Summer 9-12-2023

Theory and Application of Dynamic Analog Memory based on Fowler-Nordheim Quantum Tunneling

Mustafizur Rahman
Washington University – McKelvey School of Engineering

WASHINGTON UNIVERSITY IN ST. LOUIS

McKelvey School of Engineering
Department of Electrical & Systems Engineering

Dissertation Examination Committee:
Shantanu Chakrabartty, Chair
Netanel Raviv
Bruno Sinopoli
Chuan Wang
Ning Zhang

Theory and Application of Dynamic Analog Memory based on Fowler-Nordheim Quantum Tunneling

by

Mustafizur Rahman

A dissertation presented to
the McKelvey School of Engineering
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

August 2023
St. Louis, Missouri

Table of Contents

List of Figures

List of Tables

Acknowledgments

First, I would like to be thankful to *ALLAH* for everything that has ever happened to me.

I would also like to thank my advisor Dr.Shantanu Chakrabartty who has always provided me with the necessary knowledge to work towards my goal and whose sincere interest and active involvement in my work have motivated me to complete my thesis. I am also very grateful to my thesis committee members Dr. Netanel Raviv, Dr. Bruno Sinopoli, Dr. Chuan Wang, and Dr. Ning Zhang for their valued advice and interest in my work.

My heartfelt thanks to my lab mates, both former and present ones. They have been both friends and family to me for the past five years. I would especially like to thank Dr. Darshit Mehta who has worked alongside me as a mentor and Subhankar Bose who has been more like a younger brother to me. I will forever be indebted to them for all the help that they provided. Also, a great heartfelt thanks to Dr. Oindrila Chatterjee and Dr. Ahana Gangopadhyay for listening to my whining and guiding me through every tough time. My gratitude towards the Clemens community Kaushik Dutta, Deepangshu Chatterjee, Sneha Dasgupta, Ankita Nandi, and Shreya Shah, all of whom love and support towards me and my family has been unparalleled. St Louis would have never felt like home without the company of these people.

I would also like to be thankful to my parents and elder brother who has believed in me since my childhood and motivated me to pursue my academic life abroad to achieve greatness. I remember that my father always used to say that one day I will get a doctorate degree (if I study properly). Somehow he always knew, even in my early childhood. My special admiration for my little child Nirvaan who has never failed to put a smile on my face when I got too overwhelmed by the work. And lastly, thanks to my partner in life Tasnuva who has been my constant support and has sacrificed a lot in order to let me pursue my dreams.

Mustafizur Rahman

Washington University in St. Louis
August 2023

Dedicated to my family who make my world complete

especially *Abbu* watching me from Heaven.

ABSTRACT OF THE DISSERTATION

Theory and Application of Dynamic Analog Memory based on Fowler-Nordheim Quantum Tunneling

by

Mustafizur Rahman

Doctor of Philosophy in Electrical Engineering

Washington University in St. Louis, 2023

Professor Shantanu Chakrabartty, Chair

Traditionally, memory devices store information in a static manner be it charge-based devices like floating-gates of FeFET or any other method for that matter such as spin-based, electrochemical-based, magnetic-based, etc. But what if memory storage became dynamic in nature and you can control the dynamics? My thesis is based on these premises where I investigate the theory and application of Dynamic Analog Memory (DAM). In my thesis, I answer the questions on how to design such a DAM and what optimal characteristics it needs to have for it to demonstrate advantages over traditional memory devices.

To this end, my thesis first focuses on a self-powered dynamical system that is implemented by depositing charges on an electronically isolated poly-silicon island where the charge leaks through to a semiconductor substrate. The amount of leakage is synchronized not only across multiple tunneling junctions but also across different dies. The dynamical system can be desynchronized by coupling external signals to it. I designed a synaptic memory element that exploits the desynchronization between two dynamical systems to implement an analog memory. First, I characterize the plasticity and the energy required for updating the analog memory. Then I show tunable memory consolidation properties of these synaptic elements using a benchmark random-pattern experiment. Furthermore, I will show that when Fowler-Nordheim quantum tunneling process is used as the leakage mechanism for the DAM,

the synaptic elements can exhibit optimal memory consolidation and task-specific based consolidation which can be used in continual learning scenarios. Finally, I also demonstrate that by exploiting the tradeoff between the memory retention period and the energy required for updating the analog memory information can be stored by expending less than 1pj of energy per bit during the training phase of an artificial neural network, four orders of magnitude improvement than conventional memory.

Next, I have implemented a novel class of quantum-secure dynamic encryption key distribution and authentication protocols exploiting the security primitives and synchronization capabilities of these self-powered dynamical systems. The FN-dynamical systems are not only synchronized with each other but also the dynamic profile can be modeled accurately with respect to time. I proposed a key exchange protocol that uses publicly available identical copies of self-powered chipsets where the temporal dynamics on the hardware chipsets are synchronized with its software clone i.e. the analytical model running on a server. I show that the dynamic keys derived from these temporal dynamics meet the National Institute of Standards and Technology (NIST) criteria. I also prove the security of these protocols under a standard model and against different adversarial attacks. I have also investigated the robustness of these protocols using hardware results and propose error-correcting protocols to mitigate noise-related artifacts. Finally, I propose a synchronized pseudo-random number generator (SPRNG) that uses a combination of a fast, low-complexity linear-feedback-shift-register (LFSR) based PRNG and a slow but secure, synchronized seed generator based on self-powered FN-dynamical system. I investigated protocols to periodically and securely generate random bits using the self-powered FN-dynamical system for seeding the LFSR. The time-varying random seeds extend and break the LFSR periodic cycles, thus making it difficult for an attacker to predict the random output or the random seed.

Chapter 1

Introduction

We humans have had the need to store information from the beginning of time and as such memory devices have existed long before the modern era. Since the time when human beings started to learn to communicate we have passed down our learnings to the new generation through some means of memory or storage devices. One of the earliest examples of such a memory device dates back to 2500 B.C. when Mesopotamians stored information about their laws, literature, religion, and sciences on impressed clay tablets. However, they soon found out that clay tablets lack the durability that they required. As such they started to etch their inscriptions onto stones as shown in Fig. 1.1 for longer retention of their information [81]. Fast forward a few thousand years and we have a new generation of information storage devices as can be seen in Fig. 1.1 [123]. In 1950 magnetic tape was used for storing information which is one of the oldest technologies for electronic data storage. Gradually, over the next 70 years, we introduced newer generations of electronic storage devices which decreased in form factors but became better at storing more information. Emphasis was once again given to retaining the information as long and as accurately as possible which meant that information was stored in a static manner. Therefore, the underlying principle in information storage throughout the history of memory devices has been the fact that they are static in nature. But what if the memory devices became dynamic meaning information was stored in a dynamic system rather than the conventional static devices? Can we satisfy the general criteria of memory devices, i.e. retention and accuracy, while at the same time exploring new frontiers in modern applications with such a dynamic memory?

My thesis is based on these premises where I investigate the theory and application of Dynamic Analog Memory (DAM). From a biological perspective, it makes sense for memory devices to store information in a dynamic system as it is very well known that biological synapses store memories through multiple electrochemical processes. Therefore, we might be able to take advantage of such a dynamic memory to further our advancement in the Artificial

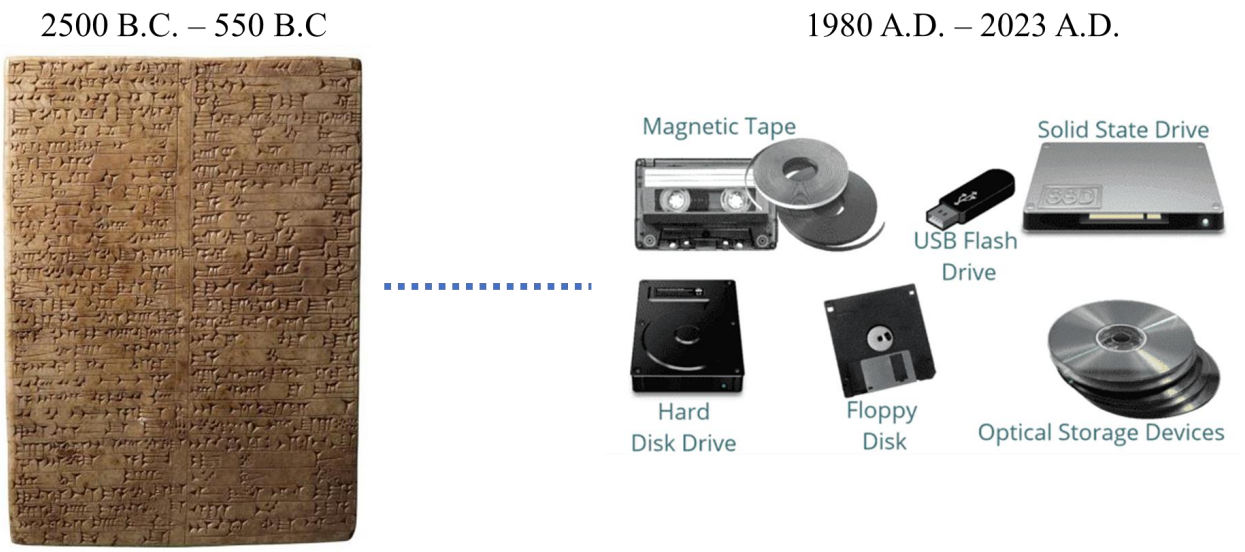

Figure 1.1: Evolution of memory device from being a stone slab in 2500 B.C. to electronic memory devices in modern era

Intelligence (AI) or neuromorphic engineering domain. In addition, when information is stored in a dynamic system, it inherently adds a layer of protection to the stored memory since it will not be enough just to know the state of the memory at any one particular moment to extract the information. Rather one would need to know other details such as the phase or initial condition of the system to decipher what is stored in them. This opens up potential applications in the cryptographic security domain. In my thesis, I answer the questions on how to design such a DAM and what optimal characteristics it needs to have for it to demonstrate advantages over traditional memory devices in the AI and cryptographic security domain.

1.1 Dynamic Memory in AI

There is a growing evidence from the field of neuroscience and neuroscience inspired AI about the importance of implementing synapses as a complex high-dimensional dynamical system [15, 46], as opposed to a simple and a static storage element, as depicted in standard neural networks [113]. This dynamical systems viewpoint has been motivated by the hypothesis that complex interactions between plethora of biochemical processes at a synapse produces *synaptic metaplasticity* [3] and plays a key role in *synaptic memory consolidation* [80]. Both

these phenomena have been observed in biological synapses [133, 132] where the synaptic plasticity (or ease of update) can vary depending on age and task-specific usage that is accumulated during the process of learning. In literature these long-term synaptic memory consolidation dynamics have been captured using different analytical models with varying degrees of complexity [15, 46, 109, 10, 44, 45]. One such model is the cascade model [15] which has been shown to achieve the theoretically optimal memory consolidation characteristic for benchmark random pattern experiments. However, the physical realization of cascade models as described in [15] uses a complex coupling of dynamical states and diffusion dynamics, which is difficult to mimic and scale *in-silico*. Similar optimal memory consolidation characteristics have been reported in the context of continual learning in artificial neural networks (ANN) where synapses that are found to be important for learning a specific task are consolidated (or become rigid) [69, 26, 82, 136, 78, 8]. As a result, when learning a new task the synaptic weight does not significantly deviate from the consolidated weights, hence, the network seeks solutions that work well for as many tasks as possible. However, these synaptic models are algorithmic in nature and it is not clear if the optimal consolidation characteristics can be naturally implemented on the synaptic device *in-silico*. Also, it is not clear if the consolidation properties of the physical synaptic device can be tuned to achieve different *plasticity-stability* trade-offs and hence can overcome the relative disadvantages of the EWC models. In my thesis, I investigate a simple differential memory device that operates using the physics of Fowler-Nordheim (FN) quantum-mechanical tunneling and show that such an on-device memory element can achieve tunable synaptic memory consolidation characteristics similar to the algorithmic consolidation models.

Another unresolved challenge in the design of energy-efficient machine learning (ML) and neuromorphic processors is the implementation of reliable and scalable synaptic weights or memory [23]. Ideally, the synaptic weights should be "analog" and should be implemented on a non-volatile, and yet easily modifiable storage device [130]. Furthermore, if these memory elements are integrated in proximity with the computing circuits or processing elements, then the resulting compute-in-memory (CIM) architecture [2, 128] has the potential to mitigate the "memory wall" [110, 61, 29] which refers to the energy-efficiency bottleneck in ML processors that arises due to repeated memory access. In most practical and scalable implementations, the processing elements are implemented using CMOS circuits; as a result, it is desirable that the analog synaptic weights be implemented using a CMOS-compatible technology. In literature, several multi-level non-volatile memory devices have been proposed for implementing analog synapses. These include two-terminal memristive devices

such as resistive random-access memories (RRAM) [6], magnetic random-access memories (MRAM) [47], Phase Change Memory (PCM) [22], Spin-Transfer Torque Magnetic RAM (STT-MRAM) [68], Conductive Bridge RAM [63] or the three-terminal devices like the floating-gate transistors [92], ferroelectric field-effect transistor-based memory (FeFET) [39], Charge Trap Memory [129] and Electrochemical RAMs (ECRAM) [120]. In all of these devices, the analog memory states are static in nature, where each of the states needs to be separated from others by an energy barrier ΔE. For example, in RRAM devices the state of the conductive filament between two electrodes determines the stored analog value, whereas in charge-based devices like floating-gates or FeFET, the state of polarization determines the analog value. To ensure non-volatile storage, it is critical that the energy-barrier ΔE is chosen to be large enough to prevent memory leakage due to thermal-fluctuations and other environmental disturbances. However, the height of the energy barrier ΔE also sets the fundamental limit on the energy dissipated to switch between different analog storage states. For example, switching the RRAM memory state requires 100 fJ per bit [131], whereas STT-MRAM requires about 4.5pJ per bit [38]. A learning/training algorithm that adapts the stored weights in quantized steps $(\dots, W_{n-1}, W_n, W_{n+1}, \dots)$ so as to minimize a system-level loss-function $L(W)$ has to dissipate minimum energy of $(\dots, \Delta E_{n-1}, \Delta En, \Delta E_{n+1}, \dots)$ for memory updates. Separating the static states by an energy-barrier also allows the learning algorithm to precisely control the parameter retention time (parameter leakage) between subsequent parameter updates, however, this mode of updates does not exploit the physics of learning to optimize for energy-efficiency. In many energy-efficient ML training formulations, and in particular analog ML systems, the loss-function $L(W)$ is represented by an equivalent energy-functional of a physical ML system [76], and learning/training involves a natural evolution of the system dynamics towards the minimum energy (optimal) state based on input stimuli (or equivalently training data). Thus, the physics of the system evolution process selects the minimum energy path toward the desired optimum. A synaptic element that is matched to this system dynamics needs to be adaptive with respect to its memory retention time which can then be traded-off with respect to the energy-dissipation per update. In my thesis I present such a synaptic element that uses dynamical states (instead of static states) to implement analog memory and is matched to the dynamics of ML training.

1.2 Dynamic Memory in cryptographic security

Securing information exchange with internet-of-things (IoTs) is becoming ever more important due to the proliferation of these platforms in domains ranging from infrastructure-IoTs [11] to medical-IoTs [12]. In one study [1] it is claimed that around 98% of the IoT data traffic is unencrypted and hence vulnerable to a data breach. Conventional data encryption techniques like RSA are too computationally prohibitive to be universally implemented on these low-resource platforms and reducing the computational complexity makes the approach vulnerable to quantum attacks. For instance, it is estimated in literature that a quantum computer with 8194 logical qubits using Shor's Algorithm would be able to break the Rivest-Shamir-Adleman(RSA)[108] system with a key size of 4096 bits in 229 hours while for Discrete log problem with a key size of 521 bits it would take 55 hours for a quantum computer with 4719 logical qubits, again using the Shor's Algorithm[95]. Symmetric key algorithms like Advanced Encryption Standard (AES-256) can be customized for IoT platforms and are considered to be secure against quantum attack [95], provided the security of the initial key-exchange can be guaranteed. Quantum key distribution(QKD)[16] which is based on the principles of quantum-mechanics, like quantum entanglement [41] or the no-cloning principle [50],[101] could be used to guarantee the security of the initial key-exchange. However, one of the major drawbacks of current state-of-the-art QKD systems is that they require dedicated and specialized peer-to-peer communication links [124],[37],[72],[134]. Not only do these links require careful maintenance and calibration to ensure quantum-coherence, but these systems are also expensive and not portable. Hence, current QKD systems cannot be scaled for internet-scale key distribution [62],[19] and communications involving lightweight IoT devices with resource constraints will still be vulnerable to quantum attacks. Therefore, we need a framework that does not require any modifications to the existing communication infrastructure, can be scaled to a large number of IoTs, and is potentially secure against quantum attacks.

One such approach could be silicon-based chipsets with the capability of integrating billions of transistors and memory elements [32] which can be manufactured on a large scale and at a low-cost [55]. If a physical feature on these chipsets could be exploited to implement a secure one-way function, then a hardware-software approach could be used to support key distribution over public channels. In the literature, a few hardware-software key exchange methods have been proposed. In [36] a hardware-software key exchange technique was proposed that

exploited correlations across chaotic wavepackets in classic optical communications channels. However, the method still requires peer-to-peer connectivity between the users and hence has similar scaling disadvantages as QKD methods. The hardware-software approach proposed in [66] used chaos synchronization to distribute random keys over public channels. However, due to the lack of reliable synchronization, this approach incurs significant errors during decryption. Recently, Physical Unclonable Function(PUF) based hardware-based encryption key distribution has been proposed. A specific variant of this technique, described in [14] as Public Physical Unclonable Function(PPUF) has been used for public-key cryptography and leverages the difficulty of accessing physical information stored on chipsets. However, in PPUF the stored information is static in nature and hence is potentially vulnerable to machine learning attacks [34, 86]. To overcome this limitation, in my thesis, I propose a novel class of symmetric key distribution protocols that leverages basic security primitives offered by low-cost, hardware chipsets containing millions of self-powered micro-dynamical systems. The secret keys are stored in a dynamic manner and the encrypting keys are derived from the temporal dynamics of the device which makes the keys immune to any potential side-channel attacks, malicious tampering, or snooping.

Another aspect of securing wireless communications in internet-of-things (IoT) is the need for both generation and synchronization of random numbers in real-time. Random-number-generators (RNGs) are used for producing cryptographic keys which are basically a sequence of random binary numbers. In addition to using a random number as a secure token, for secure communications, there is also a need for synchronization of the tokens between the communicating parties. While asymmetric key encryption could be used to avoid this challenge, they are computationally too expensive to be universally implemented on these resource-constrained devices. On the other hand, a symmetric key encryption scheme can be customized for IoT platforms but requires a shared secret key [57]. Any static information stored on the IoT, such as a SecureID, used as the shared secret will be vulnerable to a machine learning type of attack [86]. Therefore, there is a need for a piece of dynamic information embedded into these IoT devices that can be synchronized in real-time. One such method for achieving this could be using a combination of a timing reference extracted from a global-positioning-system (GPS) and a timing reference generated locally using phased-locked oscillators. Unfortunately, in many IoT applications, this framework is impractical due to resource constraints together with the fact that many IoT devices may not have access to a GPS signal. To overcome this challenge, in my thesis, I designed an architecture

of a synchronized-PRNG (SPRNG) that can be used for generating synchronized pseudo-random binary sequences without the need for any GPS reference signal. The SPRNG uses a combination of a fast, low-complexity LFSR-based PRNG and a slow but secure, synchronized seed generator based on the self-powered dynamic memory elements [137, 91, 104].

1.3 Contributions

There are four significant contributions in this dissertation and they are summarized as follows.

1: I designed a differential dynamic analog memory device that stores information on a self-powered micro-dynamic system. I show that when the dynamic system operates using the physics of Fowler-Nordheim (FN) quantum-mechanical tunneling, the fabricated memory device (FNDAM) can achieve optimal memory consolidation characteristics previously only shown in algorithmic consolidation models. This is the first work to show on-device task-specific memory consolidation properties which outperforms other algorithms when used in continual learning application. I have also characterized the memory capacity of an artificial neural network based on FNDAM and showed that on-device memory elements can achieve the theoretical limits.

2: I have investigated the energy-efficiency imbalance between the training and the inference phases observed in neuromorphic learning systems. I have proposed that data retention capacity can then be traded-off with respect to the energy dissipation per update to reduce the energy requirement during the training phase of an artificial neural network. I have shown that the dynamic nature of FNDAM can be exploited to match this profile. I have also performed chip-in-the-loop training with the fabricated FNDAM and have validated that energy usage during the training phase is much lower than other conventional memory. I show that information can be stored by expending less than 1pJ of energy per bit on FNDAM, a four orders of magnitude improvement.

3: I designed a novel class of symmetric key distribution protocols that leverages basic security primitives offered by low-cost, hardware chipsets containing millions of synchronized self-powered dynamic systems. I have derived encryption keys from the temporal dynamics of a physical device and showed that the keys are immune to any

potential side-channel attacks, malicious tampering, or snooping. I also show that the derived key-strings can pass the randomness test as defined by the National Institute of Standards and Technology (NIST) suite. The key-strings are then used in two SPoTKD (Self-Powered Timer Key Distribution) protocols that exploit the timer's dynamics as one-way functions: (a) protocol 1 facilitates secure communications between a user and a remote Server; and (b) protocol 2 facilitates secure communications between two users. I have investigated the security of these protocols under the standard model and against different adversarial attacks. Using Monte-Carlo simulations, I also investigate the robustness of these protocols in the presence of real-world operating conditions and propose error-correcting SPoTKD protocols to mitigate these noise-related artifacts.

4: I have designed a synchronized pseudo-random number generator (SPRNG) that uses a combination of a fast, low-complexity linear-feedback-shift-register (LFSR) based PRNG and a slow but secure, synchronized seed generator based on a self-powered dynamic system that has been previously used as time-keeping devices. I have explored protocols to periodically and securely generate random bits using the self-powered timers for seeding the LFSR. I have shown that the time-varying random seeds extend and break the LFSR periodic cycles, thus making it difficult for an attacker to predict the random output or the random seed. Using the National Institute of Standards and Technology (NIST) test suite I verify the randomness of the measured seeds using an ensemble of self-powered timers fabricated in a standard CMOS process and the random bit sequences generated by a software-seeded LFSR.

1.4 Organization of this Dissertation

The remaining chapters in this dissertation are organized as follows: **Chapter 2** presents the construction of the FNDAM along with the details of the fabricated array of FNDAM. It contains the analytical derivation of consolidation properties of FNDAM which shows FN-tunneling as the optimum leakage process. It shows simulated results that conform with the derived analytical expressions. It also shows results validating the memory capacity of an artificial neural network based on FNDAM. Finally, it shows the results of the application of FNDAM in continual learning scenarios. The results in this chapter are based on [104].

Chapter 3 builds up on the previous chapter and discusses the application of FNDAM regarding the energy efficiency of an artificial neural network during the training phase. It presents the argument for a tradeoff between memory retention and the energy barrier for a memory update in the training phase. It characterizes the weight update of FNDAM in response to various inputs. It includes details of a chip-in-the-loop training and corresponding results showing energy-efficiency balance between the training and the inference phases. Finally, it compares the energy usage of FNDAM with resistive random-access memories (RRAM) for training a deep neural network. The results in this chapter are based on [91].

Chapter 4 presents the application of an FN-dynamical system in the cryptographic key exchange domain. The framework for the key exchange protocol is first described. After that, it discusses the unique features of an on-device FN-dynamical system i.e. security primitives which are then exploited to design a key exchange protocol called Self- Powered Timer Key Distribution (SPoTKD). It also contains the security analysis of SPoTKD under the standard model. The performance of SPoTKD is then compared with other state-of-the-art key exchange protocols. It also presents the noise robustness of the framework and explores an error correction version of SPoTKD. The results in this chapter are based on [106].

Chapter 5 presents the concept of the synchronized pseudo-random number generator (SPRNG). It shows the complete system-on-chip prototype array of synchronized self-powered timers (SSPT) to generate dynamic random seeds for the LFSR. It also presents a secure seed exchange protocol for the SPRNG system. Measured results for random seed generation and synchronization from fabricated prototypes are shown. It also contains a statistical analysis of the random number generated using the NIST test suite. The results in this chapter are based on [105].

Chapter 6 presents a summary of this research, along with key contributions of this dissertation and potential directions for future research.

Chapter 2

Synaptic Memory Consolidation using FN-tunneling

This chapter introduces the concept of differential dynamic analog memory based on FN-tunneling along with the topic of memory consolidation. It explores how to achieve on-device synaptic memory consolidation using FNDAM. Since FNDAM was used primarily as a synaptic memory element in this chapter, it is thus being referred to as FN-synapse for the rest of the chapter. This chapter has supplemental information which is included in Appendix A. This chapter is based on the published work [104].

2.1 Introduction

There is a growing evidence from the field of neuroscience and neuroscience-inspired AI about the importance of implementing synapses as a complex high-dimensional dynamical system [15, 46], as opposed to a simple and static storage element, as depicted in standard neural networks [113]. This dynamical systems viewpoint has been motivated by the hypothesis that complex interactions between the plethora of biochemical processes at a synapse (illustrated in Fig. 2.1(a)) produces *synaptic metaplasticity* [3] and plays a key role in *synaptic memory consolidation* [80]. Both these phenomena have been observed in biological synapses [133, 132] where the synaptic plasticity (or ease of update) can vary depending on age and task-specific usage that is accumulated during the process of learning. In literature, these long-term synaptic memory consolidation dynamics have been captured using different analytical models with varying degrees of complexity [15, 46, 109, 10, 44, 45]. One such model is the cascade model [15] which has been shown to achieve the theoretically optimal memory consolidation characteristic for benchmark random pattern experiments. However, the physical realization of cascade models as described in [15] uses a complex coupling of dynamical states and

diffusion dynamics (an example illustrated in Fig. 2.1(b) using a reservoir model), which is difficult to mimic and scale *in-silico*. Similar optimal memory consolidation characteristics have been reported in the context of continual learning in artificial neural networks (ANN) where synapses that are found to be important for learning a specific task are consolidated (or become rigid) [69, 26, 82, 136, 78, 8]. As a result, when learning a new task the synaptic weight does not significantly deviate from the consolidated weights, hence, the network seeks solutions that work well for as many tasks as possible. However, these synaptic models are algorithmic in nature and it is not clear if the optimal consolidation characteristics can be naturally implemented on the synaptic device *in-silico*. Also, it is not clear if the consolidation properties of the physical synaptic device can be tuned to achieve different *plasticity-stability* trade-offs and hence can overcome the relative disadvantages of the EWC models. In this work, we report that a simple differential device that operates using the physics of Fowler-Nordheim (FN) quantum-mechanical tunneling can achieve tunable synaptic memory consolidation characteristics similar to the algorithmic consolidation models. The operation of the synaptic device, referred to in this chapter as the FN-synapse, can be understood using a reservoir model as shown in Fig. 2.1 (c)). Two reservoirs with fluid levels W^+ and W^- are coupled to each other using a sliding barrier X. The barrier is used to control the fluid flow from the respective reservoirs into an external medium. The respective flows, which are modeled by functions $J(W^+)$ and $J(W^-)$, at time-instant t are modulated by the position of the sliding barrier $X(t)$ and the level of fluid in the external reservoir $m(t)$. In this reservoir model, the synaptic weight is stored as $W_d = \frac{1}{2}(W^+ - W^-)$ whereas $W_c = \frac{1}{2}(W^+ + W^-)$ serves as an indicator of synaptic usage with respect to time.

In the Methods section we show that for a synapse based on a general differential reservoir model (without making assumptions on the nature of the flow function $J(.)$) the synaptic weight W_d evolves in response to the external input $X(t)$ according to the coupled differential equation

$$\frac{dW_d}{dt} = -r(t)W_d + X(t) \tag{2.1}$$

where

$$r(t) = \frac{d^2W_c}{dt^2}\left(\frac{dW_c}{dt}\right)^{-1} \tag{2.2}$$

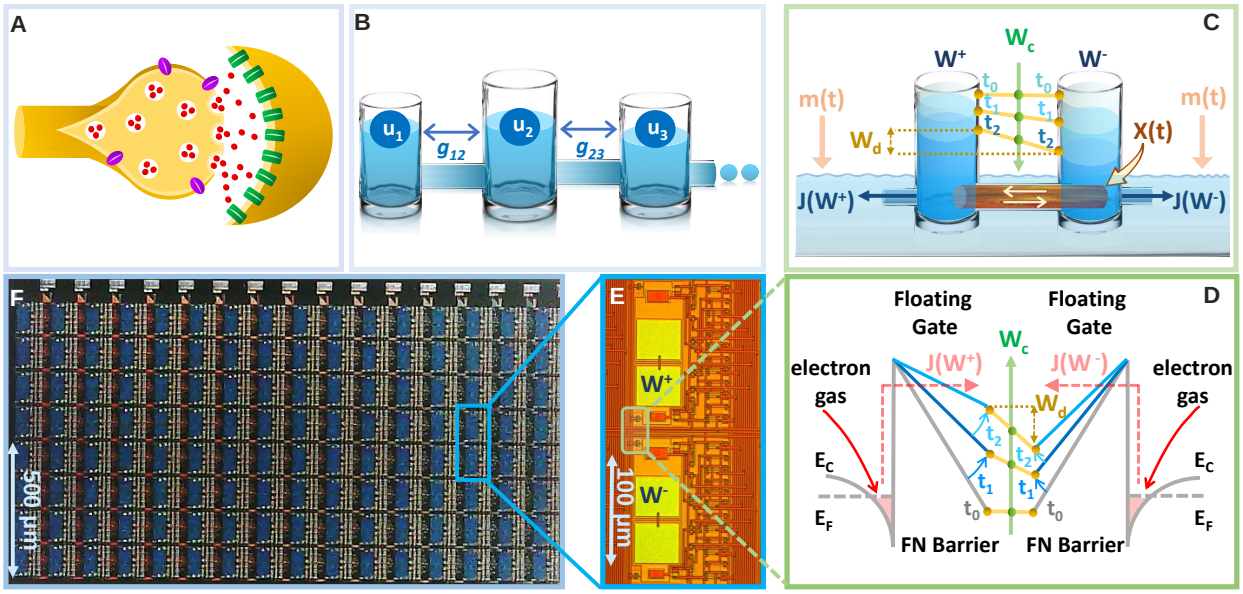

Figure 2.1: :(A) An illustration of a biological synapse with different coupled biochemical processes that determine synaptic dynamics (B) physical realization of the cascade model reported in [15] that captures the consolidation dynamics using fluid in reservoirs u_k that are coupled through parameters g_{kj}. (C) illustration of the FN-synapse dynamics using a differential reservoir model and its state at time-instants t_0,t_1, and t_2;(D) Energy-band diagram to show the implementation of the reservoir model in (C) using the physics of Fowler-Nordheim quantum-mechanical tunneling where a single synaptic element (as shown in (E)) which stores the weight W_d as the differential charge stored between each tunneling junction *i.e.* $W_d = \frac{W^+ - W^-}{2}$ and the common-mode tunneling voltage W_c as the average of the individual charges *i.e.* $W_c = \frac{W^+ + W^-}{2}$ and (E) micrograph of a single FN-synapse;(F) Micrograph of an array of FN-synaptic devices fabricated in a standard silicon process.

is a time-varying decay function that models the dynamics of the synaptic plasticity as a function of the history of synaptic activity (or its usage). The usage parameter W_c evolves according to

$$\frac{dW_c}{dt} = -J\left(W_c\right) + m(t) \tag{2.3}$$

based on the functions $J(.)$ and $m(t)$. Equations 2.1- 2.3 show that the weight W_d update does not directly depend on the non-linear function $J(.)$ but implicitly through the common-mode W_c. Furthermore, equation 2.1 conforms to the weight update equation reported in the EWC model [69] where it has been shown that if $r(t)$ varies according to the network Fisher information metric, then the strength of a stored pattern or memory (typically defined

in terms of signal-to-noise ratio) decays at an optimal rate of $1/\sqrt{t}$ when the synaptic network is subjected to random, uncorrelated memory patterns. In Methods, we show that if the objective is to maximize the operational lifetime of the synapse, then equating the time-evolution profile in equation 2.2 to $r(t) \approx \mathcal{O}(1/t)$ [69] leads to an optimal $J(.)$ of the form $J(V) \propto V^2 \exp(-\beta/V)$ where β is a constant. The expression for $J(V)$ matches the expression for a Fowler-Nordheim (FN) quantum-mechanical tunneling current [79] indicating that optimal synaptic memory consolidation could be achieved on a physical device operating on the physics of FN quantum-tunneling.

To verify on-device optimal consolidation dynamics we fabricated an array of FN-synapses and Fig. 2.1 (d)-(e) shows the micrograph of the fabricated prototype. In the Methods Section, we show the mapping of the differential reservoir model using the physical variables associated with FN quantum tunneling and Fig. 2.1(f) shows the mapping using an energy-band diagram. Similar to our previous works [137, 138, 106], the tunneling junctions have been implemented using polysilicon, silicon-di-oxide, and n-well layers, where the silicon-di-oxide forms the FN-tunneling barrier for electrons to leak out from the n-well onto a polysilicon layer. The polysilicon layer forms a floating-gate where the initial charge can be programmed using a combination of hot-electron injection or quantum-tunneling [91, 90].The synaptic weight is stored as a differential voltage $W_d = \frac{1}{2}(W^+ - W^-)$ across two floating-gates as shown in Fig. 2.1 (f). The voltages on the floating-gates W^+ and W^- at any instant of time are modified by the differential signals $\pm\frac{1}{2}X(t)$ that are coupled onto the floating-gates. The dynamics for updating W^+ and W^- are determined by the respective tunneling currents $J(.)$ which discharge the floating-gates. In the Appendix A Fig. A.1 we describe the complete equivalent circuit for the FN-synapse along with the read-out mechanism used in this work to measure W_d. The presence of additional coupling capacitors in Fig. S1 provides a mechanism to inject a common-mode modulation signal $m(t)$ into the FN-synapse. We will show in the results section that $m(t)$ can be used to tune the memory consolidation characteristics of the FN-synapse array to achieve memory capacity similar to or better than the cascade consolidation models (with different degrees of complexities) or the task-specific synaptic consolidation corresponding to the EWC model.

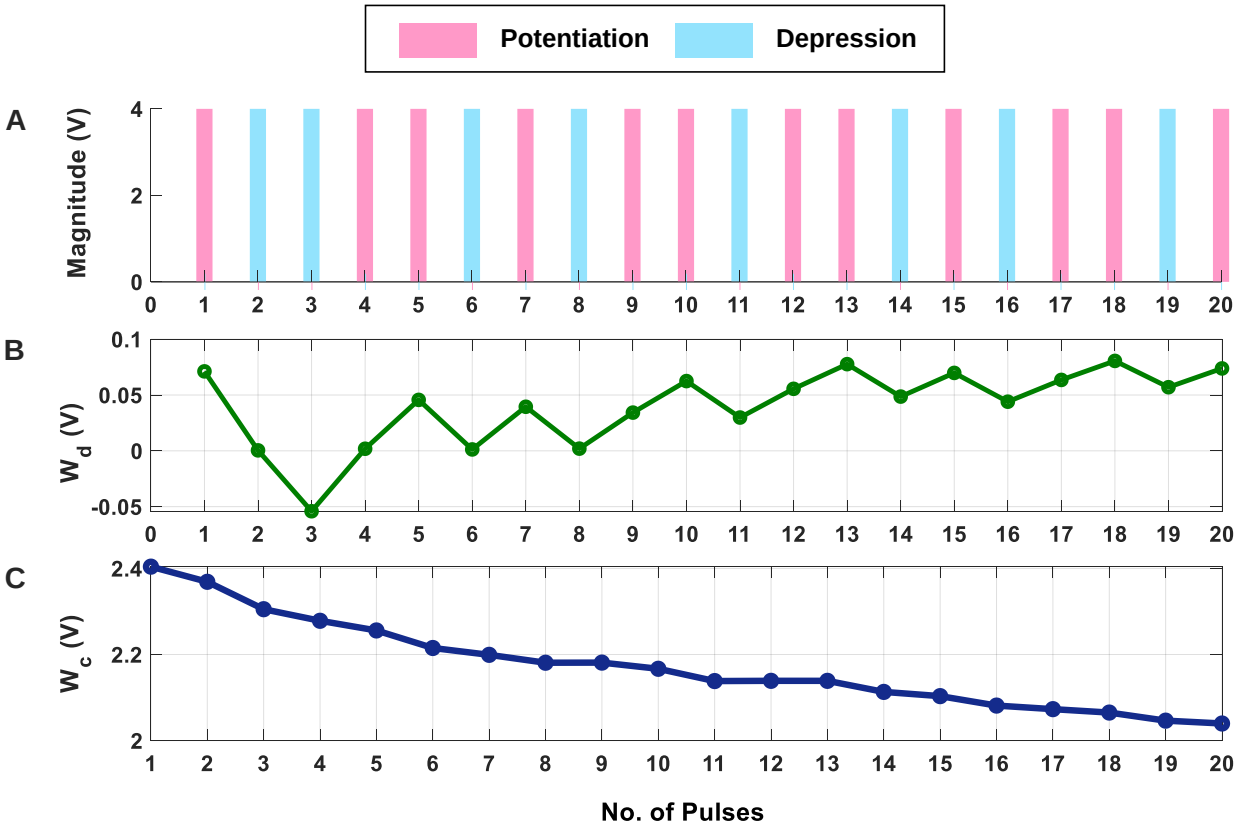

Figure 2.2: : (A) A random set of *potentiation* and *depression* pulses of equal magnitude and duration applied to the FN-synapse leading to (B) bidirectional evolution of weight (W_d) and (C) the corresponding trajectory followed by the common-mode tunneling node (W_c).

2.2 Results

2.2.1 FN-synapse Characterization

The first set of experiments was designed to understand the *metaplasticity* exhibited by FN-synapses and how the synaptic weight and usage change in response to external stimulation. The charge stored on the floating-gates in the FN-synapse was first initialized according to the procedure described in the Methods Section and in the Appendix A. The tunneling barrier thickness in FN-synapse prototype shown in Fig. 2.1 (d)-(e) was chosen to be greater than 12nm which makes the probability of direct-tunneling of electrons across the barrier to be negligible. The probability of FN-tunneling of electrons across the barrier (as shown in Fig. 2.1 (f)) is reduced to be negligible by lowering the electric potential of the tunneling

nodes W^+ and W^- (see Fig. A.1 in the Appendix A) with respect to the reference ground to be less than 5V. In this state, the FN-synapse behaves as a standard non-volatile memory storing a weight proportional to $W_d = W^+ - W^-$. To increase the magnitude of the stored weight a differential input pulse $\pm\frac{1}{2}X$ is applied across the capacitors that are coupled to the floating-gates (see Fig S1). The electric potential of the floating-gate W^- is increased beyond $7.5V$ where the FN-tunneling current $J(W^-)$ is significant. At the same time the electric potential of the floating-gate W^+ is also pushed higher but $W^- > W^+$ such that the FN-tunneling currents $J(W^+) < J(W^-)$. As a result, the W^- node discharges at a rate that is faster than the W^+ node. After the input pulse is removed, the potential of both W^- and W^+ are pulled below 5V and hence the FN-synapse returns to its non-volatile state.

Fig. 2.2 (a)-(c) shows the measured responses which show that an FN-synapse can store both the weight and the usage history. When a series of *potentiation* and *depression* pulses of equal magnitude and duration is applied to the FN-synapse, as shown in Fig. 2.2 (a), the weight stored W_d evolves bidirectionally (like a random walk) due to the input pulses (see Fig. 2.2 (b)). Meanwhile, the common-mode potential W_c decreases monotonically with the number of input pulses irrespective of the polarity of the input, as shown in Fig. 2.2 (c). Therefore, W_c reliably tracks the usage history of the FN-synapse whereas W_d stores the weight of the synapse. Fig. 2.3 (a) and (b) show the measured weight update ΔW_d in response to different magnitudes and duration of the input pulses. For this experiment the common-mode $W_c = \frac{1}{2}(W^+ + W^-)$ is held fixed. In Fig. 2.3 (a) we can observe that for a fixed magnitude of input voltage pulses (= 4V) ΔW_d changes linearly with pulse width. Whereas Fig. 2.3 (b) shows that the updated ΔW_d changes exponentially with respect to the magnitude of the input pulses (duration = 100ms). Thus, the results show that pulse width modulation or pulse density modulation provides more accurate control over the synaptic updates. Furthermore, in regard to energy dissipation per synaptic update pulse width modulation is also more attractive than using pulse magnitude variation. The energy required to write each time on FN-synapse can be estimated by measuring the energy drawn from the differential input source X in Fig. S1 to charge the coupling capacitor C_c and is given by

$$E_{write} = \frac{1}{2}C_c(X)^2 \tag{2.4}$$

This means that using a smaller pulse magnitude accompanied by longer pulse width is preferable to the other way around in the context of write energy dissipation for the same

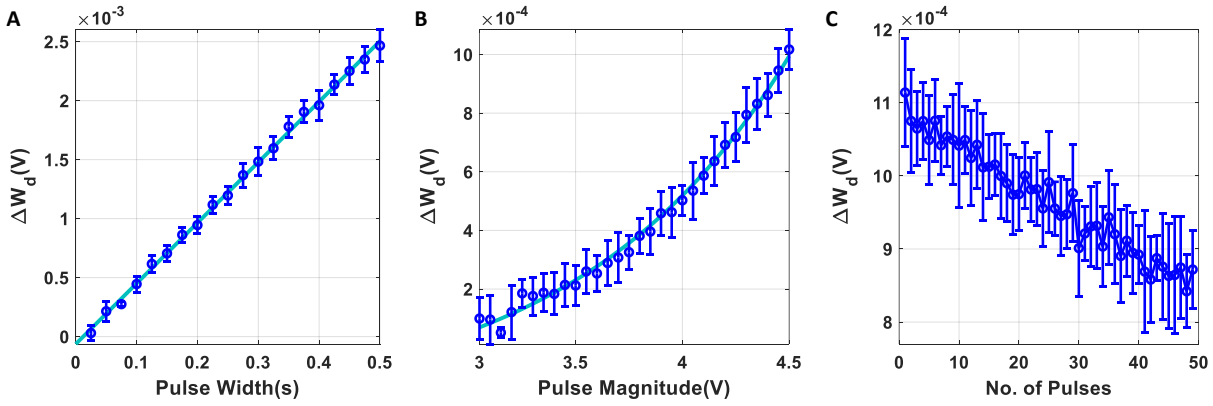

Figure 2.3: : (A) Dependence of change in magnitude of weight with change in pulse-width which follows a linear trajectory defined by $y = mx + c$ (where $m = 0.005136$ and $c = -6.227 \times 10^{-5}$). (B) Dependence on pulse magnitude of the input pulse which follows an exponential trajectory defined by $y = c \times exp(ax + b) + d$ (where $a = 1$, $b = -6.611$, $c = 0.009959$ and $d = -0.0002142$). (C) Change in the magnitude of successive weight updates (ΔW_d) corresponding to a repeated stimulus.

desired change in weight. However, this would come at a cost of slower writing speed. Therefore, a trade-off exists. For the fabricated FN-synapse prototype, the magnitude of the coupling capacitor C_c is approximately 200f F which leads to 400f J for an input voltage pulse change of 2V across C_c. For the differential input voltage pulse of 4V, a total of 800f J of energy was dissipated for each potentiation and depression of the synaptic weights. When the common-mode W_c is not held fixed, irrespective of whether the weight W_d is increased or decreased (depending on the polarity of the input signal) the common-mode always decreases. Thus, W_c serve as an indicator of the usage of the synapse. Fig. 2.3 (c) shows the *metaplasticity* exhibited by an FN-synapse where we measured ΔW_d as a function of usage by applying successive *potentiation* input pulses of constant magnitude (4V) and width (100ms). Fig. 2.3 (c) shows that when the synapse is modulated with the same excitation successively, the amount of weight update decreases monotonically with increasing usage, similar to the response illustrated in Fig. 2.1 (c) and (f).

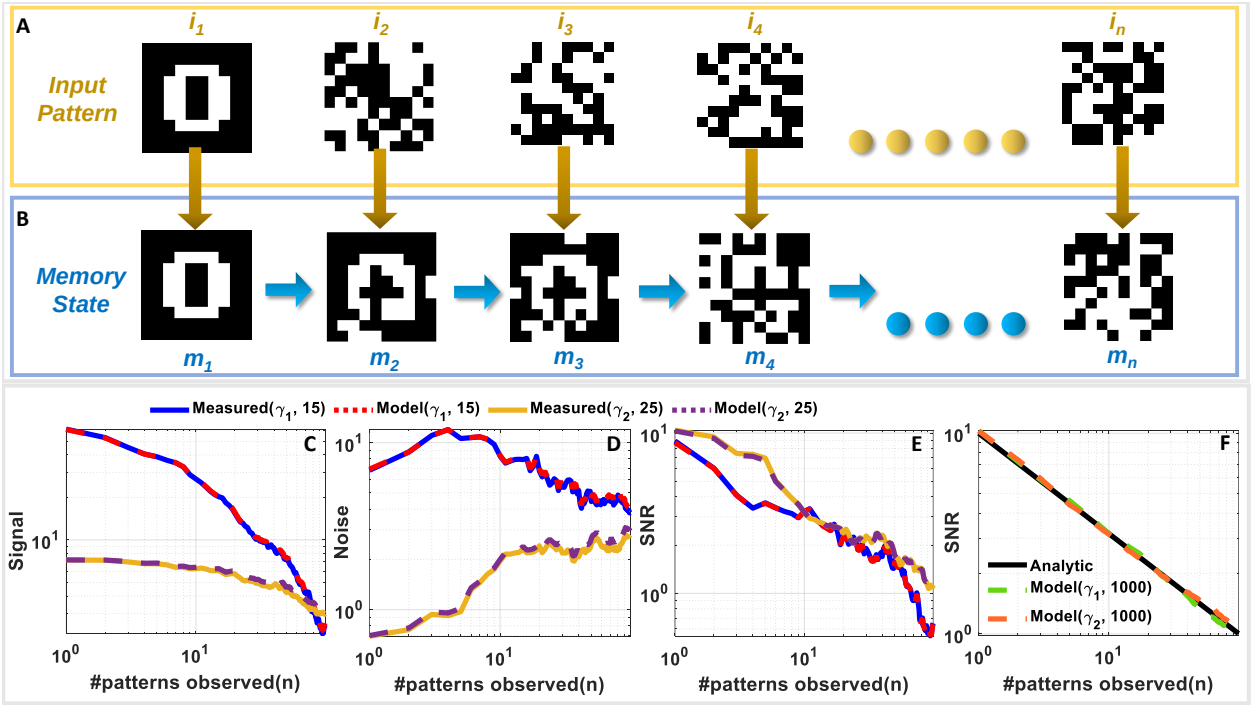

Figure 2.4: : (A) Set of 10×10 randomized noise inputs fed to a network of 100 FN-synapses initialized to store an image of the number 0 and (B) the corresponding memory evolution. Comparison of (C) signal strength, (D) noise strength and (E) SNR for a network size of 100 synapses measured using the fabricated FN-Synapse array (shown in Fig. 2.1(F)) over 25 (for γ_1) and 15 (for γ_2) Monte-Carlo runs. (F) SNR comparison of the γ_1 and γ_2 models with the analytical model over 1000 Monte Carlo simulations. The legends associated with the plots are specified as (γ, Number of Monte-Carlo runs).

2.2.2 FN-synapse Network Capacity and Memory lifetime without plasticity modulation

The next set of experiments was designed to understand the FN-synaptic memory consolidation characteristics when the array is excited using a random binary input pattern (*potentiation* or *depression* pulses). This type of benchmark experiment is used extensively in memory consolidation studies [15, 69] since analytical solutions exist for limiting cases that can be used to validate and compare the experimental results. A network comprising of N FN-synapses is first initialized to store zero weights (or equivalently $W^- = W^+$). New memories were presented as random binary patterns (N dimensional random binary vector) that are applied to the N FN-synapses through either *potentiation* or *depression* pulses. Each synaptic element was provided with balanced input i.e. an equal number of *potentiation*

and *depression* pulses. The goal of this experiment is to track the strength of a memory that is imprinted on this array in the presence of repeated new memory patterns. This is illustrated in Fig 2.4 (a) and (b) where an initial input pattern (a 2D image of the number '0' comprising of 10x10 pixels) is written on a memory array. The array is then subjected to images of noise patterns that are statistically uncorrelated to the initial input pattern. It can be envisioned that as additional new patterns are written to the same array, the strength of a specific memory (of the image '0') will degrade. Similar to the previous studies [15, 69] we quantify this degradation in terms of signal-to-noise ratio (SNR). If n denotes the number of new memory patterns that have been applied to an empty FN-synapse array (initial weight stored on the network is zero), then the Methods Section shows that for the p^{th} update the retrieval memory signal $S(n,p)$ power, the noise $\nu(n,p)$ power and the $SNR(n,p)$ can be expressed analytically as

$$S^2(n,p) = \frac{1}{(n+\gamma)^2}; \quad \nu^2(n,p) = \frac{n}{N(n+\gamma)^2}; \quad SNR(n,p) = \sqrt{\frac{N}{n}}. \tag{2.5}$$

where $\gamma > 0$ is a device parameter that depends on the initialization condition, material properties, and duration of the input stimuli.

