人工知能(機械学習）応用論Ⅱ 第13回講義資料

Hardware-Oriented Neurocomputing

(Neuromorphic Computing)

コンピュータ理工学科/コンピュータ工学部門
教授
ベンアブダラ　アブデラゼク
Ben Abdallah Abderazek

January 23, 2025, **3限目 (**12:50～14:20)

This lecture is based on the book ''Neuromorphic Computing Principles and Organization,'' Publisher: Springer; 2022 edition, ISBN-10 : 3030925242, ISBN-13 : 978-3030925246, by Abderazek Ben Abdallah, Khanh N. Dang

Lecture Contents

1. Neuromorphic Computing
2. Hardware Models of Spiking Neurons
3. Synaptic Dynamics
4. Synaptic Plasticity Mechanisms and Learning
5. Synthesizing Real-Time Neuromorphic Systems
6. Conclusions

1. Neuromorphic Computing:

What is Neuromorphic Computing?

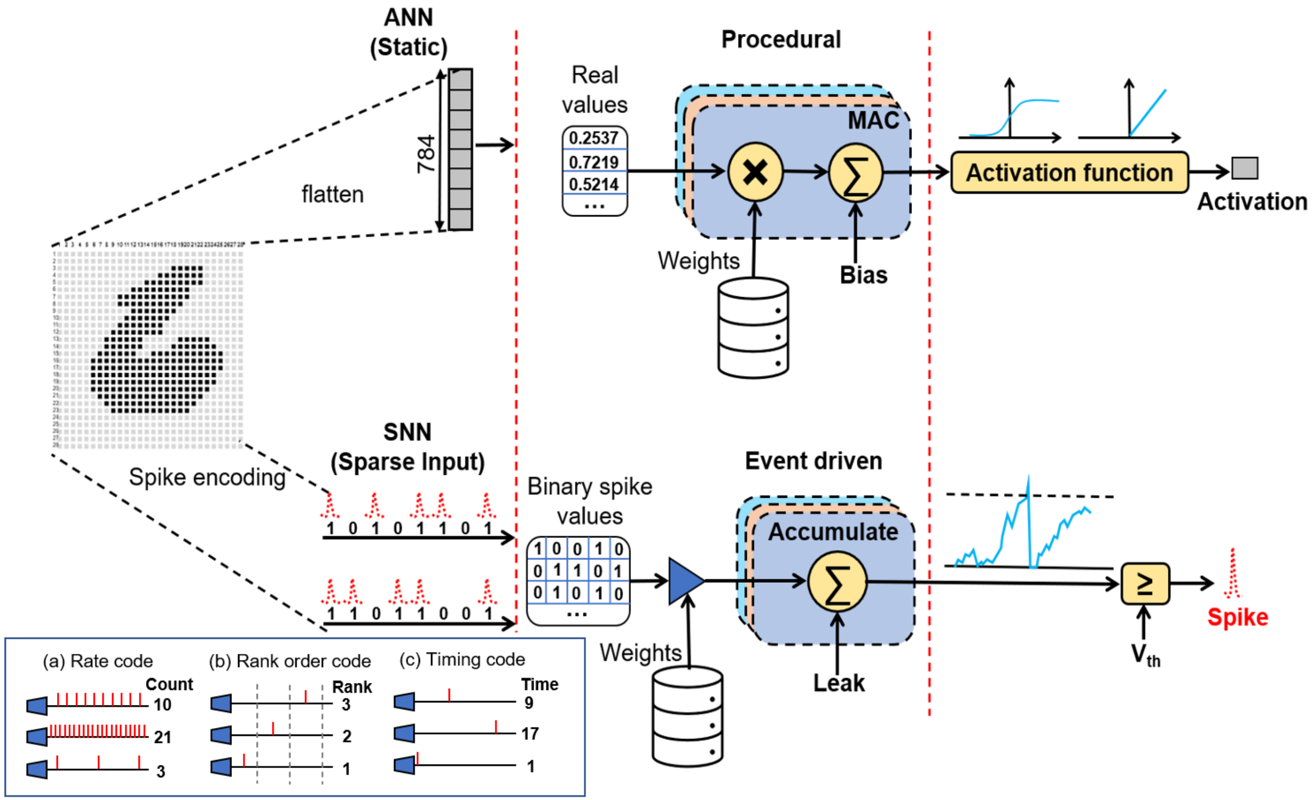
- Neuromorphic Computing (NC) is the use of hardware (VLSI) to simulate the biological architecture of the human nervous system (brain, complex network of nerves, etc.),

- NC is a new emerging field that involves **biology**, **physics**, **mathematics**, **computer science**, and **engineering** in designing hardware models **of neural and sensory systems.**

- NC opens new frontiers for neuro-robotics, artificial intelligence, and high-performance applications.

