mathbf{X}$, with \mathbf{W}_s encodes the relation of state variables in the \mathbf{X} . Another example is LASSO, a common linear programming problem that seeks a sparse representation \mathbf{X} using a dictionary \mathbf{W}_{dict} such that $\mathbf{W}_{dict} \mathbf{X}$ approximates a given sample \mathbf{Y} . The objective of LASSO [4, 133–136] is to find \mathbf{X} that minimizes $\mathbf{X}^T \mathbf{W}_{dict}^T \mathbf{W}_{dict} \mathbf{X} - 2 \mathbf{X}^T \mathbf{W}_{dict} \mathbf{Y} + |\mathbf{X}|_1$ that also meets the quadratic form the Loss function.

4.1.1.3 Solve Optimization Problem by SNN

The SNN annealer processes the flow of sampling eq. (4.5), calculate gradients eq. (4.1), and update states eq. (4.2) and eq. (4.4). This process drives the spiking neuron to the state with less loss. The spike generation units sample the state of \mathbf{X} from the latent variable of \mathbf{P} . The spiking neuron then calculates and integrates the minus gradient of the loss function, $-\nabla_{\mathbf{X}} Loss(\mathbf{X}) = -\alpha \mathbf{W}_s \mathbf{X} - \beta \mathbf{S} - \gamma \mathbf{sign}(\mathbf{X})$, where the $\mathbf{W}_s \mathbf{X}$ is processed in the synaptic array and other scalar terms are implemented in the rectifier unit. Then, the minus gradient updates the latent variable of \mathbf{P} to change the distribution of \mathbf{X} . This process is iterative until the probability of the observed \mathbf{X} converges. The feature of the SNN framework is that all the states of \mathbf{X} are equally distributed in the beginning. During the annealing process, the probability of the state \mathbf{X} is converged to the state with less loss. Different from gradient decent [137, 138], the SNN annealing machine calculates



the gradient from the sampled spikes that provide the variance to escape the local minima. Additionally, the initialization of \mathbf{P} can be configured so that \mathbf{X} follows a uniform distribution, allowing the sampled state \mathbf{X} to flexibly surmount potential barriers, mimicking the behavior of the quantum tunneling effect [139]. This approach helps to prevent the final state from being trapped by the high-wall convex where the initial state is located.

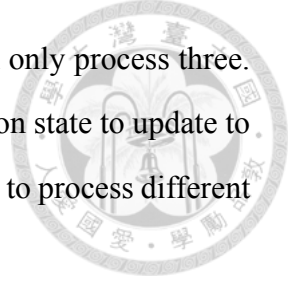
4.1.2 Spike Signal Processor

Different from the neuron-distributed spiking neural network accelerators such as TrueNorth [2] and Spinnaker [140], which primarily focus on scalability, recent advancements like SpinalFlow [13] and PTB [14] are addressing the challenge of evolving large spiking neural models under cost constraints. These newer approaches consider the reusability of spike signal processors, aiming to efficiently evaluate spiking neural networks beyond the capabilities of existing spiking neuron circuits. However, the evolution of neural states under finite neuron circuits necessitates sharing physical spiking processing devices. The associated cost of sharing these physical devices lies in the switching process, which includes loading synapse weights from external memory to scratchpad, storing current state variables, and loading variables that require processing.

Two design strategies have been proposed to mitigate the impact of moving data from off-chip memory: the tick-batching technique and computing in memory. These approaches aim to optimize the processing efficiency and minimize the challenges associated with data movement, allowing for the evolution of spiking neural networks more cost-effectively.

4.1.2.1 Tick-batching

The tick-batching technique [13, 14] aims to amortize the switching cost by reusing previously loaded weights and neural activities to evolve successive states, regardless of whether input neuron activities are up-to-date during the process. Fig. 4.1 shows the ex-



ample of processing four neurons under one processing core that can only process three. The example shows if the spiking processor priority evolves the neuron state to update to the same time points, the spiking signal processor frequently switches to process different neurons, and the frequent setting up drops the processing throughput.

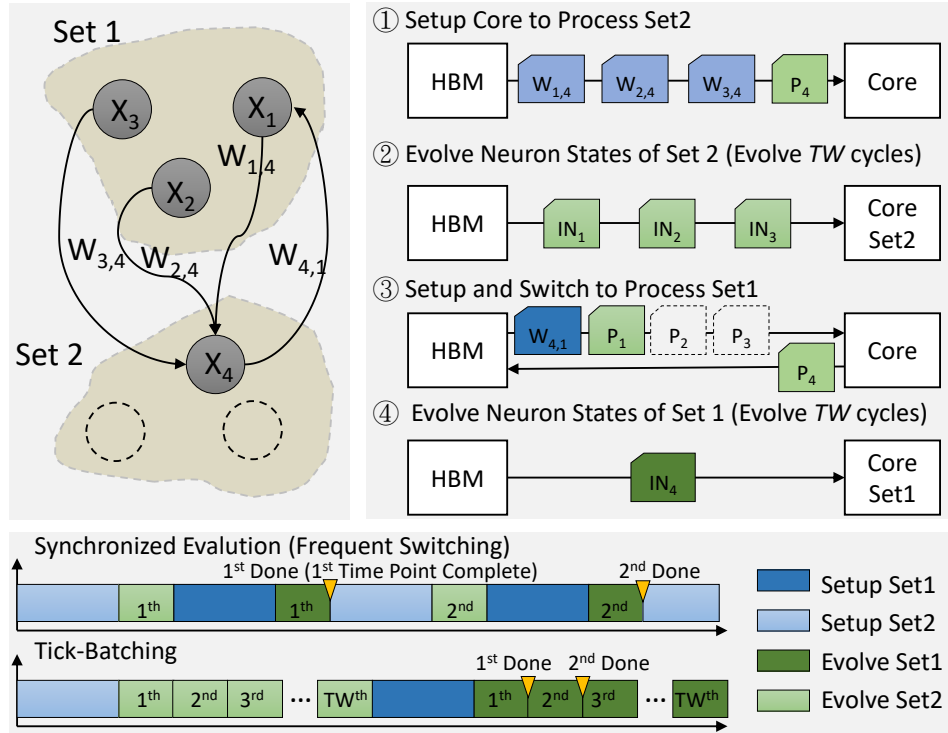


Figure 4.1: Switching Cost of a Spiking Processing Core. This example shows a processing core switching the neurons to be processed. The synchronized evolution of neuron states leads to the frequent switching of neurons, while the tick-batching technology amortizes the switching cost.

To amortize setup latency, which includes moving weights and neural activities, loaded parameters are reused throughout a time window. The window size is called tick-batching size, denoted as TW . To enable amortization, equations (4.4) and (4.2) in the neural dynamics are modified to equations (4.6) and (4.8). This tick-batching technique remodels the spiking neural dynamics, balancing the accuracy loss from non-synchronization and evolving performance by selecting the size of TW [15].

$$P_j[nTW + k] = P_j[nTW] \text{ for } k = 1, 2, 3, \dots, TW - 1 \quad (4.6)$$

$$\Delta T[n] = \sum_{t=0}^{TW-1} \Delta P_j[nTW + t] \quad (4.7)$$

$$P_j[(n + 1)TW] = Rec.(P_j[nTW] + \eta\Delta T[n])$$

(4.8)



4.1.2.2 Digital Synapse Operation

The primary function of a spiking neural network’s dynamical system is the synapse operation, $\sum_i X_i[t]W_{i,j}$. SpinalFlow [13] optimizes a synapse operation accelerator specifically for handling highly sparse input spike events. This design batches input spikes and processes them sequentially. When a spike is received from the input spike buffer, the corresponding weights are retrieved from the SRAM adjacent to the neuron circuit to execute the synapse operation, as illustrated in Fig.4.2(a). In contrast, the PTB architecture [14] employs a systolic array to facilitate the synapse operation, which is suitable for dense spike trains, as depicted in Fig.4.2(b). This architecture also includes a spike packing unit that densifies the input spikes, enhancing utility and increasing processing throughput.

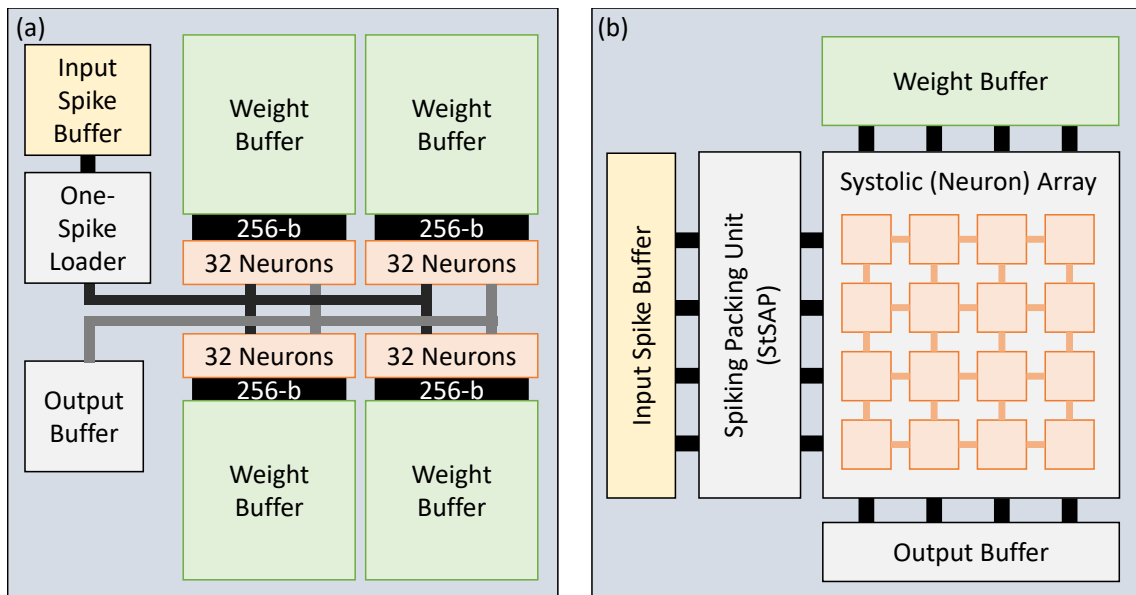


Figure 4.2: Prior Architectures of Spiking Processing Core of (a) SpinalFlow, and (b) PTB.

4.1.2.3 Synapse Operation in 3D-NOR Flash

An alternative method to mitigate the impact of weight movement is to perform synaptic operations directly at the weight’s location. The synaptic operation, given by

$\sum_i X_i[t]W_{i,j}$, can be implemented using a non-volatile synaptic array [8, 45, 46]. To address the capacity limitations that spiking signal processors face due to the restricted density of 2D non-volatile synaptic arrays, a novel approach involves using a 3D-NOR Flash synaptic array for matrix multiplication [141]. This device employs a vertically stacked synaptic array, which offers a significantly higher bit density for synaptic operations than 2D arrays. Binary input spikes are fed into a specific layer's word lines (WLs) to access the corresponding weights. The current in the source line (SL) reflects the outcome of the synaptic operation and is directed to the neuron located adjacent to the stacked array, as depicted in Fig. 4.3.

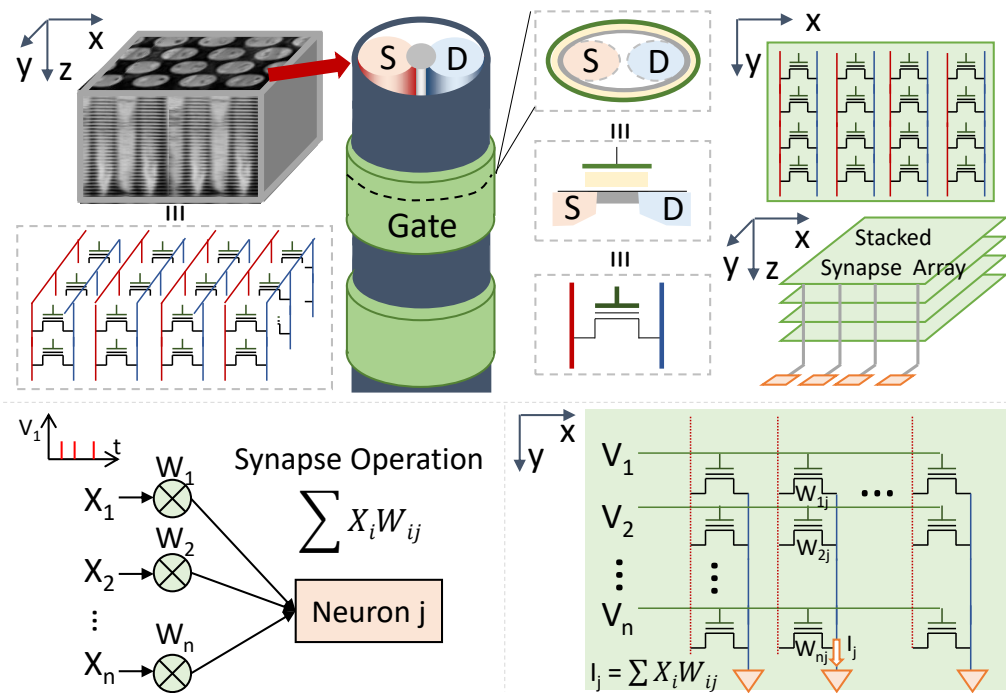
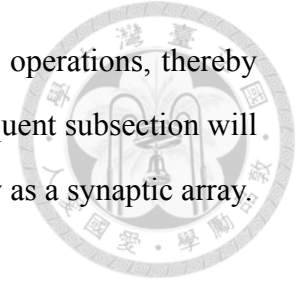


Figure 4.3: Structure of 3D-NOR Flash Synaptic Array. The core of a 3D-NOR Flash comprises vertical cylinders, with both the source and the drain located within each cylinder. A binary input vector is introduced to a designated layer's word line (WL). The ensuing current on the source line (SL) represents the synaptic operation's outcome, guided by Kirchhoff's current laws.

4.2 Motivation

This paper introduces a spiking neural network accelerator, which functions as a platform for an annealing machine aimed at solving optimization problems. We leverage

3D-NOR Flash Memory [141] to enhance the efficiency of synapse operations, thereby accelerating the evolution of the spiking neural network. The subsequent subsection will explore the benefits and challenges of using 3D-NOR Flash Memory as a synaptic array.



4.2.1 Opportunity of 3D-NOR Flash Synaptic Array for Solving Optimization Problems

The baseline architecture referred to PTB [14] is composed of spiking neural network processing cores interconnected via a high-bandwidth memory (HBM) with a bandwidth of 64GB/s [142]. Each core includes a digital-based 16x16 systolic array operated at 200MHz and 2KB SRAM for buffering weights. This architecture, depicted in Fig. 4.5, faces bottlenecks stemming from either the synapse operation or weight movement.

4.2.1.1 Computing-dominated Case

One scenario is that the capacity of spiking neuron circuits exceeds the model size, and neuron circuits can evolve neuron states without stalling for loading weight. The other scenario is that the switching step takes place, but the tick-batching technique amortizes the switching cost. Once the amortized switching time is far shorter than the spiking processing time, the processing throughput approaches the scenario without considering weight movement as $TW=256$ and $core=16$ in Fig. 4.5. In the above two scenarios, synapse operations dominate most of the processing, and the throughput of the spiking processing constrains the performance bottleneck.

4.2.1.2 Switching-dominated Case

The tick-batching technique is effective for amortizing switching costs. However, the tick-batching technique results in the asynchronous evolution of neuron states, which can lead to **diminished solution quality in solving optimization problems** as introduced in Fig. 4.7 and Fig. 4.8. When considering the limited tick-batching size, bus contention for

loading weights becomes the bottleneck, as depicted in Fig. 4.5 with $TW=64$ and $core=16$.

To alleviate the bottleneck caused by weight movement in switching-dominated cases and provide superior dense vector-matrix multiplication (VMM) processing throughput for computing-dominated scenarios, the 3D-NOR Flash synaptic array [141] is utilized to expedite synapse operations. The 3D-NOR Flash synaptic array takes advantage of the intrinsic in-situ processing of weights to eliminate weight movement while ensuring competitive spike processing throughput for efficient computation, as shown in Fig. 4.5. Compared to digital-based accelerators, the 3D NOR Flash offers double the performance-to-area efficiency and a bit cost 60 times lower than a digital-based accelerator using 28nm technology, as shown in Fig. 4.6. Moreover, compared to other memory technologies, the NOR Flash cell with a high ON-OFF ratio, stuck-at-fault free, and fine-tuned cell current provides reliable operation with less calibration cost [47, 48].

The 3D-NOR Flash has an access latency of up to a hundred nanoseconds and, therefore, requires hundreds of input granularity sizes to achieve superior processing throughput than prior digital-based synapse processing cores, as referenced in [13]. However, the challenge is fully utilizing the input granularity to prevent processing throughput loss. In the following, we will delve deeper into the challenges of input utilization rate when using 3D-NOR Flash memory as a synaptic array.

4.2.2 Challenge of applying 3D-NOR Flash as Synaptic Array

The 3D-NOR Flash synaptic array delivers superior dense VMM processing throughput. However, its large input granularity poses a challenge in fully utilizing computational parallelism. Three issues that drop the computational parallelism, including (1) spatial sparsity of input spike, (2) small input degree of network connectivity, and (3) ineffective pruning of weak neurons, are addressed in this thesis.

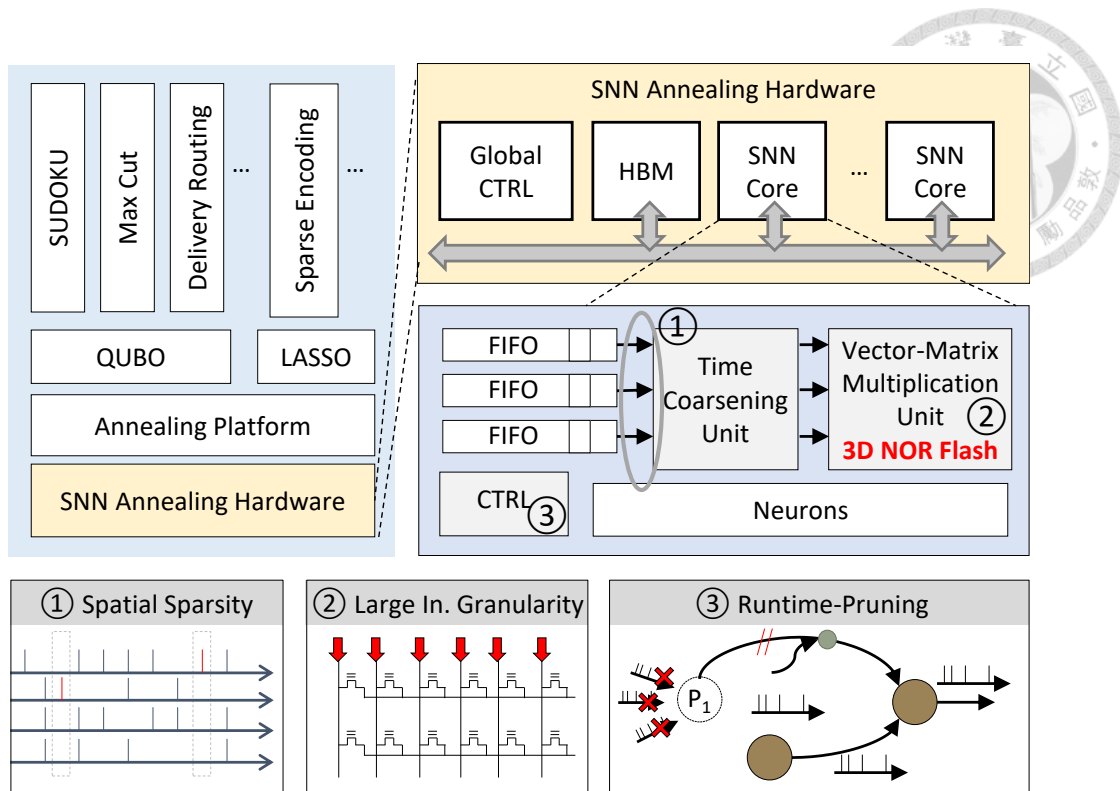


Figure 4.4: Overall Scope of Neureka. This thesis aims to provide a spiking signal processor serving the annealing platform for solving optimization problems. The 3D-NOR Flash memory is the synaptic array providing weight movement-free, low bit cost, and high processing throughput per area. To further optimize the processing throughput, we address the issues leading to processing throughput loss, including (1) sparse spike inputs, (2) small input degrees of neurons, and (3) ineffective pruning.

4.2.2.1 Sparse Input Spike

At each moment, the inputs of a neuron are loaded as a binary vector and transmitted to the synaptic array to process the synapse operation. However, if the input vector is sparse, few input spikes occupy the synaptic array's input, leading to the waste of spike processing throughput.

4.2.2.2 Small Input Degree

The input degree is the number of inward connections to a neuron. Once the input degree of neurons is smaller than the synaptic array input pins, the inputs are also not fully utilized.