Equation 2.5 shows that the initial SNR is $\sqrt{N}$ and the SNR falls off according to a power-law decay with a slope of $\frac{1}{\sqrt{n}}$. Like previous consolidation studies [15] we will assume that a specific memory pattern is retained as long as its SNR exceeds a predetermined threshold (unity in this experiment). Therefore, according to equations 2.5 the network capacity and memory lifetime for FN-synapse scales linearly with the size of the network N when the initial weight across all synapses is zero. We verified the analytical expressions in equation 2.5 for a network size of $N = 100$ using results measured from the FN-synapse chipset. Details of the hardware experiment are provided in the Methods. Fig. 2.4 (c), (d), and (e) shows the retrieval signal, noise, and SNR obtained from the fabricated FN-synapse network for two different values of γ. We observe that the SNR obtained from the hardware results conforms to the analytical expressions relatively well. The slight differences can be attributed to the Monte-Carlo simulation artifacts (only 25 and 15 iterations were carried out). In Fig. S3 we show verification of these analytic expressions using a behavioral model of the FN-synapse which mimics the hardware prototype with great accuracy (as shown in Fig. S2). Details on the derivation of FN-synapse model are provided in the Methods Section. The simulated results in Fig. 2.4 (c), (d), and (e) verify that results from the software model can accurately

track the hardware FN-synapse measurements for both values of γ when subjected to the same stimuli. Therefore, FN-synapse and its behavioral model can be used interchangeably. The results in Fig. 2.4 (f) also show that when the number of iterations on the Monte-Carlo simulation is increased (1000 iterations), the simulated SNR closely approximates the analytic expression. This verifies that hardware FN-synapse is also capable of exactly matching the optimal analytic consolidation characteristics. Fig. 2.3 (c) shows the measured evolution of weights stored in the FN-synapse where initially the weights grow quickly but after a certain number of updates settle to a steady value irrespective of new updates. This implies that the synapses have become rigid with an increase in their usage. This type of memory consolidation is also observed in EWC models which have been used for continual learning. However, note that, unlike EWC models that need to store and update some measure of Fisher information, whereas, here the physics of the FN-synapse device itself can achieve similar memory consolidation without any additional computation.

2.2.3 Plasticity modulation of FN-Synapse Models

In our next set of experiments, we verified that the plasticity of FN-synapses can be adjusted to mimic the consolidation properties of both EWC and steady-state models (such as cascade models). While the EWC model only allows for the retention of old memories, steady-state/cascade models allow for both memory retention and forgetting. As a result, these models avoid *blackout catastrophe* whereas an EWC network is unable to retrieve any previous memories or store new experiences as the network approaches its capacity. Steady-state models allow the network to gracefully forget old memories and continue to remember new experiences *indefinitely*. For an FN-synapse network, a coupling capacitor in each synapse (shown in Fig. S1) which is driven by a global voltage signal $V_{mod}(t)$ (which produces $m(t) = \frac{dV_{mod}(t)}{dt}$) can control the plasticity of the FN-synapse to mimic the characteristics of a steady-state model. Details of the modified FN-synapse achieving a steady-state response are provided in the Methods Section. To understand and compare the blackout catastrophe in FN-synapse models with a steady-state model e.g. the cascade model we define the metric *#patterns.retained* as the total number of memory patterns whose SNR exceeds 1 at any given point of time. The *#patterns.retained* for FN-synapse network with modulation profiles $m_0(t)$, $m_1(t)$, $m_2(t)$, $m_3(t)$ and $m_4(t)$ of size $N = 1000$ is shown in Fig. 2.5 (a) together with those for cascade models of different levels of complexity [15] (denoted by

$c = 1, .., 5$).In order to calculate the *#patterns.retained* the SNR resulting from each stimulus was calculated and tracked at every observation to determine the number of such stimuli that had a corresponding SNR greater than unity. The profiles of $m_1(t)$, $m_2(t)$ and $m_3(t)$ are produced by changing $V_{mod}(t)$ at each update as three quarter, half and quarter of the average of ΔW_d across all the synapses during the latest update respectively while $m_0(t)$ is achieved through a constant voltage signal $V_{mod}(t)$. We can observe in Fig. 2.5 (a) that the FN-synapse network with $m_0(t)$ forgets all observed patterns in addition to not forming any new memories as *#patterns.retained* goes to zero as the network capacity is reached starting from an empty network. Whereas in the case for FN-synapse under $m_1(t)$ and $m_2(t)$ modulation profile the *#patterns.retained* reaches a finite value similar to that of the cascade models. This indicates that the FN-synapse network when subjected to plasticity modulation profiles continues to form new memory while gracefully forgetting the old ones. For the $m_3(t)$ modulation profile the network is slowly evolving and has yet to reach the steady state condition within 2000^{th} update. The FN-synapse network under the $m_4(t)$ modulation profile, which switches between $m_0(t)$ and $m_1(t)$ periodically, is in an oscillatory steady-state with the same periodicity as the modulation profile itself. However, note that the network does not suffer from blackout catastrophe and has a variable capacity. This shows that the capacity of the FN-synapse network can also be tuned to the specificity of different applications. From the figure, we also observe that the steady state network capacity for $m_2(t)$ modulation profile is higher than that of cascade models. Note here that network capacity for cascade models may be increased by increasing the complexities of the synaptic model. Nevertheless, we find that network capacity for FN-synapse is comparable to cascade models of moderate complexities.

In order to understand the plasticity modulation further, we investigated the SNR for patterns introduced to a non-empty network. For this experiment, we tracked the 1000^{th} pattern observed by the network of $N = 1000$ synapse. Fig 2.5 (b) shows the SNR of this pattern under $m_1(t) - m_4(t)$ modulation profile along with cascade models of various complexity. Note that the x-axis now represents the age of the stimulus, i.e. the number of patterns observed after the tracked pattern. For the modulation profile $m_1(t)$ the initial SNR is large, comparable to that of cascade models, but the SNR falls off quickly indicating high plasticity. Whereas for modulation profile $m_2(t)$ and $m_3(t)$ the initial SNR is smaller than $m_1(t)$ but it falls off at a much later time similar to cascade models with high complexities. These SNR profiles for FN-synapse model with modulation $m_1(t) - m_3(t)$ are similar to that of a constant weight decay synaptic model used in deep learning neural network as a regularization method.

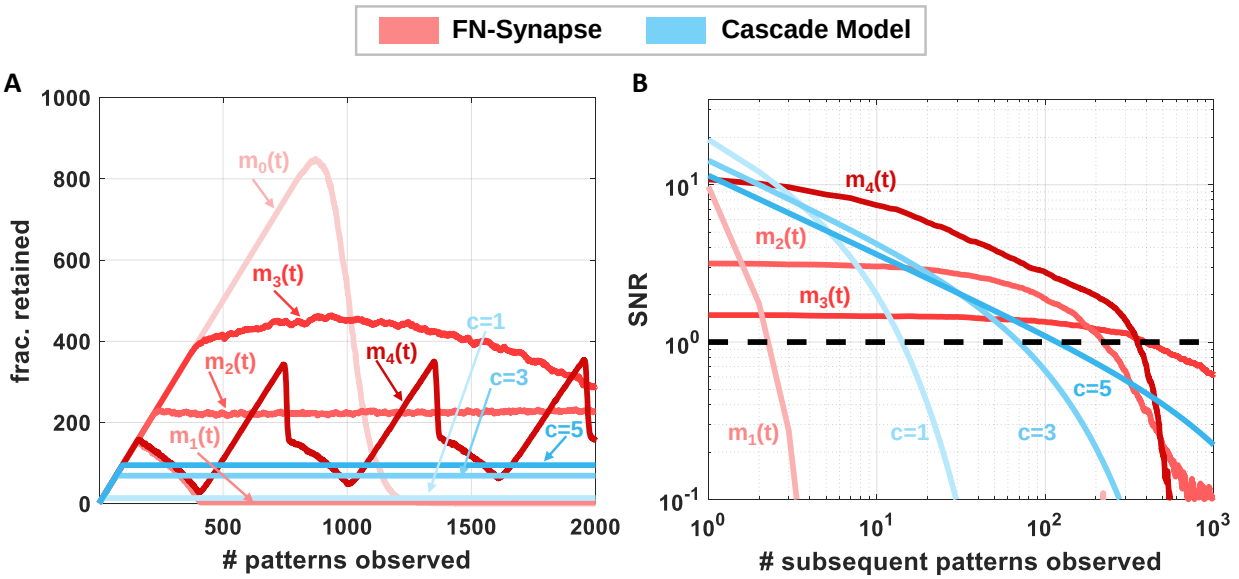

Figure 2.5: : Comparison of (A) no. of patterns retained by networks composed of 1000 synapses following different synaptic models when exposed to 2000 patterns and (B) steady-state SNR of the 1000^{th} update ($p = 1000$) of networks consisting of 1000 synapses with various synaptic models when exposed to subsequent updates.

On the other hand, the SNR profile for the 1000^{th} pattern under $m_4(t)$ modulation has both high initial SNR and a large lifetime. However, from Fig 2.5 (b) we observe that the network is in an oscillatory state which indicates that this profile is specific to the 1000^{th} pattern, and if we tracked any other pattern the SNR profile would be different. This is not the case for the cascade models which would consistently have similar SNR profiles irrespective of the pattern that is tracked. Nevertheless, this SNR profile for the FN-synapse model would repeat itself corresponding to the periodicity of the modulation profile. This suggests that the amount of plasticity and memory lifetime for the FN-synapse model is readily tunable and depends on the amount of modulation provided to the network. We have also verified that the synaptic strength of FN-synapse is bounded similarly to that of the cascade models. This can be observed in Fig. S11 which shows that the variance in retrieval signal (Noise) of an FN-synapse network with both constant modulation and time-varying modulations remains bounded. Furthermore, Fig. S12 shows that plasticity modulation indeed introduces a forgetting mechanism as the SNR for different modulation profiles (when tracked from an empty network) starts to fall off earlier than the one without modulation. In addition to different modulation profiles, the plasticity-lifetime tradeoff of the FN-synapse model can also be achieved by varying the parameter γ as shown in Fig S13. Therefore, our synaptic

models can exhibit memory consolidation properties similar to both EWC and steady-state models while being physically realizable and scalable for large networks.

2.2.4 Continual Learning using FN-synapse

The next set of experiments was designed to evaluate the performance of FN-synapse neural network for a benchmark continual learning task. A fully-connected neural network with two hidden layers was trained sequentially on multiple supervised learning tasks. Details of the neural network architecture and training are given in the Methods Section and in Appendix A. The network was trained on each task for a fixed number of epochs and after the completion of its training on a particular task t_n, the dataset from t_n was not used for the successive task t_{n+1}.

The aforementioned tasks were constructed from the Modified National Institute of Standards and Technology (MNIST) dataset, to address the problem of classifying handwritten digits in accordance with schemes popularly used in several continual-learning literature [58]. Also known as incremental domain learning using split-MNIST dataset, each task of this continual learning benchmark dictates the neural network to be trained as a binary classifier which distinguishes between a set of two hand-written digits, i.e. the network is first trained to distinguish between the set $[0, 1]$ as t_1 and is then trained to distinguish between $[2, 3]$ in t_2, $[4, 5]$ in t_3, $[6, 7]$ in t_4 and $[8, 9]$ in t_5. Thus, the network acts as an even-odd number classifier during every task.

Fig. S7 (a)-(e) compares the task-wise accuracy of networks trained with different learning and consolidation approaches. Note here that the absence of a data-point corresponding to a particular approach indicates that the accuracy obtained is below 50%. All the approaches taken into consideration perform equally well at learning t_1 as illustrated in Fig. S7 (a). However, as the networks learn t_2 (see Fig. S7 (b)), the performance of both EWC [69] and online EWC [82] degrade for task t_1 as do the networks with conventional memory using SGD and ADAM. The FN-synapse based networks on the other hand retain the accuracy of task t_1 far better in comparison. This advantage in retention comes at the cost of learning t_2 marginally poorer than others. This trend of retaining the older memories or tasks far better than other approaches continues in successive tasks. Particularly, if we consider the retention of t_1 when the networks are trained on t_3 (see Fig. S7 (c)), it can be observed

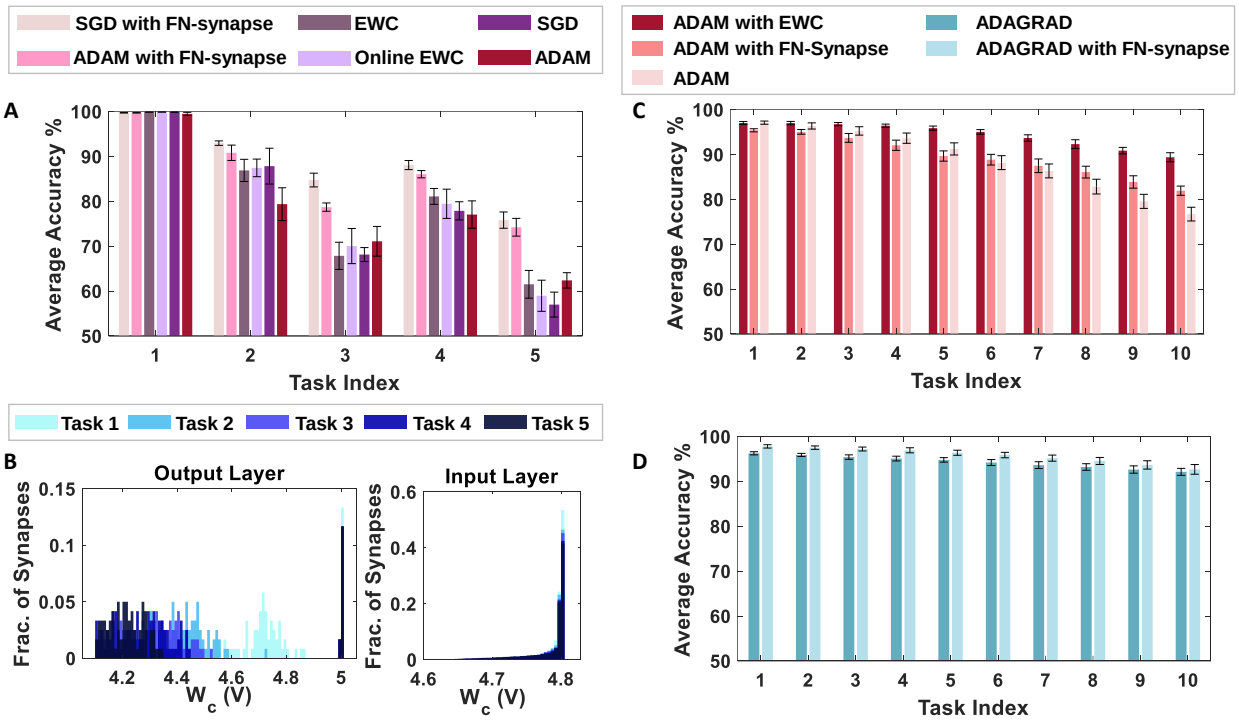

Figure 2.6: : (A) Overall average accuracy comparison of SGD and ADAM with FN-synapse, ADAM with EWC and Online EWC, SGD and ADAM with conventional memory. (B) Distribution of the usage profile of weights in the output layer and the input layer of the FN-synapse neural network. Overall Average Accuracy comparison of incremental-domain learning scenarios on the Permuted MNIST dataset using (C)ADAM with EWC, ADAM with FN-Synapse and ADAM with conventional memory and (D)ADAGRAD with conventional memory and ADAGRAD with FN-synapse

that it is only the FN-synapse based networks that retain t_1 while others fall below the 50% threshold. Similar trends can be observed in Fig. S7 (d) and (e). There are a few instances during the five tasks where the EWC variants and SGD with conventional memory marginally outperform or match the FN-synapse in terms of retention. However, if the overall average accuracy of all these approaches is compared (see Fig. 2.5 (c)), it is clearly evident that both the FN-synapse networks significantly outperform the others. It is also worth noting here that even when a network equipped with FN-synapse is trained using a computationally-inexpensive optimizer such as SGD, it shows remarkably superior performance than highly computationally-expensive approaches such as ADAM with conventional memory and ADAM with EWC variants.

The only drawback of the FN-synapse based approach is that its ability to learn the present task slightly degrades with every new task. This phenomenon results from the FN-synapses becoming more rigid and can be seen in Fig. 2.5 (d) which shows the evolution of plasticity of weights in the output and input layer of the network with successive tasks with respect to W_c. As mentioned earlier, W_c keeps track of the importance of each weight as a function of the number of times it is used. The higher the W_c of a particular weight, the less it has been used and therefore, the more plastic it is and sensitive to change. On the other hand, a more rigid and frequently used weight has a lower value of W_c. Suppose the output layer is considered from Fig. 2.5 (d). In that case, it can be observed that with each successive task the W_c of the weights of the network collectively reduces, leading to more consolidation and consequently leaving the network with fewer plastic synapses to learn a new task. In comparison, the majority of the weights in the input layer remain relatively more plastic (or less spread out) owing to the redundancies in the network arising from the vanishing gradient problem (see Discussion for more details). In Fig. S5 we show that the ability of the network to learn or forget new tasks is a function of the initial plasticity of the FN-synapses and can be readily adjusted.

In addition to the split-MNIST benchmark, the performance of FN-synapse based network was compared with EWC for the permuted MNIST benchmark. These incremental-domain learning experiments were carried out by randomly permuting the order of pixels of the images in the MNIST dataset in accordance with [58] to create new tasks. The overall average accuracy for 10 Monte Carlo simulations when using ADAM as the optimizer with EWC, FN-Synapse and conventional memory is depicted in Fig. 2.6 (c). We can observe from Fig 2.6 (c) that despite not being as retentive as EWC in this particular scenario, the network equipped with FN-synapse as the memory element performs better than the network without any memory consolidation mechanism, thereby exhibiting continual learning ability. Furthermore, when compared to a network with traditional memory employing an optimizer like ADAGRAD, which has been shown to be suitable for this learning scenario [58], the FN-synapse network with ADAGRAD exhibits marginal improvements without any drop in performance with respect to the former as shown in Fig. 2.6 (d).

2.3 Materials and Methods

The main methods are described in this section while Appendix A includes additional details, supporting information, and figures.

2.3.1 Weight Update For Differential Synaptic Model

Consider the differential synaptic model described by Fig. 2.1 (c) where the evolution of two dynamical systems with state variables W^+ and W^- is governed by

$$\frac{dW^+}{dt} = -J(W^+) + \frac{1}{2}X(t) + \frac{1}{2}m(t) \tag{2.6}$$

$$\frac{dW^-}{dt} = -J(W^-) - \frac{1}{2}X(t) + \frac{1}{2}m(t) \tag{2.7}$$

where $J(.)$ is an arbitrary function of the state variables, $+\frac{1}{2}X(t)$ or $-\frac{1}{2}X(t)$ are differential time varying inputs and $m(t)$ is a common mode modulation input. In this differential architecture, we define the weight parameter W_d as $W_d = \frac{1}{2}(W^+ - W^-)$ which represents the memory and the common-mode parameter W_c as $W_c = \frac{1}{2}(W^+ + W^-)$ which represents the usage of the synapse. Applying this definition to 2.6 and 2.7, we obtain:

$$\frac{d(W_c + W_d)}{dt} = -J(W_c + W_d) + \frac{1}{2}X(t) + \frac{1}{2}m(t) \tag{2.8}$$

$$\frac{d(W_c - W_d)}{dt} = -J(W_c - W_d) - \frac{1}{2}X(t) + \frac{1}{2}m(t) \tag{2.9}$$

Now, adding and subtracting 2.8 and 2.9, we get:

$$\frac{dW_c}{dt} = -\left(\frac{J(W_c + W_d) + J(W_c - W_d)}{2}\right) + m(t) \tag{2.10}$$

$$\frac{dW_d}{dt} = -\left(\frac{J(W_c + W_d) - J(W_c - W_d)}{2}\right) + X(t) \tag{2.11}$$

Assuming that $W_c >> W_d$, applying Taylor series expansion on 2.10 and 2.11 leads to

$$\frac{dW_c}{dt} = -J\left(W_c\right) + m(t) \tag{2.12}$$

$$\frac{dW_d}{dt} = -J'\left(W_c\right) W_d + X(t). \tag{2.13}$$

This means that the modulation input impacts the usage of the synapse. Therefore, the plasticity of the synapse can be *tuned* using $m(t)$ when needed. Now we first look into the trivial case when a constant modulation input is provided, i.e. $m(t) = c$ where c is an arbitrary constant. In this scenario the plasticity of the synapse is solely dependent on the usage of the synapse as $m(t)$ does not change with time. Substituting the derivative of W_c from 2.12, when $m(t)$ is constant, into 2.13, the rate of change in W_d can be formulated as:

$$\frac{dW_d}{dt} = -\left[\frac{d^2W_c}{dt^2}\left(\frac{dW_c}{dt}\right)^{-1}\right] W_d + X(t) \tag{2.14}$$

Please refer to the Appendix A for detailed derivation. Equation 2.14 shows that the change in weight ΔW_d is directly proportional to the *curvature* of usage while being inversely proportional to the rate of usage.

2.3.2 Optimal Usage Profile

We define the decaying term in 2.14 as

$$r(t) = -\left[\frac{d^2W_c}{dt^2}\left(\frac{dW_c}{dt}\right)^{-1}\right] \tag{2.15}$$

Now, comparing the weight update equation in 2.14 to the weight update equation for EWC in the balanced input scenario, the decay term has the following dependency with time for avoiding catastrophic forgetting.

$$r(t) = O\left(\frac{1}{t}\right) \tag{2.16}$$

Now, the usage of a synapse is always monotonically increasing and since W_c represents the usage, it too needs to be monotonic. At the same time, W_c also needs to be bounded, therefore

W_c has to monotonically decrease with increasing usage while satisfying the relationship in equation 2.16. It can be shown that equation 2.16 and 2.15 can be satisfied by any dynamical system of the form

$$W_c = \frac{1}{f(\log t)} \tag{2.17}$$

where $f(.) \geq 0$ is any monotonic function. Substituting equation 2.17 in 2.15 we obtain the corresponding usage profile as follows

$$r(t) = \frac{1}{t}\left(1 + \frac{2f'(\log t)}{\log t} - \frac{f''(\log t)}{f'(\log t)}\right) \tag{2.18}$$

where $f'(\log t)$ and $f''(\log t)$ are derivatives of $f(\log t)$ with respect to $\log t$. While several choices of $f(.)$ are possible, the simplest usage profile can be expressed as

$$W_c = \frac{\beta}{\log(t)} \tag{2.19}$$

where β is an arbitrary constant. The corresponding non-linear function in this model is determined by substituting equation 2.19 in equation 2.12 to obtain

$$J\left(W_c\right) = \frac{1}{\beta} W_c^2 \exp\left(-\frac{\beta}{W_c}\right). \tag{2.20}$$

The expression for $J(.)$ in equation 2.20 bears similarity with the form of FN quantum-tunneling current [79] and Fig. 2.1(d)-(f) shows the realization of equations 2.6 and 2.7 using FN tunneling junctions.

2.3.3 Achieving Optimal Usage Profile on FN-synapse

For the differential FN tunneling junctions shown in Fig. 2.1(f) and its equivalent circuit shown in the Fig. S1, the dynamical systems model is given by

$$C_T \frac{dW^+}{dt} = -J(W^+) + \frac{C_c}{2}\frac{dv_{in}}{dt} \tag{2.21}$$

$$C_T \frac{dW^-}{dt} = -J(W^-) - \frac{C_c}{2}\frac{dv_{in}}{dt} \tag{2.22}$$

where W^+, W^- are the tunneling junction potentials, C_c is the input coupling capacitance, $v_{in}(t)$ is the input voltage to the coupling capacitance and $C_T = C_c + C_{fg}$ is the total capacitance comprising of the coupling capacitance and the floating-gate capacitance C_{fg}. $J(.)$ are the FN tunneling currents given by

$$J\left(W^+\right) = \left(\frac{k_1}{k_2}\right)\left(W^+\right)^2 \exp\left(-\frac{k_2}{W^+}\right) \tag{2.23}$$

$$J\left(W^-\right) = \left(\frac{k_1}{k_2}\right)\left(W^-\right)^2 \exp\left(-\frac{k_2}{W^-}\right) \tag{2.24}$$

where k_1 and k_2 are device-specific and fabrication-specific parameters that remain relatively constant under isothermal conditions. Following the derivations in the previous sections and the expression in equation 2.19 leads to a common-mode voltage W_c profile as

$$W_c(t) = \frac{k_2}{\log(k_1 t + k_0)} \tag{2.25}$$

where $k_0 = \exp\left(\frac{k_2}{W_{c0}}\right)$ and W_{c0} refers to the initial voltage at the floating-gate.

2.3.4 FN-synpase Network SNR Estimation for Random Pattern Experiment

Upon following the same procedure used in previous sections, the weight update equation for an FN-synapse using equation 2.21 and equation 2.22 can be expressed as

$$C_T \frac{dW_d}{dt} = -\left[\frac{d^2W_c}{dt^2}\left(\frac{dW_c}{dt}\right)^{-1}\right] W_d + C_c\frac{dv_{in}}{dt} \tag{2.26}$$

We designed the floating-gate potential and the input voltage pulses such that the FN-dynamics is only active when there is a memory update. Therefore, the dynamics in equation 2.26 evolve in a discrete manner with respect to the number of modulations. Assuming $C_T = C_c$

we formulate a discretized version of the weight update dynamics from equation 2.26 in accordance with the floating-gate potential profile of the device expressed in equation 2.25 as follows

$$\frac{\triangle W_d(n)}{\triangle t} = -k_1 \left(1 + \frac{2}{\log\left(k_1 \triangle t n + k_0\right)}\right) \left(\frac{1}{k_1 \triangle t n + k_0}\right) W_d(n-1) + \frac{\triangle v_{in}(n)}{\triangle t} \tag{2.27}$$

$$W_d(n) = \left[1 - \left(1 + \frac{2}{\log\left(k_1 \triangle t n + k_0\right)}\right) \left(\frac{1}{n + \frac{k_0}{k_1 \triangle t}}\right)\right] W_d(n-1) + \left(v_{in}(n) - v_{in}(n-1)\right) \tag{2.28}$$

where n represents the number of patterns observed and Δt is the duration of the input pulse. Let us denote the weight decay term as

$$\alpha(n) = \left[1 - \left(1 + \frac{2}{\log\left(k_1 \triangle t n + k_0\right)}\right) \left(\frac{1}{n + \frac{k_0}{k_1 \triangle t}}\right)\right] \tag{2.29}$$

Thus, we obtain the weight update equation with respect to the number of patterns observed as

$$W_d(n) = \alpha(n) W_d(n-1) + \left(v_{in}(n) - v_{in}(n-1)\right) \tag{2.30}$$

When we start from an empty network i.e. $W_d(0) = 0$, the memory update can be expressed as a weighted sum over the past input as

$$W_d(n) = \sum_{i=1}^{n-2} \left\{ \left(\alpha(i+1) - 1\right) \left(\prod_{j=i+2}^{n} \alpha(j)\right) v_{in}(i) \right\} + \left(\alpha(n) - 1\right) v_{in}(n-1) + v_{in}(n) \tag{2.31}$$

We define the retrieval signal and the noise associated with it as per the definition in [15]. For a network comprising of N synapses, each weight in the network is indexed as $W_d(a, n)$

where $a = 1, ..., N$. Similarly, the input applied to the a^{th} synapse after n patterns is $v_{in}(a, n)$. Then, the signal strength for the p^{th} update (where $p < n$) introduced to the initially empty network tracked after n patterns can be formulated as:

$$S(n,p) = \frac{1}{N}\left\langle \sum_{a=1}^{N} W_d(a,n) v_{in}(a,p) \right\rangle \tag{2.32}$$

where angle brackets denote averaging over the ensemble of all of the input patterns seen by the network. If we assume that the input patterns are random binary events of ± 1 and are uncorrelated between different synapses and memory patterns then substituting equation 2.31 in 2.32 we obtain

$$S(n,p) = (\alpha(p+1) - 1) \prod_{j=p+2}^{n} \alpha(j) \tag{2.33}$$

Given that in equation 2.29, $k_0 = \mathcal{O}(10^{19})$ and $k_1 = \mathcal{O}(10^{16})$, the term $\left(1 + \frac{2}{\ln(k_1 \triangle t n + k_0)}\right) \approx 1$, the signal power simplifies to:

$$S^2(n,p) = \frac{1}{(n+\gamma)^2} \tag{2.34}$$

where $\gamma = \frac{k_0}{k_1 \triangle t}$ and depends on the pulse-width $\triangle t$ and the initial condition k_0. The above equation shows that the signal's strength is a function of the system parameter γ and decays with the number of memory patterns observed. If we assume that the weight $W_d(n)$ is uncorrelated from the input $v_{in}(n)$ and that the inputs $v_{in}(1), v_{in}(2), ...v_{in}(n)$ are uncorrelated from each other, then the corresponding noise power is given by the variance of the retrieval signal expressed in equation 2.32. This can be estimated as the sum of the power of all signals tracked at n except for the retrieval signal corresponding to the p^{th} update we are tracking and is given by:

$$\nu^2(n,p) = \frac{1}{N} \sum_{i=1, i \neq p}^{n} S^2(n,i) \tag{2.35}$$

However, in order to derive a more tractable analytical expression for further analysis we added the retrieval signal as well into the summation which introduces a small error in

the estimation (overestimating the noise by the retrieval signal term). This leads us to the following estimation of the noise power:

$$\nu^2(n,p) = \frac{n}{N(n+\gamma)^2} \tag{2.36}$$

Based on the value of n in comparison to γ, we obtain two trends for the noise profile. When $\gamma >> n$,

$$\nu(n,p) = \frac{1}{\sqrt{N}}\left(\frac{\sqrt{n}}{\gamma}\right) \tag{2.37}$$

which implies that noise increases with increase in updates initially. On the other hand, when $\gamma << n$,

$$\nu(n,p) = \frac{\sqrt{n}}{\sqrt{N}n} = \frac{1}{\sqrt{N}}\left(\frac{1}{\sqrt{n}}\right) \tag{2.38}$$

which implies that noise falls with increase in updates in the later stages. The signal-to-noise ratio (SNR) of a network of size N can then be obtained as:

$$SNR(n,p) = \sqrt{\frac{S^2(n,p)}{\nu^2(n,p)}} = \sqrt{\frac{N}{n}} \tag{2.39}$$

2.3.5 FN-synapse with Tunable Consolidation Characteristics

In the previous sections, we derived the analytical expressions for the memory retrieval signal, the noise associated with it, and the corresponding SNR for the case when the modulation input $m(t)$ was kept constant. This led to a synaptic memory consolidation which is similar to that of EWC. However, blackout catastrophic forgetting occurs in networks with such memory consolidation due to the absence of a balanced pattern retention and forgetting mechanism. The $forgetting$ mechanism is naturally present in a steady state model such as the cascade model which does not suffer from memory "blackouts". Since the increase in *retention* is equivalent to an increase in rigidity and $forgetting$ is tantamount to a decrease in rigidity, it is necessary to adjust the plasticity/rigidity of the synapse accordingly. From Fig. 2.2 (a) and (b) we notice that without external modulation W_c decreases monotonically

with each new update which correspondingly makes the synapse only rigid. Therefore, to balance the same, the idea is to keep W_c as steady as possible to keep the synapse plastic as long as possible by applying a modulation profile $m(t)$ that *recovers/restores* W_c after every synaptic update. This results in $m(t)$ of the form

$$m(t) = m(i)\delta(t - iT) \tag{2.40}$$

where $\delta(t)$ is the Dirac-delta, $m(i)$ is the magnitude of the modulation increment, and T is the time between each modulation increment. This increment is determined by the rate of the differential update to the FN-synapse. Integrating this form of $m(t)$ into equation 2.12 leads to

$$\frac{dW_c}{dt} = -J\left(W_c\right) + m(i)\delta(t - iT) \tag{2.41}$$

which implies a tunable plasticity profile for the FN-synapse. An analytical solution to the differential equation 2.41 is difficult and hence we resort to a recursive solution. Due to the nature of the $m(t)$, it can be seen that the initial condition of the variable W_c changes at increments of T, whereas between two modulation increments W_c evolves naturally according to equation 2.25. Thus, the dynamics of W_c in the presence of the modulation increments can be described as

$$W_c(t) = \begin{cases} W_{c0} & ; \quad t = 0 \\ W_c(t) + V_{mod}(t) & ; \quad t = iT \\ \frac{k_2}{\log(k_1(t-iT)+\exp(\frac{k_2}{W_c(iT)}))} & ; \quad iT < t < (i+1)T \end{cases} \tag{2.42}$$

where $V_{mod}(t)$ is an external voltage signal applied to the FN-synapse as shown in Fig. A.1 in the Appendix A and is given by:

$$V_{mod}(t) = \sum_{i=1}^{\infty} m(i)\delta(t - iT) \tag{2.43}$$

In this case the change in plasticity of the synapse is determined by the step-size of the staircase voltage function $V_{mod}(t)$. Note that the weight update equation in 2.13 is still valid since $m(t)$ is kept constant during differential input.

Although an analytic expression for the SNR is no longer tractable in this iterative form, the ability of the modulation term to regulate the plasticity and induce a more graceful form of forgetting is shown in the corresponding no. of patterns retained plot in Fig 2.5(a) and the SNR plot Fig 2.5(b) for various modulation input profiles.

2.3.6 Programming and Initialization of FN-synapses

The potential corresponding to the tunneling nodes W^+ and W^- can be accessed through a capacitively coupled node, as shown in Fig. S1. This configuration minimizes readout disturbances and the capacitive coupling also acts as a voltage divider so that the readout voltage is within the input dynamic range of the buffer. The configuration also prevents hot-electron injection of charge into the floating gate during readout operation. Details of initialization and programming are discussed in [90], so here we describe the methods specific for this work. The tunneling node potential was initialized at a specific region where FN-tunneling only occurs while there is a voltage pulse at the input node and the rest of the time it behaves as a non-volatile memory. This was achieved by first measuring the readout voltage every 1 second for a period of 5 min to ensure that the floating gate was not discharging naturally. During this period the noise floor of the readout voltage was measured to be $\approx 100\mu V$. At this stage, a voltage pulse of magnitude 1 V and duration 1 ms was applied at the input node and the change in readout voltage was measured. If the change was within the noise floor of the readout voltage, the potential of the tunneling nodes were increased by pumping electrons out of the floating gate using the program tunneling pin.

This process involves gradually increasing the voltage at the program tunneling pin to 20.5 V (either from external source or from on-chip charge pump). The voltage at the program tunneling pin was held for a period of 30s, after which it was set to 0 V. The process was repeated until a substantial change in the readout voltage was observed ($\approx 300\mu V$) after providing an input pulse. The readout voltage in this region was around 1.8 V.

2.3.7 Hardware and Software Experiments for Random Pattern updates

The fabricated prototype contained 128 differential FN tunneling junctions, which correspond to 64 FN-synapses. However, due to the peripheral circuitry only one tunneling node could be accessed at a time for readout and modification. Now, since the memory pattern is completely random, each synapse can be modified independently without affecting the outcome of the experiment. Therefore, two tunneling nodes were initialized following the method described in the aforementioned section. Input pulses of magnitude 4V and duration 100ms were applied to both the tunneling nodes. The change in the readout voltages was measured, and the region where the update sizes of both the tunneling node would be equal was chosen as the initial zero memory point for the rest of the experiment. The nodes were then modified with a series of 100 *potentiation* and *depression* pulses of magnitude 4.5v and duration 250 ms and the corresponding weights were recorded. This procedure represented the 100 updates of a single synapse. The tunneling nodes were then reinitialized to the zero memory point and the procedure was repeated with different random series of input pulses representing the modification of the other 99 synapses in the network. The first input pulses of each series of modifications form the tracked memory pattern. To modify the value of γ the FN-synapses were initialized at a higher tunneling node potential.

The behavioral model of the FN-synapse was generated by extracting the device parameters k_1 and k_2 from the hardware prototype. The extracted parameters have been shown to capture the hardware response with an accuracy greater than 99.5% in our previous works [137, 138]. These extracted parameters were fed into a dynamical system that follows the usage profile described in the hardware implementation subsection and follows the weight update rule elaborated in the SNR estimation subsection to reliably imitate the behavior of the FN-synapse. The behavioral model network was started with exactly the same initial condition as hardware synapses and subjected to the exact memory patterns used for the

hardware experiment for the same number of iterations. The simulation was also extended to 1000 iterations and the corresponding responses are included in Fig 3 (f).

2.3.8 Probabilistic FN-Synapse Model

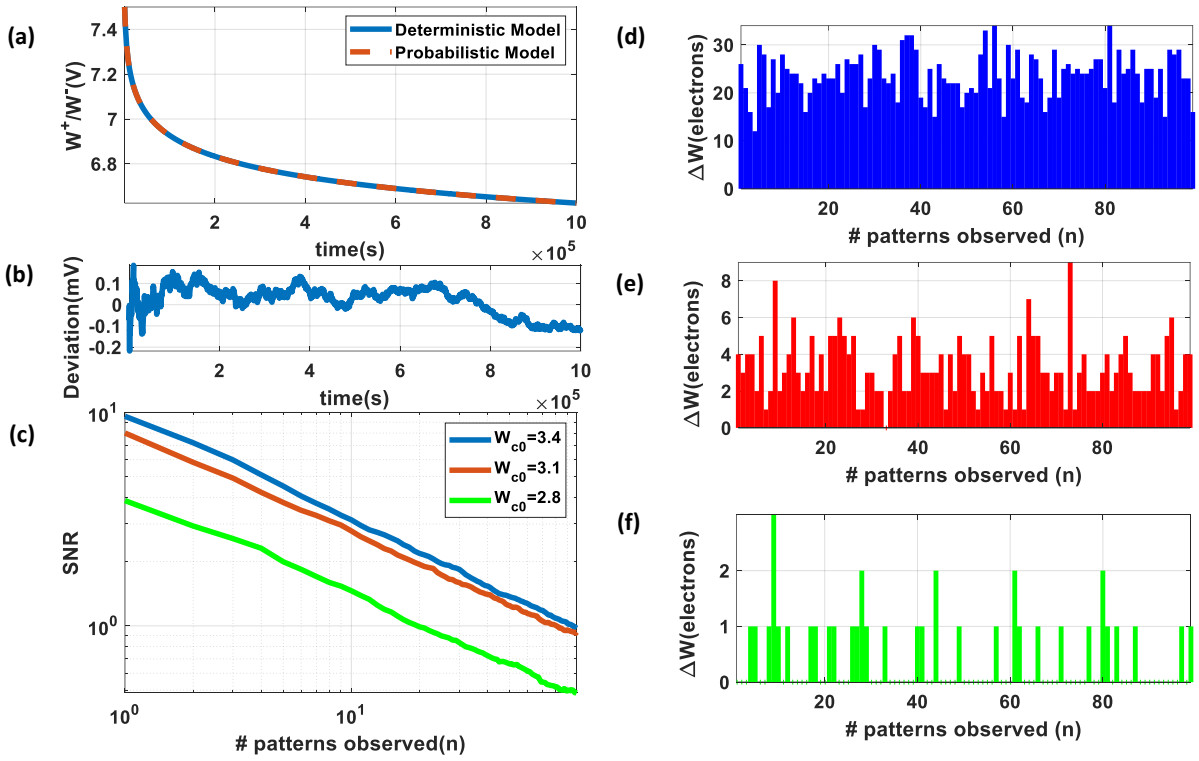

Figure 2.7: (a) Comparison between the output of the probabilistic FN-synapse model and the deterministic behavioral model and the (b) corresponding deviation. (c) The SNR of the network for different tunneling regions for $W_{c0} = 3.4$V, 3.1V, and 2.8V and (d)-(f) their corresponding update size in terms of no. of electrons per update.

Adaption of FN-synapse occurs by the tunneling of electrons through a triangular FN quantum-tunneling barrier. The tunneling current density is dependent on the barrier profile which in turn is a function of the floating-gate potential. When W^+, W^- is around 7 V the synaptic update ΔW_d due to an external pulse can be determined by the continuous and deterministic form of the FN-synapse model (as described in the previous sections). Since the number of electrons tunneling across the barrier is relatively large ($\gg 1$), the method is adequate for determining ΔW_d. However, once W^+, W^- is around 6 V, each update occurs due to the transport of a few electrons tunneling across the barrier and in the limit

by a single electron tunneling across the barrier at a time. In this regime, the continuous behavioral model is no longer valid. Therefore, the behavioral model of the FN-synapse has to switch to a probabilistic model. In this mode of operation, we can assume that each electron tunneling event follows a Poisson process where the number of electrons $e^+(n), e^-(n)$ tunneling across the two junctions during the n^{th} input pulse is estimated by sampling from a Poisson distribution with rate parameters λ^+, λ^- given by

$$\lambda^+(n) = \frac{AJ(W^+(n))}{q} \tag{2.44}$$

$$\lambda^-(n) = \frac{AJ(W^-(n))}{q}. \tag{2.45}$$

q is the charge of an electron, and A is the cross-sectional area of the tunneling junction. Using the sampled values of $e^+(n), e^-(n)$, the corresponding discrete-time stochastic equation governing the dynamics of the tunneling node potentials $W^+(n), W^-(n)$ is given by

$$W^+(n) = W^+(n-1) - \frac{qe^+(n)}{C_T} \tag{2.46}$$

$$W^-(n) = W^-(n-1) - \frac{qe^-(n)}{C_T} \tag{2.47}$$

where C_T is the equivalent capacitance of the tunneling node.

We have verified the validity/accuracy of the probabilistic model against the continuous-time deterministic model in high tunneling rate regimes. Fig. 2.7 (a) shows that the output of the probabilistic model matches closely to the deterministic model and the deviation which arises due to the random nature of the probabilistic updates (shown in Fig. 2.7 (b)) is within $200\mu V$. Using the probabilistic model we performed the memory retention and network capacity experiments (as discussed in the main manuscript) by initializing the tunneling nodes at a low potential. In this regime, each update to the FN synapse results from the tunneling of a few electrons. Fig. 2.7 (c) and (d) show that even when each update sizes are on the order of tens of electrons, the network capacity and memory retention time remains unaffected. However, as the synaptic voltage is modified by less than ten electrons per update (shown in Fig. 2.7 (e)), the SNR curve starts to shift downwards and the network capacity along with memory retention time decreases. The tunneling node potential can be pushed further down to a region where the synapses might not even register modifications at times and at other times update sizes drop down to a single electron per modification (see Fig. 2.7

(f)). In this regime, the SNR curve shifts down further, and the SNR decay still obeys the power-law curve.

2.3.9 Neural Network Implementation using FN-synapses

The MNIST dataset was split into 60,000 training images and 10,000 test images which yielded about 6000 training images and 1000 test images per digit. Each image, originally of 28×28 pixels, was converted to 32x32 pixels through zero-padding. This was followed by standard normalization to zero mean with unit variance. The code for implementing the non-FN-synapse approaches such as EWC and online EWC was obtained from the repository mentioned in [58]. To enforce an equitable comparison, the same neural network architecture (as shown in Fig. S6), in the form of a multi-layered perceptron (MLP) with an input layer of 1024 nodes, two hidden layers of 400 nodes each (paired with the ReLU activation function) and a softmax output layer of 2 nodes, has been utilized by every method mentioned in this work. Based on the optimizer in use, a learning rate of 0.001 was chosen for both SGD and ADAM (with additional parameters β_1, β_2 and ϵ set to 0.9,0.999 and 10^{-8} respectively for the latter). Each model was trained with a mini-batch size of 128 for a period of 4 epochs.

Similar to the continual learning experiments conducted on split-MNIST, benchmark incremental-domain learning experiments were also carried out by randomly permuting the order of pixels of the images in the MNIST dataset in accordance with [58] which is referred to as the Permuted-MNIST. The architecture of the neural network employed is similar to the one for the split-MNIST with the exception of being equipped with 1000 neurons in each of the two hidden layers instead of 400 and with 10 neurons in the output layer instead of 2. This essentially means that at each task, the network learns a new set of permutations of the 10 digits. The network was trained on 10 such tasks for 3 epochs using a learning rate of 0.0001 for ADAM and 0.001 for ADAGRAD.

Corresponding to every weight/bias in the MLP, an instance of the FN-synapse model was created and initialized to a tunneling region according to the initial W_c value. As demonstrated by the measured results, ΔW_d can be modulated linearly and precisely by changing the pulse-width of the *potentiation/depression* pulses. Therefore, each weight update (calculated according to the optimizer in use) is mapped as an input pulse of proportional duration for the FN synapse instance. Then, every instance of the FN-synapse model is updated according

to Eq. 2.27 and the W_d thus obtained in voltage is scaled back to a unit-less value and within the required range of the network.

2.4 Discussion

In this work, we reported a differential FN quantum-tunneling based synaptic device that can exhibit near-optimal memory consolidation that has been previously demonstrated using only algorithmic models. The device called FN-synpase, like its algorithmic counterparts, stores the value of the weight and relative usage of the weight that determines the plasticity of the synapse. Similar to algorithmic consolidation models, an FN-synapse, 'protects' important memory by reducing the plasticity of the synapse according to its usage for a specific task. However, unlike its algorithmic counterparts like the cascade or EWC models, the FN-Synapse doesn't require any additional computational or storage resources. In EWC models memory consolidation in continual learning is achieved by augmenting the loss function using penalty terms that are associated with either Fisher information [69] or the historical trajectory of the parameter over the course of learning [26, 82]. Thus, the synaptic updates require additional pre-processing of the gradients, which in some cases could be computationally and resource intensive. FN-synapse on the other hand, does not require any pre-processing of gradients and instead can exploit the physics of the device itself for synaptic intelligence and for continual learning. For some benchmark tasks, we have shown an FN-synapse network shows better multi-task accuracy compared to other continual learning approaches. This leads to the possibility that the intrinsic dynamics of the FN-synapse could provide important clues on how to improve the accuracy of other continual learning models as well.

Fig. 2.5 (c)-(d) also show the importance of the learning algorithm in fully exploiting the available network capacity. While the entropy of the FN-synapse weights for the output layer is relatively high, the entropy of the weights of the input layer is still relatively low, implying most of the input layer weights remain unused. This is an artifact of *vanishing gradients* in a standard backpropagation-based neural network learning. Thus, it is possible that improved backpropagation algorithms [35, 119] might be able to mitigate this artifact and in the process enhance the capacity and the performance of the FN-synapse network. In Fig. S9 we show that FN-synapse-based neural network is able to maintain its performance even when the network size is increased. Thus, it is possible that the network becomes capable of learning

more complex tasks due to an increase in the overall plasticity of the network while ensuring considerably better retention than neural networks with traditional synapses.

In addition to being physically realizable, the FN-synapse implementation also allows interpolation between a steady-state consolidation model and the EWC consolidation models. This is important because it is widely accepted that the EWC model can potentially suffer from blackout catastrophe [69] as the learning network approaches its capacity. During this phase, the network becomes incapable of retrieving any previous memory as well as is unable to learn new ones [69]. Steady-state models such as the cascade consolidation models and SGD-based continuous learning models avoid this catastrophe by gracefully forgetting old memories. As shown in Fig. 2.5 (a), an FN-synapse network, through the use of a global modulation factor $m(t)$, is able to interpolate between the two models. In fact, the results in Fig. 2.5 (a) and (b), show that the number of patterns/memories retained in an FN-synapse network under modulation profile $m_2(t)$ at steady state is higher compared to that of a high-complexity cascade model for a network size of $N = 1000$ synapses. Even though we have not used the interpolation feature for benchmark experiments, we believe that this attribute is going to provide significant improvements for continuous learning of a large number of tasks.

The interpolation property of FN-synapse could mimic some attributes of *metaplasticity* observed in biological synapses and dendritic spines [87]. The role of metaplasticity, the second-order plasticity of a synapse which assigns a task-specific importance to every successive task being learned [75], is widely accepted as the fundamental component of neural processes key to memory and learning in the hippocampus [4, 3]. Since unregulated plasticity leads to runaway effects resulting in previously stored memories being impaired at saturation of synaptic strength [20], metaplasticity serves as a regulatory mechanism that dynamically links the history of neuronal activity with the current response [60]. The FN-synapse mimics the same regulatory mechanism through the decaying term $r(t)$ that takes into account the history of usage or neuronal activity to determine the plasticity of the synapse for future use as well as prevents runaway effects by making the synapses rigid at saturation.