1. Neuromorphic Computing:

Conventional ANN vs Spiking Neural Networks



Conventional ANN vs Spiking Neural Network

- Sparse input in SNN means sparse memory use.
- Spike communication means minimal power per event signal
- Event based processing in SNN also contribute to low power.

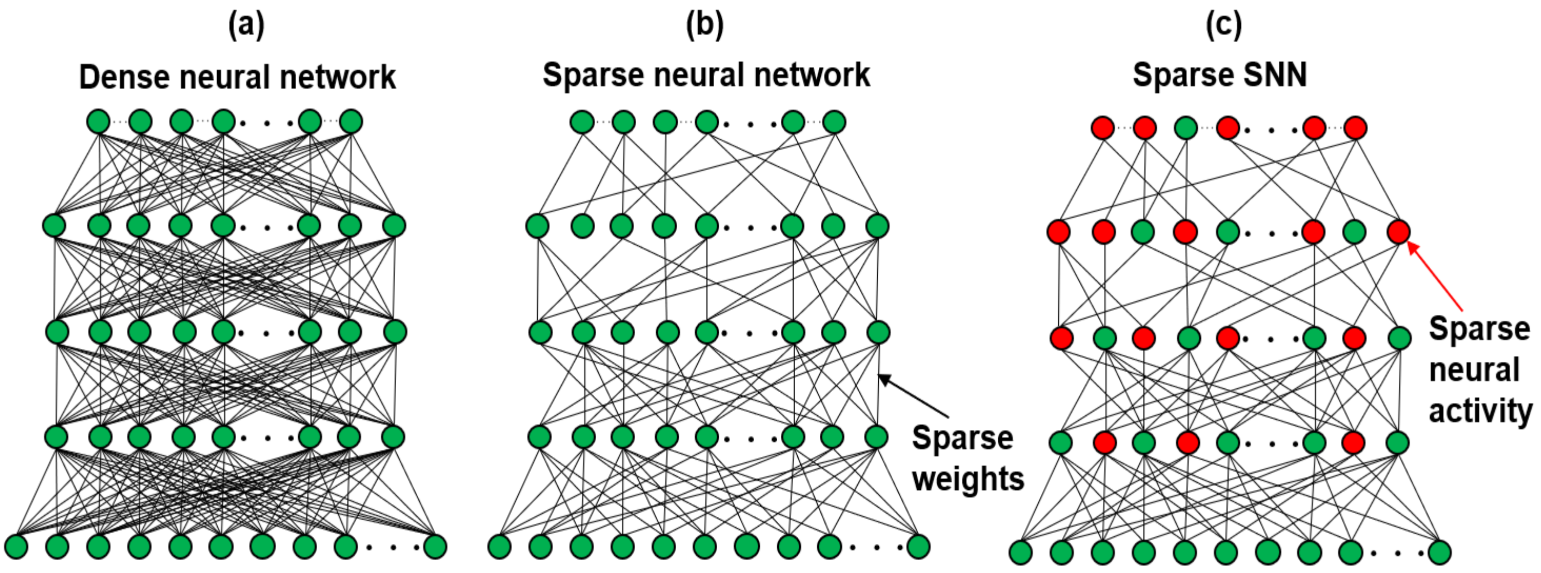
1. Neuromorphic Computing:

Conventional ANN vs Spiking Neural Networks

- Conventional ANN: Impressive results in visual and auditory cognitive applications. However, they are:
  - Slow when deployed in software, requiring a lot of time for training
  - Consume a lot of power when accelerated in hardware, requiring large servers for training as their sizes increase.
- Spiking Neural Network (Neuromorphic):
  - *More analogous to the brain*, communicating via spikes in a sparse event driven manner.
  - Exploits spike sparsity to achieve low-power.