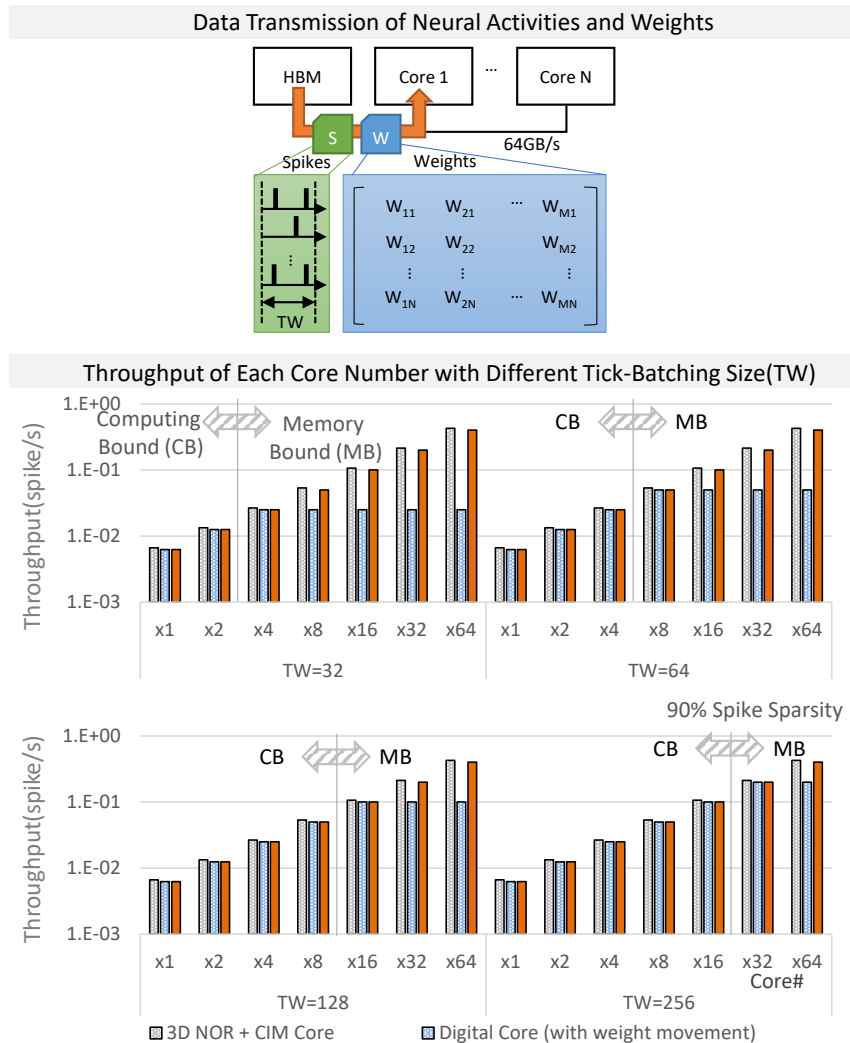


Figure 4.5: Impacts of Bus contention on Digital-based Spiking Processor Cores under Different Tick-batching Sizes (TW). When the digital-based solution considers the HBM's bus bandwidth and restricted internal cache, the performance is dropped by an increased core number. With 3D-NOR Flash Memory, the bus contention from moving weight is eliminated. The architecture synergy with 3D-NOR Flash synaptic array also provides a competitive spike processing throughput compared to digital-based accelerators

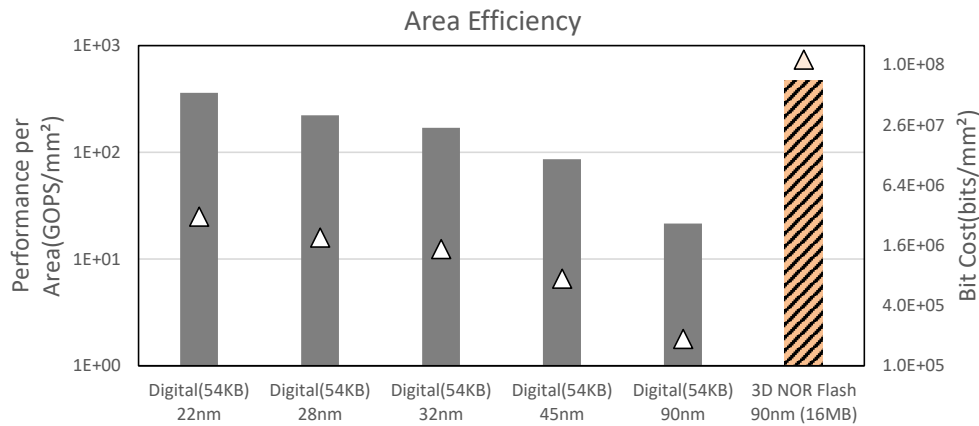


Figure 4.6: Area Efficiency Comparison between the 3D NOR Flash-based Accelerator and a Digital Accelerator. The 3D NOR Flash-based accelerator provides synaptic operation throughput comparable to a 28nm digital accelerator equipped with a 16x16 systolic array and includes a 54KB SRAM while achieving a bit cost that is an order of magnitude lower than that of digital's.

4.2.2.3 Ineffective Pruning

Two types of pruning networks to accelerate the processing of spiking neural network has been proposed. One is pruning weak weights [143–149] while the other is pruning weak neurons [150–152]. Weight pruning can not be applied to the network for solving optimization problems. The reason is that the magnitude of weight corresponds to the constraints for the target problems, and the pruning of weight would lead to the weight not meeting the constraints, resulting in the found solution not meeting the constraint. The other pruning neuron technique is to drop the neurons that hardly generate spikes. However, the pruning of weak neurons is non-structured, and the synaptic array hardly benefits from the non-structure pruning. As a result, although the neuron sparsity is increased, the required synaptic array access time is not reduced.

We propose three designs to address the above three challenges, including a time coarsening unit, granularity-reduced synaptic array, and runtime-pruning methodology as Fig. 4.4.

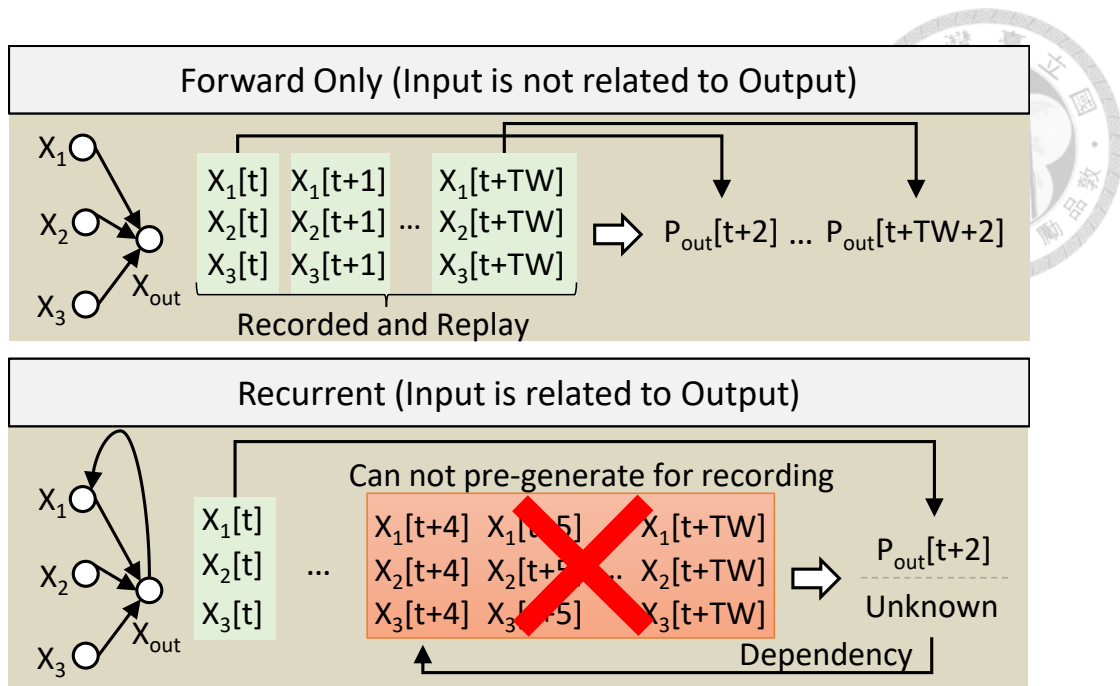


Figure 4.7: Computing Error from the Tick-batching Technique for Recurrent Network Structure. The tick-batching technique can be applied to feed-forward network structure without causing computing error. However, utilizing the tick-batching technique for the recurrent structure network will lead to a computing error due to the output and input dependency and utilization of non-up-to-date input spikes.

4.3 Proposed Architecture-Neureka

The overall architecture of the proposed design, as shown in Fig. 4.9, of which a 3D-NOR Flash memory accelerates the synaptic operation. To address the challenges associated with using the 3D-NOR Flash as a synaptic array, we introduce a time coarsening unit, granularity-reduced synapse 3D NOR array, and run-time pruning technique to enhance processing throughput.

4.3.1 Time Coarsening Unit

The proposed time coarsening unit aims to reduce the access of the synaptic array. This reduction is achieved by identifying and bypassing time points without input spikes. However, naturally-generated spike trains rarely contain a time point without spikes that can be bypassed. A time coarsening unit has been introduced to increase the occurrence of such bypassable time points without affecting the timing of output spikes, as illustrated

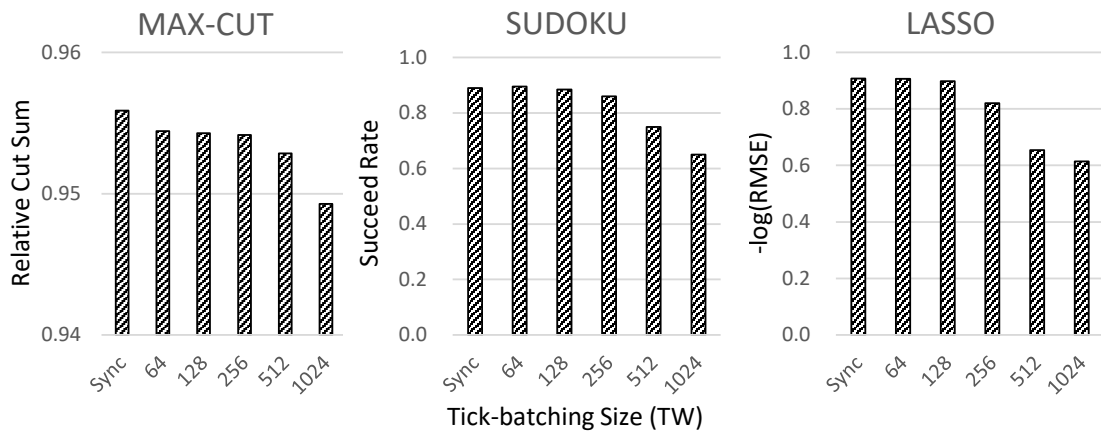


Figure 4.8: Impact of Tick-batching Technique on Solving Quality of Optimization Problems. The increasing tick-batching size (TW) will aggravate the asynchronous neuron states that the input state is not up-to-date, and the asynchronous input state will lead to a drop in the quality of solving optimization problems.

in Fig. 4.10. The proposed method accesses spikes from the latest two-time points and determines if all input trains contain no more than one spike in the coarsened time point. If this condition is met for each input train, the two time points are merged into one, allowing the other to remain empty for bypassing.

The detailed design is introduced in Fig.4.11. The spikes at two time points are accessed to re-distribute the input spike per cycle. Owing to the two-time points coarsening, whether the spike count is two can be determined simply by an AND operation. Moreover, the disturbance of input spikes is bounded by a one-time point. Each input channel i contains four registers within the FIFO: $t_0[i]$, $t_1[i]$, $t_2[i]$, and $t_3[i]$, the output register $Out[i]$, and the remaining spike register $R[i]$. If all $R[i]$ registers are empty, spikes in $t_0[i]$ and $t_1[i]$ are accessed. Otherwise, spikes in $t_0[i]$ and $R[i]$ are accessed. If at least one spike exists in the two accessed registers for channel i , the corresponding $Out[i]$ is set to 1. The AND result of the two accessed registers is considered the residual spike, which is placed back into $R[i]$ or $t_0[i]$ according to the rules shown in Fig.4.11. Three specific register-setting rules are proposed to simplify the FIFO by supporting only two element-wise shifts. If all $R[i]$ s are empty, the coarsening unit deposits the residual spike into $R[i]$. If there are spikes in $t_1[i]$ without any residual spikes, the coarsening unit also transfers $t_1[i]$ to $R[i]$. If there are spikes in $t_1[i]$ with residual spikes, the coarsening unit deposits the remaining

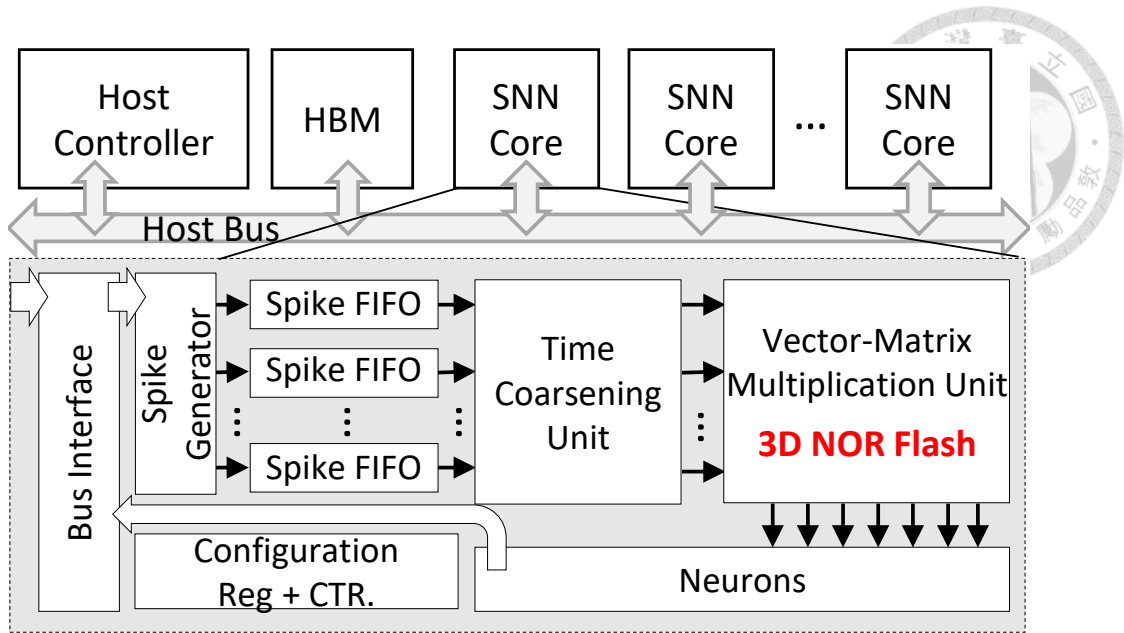


Figure 4.9: Overall Architecture of Neureka. The architecture includes SNN cores and a HBM. In each SNN core, the vector-matrix multiplication is implemented by the 3D NOR Flash memory.

spikes into $t_0[i]$ and cleans $R[i]$. These rules ensure that the FIFO requires shifting by two elements or none. Furthermore, this mechanism is implemented using a few logical operations, as shown in Fig.4.11.

The effectiveness of the time coarsening unit is demonstrated in Fig.4.12. It reveals that the coarsening unit successfully creates time points without spikes, which can be bypassed to reduce the number of synaptic array operations. The impact analysis of the timing coarsening unit, tested from random weight, input, and dynamic parameters, is also presented in Fig. 4.12. It indicates that the disturbance to output spike timing is negligible when the time coarsening unit is used to redistribute input spike trains.

4.3.2 Reduction of Input Granularity of Synaptic Array

To prevent the spike processing throughput loss from neurons with a small number of inputs, this thesis leverages temporal-wise input spikes to increase the input utilization of the synaptic array, as shown in Fig. 4.13. To leverage the sub-sequence input spikes, the bit-line (BL) inputs of the synaptic array are partitioned into several groups. The subsequent input spikes are as a vector provided to adjacent groups to increase the usage

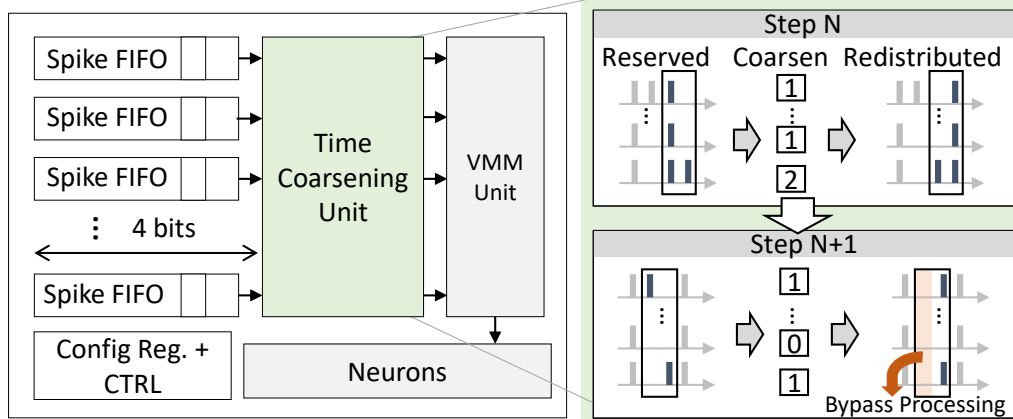


Figure 4.10: Design Concept of Time Coarsening Unit. Spikes in two-time points are redistributed to generate the time point with no spike while preserving the statistic of each coarsened time point the same. The time coarsening unit tends to move spikes one step forward to create space, as step N, to increase the probability that two-time points can be coarsened, as step N+1.

of synaptic array input. To reduce the time difference between input groups, the output current of each group is decoupled from the common source line by a gated current mirror (gCM). The gCM prevents the interference between groups' output current on the common source line during stabilization. The gCM also reduces the time difference between spikes in different time points from the whole 3D-NOR sensing time of a hundred nanoseconds into a source line signal transmission delay of 30 ns.

The gCM is controlled by an enable signal. Once the output current of a group stabilizes, the enable signal triggers the gCM to redirect the output current from the ground to the common source line, as shown in Fig.4.14. A one-dimensional systolic array architecture ensures precise timing control of the gCM. Input vectors are transferred from one side to another, enabling the control signal provided by a delay chain with the same delay as the time difference between input vectors. The flow is illustrated in Fig.4.14. The enable signal is applied to the delay chain in the first phase while the output current stabilizes. The enable signal can reach the next gCM in the subsequent phase when the output current of the next group becomes stable.

To equip the gated current mirror (gCM) to decouple the output current from the source line, the circuit under array (CuA) [153–155] is utilized, offering an additional

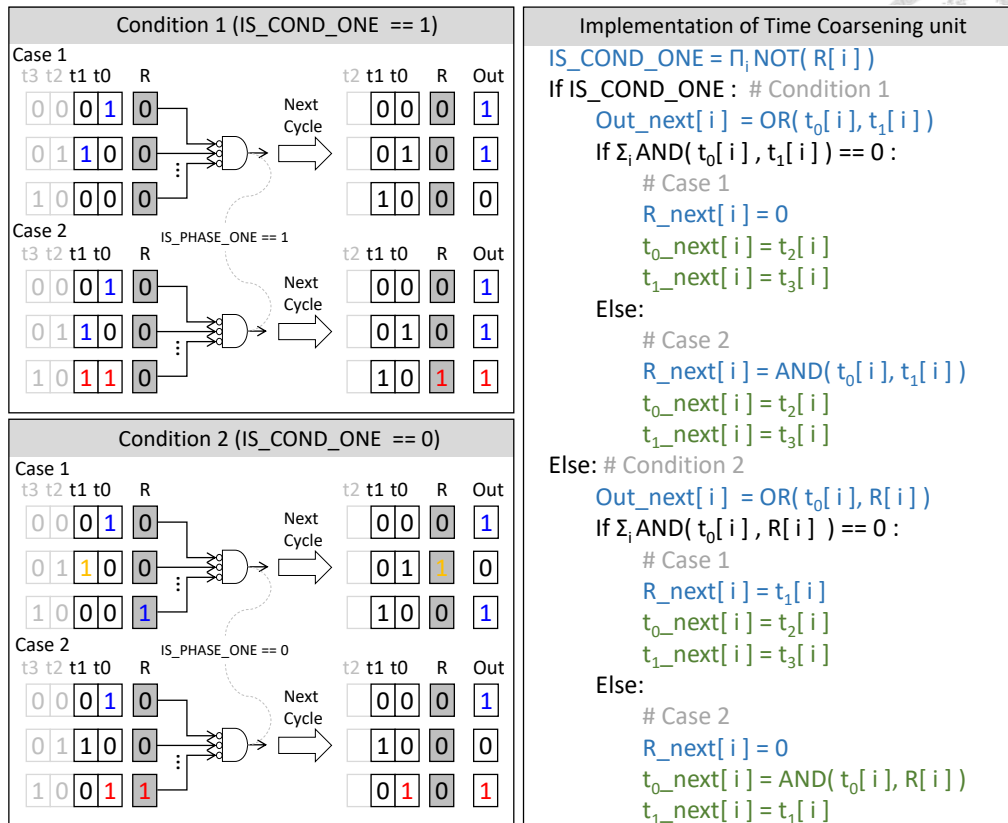


Figure 4.11: Implementation of Time Coarsening Unit. The time-coarsening access to spike in two-time points and re-distribute spike to registers. The proposed design does not require an adder; instead, it requires a few simple logic operations. Moreover, the FIFO only needs to shift two elements or non-shift, simplifying the shift logic overhead.

dimension for accessing the output current on the source line. The detailed architecture, as illustrated in Fig. 4.14, differs from the standard 3D-NOR Flash architecture by including two source line paths. One is the data source line for typical access, indicated by a red line. The other, the SNN source line, is designed to redirect the output current to the gated current mirror beneath the synaptic array. The gated current mirror and the delay chain are implemented beneath the synaptic array to facilitate the desired timing and current flow, as shown in Fig. 4.14.

4.3.3 Skipping Polarized Neuron for Optimization Applications

The prior pruning technique [150–152] prune the weak neurons, which the pruning is non-structured. However, this non-structured pruning technique is ineffective in reducing synapse operation because few highly active neurons still access the whole synaptic array.