The on-device memory consolidation in FN-synapse can not only minimize the energy requirements in continual learning tasks, additionally, the energy required for a single synaptic weight update is also lower than memristor-based synaptic updates for a fixed precision of update. This attribute has been validated in our previous works [91] where the update energy was estimated to be as low as 5f J increasing up to 2.5p J depending on the

status of the FN-synapse and the desired change in synaptic weights. Note that the energy required to change the synaptic weight is derived from the FN-tunneling current and not from the electrostatic energy used for charging the coupling capacitor. Thus, by designing more efficient charge-sharing techniques across the coupling capacitors the energy-efficiency of FN-synaptic updates can be significantly improved. Furthermore, when implemented on more advanced silicon process nodes, the capacitances could be scaled which can improve the energy-efficiency of FN-synapse by an order of magnitude. Compared to memristor-based synapses, the FN-synapse can also exhibit high endurance $10^6 - 10^7$ cycles without any deterioration. However, the key distinction lies in terms of the dynamic range of the stored weights. Generally, a single memristor has two distinct conductive states (corresponding to '0' or '1') which give each device a 1-bit resolution. When used in a crossbar array, highly-dense designs can reach densities up to $76.5nm^2$ per bit as reported by [100] where a 3-D memristor array was constructed using Perovskite quantum wires. The dynamic range or resolution of such designs is determined by the number of memristive devices that can be packed into the smallest feasible physical form factor. If we consider multi-level memristors instead, the resolution per memristor can reach up to 3-5 bits depending on the number of stable distinguishable conductive states [77, 127, 56]. In comparison, the dynamic range of the FN-synapse (a single device) is considerably higher as it is determined by the number of electrons stored on the floating-gates which in-turn is determined by the FN-synapse form-factor and the dielectric property of the tunneling barrier. Thus, theoretically, the dynamic range and the operational-life of the FN-synapse seem to be constrained by the single-electron quantization. However, at low-tunneling regimes, the transport of single electrons across the tunneling barrier becomes probabilistic where the probability of tunneling is now modulated by the external signals $X(t)$ and $m(t)$. In the Methods Section and in Fig. S4 we show that a stochastic dynamical system model emulating the single-electron dynamics in the FN-synapse can produce $\mathcal{O}(1/\sqrt{t})$ consolidation characteristics for the benchmark random input patterns experiment for an empty network. The SNR still follows the power-law curve and the FN-synapse network continues to learn new experiences even if the synaptic updates are based on discrete single-electron transport. A more pragmatic challenge in using the FN-synapse will be the ability of the read-out circuitry to discriminate between the changes in floating-gate voltage due to single-electron tunneling events. For the magnitude of the floating-gate capacitance, the change in voltage would be in the order of 100nV per tunneling event. A more realistic scenario would be to measure the change in voltage after 1000 electron tunneling events which would imply measuring 100 μV changes. Although

this will reduce the resolution of the stored weights/updates to 14 bits, recent studies have shown that neural networks with training precisions as low as 8 bits [115] and networks with inference precisions as low as 2-4 bits [30, 31] are often capable of exhibiting remarkably good learning abilities. In Fig. S10 we show that for the split-MNIST task, the performance of the FN-synapse based neural network remains robust even in the presence of 5% device mismatch.

Another point of discussion is whether the optimal decay profile $r(t) \approx \mathcal{O}(1/t)$ can be implemented by other synaptic devices, in particular, the energy-efficient memristor-based synapses that have been proposed for neuromorphic computing [89, 65, 122, 43, 97, 98]. Recent works using memristive devices have demonstrated on-device *metaplasticity* [48], however, achieving an optimal decay profile would require additional control circuitry, storage and read-out circuits. In this regard, we believe that the FN-synapse represents one of the few, if not the only class of synaptic devices that can achieve optimal memory consolidation on a single device.

Chapter 3

Adaptive Synaptic Array using FNDAM

This chapter is an extension of the work discussed in Chapter 2 which presents the application of FNDAM regarding the energy efficiency of an artificial neural network during the training phase. It shows the adaptive nature of FNDAM with respect to the loss function that the network is trying to minimize. This chapter has supplemental information which is included in Appendix B. The results in this chapter are based on [91].

3.1 Introduction

Implementation of reliable and scalable synaptic weights or memory remains an unresolved challenge in the design of energy-efficient machine learning (ML) and neuromorphic processors [23]. Ideally, the synaptic weights should be "analog" and should be implemented on a non-volatile, and yet easily modifiable storage device [130]. Furthermore, if these memory elements are integrated in proximity with the computing circuits or processing elements, then the resulting compute-in-memory (CIM) architecture [2, 128] has the potential to mitigate the "memory wall" [110, 61, 29] which refers to the energy-efficiency bottleneck in ML processors that arises due to repeated memory access. In most practical and scalable implementations, the processing elements are implemented using CMOS circuits; as a result, it is desirable that the analog synaptic weights be implemented using a CMOS-compatible technology. In literature, several multi-level non-volatile memory devices have been proposed for implementing analog synapses. These include two-terminal memristive devices such as resistive random-access memories (RRAM) [6], magnetic random-access memories (MRAM) [47], Phase Change Memory (PCM) [22], Spin-Transfer Torque Magnetic RAM (STT-MRAM) [68], Conductive Bridge RAM [63] or the three-terminal devices like the

floating-gate transistors [92], ferroelectric field-effect transistor-based memory (FeFET) [39], Charge Trap Memory [129] and Electrochemical RAMs (ECRAM) [120]. In all of these devices, the analog memory states are static in nature, where each of the states needs to be separated from others by an energy barrier ΔE. For example, in RRAM devices the state of the conductive filament between two electrodes determines the stored analog value, whereas in charge-based devices like floating-gates or FeFET, the state of polarization determines the analog value. To ensure non-volatile storage, it is critical that the energy-barrier ΔE is chosen to be large enough to prevent memory leakage due to thermal-fluctuations and other environmental disturbances. However, the height of the energy barrier ΔE also sets the fundamental limit on the energy dissipated to switch between different analog storage states. For example, switching the RRAM memory state requires 100 fJ per bit [131], whereas STT-MRAM requires about 4.5pJ per bit [38]. A learning/training algorithm that adapts the stored weights in quantized steps $(\dots, W_{n-1}, W_n, W_{n+1}, \dots)$ so as to minimize a system-level loss-function $L(W)$ has to dissipate minimum energy of $(\dots, \Delta E_{n-1}, \Delta En, \Delta E_{n+1}, \dots)$ for memory updates. Separating the static states by an energy-barrier also allows the learning algorithm to precisely control the parameter retention time (parameter leakage) between subsequent parameter updates, however, this mode of updates does not exploit the physics of learning to optimize for energy-efficiency. In many energy-efficient ML training formulations, and in particular analog ML systems, the loss-function $L(W)$ is represented by an equivalent energy-functional of a physical ML system [76], and learning/training involves a natural evolution of the system dynamics towards the minimum energy (optimal) state based on input stimuli (or equivalently training data). Thus, the physics of the system evolution process selects the minimum energy path toward the desired optimum. A synaptic element that is matched to this system dynamics needs to be adaptive with respect to its memory retention time which can then be traded-off with respect to the energy-dissipation per update.

In this work we present such a synaptic element that uses dynamical states (instead of static states) to implement analog memory and is matched to the dynamics of ML training. The core of the proposed device is itself a micro-dynamical system and the system-level learning/training process modulates the dynamical state (or state trajectory) of the memory ensembles. The concept is illustrated in Fig. 3.1(b), which shows a reference ensemble trajectory that continuously decays towards a reference zero vector without the presence of any external modulation. However, during the process of learning, the trajectory of the memory ensemble is pushed towards an optimal solution $W*$. The main premise of this work is that the extrinsic energy $(..., \Delta E_{n-1}, \Delta E_n \dots \Delta E_{n+1}, ...)$required for modulation, if

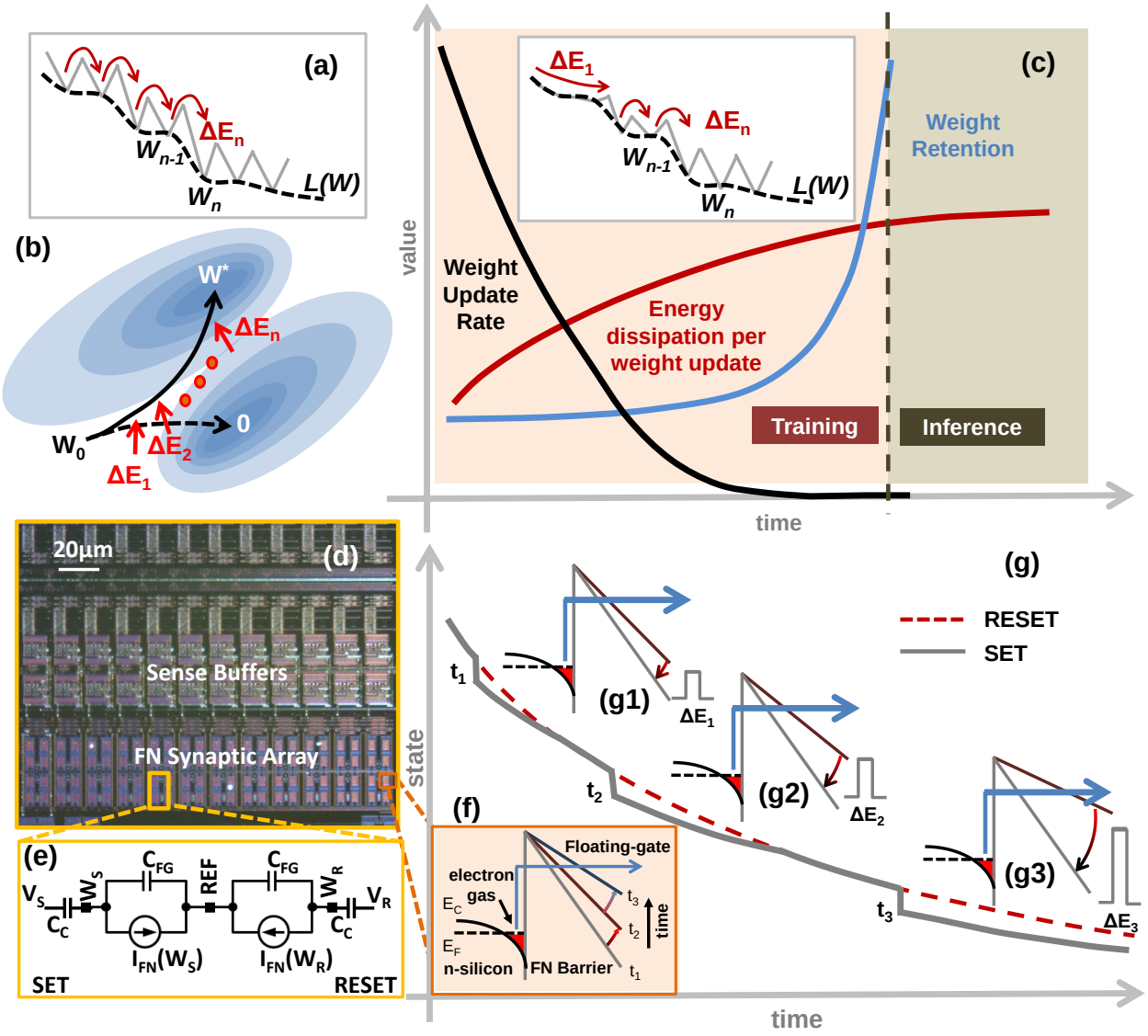

Figure 3.1: (a) conventional non-volatile analog memory where transition between analog static states (W_{n-1}, W_n) dissipates energy (ΔE_n); (b) Dynamic analog memory where an external energy $(\Delta E_1, \Delta E_2 \ldots \Delta E_n)$ is used to modulate the trajectory of the memory states $(W0)$ towards the optimal solution $(W`)$; (c) desired analog synapse characteristic where the memory retention rate is traded-off with the write energy; reducing the energy dissipation per weight update in training phase by matching the dynamics of the dynamic analog memory to the weight decay as shown in (inset) where the height of the energy barrier (ΔE_n) increases as the training progresses(d) micrograph of a fabricated DAM array along with (e) its equivalent circuit where the leakage current IFN is implemented by (f) the electron transport across a Fowler-Nordheim (FN) tunneling barrier; (g) Implementation of the FN tunneling based DAM where dynamic states g1-g3 determines the energy dissipated $(\Delta E_1, \Delta E_2 \ldots \Delta E_n)$ per memory update at time instance $t_1 - t_3$ and memory retention rate.

matched to the dynamics of learning, could reduce the energy-budget for ML training. This is illustrated in Fig. 3.1(c) which shows a convergence plot corresponding to a typical ML system as it transitions from a training phase to an inference phase. During the training phase, the synaptic weights are adapted based on some learning criterion whereas in the inference phase, the synaptic weights remain fixed or are adapted intermittently to account for changes in the operating conditions. As a result, during the training phase, the number of weight updates is significantly higher than during the inference phase. Take for example support-vector machine (SVM) training, the number of weight updates scale quadratically with the number of support vectors and the size of the training data, whereas adapting the SVM during inference only scales linearly with the number of support-vectors [102]. Thus, for a constant energy dissipation per update, the total energy-dissipated due to weight updates is significantly higher in training than during inference. However, if the energy-budget per weight update could follow a temporal profile as shown in Fig.3.1c, wherein the energy dissipation is no longer constant, but inversely proportional to the expected weight-update rate, then the total energy dissipated during training could be significantly reduced. One way to reduce the synaptic weight update or memory write energy budget is to trade-off the weight's retention rate according to the profile shown in Fig. 3.1c. The desired retention rate profile could then be achieved by adaptively changing the energy-barrier height as shown in Fig. 3.1c - inset. During the training phase, the synaptic element can tolerate lower retention rates or parameter leakage because this physical process could be matched to the process of weight decay or regularization, a technique commonly used in ML algorithms to achieve better generalization performance [83]. As shown in Fig. 3.1c, the synapse's retention rate should increase as the training progresses such that at convergence or in the inference phase the weights are stored as a non-volatile memory.

In this work, we describe a dynamic analog memory (DAM) that can exhibit a temporal profile similar to that of Fig. 3.1c. Furthermore, the memory is implemented on a standard CMOS process without the need for any additional processing layers. Fig. 3.1d shows a micrograph of a DAM array where each element of the array implements the circuit shown in Fig. 3.1e. In Appendix B I we provide additional details for implementing the circuit of Fig. 3.1e in a standard CMOS process. The proposed DAM requires a Fowler-Nordheim (FN) quantum-tunneling barrier which can be created by injecting sufficient electrons onto a polysilicon island (floating-gate) that is electrically isolated by thin silicon-di-oxide barriers [79]. As the electron tunnels through the triangular barrier, as shown in Fig. 3.1f, the barrier profile changes which further inhibits the tunneling of electrons. We have previously

shown that the dynamics of this simple system are robust enough to implement time-keeping devices [137] and self-powered sensors [90]. In this work, we use a pair of synchronized FN-dynamical systems to implement a DAM suitable for implementing ML training/inference engines. Figure 3.1(g) shows the dynamics of two FN-dynamical systems, labeled as SET and RESET, whose analog states continuously and synchronously decay with respect to time. In our previous work [137, 90], we have shown the dynamics across different FN-dynamical systems can be synchronized with respect to each other with an accuracy greater than 99.9%. However, when an external voltage pulse modulates the SET system, as shown in Fig. 3.1g, the dynamics of the SET system become desynchronized with respect to the RESET system. The degree of desynchronization is a function of the state of the memory at different time instances (Fig. 3.1g, insets g1-g3) which determines the memory's retention rate. For instance, at time-instant t_1, a small magnitude pulse would produce the same degree of desynchronization as a large magnitude pulse at the time-instant t_3. However, at t_1 the pair of desynchronized systems (SET and RESET) would resynchronize more rapidly as compared to desynchronized systems at time-instants t_2 or t_3. This resynchronization effect results in shorter data retention; however, this feature could be leveraged to implement weight-decay in ML training. At time-instant t_3, the resynchronization effect is weak enough that the FN-dynamical system acts as a persistent non-volatile memory with high data-retention time. In the Methods section, we describe how the FN-dynamical system mathematical model can be matched to ML training formulation and the weight-decay dynamics required for learning and generalization. The model also shows that the voltage or energy required for updating the memory can be annealed according to the profile shown in Fig. 3.1c.

3.2 Result

3.2.1 Dynamic analog memory with asymptotic non-volatile storage

The dynamics of the FN-tunneling based DAM (or FN-DAM) were verified using prototypes fabricated in a standard CMOS process (micrograph shown in Fig. 3.1d.). The FN-DAM devices were programmed and initialized through a combination of FN tunneling and hot electron injection. A detailed description of the general programming process can be found in

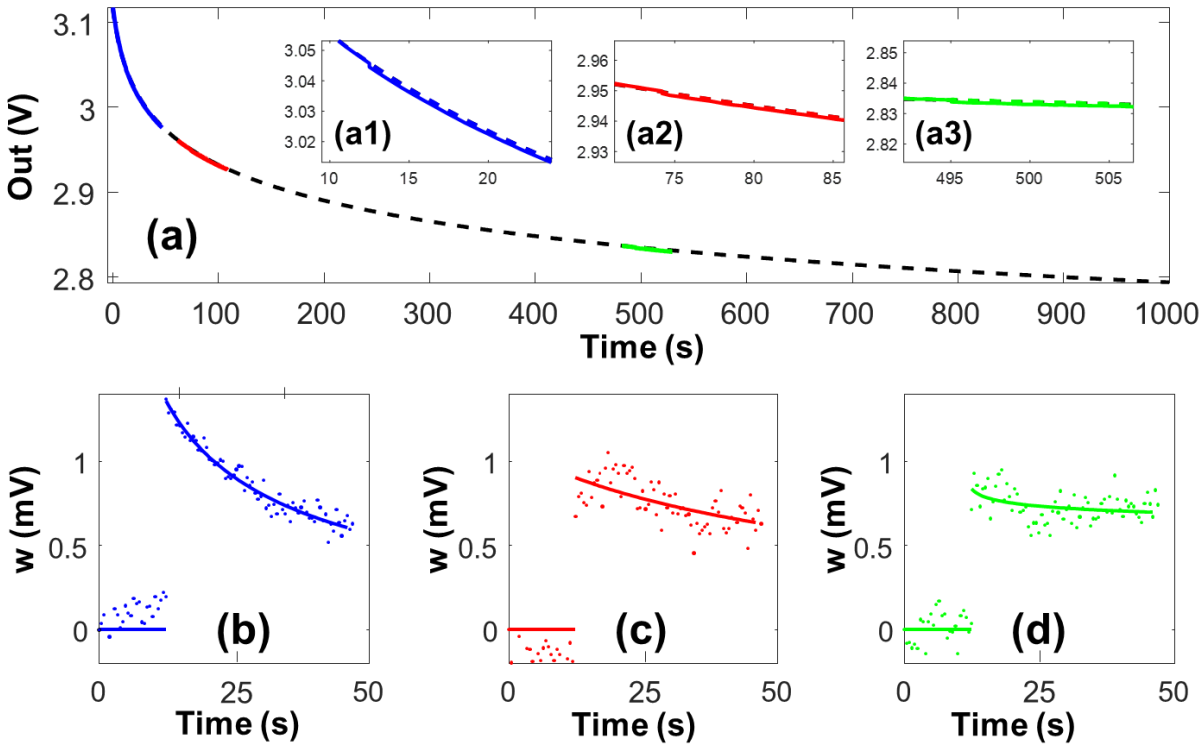

Figure 3.2: a) W_S (solid line) and W_R (dashed line) response under three different operating regimes (zoomed insets: a1, a2, a3) determined by FN-DAM initialization voltage. b)-d) FN-DAM response (w) calculated as the difference between W_S and W_R voltage values in the three regimes demonstrating different plasticity. Dots are measured data points while lines correspond to fit the data.

the Methods section where we describe implementation specific to this work. The tunneling nodes (W_S and W_R in Fig. 3.1e) were initialized to around 8 V and decoupled from the readout node by a decoupling capacitor to the sense buffers (shown in Appendix B Fig. B.1). The readout nodes were biased at a lower voltage (around 3 V) to prevent hot electron injection [118] onto the floating-gate during the readout operation. The capacitive decoupling of the read-out circuitry from the memory also reduces the effect of read disturbances and in Appendix B Fig. B.2, we show measurement results that verify that the effect of read disturb is random and the magnitude of the disturbance is less than the precision of the memory update and read-out circuits.

Fig. 3.2 shows the measured dynamics of the FN-DAM device in different initialization regimes used in ML training, as described in Fig. 3.1g. The different regimes were obtained by initializing the tunneling nodes (W_S and W_R) to different voltages (see Methods section),

whilst ensuring that the tunneling rates on the W_S and W_R nodes were equal. Initially (during the training phase), tunneling-node voltages were biased high (readout node voltage of 3.1 V), leading to faster FN tunneling (Fig. 3.2, inset a1). A square input pulse of 100 mV magnitude and 500 ms duration (5 fJ of input energy) was found to be sufficient to desynchronize the SET node by 1 mV. This desynchronization, $w = (W_S - W_R)$, stores the state of the dynamical analog memory. However, as shown in Fig. 3.2(b), the rate of resynchronization in this regime is high, which leads to a decay in the stored weight down to 30% in 40 s. At t = 90 s, the voltage at node W_S has reduced (readout node voltage of 2.9 V shown in Fig. 3.2, inset a2), and a larger voltage amplitude (500 mV) is required to achieve the same desynchronization magnitude of 1 mV. This corresponds to an energy expenditure of 125 fJ. However, as shown in Fig. 3.2(c), the rate of resynchronization is low in this regime, leading to a decay in the stored weight down to 70% of its value in 40 s. Similarly, at a later time instant t = 540 s (Fig. 3.2, inset a3), a 1 V signal desynchronizes the recorder by 1 mV, and as shown in Fig. 3.2(d), in this regime 95% of the stored weight value is retained after 40 s. This mode of operation is suitable during the inference phase of machine learning when the weights have already been trained, but the models need to be sporadically adapted to account for statistical drifts. Modeling studies described in Appendix B show that the write energy per update starts from as low as 5 fJ and increases to 2.5 pJ over a period of 12 days. During the same time, the memory becomes less plastic with the increase in the memory retention time as shown in Appendix B. Asymptotically, the FN-DAM exhibits retention times similar to that of other FLASH-based memory.

The next set of experiments verified if the analog state of an FN-DAM device can be adapted (incremented or decremented) using digital pulses (using a digital logic or a spiking neuron). Each of the differential DAM elements in the FN-DAM device was programmed by independently modulating the SET and RESET junctions shown in Fig. 3.1(e). The corresponding W_S and W_R nodes were initially synchronized with respect to each other. After a programming pulse was applied to the SET or RESET control gate, the difference between the voltages at the W_S and W_R nodes was measured using an array of sense buffers. In the results shown in Fig. 3.3a-d, a sequence of 100 ms-long 3V SET and RESET pulses was applied. The measured difference between the voltages at the W_S and W_R nodes indicates the current state of the memory. Each SET pulse increments the state while a RESET pulse decrements the state. In this way, the FN-device can implement a DAM that is bi-directionally programmable with unipolar pulses. Note that, unlike conventional FLASH memory, the magnitude of the programming pulse is significantly lower. Fig. 3.3d also shows

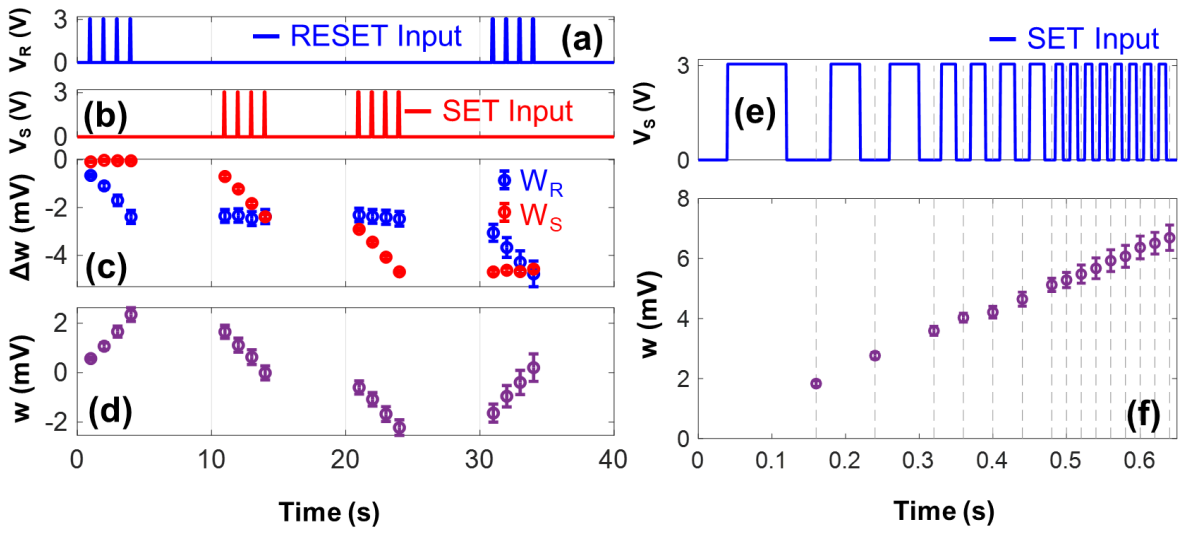

Figure 3.3: (a-b) SET and RESET input sequence. c) Change in W_S and W_R potentials due to SET and RESET pulses. d) DAM response calculated as the difference between W_S and W_R voltages. e-f) FN-DAM response to SET pulses of varying frequency. Error bars indicate standard deviation estimated across 12 devices.

the cumulative nature of the FN-DAM updates which implies that the device can work as an incremental/decremental counter.

Fig. 3.3e-f show measurement results that demonstrate the resolution at which an FN-DAM can be programmed as analog memory. The analog state can be updated by applying digital pulses of varying frequency and variable number of pulses. In Fig. 3.3e, four cases of applying a 3 V SET signal for a total of 100 ms are shown: a single 100 ms pulse; two 50 ms pulses; four 25 ms pulses; and eight 12.5 ms pulses. The results show the net change in the stored weight was consistent across the 4 cases. A higher frequency leads to finer control of analog memory updates. Note that any variations across the devices can be calibrated or mitigated by using an appropriate learning algorithm [25]. The variations could also be reduced by using careful layout techniques and precise timing of the control signals.

3.2.2 Characterization of FN-DAM

The FN-DAM device can be programmed by changing the magnitude of the SET/RESET pulse or its duration (equivalently number of pulses of fixed duration). Fig. 3.4a shows the

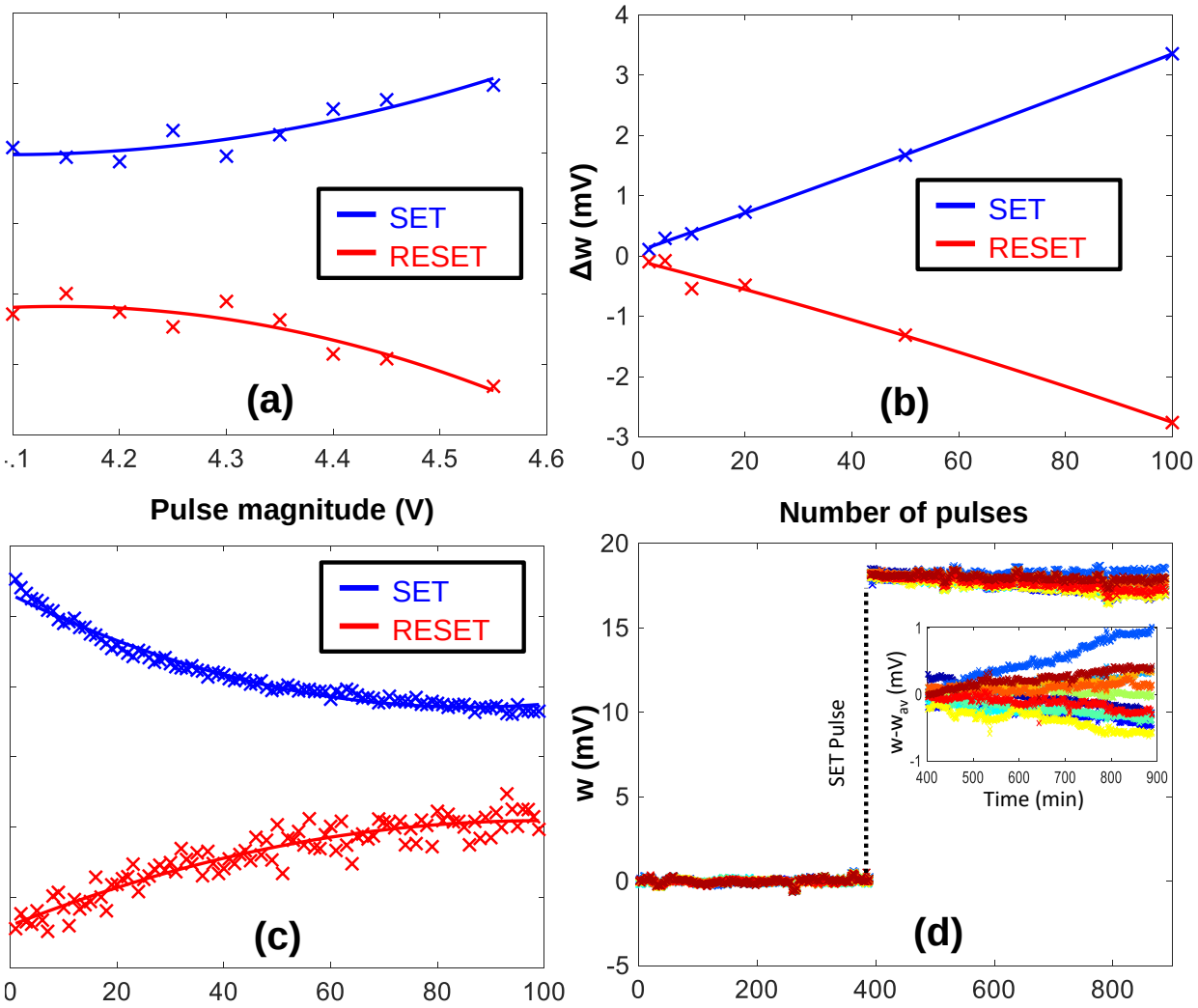

Figure 3.4: (a) DAM response to pulses of different magnitude but same duration (10 ms). b) DAM response to a varying number of pulses of 4V amplitude and 10 ms duration. c) Change in DAM response with each pulse of the same magnitude (4V) and duration (10 ms). d) FN-DAM response measured at 100oC when a SET pulse is applied to 12 different FN-DAM elements (each color corresponds to a different memory element).

response when the magnitude of the SET and RESET input signals varies from 4.1 V to 4.5 V. The measured response shown in Fig. 3.4a shows an exponential relationship with the amplitude of the signal. When short-duration (10 ms) pulses are used for programming, the stored value varies linearly with the number of pulses, as shown in Fig. 3.4b. However, repeated application of pulses with constant magnitude produces successively smaller changes in programmed value due to the dynamics of the DAM device (Fig. 3.4c). One way to achieve a constant response is to pre-compensate the SET/RESET control voltages such that

a target voltage difference $w = (W_S - W_R)$ can be realized. The differential architecture increases the device's state robustness against disruptions from thermal fluctuations (Fig. 3.4d). The stored value on DAM devices will leak due to thermal-induced processes or due to trap-assisted tunneling. However, in DAM, the weight is stored as the difference in the voltages corresponding to W_S and W_R tunneling junctions which are similarly affected by temperature fluctuations. This is shown in Fig. 3.4d where the FN-DAM array was programmed/operated at 100 C and the dynamic response was measured over a duration of 15 hours. The baseline drift due to the memory read-out circuits was first calibrated during the first 400 minutes and used for zeroing out the dynamical response of each of the FN-DAM devices. Then, at 400 min time instant a SET pulse (3.3V for 1-second duration) was applied to all the FN-DAM devices which programmed all the devices to a specific memory state. The degree of desynchronization was continuously measured and is plotted in Fig. 3.4d. Over a duration of 8 hours, the drift in the stored analog value is less than 10%. This result could also be used to estimate the memory retention time as described in Appendix B, which is expected to vary depending on the current state of the memory.

3.2.3 FN-DAM based Co-design of Classifiers and Neural Networks

We first experimentally demonstrate the benefits of FN-DAM based weights when training a simple linear classifier. For these results, two FN-DAM devices were independently programmed according to the perceptron training rule [18]. We trained the weights of a perceptron model to classify a linearly separable dataset comprising 50 instances of two-dimensional vectors, as shown in Fig. 3.5a. During each epoch, the network loss function and gradients were evaluated for every training point in a randomized order, with the time interval between successive training points being two seconds. Fig. 3.5b shows that after training for 5 epochs, the learned boundary can correctly classify the given data. Fig. 3.5c shows the evolution of weights as a function of time. As can be noted in the figure, initially the magnitude of weight updates (negative of the cost function gradient) was high for the first 50 seconds, after which the weights stabilized and required smaller updates. The energy consumption of the training algorithm can be estimated based on the magnitude and number of the SET/RESET pulses required to carry out the required update for each misclassified point. As the SET/RESET nodes evolve in time, they require larger voltages for carrying out updates, as shown in Fig. 3.5d. The gradient magnitude was mapped onto an equivalent

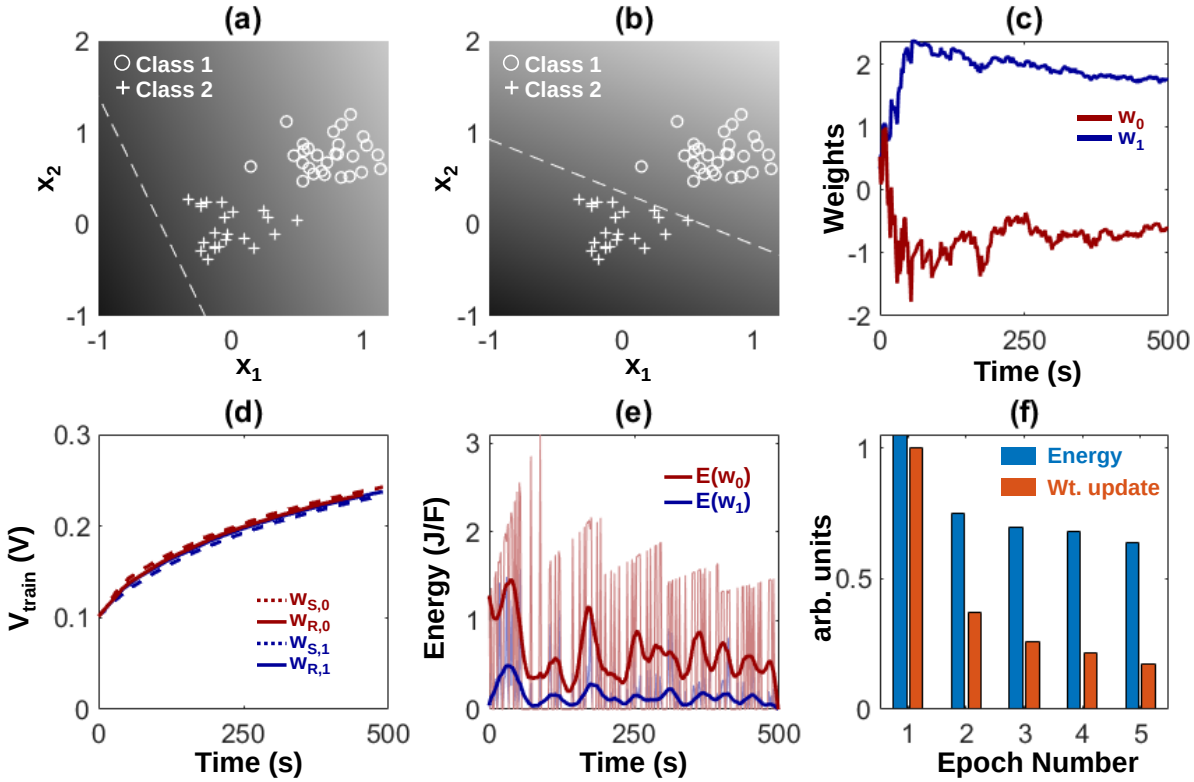

Figure 3.5: a) Test data set with randomly initialized decision boundary b) Decision boundary after training. c) Evolution of weights (w_0 and w_1) after 5 epochs. D) Input voltage required for initiating a unit change in weights ($W_{s,0}, W_{R,0}, W_{s,1}, W_{R,1}$). e) Energy ($E(w_0), E(w_1)$) expended in updating the weights (w_0 and w_1). f) Average magnitude of weight update and average energy required for each epoch.

number of 1 kHz pulses, rounding to the nearest integer. Fig. 3.5e shows the energy (per unit capacitance) required to carry out the weight update whenever a point was misclassified. Though the total magnitude of weight update decreased with each epoch, the energy required to carry out the updates had lower variation (Fig. 3.5f). The relatively larger energy required for smaller weight updates at later epochs led to longer retention times of the weights.

Similar energy-dissipation and weight update profiles were also obtained when a larger FN-DAM array is used to store the training parameters of a three-layer neural network implementing a multi-layer perceptron (MLP). Details of the network architecture and training procedure are described in the Methods section and in Appendix B. Fig. 3.6(a–c) show the FN-DAM training dynamics when the MLP neural network is trained on the Fisher Iris dataset [42]. In particular, Fig. 3.6c shows that by adapting the programming pulses, the

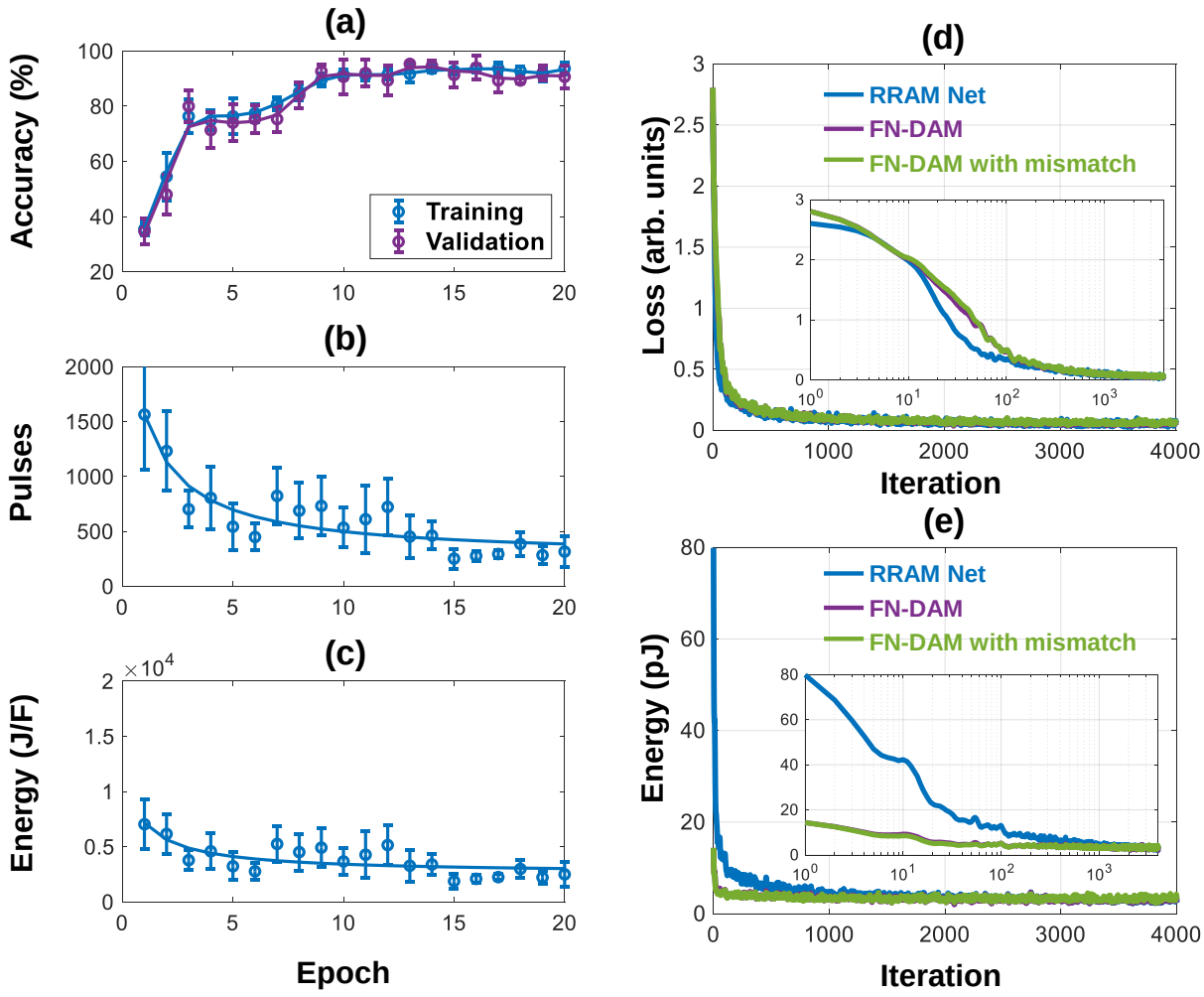

Figure 3.6: : a-c) Experimental training on Fisher Iris dataset over 5 trials: a) 5-fold cross-validation accuracy of model over 20 epochs for training set (120 points) and validation set (30 points) b) Total pulses required in implementing weight update for entire synaptic array during each epoch. c) Energy per unit capacitance expended in updating the weights. Note: the scale of the Y-axis is set to match that of the panel (b). Error bars in (a), (b), and (c) indicate standard deviation estimated across 5 trials. d-e) Simulated training on MNIST dataset: d) Network loss for 3 types of network models. Inset shows the same data with the x-axis in the log scale. e) Energy dissipated in updating the network weights for 3 types of network models. Inset shows the same data with the X-axis in the log scale.

energy-dissipation profile across training and inference can be equalized, as was proposed in Fig. 3.1c. The dynamical systems model summarized in the Methods section can also be used to evaluate the energy-efficiency gains that can be obtained by co-designing a convolutional neural network (CNN) training engine using FN-DAMs. Details of the CNN architecture

are provided in Appendix B. The result of the co-design is shown in Fig. 3.6d and in Table 1 where we show that an FN-DAM based deep neural network (DNN) can achieve similar classification accuracy as a conventional DNN. To compare the energy dissipation of the FN-DAM neural network implementation, we used an RRAM energy per bit dissipation metric (100 fJ/bit) [131] and for an FN-DAM implementation we used an energy dissipation model described in Appendix B. Note that amongst CMOS-compatible non-volatile analog memories, RRAM is one of the most energy-efficient synapses. The result in Fig. 3.6e shows that training an FN-DAM based neural network dissipates significantly lower energy compared to the RRAM-based neural network. Note that for this demonstration, only the fully connected layers were trained while the feature layers were kept static. This mode of training is common for many practical DNN implementations on edge computing platforms where the goal is not only to improve the energy-efficiency of inference but also for training [117]. The results in Fig. 3.6e and Table 1 show that the neural network training and accuracy are robust even when a mismatch is introduced into the FN-DAM model. Details of the mismatch model are provided in the Methods section.

table here

3.3 Discussions

In this work, we reported a Fowler-Nordheim quantum tunneling-based dynamic analog memory (FN-DAM) whose physical dynamics can be matched to the dynamics of weight updates used in ML or neural network training. During the training phase, the weights stored on FN-DAM are plastic in nature and decay according to a learning-rate evolution that is necessary for the convergence of the gradient-descent training [94]. As the training phase transitions to an inference phase, the FN-DAM acts as a non-volatile memory. As a result, the trained weights are persistently stored without requiring any additional refresh steps (used in volatile embedded DRAM architectures [121]). The plasticity of FN-DAM during the training phase can be traded off with the energy-required to update the weights. This is important because the number of weight updates during training scales quadratically with the number of parameters, hence the energy-budget during training is significantly higher than the energy-budget for inference. The dynamics of FN-DAM bear similarity to the process of annealing used in neural network training and other stochastic optimization engines to

overcome local minima artifacts [27]. Thus, it is possible that FN-DAM implementations or ML processors can naturally implement annealing without dissipating any additional energy. If such dynamics were to be emulated on other analog memories, it would require additional hardware and control circuitry.

Several challenges exist in scaling the FN-DAM to large neural-networks. Training a large-scale neural network could take days to months [112] depending on the complexity of the problem, the complexity of the network, and the size of the training data. This implies that the FN-DAM dynamics need to match the long training durations as well. Fortunately, the 1/log characteristic of FN devices ensures that the dynamics could last for durations greater than a year [138]. The other challenge that might limit the scaling of FN-DAM to large neural networks is the measurement precision. The resolution of the measurement and the read-out circuits limit the energy-dissipated during memory access and how fast the gradients can be computed (Appendix B Fig. B.5). For instance, a 1 pF floating-gate capacitance can be initialized to store 107 electrons. Even if one were able to measure the change in synaptic weights for every electron tunneling event, the read-out circuits would need to discriminate 100 nV changes. A more realistic scenario would be measuring the change in voltage after 1000 electron tunneling events which would imply measuring 100 µV changes. However, this will reduce the resolution of the stored weights/updates to 14 bits. This resolution might be sufficient for training a medium-sized neural network; however, it is still an open question if this resolution would be sufficient for training large-scale networks [28, 54]. A mechanism to improve the dynamic range and the measurement resolution is to use a current-mode readout integrated with current-mode neural network architecture. If the read-out transistor is biased in weak-inversion, 120 dB of dynamic range could be potentially achieved. However, note that even in this operating mode, the resolution of the weight would still be limited by the number of electrons and the quantization due to electron transport. Addressing this limitation would be a part of future research.

If the proposed FN-DAM were to be used as a static analog memory, then measuring 1mV differences to distinguish between different memory states would be challenging, especially if device mismatch were to be taken into account. However, the analog value stored on the FN-DAM array is updated within a learning loop that minimizes a system-level objective function (cumulative loss or distance). Thus, the effect of any static mismatch across the memory cells gets calibrated during the process of training. The important aspect for the calibration process to be successful is that the memory update be monotonic with respect

to error-gradient and the precision of the updates be high enough (typically greater than 12 bits). Both of these requirements are met by FN-DAM due to the physics of electron tunneling. In fact, the effect of calibration due to learning can be seen in the FN-DAM neural network training (Fig. 3.6d, e) where the classification accuracy is independent of the initial choice of the FN-DAM state and the mismatch in FN-DAM device characteristics. The effect of blurring, due to the presence of thermal noise is in fact beneficial for training the neural network since it helps in overcoming artifacts due to local minima. Once the FN-DAM has transitioned to a non-volatile state (during inference), the effect of blurring is significantly reduced as the energy barrier separating different analog states is significantly higher than energy due to thermal fluctuations. However, the effect of blurring due to measurement noise needs to be compensated by averaging or increasing the cumulative measurement time.

In this work, we have used a voltage buffer (source follower) to read the state of the FN-DAM cell. However, a current mode readout could also be used which to differentiate mV changes in FN-DAM voltages. In particular, if the read-out transistor is biased in weak-inversion, then the exponential dependence between the gate voltage and the drain current could be used to amplify the change in voltage. We have previously used this method in [84] for floating-gate current memory arrays and in [74] we reported an active feedback-based approach to improve the resolution of the voltage-mode read-out. However, in both these implementations there will exist a trade-off between the accuracy of the measurement resolution and the read-out speed.

Another limitation that arises due to a finite number of electrons stored on the floating-gate and transported across the tunneling barrier during SET and RESET, is the speed of programming. Shorter duration programming pulses would reduce the change in stored voltage (weight) which could be beneficial if precision in updates is desired. In contrast, by increasing the magnitude of the programming pulses, as shown in Fig. 3.4(a), the change in stored voltage can be coarsely adjusted. However, this would limit the number of updates before the weights saturate. Note that due to device mismatch, the programmed values would be different on different FN-DAM devices.

In terms of endurance, after a single initialization, the FN-DAM can support $10^3 - 10^4$ update cycles before the weight saturates. However, at the core FN-DAM is a FLASH technology and could potentially be reinitialized again. Given that the endurance of FLASH memory is 10^3 [6], it is anticipated that FN-DAM to have an endurance of $10^6 - 10^7$ cycles. In terms

of other memory performance metrics, the I_{ON}/I_{OFF} ratio for the FN-DAM is determined by the operating regime and the read-out mechanism. Appendix B Fig. B.5 shows the expected ratio estimated using the FN-DAM model. Also, FN-DAM when biased as a non-volatile memory requires on-chip charge-pumps only to generate high-voltage programming pulses for infrequent global erase; thus, compared to FLASH memory, FN-DAM should have fewer failure modes [47]. Since FN-DAM can also be implemented on conventional FLASH memories, the synapses could be scaled to future 3-D and 2.5D FLASH processes where high synaptic densities can be achieved for the implementation of large-scale neural networks.

The main advantage of FN-DAM compared to other emerging memory technologies is its scalability and compatibility with CMOS. At its core, FN-DAM is based on floating-gate memories which have been extensively studied in the context of machine learning architectures [92]. Furthermore, from an equivalent circuit point of view, FN-DAM could be viewed as a capacitor whose charge can be precisely programmed using CMOS processing elements. Due to its unique decay characteristics, FN-DAM also provides a balance between weight-updates that are not too small so that learning never occurs versus weight-updates being too large such that the learning becomes unstable. The physics of FN-DAM ensures that weight decay (in the absence of any updates) towards a zero vector (due to resynchronization) which is important for neural network generalization [96]. For implementing a large-scale neural network, the FN-DAM form-factor would be required to be reduced which would affect device variability and mismatch. However, in our prior work [137, 90] we have shown that the dynamics of the FN-DAM cell (in steady-state) is determined primarily by the gate-oxide thickness, a parameter that is very well controlled across processes. An oxide thickness greater than 10nm ensures that the electron-leakage mechanism is dominated by FN quantum tunneling (instead of direct quantum tunneling). Thus, FN-DAM devices should be implementable on most sub-10nm CMOS processes that allow the fabrication of thicker gate-oxide transistors for input/output devices.

Like other analog non-volatile memories, FN-DAM could be used in any previously proposed compute-in-memory (CIM) architectures. However, in conventional CIM implementations, the weights are trained offline and then downloaded on the chip without retraining the processor [23]. This makes the architecture prone to analog artifacts like offsets, mismatches, and non-linearities. On-chip learning and training mitigate this problem whereby the weights self-calibrate for the artifacts to produce the desired output [130]. However, to support on-chip training/learning, weights need to be updated at a precision greater than 12 bits [54].

In this regard, FN-DAM exhibits a significant advantage compared to other analog memories. Even though in this proof-of-concept work, we have used a hybrid chip-in-the-loop training paradigm, it is anticipated that in the future the training circuits and FN-DAM modules could be integrated together on-chip.