1. Neuromorphic Computing:

Exploiting Sparsity in Neural Network

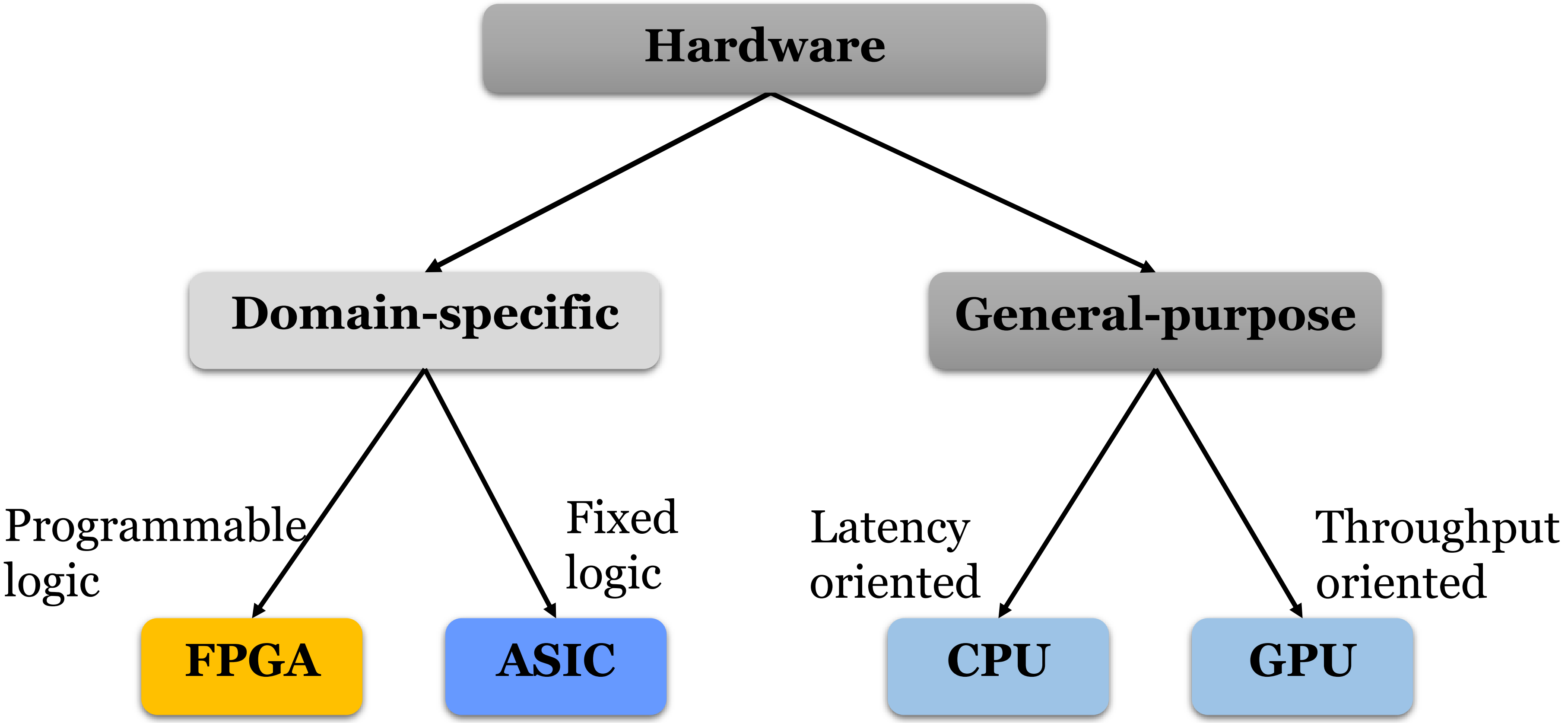


Sparsity in Neural Network

- About 0.5% to 2% of neurons in the neocortex are active at any time
- Only about 1% to 5% of connections exist between two connected layers in the neocortex and 30% of those connections change every few days