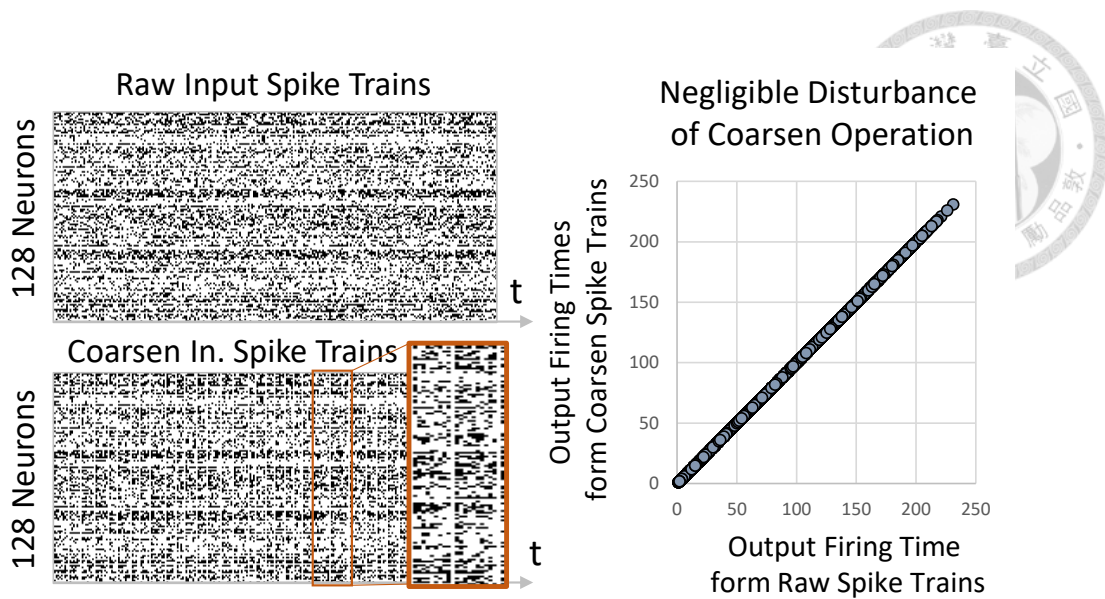


Figure 4.12: Result of Coarsening Unit on 128 Input Spike Trains over 256 cycles. The coarsened input spikes train generates time points without any spike but with negligible disturbance on the output spike time.

To tackle the issue of non-structured pruning, the run-time pruning technique is proposed to deal with those highly active winner neurons. Based on the observation that the neural firing rate polarizes to 1 or 0 while applying the SNN to solving optimization problems, the activities of those polarized neurons tend to preserve the same as the previous cycle, saving time for synapse operation and getting the same result.

The implementation of the run-time pruning is present in Fig. 4.16. After setting up the core to process a set of neurons, the local controller, CTRL, checks whether all these neurons are polarized. If all these neurons are determined as polarized, they only have a low probability, *Holdiong Rate* (HR) = 0.1, to update their activities. Once these neurons are determined to preserve the same as the previous cycle, the SNN core sets up the STOP_FLAG to the host controller to release its processing resource for updating the states of the next cycle.

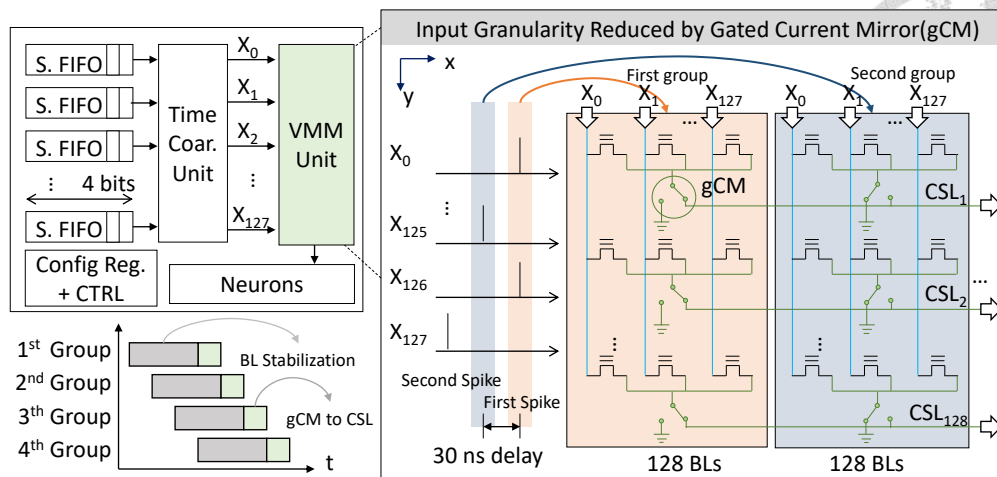


Figure 4.13: Design Concept of Input Granularity Reduction. The subsequent input spikes are provided to adjacent inputs to fully utilize the synaptic array’s bit-line (BL) input. The output source line has a gated current mirror (gCM) to decouple the output current from the output source line while current stabilization.

4.4 Evaluation Setup

4.4.1 Latency Evaluation

We built up the simulator to evaluate the latency. The simulator comprises three components: a neural activity generator, a network partitioner, and a latency evaluator. The neural activity generator evolves the state of spiking neurons considering the unsynchronization from the tick-batching technique. It generates the neural activities at each cycle for latency evaluation. The network partitioner receives the graph of SNN topology and partitions the spiking neurons into subsets, of which the size is the neuron number in an SNN core. The latency evaluator emulates the system with the architecture shown in Fig. 4.9. The latency assessment incorporates both core initialization and neuronal state evolution. In this framework, we implemented an accumulation-based evaluator to measure processing latency.

To assess the latency associated with core initialization, this tool evaluates the delay resulting from the rotational allocation of bus resources to each Spiking Neural Network (SNN) core for data access. The data exchanged encompasses both the input and output neural activities. Additionally, the loading of weights is only for architectures that incor-

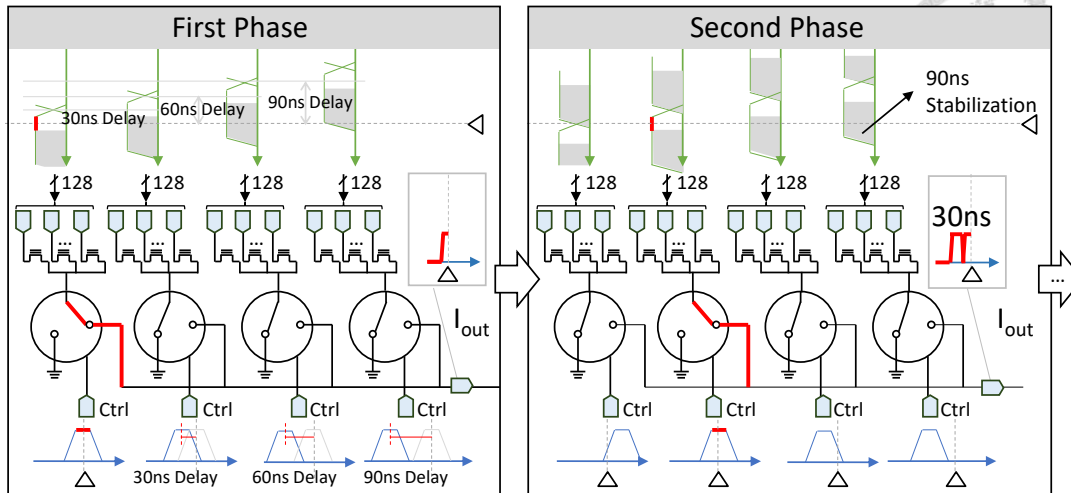


Figure 4.14: One-Dimensional Systolic Array. The encounter of the propagated control signal and stable output current redirecting to the output port is considered a one-dimensional systolic array.

porate digital synapses. Our design utilizes High Bandwidth Memory 2 (HBM2), boasting a read bandwidth of 64GB/s, as the primary memory. This configuration is similar to that of a single-die HBM2, as explored in [142]. The architecture's specifications include a row-to-row delay (t_{RRD}) of 2 cycles, a write-to-read time (t_{WTR}) of 8 cycles, and a write recovery time (t_{WR}) of 16 cycles. The data rate per pin for each of the 256 pins in the single die is 2 Gb/s.

To assess the latency associated with the assessment of neuronal state evolution, the tool evaluates the duration required to process spikes within a specified time window. After the firing rate of input neural activities is configured in the spike generator, the latency accumulator calculates the needed access cycles for the 3D-NOR Flash synaptic array, which functions with a unit size of 512×128 cells and has an access latency of 120ns.

When an SNN core finishes its task, it switches to an idle mode, waiting for the bus to become available for the next assignment. The duration required to configure weights in the non-volatile synapse array is linked to the word line's stabilization time, which is 90 ns for 3D NOR Flash. In contrast, setting up weights in a digital synapse core entails loading the weight from HBM. This loading latency is only accumulated if the necessary weight is not pre-existing in the local cache.

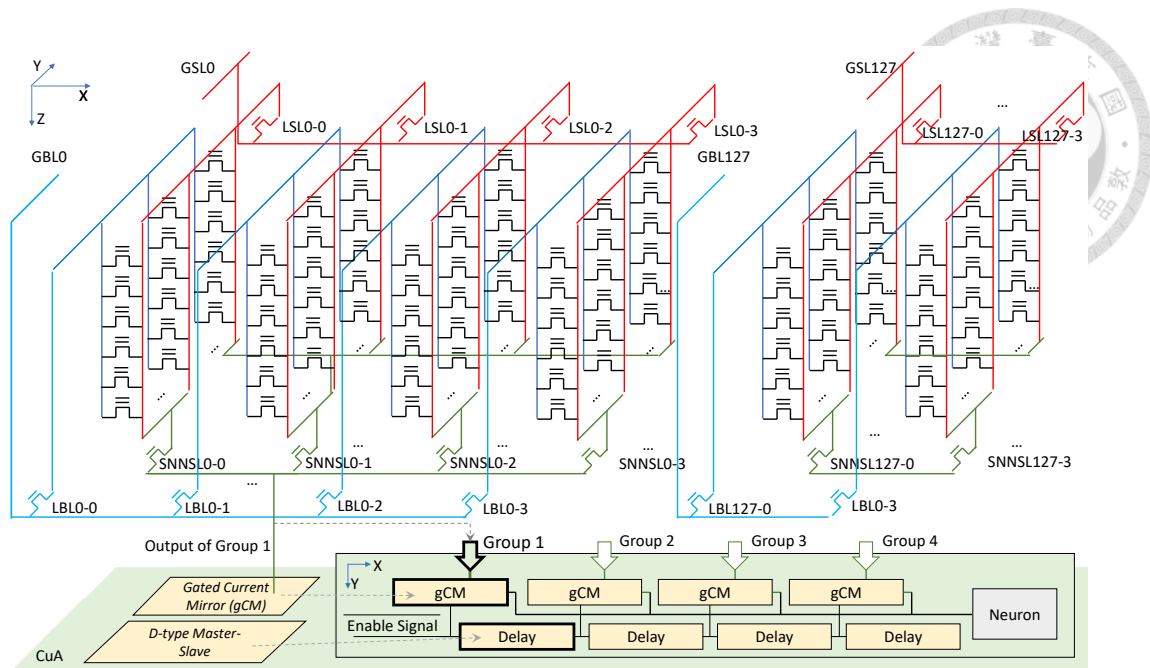


Figure 4.15: Architecture of 3D-NOR Flash Synaptic Array. The inputs are provided to the global bit-line. Then, one of the local bit-lines is selected and transmits the input current to the synaptic array. The output current is redirected to the circuit under the array. The one-dimension systolic array is implemented by gated current mirrors controlled by an enabling signal propagated by a delay chain.

The latency evaluator can also be configured to prior architectures. The implemented latency model of prior architectures includes the SpinalFlow [13], MRAM-XB in Nebula [8], and PTB [14]. The SpinalFlow processes a spike at each cycle and loads 128 weights from the nearby SRAM operated at 200 MHz. The spike count of incoming spikes is utilized to evaluate the latency. The MRAM-XB implements synapse operation through spintronics-based magnetic tunnel junction (MTJ) devices, each with a size of 128x128 and an access latency of 110ns. The latency is the count of the crossbar access multiplied by the access latency. The PTB implements the synapse operation through the 16x16 systolic array operated by the same frequency as SpinalFlow's. The input of the systolic array comprises spikes at different time points instead of input neurons. Each 16 time-point of an input spike train is grouped as a pack. Each pack is an input unit to the systolic array, and the packs without spikes are bypassed. In the above evaluations, the tick-batching size is set to 128, of which the synchronization can be tolerated as Fig. 4.8.

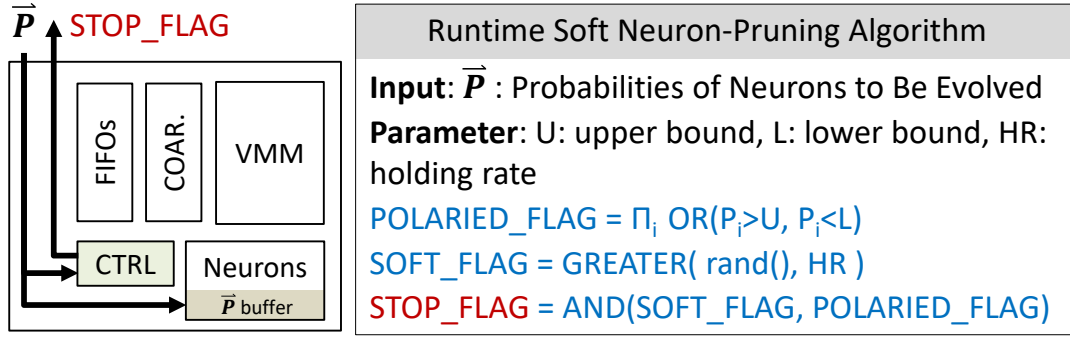


Figure 4.16: Implementation of Runtime Soft Neuron-pruning Algorithm. The local control unit checks whether the P_i (in Eq. 4.4) of Neurons in the SNN core are polarized ($P_i > upper\ bound \vee P_i < lower\ bound$). The checking is processed during transmission and setting up \vec{P} . If all these neurons are polarized, the STOP_FLAG has the probability of 1-Holding Rate to rise and skip the process.

4.4.2 Evaluation Across Diverse Applications

Three applications were selected for latency evaluation: 9x9 SUDOKU, MAX-CUT, and LASSO. SUDOKU and MAX-CUT are combination optimization problems, while LASSO involves dynamic programming. All these problems are mapped onto a recurrent network by converting the problems into weights [47, 102, 133, 156]. The 9x9 SUDOKU is to select a number from 1 to 9 for each file of a 9x9 table such that each column, row, and 3x3 block does not have any redundant number. The problem of the MAX-CUT is to separate the vertexes of a given graph into two groups, and the sum of the edge weight between different groups is maximized. We use the 800 vertexes graph of Gset/G1 from the graph dataset [157, 158] for our evaluation. The LASSO problem is to convert a CIFAR10 image into a 4k sparse vector. The construction of the recurrent network involves convolution and de-convolution operation, employing 64 kernels sized 7x7 and a stride of 4.

The metric to measure the solving quality is different for applications. For SUDOKU, the solving quality is the success rate of the result meeting all the rules. For the MAX-CUT, the solving quality is the ratio between the cut-sum from Goemans-Williamson algorithm [159] and the result from the annealing machine. For the LASSO application, the solving quality is quantified using the minus log root mean square error, i.e., $-\log(\text{RMSE})$,

between the original raw image and the image reconstructed from the encoded 4k vector representation.



4.4.3 Implementation of Heuristic Methods

We also compare our Neureka with other heuristic methods processed on the high-end i9-12900K CPU and RTX-4090 GPU. We apply the mean-field (MF) [160, 161] approximation as an alternative for the iterations of binary spikes, which can be efficiently processed on the CPU or GPU. We also implement the classic simulated annealing (SA) algorithm-metropolis sampling [162] to solve the combination optimization problem, including MAX-CUT and SUDOKU. We utilize the Adam optimizer [163] to solve the LASSO. We also implement the Goemans-Williamson (GW) algorithm [159] to solve the MAX-CUT and recurrent reasoning network [164, 165] to solve the SUDOKU. All these methods meet the constraint of cut sum $>95\%$ of GW's for MAX-CUT, success rate $>90\%$ for SUDOKU, and $-\log(\text{RMSE}) > 0.9$ for LASSO.

4.5 Evaluation Result

4.5.1 End to End Comparison

The effects breakdown of the proposed methodologies on accelerating solving optimization problems are shown in Fig. 4.17. The proposed input granularity reduction, denoted as NEU(SYS), provides 1.52x, 1.59x, and 3.83x faster than simply utilizing 3D-NOR Flash as synaptic arrays for solving MAX-CUT, SUDOKU, and LASSO, respectively. With the time coarsening unit, NEU(SYS+COAR), the processing throughput is further improved by 7%, 13%, and 29%. The breakdown in the case of LASSO shows that the processing throughput is bound by weight movement for digital-based SpinalFlow and PTB as the SNN core increases. On the contrary, both MRAM-XB and Neureka break the memory wall. As a result, the NEU(SYS+COAR) can achieve 2.19x faster than

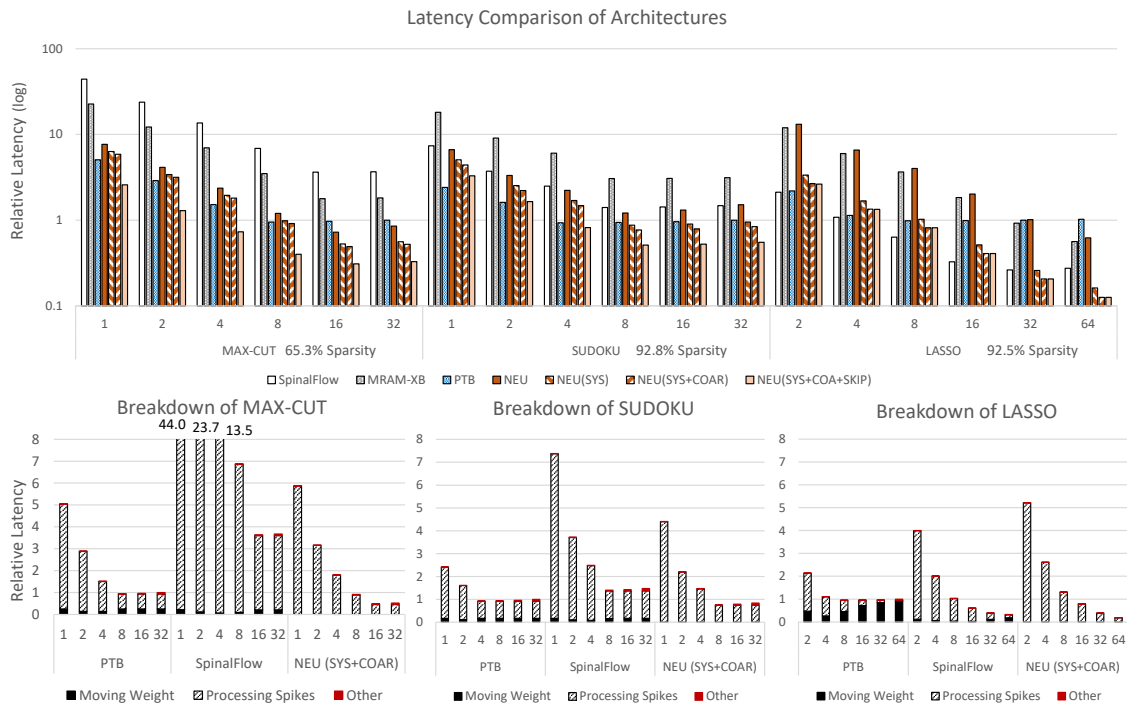


Figure 4.17: Processing Time Comparison amount Architectures. The proposed Neureka provides 3.1x and 1.8x processing throughput faster than PTB due to the outperformed processing throughput of the 3D-NOR Flash synaptic array, and it provides 2.2x faster processing throughput SpinalFlow due to breaking the memory wall of weight movement.

SpinalFlow while adapting to solve LASSO in the case of 64 cores.

The breakdowns of MAX-CUT and SUDOKU show that the processing of spikes dominates the overall performance. The reason is that the model selected for evaluation is too small to access all the spike processing cores, leading to negligible switching costs. The spiking processing throughput of NEU(SYS+COAR) provides competitive (1.19x) processing throughput of PTB operating on 92.8% sparsity while solving SODUKU and outperforming (1.91x) the processing throughput of PTB operating on 65.3% sparsity while solving MAX-CUT. Synergy with the run-time pruning scheme, NEU(SYS + COAR + SKIP), the proposed Neureka is further improved by 1.59x and 1.52x processing throughput. As a result, compared with PTB, the NEU(SYS + COAR + SKIP) provides 3.05x and 1.81x faster processing throughput for solving MAX-CUT and SUDOKU, respectively.

4.5.2 Effect of Skipping and Coarsening



The performance gain throughout the annealing process is shown in Fig. 4.18. The proposed coarsening unit, COAR, reduced the overall processing time from 0.7 to 0.9. The coarsening unit is more effective in the annealing process than the latter. The coarsening unit is less effective because some neurons become more active, i.e., high firing rate, during the annealing process. Those highly active neurons generate massive spikes, reducing the probability of obtaining the time point without a spike. Taking an example of a spiking neuron polarizing its firing rate to 1, it is impossible to redistribute spikes to empty the spikes of a time point.

On the other hand, in the latter half of iterations, the processing time reduction of the run-time pruning, SKIP, ranges from 50% and 70% for the application of SUDOKU and MAX-CUT, respectively. In contrast, the prior pruning weak neurons technique, PRUN-WEAK, is ineffective. For neuron-competing applications such as SUDOKU and MAX-CUT, neurons are polarized to either non-spike or highly activated. Those firing-rate-polarized neurons are considered stable neurons, of which the synapse operations are skipped to save processing time. Conclusively, on fore-iteration of the annealing process, the coarsening of input spike is effective thanks to the non-polarization of spiking neurons. In the latter half of iterations, the run-time pruning technique is effective due to the polarization of spiking neurons.