From a neuromorphic point of view, FN-DAMs could be used to mimic network-level synaptic adaptation or pruning which plays a pivotal role in determining the optimal network configuration during the process of learning. For instance, it has been reported that a child's brain has significantly denser connectivity than an adult brain [2] and consumes 50% of the body's resting energy metabolism (BMR). Years of learning and synaptic pruning produce a network that tends towards optimality in terms of both energy and performance in adulthood when the brain accounts for only 20% of the BMR [2]. The adaptability of the proposed FN-DAM could be used to mimic this effect in artificial machine-learning systems.

If the FN-DAM updates are driven by constant voltage pulses (or fixed energy pulses like spikes) then the memory could be used to emulate the aging effects in synaptic plasticity that are observed in neurobiological systems [128]. Like biological synapses, the relative change in the value stored on FN-DAM or synaptic efficacy reduces with time for the same magnitude of applied input voltage pulses (or stimuli) [21]. Exploiting this feature of the FN-DAM to mimic neurobiologically relevant synaptic dynamics in artificial neural networks was the concept of chapter 2.

3.4 Methods

3.4.1 Initialization of the FN-DAM array

For each node of each recorder, the readout voltage was programmed to around 3 V while the tunneling node was operating in the tunneling regime (Appendix B Fig. B.1). This was achieved through a combination of tunneling and injection. Specifically, VDD was set to 7 V, input to 5 V, and the program tunneling pin was gradually increased to 23 V. Around 12–13V the tunneling node's potential would start increasing. The coupled readout node's potential would also increase. When the readout potential went over 4.5 V, electrons would start injecting into the readout floating gate, thus ensuring its potential was clamped below

5 V. After this initial programming, VDD was set to 6 V for the rest of the experiments. See Appendix B for further details. After one-time programming, the input was set to 0 V, the input tunneling voltage was set to 21.5 V for 1 minute and then the floating gate was allowed to discharge naturally. Readout voltages for the SET and RESET nodes were measured every 500 milliseconds. The rate of discharge for each node was calculated, and a state where the tunneling rates would be equal was chosen as the initial synchronization point for the remainder of the experiments.

3.4.2 FN Tunneling dynamics

The Fowler-Nordheim tunneling current is a function of the floating-gate capacitance C_T and the floating-gate voltage $V(t)$ and is given by:

$$I_{FN}\left(V\left(t\right)\right) = C_T \frac{d\left(V\left(t\right)\right)}{dt} = C_T \left(\frac{k_1}{k_2}\right) V^2 \exp\left(-\frac{k_2}{V}\right) \tag{3.1}$$

where k_1 and k_2 are device specific parameters. Solving 3.1 leads to the floating-gate voltage $V(t)$ as [137, 90]

$$V\left(t\right) = \frac{k_2}{\log\left(k_1 t + k_0\right)} \tag{3.2}$$

where k_0 depends on the initial condition as:

$$k_0 = \exp\left(-\frac{k_2}{V_0}\right) \tag{3.3}$$

3.4.3 Weight decay model and FN-DAM dynamics

Many neural network training algorithms are based on solving an optimization problem of the form [18]:

$$\min_{\bar{w}} H(w) = \frac{\alpha}{2}\left|\bar{w}\right| + \mathcal{L}(\bar{w}) \tag{3.4}$$

where $\bar{w}$ denotes the network synaptic weights, $\mathcal{L}(\bullet)$ is a loss-function based on the training set and α is a hyper-parameter that controls the effect of the $\mathcal{L}_2$ regularization. Applying

gradient descent updates on each element w_i of the weight vector $\bar{w}$ as:

$$w_{i,n+1} - w_{i,n} = -\alpha\eta_n w_{i,n} - \eta_n \frac{\delta \mathcal{L}(\bar{w})}{\delta w_{i,n}} \tag{3.5}$$

Where the learning rate η_n is chosen to vary according to $\eta_n \approx O(1/n)$ to ensure convergence to a local minimum [94].

The naturally implemented weight decay dynamics in FN-DAM devices can be modeled by applying Kirchhoff's Current Law at the SET and RESET floating gate nodes (see Fig. 3.1e).

$$C_T \frac{d}{dt}(W_S) + I_{FN}(W_S) = C_C \frac{d}{dt}(V_{SET}) \tag{3.6}$$

$$C_T \frac{d}{dt}(W_R) + I_{FN}(W_R) = C_C \frac{d}{dt}(V_{RESET}) \tag{3.7}$$

Where $C_{FG} + C_C = C_T$ is the total capacitance at the floating gate. Taking the difference between the above two equations, we get:

$$C_T \frac{d}{dt}(W_S - W_R) + I_{FN}(W_S) - I_{FN}(W_R) = C_C \frac{d}{dt}(V_{SET} - V_{RESET}) \tag{3.8}$$

For the differential architecture, $w = W_S - W_R$. Let $V_{train} = V_{SET} - V_{RESET}$, the training voltage calculated by the training algorithm. In addition, I_{FN} is substituted from Eqn. 3.1. Let $C_C/C_T = C_R$, the input coupling ratio:

$$\frac{dw}{dt} = -\frac{(I_{FN}(W_S) - I_{FN}(W_R))}{C_T} + C_R \frac{d}{dt}(V_{train}) \tag{3.9}$$

$$\frac{dw}{dt} = \frac{-\left(\frac{k_1}{k_2}\right) W_R^2 \exp\left(-\frac{k_2}{W_R}\right) + \left(\frac{k_1}{k_2}\right) W_S^2 \exp\left(-\frac{k_2}{W_S}\right)}{W_R - W_S} w + C_R \frac{d}{dt}(V_{train}) \tag{3.10}$$

Discretizing the update for a small time-interval Δt

$$w_{n+1} = w_n + \frac{-\left(\frac{k_1}{k_2}\right) W_R^2 \exp\left(-\frac{k_2}{W_R}\right) + \left(\frac{k_1}{k_2}\right) W_S^2 \exp\left(-\frac{k_2}{W_S}\right)}{W_R - W_S} w_n \Delta t + C_R \Delta V_{train,n} \tag{3.11}$$

Let $\mu = W_R/W_S$

$$w_{n+1} = w_n - \left(\frac{k_1}{k_2}\right) W_S \exp\left(-\frac{k_2}{W_S}\right) \frac{\mu^2 \exp\left(-\frac{k_2}{W_S}\left(1 - \frac{1}{\mu}\right)\right) - 1}{\mu - 1} w_n \Delta t + C_R \Delta V_{train,n} \tag{3.12}$$

Assuming that the stored weight (measured in mV) is much smaller than node potential (¿ 6V) i.e., $w \ll W_R (and W_R \approx W_S)$ and taking the limit ($\mu \to 1$) using L'Hôpital's rule:

$$w_{n+1} = \left(1 - \left(\frac{k_1}{k_2}\right)(2W_S + k_2)\exp\left(-\frac{k_2}{W_S}\right)\Delta t\right) w_n + C_R \Delta V_{train,n} \tag{3.13}$$

W_S follows the temporal dynamics given in Eqn. 3.2,

$$w_{n+1} = \left(1 - k_1\left(\frac{2}{\log\left(k_1 n \Delta t + k_0\right)} + 1\right)\left(\frac{1}{k_1 n \Delta t + k_0}\right)\right) w_n \Delta t + C_R \Delta V_{train,n} \tag{3.14}$$

Comparing the above equation to Eqn. 3.5, the weight decay factor for the FN-DAM system is given as:

$$\alpha\eta_n = k_1\left(\frac{2}{\log\left(k_1 n \Delta t + k_0\right)} + 1\right)\left(\frac{1}{k_1 n \Delta t + k_0}\right) \to O\left(\frac{1}{n}\right) \tag{3.15}$$

Note that the assumption $w << W_{S/R}$ in equation (10) makes the mathematical model of the synapse more tractable but is not a requirement for the memory to function. The caveat in relaxing the $w << W_{S/R}$ requirement is that the weight decay factor will not scale as 1/n during the initial phases of training. However, as shown in Fig. 3.6(d)-(e) for MNIST training, the learning process is able to compensate for this deviation.

3.4.4 Chip-in-the-loop linear classifier training

A hybrid hardware-software system was implemented to carry out an online machine-learning task. The physical weights ($\bar{w} = [w_1, w_2]$) stored in two FN-DAM devices were measured and used to classify points from a labeled test data set in software. We sought to train a linear decision boundary of the form:

$$f\left(\bar{x}, \bar{w}\right) = x_2 + w_1 x_1 + w_0 \tag{3.16}$$

$\bar{x} = [x_1, x_2]$ are the features of the training set. For each point that was misclassified, the error in the classification was calculated and a gradient of the loss function with respect to the weights was calculated. Based on the gradient information, the weights were updated in hardware by application of SET and RESET pulses via a function generator.

The states of the SET and RESET nodes were measured every 2 seconds and the weight of each memory cell, i, was calculated as:

$$w_i = 1000 \times (W_{R,i} - W_{S,i}) \tag{3.17}$$

The factor of 1000 indicates that the weight is stored as the potential difference between the SET and RESET nodes as measured in mV. We followed a stochastic gradient descent method. We defined the loss function as:

$$\mathcal{L}_n(\bar{w}) = ReLU\left(1 - y_n f\left(\bar{x_n}, \bar{w}\right)\right) \tag{3.18}$$

The gradient of the loss function was calculated as:

$$G_n(\bar{w}) = \frac{\partial \mathcal{L}_n(\bar{w})}{\partial \bar{w}} \tag{3.19}$$

The weights needed to be updated as

$$w_{n+1} = w_n - \lambda_n G_n(\bar{w}) \tag{3.20}$$

Here λ_n is the learning rate as set by the learning algorithm. The gradient information is used to update FN-DAM by applying control pulses to SET/RESET nodes via a suitable mapping function T:

$$V_{train,n} = T\left(\lambda_n G_n(\bar{w})\right) \tag{3.21}$$

Positive weight updates were carried out by application of SET pulses and negative updates via RESET pulses. The magnitude of the update was implemented by modulating the number of input pulses.

3.4.5 Memory Retention Model

In FN-DAM the parameter is stored as the difference (w) between the dynamical SET (W_S) and RESET (W_R) nodes as $w = W_s - W_R$. Due to resynchronization between the dynamical nodes, there is a finite time before memory can be read. Moreover, the rate of resynchronization is a function of the state of the nodes (Fig. 3.2), therefore, w can have a range of around 1V. Assuming 8-bit storage precision to be sufficient for machine learning

applications, each memory state is separated from each other by 4 mV. The retention time corresponds to the time it takes for w to reduce from 8 mV to 4 mV. This time is determined by the time-evolution of the FN-DAM voltage $V(V_0, t)$ which is determined by the parameter array $K = [k_1,\ k_2]$, and a potential V_0 that determines the region of operation (Fig. 3.2). Based on equation (3.1) $V(V_0, t)$ is given by:

$$V(V_0, t) = \frac{k_2}{\log\left(k_1 t + \exp\left(\frac{k_2}{V_0}\right)\right)} \tag{3.22}$$

The parameter array K is estimated from experiments that were carried out at room temperature. Retention time T_{ret} is calculated by solving the following equation:

$$V\left(W_R + 0.008, T_{ret}\right) - V\left(W_R, T_{ret}\right) = 0.004 \tag{3.23}$$

where W_R is varied from 5.5 to 7 V, to simulate different operating regimes. These simulation results are shown in Appendix B Figure. 3.4a. The retention times could then be estimated at different operating temperatures by using the Arrhenius equation to estimate k_1 as a function of temperature as

$$k_1\left(T\right) = k_1\left(T_0\right) \exp\left(-\frac{E_a}{k}\left(\frac{1}{T} - \frac{1}{T_0}\right)\right) \tag{3.24}$$

For instance, in our retention estimation shown in Appendix B Figure B.4a, we assumed activation energy $E_a = 0.6\ eV$, k is the Boltzmann's constant (8.617eV/K) and $T_0 = 25^oC$. Also, Appendix B Figure B.4b. shows that the retention model given by equations (3.22) and (3.23) matches the measured results at 100^oC.

3.4.6 Chip-in-the-loop MLP training on Fisher-Iris dataset

A larger FN-DAM array chipset was used for chip-in-the-loop neural network training on the Fisher Iris dataset. The chipset contains 128 individually selectable and programmable tunneling devices. By utilizing them in pairs, 64 synaptic weights could be implemented. Of the 64 FN-DAM elements, 51 elements were used to store the training weights of a three-layer neural network model (5 units corresponding to 4 input features and one for the bias term in the input layer, 7 units (including 1 bias unit) in the hidden layer and 3 units in the output

layer). A 5-fold cross-validation analysis was performed by splitting the Iris dataset into sets of 30 points each. Over 5 training sessions, each of the 5 sets was used to validate the model trained on the remaining 4 sets (120 points). The training was conducted over 20 epochs. In each epoch, the training set was randomly shuffled, and a batch size of 10 points was selected. During each batch update, weight updates were calculated through backpropagation with stochastic gradient descent. These weight updates were carried out in hardware through the application of SET/RESET pulses to the corresponding memory cell. Retention of the trained MLP parameters was verified using bake experiments as described in Appendix B and summarized in Fig. B.9 and B.10.

3.4.7 FN-DAM based CNN Implementation

The performance of FN-DAM model was compared to that of a standard network model. A 15-layer convolutional neural network was trained on the MNIST dataset using the MATLAB Deep Learning Toolbox. For each learnable parameter in the CNN, a software FN-DAM instance corresponding to that parameter was created. In each iteration, the loss of the network function and gradients were calculated. The gradients were used to update the weights via the Stochastic Gradient Descent with Momentum (SGDM) algorithm. The updated weights were mapped onto the FN-DAM array. The weights in the FN-DAM array were decayed according to Eqn. 3.20. These weights were then mapped back into the CNN. This learning process was carried on for 9 epochs. In the 10th epoch, no gradient updates were performed. However, the weights were allowed to decay for the last epoch (note that in the standard CNN case, the memory was static). A special case with a 0.1% randomly assigned mismatch in the floating gate parameters (k_1 and k_2) was also implemented.

Chapter 4

Cryptographic Key exchange based on FN-dynamical system

This chapter presents the application of the FN-dynamical system in the cryptographic security domain. The FN-dynamical system was previously used as a time-keeping device and also time itself has been used in this work as a one-way function for the key exchange protocol. As such, from this point onward the memory elements will be referred to as FN-timers. This work exploits both the dynamic nature of the information stored as well as the fact that the dynamical system is self-powered to design a key exchange framework that will be secure even with the advent of a quantum computer (proved in the following sections). The results in this chapter are based on [106].

4.1 Introduction

Securing information exchange with internet-of-things (IoTs) is becoming ever more important due to the proliferation of these platforms in domains ranging from infrastructure-IoTs [11] to medical-IoTs [12]. In one study [1] it is claimed that around 98% of the IoT data traffic is unencrypted and hence vulnerable to a data breach. Conventional data encryption techniques like RSA are too computationally prohibitive to be universally implemented on these low-resource platforms and reducing the computational complexity makes the approach vulnerable to quantum attacks. For instance, it is estimated in the literature that a quantum computer with 8194 logical qubits using Shor's Algorithm would be able to break the Rivest-Shamir-Adleman(RSA)[108] system with a key size of 4096 bits in 229 hours while for Discrete log problem with a key size of 521 bits, it would take 55 hours for a quantum computer with 4719 logical qubits, again using the Shor's Algorithm[95]. Symmetric key algorithms like Advanced Encryption Standard (AES-256) can be customized for IoT platforms and are considered

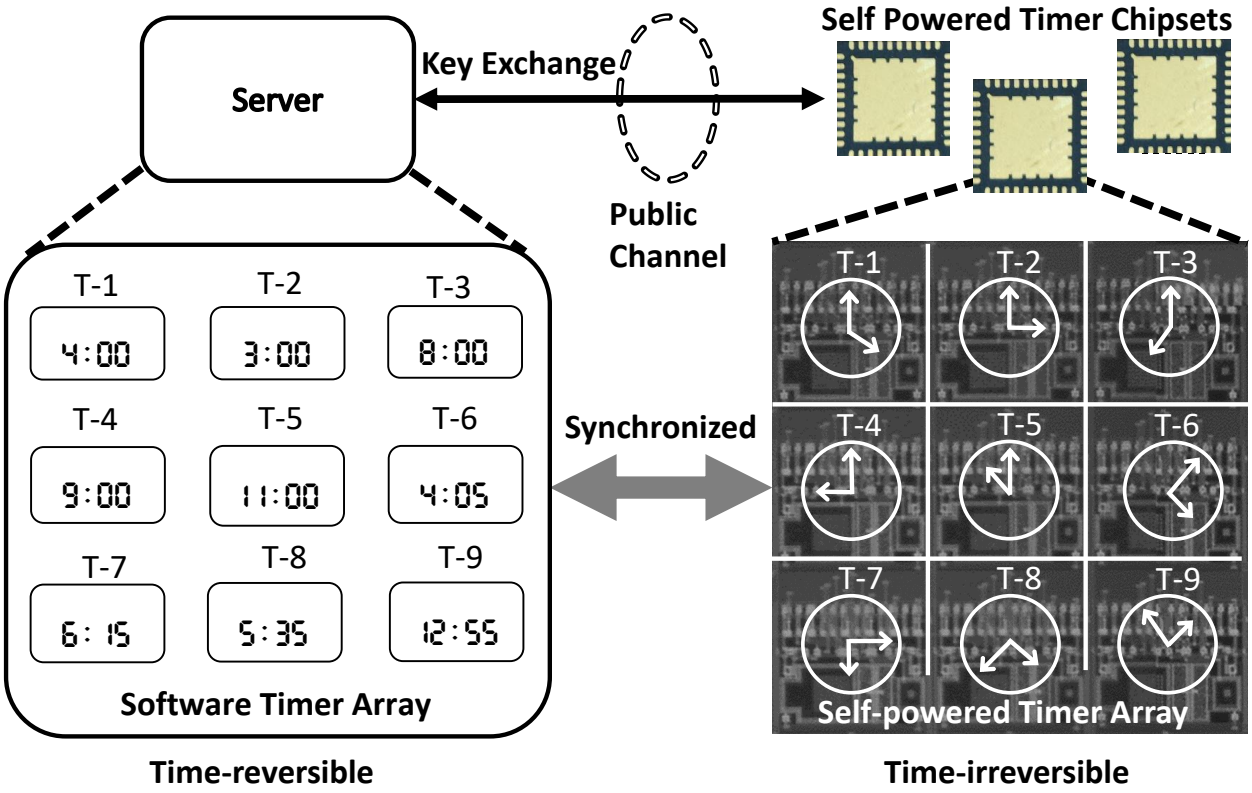

Figure 4.1: Framework underlying SPoTKD protocols: the synchronization and time-irreversibility of self-powered timers is exploited to implement one-way functions and facilitate secure key exchange over public channels.

to be secure against quantum attack [95], provided the security of the initial key-exchange can be guaranteed. Quantum key distribution(QKD)[16] which is based on the principles of quantum-mechanics, like quantum entanglement [41] or the no-cloning principle [50],[101] could be used to guarantee the security of the initial key-exchange. However, one of the major drawbacks of current state-of-the-art QKD systems is that they require dedicated and specialized peer-to-peer communication links [124],[37],[72],[134]. Not only do these links require careful maintenance and calibration to ensure quantum-coherence, but these systems are also expensive and not portable. Hence, current QKD systems cannot be scaled for internet-scale key distribution [62],[19] and communications involving lightweight IoT devices with resource constraints will still be vulnerable to quantum attacks.

In this work, we propose a hardware-software Self-Powered Timer based Key distribution (SPoTKD) framework that does not require any modifications to the existing communication infrastructure, can be scaled to a large number of IoTs, and is potentially secure against

quantum attacks. The approach relies on the trend that silicon-based chipsets with the capability of integrating billions of transistors and memory elements [32] can be manufactured on a large scale and at a low-cost [55]. If a physical feature on these chipsets could be exploited to implement a secure one-way function, then a hardware-software approach could be used to support key distribution over public channels. In this work, we propose one such method that exploits the synchronization capabilities and security features of our previously reported [137] self-powered timekeeping devices. The basic framework for SPoTKD is illustrated in Figure 4.1 where multiple identical copies of self-powered timer chipsets are openly distributed to all the users. Each of the timers on these chipsets is synchronized with its software clone running on a server. The key exchange between the server and the user is achieved based on this synchronization and time-evolution is used to implement a secure one-way function. It is to be noted that once the secret keys have been established and exchanged between the two parties, traditional symmetric cryptographic algorithms can be used for secure communications and user authentication[9].

The rest of this chapter is organized as follows. Section 4.2 briefly describes other related protocols based on hardware-software based key distribution. Section 4.3 provides a brief background of the previously reported self-powered timers and their essential security features that have been exploited in the design of the SPoTKD protocols. In Section 4.4, we propose two SPoTKD protocols, one between a server and any user, and the other between two users. In Section 4.5 we analyze the security of the proposed protocols under various adversarial attacks. The robustness of the protocol to operating and hardware artifacts have been analyzed in Section 4.6 and in Section 4.7 we introduce a variant of the protocol that uses error-correction codes to improve noise-robustness. We conclude the chapter in Section 4.8 with discussions about the challenges.

4.2 Related Works

In the literature, a few hardware-software key exchange methods have been proposed. In [93] a hardware-software public-key cryptography system for wireless networks was proposed based on Rabin's Scheme [103]. However, the security of Rabin's Scheme relies on the difficulty of factorizing large numbers, hence, it has similar vulnerabilities as the classical DH or RSA methods. Meanwhile, the one-way function (time irreversibility) implemented

in SPoTKD is based on the principle of physics. Thereby SPoTKD does not suffer from such vulnerabilities. In [36] a hardware-software key exchange technique was proposed that exploited correlations across chaotic wavepackets in classic optical communications channels. However, the method still requires peer-to-peer connectivity between the users and hence has similar scaling disadvantages as QKD methods. On the other hand, SPoTKD uses silicon-based chipsets containing self-powered timers. Therefore, SPoTKD has the advantage against such key distribution methods for platforms with low computational resources. The hardware-software approach proposed in [66] used chaos synchronization to distribute random keys over public channels. However, due to the lack of reliable synchronization, this approach incurs significant errors during decryption. Recently, Physical Unclonable Function(PUF) based hardware-based encryption key distribution has been proposed. A specific variant of this technique, described in [14] as Public Physical Unclonable Function(PPUF) has been used for public-key cryptography and leverages the difficulty of accessing physical information stored on chipsets. However, in PPUF the stored information is static in nature and hence is potentially vulnerable to machine learning attacks [34, 86]. Whereas in SPoTKD the keys are derived from dynamic information that changes with time.

4.3 Self-powered Timer Security Primitives

The SPoTKD protocol exploits the physical features of self-powered timers to ensure the security of the key exchange. The design and the operating principle of self-powered timers have been previously reported in [137, 138]. In this section, we discuss the basic security primitives offered by the timer's physical response that will form the axiomatic core of the security analysis for SPoTKD that is presented later in this chapter.

4.3.1 Self-powered timers are immune to power side-channel attacks

A simplified equivalent circuit model of the self-powered timer is shown in Fig 4.2(a) where a leakage-current J_{tunnel} is used to discharge a floating-gate capacitor C_T. Thus, once the floating-gate capacitor C_T is charged or programmed initially, no external power is required to drive the dynamics of the discharge process. The change in the floating-gate charge/voltage

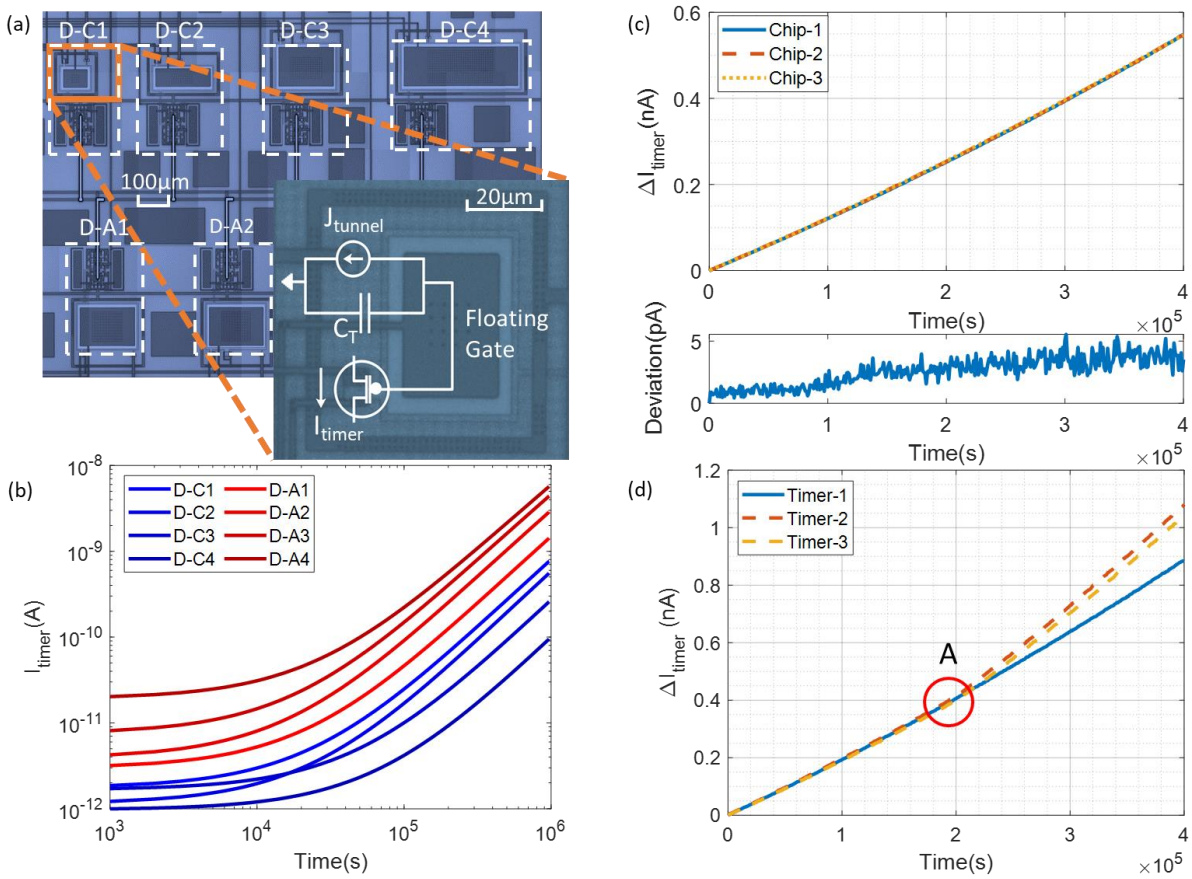

Figure 4.2: (a) Micrographs of self-powered timers (labeled as D-C1, D-C2, D-C3, D-C4) with different form factors and features that determine the parameters of the timer behavioral model in equation 4.1. The equivalent circuit model for a single timer along with the readout circuit is shown in the inset. (b) The temporal responses were measured using these timers for (a) different initialization conditions. (c) Synchronization of a timer's temporal response with the same form factors across multiple chipsets after the initial transient response. (d) Desynchronizing the temporal response of different timers by coupling an external source of energy into two of the timers at the time-instant denoted by A.

is monotonic with respect to the time elapsed and this feature has been previously used for time-keeping, synchronization, and authentication [137, 5]. For this work, the self-powered operation decouples the timer from the external power supply. This provides security against any power side-channel attack that might be aimed at gaining knowledge about the current state of the timer by observing fluctuations in the supply-current.

4.3.2 Self-powered timers are immune to electromagnetic side-channel attacks

The leakage current J_{tunnel} in the self-powered timer is implemented using Fowler-Nordheim(FN) tunneling of electrons through a thin gate-oxide barrier. In [138], we have shown that the operation of the timers is robust even when the FN tunneling current is as low as one electron per second (or less than an attoampere). From a security point of view, the low tunneling current practically eliminates any electromagnetic (EM) emission and hence any EM side-channels. Also, any unauthorized attempt to access the timer-state using an EM probe desynchronizes or destroys the state of the timer.

4.3.3 Dynamics of the self-powered timers can be synchronized

One of the essential attributes of the timer that is important for the realization of the SPoTKD protocol is that the timer's temporal responses can be synchronized not only with respect to each other but also to a well-defined behavioral (or software) model. For this work, we use a specific form of the timer behavioral model that is given by

$$I_{timer}(t) = p_3 \exp\Big[-\frac{p_2}{\log(p_1 t + p_0)}\Big]. \tag{4.1}$$

where $I_{timer}(t)$ is the current measured at time instant t quantifying the state of the timer. The current is measured using a read-out metal-oxide-semiconductor field-effect transistor (MOSFET) whose gate is coupled to the floating-gate, as shown in Fig. 4.2(a). The behavioral model in equation 4.1 assumes that the read-out transistor is biased in a specific regime, details of the derivation of the behavioral model are given below.

In [137] it was shown that the floating gate(FG) potential of the timer can be described using a first-order differential equation

$$V_{FG}(t) = \frac{\beta t_{ox0}}{ln(p_1 t + p_0)} \tag{4.2}$$

where $V_{FG}(t)$ is the potential of the timer at time instant t and p_0 and p_1 are the model parameters defined as

$$p_0 = \exp\Big(\frac{\beta t_{ox0}}{V_0}\Big), p_1 = \frac{A_0 \alpha \beta}{C_T t_{ox0}}$$

Here V_0 refers to the initial voltage, A_0 is the tunneling junction area, t_{ox0} is the average oxide thickness of the device and C_T is the total capacitance of FG. α and β are functions of the material properties and depends on the technology used for fabricating the timers. When the floating gate is coupled to the gate of a readout MOSFET which is biased in weak-inversion, then its drain to source current($I_{timer}(t)$) can be expressed as

$$I_{timer}(t) = I_0 \exp\Big[\frac{K_p(V_s - V_{FG}(t) - V_T)}{U_T}\Big] \tag{4.3}$$

where V_s is the source voltage, V_T is the threshold voltage of the MOSFET, I_0 represents a characteristic current of the MOSFET, K_p is the gate efficiency and U_T is the thermal voltage. Substituting $V_{FG}(t)$ from equation 4.2 we get

$$I_{timer}(t) = p_3 \exp\Big[-\frac{p_2}{ln(p_1 t + p_0)}\Big] \tag{4.4}$$

where $p_2 = \frac{K_p}{U_T}\beta t_{ox0}$, and $p_3 = I_0 \exp\Big[\frac{K_p(V_s - V_T)}{U_T}\Big]$ The tuple $\overline{P} = [p_0, p_1, p_2, p_3]$ in equation (4.1) are the timer parameters that are determined by the device form factors and the device initialization conditions. Figure 4.2(a) shows an example of a system-on-chip implementation that integrates different timer structures with varying form-factors. The responses of these timers with different initialization conditions are presented in Figure 4.2(b) which shows that the temporal dynamics of each timer is unique and is determined by the tuple $\overline{P}$. We have previously shown that for a fixed set of timer parameters $\overline{P}$ the mathematical model in equation 4.1 can capture the temporal behavior of the timer for more than a year with an accuracy of greater than 0.5% [138]. This is shown in Figure 4.2(c), where the timers with the same form-factor but integrated on different chipsets remain synchronized with each other. The deviation between the timer's responses is in the range of pico-amperes and this synchronization error can be attributed to the measurement noise and not to the synchronization error. For the SPoTKD protocol, the synchronization between the behavioral model (or software timer) and the hardware timers will be used for key exchange. The key exchange will exploit the asymmetry between the software timers and hardware timers where that the hardware timer cannot be rewound (or time-irreversible) whereas its software clone can be rewound to any previous time instant. This asymmetry is exploited as a one-way function for securing the SPoTKD protocol. Note that the parameters $\overline{P}$ which determine the dynamics of each timer, are never revealed publicly and therefore function as a private key in our protocol. Later in Section 4.5, we show that it is practically impossible to extract these parameters from measurements on the hardware timer itself.

4.3.4 Self-powered timers are designed for one-time read and tamper-resistant

In [138] we showed that the synchronization between the timers could be broken by injecting an external signal into the floating-gate. This is demonstrated in Figure 4.2(d) where three timers (with similar form factors) are synchronized with respect to each other till time-instant 'A'. Then at time-instant 'A' an external energy-source is coupled to timers 2 and 3 (in this case using capacitive coupling). As a result, these timers become de-synchronized from each other. We will use this controlled de-synchronization feature to intentionally destroy the dynamical state information stored on each timer once its state has been accessed. Thus, each of the timers can only be used once to generate the key-string after which the state of the timer is destroyed (or desynchronized). Note, the desynchronization of the timer can also result when the timer is unintentionally probed (using hardware delamination or using electromagnetic probing). This feature makes the basic timer tamper-resistant.

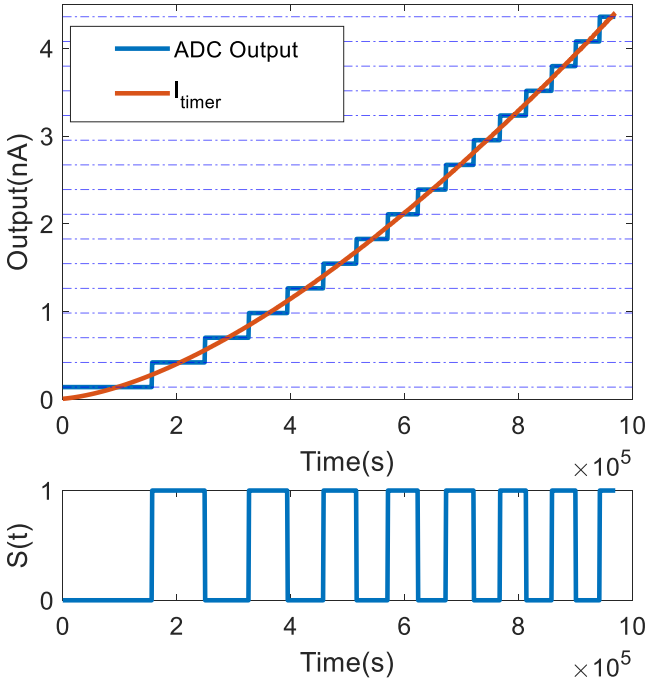

Figure 4.3: Dynamic binary state $s(t)$ of a timer generated after the analog current is read-out with an ADC. Illustration here shows the state $s(t)$ corresponding to a 4-bit ADC.

4.3.5 Bit generation using self-powered timer

We will assume that the state of the self-powered timer can be measured using an on-chip analog-to-digital converter(ADC) where the least-significant-bit (LSB) represents a modulo-2 measurement of the timer value. Denoting the binary state $s(t) \in \{0, 1\}$ of the timer as the LSB obtained after the $I_{timer}(t)$ is measured at a time-instant t, then $s(t)$ can be expressed as

$$s(t) = \lfloor \frac{I_{timer}(t)}{\delta} \rfloor \mod 2 \tag{4.5}$$

where δ is the resolution of the ADC. This is illustrated in Figure 4.3 where a 4-bit ADC is used to measure $I_{timer}(t)$ to generate the LSB or $s(t)$. For the protocols proposed in this work, we will also assume that once the binary state of a timer is measured, its state is destroyed through a process of desynchronization, as described in the section 4.3.4. This implies that each timer can only be used once to generate a single bit '$s(t)$' at a given time t for key-generation.

4.3.6 Summary of hardware security primitives offered by self-powered timers

Here we summarize the security primitives that are offered by self-powered timers and will serve as axioms for the proposed SPoTKD protocol:

SP1: It is practically impossible to access any information about the secret parameters or the state of the timer using side-channels (power or electromagnetic) attacks.

SP2: The temporal behavior of each timer is unique and is determined by the timer's secret parameter tuple $\overline{P}$.

SP3: The binary state of a timer $s(t)$ as defined in equation 4.5 is dynamic in nature and changes with time. As a result, the state of a timer is unpredictable without knowledge about the secret parameters of the timer.

SP4: The hardware chipsets are designed in such a manner so that users are limited to only the output of the chipsets after following a specific protocol (discussed in Section IV). Any attempt to snoop on the hardware chipsets otherwise would result in the destruction of the information embedded in the timers.

SP5: The state of each timer in a chipset can only be accessed once, after which the state is erased or destroyed.

SP6: The number of hardware chipsets that are available at any given instance of time is finite.

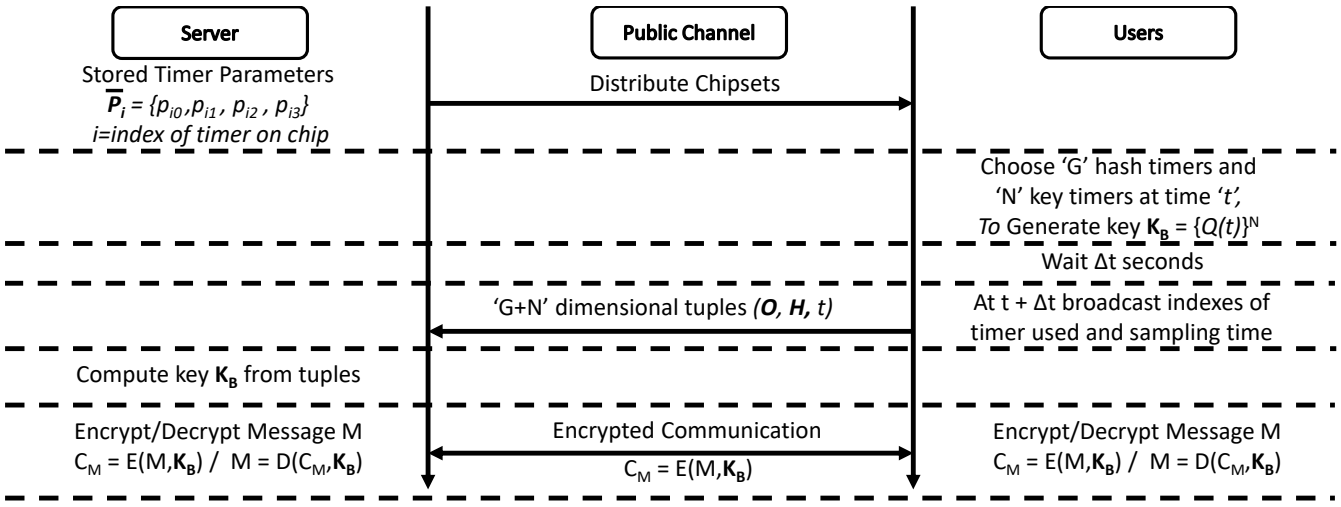

Figure 4.4: Basic SPoTKD protocol between the server and a user. Here $E(M, \mathbf{K_B})$ represents an encryption function where message M is encrypted with key $\mathbf{K_B}$, $D(C_M, \mathbf{K_B})$ is the decrypting function where C_M is the cipher text being deciphered with key $\mathbf{K_B}$.

4.4 SPoTKD Protocol

The basic SPoTKD protocol is shown in Figure 4.4. A server creates multiple replicas of chipsets each of which integrates a set $\mathcal{T}$ of $C \in \mathbb{Z}^+$ timers. Each timer in the set is assumed to be initialized according to a parameter tuple $\overline{P}_i$, where $1 \leq i \leq C$, as defined in equation (4.1). Note that some of the parameters (initial charge on the floating-gate) in the tuple are programmed by the server and some of the parameters (device form-factor) are fixed post-fabrication. Also, note that only the server has access to this information, and is kept secret from the users. These identically programmed chipsets are then distributed to all the users over a public distribution channel, as shown in Fig. 4.1. When an intended user wishes to communicate with the server, they arbitrarily choose to measure the binary states of two sets of timers which will be referred to as 'hash' timers and 'key' timers. The

objective is to use the G 'hash' timers and N 'key' timers to generate an N bit long binary key $\mathbf{K_B} \in \{0,1\}^N$. To achieve this the outputs of G randomly chosen hash timers $s_{H_1}(t), .., s_{H_G}(t)$, $1 \leq H_1, .., H_G \leq C$ measured at time instant t are XOR-ed with each other to generate a single bit $X(t)$ according to

$$X(t) = s_{H_1}(t) \oplus s_{H_2}(t) \oplus s_{H_3}(t) ..... \oplus s_{H_G}(t) \tag{4.6}$$

Note that the time instant $t \in \mathbb{R}^+$ is referenced according to a universal standard time. The key bits $Q_L(t), L = 1, .., N$ are then generated at time t by XOR-ing the binary states of each of the 'key' timers $s_{O_1}(t), .., s_{O_N}(t); 1 \leq O_1, .., O_N \leq C$ with $X(t)$ according to

$$Q_L(t) = s_{O_L}(t) \oplus X(t) \tag{4.7}$$

to generate $\mathbf{K_B} = \{Q_L\}^N$.Note that since the state of each of the timers can only be accessed once, the 'hash' and the 'key' timers need to be different, namely $\{O_1, .., O_N\} \cap \{H_1, .., H_G\} = \emptyset$. Also, note that the user can only access the N bit key string $\{Q_L\}^N$ and not the binary states of the 'key' timers or $X(t)$ from the hardware chipsets.

In the next step of the SPoTKD protocol, as shown in Figure 4.4, the user waits for a random time-duration Δt seconds after which they broadcast a $G + N$ dimensional tuple $(\mathbf{O}, \mathbf{H}, t)$ over the public channel. Note that here t indicates the time at which the G 'hash' and N 'key' timers were accessed and only the indices of the timers are broadcasted (and not measured output). The server then uses the tuples $(\mathbf{O}, \mathbf{H}, t)$ and its knowledge of the 'secret' parameters $\overline{P_i}, 1 \leq i \leq C$ to decipher the binary states of all these timers and compute the key $\mathbf{K_B}$ completing the key exchange.

The SPoTKD protocol shown in Figure 4.4 is suitable for communicating between a user and a server that owns and initializes all the timer chipsets. However, key exchange between two users can also be facilitated with the help of the server acting as a trusted third party, as shown in Figure 4.5. In this protocol, both the users broadcast their tuples $(\mathbf{O}, \mathbf{H}, t)_A$ and $(\mathbf{O}, \mathbf{H}, t)_B$ over a public channel. The server deciphers both keys, $\mathbf{K_A}$ and $\mathbf{K_B}$ according to previous protocol. The server then generates a new key $\mathbf{K_R}$ which is a function of the keys $\mathbf{K_A}$ and $\mathbf{K_B}$. This function $f : \{0,1\}^{2N} \rightarrow \{0,1\}^N$ is decided by the server and can

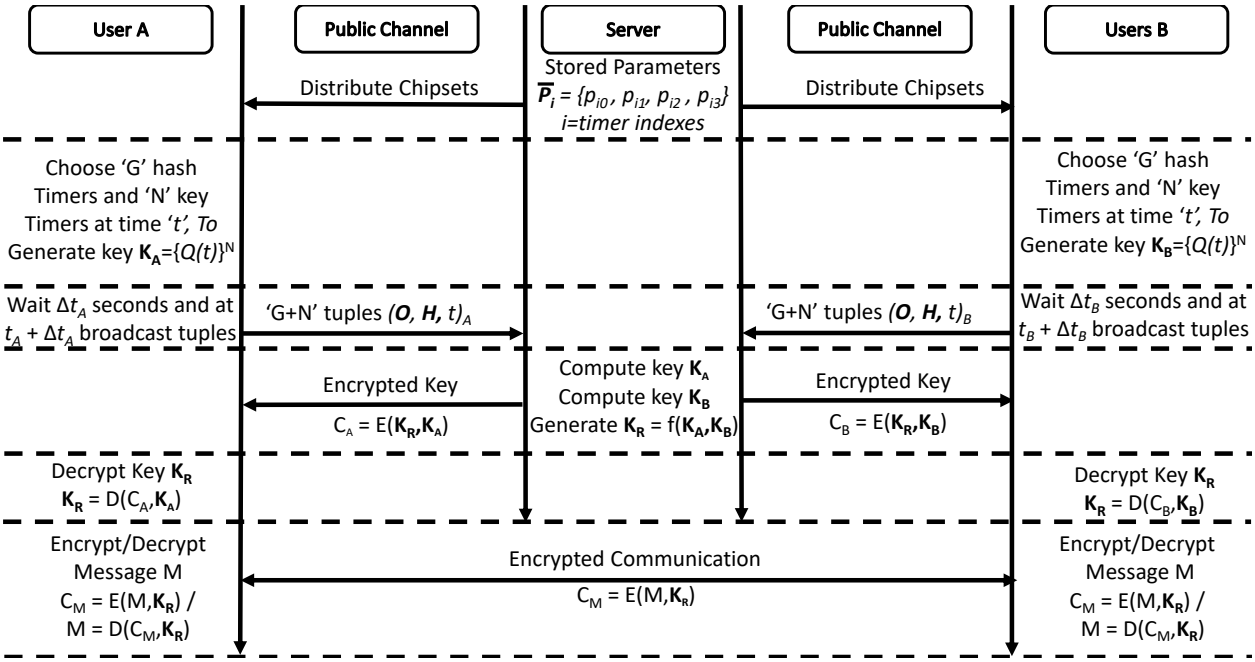

Figure 4.5: SPoTKD protocol for exchanging keys between two users with the server acting as a trusted third party.

be any mathematical operation ranging anything from multiplication to complex hashing. This operation is never revealed and changed for every session. The server then sends cipher texts $C_A = E(\mathbf{K_R}, \mathbf{K_A})$ to user A and $C_B = E(\mathbf{K_R}, \mathbf{K_B})$ to user B containing the key $\mathbf{K_R}$ encrypted using $\mathbf{K_A}$ and $\mathbf{K_B}$ respectively. The users can decrypt the cipher text to know the secret key $\mathbf{K_R}$. For further communication, each user uses this key $\mathbf{K_R}$ to encrypt and decrypt their messages with each other. Since all keys are randomly generated and have never been used before then anyone intercepting the cipher text will not gain any information regarding the secret key being used. Note that in this protocol the users do not need to match either the timers they used in the chip or the time at which they will generate their respective keys. They only need to agree upon their time of communication and can generate their keys beforehand individually. In order to update any new session key between two users, the users would need to use a new set of timers and follow the same protocol for exchanging keys with the server acting as the trusted third party.

4.5 Security and Performance Analysis

According to the recommendation by the National Institute of Standards and Technology (NIST), a 256-bit key is sufficient for symmetric key algorithms to be secure [49] even in the presence of a quantum computer. While using Grover's algorithm, a quantum computer with 6681 logical qubits and approximately 3.36×10^7 physical qubits would require around 2.29×10^{32} years for a brute force search attack on AES-GCM cryptosystem with a 256-bit key size [95]. Hence, for the rest of the chapter, we will show test results corresponding to 256-bit keys, generated from $G = 128$ hash timers and $N = 256$ key timers for all analysis purposes. Note that, the number of hash timers used in key generation determines the complexity of the key generation. We will show that $G = 128$ hash timers are sufficient for the protocol to be secure. Increasing the number of hash timers would further increase the complexity but would come at the cost of noise robustness. Since the scope of this work is only to propose a secure key exchange protocol that can be used for symmetric-key encryption schemes, our security analysis will only focus on showing that the key exchange protocol is quantum secure.

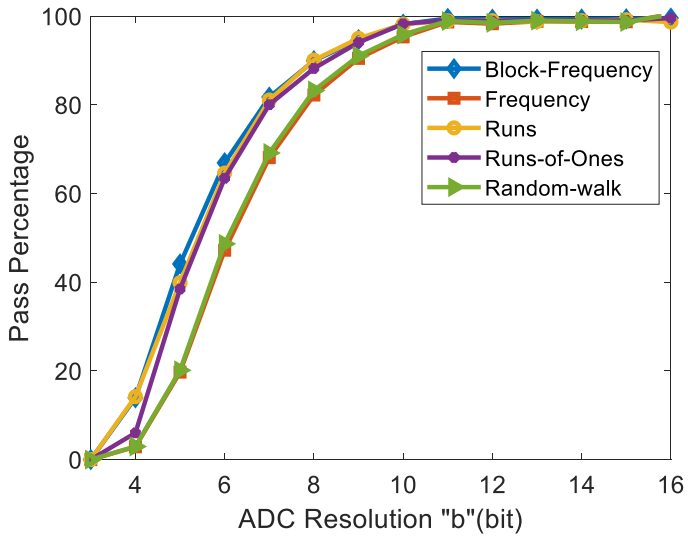

Figure 4.6: Pass percentage obtained using the NIST randomness test suite applied to the keys generated using the SPoTKD protocol, as a function of the resolution b of the ADC used to measure the state of the 'key' timers.

For our first analysis, we consider the scenario where an attacker simply attempts to guess the key without any information about the key generation system. As long as the keys that are generated from the timers are completely random in nature, the attacker will not gain any unfair advantage. So we tested the secret keys generated according to the SPoTKD protocol described in Section 4.4 with the NIST test suite for checking the randomness of bit stream[13]. The suite usually consists of fifteen different tests to measure the randomness in a certain bitstream. However, a few of these tests require a large sequence of bitstreams which does not apply for a length of 256-bit keys. Therefore, in our analysis, we show the test result for 5 of the suitable tests. The binary states for the hash timers were always measured with an 11-bit ADC irrespective of the key timers. This was performed to ensure better noise robustness. If a higher resolution ADC was used to sample the hash timers, then the noise robustness of the protocol would decrease (discussed in Section VI). Using Monte Carlo simulations, we sampled 10^6 keys at random time instances using a b-bit ADC (or $2^b - 1$ level quantizer) for the key timers. We extracted the parameter tuples $\overline{P} = [p_0, p_1, p_2, p_3]$ from the timer responses shown in Figure 2(b) and then randomized within the range of these actual hardware parameters to represent unique timers in our simulations. This ensures that each timer used in the simulation can actually be realized in hardware chipsets. Figure 4.6 shows the pass percentage, i.e. the percentage of keys from the 10^6 samples that passed the test, as the resolution 'b' of the ADC is varied for the key timers. We can observe from the plots that for large values of 'b', almost all the generated keys pass the test. The randomness degrades for ADC resolution less than 8 bits showing that a 9-bit ADC for the key timers should be sufficient to generate high-quality keys. This shows that keys derived from the timer responses are completely random in nature and any attempt to guess the key would result in a brute-force search which is the same as breaking the AES-256 encryption scheme discussed above. Moreover, it also means that the binary state of each timer is uncorrelated with other timers and an attacker cannot simply sample the binary states of any one timer and can predict what other timers' response would be at any given point in time. This is in accordance with the axioms SP2 and SP3 discussed in Section 4.3.

Next, we consider the information that is available to an attacker and investigate whether he can gain any advantage while predicting the key-string with the information available to him. So, here we note down all the information and resources about the key exchange framework that is potentially available to an attacker:

I1: We assume that the attacker can passively eavesdrop on the communications over the public channel. This means that the attacker would know which timers were used for a particular key-string.

I2: We also assume that the attacker has access to the hardware chipsets.

I3: The attacker knows the underlying principle of the timers' behavior and other logistics of the protocol as described in this work.

I4: The attacker has access to a fully functioning quantum computer.

Now considering I1 and I2, a potential attack could be launched by the adversary where they sample the timers on their copy of the chipset as soon as the user broadcasts a tuple $(\mathbf{O}, \mathbf{H}, t)$ over the public channel. However, the key that the attacker generates will be at a time instant $t + \Delta t$, where Δt is the time that the user waits after they have generated the key. Since the timer values are dynamic in nature, the key generated by the attacker $\mathbf{K_E}$ will be different from the key generated by the user $\mathbf{K_B}$. To quantify the disparity between the keys, we use Shannon information entropy to measure how much information can the attacker gain about $\mathbf{K_B}$ using their own key $\mathbf{K_E}$. The average Shannon information entropy contained in each bit generated by the attacker can be expressed as

$$H_{SE} = -d \log_2 d - (1-d) \log_2 (1-d) \tag{4.8}$$

where d is the average difference in bits between $\mathbf{K_B}$ and $\mathbf{K_E}$. The parameter H_{SE} quantifies the uncertainty of the attacker for every bit of the key $\mathbf{K_B}$ that he or she tries to predict using $\mathbf{K_E}$. When $d = 0$ i.e. the attacker generates the same key as the user, the information entropy of the attacker is zero, this is because the attacker can predict the key with perfect certainty. A similar argument can be made for the other extreme scenario, when $d = 1$, as the attacker can simply invert each bit that he or she generates and produce $\mathbf{K_B}$. The entropy H_{SE} is also equal to 0 in this case. On the other hand, when $d = 0.5$ exactly half of the bits of $\mathbf{K_E}$ do not match with $\mathbf{K_B}$. This means that if the attacker were to randomly guess all the key-bits they would, on average, end up with the same number of matched bits. Therefore, the attacker has 1 bit of uncertainty for every bit generated and zero information gain on the key. The entropy H_{SE} thus takes the maximum value of 1 in this case.

In order to mimic such a kind of attack we sampled a set of timers and generated keys at random time instances, representing the user's key, and also sampled the same set of timers

at a later instant, which represents the attacker's key. After that, we calculated the entropy for each sample. Figure 4.7 shows the average uncertainty per bit generated by the attacker when he or she samples the same timer array used by the user. We can observe from the figure that for keys generated with high-resolution ADC, the attacker has almost 1 bit of uncertainty per bit. This means that the attacker is unable to gain any information about the user's key from sampling their own timer chipset. The overall trend for the curves with the same wait period (which corresponds to Δt) can be explained by the fact that at higher resolution, the LSB contains minimum information about the whole dynamic response of the timer. Moreover, the LSB changes much more frequently, and therefore key generated using LSB is more difficult to predict for the same waiting period. It gets increasingly easier to predict as the resolution of the ADC is decreased since the LSB changes slowly and sampling yields more information. The uncertainty can be increased for a lower-resolution ADC by

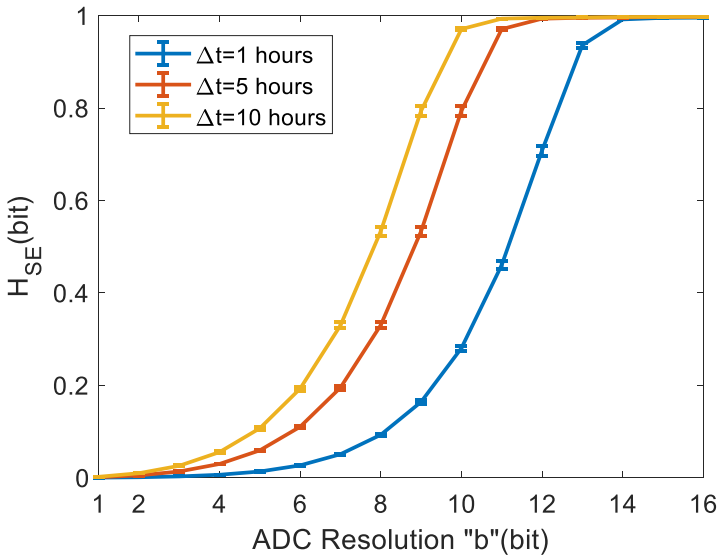

Figure 4.7: Uncertainty per bit measured for three different waiting periods Δt as a function of the resolution b of the ADC used for measuring the state of the 'key' timers.

increasing the waiting period Δt which is shown in Figure 4.7 where the curve shifts towards the left as we increase Δt. This is because as Δt is increased, the probability that an ADC bit has changed will also increase, thereby sampling the bits will not provide any useful information.

However, average Shannon information entropy H_{SE} is agnostic of the position of the mismatched bits. For instance, one pathological case could be that always the first half of the key-string obtained by the attacker is mismatched while the second half always matches with the true key-string. In this scenario, HSE would still be 1, but the attacker can easily guess the correct key-string. In our next analysis, we show that the probability of such a case is negligible (practically does not exist). Using Monte Carlo simulations with ADC resolution b=12-bits, we counted how many times each of the key-bits in the 256-bit key string gets mismatched among all the iterations and calculated the probability of mismatch for each bit index. Figure 8 shows the probability of mismatch for each bit index after different waiting periods. We can observe that as the waiting period increases each bit index has an approximately equal probability of 0.5 for being mismatched. This shows that there is no bias with respect to the positioning of the mismatched bits and each key bit generated by the attacker has an equal probability of being correct or incorrect which is the same as purely guessing. For a lower waiting period, the probability of mismatch decreases for all the bit indices which is in accordance with our previous analysis, but the mismatch probability is approximately the same irrespective of the bit position. Thus, as long as a reasonable resolution ADC is used for measuring the state of the timer and the waiting period is large enough, the attacker will not be able to predict what key string was generated by a user. Therefore, I1 and I2 do not reveal any information about the secret key and the attacker would still need to resort to brute force search for a successful attack. So far we have shown that the key exchange protocol is secure based on the facts that the keys used are completely random in nature and from the public information available during the key exchange the attacker can not gain any information about the random keys. Next, we consider I3 available to an attacker and investigate whether they could predict the keys by using their knowledge about the timer behavioral (or software) model. However, since they do not have access to the timer initialization parameters $\overline{P_i}$, they cannot use the public information $(\mathbf{O}, \mathbf{H}, t)$ to decipher the states $s_{O_l}(t)$. Also, the attacker is unable to rewind the hardware timer on their copy of the chipset to measure the states $s_{O_l}(t)$ going back in time. Therefore, the only way to predict $\mathbf{K_B}$ would be to solve equation 4.7 for each bit of the key for finding the secret parameters $\overline{P_1}, \overline{P_2}...\overline{P_N}$. In the next set of analyses, we will show that it is practically impossible to find the secret parameters from the hardware chipsets themselves.

First, we consider equation 1 where the parameters could be regressed if a timer is sampled multiple times to measure $I_{timer}(t)$ at different time instances. However, this is only true if the attacker can get access to the precise value of $I_{timer}(t)$. From equation 2 we observe that the

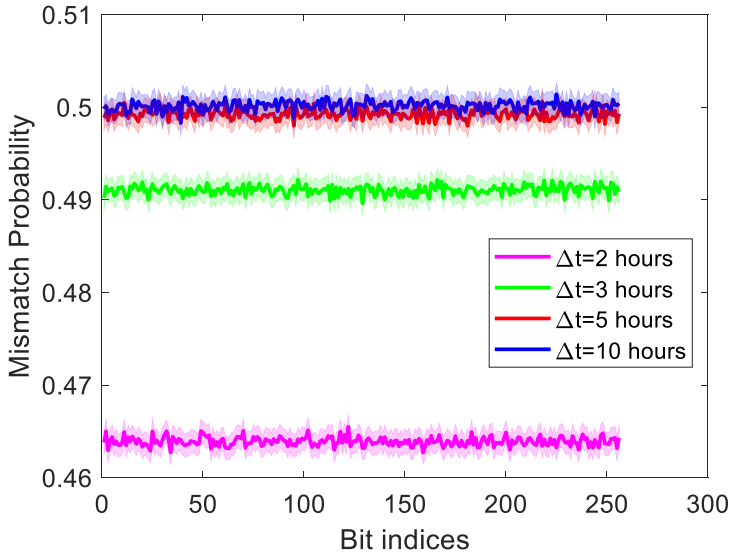

Figure 4.8: Probability that the binary states of timers used in key generation have changed after the waiting period Δt hours. Here the resolution of the ADC used for key generation is b=12-bits. The variance across different Monte-carlo trials is highlighted by the shaded region.

binary state of the timer only provides a single bit of information about $I_{timer}(t)$. Moreover, axiom SP4 dictates that even the single bit of information about $I_{timer}(t)$ is XOR-ed with other G hash timers' binary states. The attacker has only access to the XOR-ed output due to the manner in which hardware chipsets are designed. Therefore each bit of key-string the attacker samples from the hardware chipsets will be derived from $G+1$ timers. Note that there is no analytical solution for equation 4.7 so the attacker will have to resort to a brute-force numerical search. We now show how the SPoTKD protocol is secure against such attacks under the standard model.

Claim 1. The SPoTKD protocol is secure under the standard model.

Proof. Each key bit $Q_L(t)$ is derived from the temporal responses of $G+1$ timers where G is the number of hash timers used in key generation. Now, the temporal response of each timer is determined by the secret parameter tuples $\overline{P} = [p_0, p_1, p_2, p_3]$. Therefore, each key bit, in turn, is determined by $G+1$ tuples of $\overline{P}$. We define p_{Total} as the total number of

parameters from which each bit is derived which is given by

$$p_{Total} = 4(G + 1) \tag{4.9}$$

This means that the search space would be a matrix with p_{Total} dimensions. Now, let R be the range of possible values for each of the p_{Total} parameters. Then the total number of elements in the matrix i.e. the total search space SP_{Total} would be given by

$$SP_{Total} = R^{4(G+1)} \tag{4.10}$$

Even though the parameters $\overline{P}$ are determined by the timer initialization conditions and timer form factors, they are calibration parameters. Assuming a double-precision floating-point for the parameters implies that $R = 2^{63}$. This yields

$$SP_{Total} = 2^{252(G+1)} \tag{4.11}$$

For $G = 128$ hash timers (which were used in our simulations for generating the key string) this would result in a search space of 2^{32508} possible combinations. Therefore, an attacker employing a brute-force search strategy would require 2^{32508} bits of storage, which is prohibitively large. Moreover, even if the attacker uses the fastest computer in the world [126], which can perform 10^{19} computations per second, it will take them approximately 2^{32444} seconds, or 2^{32419} years to search the entire space. Since we assumed that the attacker is only constrained by the computational/storage resources and time available to them, hence, under the standard model, the SPoTKD protocol is secure. □

Next, we consider I4 where we assume that the attacker has access to a quantum computer with large enough storage space and computational resources to search the aforementioned solution space in a reasonable amount of time. In this analysis, we show that our protocol remains secure if we impose a physical constraint that limits the number of hardware chips that the attacker can use for measurement.

Claim 2. The SPoTKD protocol is resistant to quantum attacks.

Proof. Equation (4.7) has no unique solution and since the parameters are randomly chosen by the server, every solution within the search space is equally likely to be the correct one. The only way to eliminate possible combinations from the solution set would be to sample

each hardware timer at multiple time instances and solve equation 4.7 repeatedly. Since equation 4.5 is symmetric the expected size of the solution set, denoted as $\mathbb{E}(SP_J)$, after each sampling reduces by

$$\begin{aligned}\mathbb{E}(SP_J) &= \frac{SP_{Total}}{2^J} \\ &= \frac{2^{252(G+1)}}{2^J}\end{aligned} \tag{4.12}$$

where $J \in \mathbb{Z}^+$ indicates the number of samples. This means that if the attacker can sample each timer enough number of times, they can find out the initialization parameter $\overline{P}$. However, since the timers are designed for one-time reading (Axiom SP5 in section 4.3), the attacker is unable to make multiple measurements on a timer using the same chipset. For each measurement, the attacker would therefore require a new chipset. Thus, there is an upper bound to the number of measurements that an attacker can perform, which is the total number of chipsets C_{Total} available. Therefore we have

$$J \leq C_{Total} \tag{4.13}$$

Now if we constrain the total number of chipsets C_{Total} according to

$$C_{Total} < 252(G+1) \tag{4.14}$$

then the attacker would still be unable to find the unique solution to equation 4.7 since

$$\mathbb{E}(SP_J) > 2 \qquad \forall J \tag{4.15}$$

Note that, the constraint here for an attacker is not the computational power available to them but rather the physical resources they can acquire. Thus, the key exchange protocol is resistant to quantum attacks. □

In the next set of analyses, we want to show how the proposed key exchange protocol is secure against the most popular kinds of attacks.

Claim 3. The proposed protocol is secure against man-in-the-middle attacks.

Proof. During the SPoTKD protocol, a user publicly broadcasts the tuples $(\mathbf{O}, \mathbf{H}, t)$ indicating the timer indexes the user sampled along with the time at which they were sampled. For

an attacker to successfully impersonate the server, they will need to know the secret timer parameters $\overline{P}$, which is never revealed during any phase of the protocol. Also, our previous analysis shows that it is practically impossible to find out these parameters using brute-force search. Note that all the publicly distributed chipsets store the same information on the timers and authentication is carried out only after the server and user have established a secure channel subsequent to a successful key exchange. Thus, the attacker cannot impersonate any user. □

Claim 4. The proposed protocol is secure against replay attacks.

Proof. Once a set of timers is used for key exchange, they are desynchronized with respect to the server's model (Axiom SP5 in section 4.3). Thus, during every session, a new set of timers is used to exchange keys. This means that a new key is generated for every new session. Also, during the key exchange protocol, the measured states of the timers are never made public. Therefore, the attacker cannot use any information from previous sessions to their advantage. This implies that the SPoTKD protocol is secure against replay attacks. □

Claim 5. SPoTKD protocol is secure against backward and forward traceability attacks.

Proof. In our protocol, the keys generated are random in nature as shown in figure 4.6 that are not predictable. Also, each key is used only once. Therefore the key exchange at session instance SS_a can not be inferred from other keys at any other session SS_b, where $a \neq b$. Moreover, we have shown in the previous claims that inferring any knowledge about the secret parameters is also practically impossible. Therefore, the SPoTKD protocol is immune to forward or backward traceability attacks. □

Claim 6. SPoTKD protocol is resistant to de-synchronization attacks.

Proof. The robustness of the timer response ensures that the dynamics of the hardware timer remain synchronized with its software model on the server. According to Axioms SP1-SP4 in section 4.3.6, the timer's dynamic response on any user's chip cannot be programmed or altered by the attacker unless and until the attacker gets access to the chip physically. In such a case where the user suspects that his or her chip may have been compromised physically by an attacker, the user can simply discard the chip and procure a new one, since

all the chipsets have the same information that is stored. Thus, the protocol is resistant to de-synchronization attacks. □

In addition, the construction, operating principle and inherent security of the quantum-tunneling device i.e. the self-powered timer [137] also prevent the attacker to probe the state of the timer by using any side-channel (power or electromagnetic) without affecting the state of the timer (Axioms SP1-SP6 in section 4.3.6). Therefore, in this regard, the timer chipset emulates a quantum communication channel [9] but uses an analog dynamical system that is secure against any side-channel attacks.

We have evaluated the performance of our proposed protocol with similar hardware-software based key exchange protocols such as PPUF [14] and some state-of-the-art key exchange protocols such as RSA [108] that are currently being used. The comparison is summarized in Table 1 with respect to criteria such as key length, security strength, computational cost, and scalability. Here security strength measures the number of trials required to brute-force a key irrespective of the key length. A 128-bit security means 2^{128} trials to break the protocol. We also compared the computational resources required to perform a single key exchange in terms of the number of computation cycles. And finally, scalability indicates the ease at which the key exchange protocol can accommodate a large number of users. Since our goal is to provide secure key exchange among a large number of users using low resources, these features are extremely important to evaluate and compare different designs.

We start by evaluating the security strength of each protocol. For PPUF using 1024 bit key, an attacker needs to perform $1.7x10^{29}$ cycles of simulation on average to find the secret key [14]. Accounting for overhead computation this roughly translates to a 112-bit security. According to NIST 2020 recommendations, RSA requires a key length of 3072-bits to achieve a security strength of 128-bit. Now, since both QKD and SPoTKD use symmetric key encryption (AES), a 256-bit key length corresponds to a security strength of 256-bit. Due to the use of a large key size, both PPUF and RSA are computationally expensive. The PPUF based key exchange protocol requires approximately 10^{16} cycles of computation [14] and RSA requires $\mathcal{O}(10^7)$ computational cycles [107]. Even though QKD uses a much smaller key-string, additional computation needs to be performed for the error reconciliation protocol. The computational complexity is of order $\mathcal{O}(10^4)$ for a 256-bit key using common error-correcting code [24]. Meanwhile, for the basic SPoTKD, the user needs to simply measure the state of the timers once and perform $G = 128$ bit-wise XOR from the hash timers to generate

Table 4.1: Performance Comparison between SPoTKD and other state-of-the-art key exchange protocol

Protocol	**Key Length** (bits)	**Security Strength** (bits)	**Computational Cost** (no. of cycles)	**Scalability**
PPUF[14]	1024	112	$\mathcal{O}(10^{16})$	High
QKD[16]	256	256	$\mathcal{O}(10^{4})$	Low
RSA[108]	3072	128	$\mathcal{O}(10^{7})$	High
SpoTKD	**256**	**256**	$\mathcal{O}(\mathbf{10^{2}})$	**High**

the bit $X(t)$. This can be done in $log_2(G)$ computational cycles. After that, the outputs of the N key timers are XOR-ed with the bit $X(t)$ in N cycles. In this regard, our protocol is by far the most efficient. If error-correcting SPoTKD is used(discussed in Section VII), the computational cost will be similar to that of QKD. In addition, we have shown in our analysis that our protocol is resistant to quantum attacks, similar to QKD. In comparison, however, QKD is expensive and in its current state is not portable or scalable to support a large number of users. On the other hand, our protocol is based on silicon fabrication

technology which is relatively inexpensive at a production scale and the fabricated chipsets can be easily distributed to millions of users.

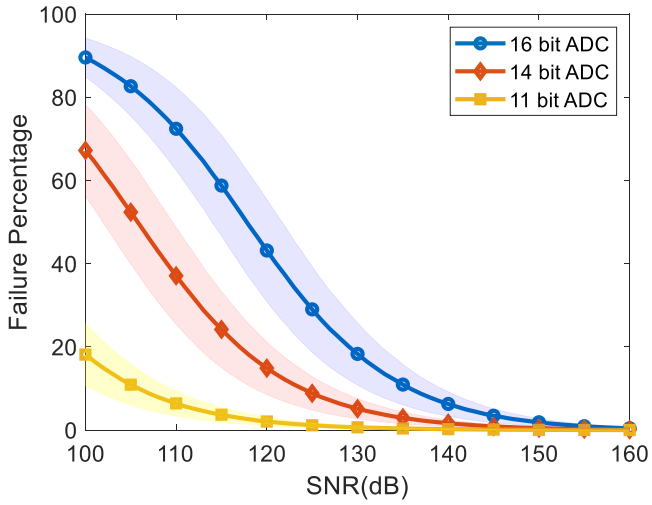

Figure 4.9: Improvement in noise-robustness of the SPoTKD protocol when the resolution b of the ADC used for measuring the state of the 'key' timers is decreased. The variance across different Monte-carlo trials is highlighted by the shaded region.

4.6 Noise Robustness

In the next set of experiments, we quantified the robustness of the SPoTKD protocols in the presence of real-world operational artifacts. For instance, the timer on a physical chip could inadvertently desynchronize with the software model on the server. This could be due to fabrication mismatch, environmental variations, device degradation, and measurement noise. To emulate this effect we performed a Monte Carlo study where we added White Gaussian Noise to the timer response and then generated the keys by sampling at random time instances.

In this case, the SNR is defined as

$$SNR = \frac{P_{Signal}}{P_{Noise}}$$

where P_{Signal} is the square of the signal output measured from the timer and P_{Noise} is the signal variance. This 'measured' key was compared against the 'gold' key generated from the software model in the server i.e. without any noise. Every instance where the keys do not match perfectly is counted as a failure. Figure 4.9 shows the failure percentage, calculated as the average number of failure instances over all the instances of simulation, at each noise level. As expected, the failure percentage reduces with an increase in SNR.

Better noise robustness could be achieved by using low-resolution ADC for the key timers, as shown in Figure 4.9. However, as we have shown in the previous section this could lead to more information gained by a 'knowledgeable' attacker to predict the key. In order to mitigate this threat, the server can recommend the user to opt for an increase in the wait-period Δt and achieve the same level of uncertainty even for low-resolution ADC, as illustrated in Figure 4.7. Thereby, a tradeoff exists between the level of security and the waiting period, and the preference for one or the other depends on the target application.

4.7 Error Correcting SPoTKD

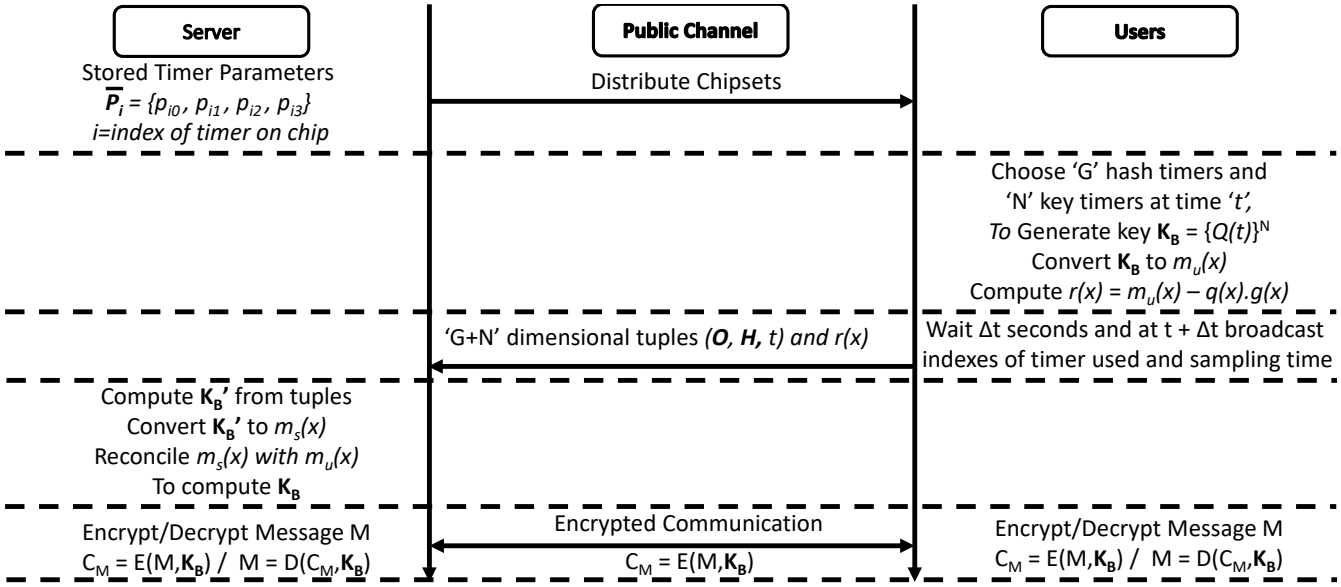

Figure 4.10: Modified SPoTKD protocol between a server and a user incorporating error-correction

In the previous section, we discussed how the protocol's robustness to noise could be increased by either trading off security or a waiting period. In this section, we will discuss a new protocol shown in Figure 4.10 in which noise robustness can be improved without compromising either security or waiting time by using standard error-correcting codes which are generally used in digital communication. For our purpose, we will use cyclic-redundancy-check (CRC) for error correction [99], even though other error-correcting codes could also be used.

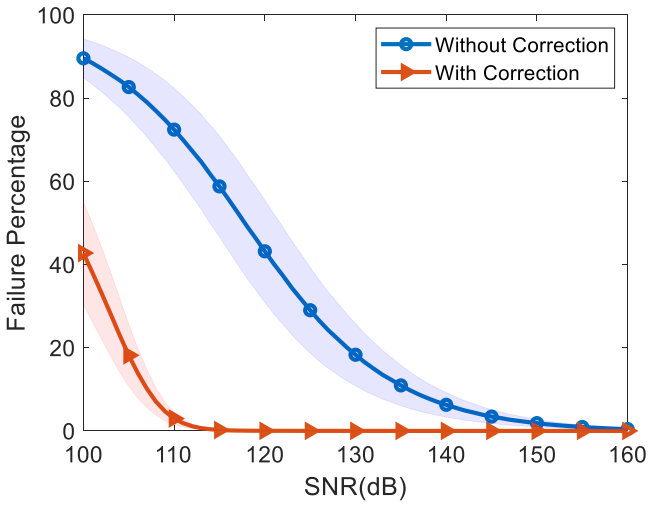

Figure 4.11: Performance of the SPoKTD protocol in the presence of noise when error-correction is used. A 16-bit ADC was used to measure the state of the 'key' timers. The variance across different Monte-carlo trials is highlighted by the shaded region.

The string of key-bits are represented as the coefficients of a message polynomial, $m(x)$, over a Galois field (GF2) and to find the CRC, the message polynomial is multiplied by x^n and then the remainder $r(x)$ is found by dividing with an n-degree generator polynomial $g(x)$. The coefficients of the remainder polynomial are the bits of the CRC. This can be expressed as

$$m_u(x).x^n = q(x).g(x) + r(x) \tag{4.16}$$

where $q(x)$ is the quotient. Typically, $m_u(x).x^n - r(x)$ and $g(x)$ is sent over the communication channel. However, in this protocol we are sending $r(x)$ i.e. only the CRC bits together with the tuples $(\mathbf{O}, \mathbf{H}, t))$ over an insecure channel as illustrated in Figure 4.10, and $g(x)$ is assumed to be predetermined and a piece of public knowledge. This is because we do not want to share the message $m_u(x)$ which is the key itself. The server generates the $m_s(x)$ using the tuples $(\mathbf{O}, \mathbf{H}, t))$ information and the software model. Then, together with $r(x)$ and

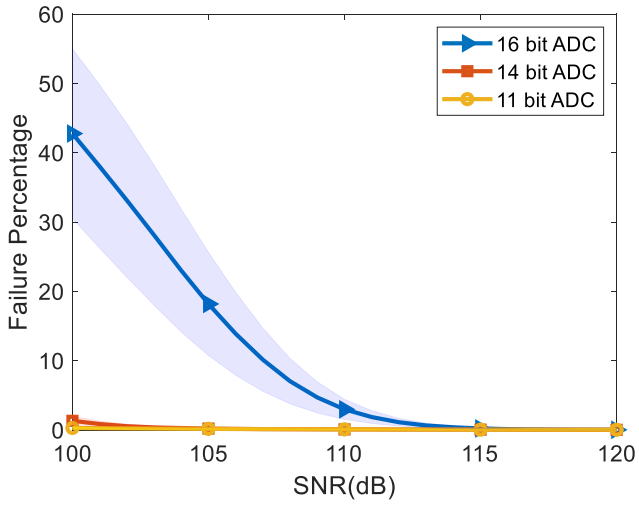

Figure 4.12: Performance of the SPoTKD protocol in the presence of noise when using error-correction and when the resolution b of the ADC used for measuring the state of the 'key' timer is reduced. The variance across different Monte-carlo trials is highlighted by the shaded region.

$g(x)$ the sever can reconcile $m_s(x)$ with $m_u(x)$ up to a certain hamming distance. Thereby, tolerating erroneous key-bits measured by the user due to noise.

From the security point of view, the attacker now has more information about the key as the remainder $r(x)$ is broadcast along with the $(\mathbf{O}, \mathbf{H}, t))$ tuples. For example, let $m(x)$ be the representation of a 256-bit key. Then the number of possible keys $= 2^{256}$. We assume that the attacker has an identical chip himself. Let $g(x)$ be a 28-degree polynomial, then with the knowledge of $r(x)$ the number of possible keys is reduced to $2^{256-28} = 2^{228}$. Therefore, the search complexity for an attacker decreases proportionally to the degree of the generator polynomial used i.e. number of CRC bits.

In order to counteract this effect, the length of the key can be increased by an amount equal to the degree of $g(x)$. This would mean more timers are needed to be used for an effective key length equal to the number of timers used minus the degree of $g(x)$. In the example described above, the number of timers required for a 256-bit effective key length would be 284.

According to Philip Koopman's table of CRC generator polynomial [71], for a $g(x)$ of 28 degrees and data-word length less than 483 bit, the least hamming distance that can be

corrected is 8. Therefore, we can allow up to 8 mismatches for the 284-bit key, which has an effective key length of 256-bits, and then compare the noise robustness to the 256-bit key. This is illustrated in Figure 4.11 which shows significant noise robustness improvement. This is achieved without sacrificing any complexity and does not come at the cost of a longer waiting period. Robustness can be further improved by using lower resolution ADC for key-generation as shown in Figure 4.12 if the user opts for more accuracy and is compliant with a longer waiting period.

4.8 Discussions and Conclusions

In this work, we introduced a novel key distribution framework, SPotKD, based on specific security features of the previously reported self-powered time-keeping devices. We described the key exchange protocol and also analyzed it both from a security and noise robustness point of view. Our protocol is not only secure against most kinds of attacks but also proved to be secure in the advent of a fully functional quantum computer in the future. We have also evaluated the performance of our protocol against some state-of-the-art key distribution schemes.

Several challenges exist in implementing the proposed key distribution system from a practical point of view. At the core of the system is the self-powered timer technology which has been successfully demonstrated in our prior work [137, 138]. However, designing the peripheral circuitry that can realize the key generation protocol on-chip is yet to be accomplished. A complete system-on-chip (SoC) should consist of an array of these timers and a combinational logic circuit that will allow the user to arbitrarily choose any set of timers for key generation. In addition, the circuit for destroying the timer's information should also be integrated into the chipsets. Furthermore, the design of the destruction circuit should be done in such a manner that the timers' temporal response becomes desynchronized even before the user can access the output of the chipsets and remain desynchronized for a significant period after read-out. Only then, the timers can be considered to be a one-time read device. Addressing this challenge would be a part of future research.

Another limitation arises in scaling the framework due to the limited number of chips that can be distributed while maintaining security against quantum attacks as discussed in claim 2. However, the limit on the number of chipsets can be increased by using more hash timers

during key generation. It should also be noted that due to real-world artifact noise, increasing the number of hash timers may lead to high failure rates during key exchange. The protocol will remain secure albeit slight increase in the probability of a key exchange failure and a trade-off exists between the security and the reliability of the SPoTKD protocol. In this regard, incorporating error correcting mechanisms in the SPoTKD protocol will help to address these limitations.

One other limitation to consider is that the underlying assumption for SPoTKD dictates that the server has ample resources to securely store the timer parameters and to secure access control. With respect to secure storage, the server can adopt traditional, high-end and computationally intensive symmetric key encryption approaches. However, the protocol in its current state will not remain secure if the server becomes compromised and the attacker gains access to the timer initialization parameters using phishing techniques or by compromising the access control protocols (similar to the attack models demonstrated for trusted program modules [52]). This vulnerability can be overcome by adopting a distributed server (Decentralized Cloud Storage) approach. The security of these types of storage systems is well established [33] where AES-256 is used to encrypt the data and then each data is split and stored across a distributed network. Another solution that we are currently investigating, is storing the timer initialization parameters in a semi-persistent storage (memory whose content is destroyed after a pre-determined time). This attribute will prevent against the "record now decode later" attacks where the attacker logs the encrypted data with the hope that a powerful computer will be available to successfully decrypt the data or the server storage will be compromised at a much later time.

Our future work would focus on prototyping a self-powered timer system-on-chip with all the basic hardware security primitives. We will then validate the SPoTKD protocol under real-world conditions and over different distribution channels. This will open the possibility of applying SPoTKD in areas such as quantum secure blockchains (based on symmetric-key) and electronic voting.

Chapter 5

SPRNG based on FN-dynamical system

This chapter builds upon the work discussed in Chapter 4. Here, I design the framework of a lightweight pseudo-random-number generator that is synchronized across multiple platforms based on the FN-timers. It provides a faster method for generating random numbers to be used by resource-constrained IoT. In addition to generating random numbers for cryptographic keys, the framework also facilitates the secure exchange of the key between two users. The key exchange protocol is an updated version of SPoTKD discussed in the previous chapter. The results in this chapter are based on [105].

5.1 Introduction

Random-number-generators (RNGs) play an important role in many applications ranging from optimization, game-theory, and simulations [7, 73, 85]. However, one of the most important uses of RNGs is in the area of secure communications [111]. Traditionally, this is achieved by encrypting the data using a sequence of random numbers i.e. cryptographic keys produced by an RNG. These keys are then synchronized using a timing reference extracted from a global-positioning-system (GPS) which also facilitates the exchange of encryption keys [51, 125]. However, for battery-powered or passive internet-of-things (IoT) devices where computational and energy resources are severely constrained this paradigm of secure communication using traditional RNGs is not practical. In this chapter, we propose a novel RNG architecture that can be used for securing communications in IoTs.

RNGs fall into two major categories, namely, the true-RNGs (TRNG) and the pseudo-RNGs (PRNG). TRNGs generate random numbers based on non-deterministic physical processes

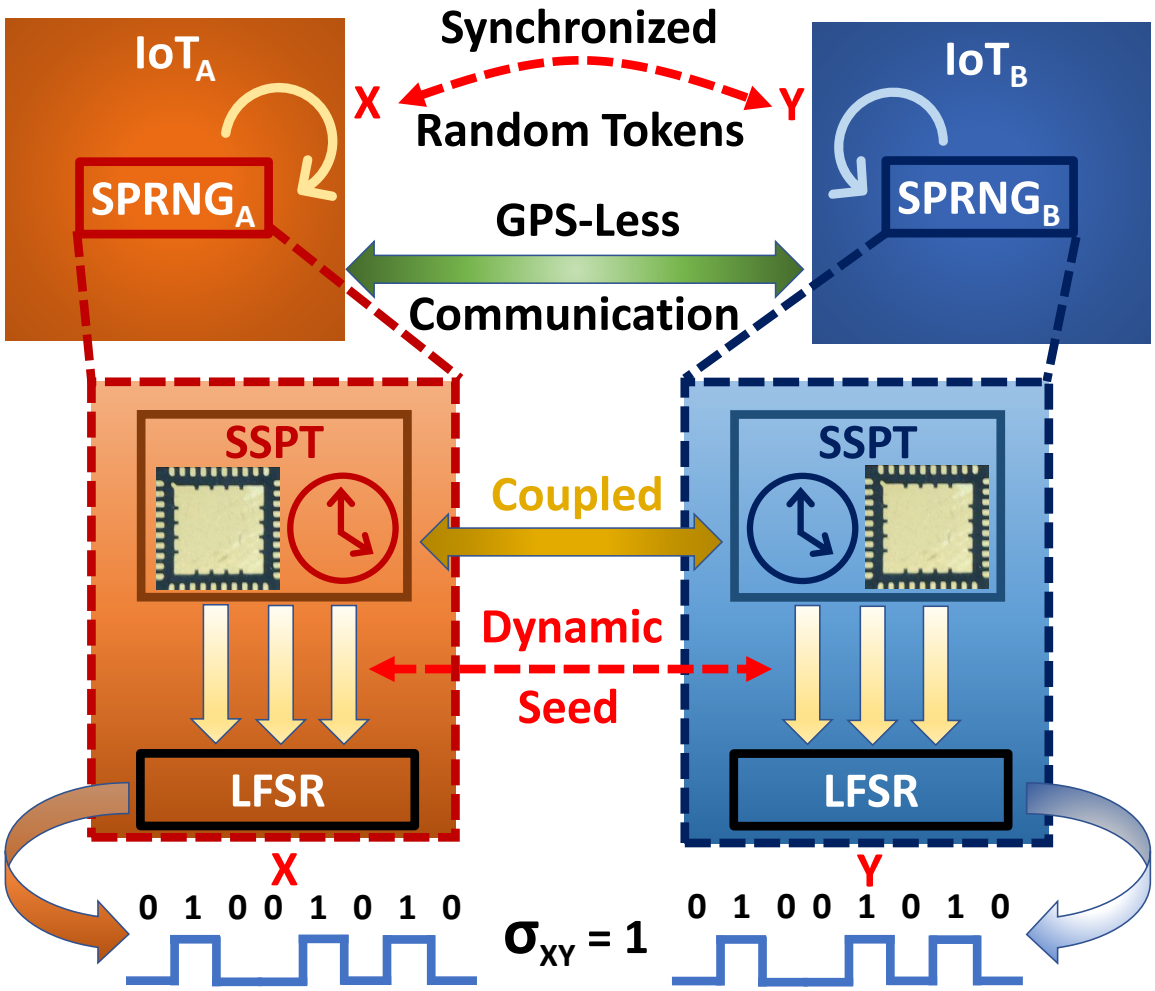

Figure 5.1: The concept of GPS-free secure Communication in spatially separated IoTs with SPRNGs: The IoTs generate random tokens using the SPRNG for use as cryptographic keys. The tokens are generated using a combination of a fast, low-complexity LFSR seeded by the Secure self-powered timers (SSPT). The synchronization of SSPT across both IoT_A and IoT_B ensures that the random tokens X and Y exhibit a perfect cross-correlation, $\sigma_{XY} = 1$

such as thermal noise and entropy of natural phenomena [116]. Even though TRNGs are preferred for cryptographic applications, they are generally expensive and might not produce random numbers fast enough to be suitable for use in resource-constrained IoTs [59]. On the other hand, a PRNG algorithm generates a sequence of numbers that is not truly random but whose statistical properties match that of a random number. In literature, there are many different types of PRNG that have been proposed [17]. However, for resource-constrained IoTs, the preferable PRNG is the one that is computationally inexpensive, fast, and can

be easily fabricated and integrated into a System-on-Chip (SoC). In this regard, a Linear-Feedback-Shift-Register (LFSR) architecture is an optimal choice [70]. It can be efficiently implemented using only flip-flops and XOR gates. An LFSR takes an initial value called seed as an input and generates each output bit with only a single shift operation, which satisfies both the low resource and the fast output requirement. However, one of the biggest challenges for an LFSR-based PRNG is the fact that the period is fixed and there is a need for reseeding to break the periodicity. While using a longer length LFSR or multiple LFSR would increase this periodicity, the problem still remains where once the period is reached the LFSR would start to repeat the random sequence. Furthermore, if the LFSR is seeded with a pre-stored static seed on boot-up, then it produces the same sequence of random numbers over and over again. One method to mitigate this issue would be to generate a dynamic seed. However, a resource-constrained IoT may not have access to a continuously running system clock or the GPS signal.

In addition to using a random number as a secure token, for secure communications there is also a need for synchronization of the tokens between the communicating parties. While asymmetric key encryption could be used to avoid this challenge, they are computationally too expensive to be universally implemented on these resource-constrained devices. On the other hand, a symmetric key encryption scheme can be customized for IoT platforms but requires a shared secret key [57]. Any static information stored on the IoT, such as a SecureID, used as the shared secret will be vulnerable to a machine learning type of attack [86]. Therefore, there is a need for a piece of dynamic information embedded into these IoT devices that can be synchronized in real-time. One such method for achieving this could be using a combination of a timing reference extracted from a global-positioning-system (GPS) and a timing reference generated locally using phased-locked oscillators. Unfortunately, in many IoT applications, this framework is impractical due to resource constraints together with the fact that many IoT devices may not have access to a GPS signal.

In this chapter, we describe an architecture of a synchronized-PRNG (SPRNG) that can be used for generating synchronized pseudo-random binary sequences without the need for any GPS reference signal. The SPRNG uses a combination of a fast, low-complexity LFSR-based PRNG and a slow but secure, synchronized seed generator based on our previously reported self-powered timers [137, 91, 104]. The self-powered timers use quantum-mechanical tunneling of electrons to operate without any external power and are practically secure against tampering, snooping, and side-channel attacks (both power and electromagnetic). In

this work, we explore different protocols to periodically and securely generate synchronized random bits by seeding the LFSR using an array of self-powered timers. The concept is illustrated in Fig. 5.1 in the context of IoT communications. The spatially separated IoT devices, IoT_A and IoT_B integrate a copy of the SPRNG for generating random tokens. The self-powered timers in these SPRNG form a clone where their temporal dynamics remain synchronized for long-period of time. When these synchronized-self-powered timers (SSPT) are used to dynamically seed the LFSR the random tokens generated by the LFSRs X and Y are precisely correlated. Therefore, these tokens can then be used as a shared secret key for facilitating secure communications between the IoTs. Furthermore, between power-ups, cold reboots, brown-outs, and system black-outs, the random keys generated using the approach shown in Fig. 5.1 remain unique and aperiodic, which obviates the possibility of replay attacks.

5.2 Results

5.2.1 Secure Self-powered Timers and Spatial Synchronization

Fig 5.1 (A) shows the micrograph of an array of self-powered timers along with the programming and readout circuit fabricated in a standard silicon process. A simplified equivalent circuit model for each of the timers on the fabricated prototype is shown in Fig 5.2 (B). The operating principle of the timers involves injecting charge on an electrically isolated floating-gate capacitor C_{fg}. This is achieved by using a combination of hot-electron injection or quantum mechanical tunneling which are described in the Method Section 5.3.1. After the initial programming, the charge on C_{fg} is allowed to leak through the dielectric barrier and is governed by the physics of Fowler-Nordheim (FN) quantum tunneling. Here the leakage current is denoted as J_{FN}. Note that the leakage process is thermodynamically and quantum-mechanically driven and hence does not require any external powering. This self-powered operation makes the timers immune to any power side-channel attack. Furthermore, J_{FN} is typically below attoamperes (or 10^{-18}A) which does not produce any measurable electromagnetic (EM) trace or fingerprint. Thus, the timers are practically immune to EM side-channel attacks. Furthermore, once the dynamics of the timers reach an *equilibrium* condition, any external probing using an EM source or using physical delamination disturbs the equilibrium and hence destroys the state of the timers. This implies that the self-powered

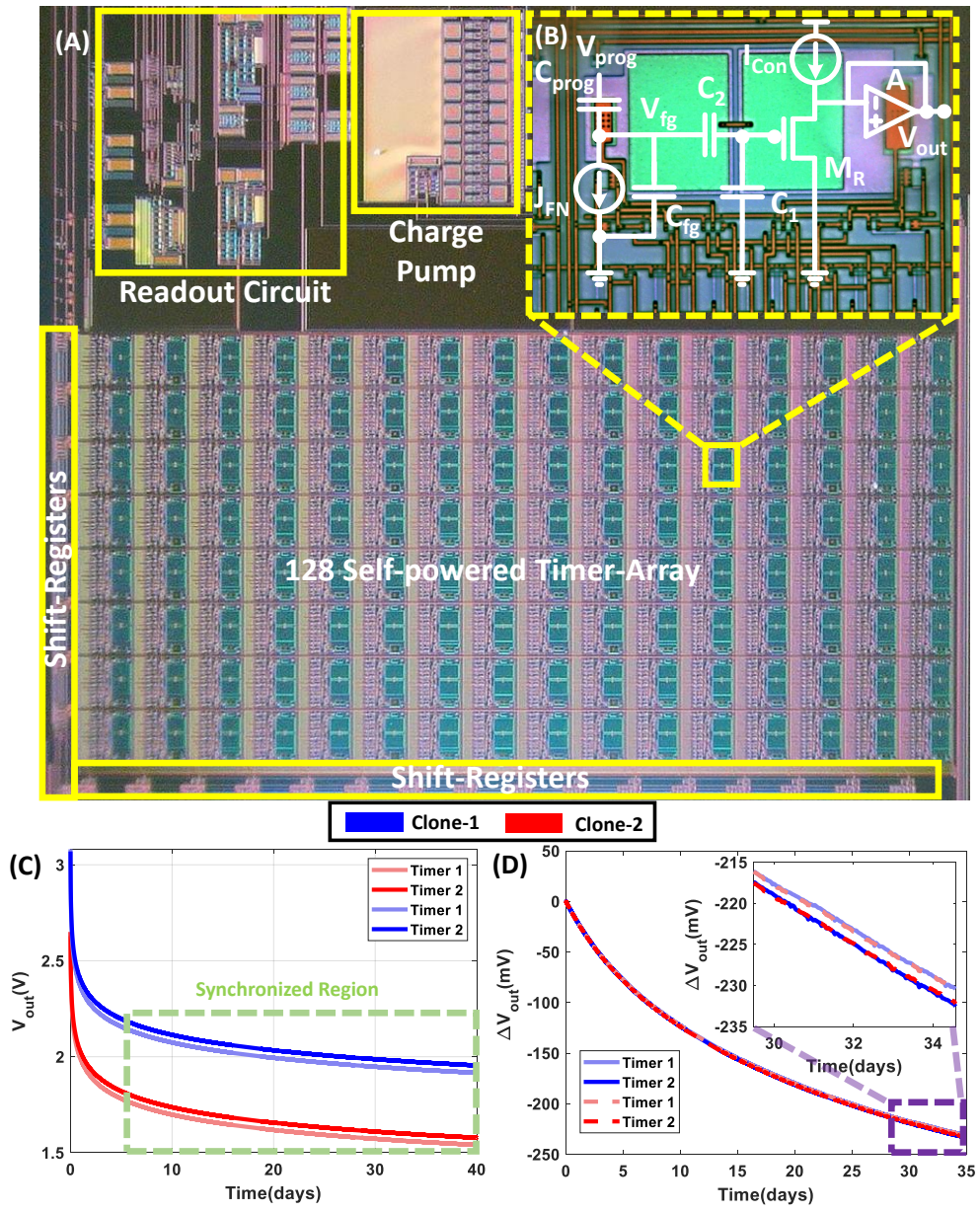

Figure 5.2: Measured dynamics and synchronization results using fabricated SSPT array:(A) Micrograph of an array of self-powered timers along with programming and readout circuit fabricated in a standard silicon process and (B) shows the micrograph of a single FN-timer in that array along with the equivalent circuit which is the building block of the SSPT. (C) The temporal response of the timers with different form factors on two different clones enters into the synchronized region after an initial settling stage. (D) Timers with the same form factors in the synchronized region have the same change in the output voltage over a long period of time across multiple clones.

timers are not only tamper-resistant but can only be copied through well-defined read-out mechanisms. Thus, we can assume that an array of FN-timers forms a secure dynamical system whose internal states could provide a secure mechanism for generating dynamic seeds for an LFSR.

In addition to its security features, FN-timers exhibit a unique synchronization feature where a pair of timers can be synchronized with each other, even if the devices are integrated on two different, spatially separated chipsets. A 'pair' of timers is defined as two timers designed with similar form factors. The synchronization feature is demonstrated by the experimental results in Fig. 5.2 (C) and (D) where we show the dynamics of two pairs of timers integrated on different chipsets that are spatially separated. Initially, the timers discharge quickly and the synchronization between different temporal dynamics is determined by device mismatch. However, as shown in Fig. 5.2 (C), after a period of 5 days the temporal responses become "practically" independent of device mismatch and hence become synchronized to each other. This is shown in Fig 5.2 (D) where after entering the *equilibrium* region, the dynamics of timer pairs remain synchronized. In our previous work [138] we have shown that the timer pairs can maintain synchronization for a duration greater than a year. This implies that if we can derive the LFSR seed from the temporal response of the timers, then all IoT devices integrated with the SSPT can securely generate synchronized random numbers.

5.2.2 Secure Seed exchange protocol

In order to use the synchronized random numbers as a cryptographic key for secure communication, the two IoT devices followed a simple protocol to synchronize their seeds. The seed generation protocol is described below:

IoT_A initiates the exchange protocol and generates a seed based on the digitized output of a set of timers. Information regarding the set of timers used by IoT_A is sent over an insecure public channel to IoT_B. On receiving this information IoT_B also generates a seed on its own. Once both seeds are generated the two IoTs can begin generating random numbers at higher-speed using an LFSR and start communicating using the synchronized random numbers as the encryption key.

SSPT Seed Exchange Protocol Steps that IoT_A and IoT_B follows to obtain common encryption key K_E

1: IoT_A: Selects a set of 'N' timers to be sampled for generating the seed.
2: IoT_A: Measures the output of these timers to generate seed $S_A = \{V_{out}^i\}^N$, where $1 < i < N_T$ is the index of the timer, N_T is the total number of timer on the chip and V_{out} is the digitized output of the timers.
3: $IoT_A \rightarrow IoT_B$: Sends the indexes of the timer $\mathbf{I} = \{i\}^N$ along with the order of sampling.