The requirement of soft skipping, having a probability of not skipping polarized neurons, is shown in the annealing process of SUDOKU, in which the neurons are temporally polarized but not yet anneal to the solution at the beginning. In the case of the neuron being considered as polarized but not yet annealing to the solution, the non-update of temporally polarized neurons leads to the SNN not reaching final solutions. The proposed soft skipping technique provides a small probability, i.e., 0.1, being skipped to enable the update of those temporal polarized neurons and reach the SUDOKU solution.

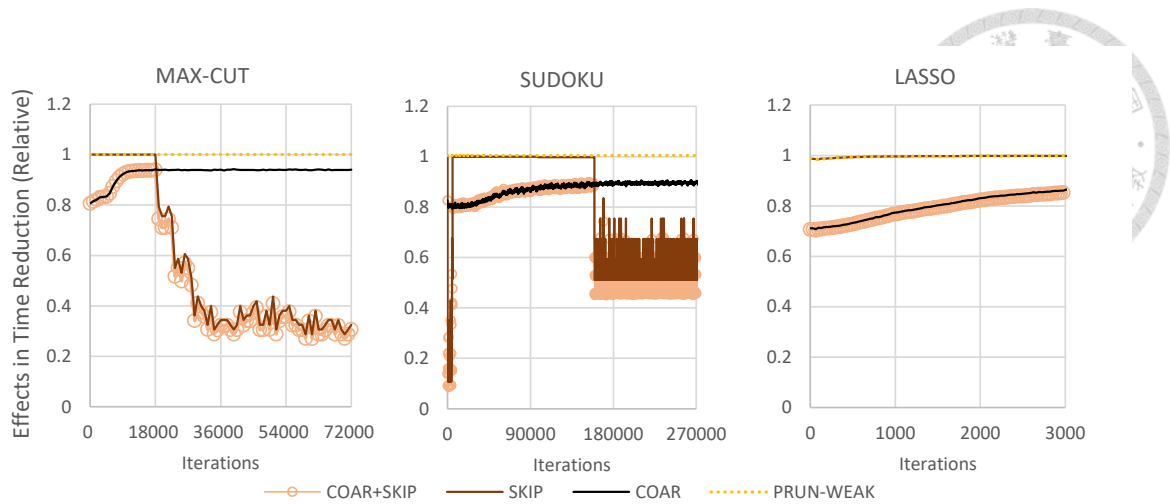


Figure 4.18: Effective of Proposed Design Throughout the Annealing Process. The time coarsening unit, COAR, is initially effective, and the proposed run-time pruning technique, SKIP, becomes effective during annealing. In contrast, the prior pruning weak-spiking neuron is ineffective.

4.5.3 Discussion of Neural Firing Rate Scaling

The impact of the scaling firing rate is shown in Fig. 4.19. The neural activities are linearly encoded to the maximal firing rate. The less the maximal firing rate is, the higher the probability that an empty time point can be bypassed. However, the reducing of the scale down of the firing rate leads to the enlarge of noise due to the scaling back of the observed firing rate, i.e., $p^{observed} = \frac{Spike\ Count}{TW} \frac{1}{Maximal\ Firing\ Rate}$, enlarge the variation by $(\frac{1}{Maximal\ Firing\ Rate})^2$. The result shows that for all the applications in this thesis, the scaling down of the maximal firing rate drops the solving quality. In solving MAX-CUT and SODOKU, the maximal firing rate = 0.6 selection can hold the solving quality. Yet, for solving LASSO, the $-\log(RMSE)$ drops by reducing the maximal firing rate. The maximal firing rate needs to be set to one to get the best-solving result.

4.5.4 Comparison with CPU and GPU

The comparison of other heuristic algorithms running on Nvidia RTX4090 and Intel i9-12900K CPU under the same solving quality is present in Fig. 4.20. The result shows that the proposed Neureka provides 6.58x, 1.78x, and 2.96x faster than other heuristic

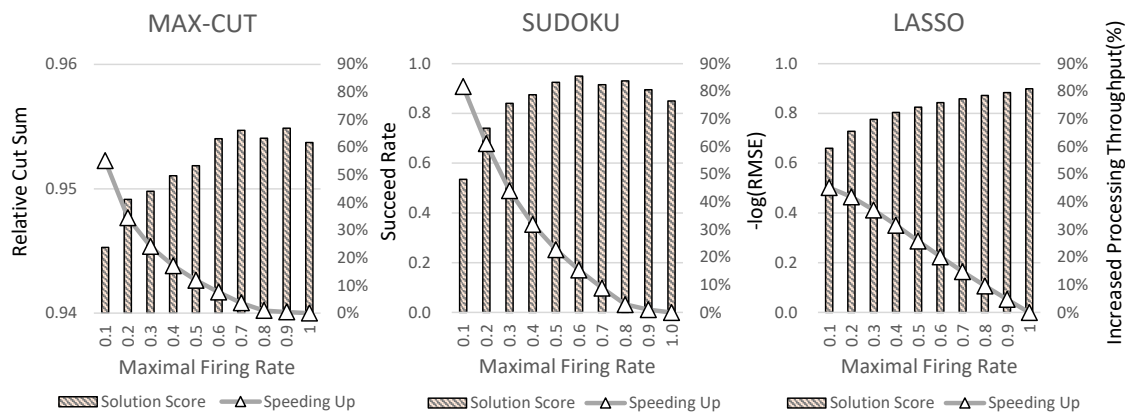


Figure 4.19: [Impact of Maximal Firing Rate. MAX-CUT and SUDOKU tolerate the scale of the max firing rate to 0.6

methods running on the high-end CPU or GPU while solving MAX-CUT, SUDOKU, and LASSO, respectively. The simulated annealing algorithm [162], SA, for solving MAX-CUT and SUDOKU requires calculating the loss to determine the probability of accepting the sampled state. The calculation of the loss involves access to all the neuron states. Moreover, sampling to flip states of a few variables requires the index access of the state vector, and additional condition branch [166] is also included in the code. As a result, the simulated annealing process is not purely vector-matrix multiplication, which discounts the process throughput of GPU. Moreover, the Adam optimizer [163], a widely used optimization tool nowadays, for solving LASSO requires accessing all the neuron states to calculate the global loss and back propagates to the neuron states, which increases additional memory access and calculation. Compared to the SA and Adam, the spiking neural network annealing machine does not access all the neuron states to calculate the global loss. Each neuron evolves its state through a rule inspired by biological neurons. The cooperation of these neurons realizes the annealing process.

Moreover, the spiking neural network can process the annealing process without training the model. The recurrent reasoning network (RRN) [164, 165] proposed that the optimization problem can be solved by training a reasoning network. Despite the RRN achieving the proposed Neureka's performance, data collection is required for training. Compared to the RRN, the spiking network does not require training the model. Instead, the weight can be transformed directly from the QUBO form.

The spiking neural network can be approximated by mean-field (MF) approximation [160, 161], which simplifies the neuron state by the expected firing rate with additional disturbance, which allows the GPU to evolve the spiking neural network efficiently but with floating-point input and high-precision computation for each iteration being required. The result shows that the proposed Neureka with eight cores provides more than 25.7x, 81.1x, and 4.86x faster than MF running on GPU for solving MAX-CUT, SUDOKU, and LASSO, respectively. For solving LASSO, the acceleration is up to 15.8x while setting the core to 256.

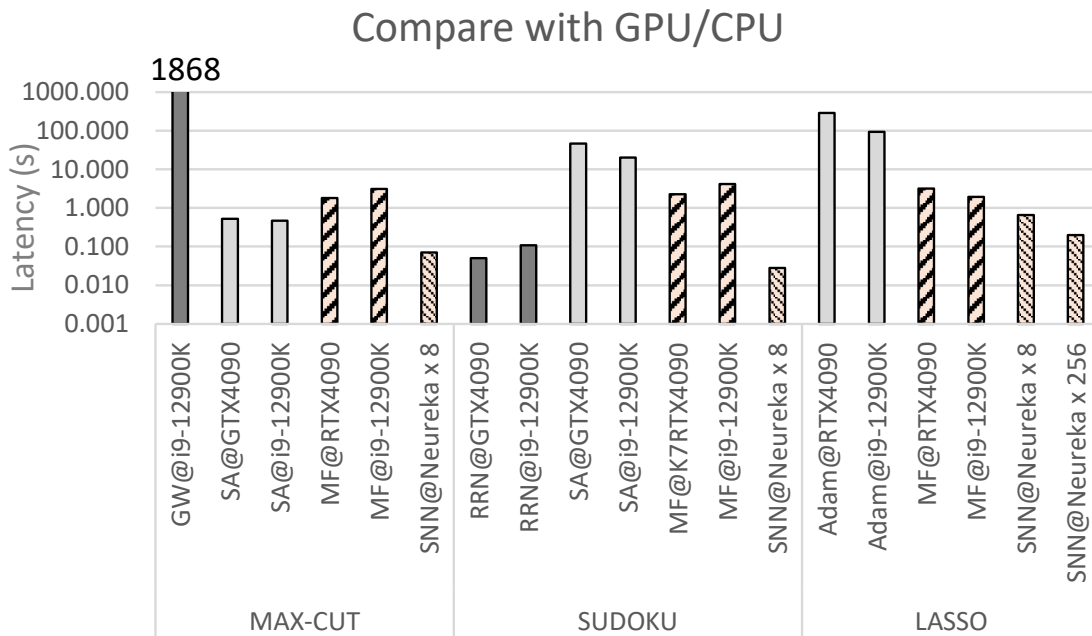


Figure 4.20: Comparison with other Heuristic Method Processing on CPU and GPU. The Neureka achieves 6.58x, 1.78x, and 2.96x faster than Von-Neumann’s architecture for solving MAX-CUT, SUDOKU, and LASSO, respectively.



Chapter 5

Efficient and Error-resilient Spiking Neural Networks through Binarization

5.1 Background and Motivation

The success of deep neural networks (DNNs) has brought significant benefits to numerous fields while impacting our daily lives. However, the high inference accuracy comes at the cost of large resource demand because NNs require a huge number of parameters and a massive number of MAC operations. This poses an immense challenge because high-performing NN models are becoming increasingly larger, while low-power operation for sustainability is rapidly gaining importance in a wide range of application domains, especially for embedded systems.

5.1.1 Spiking Neural Networks (SNNs)

SNNs are a computing scheme that is heavily influenced by the dynamics of biological neurons and are similar to them. In SNNs, neural activity is event-driven and described by the integration of voltage spikes over time. When a neuron receives a certain number of spikes, a predetermined threshold potential may be passed, causing the *firing* of an output spike. The sparse-spike-based operations use efficient coding and enable low-power operations because accelerators built with simple analog or digital components can be

employed for computation [167].



5.1.2 Existing challenges in SNNs

One major challenge in SNNs is to process inputs coded in the time domain efficiently without influencing inference accuracy. Although recent studies claim that temporal input coding with one bit can implement multi-bit multiplication through single-bit operations, they are not yet proven to be effective on complex tasks [168–170]. If stochastic encoding of the inputs is used instead of temporal coding, the high accuracy from well-trained models is inherited. However, stochastic coding necessitates many computation cycles to achieve high accuracy. Due to this, in analog computing-based implementations, a large membrane capacitor (C_{mem}) is inevitably required to reliably accumulate and hold the charges of multiple cycles. Otherwise, the required inference accuracy cannot be sustained. A large C_{mem} size results in high energy, long latency, and expensive area cost, constituting one of the major bottlenecks in SNN hardware accelerators [46, 48]. Although reducing the size of C_{mem} offers numerous benefits, the reduction of C_{mem} dramatically decreases inference accuracy. It is because the reduction of C_{mem} decay the precision of the firing time, which encodes the inner product result. Hence, if the SNN is error-prone, reducing C_{mem} size will aggressively degrade the inference results. *All in all, efficient and error-resilient SNN models allow the optimization of membrane capacitors, resulting in energy, area, and speed improvements with minimal accuracy loss. Achieving this goal is concisely the focus of this thesis.*

5.1.3 Binarized Neural Networks (BNNs)

Due to the lightweight implementation, BNNs have recently received remarkable attention since their inception [72, 171]. Binarization has three major benefits for the efficiency of NNs. (1) BNNs have significantly smaller model sizes and hence reduce movements of data between memories and computing units. (2) Binarization of the weights and

activations enables replacing complex MAC circuits with simple XNOR logic gates. (3) BNNs can be optimized to achieve high error resiliency [62, 63].



5.1.4 Binarized Spiking Neural Networks (BSNNs)

To optimize SNN modes computations in which C_{mem} size is minimized while the required level of inference accuracy is still sustained, we investigate the effectiveness of SNNs based on binary representation of model parameters instead of integer representation, as done in state of the art. SNNs and BNNs synergize outstandingly in their motivations and were designed with similar objectives in mind. For both BNNs and SNNs, energy efficiency, low latency, and error tolerance are crucial properties. Therefore, recent studies have focused on deploying BNNs on SNN hardware [73, 172]. The promise of BNNs for SNNs is twofold: First, concerning computations, XNOR can be used instead of multiplications, and at the same time, *stochastic input representation is not anymore necessary* leading to single-sample computation. Secondly, BNNs can be optimized for high error resiliency, which enables a significant reduction of C_{mem} size. This leads to a profound increase in the efficiency of SNNs because C_{mem} size determines energy, latency, and chip area. However, *the benefit of BNNs for SNNs on reducing C_{mem} size and hence obtaining area, latency, and power savings without sacrificing accuracy has not been researched yet, and this is the first thesis.*

5.2 System Model

5.2.1 Binarized Neural Networks (BNNs)

BNNs training: A common method for NN training applies stochastic gradient descent (SGD) with mini-batches. The training data is described with

$$\mathcal{D} = \{(x_1, y_1), \dots, (x_I, y_I)\} \text{ with}$$

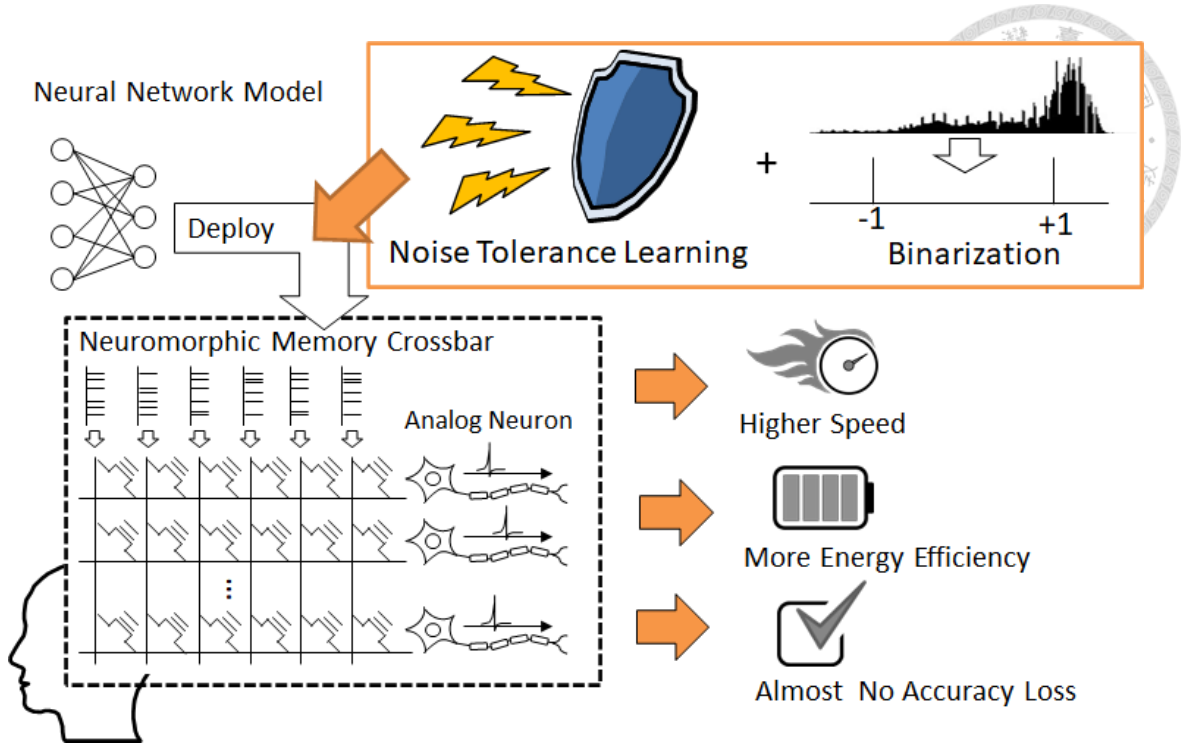


Figure 5.1: Overall Structure of Binarized SNN

$x_j \in \mathcal{X}$ as the inputs,

$y_j \in \mathcal{Y}$ as the labels, and

$\mathcal{L}: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ as the loss function.

$\mathbf{W} = (W^0, \dots, W^L)$ are the weight tensors of layer $0, \dots, L$ and $f_W(x)$ is the output of the NN. The goal is to find a solution for the optimization problem, i.e., $\mathbf{W} = \arg \min_W \frac{1}{I} \sum_{(x,y) \in \mathcal{D}} \mathcal{L}(f_W(x), y)$ by a mini-batch SGD strategy, computing gradients using backpropagation. To train BNNs, the weights and activations are binarized in the forward pass. For backpropagation, the floating point numbers are used for parameter updates [72].

BNNs inference: In BNNs, the weights and activations are binarized. Due to this, the MAC operations of a layer can be computed as follows:

$$\text{popcount}(\text{XNOR}(\mathbf{W}_i^\ell, \mathbf{X}^{\ell-1})) > \mathbf{T}, \quad (5.1)$$

In this equation, the *popcount* counts the number of ‘1’s in the XNOR result, \mathbf{W}_i^ℓ describes the weights, $\mathbf{X}^{\ell-1}$ the inputs, ℓ the layer number, i the filter, and \mathbf{T} is a learnable thresh-

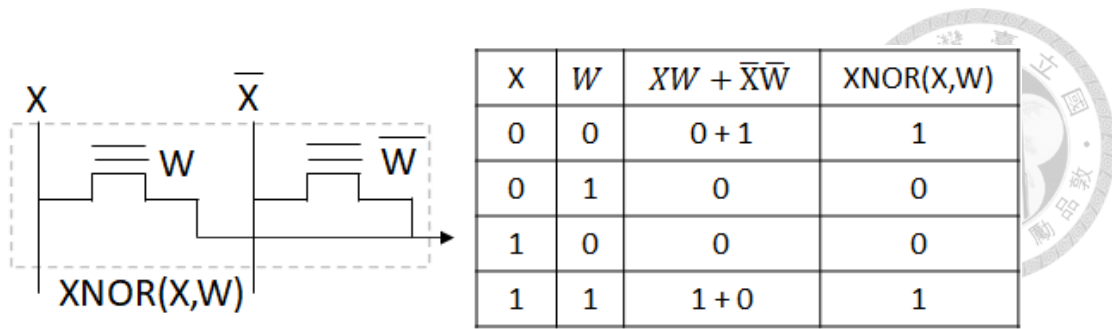


Figure 5.2: XNOR Logic Gate Realization for Binarized Multiplication in SNNs.

old parameter (attained by the batch normalization parameters [173]), whose comparison produces binarized values, which are fed to the subsequent hidden layer as inputs [72].

Error tolerance of BNNs: It has been shown that BNNs feature remarkable error resiliency under bit errors in the weights. Importantly, if errors are induced during training, the error resiliency can be increased even further [62]. To optimize BNNs for error resiliency, the state-of-the-art method for multiclass classification problems applies a modified hinge loss based on margin-maximization alongside error induction during training [63]. The advantage of this method over the standard cross entropy loss has been evaluated for a variety of BNN models in [63].

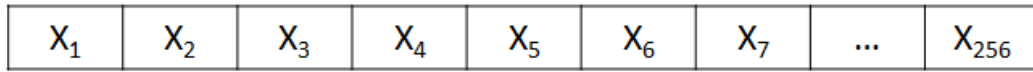
5.2.2 Spiking Neural Networks (SNNs)

Stochastic input coding in SNNs: State-of-the-art SNN implementations use stochastic input coding for multiplication with multi-bit or binary weights. The stochastic input encoding is widely used [174–176], because it exploits the noise tolerance capability of NNs for computing efficiency. The input to the bit-line is the random variable following a Bernoulli distribution, with the firing rate $\frac{X_i}{X_{MAX}}$, where X_i is the input and X_{MAX} the maximum range of the input value in binary representation. The firing rate is interpreted as the probability of a spike. For the binary input case, the X_{MAX} is 1, and p_i is either 1 or 0.