4: IoT_B: Measures the output of all the timers in $\mathbf{I}$ in the specified order to generate $S_B = \{V_{out}^j\}^N$, where $j \in \mathbf{I}$ and V_{out} is the digitized output of the timers.
5: IoT_A, IoT_B: Generate random numbers based on the seed from the timers, $K_A = PRNG(S_A)$ and $K_B = PRNG(S_B)$. Here $PRNG()$ denotes the output of an LFSR seeded by S_A, S_B. K_A and K_B can then be used to encrypt and decrypt further communication.
Since the same set of timers $\mathbf{I}$, which are synchronized, is used for generating both S_A and S_B, therefore IoT_A and IoT_B have a common encryption key $K_E = K_A = K_B$.

Only Step 3 in the seed exchange protocol is assumed to be vulnerable as the communication is performed over a presumably insecure channel where an adversary can eavesdrop and learn this information. However, note that in order to derive the encryption key K_E the adversary needs to have access to one of the timer clones at the time of communication. However, by construction, only Iot_A and IoT_B have access to one of the clones and the adversary cannot clone or copy the timers (one of the security properties of the FN-timers). This means that the adversary cannot sample the hardware timers to generate a seed. In addition to this, we have also discussed in the previous section 5.2.1 how the hardware timers are immune to any side-channel and snooping attacks. Thereby, it is also not possible for an adversary to deduce any information about the timers' output and generate the seed without actually sampling a clone. Note here that the actual output of the timers is also not accessible, only the random numbers from the LFSR are made attainable. This protects the seed exchange protocol from any kind of regression or Machine learning attack for predicting the output of the timers. In our previous work [106] we show that the parameters determining the temporal response of the timers cannot be determined by an adversary without the knowledge of the initialization condition. Furthermore, since the seed is derived from a dynamic process, it will change with time thereby breaking the period of the LFSR. In this regard, the output of the LFSR will appear to be a 'true' random number for any adversary.

5.2.3 Noise Robustness of Seed exchange protocol

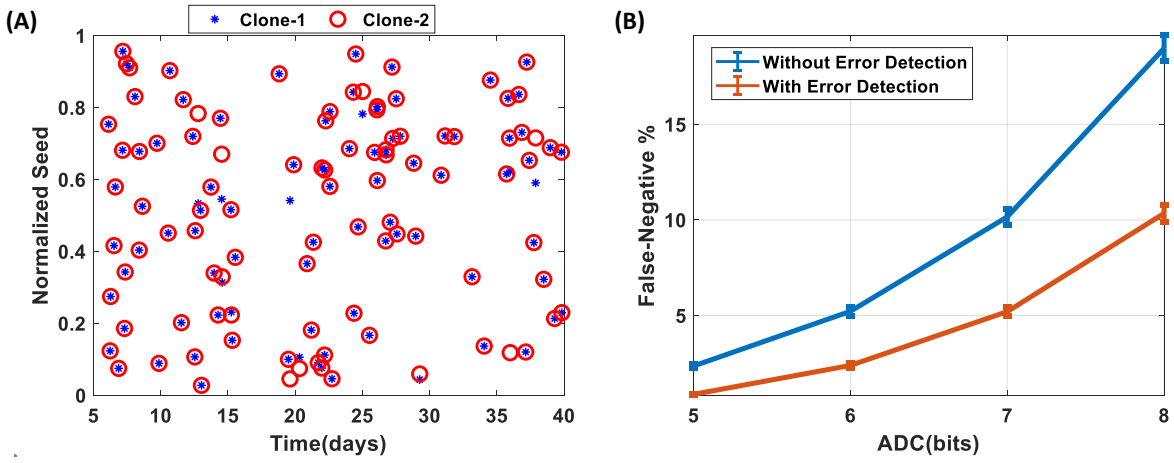

Figure 5.3: Random Seed Generation and Synchronization from fabricated prototypes: (A) The normalized seed generated from the SSPT at different times are spread evenly across the domain. (B) The percentage of False-Negative in the synchronization of two valid seeds due to readout and quantization error.

In the next set of experiments, we quantified the noise robustness of the protocol using a fabricated FN-timer array. We generated seeds from two fabricated prototypes of SSPT using the same set of timers. The details of the experiment are provided in the Method Section 5.3.2. Fig 5.3(A) shows the normalized seeds generated from both clones sampled at different time instances. We can observe that the seeds derived from the temporal response of the timers are uniformly distributed across the whole dynamic range with time. This implies that the seeds are unpredictable without knowledge of the underlying principle, timers' output. However, we do observe that there are a few mismatches among the seeds from the two clones. This is due to the readout and quantization noise of the analog-to-digital converter (ADC). To mitigate this issue a lower-resolution ADC can be used. In order to find out the expected number of mismatches at different resolutions of ADC we performed a Monte Carlo study where we generated seeds at random time instances with $5, 6, 7$, and 8 bits ADC (details in Method Section 5.3.2). Fig 5.3 (B) shows that as we decrease the resolution of the ADC, the percentage of mismatches between valid seeds i.e. False-Negatives, also decreases. However, this comes at a cost to the security of the protocol. This is because using lower-resolution ADC would result in less frequent changes in the value of the seed and might not be enough to break the period of the LFSR. Therefore, a tradeoff exists between the security and

robustness of the protocol. Another method that could be used to reduce the possibility of False-negatives is by using error correcting code such as Cyclic-Redundancy-Check (CRC). Even with a simple CRC code of size 3 bits detecting at least 2 bits hamming distance between the two seeds the percentage of False-Negatives can be reduced at all resolutions of ADC as shown in Fig 5.3 (B). The details of this experiment are provided in the Method Section 5.3.2. Note that, this improvement in accuracy comes at a cost of more computational resources required for the protocol. Thereby the usage of such methods would depend on the application and resource availability of the IoT device in question and the demand for accuracy.

5.2.4 Statistical Test for SPRNG

In order to evaluate the randomness of the numbers generated by SPRNG we performed benchmark tests using the Statistical Randomness Test Suite (SP800-22 Rev 1a) made available by the National Institute of Standards and Technology (NIST) [13]. The suite consists of 15 statistical tests the results of which are represented in the form of P-values in the range [0, 1]. A binary string is tested to be random if the P-value exceeds a certain threshold value in all 15 tests. This threshold value was chosen to be 0.01, as recommended by the NIST specification, which suggests that the string is random with a probability of 99%. The details of the experiment are provided in the Method Section 5.3.3 and the results of all 15 tests are tabulated in Table 1.

The first experiment was done with a single LFSR as the random number generator seeded with the digitized output of the timers. We observe that the minimum pass rate is approximately 91 for a sample of 100 binary strings, in the case of the Linear Complexity Test and Random Excursion Test. These results could be further improved by using two independent LFSRs of different sizes, randomly seeded by the SSPT, and then XORing the output of them to generate the random binary strings. In this case, the minimum pass rate is 96 out of a sample of 100 binary strings. Note here that this technique not only improves the quality of random numbers generated but also increases the periodicity of the overall sequences. This would ultimately increase the lifetime of the SSPT as discussed in the following section. However, this comes at a cost of the efficiency of the SPRNG as more measurements and computations are needed to be done. Therefore, a tradeoff exists between efficiency and lifetime and security. Depending on the application (how long the IoT will be in use) and

NIST TESTs	Single LFSR		XORed LFSRs	
	Average P-value	Pass Ratio	Average P-value	Pass Ratio
Monobit Test	0.46	100/100	0.56	100/100
Frequency within block Test	0.41	98/100	0.52	98/100
Runs Test	0.45	100/100	0.54	100/100
Longest run ones in a block Test	0.55	100/100	0.37	100/100
Binary Matrix Rank Test	0.54	100/100	0.48	100/100
DFT Test	0.51	100/100	0.49	100/100
Non-overlapping Template Matching Test	1	100/100	1	100/100
Overlapping Template Matching Test	0.53	100/100	0.49	100/100
Maurers Universal Test	0.49	100/100	0.55	100/100
Linear Complexity Test	0.28	91/100	0.47	100/100
Serial Test	0.35	97/100	0.44	100/100
Approximate Entropy Test	0.42	100/100	0.56	100/100
Cumulative Sums Test	0.39	100/100	0.4	100/100
Random Excursion Test	0.1	91/100	0.16	96/100
Random Excursion Variant Test	0.13	93/100	0.13	96/100

Table 5.1: Results showing the randomness of the numbers generated by SPRNG when tested with NIST test suites for both cases, a single LFSR and when two LFSR are XORed.

specification (how secure the communication needs to be) of the IoT device either single-LFSR or double-LFSR implementation of SPRNG can be used. Nevertheless, from the results in Table 1, we can definitely conclude that the bit strings generated by the SPRNG, both with single and double LFSRs implementation, are statistically random in nature.

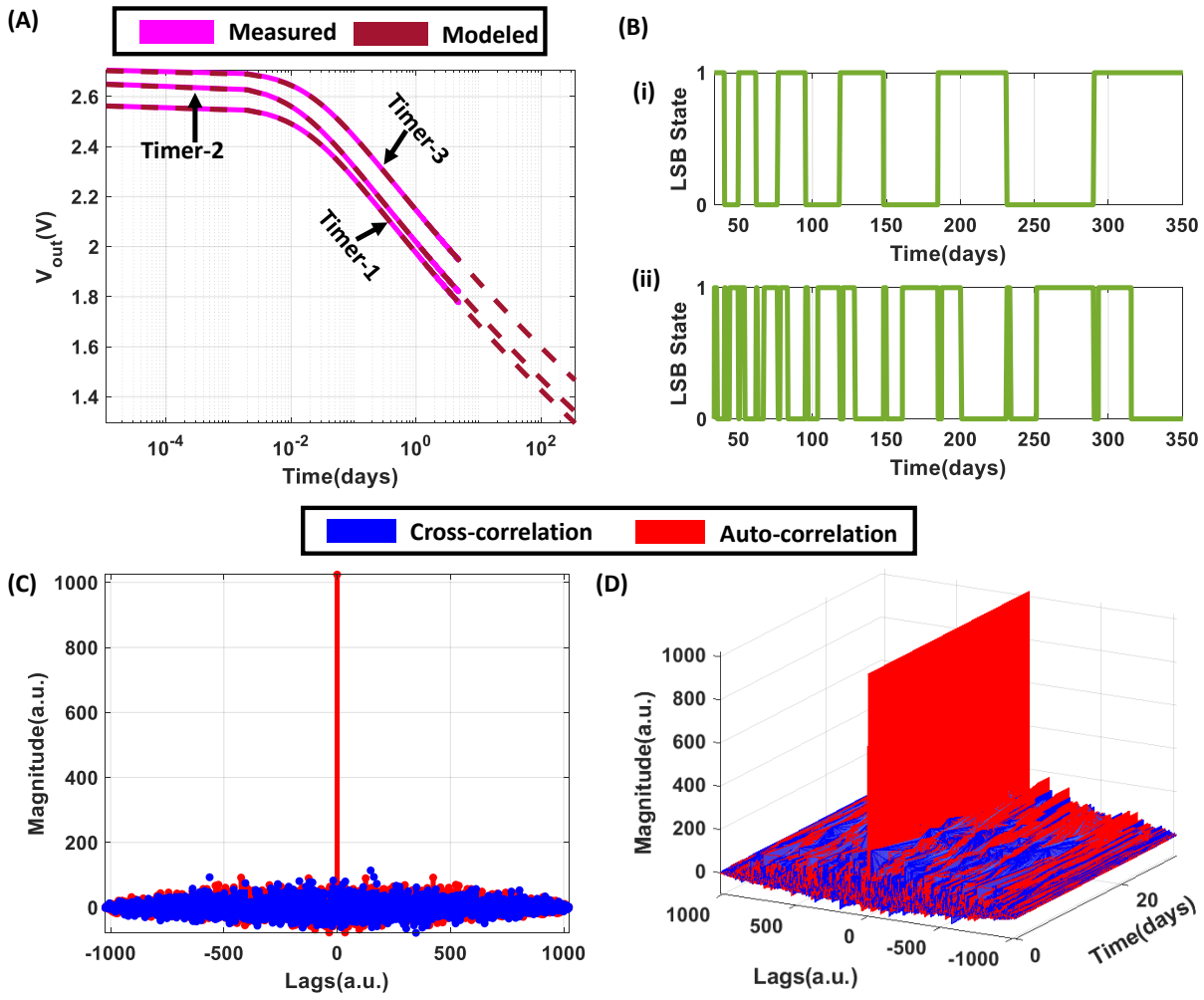

Figure 5.4: Correlation between the outputs of LFSR for shifted seed: (A) Accurate representation of the fabricated timers by its analytical modeled counterpart. (B) The state of the LSB for the digitized output of the fabricated (i) Timer-1 in (A) and (ii) Timer-1, 2, and 3 in (A) XORed. m(C) Autocorrelation for the output of LFSR for a particular seed along with the cross-correlation with another output generated at the same time instance with the order of seed generation changed. (D) The results when the procedure in (C) is repeated across multiple different time instances.

5.2.5 SSPT Lifetime Analysis

In our previous work [137, 138] we have shown that the temporal response of the fabricated timers can be modeled as

$$V_{out}(t) = \frac{k_2}{\log(k_1 t + k_0)} \tag{5.1}$$

where k_1, k_2 are device specific and fabrication specific parameters, $k_0 = \exp\left(\frac{k_2}{V_0}\right)$, V_0 refers to the initial voltage at the floating-gate, and t refers to the time elapsed after initialization. The detailed derivation is excluded here for the sake of brevity and can be found in [137, 138]. In Fig 5.4 (A) we can observe that this analytical model can accurately track the output of the fabricated timers once the parameters are regressed from the measured data. The details of the regression process are provided in the Method Section 5.3.4. We use the analytical model to determine the lifetime of the SSPT. Note that since the timers are initialized with a fixed amount of charge, the dynamics of timers will slow down to single-electron tunneling events. The question being analyzed here is whether an ensemble of FN-timers can still exhibit state-change that is faster than the period of the LFSR. Fig 5.4 (B) (i) shows the state of the LSB of the digitized output for a single timer. Note here that in order for the dynamic seed to change only a single-bit flip of the digitized output would suffice. Therefore, the change in the LSB state represents the change in the dynamic seed. Since the dynamical system slows down non-linearly as time passes the rate of change of the dynamic seed will also decrease over time, as evident from Fig 5.4 (B) (i). However, to break the periodicity of the LFSR the dynamic seed needs to change before we generate the maximum length of a random number. For example, let an LFSR generate random numbers with a clock speed of 1 GHz. Then the LFSR with seed length of 49 bits will generate the maximum length of a random number in $\frac{2^{49}}{10^9}$s, which is ≈ 6.5 days. Now, if a single timer was used to generate the seed in this case, then from Fig 5.4 (B) (i) we can derive that after a period of ≈ 60 days, the dynamic seed no longer breaks the periodicity of the LFSR. We denote this period as the 'lifetime' of the SPRNG, as after this period the random sequence will start to repeat itself. The lifetime of the SPRNG can be increased by increasing the resolution of the ADC used for digitizing the output since this would change the seed more frequently. However, as shown previously, this would come at the cost of the seed exchange accuracy of the protocol. Another method to increase the lifetime would be to use multiple timers for generating the dynamic seed. This is because as the number of timers used for generating the seed increases, the probability that at least one of the timer's digitized output changes also increases. This

can be observed from Fig 5.4 (B) (ii) where three timers were used to generate the dynamic seed which subsequently increases the lifetime of SPRNG. Furthermore, while using multiple timers, the order in which the digitized output of the timers is sequenced can also be changed to break the periodicity of the LFSR. This can be observed in Fig 5.4 (C) which shows the correlation between two random numbers generated with the same set of timers sampled at the same instance, but only the order of sequencing their digitized output to generate the seed is changed. In order to obtain a reference for the noise floor we also calculated the autocorrelation of one of the random numbers. Fig 5.4 (C) shows that when the lag in the case of autocorrelation is zero, the correlation is at maximum. For any other lag, the autocorrelation is 5 times less than that of the maximum magnitude meaning that there is hardly any similarity or periodicity in the random number itself. This is trivial for a random number as there should not be any correlation between two blocks of sequences within the same number. Next, we observe that the magnitude of the cross-correlation between the two random numbers is also within this range. Therefore, we can conclude that changing the order of the timers' sequence while generating the dynamic seed will also break the periodicity of the LFSR. Fig 5.4 (D) shows that this is true across all time instances.

5.3 Methods

5.3.1 Programming The SSPT

The programming of the SSPT requires injecting charges on the electrically isolated floating gates such that the floating-gate potential (V_{fg} in Fig 5.2 (B)) can be set to a level at which FN-tunneling is measurable. This is accomplished by setting V_{prog} to a high-potential of 22V using an internal (on-chip) charge-pump. After the initial programming, the floating-gate is allowed to discharge while the potential V_{out} is measured periodically. The measurement is performed using a capacitive voltage divider formed by C_1 and C_2 such that the attenuated voltage can be measured using standard readout buffers. Furthermore, since the tunneling nodes are electrically isolated we use a readout MOSFET M_r configured as a source-follower using a constant current-source I_r to read the voltage V_{fg} at C_1. The voltage of the source follower is buffered using A to avoid any coupling to the tunneling junction. The readout voltage was programmed to around 3V during the initialization of the tunneling node. For the results shown in Fig 5.2 (C) and (D), each individual timer cell, shown in Fig 5.2 (A),

on two separate timer pairs having the same configuration was initialized independently to a high FN-tunneling region. After the one-time programming, V_{prog} was set to 0V and the timers discharged naturally. In this mode of operation, no external power is required. The readout voltages of each timer cell on both the clones were measured every 180s for a duration of over 40 days.

5.3.2 Seed Generation

The measured outputs from the two hardware clones were used to generate the seed at each time instance. The analog readout voltages V_{out} for each timer cell were quantized using an ADC to generate a binary string. These binary strings of multiple timers were concatenated to generate seeds of variable length depending on the size of the LFSR used. For the results shown in Fig 5.3 (A) 7 timers were used from each clone to generate the seed with their outputs quantized to 7 bits precision. This resulted in a seed of length 49 bits. The measured outputs were sampled randomly at 100 different time instances and the generated seeds were normalized for visual comparison.

For results shown in Fig 5.3 (B) we generated 1000 seeds at random time instances using the same procedure as discussed above where measured outputs were quantized with $5, 6, 7,$ and 8 bits precision. Each instance of sampling where the seeds from both hardware clones did not match perfectly was counted as a False-negative. The experiment was repeated 1000 times, each time the sampling and seed generation was performed on a different set of random time instances. This represented the case where no error detection was used. In the case of error detection, the digitized seeds are represented as the coefficients of a message polynomial which is then divided by a pre-determined generator polynomial to calculate the CRC bits i.e. the coefficient of the remainder polynomial. In the seed exchange protocol at Step-3, IoT_A sends the CRC bits along with the other information. IoT_B can use these CRC bits to check whether the seed that it generated, S_B, matches with that of IoT_A. For a generator polynomial of size 3-bits, IoT_B can detect at least 2-bits of error [71]. Therefore in Fig 5.3 (B), seeds from two clones with a hamming distance of 2 or less were not counted as False-negative. The mean and variance of the percentage of False-negative were calculated across all experiments.

5.3.3 Randomness Test

The seeds for the LFSR were generated using the measured output from the hardware clones as described in the previous sections. These seeds were then fed into an LFSR which was simulated in MATLAB. In the case of the Single LFSR, shown in Table 5.1, the length of the LFSR chosen was 49 bits, which means 7 timers' output was used each quantized with an ADC of 7 bit precision. The seeds were generated across 100 random time instances and corresponding to each seed 1 MiB ($2^{20} bits$) were simulated from the LFSR. These bit strings were then tested with the NIST SP800-22 Rev 1a PRNG test suite using the Python implementation by David Johnston [64]. This process was repeated for the XORed LFSR, however this time two sets of seeds were generated at each time instance. The length of one of the LFSRs used in this case was 49 bits, the same as before, and the other one was 42 bits. To generate the seeds 7 and 6 timers' outputs were used respectively with a 7 bit precision ADC. The individual outputs of the LFSRs, 1 MiB, were then XORed with each other for producing the random bits which were then tested in a similar manner as before.

5.3.4 Extending SSPT lifetime through shifted seed generation

The measured output shown in Fig 5.2 (C) was used to regress the parameters k_0, k_1 and k_2 as shown in equation 5.1 for each timer in the fabricated prototype. Even though each timer's output was measured for a period of $\approx$ 40 days, only the data for the first 5 days were used to regress the parameter. In this manner, we could verify that the regressed parameters, when used to represent the measured results, accurately predicted the temporal response of each timer against the measured result for the rest of the 35 days. This is validated in Fig 5.4 (A).

Each of the timer cells in the fabricated prototype can be selected for reading out the output values using a serial shift-register. However, depending on the order of read-out of these timer cells the seed that is generated from the quantization of their output is different for every permutation. This means that with the same set of timers multiple seeds can be generated. For the results shown in Fig 5.4 (C), two sets of seeds were generated at the same time instance using the same set of 7 timers with only the order of the timers shifted by one in a cyclic manner. For example, if one of the seeds was generated using the order $[19, 45, 54, 61, 89, 119, 120]$, then the order for the other seed was $[120, 19, 45, 54, 61, 89, 119]$.

These seeds were then used by the same LFSR (length 49 bits) and 1 KiB of random binary strings were generated. The auto-correlation of one of the binary strings was calculated along with the cross-correlation with the other binary strings. Note here that for these calculations the binary states were represented as $[-1, 1]$ instead of $[0, 1]$. This process was repeated across 1000 random time instances, the result for which is shown in Fig 5.4 (D).

Discussion

In this chapter, we described an architecture of a lightweight synchronized pseudo-random-number generator (SPRNG) that can be used for securing wireless communications in IoTs. The solution does not require access to GPS and therefore could be used in many resource-constrained and adversarial environments. Some of the applications include personal IoTs used in health-care [12], key-fobs [40] to military-grade IoTs that need to operate in RF-jamming environments [114]. The combination of ultra-secure slow-dynamics exhibited by the FN-timers and fast-dynamics exhibited by standard LFSR provides an ultra-fast and yet secure mechanism to generate secure tokens that could potentially be used for high-speed transactions [135]. However, note that for this application, clock-frequency and clock-phase synchronization between the communicating devices are required and have not been addressed in the work. The inherent security of the proposed approach lies in the *no-cloning* property of the FN-timers, therefore, only the communicating IoTs will have access to the secure random tokens. During each communication session, and even after a cold reboot the tokens are randomly generated and hence an adversary cannot initiate a replay attack.

A potential limitation of SPRNG proposed in this work in cryptographic applications arises due to the usage of an LFSR as the PRNG. If an adversary manages to extract $2L$ bits of the LFSR output, where L is the length of the dynamic seed, then by using Berlekamp–Massey algorithm [88] they can represent the LFSR in an analytic form. This significantly reduces the lifetime of SPRNG since the LFSR are now needed to be seeded much more frequently. One method to achieve this would be to use a different set of timers to dynamically seed the LFSR every $2L$ bits. Another method to mitigate this issue would be to use an Alternating Step Generator (ASG) proposed by C. G. Günther [53] where three LFSRs are used in conjunction to produce the random sequences. Note that in this implementation all three LFSRs would be dynamically seeded by three different sets of timers and the synchronization between the

two random tokens on spatially separated devices can still be achieved. The best possible attack in this scenario that can be mounted will require $\mathcal{O}(2^{\frac{2L}{3}})$ bits [67]. This practically ensures that the lifetime analysis in the Results Section 5.2.5 remains valid.

One consideration that has not been discussed before in the chapter is the effect of environmental variations (for example temperature) on the synchronization of the FN-timers. In [138] we reported that the dynamics of FN-timers exhibit a temperature dependence, however, when the temperature remains static, the dynamics of the FN-timer still remain synchronized with respect to each other. Therefore, one of the key requirements for the proposed SPRNG-based secure communication is to ensure proper temperature controls. However, this feature could also be used to further enhance security where the operating temperature could be treated as private information that is only known to the communicating parties.

Future work in this area would require developing a complete system-on-chip solution where the SPRNG acts as a core trusted-platform-module (TPM) that like the commercial AES core in secure processors can generate tokens for use by the rest of the SoC modules.

Chapter 6

Conclusion

6.1 Concluding Remarks

In my thesis, I designed and fabricated synaptic memory element, FN-synapse, that exploits the desynchronization between two dynamical systems to implement an analog memory. I showed that when the dynamical system is based on FN quantum-tunneling the synaptic device can exhibit near-optimal memory consolidation that has been previously demonstrated using only algorithmic models. However, unlike its algorithmic counterparts like the cascade or EWC models, the FN-Synapse doesn't require any additional computational or storage resources. FN-synapse exploits the physics of the device itself for synaptic intelligence and for continual learning. I have also shown that the physical dynamics of FN-synapse can be matched to the dynamics of weight updates used in ML or neural network training. The plasticity of FN-DAM during the training phase can be traded off with the energy-required to update the weights. By exploiting these characteristics I have shown that energy efficiency during the training phase of a neural network can be increased by orders of magnitude.

In addition to my work in the AI domain, I have also explored the application of a dynamical information storage device in the cryptographic security domain in my thesis. Here, I exploited the synchronizing capability of the FN-dynamical system to design a symmetric key exchange protocol. I have shown that the protocol is secure even with the advent of a quantum computer since the security of the protocol lies in the hardware and using time itself as a one-way function. Furthermore, using the self-powered nature of the dynamical system I show that the hardware is practically secure against any side-channel adversarial attack. I have also designed a synchronized PRNG with the FN-dynamical system at the heart of it.

6.2 Future Direction

The FN-dynamical system-based memory elements provide a number of interesting avenues that can be explored in the future:

• The memory retention time for FN-synapse varies with both the initial memory imprinted on it and also the stage at which it is biased. Therefore, the information written on it can be erased in a controlled manner based on the initialization condition of the memory. This can be exploited to design self-erasing memory for sensitive information such as medical data.

• The FN-dynamical system is affected by surrounding environmental conditions such as temperature and RF-signal. This means that the information stored in the FN-synapse also gets destroyed due to variations in the surrounding. This feature can be used to design a geofencing framework that does not require the use of GPS signals.

• The weight update process of FN-synapse is determined by the number of electrons tunneling in a certain period of time and is quantized. As a result, the weight update process is stochastic in nature which can be used to mimic the stochastic nature of biological synapses.

• Another aspect to focus on would be to prototype a self-powered timer system-on-chip with all the basic hardware security primitives and validate the SPoTKD protocol under real-world conditions and over different distribution channels. This will open the possibility of applying SPoTKD in areas such as quantum secure blockchains (based on symmetric-key) and electronic voting.

References

[1] U. 42. 2020 unit 42 iot threat report. https://start.paloaltonetworks.com/unit-42-iot-threat-report, 2020.

[2] R. K.-A. A. Sebastian, M. Le Gallo and E. Eleftheriou. Memory devices and applications for in-memory computing. *Nature nanotechnology*, 15(7):529–544, 2020.

[3] W. C. Abraham. Metaplasticity: tuning synapses and networks for plasticity. *Nature Reviews Neuroscience*, 9(5):387–387, 2008.

[4] W. C. Abraham and M. F. Bear. Metaplasticity: the plasticity of synaptic plasticity. *Trends in neurosciences*, 19(4):126–130, 1996.

[5] M. H. Afifi, L. Zhou, S. Chakrabartty, and J. Ren. Dynamic authentication protocol using self-powered timers for passive internet of things. *IEEE Internet of Things Journal*, 5(4):2927–2935, 2018.

[6] H. Akinaga and H. Shima. Resistive random access memory (reram) based on metal oxides. *Proceedings of the IEEE*, 98(12):2237–2251, 2010.

[7] M. Alimomeni, R. Safavi-Naini, and S. Sharifian. A true random generator using human gameplay. In S. K. Das, C. Nita-Rotaru, and M. Kantarcioglu, editors, *Decision and Game Theory for Security*, pages 10–28, Cham, 2013. Springer International Publishing.

[8] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. *CoRR*, abs/1711.09601, 2017.

[9] R. Alléaume, C. Branciard, J. Bouda, T. Debuisschert, M. Dianati, N. Gisin, M. Godfrey, P. Grangier, T. Länger, N. Lütkenhaus, et al. Using quantum key distribution for cryptographic purposes: a survey. *Theoretical Computer Science*, 560:62–81, 2014.

[10] D. J. Amit and S. Fusi. Learning in neural networks with material synapses. *Neural Computation*, 6(5):957–982, 1994.

[11] K. Aono, N. Lajnef, F. Faridazar, and S. Chakrabartty. Infrastructural health monitoring using self-powered internet-of-things. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2058–2061, 2016.

[12] S. B. Baker, W. Xiang, and I. Atkinson. Internet of things for smart healthcare: Technologies, challenges, and opportunities. *IEEE Access*, 5:26521–26544, 2017.

[13] L. E. Bassham, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, S. D. Leigh, M. Levenson, M. Vangel, D. L. Banks, N. A. Heckert, J. F. Dray, and S. Vo. Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, Gaithersburg, MD, USA, 2010.

[14] N. Beckmann and M. Potkonjak. Hardware-based public-key cryptography with public physically unclonable functions. In *International Workshop on Information Hiding*, pages 206–220. Springer, 2009.

[15] M. Benna and S. Fusi. Computational principles of synaptic memory consolidation. *Nature Neuroscience*, 19, 10 2016.

[16] C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. *In Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, 175:8, New York, 1984.

[17] K. Bhattacharjee and S. Das. A search for good pseudo-random number generators: Survey and empirical studies. *Computer Science Review*, 45:100471, 2022.

[18] C. M. Bishop. Pattern recognition and machine learning. *Springer*, 2006.

[19] G. Brassard, N. Lütkenhaus, T. Mor, and B. Sanders. Limitations on practical quantum cryptography. *Physical review letters*, 85:1330–3, 09 2000.

[20] V. H. Brun, K. Ytterbø, R. G. Morris, M.-B. Moser, and E. I. Moser. Retrograde amnesia for spatial memory induced by nmda receptor-mediated long-term potentiation. *Journal of Neuroscience*, 21(1):356–362, 2001.

[21] S. N. Burke and C. A. Barnes. Neural plasticity in the ageing brain. *Nature Reviews Neuroscience*, 7(1):30–40, 2006.

[22] G. W. Burr, M. J. BrightSky, A. Sebastian, H.-Y. Cheng, J.-Y. Wu, S. Kim, N. E. Sosa, N. Papandreou, H.-L. Lung, H. Pozidis, E. Eleftheriou, and C. H. Lam. Recent

progress in phase-change memory technology. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 6:146–162, 2016.

[23] G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola, L. L. Sanches, I. Boybat, M. L. Gallo, K. Moon, J. Woo, H. Hwang, and Y. Leblebici. Neuromorphic computing using non-volatile memory. *Advances in Physics: X*, 2(1):89–124, 2017.

[24] T. Calver, M. Grimaila, and J. Humphries. An empirical analysis of the cascade error reconciliation protocol for quantum key distribution. In *Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research*, CSIIRW '11, New York, NY, USA, 2011. Association for Computing Machinery.

[25] G. Cauwenberghs and M. Bayoumi. Learning on silicon: Adaptive vlsi neural systems. *Springer Science and Business Media*, 1999.

[26] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. *CoRR*, abs/1801.10112, 2018.

[27] L. Chen and K. Aihara. Chaotic simulated annealing by a neural network model with transient chaos. *Neural Networks*, 8(6):915–930, 1995.

[28] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam. Dadiannao: A machine-learning supercomputer. In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 609–622, 2014.

[29] Z. Chen, Q. Jin, J. Wang, Y. Wang, and K. Yang. Mc2-ram: An in-8t-sram computing macro featuring multi-bit charge-domain computing and adc-reduction weight encoding. In *2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6, 2021.

[30] J. Choi, S. Venkataramani, V. V. Srinivasan, K. Gopalakrishnan, Z. Wang, and P. Chuang. Accurate and efficient 2-bit quantized neural networks. *Proceedings of Machine Learning and Systems*, 1:348–359, 2019.

[31] J. Choi, Z. Wang, S. Venkataramani, P. I. Chuang, V. Srinivasan, and K. Gopalakrishnan. PACT: parameterized clipping activation for quantized neural networks. *CoRR*, abs/1805.06085, 2018.

[32] R. Courtland. Intel now packs 100 million transistors in each square millimeter. https://spectrum.ieee.org/nanoclast/semiconductors/processors/intel-now-packs-100-million-transistors-in-each-square-millimeter, 2017.

[33] Cryptopedia. An overview of decentralized cloud storage services. https://www.gemini.com/cryptopedia/crypto-cloud-storage-decentralized-cloud-storage-providers, December 21, 2021.

[34] B. Danev, H. Luecken, S. Capkun, and K. El Defrawy. Attacks on physical-layer identification. In *Proceedings of the third ACM conference on Wireless network security*, pages 89–98, 2010.

[35] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3):653–664, 2016.

[36] A. Di Falco, V. Mazzone, A. Cruz, and A. Fratalocchi. Perfect secrecy cryptography via mixing of chaotic waves in irreversible time-varying silicon chips. *Nature Communications*, 10, 12 2019.

[37] A. R. Dixon, Z. L. Yuan, J. F. Dynes, A. W. Sharpe, and A. J. Shields. Gigahertz decoy quantum key distribution with 1 mbit/s secure key rate. *Optics Express*, 16(23):18790, Oct 2008.

[38] Q. Dong, Z. Wang, J. Lim, Y. Zhang, M. E. Sinangil, Y.-C. Shih, Y.-D. Chih, J. Chang, D. Blaauw, and D. Sylvester. A 1-mb 28-nm 1t1mtj stt-mram with single-cap offset-cancelled sense amplifier and in situ self-write-termination. *IEEE Journal of Solid-State Circuits*, 54(1):231–239, 2019.

[39] S. Dünkel, M. Trentzsch, R. Richter, P. Moll, C. Fuchs, O. Gehring, M. Majer, S. Wittek, B. Müller, T. Melde, H. Mulaosmanovic, S. Slesazeck, S. Müller, J. Ocker, M. Noack, D.-A. Löhr, P. Polakowski, J. Müller, T. Mikolajick, J. Höntschel, B. Rice, J. Pellerin, and S. Beyer. A fefet based super-low-power ultra-fast embedded nvm technology for 22nm fdsoi and beyond. In *2017 IEEE International Electron Devices Meeting (IEDM)*, pages 19.7.1–19.7.4, 2017.

[40] M. Eddy. Is your car key fob vulnerable to this simple replay attack?, 2022.

[41] A. K. Ekert. Quantum cryptography based on bell's theorem. *Phys. Rev. Lett.*, 67:661–663, Aug 1991.

[42] R. A. FISHER. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[43] E. J. Fuller, S. T. Keene, A. Melianas, Z. Wang, S. Agarwal, Y. Li, Y. Tuchman, C. D. James, M. J. Marinella, J. J. Yang, et al. Parallel programming of an ionic floating-gate memory array for scalable neuromorphic computing. *Science*, 364(6440):570–574, 2019.

[44] S. Fusi. Hebbian spike-driven synaptic plasticity for learning patterns of mean firing rates. *Biological cybernetics*, 87(5):459–470, 2002.

[45] S. Fusi and L. Abbott. Limits on the memory storage capacity of bounded synapses. *Nature neuroscience*, 10(4):485–493, 2007.

[46] S. Fusi, P. J. Drew, and L. Abbott. Cascade models of synaptically stored memories. *Neuron*, 45(4):599–611, 2005.

[47] A. S. G. Srinivasan and K. Roy. Magnetic tunnel junction based long-term short-term stochastic synapse for a spiking neural network with on-chip stdp learning. *Scientific reports*, 6(1):1–3, 2016.

[48] C. Giotis, A. Serb, V. Manouras, S. Stathopoulos, and T. Prodromakis. Palimpsest memories stored in memristive synapses. *Science Advances*, 8(25):eabn7920, 2022.

[49] D. Giry. Bluekrypt:cryptographic key length recommendation. https://www.keylength.com/en/4/, 2020.

[50] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden. Quantum cryptography. *Reviews of Modern Physics*, 74(1):145–195, Mar 2002.

[51] M. J. Golino. *System and method of secure remote authentication of acquired data.* 20140281523, September 2014.

[52] D. Group. From stolen laptop to inside the company network. https://dolosgroup.io/blog/2021/7/9/from-stolen-laptop-to-inside-the-company-network, July 28, 2021.

[53] C. G. Günther. Alternating step generators controlled by de bruijn sequences. In D. Chaum and W. L. Price, editors, *Advances in Cryptology — EUROCRYPT' 87*, pages 5–14, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[54] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. Deep learning with limited numerical precision. *International conference on machine learning*, 37:1737–1746, 07-09 Jul 2015.

[55] G. Halfacree. Onchip unveils itsy-chipsy ultra-low-cost ic fabrication platform. https://abopen.com/news/onchip-unveils-itsy-chipsy-ultra-low-cost-ic-fabrication-platform/, 2017.

[56] W. He, H. Sun, Y. Zhou, K. Lu, K. Xue, and X. Miao. Customized binary and multi-level hfo2- x-based memristors tuned by oxidation conditions. *Scientific reports*, 7(1):1–9, 2017.

[57] M. S. Henriques and N. K. Vernekar. Using symmetric and asymmetric cryptography to secure communication between devices in iot. In *2017 International Conference on IoT and Application (ICIOT)*, pages 1–4, 2017.

[58] Y.-C. Hsu, Y.-C. Liu, A. Ramasamy, and Z. Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines, 2018.

[59] J.-C. Hsueh and V. H.-C. Chen. An ultra-low voltage chaos-based true random number generator for iot applications. *Microelectronics Journal*, 87:55–64, 2019.

[60] S. R. Hulme, O. D. Jones, C. R. Raymond, P. Sah, and W. C. Abraham. Mechanisms of heterosynaptic metaplasticity. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1633):20130148, 2014.

[61] D. Ielmini and H.-S. P. Wong. In-memory computing with resistive switching devices. *Nature Electronics*, 1(6):333–343, 2018.

[62] N. Jain, C. Wittmann, L. Lydersen, C. Wiechers, D. Elser, C. Marquardt, V. Makarov, and G. Leuchs. Device calibration impacts security of quantum key distribution. *Physical review letters*, 107:110501, 09 2011.

[63] J. R. Jameson, N. Gilbert, F. Koushan, J. Saenz, J. Wang, S. Hollmer, M. Kozicki, and N. Derhacobian. Quantized conductance in $Ag/GeS_2/W$ conductive-bridge memory cells. *IEEE Electron Device Letters*, 33(2):257–259, 2012.

[64] D. Johnston. sp800_22_tests, 2021.

[65] G. Karunaratne, M. Le Gallo, G. Cherubini, L. Benini, A. Rahimi, and A. Sebastian. In-memory hyperdimensional computing. *Nature Electronics*, 3(6):327–337, 2020.

[66] L. Keuninckx, M. Soriano, I. Fischer, C. Mirasso, R. Nguimdo, and G. Van der Sande. Encryption key distribution via chaos synchronization. *Scientific Reports*, 7:43428, 02 2017.

[67] S. Khazaei, S. Fischer, and W. Meier. Reduced complexity attacks on the alternating step generator. In C. Adams, A. Miri, and M. Wiener, editors, *Selected Areas in Cryptography*, pages 1–16, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[68] A. V. Khvalkovskiy, D. Apalkov, S. Watts, R. Chepulskii, R. S. Beach, A. Ong, X. Tang, A. Driskill-Smith, W. H. Butler, P. B. Visscher, D. Lottis, E. Chen, V. Nikitin, and M. Krounbi. Basic principles of stt-mram cell operation in memory arrays. *Journal of Physics D: Applied Physics*, 46(7):074001, jan 2013.

[69] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[70] A. Klein. *Linear Feedback Shift Registers*, pages 17–58. Springer London, London, 2013.

[71] P. Koopman. Best crc polynomials. https://users.ece.cmu.edu/ koopman/crc/, 2015.

[72] B. Korzh, C. C. W. Lim, R. Houlmann, N. Gisin, M. J. Li, D. Nolan, B. Sanguinetti, R. Thew, and H. Zbinden. Provably secure and practical quantum key distribution over 307km of optical fibre. *Nature Photonics*, 9(3):163–168, Feb 2015.

[73] D. P. Kroese and R. Y. Rubinstein. Monte carlo methods. *WIREs Computational Statistics*, 4(1):48–58, 2012.

[74] S. C. L. Zhou. A 7-transistor-per-cell, high-density analog storage array with 500µv update accuracy and greater than 60db linearity. *IEEE Symposium on Circuits and Systems (ISCAS 2014)*, 2014.

[75] A. Laborieux, M. Ernoult, T. Hirtzlin, and D. Querlioz. Synaptic metaplasticity in binarized neural networks. *Nature communications*, 12(1):1–12, 2021.

[76] Y. Lecun, S. Chopra, and R. Hadsell. A tutorial on energy-based learning. In *Predicting Structured Data*, 01 2006.

[77] S. Lee, J. Jeon, K. Eom, C. Jeong, Y. Yang, J.-Y. Park, C.-B. Eom, and H. Lee. Multi-level memristors based on two-dimensional electron gases in oxide heterostructures for high precision neuromorphic computing. *Research Square*, 2021.

[78] S. Lee, J. Kim, J. Ha, and B. Zhang. Overcoming catastrophic forgetting by incremental moment matching. *CoRR*, abs/1703.08475, 2017.

[79] M. Lenzlinger and E. H. Snow. Fowler-nordheim tunneling into thermally grown sio2. *Journal of Applied Physics*, 40(1):278–283, 1969.

[80] Q. Li, S. Navakkode, M. Rothkegel, T. W. Soong, S. Sajikumar, and M. Korte. Metaplasticity mechanisms restore plasticity and associativity in an animal model of alzheimer's disease. *Proceedings of the National Academy of Sciences*, 114(21):5527–5532, 2017.

[81] T. M. Library. Stone foundation tablet assyrian. https://www.themorgan.org/collection/Written-in-Stone, 2023.

[82] X. Liu, M. Masana, L. Herranz, J. van de Weijer, A. M. López, and A. D. Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. *CoRR*, abs/1802.02950, 2018.

[83] I. Loshchilov and F. Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.

[84] S. C. M. Gu. A varactor-driven, temperature compensated cmos floating-gate current memory with 130ppm/k temperature sensitivity. *IEEE Journal of Solid-State Circuits*, 47(11):2846–2856, 2012.

[85] Z. Ma and G. A. E. Vandenbosch. Impact of random number generators on the performance of particle swarm optimization in antenna design. In *2012 6th European Conference on Antennas and Propagation (EUCAP)*, pages 925–929, 2012.

[86] H. Maghrebi, T. Portigliatti, and E. Prouff. Breaking cryptographic implementations using deep learning techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 3–26. Springer, 2016.

[87] G. Mahajan and S. Nadkarni. Intracellular calcium stores mediate metaplasticity at hippocampal dendritic spines. *The Journal of physiology*, 597(13):3473–3502, 2019.

[88] J. Massey. Shift-register synthesis and bch decoding. *IEEE Transactions on Information Theory*, 15(1):122–127, 1969.

[89] A. Mehonic, A. Sebastian, B. Rajendran, O. Simeone, E. Vasilaki, and A. J. Kenyon. Memristors—from in-memory computing, deep learning acceleration, and spiking neural networks to the future of neuromorphic and bio-inspired computing. *Advanced Intelligent Systems*, 2(11):2000085, 2020.

[90] D. Mehta, K. Aono, and S. Chakrabartty. A self-powered analog sensor-data-logging device based on fowler-nordheim dynamical systems. *Nature Communications*, 11(1), Oct 2020.

[91] D. Mehta, M. Rahman, K. Aono, and S. Chakrabartty. An adaptive synaptic array using fowler–nordheim dynamic analog memory. *Nature Communications*, 13(1):1–11, 2022.

[92] F. Merrikh-Bayat, X. Guo, M. Klachko, M. Prezioso, K. K. Likharev, and D. B. Strukov. High-performance mixed-signal neurocomputing with nanoscale floating-gate memory cell arrays. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10):4782–4790, 2018.

[93] G. Murphy, A. Keeshan, R. Agarwal, and E. Popovici. Hardware - software implementation of public-key cryptography for wireless sensor networks. In *2006 IET Irish Signals and Systems Conference*, pages 463–468, 2006.

[94] J. Nocedal and S. Wright. Numerical optimization. *Springer Science and Business Media*, 2006.

[95] N. A. of Sciences, Engineering, and Medicine. 4 quantum computing's implications for cryptography. *Quantum Computing: Progress and Prospects*, 2019.

[96] D. C. P. Cappelletti, R. Bez and L. Fratin. Failure mechanisms of flash cell in program/erase cycling. *Proceedings of 1994 IEEE International Electron Devices Meeting*, page 291–294, 1994.

[97] S. Pal, S. Bose, and A. Islam. Design of memristor based low power and highly reliable reram cell. *Microsystem Technologies*, pages 1–15, 2019.

[98] S. Pal, S. Bose, W.-H. Ki, and A. Islam. Design of power-and variability-aware nonvolatile rram cell using memristor as a memory element. *IEEE Journal of the Electron Devices Society*, 7:701–709, 2019.

[99] W. W. Peterson and D. T. Brown. Cyclic codes for error detection. *Proceedings of the IRE*, 49(1):228–235, 1961.

[100] S. Poddar, Y. Zhang, L. Gu, D. Zhang, Q. Zhang, S. Yan, M. Kam, S. Zhang, Z. Song, W. Hu, et al. Down-scalable and ultra-fast memristors with ultra-high density three-dimensional arrays of perovskite quantum wires. *Nano Letters*, 21(12):5036–5044, 2021.

[101] C. Portmann and R. Renner. Cryptographic security of quantum key distribution. *arXiv preprint arXiv:1409.3525*, 2014.

[102] S. C. R. Genov and G. Cauwenberghs. Silicon support vector machine with on-line learning. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(3):385–404, 2003.

[103] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, USA, 1979.

[104] M. Rahman, S. Bose, and S. Chakrabartty. On-device synaptic memory consolidation using fowler-nordheim quantum-tunneling. *Frontiers in Neuroscience*, 16, 2023.

[105] M. Rahman and S. Chakrabartty. Gps-free synchronized pseudo-random number generators for internet-of-things. *Frontiers in Computer Science*, 5, 2023.

[106] M. Rahman, L. Zhou, and S. Chakrabartty. Spotkd: A protocol for symmetric key distribution over public channels using self-powered timekeeping devices. *IEEE Transactions on Information Forensics and Security*, 17:1159–1171, 2022.

[107] G. A. V. R. C. Rao, P. V. Lakshmi, and N. R. Shankar. Article: Rsa public key cryptosystem using modular multiplication. *International Journal of Computer Applications*, 80(5):38–42, October 2013. Full text available.

[108] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, Feb. 1978.

[109] A. Roxin and S. Fusi. Efficient partitioning of memory systems and its importance for memory consolidation. *PLoS computational biology*, 9(7):e1003146, 2013.

[110] A. Saulsbury, F. Pong, and A. Nowatzyk. Missing the memory wall: The case for processor/memory integration. In *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, ISCA '96, page 90–101, New York, NY, USA, 1996. Association for Computing Machinery.

[111] W. Schindler and Ç. K. Koç. *Random Number Generators for Cryptographic Applications*, pages 5–23. Springer US, Boston, MA, 2009.

[112] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan. 2016.

[113] N. S. Sohoni, C. R. Aberger, M. Leszczynski, J. Zhang, and C. Ré. Low-memory neural network training: A technical report, 2019.

[114] K. Staniec and M. Kowal. On vulnerability of selected iot systems to radio jamming—a proposal of deployment practices. *Sensors*, 20(21), 2020.

[115] X. Sun, J. Choi, C.-Y. Chen, N. Wang, S. Venkataramani, V. V. Srinivasan, X. Cui, W. Zhang, and K. Gopalakrishnan. Hybrid 8-bit floating point (hfp8) training and inference for deep neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[116] B. Sunar and Ç. K. Koç. *True Random Number Generators for Cryptography*, pages 55–73. Springer US, Boston, MA, 2009.

[117] G. M. T. Semwal, P. Yenigalla and S. B. Nair. A practitioners' guide to transfer learning for text classification using convolutional neural networks. *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 1–9, 2018.

[118] E. Takeda and N. Suzuki. An empirical model for device degradation due to hot-carrier injection. *IEEE Electron Device Letters*, 4(4):111–113, 1983.

[119] H. H. Tan and K. H. Lim. Vanishing gradient mitigation with deep learning neural network optimization. In *2019 7th international conference on smart computing & communications (ICSCC)*, pages 1–4. IEEE, 2019.

[120] J. Tang, D. Bishop, S. Kim, M. Copel, T. Gokmen, T. Todorov, S. Shin, K.-T. Lee, P. Solomon, K. Chan, W. Haensch, and J. Rozen. Ecram as scalable synaptic cell for high-speed, low-power neuromorphic computing. In *2018 IEEE International Electron Devices Meeting (IEDM)*, pages 13.1.1–13.1.4, 2018.

[121] F. Tu, W. Wu, S. Yin, L. Liu, and S. Wei. Rana: Towards efficient neural acceleration with refresh-optimized embedded dram. In *Proceedings of the 45th Annual International Symposium on Computer Architecture*, page 340–352. IEEE Press, 2018.

[122] T. Tuma, A. Pantazi, M. Le Gallo, A. Sebastian, and E. Eleftheriou. Stochastic phase-change neurons. *Nature nanotechnology*, 11(8):693–699, 2016.

[123] TutorialsMate. Types of computer memory. https://www.tutorialsmate.com/2020/04/types-of-computer-memory.html, 2023.

[124] R. Ursin, F. Tiefenbacher, T. Schmitt-Manderbach, H. Weier, T. Scheidl, M. Lindenthal, B. Blauensteiner, T. Jennewein, J. Perdigues, P. Trojek, and et al. Entanglement-based quantum communication over 144km. *Nature Physics*, 3(7):481–486, Jun 2007.

[125] S. A. Wilber. *Synchronized True Random Number Generator*. US20180039485A1, 2017.

[126] H. Wire. Fugaku retains title as world's fastest supercomputer. https://www.hpcwire.com/off-the-wire/fugaku-retains-title-as-worlds-fastest-supercomputer/, November 17, 2020.

[127] L. Wu, H. Liu, J. Li, S. Wang, and X. Wang. A multi-level memristor based on al-doped hfo 2 thin film. *Nanoscale research letters*, 14(1):1–7, 2019.

[128] W. A. Wulf and S. A. McKee. Hitting the memory wall: Implications of the obvious. *SIGARCH Comput. Archit. News*, 23(1):20–24, mar 1995.

[129] Z. W. X. Gu and S. S. Iyer. Charge-trap transistors for cmos-only analog memory. *IEEE Transactions on Electron Devices*, 66(10):4183–4187, 2019.

[130] T. P. Xiao, C. H. Bennett, B. Feinberg, S. Agarwal, and M. J. Marinella. Analog architectures for neural network acceleration based on non-volatile memory. *Applied Physics Reviews*, 7(3):031301, 07 2020.

[131] C.-X. Xue, T.-Y. Huang, J.-S. Liu, T.-W. Chang, H.-Y. Kao, J.-H. Wang, T.-W. Liu, S.-Y. Wei, S.-P. Huang, W.-C. Wei, Y.-R. Chen, T.-H. Hsu, Y.-K. Chen, Y.-C. Lo, T.-H. Wen, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, and M.-F. Chang. 15.4 a 22nm 2mb reram compute-in-memory macro with 121-28tops/w for multibit mac computing for tiny ai edge devices. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, pages 244–246, 2020.

[132] G. Yang, C. S. W. Lai, J. Cichon, L. Ma, W. Li, and W.-B. Gan. Sleep promotes branch-specific formation of dendritic spines after learning. *Science*, 344(6188):1173–1178, 2014.

[133] G. Yang, F. Pan, and W.-B. Gan. Stably maintained dendritic spines are associated with lifelong memories. *Nature*, 462(7275):920–924, 2009.

[134] J. Yin, Y. Cao, Y.-H. Li, S.-K. Liao, L. Zhang, J.-G. Ren, W.-Q. Cai, W.-Y. Liu, B. Li, H. Dai, G.-B. Li, Q.-M. Lu, Y.-H. Gong, Y. Xu, S.-L. Li, F.-Z. Li, Y.-Y. Yin, Z.-Q. Jiang, M. Li, J.-J. Jia, G. Ren, D. He, Y.-L. Zhou, X.-X. Zhang, N. Wang, X. Chang, Z.-C. Zhu, N.-L. Liu, Y.-A. Chen, C.-Y. Lu, R. Shu, C.-Z. Peng, J.-Y. Wang, and J.-W. Pan. Satellite-based entanglement distribution over 1200 kilometers. *Science*, 356(6343):1140–1144, 2017.

[135] C. Yukonhiatou, T. Yoshihisa, T. Kawakami, Y. Teranishi, and S. Shimojo. A fast stream transaction system for real-time iot applications. *Internet of Things*, 11:100182, 2020.

[136] F. Zenke, B. Poole, and S. Ganguli. Improved multitask learning through synaptic intelligence. *CoRR*, abs/1703.04200, 2017.

[137] L. Zhou and S. Chakrabartty. Self-powered timekeeping and synchronization using fowler-nordheim tunneling-based floating-gate integrators. *IEEE Transactions on Electron Devices*, PP:1–7, 01 2017.

[138] L. Zhou, S. H. Kondapalli, K. Aono, and S. Chakrabartty. Desynchronization of self-powered fn tunneling timers for trust verification of iot supply chain. *IEEE Internet of Things Journal*, 6(4):6537–6547, 2019.

Appendix A

Supplementary Information for Chapter 2

A.1 Equivalent Circuit Model of FN-Synapse

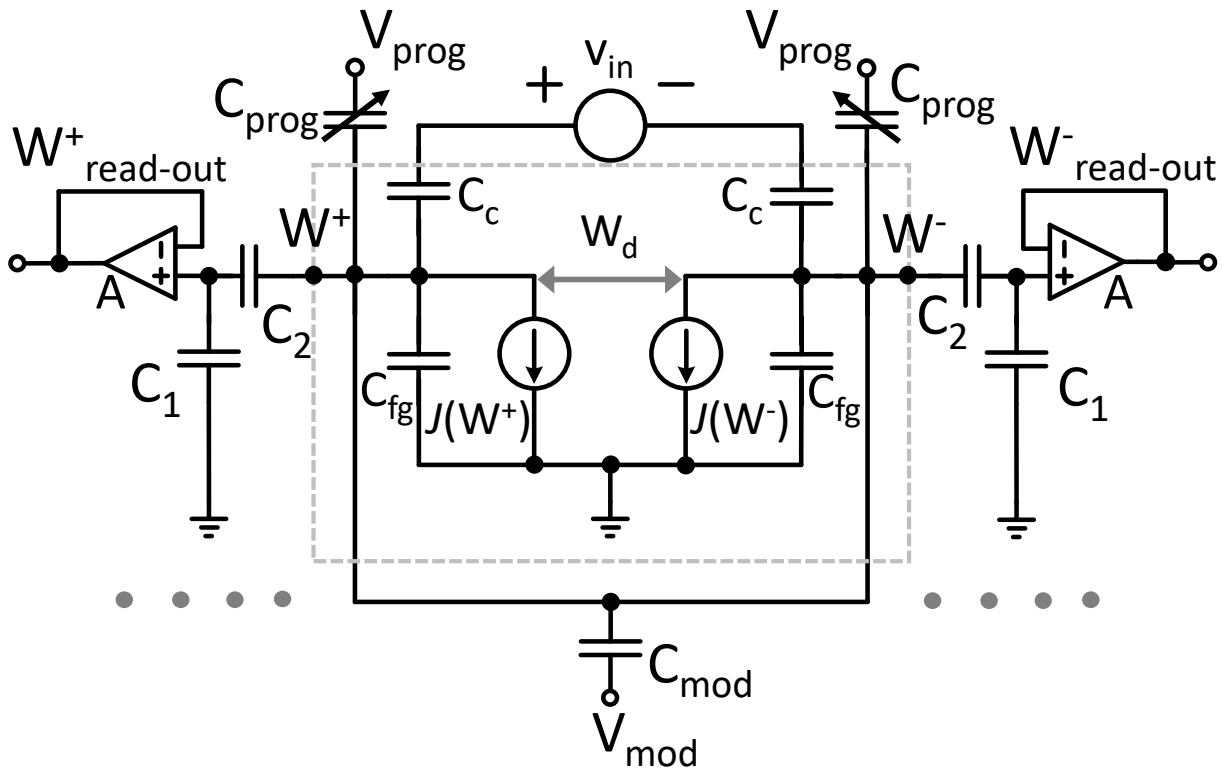

Figure A.1: Equivalent circuit model of an FN-synapse

The equivalent circuit model of a single FN-synapse is shown in Fig. A.1. The synaptic weight W_d is stored as a difference between the voltages (W^+ and W^-) on the floating-gates. The FN tunneling current is modeled using voltage dependent current sources $J(W^+)$, $J(W^-)$

that discharge the floating-gate capacitances C_{fg}. Both W_d and the common-mode voltage W_c are estimated by measuring W^+ and W^- using a capacitive divider formed by C_1 and C_2 and respective source-followers A. This configuration has been previously demonstrated to avoid read-disturbances when measuring the floating-gate voltages [90, 137]. External input v_{in} is differentially coupled to the FN-synapse through the capacitances C_c and C_{mod} is used to couple the signal $m(t) = \frac{dv_{mod}(t)}{dt}$ common to all synapses. $m(t)$ is used to adjust the plasticity of the entire synaptic array. The initial charge on the floating-gates are programmed using a combination of FN quantum-tunneling and hot-electron injection, details of which can be found in [90].

A.2 Modeling Results

A.2.1 Behavioral Model of the FN-Synapse

The fabricated prototype of the FN-synapse array comprises of 64 FN-synaptic elements. Thus, for large-scale memory consolidation experiments and for large-scale continual learning experiments, we require a behavioral model that can accurately capture the response of each FN-synapse in the array. In our previous works [138, 137] we have validated that equation 25 in the main manuscript can accurately (accuracy greater than 99%) model the dynamic response of a single FN tunneling junction and a corresponding integrator. For this work we instantiated two tunneling junctions corresponding to the floating-gates W^+ and W^- and the model parameters k_0, k_1 and k_2 were estimated using measured results. A non-linear regression was specifically used to estimate k_1 and k_2 [90, 137], whereas k_0 was determined from the voltage to which each of the floating-gates were initialized. To validate the behavioral model of the FN-synapse, we carried out a set of experiments and compared the outputs against the analytical results shown in equations $33, 36 - 38$ in the Methods Section of the main manuscript. Note that these analytical expressions were derived for a constant modulation input, therefore $V_{mod}(t)$ was kept constant at $0V$ in all the simulated experiments. Fig A.2 and A.3 summarizes all the results obtained from the behavioral model.

In the first experiment, we measured the weight evolution of an FN-synapse using the fabricated prototype for a series of potentiation/depression pulses. The same input was provided to the software model and the weight evolution was simulated. Fig A.2 (a) shows

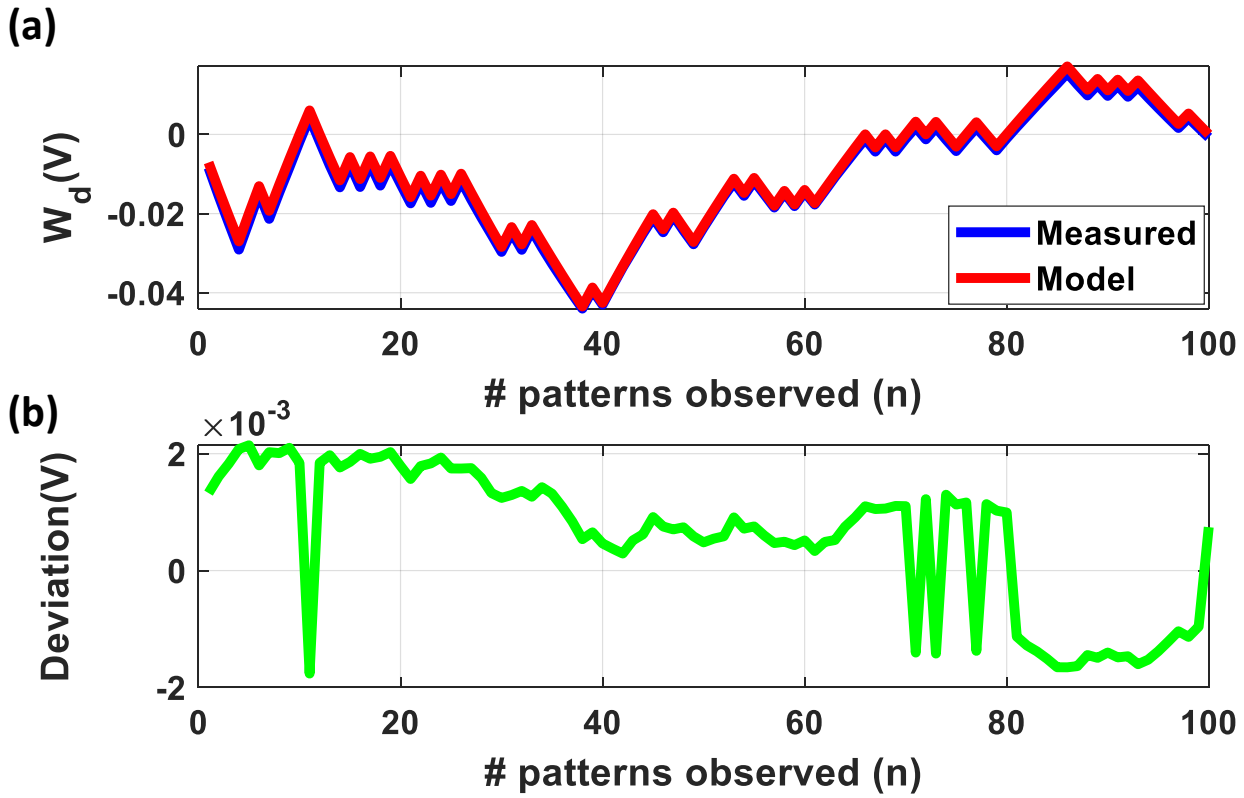

Figure A.2: (a) of equivalent plasticity and initial conditions when exposed to the same input pattern and (b) the corresponding deviation.

that the stored weight of the software model accurately matches with that of the hardware FN-synapse with a small deviation as shown in Fig A.2 (b). This verifies that both hardware FN-synapse and software model behaves similarly when subjected to same stimuli. Next, we ran a Monte-Carlo simulation where we updated a network of $N = 10000$ FN-synapses with random binary pattern. Each tunneling junction of FN-synapses were initialized at $W_{c0} = 4.5V$. The updates were provided as a differential input voltage pulses of magnitude $4V$ and duration $\Delta t = 100ms$ to each synapses. The experiment was repeated for 1000 Monte-Carlo simulations. Fig A.3 (a), (b), and (c) shows the SNR, memory retrieval signal $S(n)$ and the noise $\nu(n)$ respectively obtained from the software model of FN-synapse network. In Fig A.3 (a) we observe that the SNR from the software model matches accurately with the analytical expression. Both $S(n)$ and $\nu(n)$ described in equation 4 in the main manuscript have two different regimes depending on the value of γ. When $n << \gamma$, $S(n)$ is approximately constant and $\nu(n)$ increases at a rate of $\sqrt{n}$. On the other hand, when $n >> \gamma$, $S(n)$ and $\nu(n)$ falls off at a rate of $\frac{1}{n}$ and $\frac{1}{\sqrt{n}}$ respectively. Fig A.3 (b) and (c) shows that the response from the software model follows theses trends and captures both the regimes accurately. In the next set of numerical experiments, we verified whether the FN-synapse network shows similar trends as the analytic expression in response to changing the value of γ in equation 5 in the main manuscript. Note that the parameter γ is defined as

$$\gamma = \frac{k_0}{k_1 \Delta t} \tag{A.1}$$

where $k_0 = exp(\frac{k_2}{W_{c0}})$. Therefore, γ for the same set of FN-synapses increases when Δt or W_{c0} decreases and vice versa. According to equation 4, the value of n at which the regimes in these responses changes also shifts. Moreover, the initial values for both $S(n)$ and $\nu(n)$ depends on the value of γ while SNR is agnostic to changes in γ. Fig A.3 (d)-(i) show the FN-synapse responses in relation to changing the pulse width and the initialization condition for a network size of $N = 1000$. From the figures we can observe that the software model is in very good agreement with the analytic expressions. Finally, we verify the behavioral model in relation to change in the size N of the FN-synapse network. From the analytic expressions in equation 4 in the main manuscript, $SNR \propto \sqrt{N}$ and $\nu(n) \propto \frac{1}{\sqrt{N}}$ while $S(n)$ remains constant with respect to N. Fig A.3 (j)-(l) shows that the FN-synapse network exhibits these attributes accurately. Note that the regime switching point in $S(n)$ and $\nu(n)$ remains constant, since γ does not depend on the size of the network.

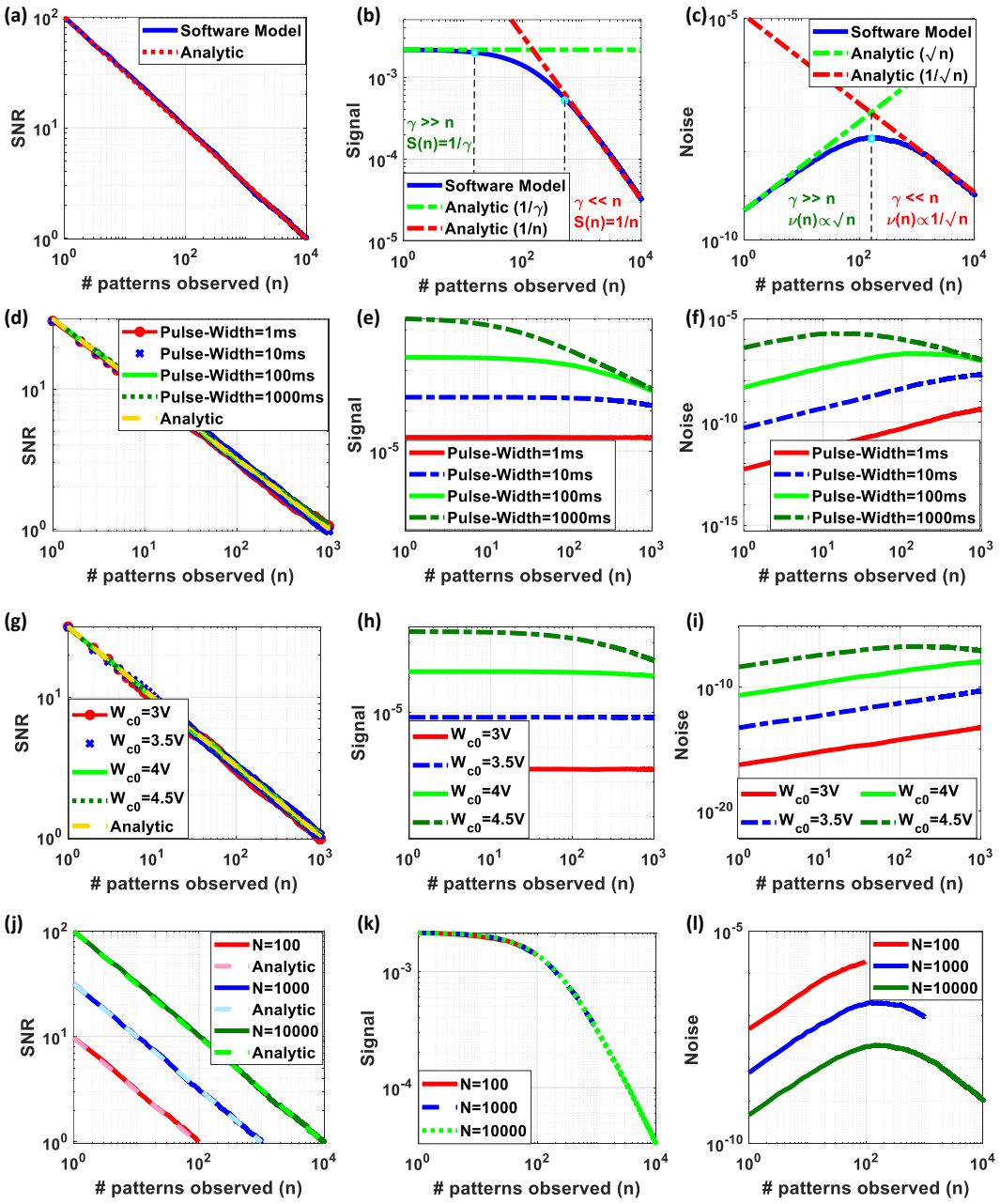

Figure A.3: in terms of (a) SNR, (b) signal and (c) noise. The effect on the SNR, signal and noise of the software model when (d)-(f) the pulse-width of the input pulse is varied and when (g)-(i) the magnitude of the input pulse is varied. (j)-(l) The impact of change in network size on SNR, signal and noise .

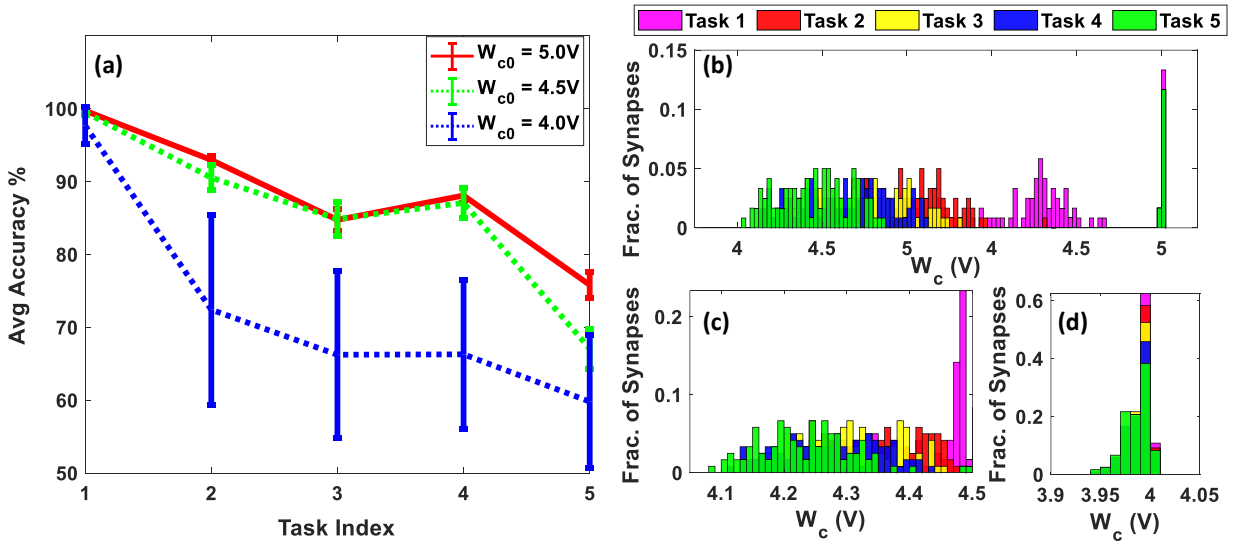

Figure A.4: on (a) overall average accuracy of the split-MNIST incremental domain learning tasks as a result of the degree of change in plasticity of their corresponding weights for (b) $W_{c0} = 5.0$V, (c) $W_{c0} = 4.5$V and (d) $W_{c0} = 4.0$V.

A.3 Plasticity and Consolidation

The ability of a network to learn new tasks is contingent on the availability of adequate range of plasticity of the synapses so that the weights learned from previous tasks can adapt sufficiently to reflect the requirements for the new tasks. Traditional volatile memories have practically infinite range of plasticity and can therefore change the weights stored to any extent that is required. However, this feature might not be beneficial for continual learning where the network needs to learn new tasks without forgetting the previous ones. This *rigidity-plasticity* dilemma is at the core underpinning of memory consolidation where more frequently used synapses become more rigid in comparison to the less frequently used synapses. Thus, a balance between the range of plasticity required to learn successive tasks and the consolidation of the weights learned in the process is key to continual learning. In the case of FN-synapse based neural networks, the range of plasticity is determined by the initial tunneling region of the device. A high tunneling region, denoted by a larger value of W_{c0}, ensures that the synapses are plastic enough to learn several successive tasks and slowly become rigid over time. This is seen in the case of $W_{c0} = 5$V and $W_{c0} = 4.5$V, which exhibit significantly better overall average accuracy over five tasks as shown in SI Fig. A.4 (a) as the weights stored in their synapses (shown in SI Fig. A.4 (b) and A.4 (c) respectively) slowly

spread from a highly plastic to a rigid region over the course of the five tasks. In contrast, a relatively low initial tunneling region, such as in the case of $W_{c0} = 4$V, does not learn new tasks as well as the previous couple of cases as shown in SI Fig. A.4 (a) since in this case the weights stored in the synapse are already relatively rigid at the point of initiation and barely undergo any change as illustrated in SI Fig. A.4 (d). Therefore by choosing the initial plasticity level appropriately, we can achieve an optimal balance between the range of plasticity and consolidation suitable for continual learning. It is worth mentioning here that while choosing an appropriate temporal profile of $m(t)$ can be used to re-adjust the plasticity of the synapses after each update, it does not however change the range of plasticity afforded to the network since that is determined by the initial W_{c0}.

A.4 Neural Network Architecture

The architecture of the 4-layer fully-connected MLP is shown in SI Fig. A.5 (a). Comprising an input layer of 1024 neurons corresponding to images of 32x32 pixels, two hidden layers of 80 and 60 neurons each and an output layer of 2 neurons that differentiates between (0,1) in t_1, (2,3) in t_2, (4,5) in t_3, (6,7) in t_4 and (8,9) in t_5 the network was constructed in MATLAB and trained with SGD and ADAM with learning rate of 0.001 for 4 epochs with a mini-batch size of 128. For comparisons with EWC and Online EWC, the network was replicated in python and trained with exactly the same parameters.

The evolution of the plasticity/usage of weights of the different layers of the FN-synapse based neural network are shown in SI Fig. A.5 (b)-(d). Given the relatively large number of weights between layer 1-2 and layer 2-3, the amount of change in plasticity that they undergo (as shown in Fig A.5 (b) and A.5(c) respectively) is much lesser in comparison with those between layer 3-4 (as shown in Fig A.5 (d)) as the presence of much fewer weights ensures that they are modified considerably frequently due to lack of any redundancy.

Fig. 6 of the main manuscript and SI Fig. A.6 already depict the advantages of the FN-synapse based neural networks using either SGD or ADAM as the optimizer when employed within the aforementioned architecture. In addition, if the size of the neural network is increased by increasing the number of neurons in the hidden layers from 80/60 in layer 2/3 to 400/400, it can be observed from SI Fig. A.7 (a)-(b) that the average overall accuracy of the FN-synapse based network still outperforms the ones without it as the memory element.

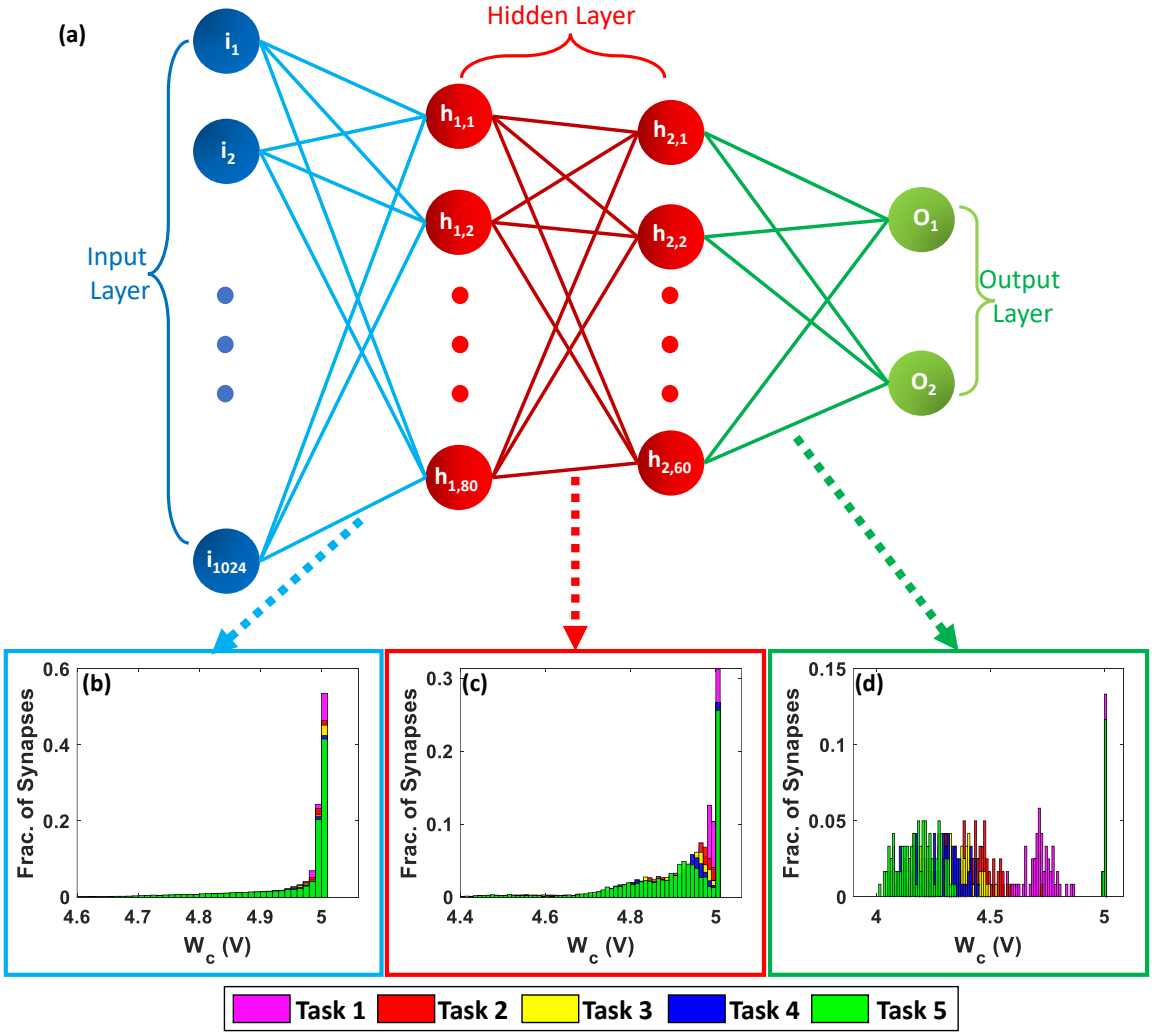

Figure A.5: (a) used in the report and the evolution of corresponding weights in between (b) layer 1 and 2, (c) layer 2 and 3, and (d) layer 3 and 4 over five successive tasks.

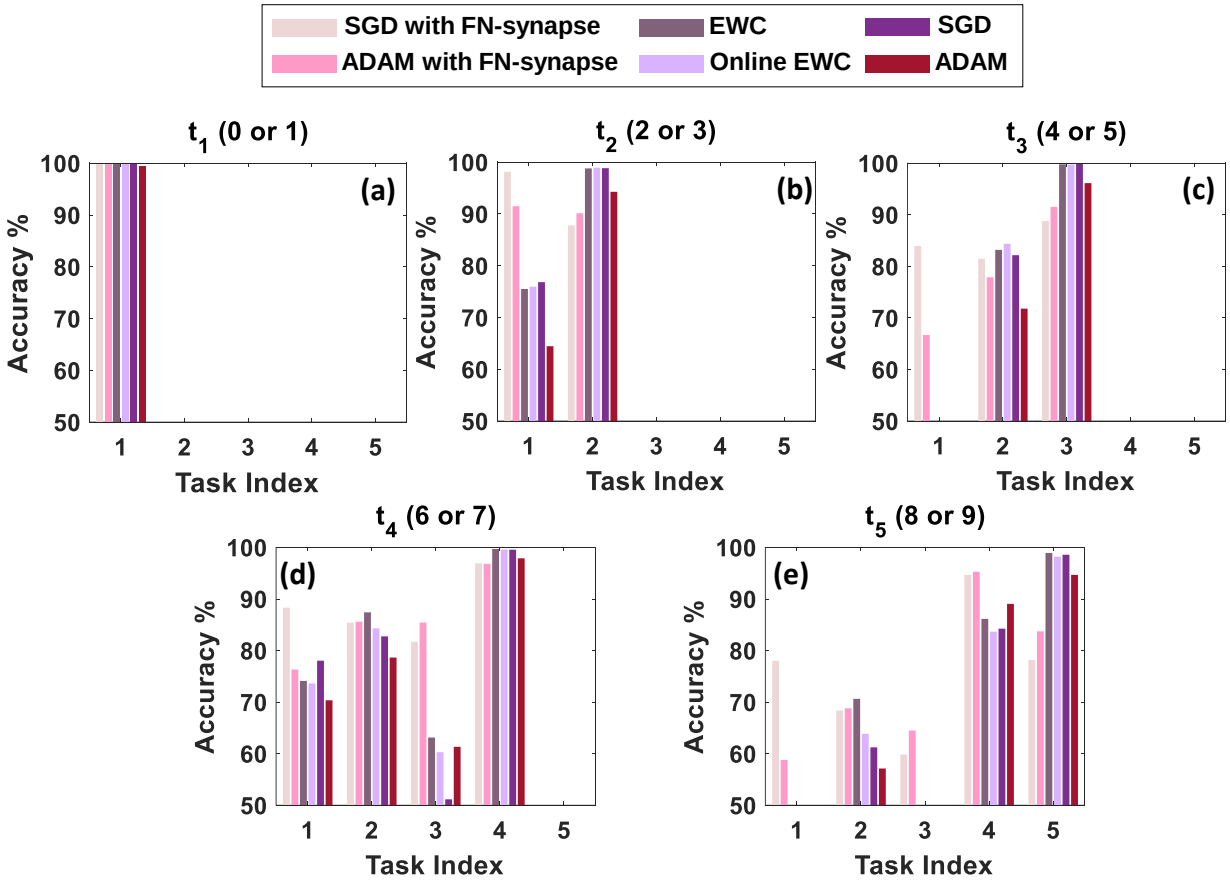

Figure A.6: (a)-(e) of SGD and ADAM with FN-synapse, ADAM with EWC and Online EWC, SGD and ADAM with conventional memory.

Interestingly, the accuracy of the larger network with FN-synapse is slightly lower than that of the smaller network with FN-synapse for task 3 and beyond. This dip is actually an indication of higher plasticity, and therefore slower consolidation, of the larger network due to presence of many more synapses which are still highly plastic after several tasks, which makes FN-synapse based large neural networks equipped with the capability of learning more complicated tasks than split-MNIST and yet exhibit far better consolidation than conventional memory.

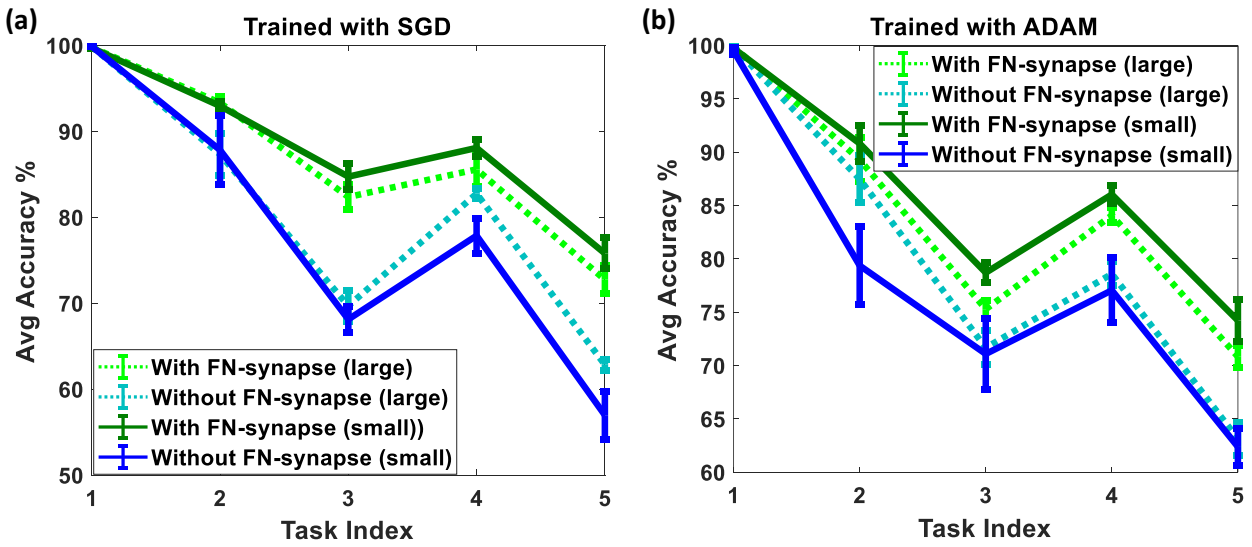

Figure A.7: when trained with (a) SGD and (b) ADAM.

A.5 Effects of Mismatch

The FN-synapse comprises of two differential FN tunneling junctions and the operation of the synapse assumes that the junctions are well matched. This will ensure that the weights stored in the synapse are equally plastic/rigid, when increasing or decreasing the magnitude of the weight. A key requirement is that the tunneling rates of the two junctions corresponding to W^+ and W^- are synchronized with each other. Previously, we have shown in [91, 90] that two such FN-dynamical systems can be synchronized to a very high degree of accuracy even in the presence of temperature variations or device mismatch. On the other hand, a mismatch in device characteristics across one or more FN synapses, specifically the parameters k_1 and k_2, must be taken into consideration. This is because a neural network could comprise of billions of synapses and mismatch in synaptic behavior could pose a problem. SI Fig. A.8 (a) shows the effect of a 5% mismatch in device characteristics across synapses on the SNR of an FN-synapse network comprising of 10,000 synapses. In this experiment, the network was subjected to 10,000 randomized balanced updates, similar to the previous consolidation experiments. It can be observed that the network with mismatch shows a small degradation in SNR or memory retention compared to the one without any mismatch. However, the SNR still follows the power-law curve. On the contrary a mismatch of 5% does not lead to any deterioration whatsoever of the average overall accuracy of the network when trained with SGD over the split-MNIST dataset with the incremental domain learning tasks as depicted

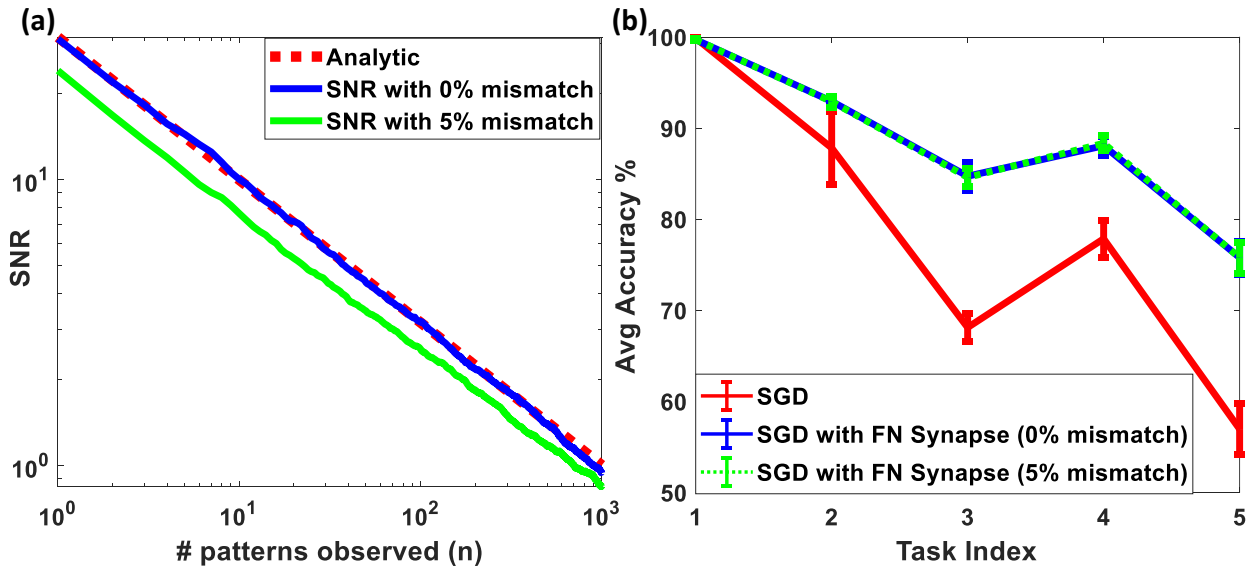

Figure A.8: across FN synapses on (a) memory retention and (b) learning ability on the split-MNIST based incremental domain learning tasks.

in Fig A.8 (b). This shows the robustness of the FN-synapse based network and the ability of learning to compensate for device mismatch.

A.6 Detailed Derivations

In this section, few additional steps were added to the modelling derivations in the main manuscript for the reader's interest. However, for the sake of brevity and to avoid repetition, unnecessary descriptions have been avoided.

A.6.1 Weight Update For Differential Synaptic Model

The state equations of two dynamical systems (corresponding to state variables W^+ and W^- with $J(.)$ defining their rate of change), when subjected to differential input $\pm\frac{1}{2}X(t)$ and common-mode modulation input $m(t)$ is given by:

$$\frac{dW^+}{dt} = -J(W^+) + \frac{1}{2}X(t) + \frac{1}{2}m(t) \tag{A.2}$$

$$\frac{dW^-}{dt} = -J(W^-) - \frac{1}{2}X(t) + \frac{1}{2}m(t) \tag{A.3}$$

Since, $W_d = \frac{W^+ - W^-}{2}$ and $W_c = \frac{W^+ + W^-}{2}$, A.2 and A.3 can be written as:

$$\frac{d(W_c + W_d)}{dt} = -J(W_c + W_d) + \frac{1}{2}X(t) + \frac{1}{2}m(t) \tag{A.4}$$

$$\frac{d(W_c - W_d)}{dt} = -J(W_c - W_d) - \frac{1}{2}X(t) + \frac{1}{2}m(t) \tag{A.5}$$

Then, by adding and subtracting A.4 and A.5, the following is obtained:

$$\frac{dW_c}{dt} = -\left(\frac{J(W_c + W_d) + J(W_c - W_d)}{2}\right) + m(t) \tag{A.6}$$

$$\frac{dW_d}{dt} = -\left(\frac{J(W_c + W_d) - J(W_c - W_d)}{2}\right) + X(t) \tag{A.7}$$

Upon applying Taylor series expansion on A.6 and A.7, with the assumption that $W_c >> W_d$, we get:

$$\frac{dW_c}{dt} = -J\left(W_c\right) + m(t) \tag{A.8}$$

$$\frac{dW_d}{dt} = -J'\left(W_c\right) W_d + X(t) \tag{A.9}$$

Therefore to obtain an expression of weight update ($\frac{dW_d}{dt}$) with respect to the common-mode usage (W_c), we need to obtain an expression for $J'(W_c)$. Thus, by differentiating A.8 w.r.t t, we obtain:

$$\frac{d^2W_c}{dt^2} = -J'\left(W_c\right)\frac{dW_c}{dt} + m'(t) \tag{A.10}$$

$$J'\left(W_c\right) = -\frac{\left(\frac{d^2W_c}{dt^2} - m'(t)\right)}{\frac{dW_c}{dt}} \tag{A.11}$$

Inserting A.11 into A.9, we get:

$$\frac{dW_d}{dt} = -\left[\frac{\frac{d^2W_c}{dt^2} - m'(t)}{\frac{dW_c}{dt}}\right] W_d + X(t) \tag{A.12}$$

Now, for the trivial case where $m(t) = c$, where c is an arbitrary constant, $m'(t) = 0$ and thus A.12 becomes:

$$\frac{dW_d}{dt} = -\left[\frac{d^2W_c}{dt^2}\left(\frac{dW_c}{dt}\right)^{-1}\right] W_d + X(t) \tag{A.13}$$

A.6.2 Optimal Usage Profile

The decay rate ($r(t)$) obtained from the weight update rule in A.13 is given by:

$$r(t) = -\left[\frac{d^2W_c}{dt^2}\left(\frac{dW_c}{dt}\right)^{-1}\right] \tag{A.14}$$

To avoid catastrophic forgetting, the decay rate associated with the EWC model's weight update rule, for the case of balanced inputs, is $r(t) = O\left(\frac{1}{t}\right)$. Therefore, by choosing $W_c = \frac{1}{f(\log t)}$, where $f(.) \geq 0$ is a monotonic function we obtain

$$r(t) = \frac{1}{t}\left(1 + \frac{2f'(\log t)}{\log t} - \frac{f''(\log t)}{f'(\log t)}\right) \tag{A.15}$$

which is of the order $O\left(\frac{1}{t}\right)$. The simplest form of $f(.)$ such that W_c satisfies both monotonicity and the order of decay, is given by:

$$W_c = \frac{\beta}{\log(t)} \tag{A.16}$$

where β is an arbitrary constant. Consequently, to obtain the non-linear function $J(.)$ which enforces the above constraint, we substitute A.16 into A.8 to get:

$$\frac{d\left(\frac{\beta}{\log(t)}\right)}{dt} = -J(W_c) + m(t) \tag{A.17}$$

$$\frac{-\beta}{t(log(t))^2} = -J(W_c) + m(t) \tag{A.18}$$

For the case of $m(t) = 0$, A.18 becomes:

$$J(W_c) = \frac{\beta}{t(log(t))^2} \tag{A.19}$$

Now, from A.16, we can obtain an expression for $log(t)$ as

$$log(t) = \frac{\beta}{W_c} \tag{A.20}$$

and an expression for t as follows:

$$exp(log(t)) = exp\left(\frac{\beta}{W_c}\right) \tag{A.21}$$

$$t = exp\left(\frac{\beta}{W_c}\right) \tag{A.22}$$

Then, by substituting A.20 and A.22 in A.19, we obtain:

$$J\left(W_c\right) = \frac{1}{\beta} W_c^2 \exp\left(-\frac{\beta}{W_c}\right). \tag{A.23}$$

A.6.3 Signal-to-noise Ratio Estimation for Random Pattern Experiment

The weight update equation for an FN-synapse (similar to A.13) is given by:

$$C_T \frac{dW_d}{dt} = -\left[\frac{d^2W_c}{dt^2}\left(\frac{dW_c}{dt}\right)^{-1}\right] W_d + C_c \frac{dv_{in}}{dt} \tag{A.24}$$

where $C_T = f(C_1, C_2, C_{fg})$ is the cumulative capacitance and C_c is the coupling capacitance of the FN-synapse equivalent circuit as shown in Fig. A.1. Since, the physics of FN-tunneling leads to a common-mode voltage W_c profile such that

$$W_c(t) = \frac{k_2}{\log(k_1 t + k_0)} \tag{A.25}$$

where $k_0 = \exp\left(\frac{k_2}{W_{c0}}\right)$ and W_{c0} refers to the initial voltage at the floating-gate, by substituting A.25 in A.24, we get:

$$C_T \frac{dW_d}{dt} = -\left[\frac{\left(\frac{k_1^2 k_2}{(k_1 t + k_0)^2 log^2(k_1 t + k_0)}\right)}{\left(\frac{k_1 k_2}{(k_1 t + k_0) log^2(k_1 t + k_0)}\right)}\left(1 + \frac{2}{log(k_1 t + k_0)}\right)\right] W_d + C_c \frac{dv_{in}}{dt} \tag{A.26}$$

$$C_T \frac{dW_d}{dt} = -\left[\left(\frac{k_1}{(k_1 t + k_0)}\right)\left(1 + \frac{2}{log(k_1 t + k_0)}\right)\right] W_d + C_c \frac{dv_{in}}{dt} \tag{A.27}$$

In the scenario where $C_T = C_c$, we get:

$$\frac{dW_d}{dt} = -\left[\left(\frac{k_1}{(k_1 t + k_0)}\right)\left(1 + \frac{2}{log(k_1 t + k_0)}\right)\right] W_d + \frac{dv_{in}}{dt} \tag{A.28}$$

Then, we can formulate a discrete-time weight update as:

$$\begin{aligned}\frac{\triangle W_d(n)}{\triangle t} = &-k_1\left(1 + \frac{2}{\log(k_1 \triangle t n + k_0)}\right)\left(\frac{1}{k_1 \triangle t n + k_0}\right) W_d(n-1) \\ &+ \frac{\triangle v_{in}(n)}{\triangle t}\end{aligned} \tag{A.29}$$

$$\begin{aligned}W_d(n) = &\left[1 - \left(1 + \frac{2}{\log(k_1 \triangle t n + k_0)}\right)\left(\frac{1}{n + \frac{k_0}{k_1 \triangle t}}\right)\right] W_d(n-1) \\ &+ (v_{in}(n) - v_{in}(n-1))\end{aligned} \tag{A.30}$$

where n represents the number of patterns observed and Δt is the duration of the input pulse. Let us denote the weight decay term as

$$\alpha(n) = \left[1 - \left(1 + \frac{2}{\log\left(k_1 \triangle t n + k_0\right)}\right)\left(\frac{1}{n + \frac{k_0}{k_1 \triangle t}}\right)\right] \tag{A.31}$$

Thus, we obtain the weight update equation with respect to number of patterns observed as

$$W_d(n) = \alpha(n) W_d(n-1) + (v_{in}(n) - v_{in}(n-1)) \tag{A.32}$$

Then the equation can be unfolded as follows:

$$W_d(n) = \alpha(n) W_d(n-1) + (v_{in}(n) - v_{in}(n-1)) \tag{A.33}$$

$$W_d(n-1) = \alpha(n-1) W_d(n-2) + (v_{in}(n-1) - v_{in}(n-2)) \tag{A.34}$$

and so on, till ...

$$W_d(2) = \alpha(2) W_d(1) + (v_{in}(2) - v_{in}(1)) \tag{A.35}$$

$$W_d(1) = \alpha(1) W_d(0) + (v_{in}(1) - v_{in}(0)) \tag{A.36}$$

Assuming the initial condition that $W_d(0) = 0$ and $x(0) = 0$, if we multiply each $W_d(i)$ with the product of all $\alpha(i)$s succeeding it and sum them up, we get:

$$\begin{aligned} W_d(n) &= (v_{in}(n) - v_{in}(n-1)) + \alpha(n)(v_{in}(n-1) - v_{in}(n-2)) \\ &+ \alpha(n)\alpha(n-1)(v_{in}(n-2) - v_{in}(n-3)) + ... \\ &+ \alpha(n)\alpha(n-1)...\alpha(4)\alpha(3)(v_{in}(2) - v_{in}(1)) \\ &+ \alpha(n)\alpha(n-1)...\alpha(3)\alpha(2)v_{in}(1) \end{aligned} \tag{A.37}$$