Crossbar array for general multiplication: The typical operation of computing in memory is to compute logical AND between input and stored weights. The AND operation

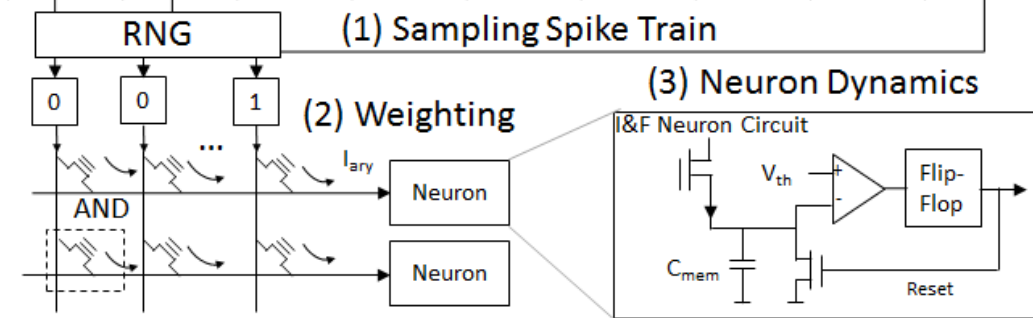
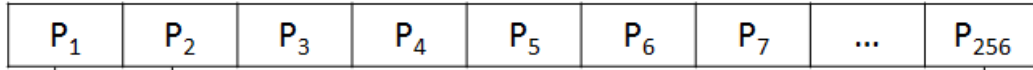
AND Operation

Input Vector (4 bit number as example)



Probability Vector

$\downarrow 1/X_{MAX}$: (Note: $X_{MAX} = 15$)



RNG: Random Number Generator

XNOR Operation (for Case of BNN)

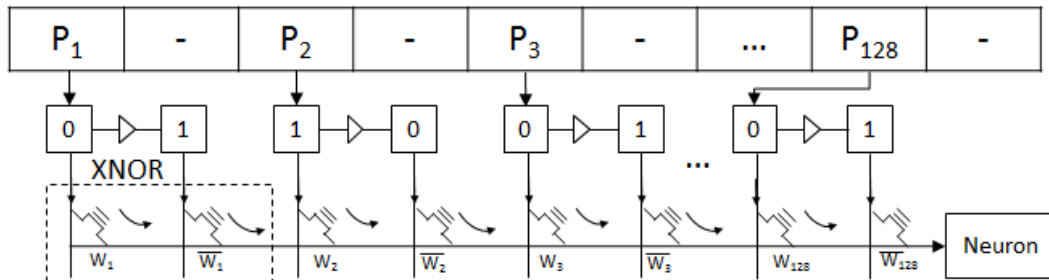


Figure 5.3: Typical AND Mode and BNN XNOR Mode

becomes binary multiplication when both input and weight are in binary representation. In our case, X_i is always binarized after stochastic conversion. This simplifies the multiplication to a certain number of AND operations determined by the number of binary cells to represent the weights.

XNOR-based crossbar for binarized SNNs: In this thesis, the XNOR array is adopted for the implementation of BNNs. To realize XNOR in function, two memory cells (e.g., FeFET transistors, see [177]) are needed. Our XNOR design is based on the work in [178], see Fig. 5.2. The XNOR crossbar needs to be extended for the first layer, whose inputs are positive multiple-bit. To realize the multiplication in this case, the method of subtracting NOT(W), inspired from [174] is adopted.

Analog implementation of SNNs: The implementation of SNNs using a memory crossbar and the input transformation is shown in Fig. 5.3. The design of the memory crossbar with analog neuron circuit is based on [107]. The crossbar can implement both AND and XNOR operations, as Fig. 5.3. The steps of operation for SNNs are as follows:

- (1) The input spikes are provided to the bit-lines of all memory cells in parallel. The binary operation results (XNOR or AND) in all word lines are computed in parallel as well. Finally, the output currents of all binary gates in a word line are accumulated by Kirchhoff's circuit law and passed to the neuron circuit.
- (2) The accumulated output current charges the membrane capacitor C_{mem} . Note that when stochastic input encoding is used, the same binary operation and accumulation in steps 1 and 2 may need to be repeated for several cycles, with new input samples in each cycle, until the voltage across the capacitor surpasses the predetermined threshold voltage V_{th} . Once the voltage across the membrane capacitor reaches V_{th} , the neuron generates an output spike. The observed frequency of the output spike is determined by 1 over the time to first spike t_{fire} and converted to an inner product by $\sum_i W_i X_i = X_{MAX} \frac{C_{mem} \cdot V_{th}}{I_{on}} \frac{1}{t_{fire}}$, where i denotes the index of different partial inner products and $I_{ON} = V_{BL} G_{cell}$ is the on-state current of the cell in the binary gate (G_{cell} is the conductance, V_{BL} the bitline voltage).
- (3) The firing time for each neuron may be different. In the crossbar architecture, bit-line signals held for those neurons that are not fired yet still contribute current to fired neurons. To eliminate cell currents flowing to the fired neuron, the fired neuron will generate a signal to turn off cells on the word line that are connected to the fired neuron.
- (4) The t_{fire} is acquired by a digital counter. Subsequently, the sampling number is converted to an inner product by a time-to-digit unit. Because of the limited crossbar size, all the inner product computations need to be separated into a series of smaller

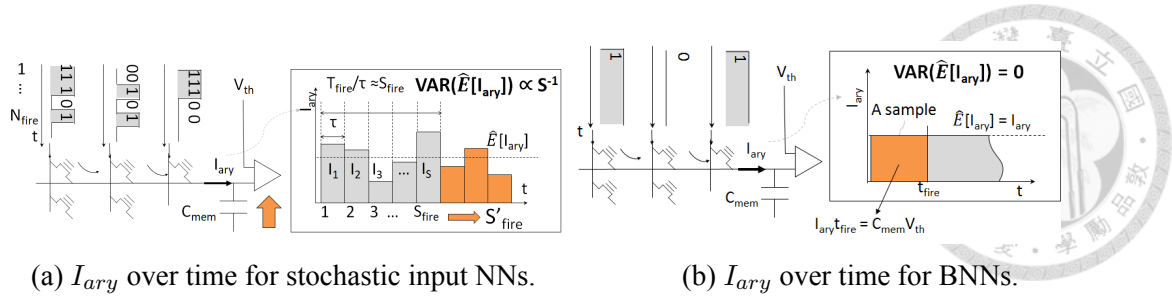


Figure 5.4: I_{ary} over Time

inner products. For this reason, adders are needed for accumulation. After that, the result is ready for subsequent layer computations. As this follows conventional designs, the details of these devices are not shown in this thesis for brevity.

5.3 Impacts of Binarization on SNNs

As shown in Fig. 5.4a, all binary inputs are sampled from the given probability at each cycle. The expected current, $E[I_{ary}]$, is equal to $I_{on} \sum_i W_i P_i$, where P_i is the normalized input of X_i , W_i the binary weight, and I_{on} is ON-state cell current. Due to the stochasticity of the SNN computing scheme, sampled inner product currents can vary. To reduce discrepancy, multiple samples are required for averaging.

In analog computing SNNs, the sampled current contributes charges to the membrane capacitor over time.

The observed average current, $\hat{E}[I_{ary}]$, is equal to the total charges in the capacitor divided by the total charging time. When a neuron fires, it follows that the total charge in the capacitor has reached $C_{mem} V_{th}$. The observed average current can be obtained from

$$\hat{E}[I_{ary}] = \frac{C_{mem} V_{th}}{t_{fire}}. \quad (5.2)$$

To reduce the discrepancy between observed average currents $\hat{E}[I_{ary}]$ and system expected current $E[I_{ary}]$, a certain number of samples is required based on the law of large numbers. The way to increase the number of samples is by increasing C_{mem} size such that more

samples of inner product currents are required to make the neuron fire.

In short, the inner product precision is a function of the membrane capacitor size, which needs to be configured based on the number of required samples to achieve a certain inference accuracy. In the following, we will discuss the implications of binarization and resiliency, focusing on membrane capacitor size reduction and its effects on analog SNN accelerators with respect to energy and latency.

5.3.1 Impact of Binarization in SNNs on Membrane Capacitor

When deploying BNNs as SNNs, the inputs of the BNN are bi-state, which is either 1 or 0. The probability of binary input is also either 1 or 0. In other words, the inputs are the *deterministic* values, and the output current is a constant value. Taking advantage of the deterministic output when deploying BNNs as SNNs, the SNN does not need multiple cycles to collect multiple output currents to achieve high precision output, unlike multi-bit SNNs. That is, the stochastic behavior of SNN is eliminated for any size of membrane capacitor selected. Therefore, in the case of binarized weights and inputs, it is possible to reduce the size of the membrane capacitor. In the following, we focus on two impacts of binarization on SNN computation. First, we discuss the one-cycle computation stemming from binarization and why this allows a reduction of C_{mem} size. Then, we explain why the accumulated charges are expected to be smaller with binarization.

Binarization allows Cmem reduction: The computing error in stochastic sampling needs to be small enough to prevent inference accuracy VAR

loss [174], and we describe it as $\text{VAR}(\hat{E}[I_{ary}^{stoch}]) \leq K$, where $K > 0$ is a certain model-dependent criterion, which needs to be defined at design or run time to sustain a certain SNN inference accuracy. To reduce $\text{VAR}(\hat{E}[I_{ary}^{stoch}])$, the sampling number S needs to be increased, since the variance becomes smaller with a higher number of samples S due to the law of large numbers, i.e. $\text{VAR}(\hat{E}[I_{ary}^{stoch}]) \propto \frac{1}{S}$. The sample number depends on $S = \frac{C_{mem}^{stoch} V_{th}}{I_{on}} \cdot \frac{1}{\sum_i W_i P_i} \cdot \frac{1}{\tau}$, and since the on-state current I_{on} and the targeted inner

product value $\sum_i w_i p_i$ can be replaced with $\hat{E}[I_{ary}^{stoch}] = I_{on} \sum_i W_i P_i$, we can write

$$\hat{S} = \frac{C_{mem}^{stoch} V_{th}}{\hat{E}[I_{ary}^{stoch}]} \cdot \frac{1}{\tau} \geq Q,$$



where $\tau > 0$ is the pulse width, which is limited by circuit parasitics. For sustaining accuracy, the number of samples S needs to be high enough, i.e., $S > Q$ such that the variance has an upper bound $\text{VAR}(\hat{E}[I_{ary}^{stoch}]) < K$ for any inner product value. The number of minimum samples Q must be found empirically based on the NN model for the stochastic input case. For BNNs, there is no bound Q since $S = 1$. For BNNs, S can be set to 1 because $\text{VAR}(\hat{E}[I_{ary}^{bin}]) = 0$, since the input is deterministic. Due to this, in BNNs, no Q needs to be found, and the variance is independent of the size of the membrane capacitor. *Thus, in BNNs, the membrane capacitor size can be reduced without accuracy cost stemming from a low number of samples.*

When there is a minimal requirement for the precision of the sampled current, in the stochastic case, more charges (proportional to sample number S) may be needed compared to the binarized case. We assume there is a constraint, the minimum charge unit $q^{min} = I_{ON}\tau$, for storing accumulated currents in the membrane capacitor, where I_{ON} is the cell current and τ is the pulse width determined by circuit parasitics. In the stochastic case, $\sum_s \sum_i q^{min}$ will be stored in the capacitor. In the binarized case, this is $\sum_i q^{min}$, since only one sample is necessary. *Charge needs to be accumulated equal or more times in the stochastic case than in the binarized case, i.e., $\sum_i q^{min} \leq \sum_s \sum_i q^{min}$.* This implies that a larger membrane capacitor size is needed in the stochastic case to hold the charge. When the capacitor size is set to be smaller in BNNs, the voltage across the capacitor increases faster, causing earlier spikes. For the stochastic case, the required charge is larger, and the required spiking time is also increased. However, the firing times vary based on the distribution of the inner product. There are still minimum and maximum firing times, which are model-dependent. Although firing time is related to latency, it cannot be used as a metric for measuring and comparing latency between binarized and stochastic cases since the slowest firing time of neurons determines latency.



5.3.2 Impact of Binarization in SNNs on Latency

As described above, the latency of a neuron depends on the inner product value, determining t_{fire} . However, for different numbers of samples, S , and a number of bits used for encoding weights or inputs, the distribution and the range of the inner product may differ greatly. This is why it is difficult to compare t_{fire} among different models and sizes of C_{mem} . To fairly compare latency among different cases, we propose a metric we call guaranteed response time (GRT).

The guaranteed response time (GRT) $\log(t_G)$ is an upper bound for the latency of a neuron, as illustrated in Fig. 5.5. We define the GRT as the maximum time that is given to a neuron, within which it must respond with a spike, i.e., the maximum allowed latency of a neuron is set to t_G . The neurons with $\log(t_{fire}) > t_G$ are considered as no-fire, and their t_{fire} s are set to infinity (inner product 0). Once the neurons are considered as no-fire, the information of those neurons is lost. The information loss metric is the available region ratio of output value defined by $\frac{B-K}{B}$. The relation between t_{fire} and $\sum_i W_i X_i$ is defined as $t_{fire} = \frac{C_{mem} V_{th}}{I_{ON}} \cdot \frac{X_{MAX}}{\sum_i W_i X_i}$. For convenience, we take the logarithm, which results in: $\log(t_{fire}) = \log(\frac{C_{mem} V_{th} X_{MAX}}{I_{ON}}) - \log(\sum_i W_i X_i)$. For values with $\log(\sum W_i X_i) > B-K$ of inner product values is lost. Thus, we define the information loss metric as:

$$\frac{B - K}{B} = 1 - \frac{K}{B} = 1 - \frac{t_G}{\log(\frac{C_{mem} X_{MAX} V_{th}}{I_{ON}})} \quad (5.3)$$

where B and K are labels in Fig. 5.5, $\frac{1}{X_{MAX}}$ is unit of input x, and 10^{t_G} is provided guarantee response time.

Our metric describes the information loss for an arbitrary guaranteed response time of 10^{t_G} . It is evident in Eq. (5.3) that the metric is indexed by the product of $C_{mem} X_{MAX}$, which implies that a *smaller membrane capacitor leads to higher information loss*. To minimize information loss, C_{mem} should have a large value. However, when a low value of the information metric is desired, the guaranteed response time needs to be set to a large value, affecting the latency. +

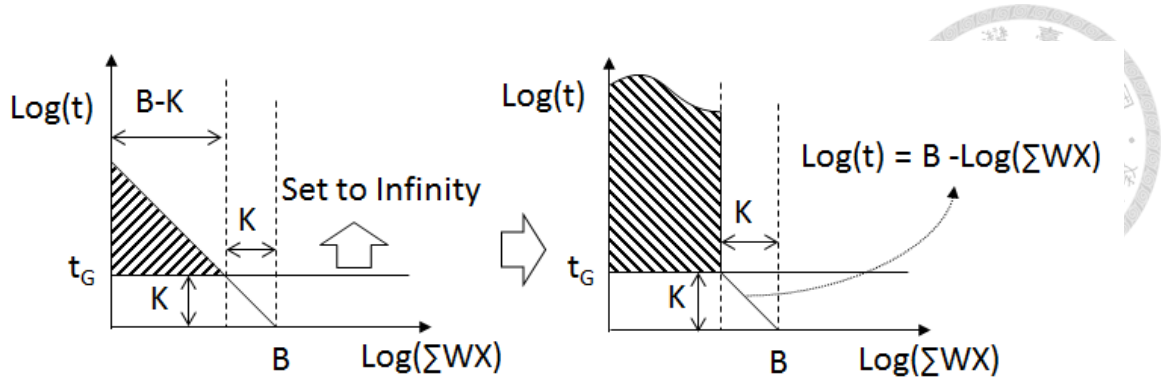


Figure 5.5: Description of Information Loss Metric.

5.3.3 Impact of Binarization in SNNs on Energy

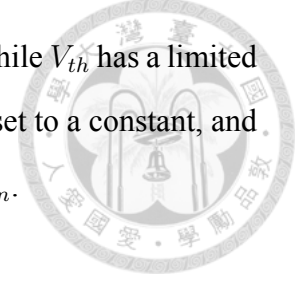
The required energy to make the neuron fire is reduced with a smaller membrane capacitor size. The energy consumption of analog-based SNNs is mainly determined by the synapse array, neuron circuit, and analog-to-digital conversion. When emerging devices with steep sub-threshold slopes are employed, the static energy consumption of an amplifier in a neuron circuit is eliminated by a single transistor [179]. Furthermore, the conversion from t_{fire} to a digital representation only needs to be performed at firing time. This means the energy consumption is mainly determined by the current that is produced by memory arrays performing inner products.

The consumed energy of the crossbar array is proportional to the charge coming out from the memory array as shown in Eq. (5.4). Charge from the memory array is stored in the memory capacitor and increases the membrane voltage. The neuron only fires when the required charge reaches $C_{mem}V_{th}$. Therefore, with a reduction of the membrane capacitor size, the energy consumed from the array is also reduced.

$$E = V_{BL}V_{th}C_{mem} \quad (5.4)$$

As shown in (5.4), due to $E = V_{BL} \int_0^{t_{fire}} I_{ary} dt$ and $\int_0^{t_{fire}} I_{ary} dt = V_{th}C_{mem}$, the energy of the systems depends on bitline voltage V_{BL} , threshold voltage V_{th} , and membrane capacitor size C_{mem} . V_{BL} is set such that reduction of V_{BL} would cause further reliability issues. The required capacity for charges corresponds to the factor of $C_{mem}V_{th}$.

The scalability of the charge capacity depends on the size of C_{mem} , while V_{th} has a limited operation margin in an analog circuit. Thus, in this thesis, the V_{th} is set to a constant, and the requirement of the charge capacity is indexed by the size of C_{mem} .



5.3.4 Errors by Cmem Reduction and Countermeasures

Errors from information loss: With a small capacitor size, parts of the inner product may not be considered during computation due to the voltage increasing faster, causing earlier spikes. This error can be described with $t_{fire}^{LL} = t_{fire} - \epsilon_{IL}$ due to earlier firing time. Using a small GRT has the same effect since it reduces t_{fire} by a constant value. To control the information loss, the capacitor size needs to be set accordingly.

Errors from limited sampling frequency: With smaller firing times caused by smaller C_{mem} size, the sensing frequency of the FF needs to be increased. Consider the case of one rising edge at time t_{re} and one spiking signal at t_{fire} . The spiking signal is not latched until the following rising edge. The sensing error is then described by $t_{re} - t_{fire}$ (see Fig. 5.6). The error manifests itself as a neuron timing error, i.e., the shifted firing time is $t_{fire}^* = t_{fire} + \epsilon_{SF}$. To alleviate this, the sampling frequency of the FF needs to be increased accordingly with a smaller capacitor size to minimize ϵ_{SF} . However, f_{SF} cannot be set arbitrarily high due to fundamental limits.

Combined effect of errors and countermeasures: We denote the error due to reduced capacitor size, with errors from information loss (IL) and from limited sampling frequency (SF) as

$$t_{fire}^{IL,SF} = t_{fire} - \epsilon_{IL} + \epsilon_{SF}. \quad (5.5)$$

These errors may degrade inference accuracy significantly. Fortunately, the BNNs employed in deploying SNNs can be optimized for error resiliency, making it feasible to tolerate timing errors without substantial degradation in accuracy. For this reason, we can utilize the method for resiliency optimization proposed in [63], where bits of the weights W_i are flipped. Due to the property of XNOR, a flip in one W_i from 0 to 1 causes the popcount-

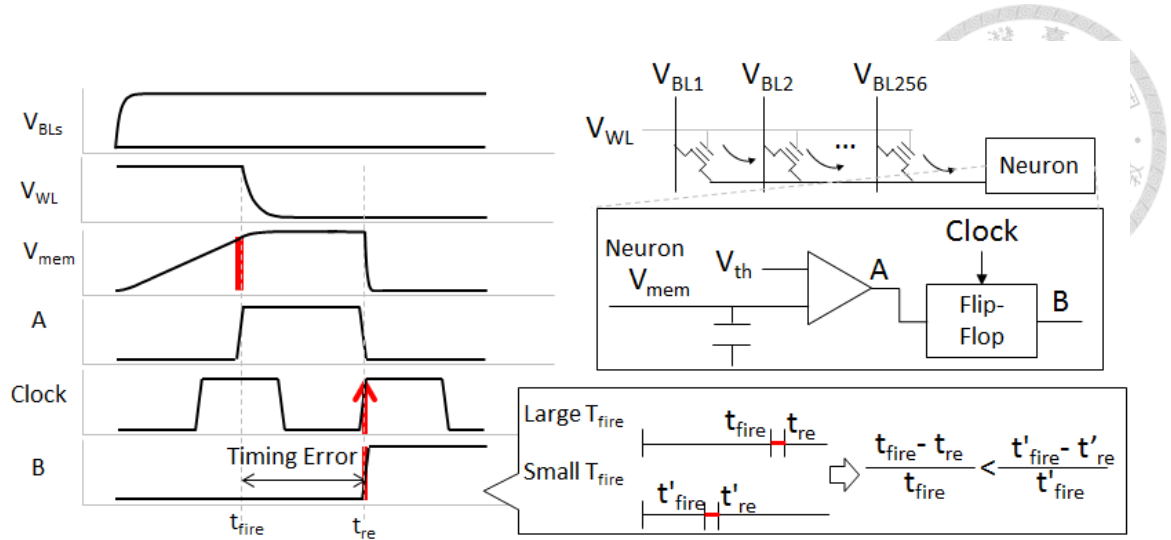


Figure 5.6: Waveform of SNN Circuit.

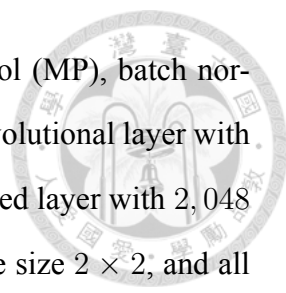
result (corresponds to inner product, see Eq. (5.1)) to increase by 1, whereas a flip from 1 to 0 causes a reduction by 1, which we denote with the flipping error ϵ_{FLIP} . With this flipping scheme, in BNNs, we can describe the errors with $\sum_i W_i X_i \pm \epsilon_{FLIP} > T$. Due to the similarity of the errors, we expect that the method for increasing general error resiliency of BNNs in [63], i.e., using the modified hinge loss in combination with bit flip injection with a certain flipping probability p_{FLIP} , will allow reduction of capacitor size in SNNs with minimal accuracy cost.

5.4 Evaluation

To evaluate the impact of binarization in SNNs, we consider as a baseline a stochastic input NN with multi-bit weights, given C_{mem} size and a certain inference accuracy. Our goal is to evaluate to what extent binarization combined with error resiliency optimization can reduce capacitor size, energy, and latency while sustaining inference accuracy.

5.4.1 Experiment Setup

For evaluation, we utilize the FashionMNIST dataset to prove our primary concept. The NNs have two convolutional and two fully connected layers. As the first layer, the



NNs have a convolutional layer with 64 filters, followed by maxpool (MP), batch normalization (BN), and activation (A). This is followed by another convolutional layer with 64 filters, then by MP-BN-A. The subsequent layer is a fully connected layer with 2,048 neurons, then BN-A. The output layer has 10 neurons. All MPs have size 2×2 , and all filter sizes are $3 \times 3 \times depth$. In the case of BNNs, the A-functions return binarized values, while in the 4-bit weights baseline model, ReLU is used. Both the BNNs and 4-bit NNs have the same architecture, except for the activation function and weight precisions.

For the BNNs, we use the modified hinge loss (MHL) with the hyperparameter $b = 128$ (see [63]). We use BNNs that were trained in two separate ways. One BNN was trained only with the MHL. The other BNN was optimized for error resiliency by injecting a probability of error $p_{FLIP} = 10\%$ per binary weight. We use batch sizes of 256 and initial learning rates of 10^{-3} . The learning rate is decreased exponentially every 25 epochs by 50 percent. We used the Adam optimizer [163] for 200 training epochs. For the 4-bit model, which serves as the baseline to compare the binarized SNNs, we use quantization-aware training. The number of training epochs is 33, as it reaches full convergence earlier. The initial learning rate is 0.01, which we decay by 0.1 every tenth epoch. We use the Adam optimizer here as well.

The stochastic model of SNNs for the binarized and 4-bit case is built from samples of circuit simulation. Samples are pairs of fire time and inner product generated from randomly generated input and weight vectors. The stochastic model is provided in the SNN evaluation tool. The crossbar size is set to be 256×256 , which can operate 128 inputs in parallel.

The memory technology used for evaluation is FeFET [111]. The cell architecture is simulated based on the measured data. The design scenario of high polarization (PR) and coercive field (EC) is adopted because of its high ON-OFF ratio. In this design scenario, the ON-state cell current is 10uA, operating at $V_g = 0.6V$ with the ON-OFF ratio over 1000.

Component	Spec	Energy (pJ)
Counter [180]	1.4 pJ/cycle	1.4 GRT/25ns
Adder (8 bits) [181]	0.16 pJ/spike	40
Crossbar+Neuron	$2V_{BL}V_{th}C_{mem}$ pJ/spike	$512V_{BL}V_{th}C_{mem}$
Time-to-digit [182]	107.5 pJ/spike	27520
RNG (8bits) [183]	135.76 pJ/cycle	135.76 GRT/25ns

Table 5.1: Energy Configurations of SNN Macro.

SNN Model	SF=1GHz	SF=2GHz
4-bits NN	15.8pF	15.8pF
BNN	12.6pF	10pF
ER-BNN	10pF	7.9pF

Table 5.2: The C_{mem} Size to Sustain Inference Accuracy ≥ 0.88

The circuit energy configurations are listed in Table 5.1. The role of each component is described in Subsec. 5.2.2.

5.4.2 Experiment Results

In the following, we first reduce the membrane capacitor in BNNs and 4-bit NNs without employing any countermeasures and show that the errors (from information loss and small sensing frequency) become large and drastically impact inference accuracy. As the first countermeasure, we increase the sensing frequency to reduce the sensing errors. We then evaluate to which extent the membrane capacitor can be reduced when we fully exploit the error resiliency of ER-BNNs. Finally, we compare the latency and energy consumption of the 4-bit model, BNN, and ER-BNN with maximum capacitor size reduction under an accuracy goal.

Information loss: To ideally have no error interference by the limited sampling frequency, in Fig. 5.7, we set the sampling frequency to infinity. In the figure, we show the errors from information loss (see Sec. 5.3.4) for the BNNs in subplot (a) and for the 4-bit case in subplots (b) - (e). We observe that in BNNs, the errors from information loss are always zero in the tested cases. This is due to the small charge in the capacitor when BNNs

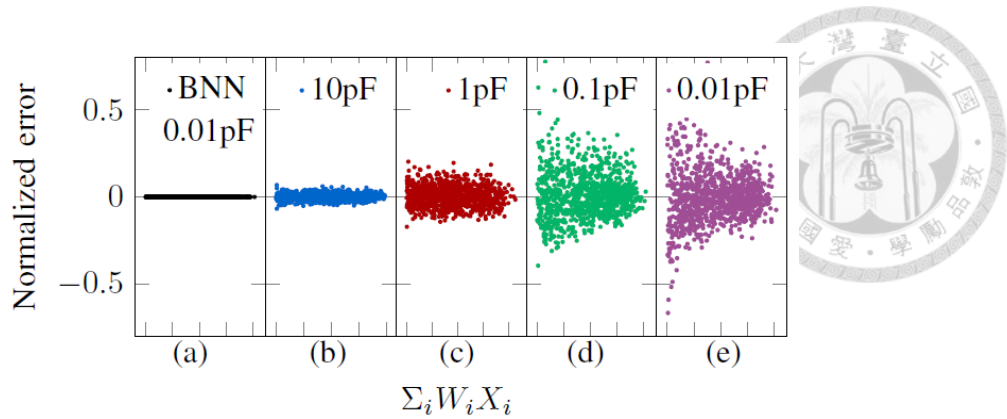


Figure 5.7: Normalized Error from Information Loss

are run. We do not show the errors for all capacitor sizes for BNNs since the errors are always zero. In the 4-bit case ((b) - (e)), the errors from information loss get larger when the capacitor size is reduced. There are larger charges in the 4-bit case than in BNNs since, in the 4-bit case, multiple samples are needed to estimate the inner product. Combined with a small membrane capacitor, early firings will be induced, leading to information loss.

Capacitor optimization under sensing frequency: Here, we focus on errors caused by limited sampling frequency for the case of BNNs since BNNs do not experience errors from information loss from small capacitors and, therefore, provide a better initial point for optimization. In Fig. 5.8a, we show how increasing the sensing frequency in BNNs from 1 GHz to 2 GHz allows capacitor size reduction while sustaining accuracy at a high level. In Fig. 5.8b, we show that the normalized computing error in BNNs can be reduced by increasing sensing frequency. The reason is the increase of sensing frequency reduces the upper bound of timing error described in Fig. 5.6.

Capacitor optimization under ER-BNNs: Although deploying BNNs on SNNs can reduce the size of membrane capacitors with high sensing frequency, there are fundamental limits, which we consider here to be 2 GHz. To enable further reduction of membrane capacitor size without increasing sensing frequency, we aim to exploit the error resiliency of BNNs to tolerate the sensing errors. We show the benefit of ER-BNNs in Fig. 5.9a. In the ER-BNN, the capacitor size can be reduced by 20% compared to the BNN model without ER. In Fig. 5.9b, we plot the sensing frequency error for the case in which a

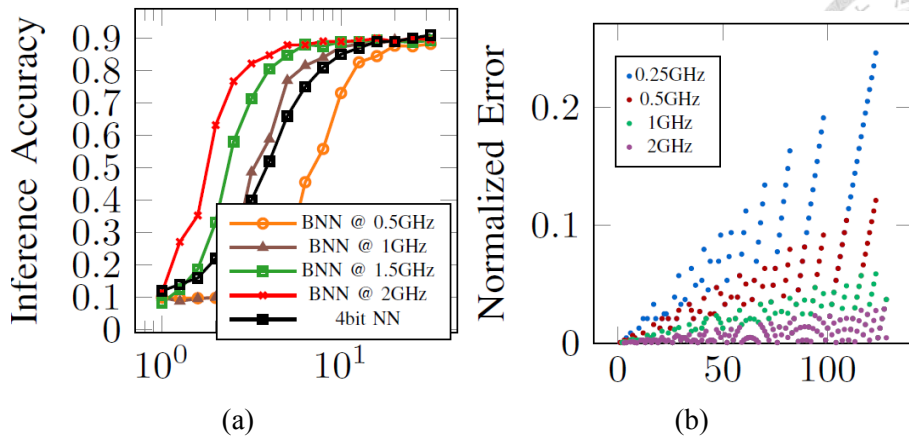


Figure 5.8: Effect of Sensing Frequency (SF)

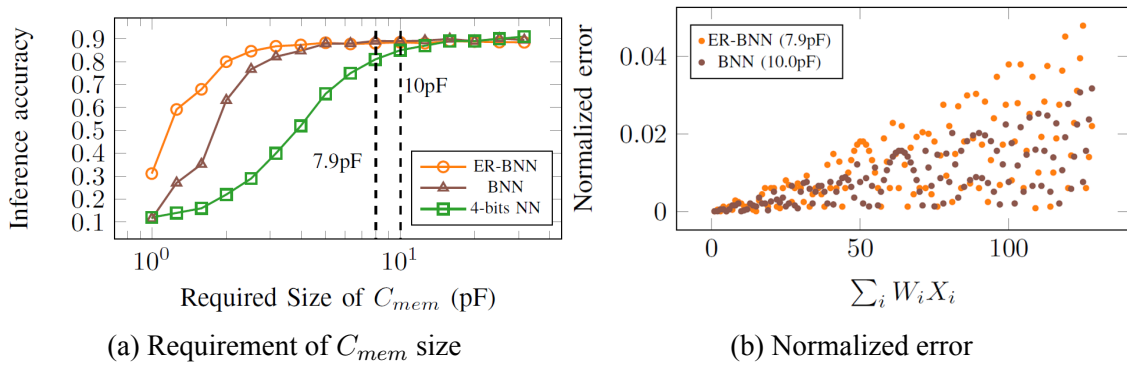
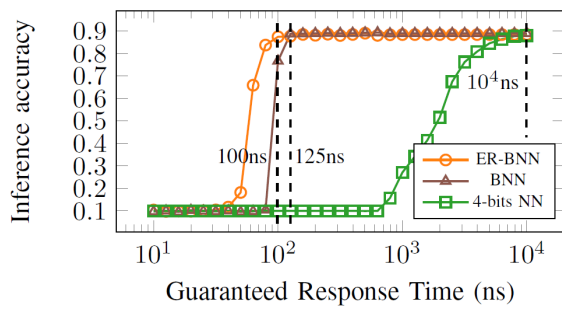


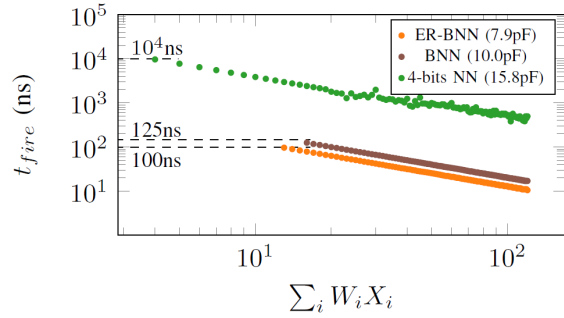
Figure 5.9: Noise Tolerance with Sensing Frequency (SF) 2GHz

capacitor of size 7.9 pF is selected such that the accuracy is above 0.88. Although the sensing errors in the case with 7.9 pF are larger than with 10 pF, the ER-BNN can tolerate them. As a result, the classification accuracy is sustained for smaller capacitor sizes, where the size of C_{mem} for ER-BNN is 50% smaller than that for the 4-bit model.

Latency improvement: The guaranteed response time (GRT) over accuracy is shown in Fig. 5.10a. The BNN can sustain accuracy with a smaller GRT than the 4-bit model, while the ER-BNN can sustain high accuracy with a smaller GRT than the BNN with no ER. In Fig. 5.10b, the GRT for the 4-bit model is higher. The reason is that a lower GRT causes more information loss than BNNs. In some cases, the expected firing time may exceed GRT, which is assumed to be a no-fire. The 4-bit model cannot sustain accuracy in these cases due to high information loss. In summary, the GRT of the different models are shown in Fig. 5.11a. The BNN has two orders of magnitude in GRT improvement compared to the 4-bit model, while the ER-BNN shows 20% GRT improvement compared

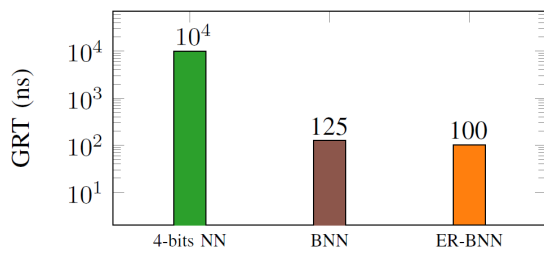


(a) Effects of guarantee response time

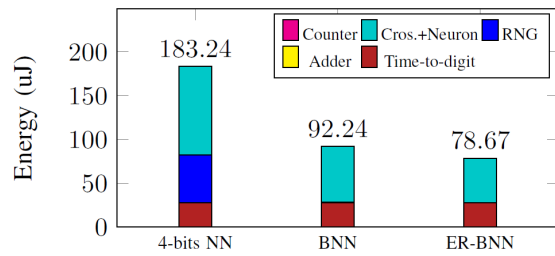


(b) t_{fire} plots considering GRT

Figure 5.10: Comparisons of Guaranteed Response Time (GRT).



(a) GRT for Models



(b) Energy for 256 Neurons

Figure 5.11: Computing Efficiency for Each Model

to the BNN without ER.

Energy saving: The energy configuration of our considered system is shown in Tab. 5.1. The energy used in each model is as shown in Fig. 5.11b. The result indicates that the required energy to generate a spike in BNNs without ER is 36.7% less than the 4-bit model, while the ER-BNN can reduce by 57% of energy compared to the 4-bit model.





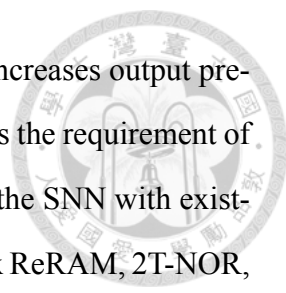
Chapter 6 Conclusions

The spiking neural network is a brain-inspired probabilistic inference machine where spiking neurons connect and communicate via binary spikes. We propose a synergy of computing-in-memory-based architecture and model optimization to enhance processing speed. To ensure reliable and superior processing speeds, we delve into reliability analysis, architecture design, and model-level optimization in this dissertation. For reliable analysis, we assess the noise from various memory devices in image classification and optimization-solving applications, finding that devices with high ON-OFF ratios and adjustable currents are preferable. For architecture design, we tackle the low process throughput utilization rate with three designs, including a 1D-systolic array, a time coarsening unit, and a skipping processing scheme. For model optimization, we reveal that deploying an error-resilient binary neural network to the spiking neural network accelerator surpasses the image classification throughput of the 4-bit neural model. We present results on reliability evaluation considering circuit constraints (Chapter 3), acceleration through granularity reduction and structured sparsity (Chapter 5), and achieving superior throughput with noise-tolerated binary neural networks (Chapter 4).

6.1 Memory Device Reliability Analysis

6.1.1 Reliability on Image Classification

This thesis provides a reliable perspective on the accuracy of SNNs using NVM synapses. Both theoretical analysis and simulation are presented. The theoretical analysis



shows that the membrane capacitor can be used as a modulator that increases output precision with the cost of computing energy. The low output current eases the requirement of a large membrane capacitor. We tested our hypothesis by simulating the SNN with existing memory technologies, including NOR Flash, WO_x ReRAM, HfO_x ReRAM, 2T-NOR, and Ferroelectric FET (FeFET). The simulation results show that low cell current (<10uA) and high ON-OFF ratio (>1000) are required to preserve the accuracy using an affordable membrane capacitor of 1pF. Moreover, the normalized standard deviation of ON-current should be < 10% to ensure network accuracy. It is recommended that the NOR-type Flash and FeFET with the large ON-OFF current ratios and tunable cell currents <10uA are good candidates for realizing accurate SNN circuits.

6.1.2 Reliability on Solving Optimization Problems

Constrained Satisfaction Problems (CSPs) happen in a wide range of applications. The spiking neural network (SNN) is a viable framework for solving CSPs. Combining non-volatile memory (NVM) arrays with analog computations provides high computational parallelism and area efficiency. Yet, reliability issues such as current variation, finite ON-OFF ratio, and current drift affect the success rate of finding constraint-fitted solutions under finite membrane capacitance in the SNN circuit. We simulated SNN circuits with memory technologies 2T-NOR, WO_x, HfO_x resistive memory, and FeFET to evaluate the effect of reliability issues on the success rate. Throughout the simulations, 2T-NOR and FeFET-High with normalized standard deviation (< 5%) and ON-OFF > 1000 are good candidates to preserve a success rate of finding constraint-fitted solutions. Thus, they are good candidates for realizing synapses for analog computing at room temperature. However, as the temperature increases from 300K to 358K, the ON-OFF ratio reduction of FeFET and the ON-state current drift of 2T-NOR drop the success rate between 20% and 90% for solving TSP and Sudoku. The requirements of capacitor size and design to tolerate the effects of temperature need to be explored in future work.

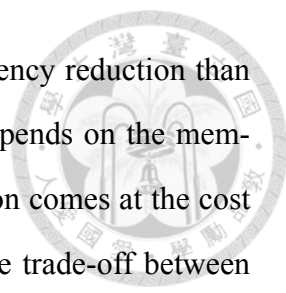
6.2 Architecture Design Optimized for Solving Optimization Problems



Optimization problems nowadays have a wide variety of applications. There is a high demand for platforms that can effectively solve optimization problems. The brain-inspired annealing machine offers a novel approach to meet this requirement. However, the problem-solving time is bounded by restricted synapse operation computation or the movement of weights. The proposed architecture utilizes a 3D-NOR Flash as a synaptic array, outperforming processing throughput and negating the need for weight movement. To further improve processing throughput, we design a finite-disturbance spike alignment scheme to reduce processing cycles that reduces the processing cycle from spike sparsity, leverage the CuA technique to realize 1-D systolic arrays, which equivalently reduce the input granularity, and propose a runtime pruning scheme to skip the synapse operation of stable neurons. The result shows that the proposed Neureka provides 3.1x, 1.8x, and 2.2x faster than prior digital-based SNN processors for applying MAX-CUT, SUDOKU, and LASSO, respectively. Neureka, with eight cores, also performs 6.6x, 1.8x, and 3.0x faster than other heuristic methods processed on high-end computation devices while maintaining the same solving quality.

6.3 Binary Spiking Neural Network

We studied the impact of deploying error-resilient BNNs (ER-BNNs) on analog implementations of SNNs. We focused on the reduction of the membrane capacitor size since it constitutes one of the major bottlenecks in analog SNNs, determining inference accuracy, energy usage, latency, and area. By analyzing the properties of analog SNN circuits, we showed that binarization allows capacitor size reduction with less accuracy cost than in multi-bit SNNs since binarization leads to deterministic inputs and less capacitor charge. We also established the connection between the latency and the membrane capac-

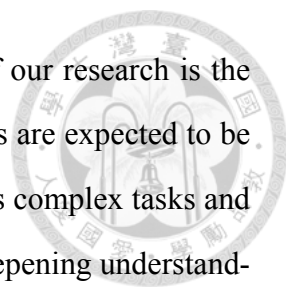


itor size, from which we deduced that binarization allows higher latency reduction than in multi-bit cases. We also showed that the energy consumption depends on the membrane capacitor size. However, the membrane capacitor size reduction comes at the cost of timing errors, for which we developed a model for evaluating the trade-off between membrane capacitor size and the SNN inference accuracy. As a countermeasure to the errors, we optimize BNNs for ER to tolerate the errors while sustaining high accuracy. Our experiments evaluated the trade-off between the capacitor size reduction and the errors. Our results indicate that, compared to 4-bit SNNs, deploying ER-BNNs as SNNs leads to 50% and 57% reduction of capacitor size and energy, respectively, and two orders of magnitude in improvement in latency, while high inference accuracy is sustained.

6.4 Future Works

A pivotal component of our future work involves the refinement and expansion of the spiking neural behavior model. This model, crucial for system-level simulations, has been our cornerstone in understanding and emulating neural network behaviors. The next phase will involve the implementation of this model using a field-programmable analog array (FPAA). The use of FPAA will facilitate real-time emulation, a crucial step toward achieving more dynamic and responsive spiking neural network systems. The development of an advanced emulator is a key objective. This emulator will not only serve as a proof-of-concept for the theoretical models we have developed but also enable the emulation of large-scale neural models. Such a tool will be invaluable in testing and refining neural network designs before their practical application, thereby reducing the time and resources needed for development.

One of the most formidable challenges in neural network research is replicating human-like abstract thinking and reasoning. Our future endeavors will focus on bridging this gap. While our current work has made strides in this direction, there remains a substantial amount of research and experimentation necessary to match and eventually



exceed human brain capabilities in this area. The long-term goal of our research is the development of human-like general intelligence agents. These agents are expected to be more energy-efficient and time-effective, capable of handling various complex tasks and challenges. Future efforts will be directed towards integrating our deepening understanding of brain functions into the design of these agents. This integration is anticipated to revolutionize the way we approach problem-solving and decision-making processes in artificial intelligence.

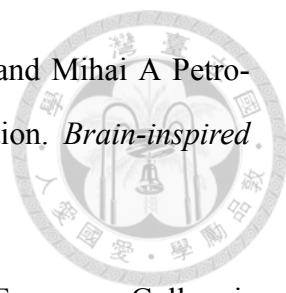
In conclusion, the path ahead is both challenging and exhilarating. As we delve deeper into the mysteries of the brain and its functioning, we anticipate groundbreaking advancements in neural network technology. Our journey is just beginning, and the potential for what can be achieved is immense. This future work will not only contribute significantly to the field of artificial intelligence but also has the potential to profoundly impact various aspects of society and industry.