This can be generalized as

$$\begin{aligned} W_d(n) &= \{v_{in}(n) + (\alpha(n) - 1)v_{in}(n-1) \\ &+ (\alpha(n-1) - 1)\alpha(n)v_{in}(n-2) \\ &+ ... \\ &+ \alpha(n)\alpha(n-1)...\alpha(3)(\alpha(2) - 1)v_{in}(1)\} \end{aligned} \tag{A.38}$$

$$W_d(n) = \sum_{i=1}^{n-2} \left\{ (\alpha(i+1) - 1) \left(\prod_{j=i+2}^{n} \alpha(j) \right) v_{in}(i) \right\} + (\alpha(n) - 1) v_{in}(n-1) + v_{in}(n) \tag{A.39}$$

Therefore, each weight $W_d(n)$ at time instance n can be represented as a summation of the product of synaptic modifications or patterns $v_{in}(n-1), v_{in}(n-2)...v_{in}(1)$ and cumulative decay rate $r_c(n, n-1), r_c(n, n-2), ...r_c(n, 1)$ for instances preceeding n as:

$$W_d(n) = \sum_{i=1}^{n-1} v_{in}(i) r_c(n, i) + v_{in}(n) \tag{A.40}$$

where

$$r_c(n, i) = (\alpha(i+1) - 1) \left(\prod_{j=i+2, j \leq n}^{n} \alpha(j) \right) \tag{A.41}$$

Then, for a network of N synapses, each indexed as $W_d(a, n)$ (where $a = 1, ..., N$), with the input applied to the a^{th} synapse after n patterns represented by $v_{in}(a, n)$, the signal strength for the p^{th} update (where $p < n$) tracked after n patterns is given by:

$$S(n, p) = \frac{1}{N} \left\langle \sum_{a=1}^{N} W_d(a, n) v_{in}(a, p) \right\rangle \tag{A.42}$$

where angle brackets denote averaging over the ensemble of all of the random uncorrelated patterns seen by the network. Since, the signal corresponding to a certain update is essentially determined by the overlap of the associated history of synaptic modifications with the present synaptic weights, by substituting A.40 into A.42, we get the signal strength of the p^{th} update as:

$$S(n, p) = \frac{1}{N} \left\langle \sum_{a=1}^{N} W_d(a, n) v_{in}(a, p) \right\rangle = r_c(n, p) = (\alpha(p+1) - 1) \prod_{j=p+2}^{n} \alpha(j) \tag{A.43}$$

Given that in A.31, $k_0 = \mathcal{O}(10^{19})$ and $k_1 = \mathcal{O}(10^{16})$, the term $\left(1 + \frac{2}{\ln(k_1 \triangle t n + k_0)}\right) \approx 1$, the above equation can be simplified as follows:

$$\begin{aligned} S(n,p) &= \frac{-1}{p+1+\gamma}\left(1 - \frac{1}{p+2+\gamma}\right)\left(1 - \frac{1}{p+3+\gamma}\right) \\ &\ldots \left(1 - \frac{1}{n-1+\gamma}\right)\left(1 - \frac{1}{n+\gamma}\right) \\ S(n,p) &= \frac{-1}{n+\gamma} \end{aligned} \tag{A.44}$$

where $\gamma = \frac{k_0}{k_1 \triangle t}$.This leads to the following expression for signal power:

$$S^2(n,p) = \frac{1}{(n+\gamma)^2} \tag{A.45}$$

By assuming that the weight $W_d(n)$ is uncorrelated from the input pattern $v_{in}(n)$ and that the inputs $v_{in}(1), v_{in}(2) \ldots v_{in}(n)$ are all uncorrelated from each other, we can obtain the noise power associated with the retrieved signal (which is essentially the variance of the retrieved signal). It is measured as the summation of the power of all signals tracked at n except for the retrieval signal of the p^{th} pattern and is expressed as:

$$\nu^2(n,p) = \frac{1}{N} \sum_{i=1, i \neq p}^{n} S^2(n,i) \tag{A.46}$$

By incorporating the retrieval signal into the summation in A.46 we can obtain a more tractable analytical expression for noise power despite the marginal error it introduces. The resulting expression is given by

$$\nu^2(n,p) = \frac{1}{N} \sum_{i=1}^{n} S^2(n,i) = \frac{n}{N(n+\gamma)^2} \tag{A.47}$$

Based on the value of n in comparison to γ, we obtain two trends for the noise profile. When $\gamma >> n$,

$$\nu(n,p) = \frac{1}{\sqrt{N}} \left(\frac{\sqrt{n}}{\gamma} \right) \tag{A.48}$$

which implies that noise increases with increase in updates initially. On the other hand, when $\gamma << n$,

$$\nu(n,p) = \frac{\sqrt{n}}{\sqrt{N}n} = \frac{1}{\sqrt{N}} \left(\frac{1}{\sqrt{n}} \right) \tag{A.49}$$

which implies that noise falls with increase in updates in the later stages. The signal-to-noise ratio (SNR) of a network of size N can then be obtained as:

$$SNR(n,p) = \sqrt{\frac{S^2(n,p)}{\nu^2(n,p)}} = \sqrt{\frac{N}{n}} \tag{A.50}$$

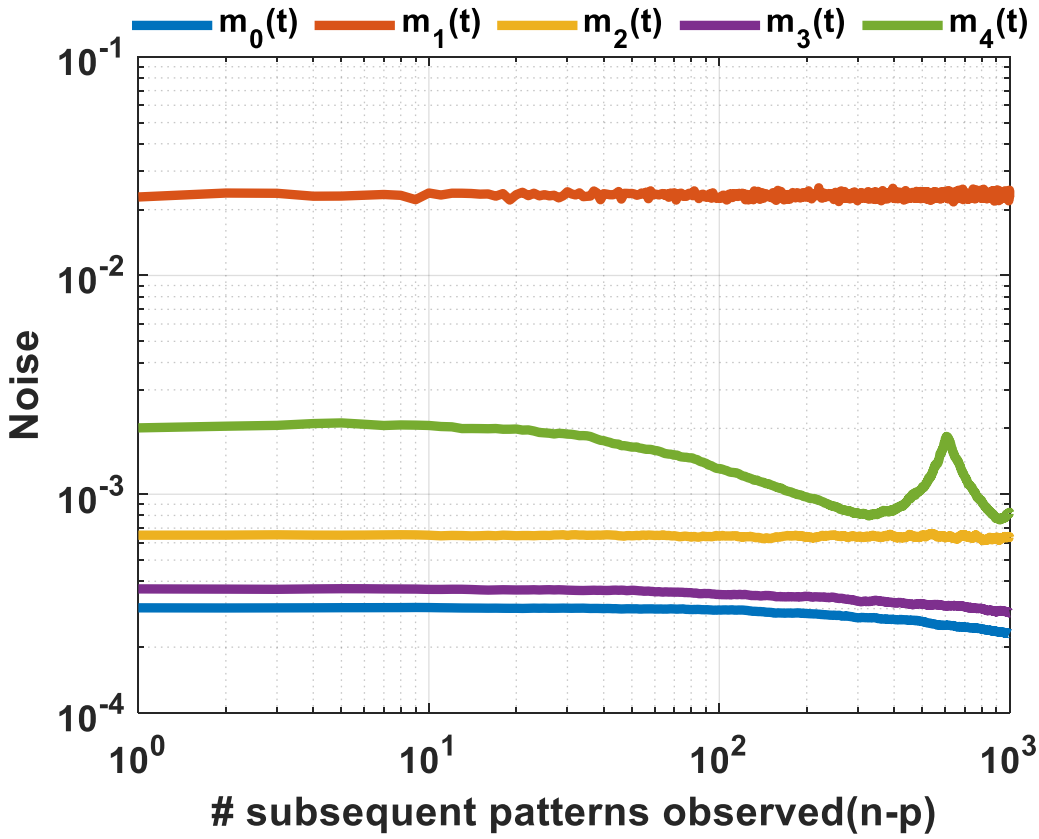

Figure A.9: Comparison of noise of FN-synapse networks composed of 1000 synapses following different synaptic models when exposed to 2000 patterns.

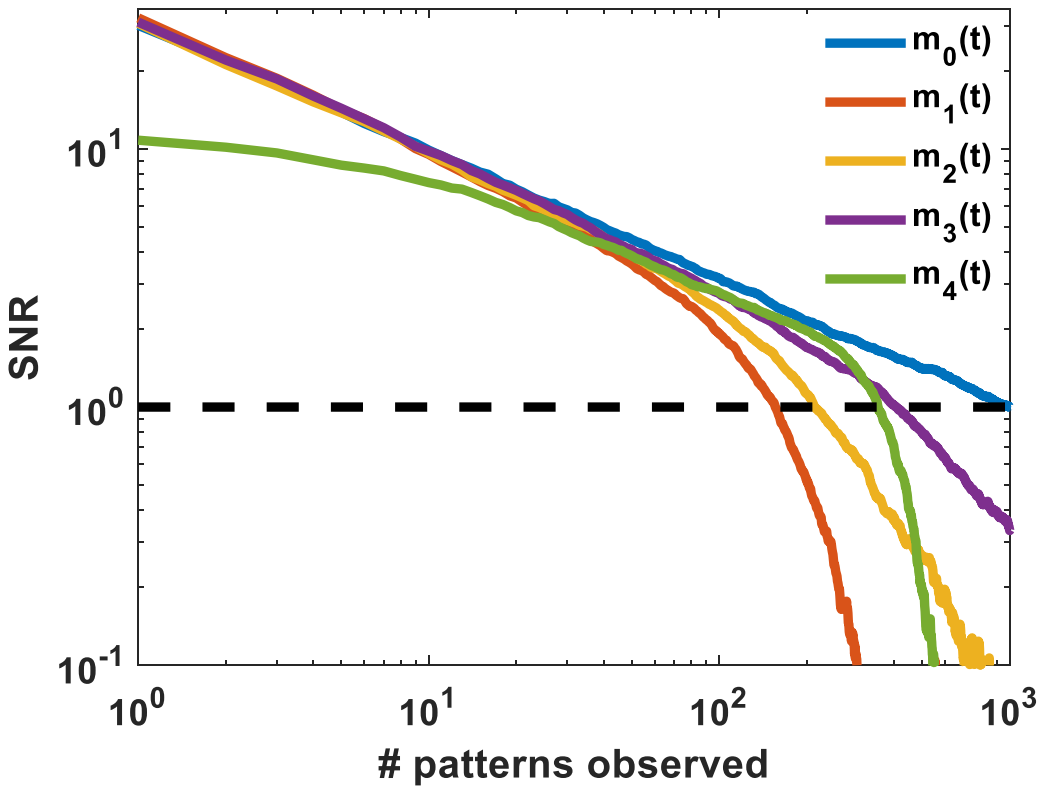

Figure A.10: Comparison of SNR of an empty network of 1000 synapses with different modulation profiles $m(t)$ when exposed to 2000 patterns.

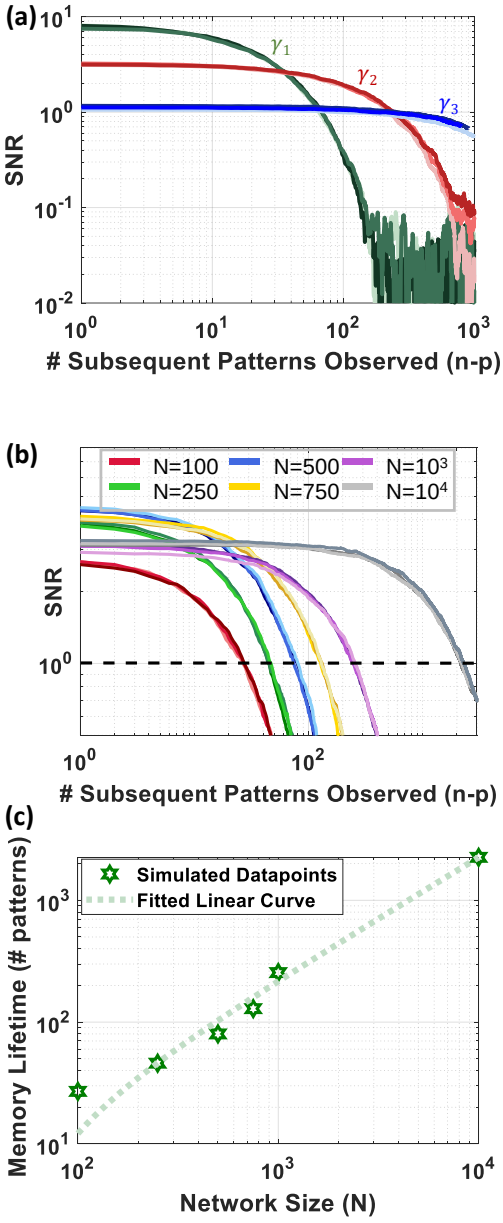

Figure A.11: (a) with different magnitude of γ where $\gamma_3 > \gamma_2 > \gamma_1$ under modulation profile of $m_2(t)$. The magnitude of γ was varied by using three different input modulation pulse width Δt.(b) Tracking the steady-state SNR of various updates (p) for FN-synapse networks of different sizes (N) with modulation profile $m_2(t)$ when exposed to subsequent updates and (c) their corresponding memory lifetime which scales linearly according to $y = mx + c$, where $m = 0.2264$ and $c = -10.46$.

Appendix B

Supplementary Information for Chapter 3

B.1 Circuit implementation and programming of FN-DAM

Each FN-DAM cell comprises two dynamical systems labeled as SET and RESET as shown in Fig. 3.1(e) and reproduced in Figure B.1. Each of these dynamical systems comprises two floating-nodes: a tunneling node (W) that stores the dynamical state; and a readout-node (W_{read}) that is used to read the dynamical state through a decoupling capacitor C_{read} and a read-out PMOS transistor M_{read}, as shown in Figure B.1. Each of these cells can be selected for READ and PROGRAMMING using the switch S_j. For this work, the selection of S_j is performed using a serial shift-register but a row-column decoder configuration could also have been used for faster access. When a cell is selected for READ using the terminal W_{out}, the read-out transistor M_{read} is configured as a source-follower using a current-source I_{read} (located in the periphery of the FN-DAM array) and the voltage of the source follower is buffered using B. The initial charge on each of the floating-nodes (SET or RESET) can be programmed individually using a combination of tunneling (to increase charge, coarse tuning) and hot-electron injection (to decrease charge, fine tuning). Details of such programming have been described in the main text and here we describe some of the programming parameters. The tunneling gate W, which stores the dynamic analog memory, is first biased in the FN-tunneling regime. By setting V_{prog} to a high-potential of 22 V (using an internal or external charge-pump), the tunneling node is pushed to $\approx 8\ V$ which is sufficient to initiate observable FN-tunneling across the gate-oxide (approximately 13 nm for work). Note that W_{read} is capacitively decoupled from the tunneling node to avert readout disturbances. The readout node is also biased at a lower voltage to prevent injection into the readout node

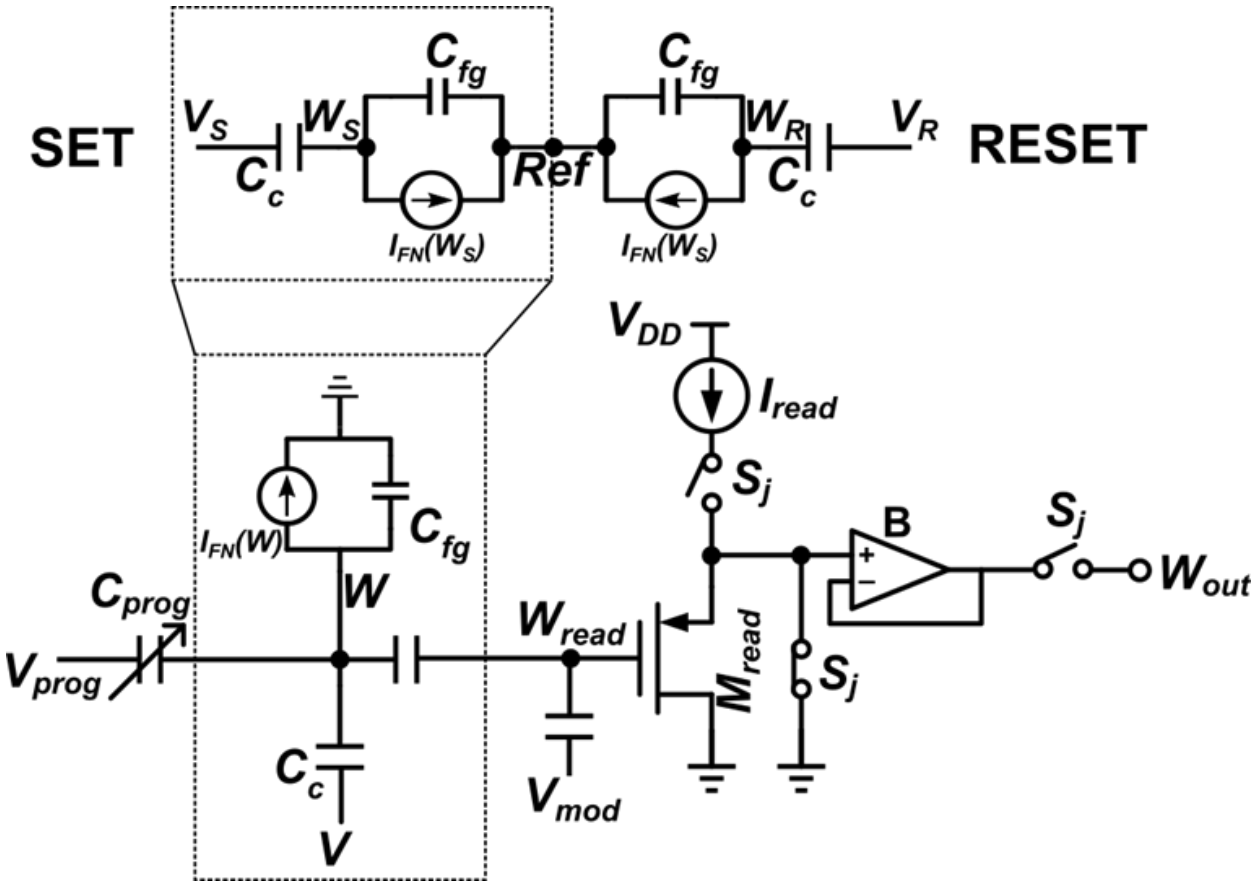

Figure B.1: Circuit implementation of the FN-DAM cell with read-out and programming circuitry

during operation. The potential of the readout node is lowered through hot-electron injection. Hot-electron injection is initiated by setting $V_{DD} = 7V$ and the input pin to a value such that the drain-to-source voltage across M_{read} is above 4.2 V. The switch S_j allows for individual control of each FN-DAM block for reading and programming.

B.2 Read-disturbance characterization

To reduce the effect of read disturbance, in our implementation, we have capacitively decoupled the readout circuit from the memory as shown in Figure B.1. We conducted read-disturbance experiments, where twelve FN-DAM memory elements were randomly accessed every minute for 1000 cycles and the relative change in weight was measured after every read. The measured

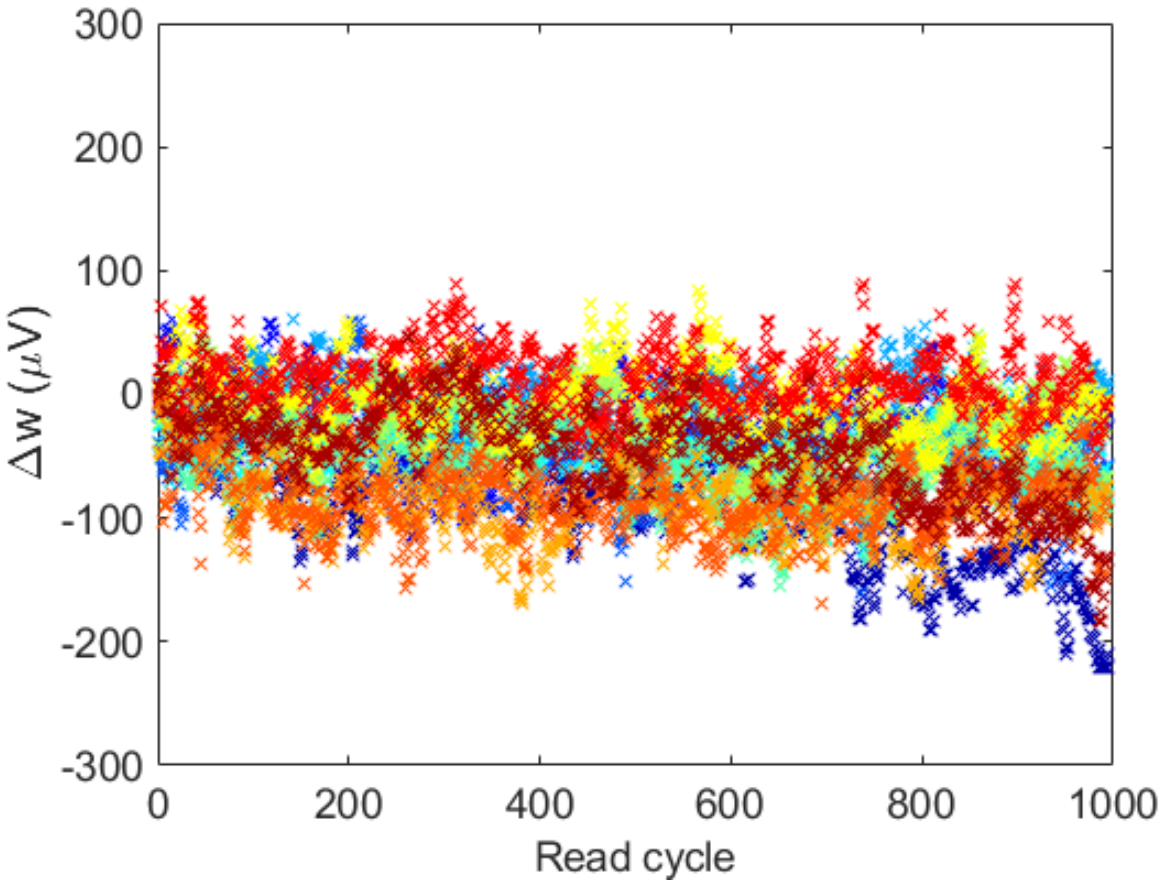

Figure B.2: Read disturbance reflected as a change in weight parameters measured from 12 FN-DAM devices over 1000 read cycles. Each color in the figure represents one FN-DAM device.

result shown in Figure. B.2 verifies that read-disturbance in our implementation of FN-DAM is random and the magnitude is less than the precision of the update and measurement.

B.3 Write Energy Dissipation Estimation

The magnitude of input pulse required, $V_{train}(t)$ (Fig. B.3a) so that the floating gate node at current potential $V_{FG}(t)$ shifts to a target voltage V_T is given by:

$$V_{train}(t) = \frac{V_T - V_{FG}(t)}{C_R} \tag{B.1}$$

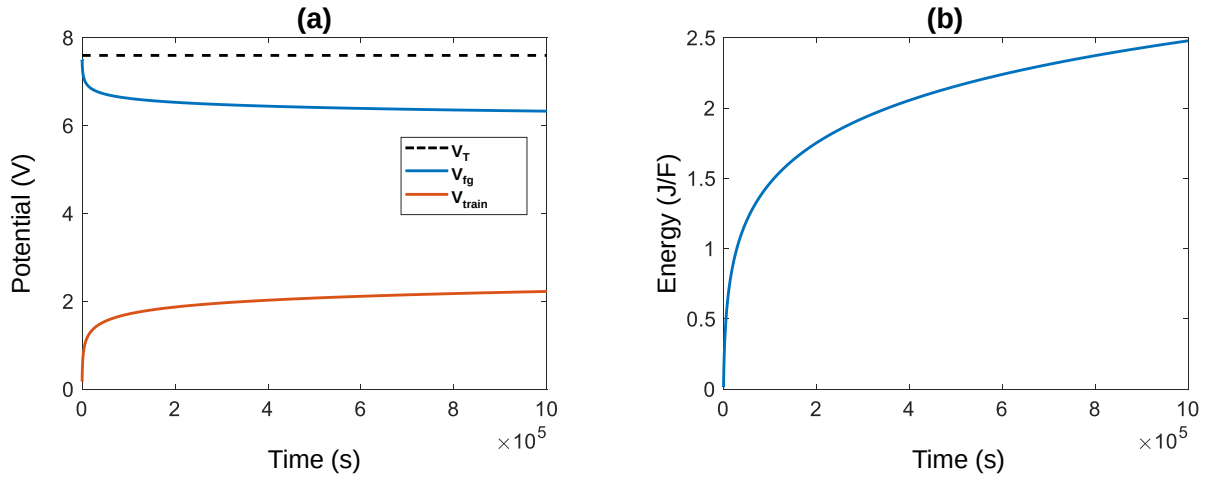

Figure B.3: Target voltage, floating gate voltage, and training voltage as a function of time. B) Energy required to charge unit capacitance as a function of time.

Where C_R is the input capacitive coupling ratio $C_R = \frac{C_C}{C_C + C_{FG}}$. The floating gate voltage $V_{FG}(t)$ is approximated by the following dynamic:

$$V_{FG}(t) = \frac{k_2}{\log(k_1 t + k_0)} \tag{B.2}$$

The energy required to charge the input capacitor is given as

$$E(t) = \frac{1}{2} C_{in} \left(V_{in}(t)\right)^2 \tag{B.3}$$

Fig. B.3b shows the instantaneous energy required to charge unit capacitance when $V_T = 7.6V$ and $V_{FG}(0) = 7.5V$. The input capacitance of our device was 1 pF, and the instantaneous write energy per update increased from 5 fJ to 2.5pJ over 12 days.

B.4 Memory Retention

Fig. B.4a shows retention times for different T (25°C, 60°C, 100°C) estimated using the retention model equations 3.22 and 3.23. These models have been verified using experiments conducted at 100^oC. Fig. B.4b shows the measured results where the weights stored in 12 FN-DAM devices kept at 100^oC were measured over a duration of 15 hours. Note that compared

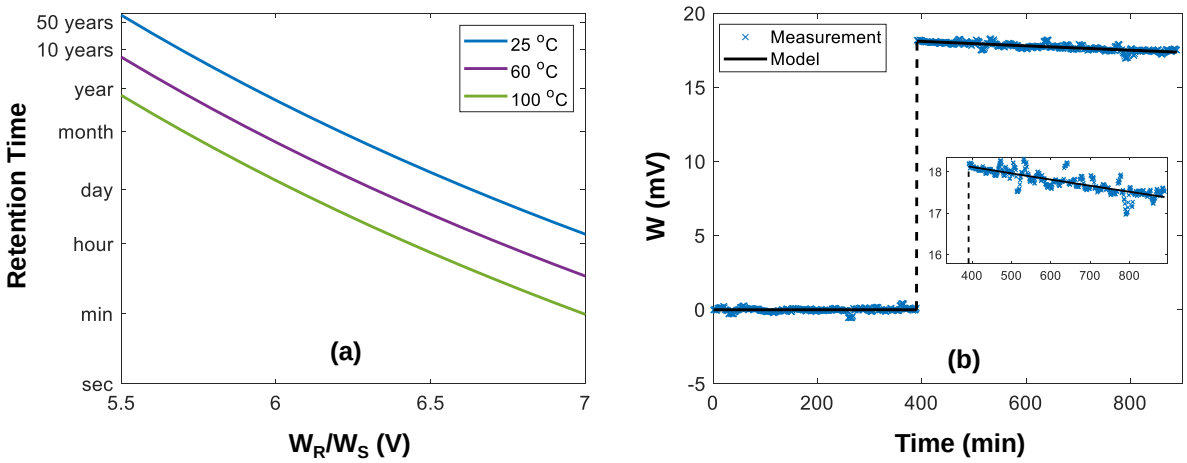

Figure B.4: a) Simulated retention time as a function of SET/RESET node voltage for different operating temperatures. b) Measurement and modeling results from 12 FN-DAM devices desynchronized at 100^oC.

to standard reliability testing of non-volatile memories where the chipsets are baked offline and then the retention measurements could be performed under standard operating temperature, for testing FN-DAM, the data needs to be measured continuously under high-temperature condition. This is because FN-DAM is a dynamical memory that stores information in the degree of temporal desynchronization between two dynamical systems. Therefore, performing continuous measurements under 250^oC operating conditions would have required the use of temperature-compensated read-out circuits. However, the required reliability information can also be inferred from continuous measurements at 100^oC, shown in Fig. B.4b. The baseline drift due to the memory read-out circuits was first calibrated during the first 400 min and used to zero out the dynamical response of each of the FN-DAM devices. Then, at 400 min time instant a SET pulse (3.3V for 1-second duration) was applied to all the memory devices which programmed all the devices to a specific memory state. The degree of desynchronization was continuously measured and is plotted in Fig. B.4b. The resynchronization process is accurately predicted by the model at 100^oC (Fig. B.4b inset)

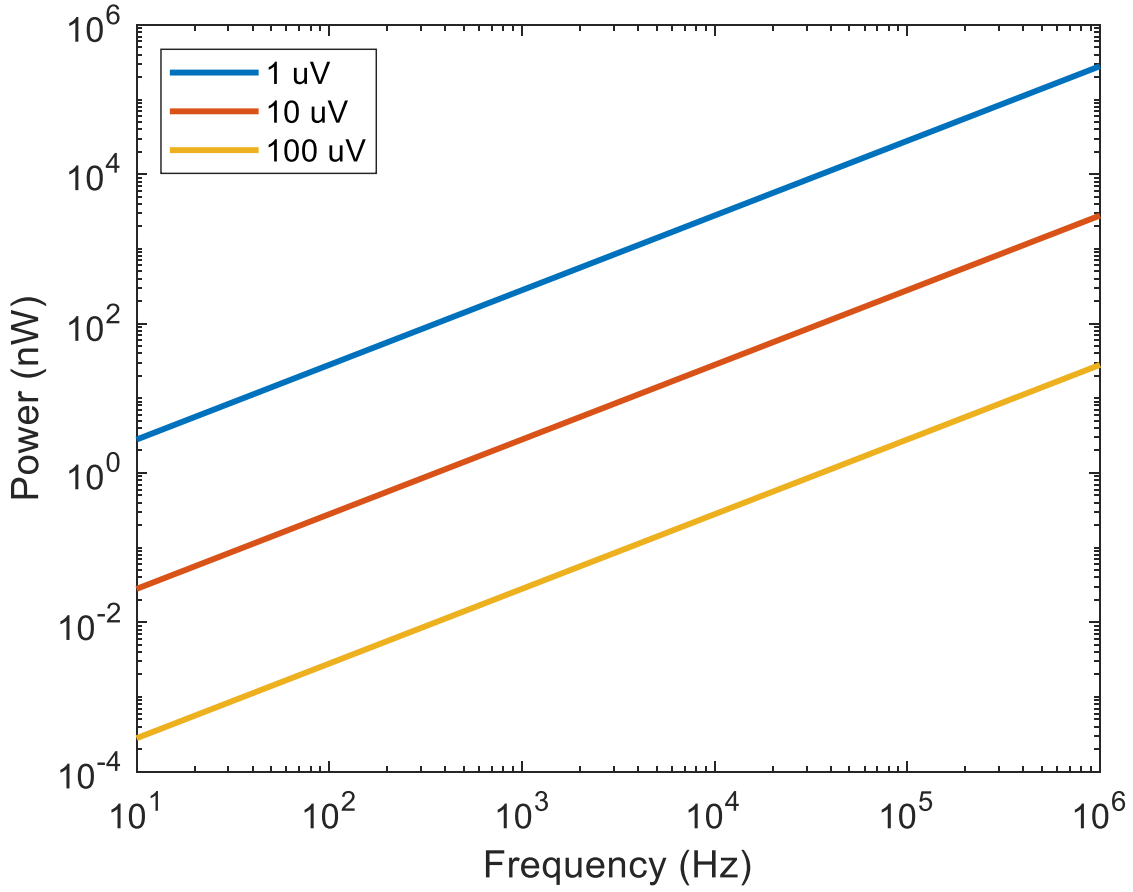

Figure B.5: Minimum power required to read floating gate voltage as a function of required readout speed. Noise floors are shown in the legend.

B.5 Read Energy Dissipation

The readout power is dependent on the readout accuracy required and the speed at which it operates. For a PMOS in a source follower configuration, the readout noise is given by:

$$V_n^2 = \frac{4kT}{g_m}\Delta f = \frac{4kT}{q} * \frac{q}{g_m}\Delta f = \frac{4U_T q}{g_m}\Delta f \tag{B.4}$$

For subthreshold operation,

$$g_m = \frac{\kappa I_d}{U_T} \tag{B.5}$$

$$\therefore V_n^2 = \frac{4U_T^2 q}{\kappa I_d}\Delta f = \frac{4U_T^2 q V_{DD}}{\kappa P_{read}}\Delta f \tag{B.6}$$

Above equation is plotted in Figure B.5 for different noise floors and readout frequency for $V_{dd} = 5V$, $U_T = 26\ mV$ and $\kappa = 0.7$

B.6 Programming dynamics

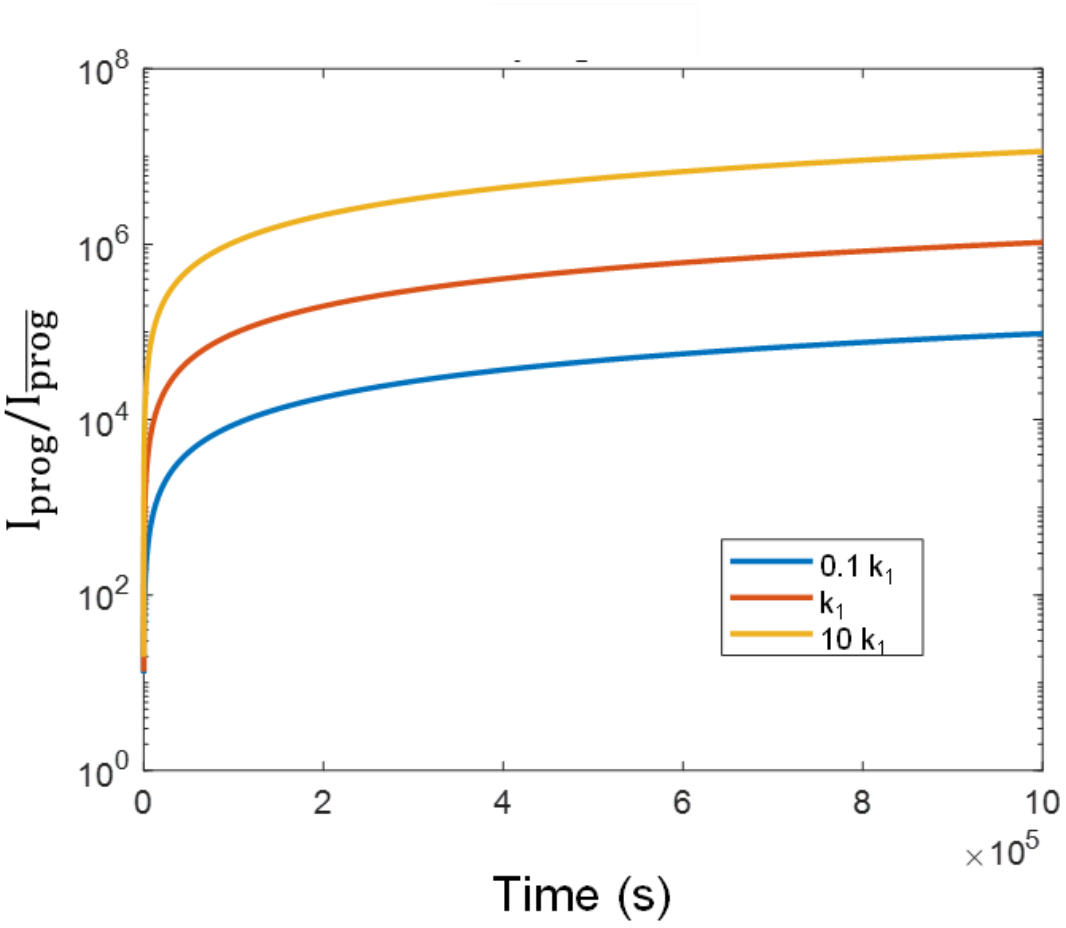

Figure B.6: Programming ratio for different k_1 parameters which can be controlled by changing the size of tunneling junction.

The FN-DAM is programmed by applying a pulse of magnitude $V_{train}(t)$ so that the node reaches a potential of V_T through the input coupling capacitor, as derived in the previous section. The programming ratio is given by:

$$\frac{I_{prog}}{I_{\bar{prog}}} = \frac{I_{FN}\left(V_T\right)}{I_{FN}\left(V_{FG}\left(t\right)\right)} \tag{B.7}$$

Dynamics of FN tunneling current are given by:

$$\frac{I_{FN}\left(V\left(t\right)\right)}{C_T} = \frac{d\left(V\left(t\right)\right)}{dt} = \left(\frac{k_1}{k_2}\right) V^2 \exp\left(-\frac{k_2}{V}\right) \tag{B.8}$$

$$\frac{I_{prog}}{I_{\bar{prog}}} = \left(\frac{V_T}{V_{FG}\left(t\right)}\right)^2 \exp\left(\frac{k_2}{V_{FG}\left(t\right)} - \frac{k_2}{V_T}\right) \tag{B.9}$$

The above equation is plotted for 3 values of k_1 in Fig B.6 which affect the dynamics of $V_{FG}\left(t\right)$. The parameter k_1 can be altered during the design phase by changing the area and capacitance of the floating gate node.

B.7 MLP and CNN architecture and training parameters

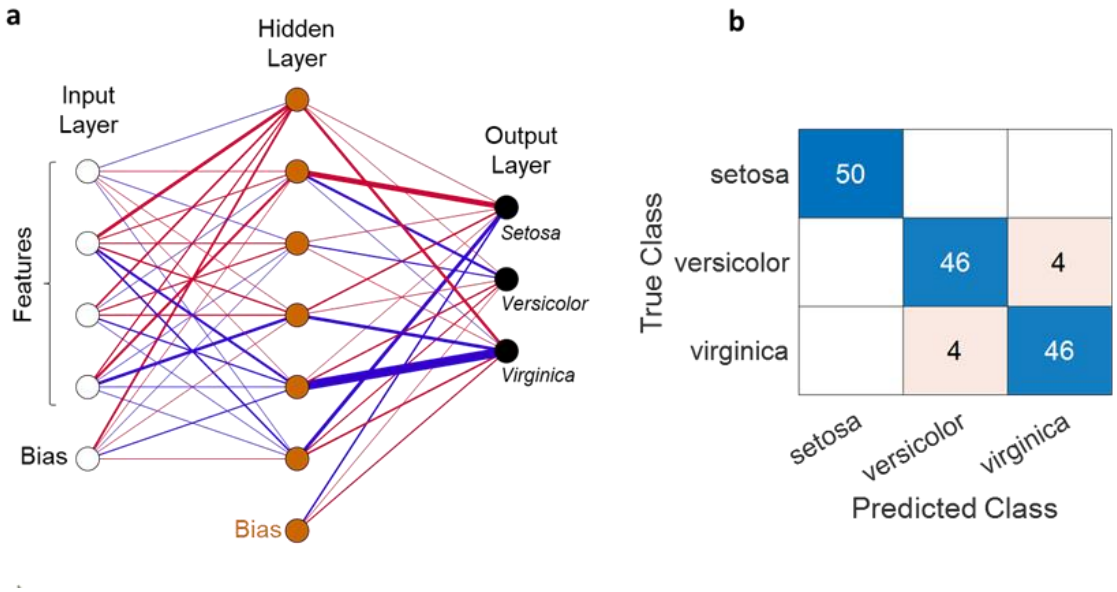

Figure B.7: a) Experimentally trained network on Fisher Iris dataset. The thickness of connections between units indicates magnitudes of learned weights. Blue (red) connection indicates positive (negative) weight. b) Confusion matrix for Fisher Iris dataset.

The neural network used for experiments with the Fisher-Iris dataset is a three-layer multilayer perceptron as shown in Fig. B.7a. which also shows the magnitude of the weights obtained after training, as indicated by the thickness of the edges between the neurons across

different layers. The blue edge represents an excitatory weight, and the red edge represents an inhibitory weight. Fig. B.7b. shows the confusion matrix computed for one training run and for the entire Fisher-Iris dataset (150 data points).

The convolutional neural network used for the MNIST experiment comprised the following layers

Layer	Name	Description	Activations	Parameters
1	Image Input	28×28×1 images	28 x 28 x 1	0
2	Convolution	20 5×5×1 convolutions with stride 1	24 x 24 x 20	520
3	Batch Normalization	Batch normalization with 20 channels	24 x 24 x 20	40
4	ReLU	ReLU	24 x 24 x 20	0
5	Convolution	20 3×3×20 convolutions with stride 1	24 x 24 x 20	3620
6	Batch Normalization	Batch normalization with 20 channels	24 x 24 x 20	40
7	ReLU	ReLU	24 x 24 x 20	0
8	Max	Pooling	12 x 12 x 20	0
9	Convolution	40 3×3×20 convolutions with stride 1	12 x 12 x 40	7240
10	preluLayer	Parametric ReLU with 40 channels	12 x 12 x 40	40
11	Convolution	20 3×3×40 convolutions with stride 1	12 x 12 x 20	7220
12	Batch Normalization	Batch normalization with 20 channels	12 x 12 x 20	40
13	ReLU	ReLU	12 x 12 x 20	0
14	Fully Connected	10 fully connected layer	10	28810
15	Softmax	softmax	10	0

The network was constructed in MATLAB using the Deep Learning toolbox and was trained using Stochastic Gradient Descent with Momentum. Only the weights in the Fully Connected

layer were updated during training. Fig. B.8 shows the confusion matrix with recognition accuracy obtained for each class of digits.

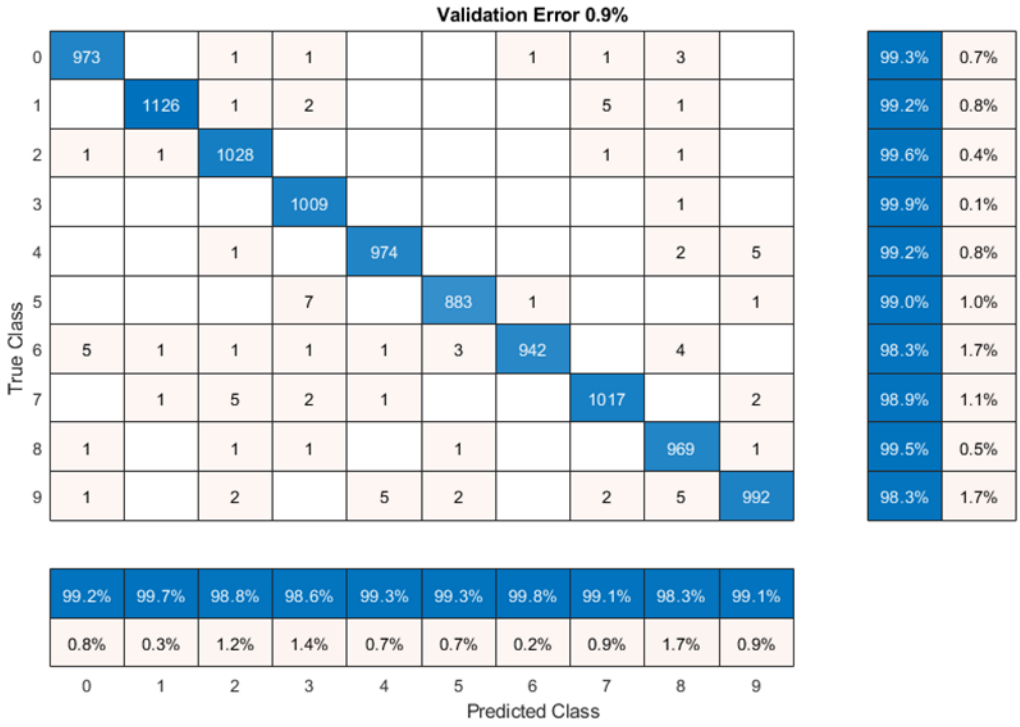

Figure B.8: Confusion matrix with for simulated CNN implementation of FN-DAM on MNIST dataset.

B.8 Retention of MLP parameters

The data retention times for FN-DAM cells (or the cells' volatility) is a smooth function with respect to the dynamic state of the memory (SET and RESET voltages). This is also shown in Fig. B.4a. During training the FN-DAMs are biased to operate in the volatile region where lower data retention is traded-off for lower energy consumption. Once the neural network has been trained, it is not necessary for the system to reach a non-volatile regime. Because of the differential architecture of the FN-DAM, both the SET and RESET nodes discharge/decay down to the slow-tunneling regime and at a discharge-rate that is approximately constant across all the memory cells. In this case, the performance of the learning algorithm (neural

network) that normalizes the weights should remain robust with minimal degradation in recognition accuracy. To verify this, we trained a neural network, shown in Fig. B.7(a) on the Fisher's Iris dataset using FN-DAM as storage for network parameters. Post-training, we transferred our chip into a baking oven which was set to 225^oC. Note that this is the maximum temperature setting for the baking oven (Quincy lab Model 40) set at 225^oC. Care was taken in transporting the chipsets to prevent any electrostatic discharge (ESD) issues. After 6 hours of baking, the chips were taken out, weights were read-out and the classification accuracy of the network was measured. SI Fig. B.9a. below compares the weights stored on the FN-DAM cells, before and after baking. The result in SI Fig. B.9b. shows that even though all the post-bake weights exhibit a decay with respect to their pre-bake values when normalized ($w_{norm} = w/||w||_1$), both the pre-bake and post-bake values remain relatively invariant.

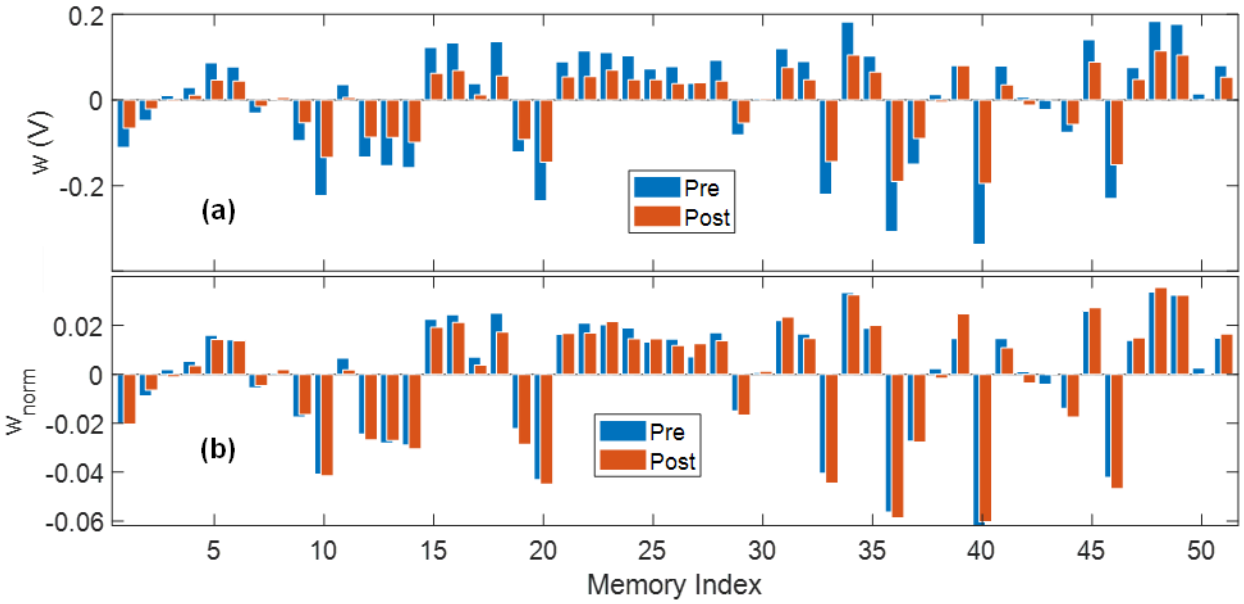

Figure B.9: (a) Weights stored on FN-DAM memory cells after training a neural network on the Fisher-Iris dataset, before baking (Pre) and after baking (Post); and (b) normalized weights before baking and after baking.

SI Fig. B.10. compares the training and test accuracy obtained using weights stored on the FN-DAM before baking and after baking. The result shows that while the training accuracy reduces nominally (97.5% to 95%), the test accuracy remains unchanged. Note that after the bake, the respective SET and RESET voltages (W_S and W_R) decay such that the FN-DAM enters the high-retention regime.

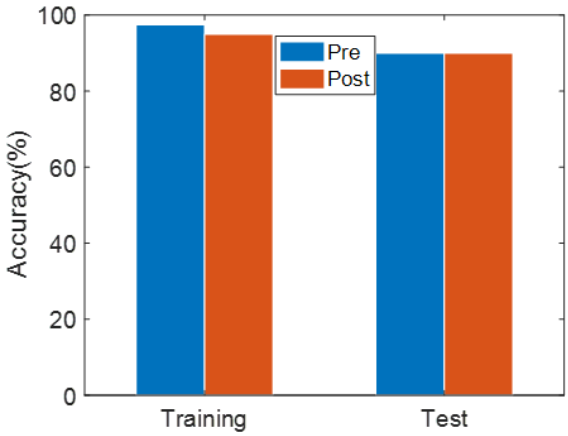

Figure B.10: Training and test accuracy obtained using the pre-bake and post-bake values of weights stored on the FN-DAM.