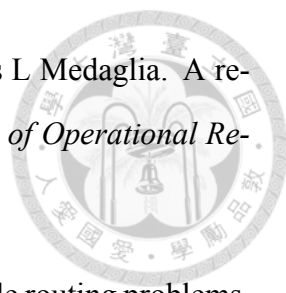



References

- [1] Iulia M Comsa, Krzysztof Potempa, Luca Versari, Thomas Fischbacher, Andrea Gesmundo, and Jyrki Alakuijala. Temporal coding in spiking neural networks with alpha synaptic function. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8529–8533. IEEE, 2020.
- [2] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, Brian Taba, Michael Beakes, Bernard Brezzo, Jente B. Kuang, Rajit Manohar, William P. Risk, Bryan Jackson, and Dharmendra S. Modha. 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.
- [3] Ben Varkey Benjamin, Peiran Gao, Emmett McQuinn, Swadesh Choudhary, Anand R Chandrasekaran, Jean-Marie Bussat, Rodrigo Alvarez-Icaza, John V Arthur, Paul A Merolla, and Kwabena Boahen. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5):699–716, 2014.
- [4] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

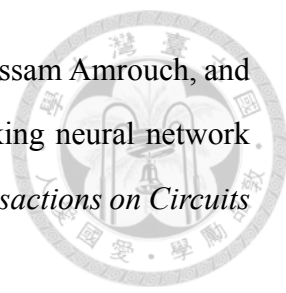
- 
- [5] Andreas Baumbach, Sebastian Billaudelle, Virginie Sabado, and Mihai A Petrovici. Brainscales: Greater versatility for neuromorphic emulation. *Brain-inspired Computing*, page 15, 2021.
- [6] Eustace Painkras, Luis A Plana, Jim Garside, Steve Temple, Francesco Galluppi, Cameron Patterson, David R Lester, Andrew D Brown, and Steve B Furber. Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation. *IEEE Journal of Solid-State Circuits*, 48(8):1943–1953, 2013.
- [7] Surya Narayanan, Ali Shafiee, and Rajeev Balasubramonian. Inxs: Bridging the throughput and energy gap for spiking neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2451–2459. IEEE, 2017.
- [8] Sonali Singh, Anup Sarma, Nicholas Jao, Ashutosh Pattnaik, Sen Lu, Kezhou Yang, Abhronil Sengupta, Vijaykrishnan Narayanan, and Chita R Das. Nebula: A neuro-morphic spin-based ultra-low power architecture for snns and anns. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 363–376. IEEE, 2020.
- [9] Matthew Kay Fei Lee, Yingnan Cui, Thannirmalai Somu, Tao Luo, Jun Zhou, Wai Teng Tang, Weng-Fai Wong, and Rick Siow Mong Goh. A system-level simulator for rram-based neuromorphic computing chips. *ACM Transactions on Architecture and Code Optimization (TACO)*, 15(4):1–24, 2019.
- [10] Abhishek Moitra, Abhiroop Bhattacharjee, Runcong Kuang, Gokul Krishnan, Yu Cao, and Priyadarshini Panda. Spikesim: An end-to-end compute-in-memory hardware evaluation tool for benchmarking spiking neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.
- [11] Pai-Yu Chen and Shimeng Yu. Reliability perspective of resistive synaptic devices on the neuromorphic system performance. In *2018 IEEE International Reliability Physics Symposium (IRPS)*, pages 5C–4. IEEE, 2018.


- 
- [12] Lei Deng, Guanrui Wang, Guoqi Li, Shuangchen Li, Ling Liang, Maohua Zhu, Yujie Wu, Zheyu Yang, Zhe Zou, Jing Pei, et al. Tianjic: A unified and scalable chip bridging spike-based and continuous neural computation. *IEEE Journal of Solid-State Circuits*, 55(8):2228–2246, 2020.
- [13] Surya Narayanan, Karl Taht, Rajeev Balasubramonian, Edouard Giacomin, and Pierre-Emmanuel Gaillardon. Spinalflow: An architecture and dataflow tailored for spiking neural networks. In *ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 349–362. IEEE, 2020.
- [14] Jeong-Jun Lee, Wenrui Zhang, and Peng Li. Parallel time batching: Systolic-array acceleration of sparse spiking neural computation. In *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 317–330. IEEE, 2022.
- [15] Anshujit Sharma, Richard Afoakwa, Zeljko Ignjatovic, and Michael Huang. Increasing ising machine capacity with multi-chip architectures. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, pages 508–521, 2022.
- [16] Jeong-Jun Lee, Wenrui Zhang, Yuan Xie, and Peng Li. Saarsp: An architecture for systolic-array acceleration of recurrent spiking neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 18(4):1–23, 2022.
- [17] Jean-François Cordeau, Gilbert Laporte, Martin WP Savelsbergh, and Daniele Vigo. Vehicle routing. *Handbooks in operations research and management science*, 14:367–428, 2007.
- [18] Marshall Fisher. Vehicle routing. *Handbooks in operations research and management science*, 8:1–33, 1995.
- [19] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

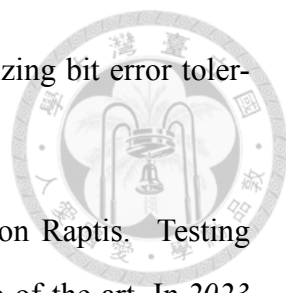
- 
- [20] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.
- [21] David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers & operations research*, 34(8):2403–2435, 2007.
- [22] Michel Gendreau, Gilbert Laporte, and René Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12, 1996.
- [23] Bruce L Golden, Subramanian Raghavan, Edward A Wasil, et al. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer, 2008.
- [24] Baofeng Sun, Yue Yang, Junyan Shi, and Lili Zheng. Dynamic pick-up and delivery optimization with multiple dynamic events in real-world environment. *IEEE Access*, 7:146209–146220, 2019.
- [25] Kevin Dorling, Jordan Heinrichs, Geoffrey G Messier, and Sebastian Magierowski. Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85, 2016.
- [26] Rui Wang, Xiangping Bryce Zhai, Yunlong Zhao, and Xuedong Zhao. Delivery optimization for unmanned aerial vehicles based on minimum cost maximum flow with limited battery capacity. In *International Conference on Wireless Algorithms, Systems, and Applications*, pages 77–85. Springer, 2021.
- [27] Sebastián Dávila, Martine Labbé, Vladimir Marianov, Fernando Ordóñez, and Frédéric Semet. Product line optimization with multiples sites. *Computers & Operations Research*, 148:105978, 2022.
- [28] Dimitris Bertsimas and Velibor V Mišić. Exact first-choice product line optimization. *Operations Research*, 67(3):651–670, 2019.
- [29] Stelios Tsafarakis, Yannis Marinakis, and Nikolaos Matsatsinis. Particle swarm


- 
- optimization for optimal product line design. *International Journal of Research in Marketing*, 28(1):13–22, 2011.
- [30] I Antonioli, P Guariente, T Pereira, L Pinto Ferreira, and FJG Silva. Standardization and optimization of an automotive components production line. *Procedia Manufacturing*, 13:1120–1127, 2017.
- [31] Diomidis Spinellis, Chrissoleon Papadopoulos, and J MacGregor Smith. Large production line optimization using simulated annealing. *International journal of production research*, 38(3):509–541, 2000.
- [32] N Rezg, X Xie, and Y Mati. Joint optimization of preventive maintenance and inventory control in a production line using simulation. *International Journal of Production Research*, 42(10):2029–2046, 2004.
- [33] Haoyu Cheng, Erich D Jarvis, Olivier Fedrigo, Klaus-Peter Koepfli, Lara Urban, Neil J Gemmell, and Heng Li. Haplotype-resolved assembly of diploid genomes without parental data. *Nature Biotechnology*, 40(9):1332–1335, 2022.
- [34] Qianxing Mo and Faming Liang. A hidden ising model for chip-chip data analysis. *Bioinformatics*, 26(6):777–783, 2010.
- [35] Natalia N Vtyurina, David Dulin, Margreet W Docter, Anne S Meyer, Nynke H Dekker, and Elio A Abbondanzieri. An ising model describes hysteresis in a novel form of cooperative binding. *Biophysical Journal*, 110(3):239a, 2016.
- [36] Jacek Majewski, Hao Li, and Jurg Ott. The ising model in physics and statistical genetics. *The American Journal of Human Genetics*, 69(4):853–862, 2001.
- [37] AS Boev, AS Rakitko, SR Usmanov, AN Kobzeva, IV Popov, VV Ilinsky, EO Kiktenko, and AK Fedorov. Genome assembly using quantum and quantum-inspired annealing. *Scientific Reports*, 11(1):13183, 2021.
- [38] Xumeng Li, F Alex Feltus, Xiaoqian Sun, Zijun Wang, and Feng Luo. A non-parameter ising model for network-based identification of differentially expressed

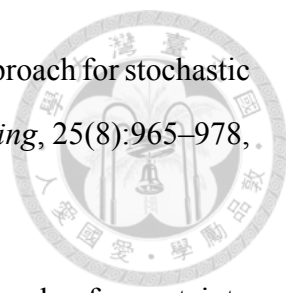
- genes in recurrent breast cancer patients. In *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 214–217. IEEE, 2010.
- [39] Audun Bakk and Johan S Høye. One-dimensional ising model applied to protein folding. *Physica A: Statistical Mechanics and its Applications*, 323:504–518, 2003.
- [40] Michail Yu Lobanov and Oxana V Galzitskaya. The ising model for prediction of disordered residues from protein sequence alone. *Physical biology*, 8(3):035004, 2011.
- [41] Bernd Rosenow, Vasiliki Plerou, Parameswaran Gopikrishnan, and H Eugene Stanley. Portfolio optimization and the random magnet problem. *Europhysics Letters*, 59(4):500, 2002.
- [42] Philippe Jorion. Portfolio optimization in practice. *Financial analysts journal*, 48(1):68–74, 1992.
- [43] Tunchan Cura. Particle swarm optimization approach to portfolio optimization. *Nonlinear analysis: Real world applications*, 10(4):2396–2406, 2009.
- [44] Michael Marzec. Portfolio optimization: Applications in quantum computing. *Handbook of High-Frequency Trading and Modeling in Finance*, pages 73–106, 2016.
- [45] Aayush Ankit, Abhronil Sengupta, Priyadarshini Panda, and Kaushik Roy. Resparc: A reconfigurable and energy-efficient architecture with memristive cross-bars for deep spiking neural networks. In *Proceedings of the 54th Annual Design Automation Conference (DAC)*, pages 1–6, 2017.
- [46] Yachen Xiang, Peng Huang, Runze Han, Chu Li, Kunliang Wang, Xiaoyan Liu, and Jinfeng Kang. Efficient and robust spike-driven deep convolutional neural networks based on nor flash computing array. *IEEE Transactions on Electron Devices (TED)*, 67(6):2329–2335, 2020.


- 
- [47] Ming-Liang Wei, Mikail Yayla, Shu-Yin Ho, Jian-Jia Chen, Hussam Amrouch, and Chia-Lin Yang. Impact of non-volatile memory cells on spiking neural network annealing machine with in-situ synapse processing. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2023.
- [48] Ming-Liang Wei, Hussam Amrouch, Cheng-Lin Sung, Hang-Ting Lue, Chia-Lin Yang, Keh-Chung Wang, and Chih-Yuan Lu. Robust brain-inspired computing: On the reliability of spiking neural network using emerging non-volatile synapses. In *International Reliability Physics Symposium (IRPS)*, pages 1–8. IEEE, 2021.
- [49] Xiaoling Xia, Cui Xu, and Bing Nan. Inception-v3 for flower classification. In *2017 2nd international conference on image, vision and computing (ICIVC)*, pages 783–787. IEEE, 2017.
- [50] Sasha Targ, Diogo Almeida, and Kevin Lyman. Resnet in resnet: Generalizing residual architectures. *arXiv preprint arXiv:1603.08029*, 2016.
- [51] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014.
- [52] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.
- [53] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.
- [54] Thomas Natschläger and Berthold Ruf. Pattern analysis with spiking neurons using delay coding. *Neurocomputing*, 26:463–469, 1999.
- [55] Seongsik Park, Seijoon Kim, Byungook Na, and Sungroh Yoon. T2fsnn: Deep

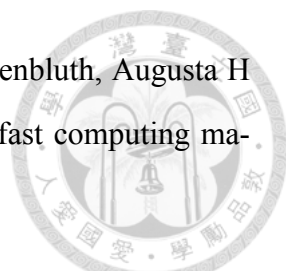
- 
- spiking neural networks with time-to-first-spike coding. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.
- [56] Kwabena Boahen. Dendrocentric learning for synthetic intelligence. *Nature*, 612(7938):43–50, 2022.
- [57] Geoffrey Portelli, John M Barrett, Gerrit Hilgen, Timothée Masquelier, Alessandro Maccione, Stefano Di Marco, Luca Berdondini, Pierre Kornprobst, and Evelyne Sernagor. Rank order coding: a retinal information decoding strategy revealed by large-scale multielectrode array retinal recordings. *eneuro*, 3(3), 2016.
- [58] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International joint conference on neural networks (IJCNN)*, pages 1–8. iee, 2015.
- [59] Daniel Auge, Julian Hille, Etienne Mueller, and Alois Knoll. A survey of encoding techniques for signal processing in spiking neural networks. *Neural Processing Letters*, 53(6):4693–4710, 2021.
- [60] Chunyu Yuan and Sos S Aghaian. A comprehensive review of binary neural network. *Artificial Intelligence Review*, pages 1–65, 2023.
- [61] Mingzhu Shen, Kai Han, Chunjing Xu, and Yunhe Wang. Searching for accurate binary neural architectures. In *Proceedings of the IEEE/CVF international conference on computer vision workshops*, pages 0–0, 2019.
- [62] Tifenn Hirtzlin, Marc Bocquet, Jacques-Olivier Klein, Etienne Nowak, Elisa Vianello, Jean Michel Portal, and Damien Querlioz. Outstanding bit error tolerance of resistive ram-based binarized neural networks. In *International Conference on Artificial Intelligence Circuits and Systems, AICAS*, pages 288–292, 2019.
- [63] Sebastian Buschjäger, Jian-Jia Chen, Kuan-Hsun Chen, Mario Günzel, Christian Hakert, Katharina Morik, Rodion Novkin, Lukas Pfahler, and Mikail Yayla.

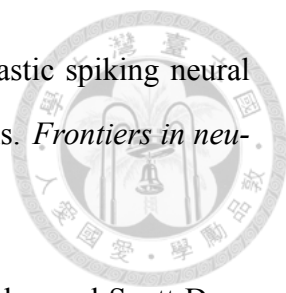
- 
- Margin-maximization in binarized neural networks for optimizing bit error tolerance. *DATE '21*, 2020.
- [64] Haralampos-G Stratigopoulos, Theofilos Spyrou, and Spyridon Raptis. Testing and reliability of spiking neural networks: A review of the state-of-the-art. In *2023 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 1–8. IEEE, 2023.
- [65] Fei Su, Chunsheng Liu, and Haralampos-G Stratigopoulos. Testability and dependability of ai hardware: Survey, trends, challenges, and perspectives. *IEEE Design & Test*, 2023.
- [66] Theofilos Spyrou, Sarah A El-Sayed, Engin Afacan, Luis A Camuñas-Mesa, Bernabé Linares-Barranco, and Haralampos-G Stratigopoulos. Reliability analysis of a spiking neural network hardware accelerator. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 370–375. IEEE, 2022.
- [67] Elena-Ioana Vatajelu, Giorgio Di Natale, and Lorena Anghel. Special session: Reliability of hardware-implemented spiking neural networks (snn). In *2019 IEEE 37th VLSI Test Symposium (VTS)*, pages 1–8. IEEE, 2019.
- [68] Rachmad Vidya Wicaksana Putra, Muhammad Abdullah Hanif, and Muhammad Shafique. Softsnn: Low-cost fault tolerance for spiking neural network accelerators under soft errors. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 151–156, 2022.
- [69] Wilkie Olin-Ammentorp, Karsten Beckmann, Catherine D Schuman, James S Plank, and Nathaniel C Cady. Stochasticity and robustness in spiking neural networks. *Neurocomputing*, 419:23–36, 2021.
- [70] Sebastian Schmitt, Johann Klähn, Guillaume Bellec, Andreas Grübl, Maurice Guetler, Andreas Hartel, Stephan Hartmann, Dan Husmann, Kai Husmann, Sebastian Jeltsch, et al. Neuromorphic hardware in the loop: Training a deep spiking network


- 
- on the brainscales wafer-scale system. In *2017 international joint conference on neural networks (IJCNN)*, pages 2227–2234. IEEE, 2017.
- [71] Dohun Kim, Guhyun Kim, Cheol Seong Hwang, and Doo Seok Jeong. ewb: Event-based weight binarization algorithm for spiking neural networks. *IEEE Access*, 9:38097–38106, 2021.
- [72] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.
- [73] Sen Lu and Abhronil Sengupta. Exploring the connection between binary and spiking neural networks. *Frontiers in Neuroscience*, 14, Jun 2020.
- [74] Catherine D. Schuman, J. Parker Mitchell, J. Travis Johnston, Maryam Parsa, Bill Kay, Prasanna Date, and Robert M. Patton. Resilience and robustness of spiking neural networks for neuromorphic systems. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10, 2020.
- [75] Elena-Ioana Vatajelu, Giorgio Di Natale, and Lorena Anghel. Special session: Reliability of hardware-implemented spiking neural networks (snn). In *2019 IEEE 37th VLSI Test Symposium (VTS)*, pages 1–8, 2019.
- [76] Theofilos Spyrou, Sarah A El-Sayed, Engin Afacan, Luis A Camuñas-Mesa, Bernabé Linares-Barranco, and Haralampos-G. Stratigopoulos. Neuron Fault Tolerance in Spiking Neural Networks. In *DATE 2021*, Grenoble (virtuel), France, February 2021.
- [77] Chris Eliasmith. *How to build a brain: A neural architecture for biological cognition*. OUP USA, 2013.
- [78] Yanping Huang and Rajesh P Rao. Neurons as monte carlo samplers: Bayesian inference and learning in spiking networks. *Advances in neural information processing systems*, 27, 2014.

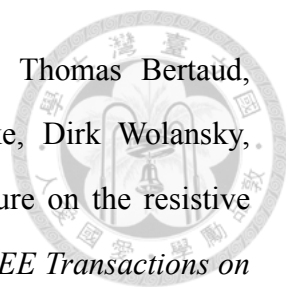
- 
- [79] Ola Friman, Gunnar Farneback, and C-F Westin. A bayesian approach for stochastic white matter tractography. *IEEE transactions on medical imaging*, 25(8):965–978, 2006.
- [80] David C Knill and Alexandre Pouget. The bayesian brain: the role of uncertainty in neural coding and computation. *TRENDS in Neurosciences*, 27(12):712–719, 2004.
- [81] Ting-Shuo Chou, Hirak J Kashyap, Jinwei Xing, Stanislav Listopad, Emily L Rounds, Michael Beyeler, Nikil Dutt, and Jeffrey L Krichmar. Carlsim 4: An open source library for large scale, biologically detailed spiking neural network simulation using heterogeneous clusters. In *International joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2018.
- [82] Birgit Kriener, Hkon Enger, Tom Tetzlaff, Hans E Plesser, Marc-Oliver Gewaltig, and Gaute T Einevoll. Dynamics of self-sustained asynchronous-irregular activity in random networks of spiking neurons with strong synapses. *Frontiers in computational neuroscience*, 8:136, 2014.
- [83] Shen-Fu Hsiao, Chia-Sheng Wen, and Ming-Yu Tsai. Low-cost design of reciprocal function units using shared multipliers and adders for polynomial approximation and newton raphson interpolation. In *International Symposium on Next Generation Electronics*, pages 40–43. IEEE, 2010.
- [84] Hande Alemdar, Vincent Leroy, Adrien Prost-Boucle, and Frédéric Pétrot. Ternary neural networks for resource-efficient ai applications. In *2017 international joint conference on neural networks (IJCNN)*, pages 2547–2554. IEEE, 2017.
- [85] Lei Deng, Peng Jiao, Jing Pei, Zhenzhi Wu, and Guoqi Li. Gxnor-net: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework. *Neural Networks*, 100:49–58, 2018.

- 
- [86] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Euro-pean conference on computer vision*, pages 525–542. Springer, 2016.
- [87] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omni-directionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*, 2023.
- [88] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 293–302, 2019.
- [89] Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. *Advances in neural information processing systems*, 33:18518–18529, 2020.
- [90] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, et al. Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning*, pages 11875–11886. PMLR, 2021.
- [91] Hannes Uppman. *On Some Combinatorial Optimization Problems: Algorithms and Complexity*. PhD thesis, Linköping University Electronic Press, 2015.
- [92] Fred Glover, Gary Kochenberger, and Yu Du. A tutorial on formulating and using qubo models. *arXiv preprint arXiv:1811.11538*, 2018.
- [93] Abraham P Punnen. Introduction to qubo. In *The Quadratic Unconstrained Binary Optimization Problem*, pages 1–37. Springer, 2022.

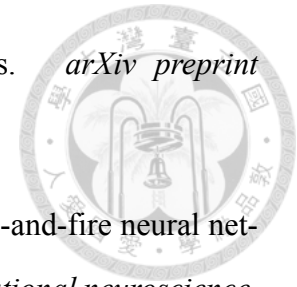
- 
- [94] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *Journal of chemical physics*, 21(6):1087–1092, 1953.
- [95] Wally R Gilks, Nicky G Best, and Keith KC Tan. Adaptive rejection metropolis sampling within gibbs sampling. *Journal of the royal statistical society: series c*, 44(4):455–472, 1995.
- [96] Takashi Takemoto, Masato Hayashi, Chihiro Yoshimura, and Masanao Yamaoka. A 2 x 30k-spin multi-chip scalable cmos annealing processor based on a processing-in-memory approach for solving large-scale combinatorial optimization problems. *IEEE Journal of Solid-State Circuits (JSSC)*, 55(1):145–156, 2019.
- [97] Takashi Takemoto, Masato Hayashi, Chihiro Yoshimura, and Masanao Yamaoka. A 2 x 30k-spin multichip scalable annealing processor based on a processing-in-memory approach for solving large-scale combinatorial optimization problems. In *International Solid-State Circuits Conference (ISSCC)*, pages 52–54. IEEE, 2019.
- [98] Ming-Chun Hong, Le-Chih Cho, Chih-Sheng Lin, Yu-Hui Lin, Po-An Chen, I-Ting Wang, Pei-Jer Tzeng, Shyh-Shyuan Sheu, Wei-Chung Lo, Chih-I Wu, et al. In-memory annealing unit (imau): Energy-efficient (2000 tops/w) combinatorial optimizer for solving travelling salesman problem. In *International Electron Devices Meeting (IEDM)*, pages 21–3. IEEE, 2021.
- [99] John J Hopfield and David W Tank. Computing with neural circuits: A model. *Science*, 233(4764):625–633, 1986.
- [100] SG Hu, Y Liu, Z Liu, TP Chen, JJ Wang, Q Yu, LJ Deng, Y Yin, and Sumio Hosaka. Associative memory realized by a reconfigurable memristive hopfield neural network. *Nature communications*, 6(1):1–8, 2015.
- [101] Zeno Jonke, Stefan Habenschuss, and Wolfgang Maass. Solving constraint satisfaction problems with networks of spiking neurons. *Frontiers in neuroscience*, 10:118, 2016.

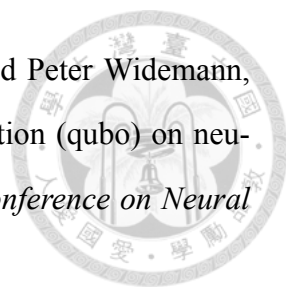
- 
- [102] Gabriel A Fonseca Guerra and Steve B Furber. Using stochastic spiking neural networks on spinnaker to solve constraint satisfaction problems. *Frontiers in neuroscience*, 11:714, 2017.
- [103] Chris Yakopcic, Nayim Rahman, Tanvir Atahary, Tarek M Taha, and Scott Douglas. Solving constraint satisfaction problems using the loihi spiking neuromorphic processor. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1079–1084. IEEE, 2020.
- [104] Sourav Dutta, Clemens Schafer, Jorge Gomez, Kai Ni, Siddharth Joshi, and Suman Datta. Supervised learning in all fetet-based spiking neural network: Opportunities and challenges. *Frontiers in Neuroscience*, 14, 2020.
- [105] David Heeger et al. Poisson model of spike generation. *Handout, University of Stanford*, 5(1-13):76, 2000.
- [106] Hang-Ting Lue, Po-Kai Hsu, Ming-Liang Wei, Teng-Hao Yeh, Pei-Ying Du, Wei-Chen Chen, Keh-Chung Wang, and Chih-Yuan Lu. Optimal design methods to transform 3d nand flash into a high-density, high-bandwidth and low-power non-volatile computing in memory (nvcim) accelerator for deep-learning neural networks (dnn). In *2019 IEEE International Electron Devices Meeting (IEDM)*, pages 38–1. IEEE, 2019.
- [107] Tianqi Tang, Lixue Xia, Boxun Li, Rong Luo, Yiran Chen, Yu Wang, and Huazhong Yang. Spiking neural network with rram: Can we use it for real-world application? In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 860–865. IEEE, 2015.
- [108] Wei-Chih Chien, Ming-Hsiu Lee, Feng-Ming Lee, Yu-Yu Lin, Hsiang-Lan Lung, Kuang-Yeu Hsieh, and Chih-Yuan Lu. Multi-level 40nm wo x resistive memory with excellent reliability. In *2011 International electron devices meeting*, pages 31–5. IEEE, 2011.

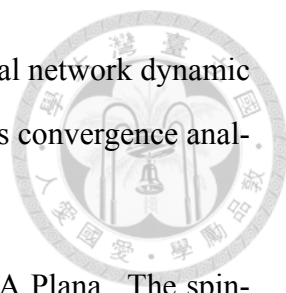
- 
- [109] Xiaokang Li, Baotong Zhang, Bowen Wang, Xiaoyan Xu, Yuancheng Yang, Shuang Sun, Qifeng Cai, Shijie Hu, Xia An, Ming Li, et al. Low power and high uniformity of hfox-based rram via tip-enhanced electric fields. *Science China Information Sciences*, 62(10):1–7, 2019.
- [110] Hang-Ting Lue, Tzu-Hsuan Hsu, Teng-Hao Yeh, Wei-Chen Chen, Chieh Roger Lo, Chia-Tze Huang, Guan-Ru Lee, Chia-Jung Chiu, Keh-Chung Wang, and Chih-Yuan Lu. A vertical 2t nor (v2t) architecture to enable scaling and low-power solutions for nor flash technology. In *2020 IEEE Symposium on VLSI Technology*, pages 1–2. IEEE, 2020.
- [111] Kai Ni, Aniket Gupta, Om Prakash, Simon Thomann, X Sharon Hu, and Hussam Amrouch. Impact of extrinsic variation sources on the device-to-device variation in ferroelectric fet. In *International Reliability Physics Symposium (IRPS)*, pages 1–5. IEEE, 2020.
- [112] Wei-Hao Chen, Kai-Xiang Li, Wei-Yu Lin, Kuo-Hsiang Hsu, Pin-Yi Li, Cheng-Han Yang, Cheng-Xin Xue, En-Yu Yang, Yen-Kai Chen, Yun-Sheng Chang, et al. A 65nm 1mb nonvolatile computing-in-memory rram 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.
- [113] Gerhard Reinelt. Tsplib95. *Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR), Heidelberg*, 338:1–16, 1995.
- [114] Ming-Liang Wei, Mikail Yayla, Shu-Yin Ho, Jiap-Jia Chen, Chia-Lin Yang, and Hussam Amrouch. Binarized snns: Efficient and error-resilient spiking neural networks through binarization. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2021.
- [115] Aniket Gupta, Kai Ni, Om Prakash, X Sharon Hu, and Hussam Amrouch. Temperature dependence and temperature-aware sensing in ferroelectric fet. In *IEEE International Reliability Physics Symposium (IRPS)*, pages 1–5. IEEE, 2020.

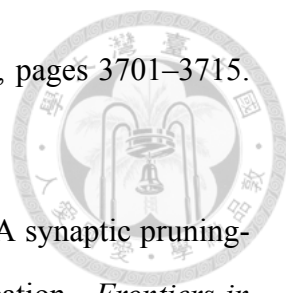
- 
- [116] Christian Walczyk, Damian Walczyk, Thomas Schroeder, Thomas Bertaud, Magorzata Sowinska, Mindaugas Lukosius, Mirko Fraschke, Dirk Wolansky, Bernd Tillack, Enrique Miranda, et al. Impact of temperature on the resistive switching behavior of embedded hfo₂-based rram devices. *IEEE Transactions on Electron Devices (TED)*, 58(9):3124–3131, 2011.
- [117] Xinjie Guo, F Merrikh Bayat, Mirko Prezioso, Y Chen, B Nguyen, N Do, and Dmitri B Strukov. Temperature-insensitive analog vector-by-matrix multiplier based on 55 nm nor flash memory cells. In *Custom Integrated Circuits Conference (CICC)*, pages 1–4. IEEE, 2017.
- [118] Takahiro Inagaki, Yoshitaka Haribara, Koji Igarashi, Tomohiro Sonobe, Shuhei Tamate, Toshimori Honjo, Alireza Marandi, Peter L McMahon, Takeshi Umeki, Koji Enbutsu, et al. A coherent ising machine for 2000-node optimization problems. *Science*, 354(6312):603–606, 2016.
- [119] Masanao Yamaoka, Takuya Okuyama, Masato Hayashi, Chihiro Yoshimura, and Takashi Takemoto. Cmos annealing machine: an in-memory computing accelerator to process combinatorial optimization problems. In *IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–8. IEEE, 2019.
- [120] Giacomo Indiveri. A low-power adaptive integrate-and-fire neuron circuit. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, volume 4, pages IV–IV. IEEE, 2003.
- [121] Cheng-Lin Sung, Hang-Ting Lue, Ming-Liang Wei, Shu-Yin Ho, Han-Wen Hu, Pei-Ying Du, Wei-Chen Chen, Chieh Roger Lo, Teng-Hao Yeh, Keh-Chung Wang, et al. A novel super-steep slope (~ 0.015 mv/dec) gate-controlled thyristor (gct) functional memory device to support the integrate-and-fire circuit for spiking neural networks. In *2020 IEEE International Electron Devices Meeting (IEDM)*, pages 21–3. IEEE, 2020.
- [122] Valentin Schmutz, Johanni Brea, and Wulfram Gerstner. Convergence of

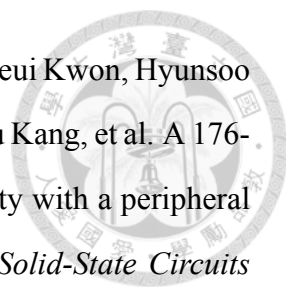
- redundancy-free spiking neural networks to rate networks. *arXiv preprint arXiv:2303.05174*, 2023.
- [123] Bruno Cessac and Thierry Viéville. On dynamics of integrate-and-fire neural networks with conductance based synapses. *Frontiers in computational neuroscience*, 2:228, 2008.
- [124] Wulfram Gerstner. Integrate-and-fire neurons and networks. *The handbook of brain theory and neural networks*, 2:577–581, 2002.
- [125] Manon Dampfhofer, Thomas Mesquida, Alexandre Valentian, and Lorena Anghel. Investigating current-based and gating approaches for accurate and energy-efficient spiking recurrent neural networks. In *International Conference on Artificial Neural Networks*, pages 359–370. Springer, 2022.
- [126] Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel A Fonseca Guerra, Prasad Joshi, Philipp Plank, and Sumedh R Risbud. Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5):911–934, 2021.
- [127] Giacomo Pedretti, Piergiulio Mannocci, Shahin Hashemkhani, Valerio Milo, Octavian Melnic, Elisabetta Chicca, and Daniele Ielmini. A spiking recurrent neural network with phase-change memory neurons and synapses for the accelerated solution of constraint satisfaction problems. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 6(1):89–97, 2020.
- [128] Lars Buesing, Johannes Bill, Bernhard Nessler, and Wolfgang Maass. Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS computational biology*, 7(11):e1002211, 2011.
- [129] Prasanna Date, Robert Patton, Catherine Schuman, and Thomas Potok. Efficiently embedding qubo problems on adiabatic quantum computers. *Quantum Information Processing*, 18:1–31, 2019.





- 
- [130] Md Zahangir Alom, Brian Van Essen, Adam T Moody, David Peter Widemann, and Tarek M Taha. Quadratic unconstrained binary optimization (qubo) on neuromorphic computing system. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3922–3929. IEEE, 2017.
- [131] Endre Boros, Peter L Hammer, and Gabriel Tavares. Local search heuristics for quadratic unconstrained binary optimization (qubo). *Journal of Heuristics*, 13:99–132, 2007.
- [132] Iain Dunning, Swati Gupta, and John Silberholz. What works best when? a systematic evaluation of heuristics for max-cut and qubo. *INFORMS Journal on Computing*, 30(3):608–624, 2018.
- [133] Kyle Henke, Michael Teti, Garrett Kenyon, Ben Migliori, and Gerd Kunde. Apples-to-spikes: The first detailed comparison of lasso solutions generated by a spiking neuromorphic processor. In *Proceedings of the International Conference on Neuromorphic Systems 2022*, pages 1–8, 2022.
- [134] Samuel Shapero, Christopher Rozell, and Paul Hasler. Configurable hardware integrate and fire neurons for sparse approximation. *Neural Networks*, 45:134–143, 2013.
- [135] Ping Tak Peter Tang, Tsung-Han Lin, and Mike Davies. Sparse coding by spiking neural networks: Convergence theory and computational results. *arXiv preprint arXiv:1705.05475*, 2017.
- [136] Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429, 2006.
- [137] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
- [138] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

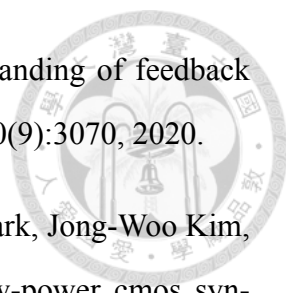
- 
- [139] Chenhui Zhao, Zenan Huang, and Donghui Guo. Spiking neural network dynamic system modeling for computation of quantum annealing and its convergence analysis. *Quantum Information Processing*, 20(2):1–16, 2021.
- [140] Steve B Furber, Francesco Galluppi, Steve Temple, and Luis A Plana. The spinaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.
- [141] Ming-Liang Wei, Hang-Ting Lue, Shu-Yin Ho, Yen-Po Lin, Tzu-Hsuan Hsu, Chih-Chang Hsieh, Yung-Chun Li, Teng-Hao Yeh, Shih-Hung Chen, Yi-Hao Jhu, et al. Analog computing in memory (cim) technique for general matrix multiplication (gemm) to support deep neural network (dnn) and cosine similarity search computing using 3d and-type nor flash devices. In *International Electron Devices Meeting (IEDM)*, pages 33–3. IEEE, 2022.
- [142] Mike O’Connor, Niladrish Chatterjee, Donghyuk Lee, John Wilson, Aditya Agrawal, Stephen W Keckler, and William J Dally. Fine-grained dram: Energy-efficient dram for extreme bandwidth systems. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 41–54, 2017.
- [143] Javier Iglesias, Jan Eriksson, François Grize, Marco Tomassini, and Alessandro EP Villa. Dynamics of pruning in simulated large-scale spiking neural networks. *Biosystems*, 79(1-3):11–20, 2005.
- [144] Nitin Rathi, Priyadarshini Panda, and Kaushik Roy. Stdp-based pruning of connections and weight quantization in spiking neural networks for energy-efficient recognition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(4):668–677, 2018.
- [145] Yuhan Shi, Leon Nguyen, Sangheon Oh, Xin Liu, and Duygu Kuzum. A soft-pruning method applied during training of spiking neural networks for in-memory computing applications. *Frontiers in neuroscience*, 13:405, 2019.
- [146] Yanqi Chen, Zhaofei Yu, Wei Fang, Zhengyu Ma, Tiejun Huang, and Yonghong Tian. State transition of dendritic spines improves learning of sparse spiking neural

- 
- networks. In *International Conference on Machine Learning*, pages 3701–3715. PMLR, 2022.
- [147] Faramarz Faghihi, Hany Alashwal, and Ahmed A Moustafa. A synaptic pruning-based spiking neural network for hand-written digits classification. *Frontiers in Artificial Intelligence*, 5:680165, 2022.
- [148] Thao NN Nguyen, Bharadwaj Veeravalli, and Xuanyao Fong. Connection pruning for deep spiking neural networks with on-chip learning. In *International Conference on Neuromorphic Systems 2021*, pages 1–8, 2021.
- [149] Ruokai Yin, Youngeun Kim, Yuhang Li, Abhishek Moitra, Nitin Satpute, Anna Hambitzer, and Priyadarshini Panda. Workload-balanced pruning for sparse spiking neural networks. *arXiv preprint arXiv:2302.06746*, 2023.
- [150] Ruizhi Chen, Hong Ma, Shaolin Xie, Peng Guo, Pin Li, and Donglin Wang. Fast and efficient deep sparse multi-strength spiking neural networks with dynamic pruning. In *2018 international joint conference on neural networks (ijcnn)*, pages 1–8. IEEE, 2018.
- [151] LW Meng, GC Qiao, XY Zhang, J Bai, Y Zuo, PJ Zhou, Y Liu, and SG Hu. An efficient pruning and fine-tuning method for deep spiking neural network. *Applied Intelligence*, pages 1–14, 2023.
- [152] Bing Han, Feifei Zhao, Yi Zeng, and Wenxuan Pan. Adaptive sparse structure development with pruning and regeneration for spiking neural networks. *arXiv preprint arXiv:2211.12219*, 2022.
- [153] Chang Siau, Kwang-Ho Kim, Seungpil Lee, Katsuaki Isobe, Noboru Shibata, Kapil Verma, Takuya Arika, Jason Li, Jong Yuh, Anirudh Amarnath, et al. A 512gb 3-bit/cell 3d flash memory on 128-wordline-layer with 132mb/s write performance featuring circuit-under-array technology. In *IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 218–220. IEEE, 2019.

- 
- [154] Jae-Woo Park, Doogon Kim, Sunghwa Ok, Jaebeom Park, Taeheui Kwon, Hyunsoo Lee, Sungmook Lim, Sun-Young Jung, Hyeongjin Choi, Taikyu Kang, et al. A 176-stacked 512gb 3b/cell 3d-nand flash with 10.8 gb/mm² density with a peripheral circuit under cell array architecture. In *IEEE International Solid-State Circuits Conference (ISSCC)*, volume 64, pages 422–423. IEEE, 2021.
- [155] Hwang Huh, Wanik Cho, Jinhaeng Lee, Yujong Noh, Yongsoon Park, Sunghwa Ok, Jongwoo Kim, Kayoung Cho, Hyunchul Lee, Geonu Kim, et al. A 1tb 4b/cell 96-stacked-wl 3d nand flash memory with 30mb/s program throughput using peripheral circuit under memory cell array technique. In *IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 220–221. IEEE, 2020.
- [156] Bo Lu, Chen-Rui Fan, Lu Liu, Kai Wen, and Chuan Wang. Speed-up coherent ising machine with a spiking neural network. *Optics Express*, 31(3):3676–3684, 2023.
- [157] Timothy A Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–25, 2011.
- [158] Scott P Kolodziej, Mohsen Aznaveh, Matthew Bullock, Jarrett David, Timothy A Davis, Matthew Henderson, Yifan Hu, and Read Sandstrom. The suitesparse matrix collection website interface. *Journal of Open Source Software*, 4(35):1244, 2019.
- [159] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.
- [160] Jeffrey P Gavornik and Harel Z Shouval. A network of spiking neurons that can represent interval timing: mean field analysis. *Journal of computational neuroscience*, 30:501–513, 2011.
- [161] Alessandro Treves. Mean-field analysis of neuronal spike dynamics. *Network: Computation in Neural Systems*, 4(3):259, 1993.

- 
- [162] Radford M Neal. Annealed importance sampling. *Statistics and computing*, 11:125–139, 2001.
- [163] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [164] Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pages 6545–6554. PMLR, 2019.
- [165] Rasmus Palm, Ulrich Paquet, and Ole Winther. Recurrent relational networks. *Advances in neural information processing systems*, 31, 2018.
- [166] Hongqing Cao, Friedrich Recknagel, and Philip T Orr. Parameter optimization algorithms for evolving rule models applied to freshwater ecosystems. *IEEE Transactions on Evolutionary Computation*, 18(6):793–806, 2013.
- [167] Maxence Bouvier, Alexandre Valentian, Thomas Mesquida, Francois Rummens, Marina Reyboz, Elisa Vianello, and Edith Beigne. Spiking neural networks hardware implementations and challenges: A survey. *J. Emerg. Technol. Comput. Syst.*, 15(2), April 2019.
- [168] Hesham Mostafa. Supervised learning based on temporal coding in spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 29(7):3227–3235, 2018.
- [169] Iulia-Maria Comşa, Krzysztof Potempa, Luca Versari, Thomas Fischbacher, Andrea Gesmundo, and Jyrki Alakuijala. Temporal coding in spiking neural networks with alpha synaptic function: Learning with backpropagation. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2021.
- [170] Tao Liu, Zihao Liu, Fuhong Lin, Yier Jin, Gang Quan, and Wujie Wen. Mt-spike: A multilayer time-based spiking neuromorphic architecture with temporal error back-

- 
- propagation. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 450–457. IEEE, 2017.
- [171] Axel Laborieux, Maxence Ernout, Tifenn Hirtzlin, and Damien Querlioz. Synaptic metaplasticity in binarized neural networks. *Nature Communications*, 12(2549), 2021.
- [172] Saeed Reza Kheradpisheh, Maryam Mirsadeghi, and Timothée Masquelier. Bs4nn: Binarized spiking neural networks with temporal coding and learning, 2020.
- [173] Eyyüb Sari, Mouloud Belbahri, and Vahid Partovi Nia. How does batch normalization help binary training? *arXiv:1909.09139*, 2019.
- [174] Tifenn Hirtzlin, Bogdan Penkovsky, Marc Bocquet, Jacques-Olivier Klein, Jean-Michel Portal, and Damien Querlioz. Stochastic computing for hardware implementation of binarized neural networks. *IEEE Access*, 7:76394–76403, 2019.
- [175] Abhronil Sengupta, Gopalakrishnan Srinivasan, Deboleena Roy, and Kaushik Roy. Stochastic inference and learning enabled by magnetic tunnel junctions. In *2018 IEEE International Electron Devices Meeting (IEDM)*, pages 15–6. IEEE, 2018.
- [176] Minsuk Koo, Gopalakrishnan Srinivasan, Yong Shim, and Kaushik Roy. Sbsnn: Stochastic-bits enabled binary spiking neural network with on-chip learning for energy efficient neuromorphic computing at the edge. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(8):2546–2555, 2020.
- [177] Kai Ni et al. Ferroelectric ternary content-addressable memory for one-shot learning. *Nature Electronics*, 2(11):521–529, 2019.
- [178] Xiaoming Chen, Xunzhao Yin, Michael Niemier, and Xiaobo Sharon Hu. Design and optimization of fefet-based crossbars for binary convolution neural networks. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1205–1210, 2018.

- 
- [179] Changhoon Lee, Juho Sung, and Changhwan Shin. Understanding of feedback field-effect transistor and its applications. *Applied Sciences*, 10(9):3070, 2020.
- [180] Young-Won Kim, Joo-Seong Kim, Jae-Hyuk Oh, Yoon-Suk Park, Jong-Woo Kim, Kwang-Il Park, Bai-Sun Kong, and Young-Hyun Jun. Low-power cmos synchronous counter with clock gating embedded into carry propagation. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 56(8):649–653, 2009.
- [181] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R Stanley Williams, and Vivek Srikumar. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in cross-bars. *ACM SIGARCH Computer Architecture News*, 44(3):14–26, 2016.
- [182] Vagner Guidotti, Guilherme Paim, Leandro MG Rocha, Eduardo Costa, Sérgio Almeida, and Sergio Bampi. Power-efficient approximate newton–raphson integer divider applied to nlms adaptive filter for high-quality interference cancelling. *Circuits, Systems, and Signal Processing*, 39:5729–5757, 2020.
- [183] Ben Perach et al. An asynchronous and low-power true random number generator using stt-mtj. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(11):2473–2484, 2019.