1. Neuromorphic Computing:

Neural Algorithms Computing in Hardware



Hardware

Domain-specific

General-purpose

Programmable logic

Fixed logic

Latency oriented

Throughput oriented

FPGA

ASIC

CPU

GPU

- General; requires HDL
- Moderate performance & efficiency

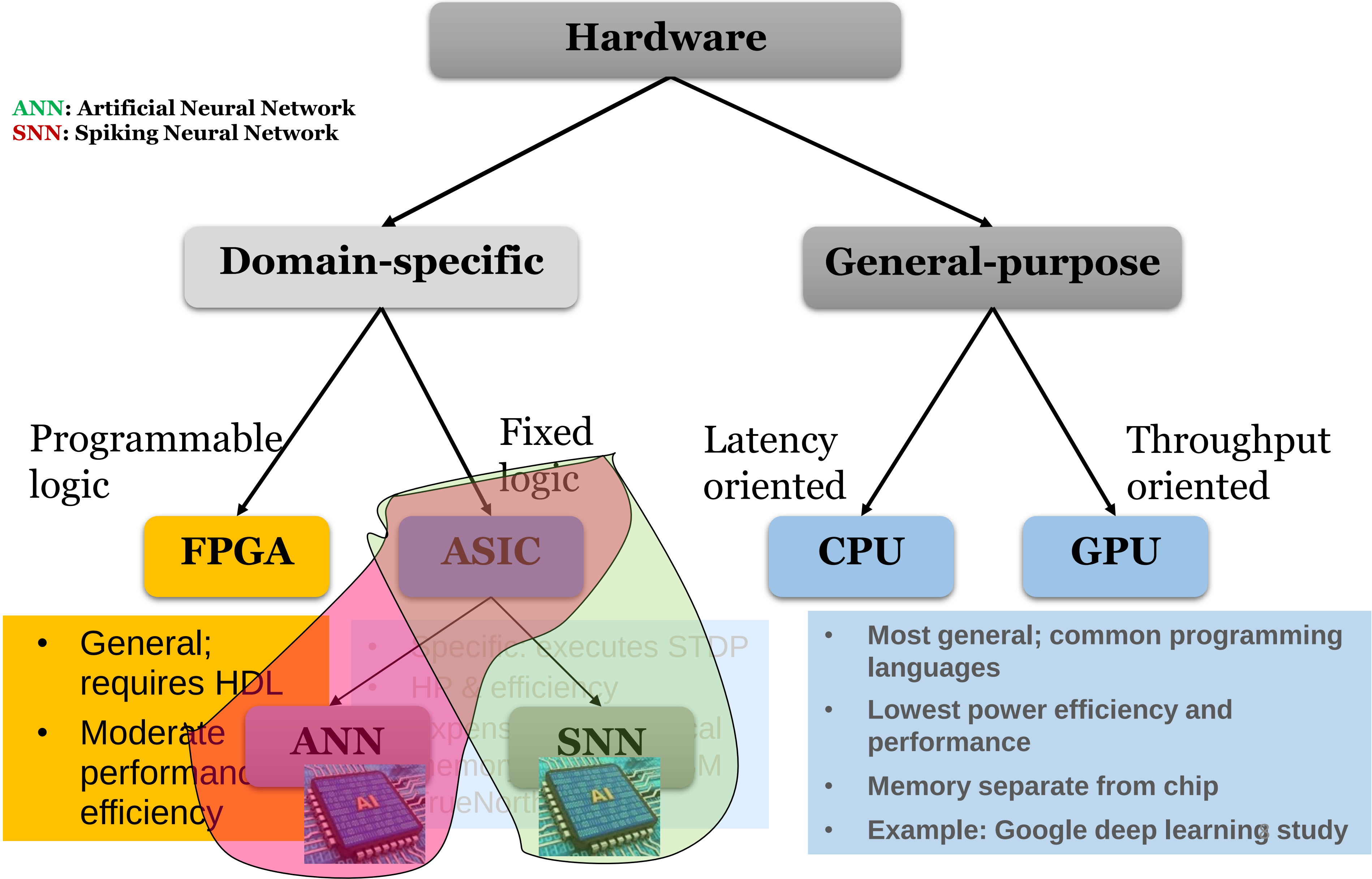
- Specific: executes STDP
- HP & efficiency
- Expensive, 40MB local memory Example: IBM TrueNorth

- Most general; common programming languages
- Lowest power efficiency and performance
- Memory separate from chip
- Example: Google deep learning study

1. Neuromorphic Computing:

Neural Algorithms Computing in Hardware

ANN: Artificial Neural Network
SNN: Spiking Neural Network



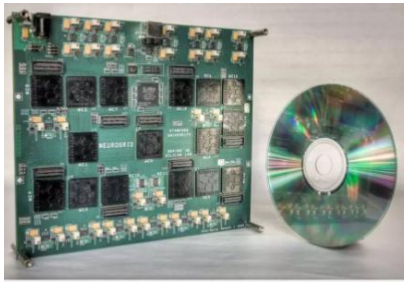
- General; requires HDL
- Moderate performance & efficiency

- Specific: executes STDP
- HP & efficiency
- Expens cal memory M
- TrueNorth

- Most general; common programming languages
- Lowest power efficiency and performance
- Memory separate from chip
- Example: Google deep learning study

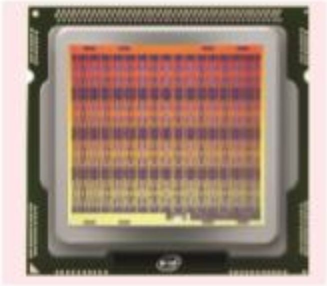
1. Neuromorphic Computing:

Examples of Neuromorphic Chips/Systems



Neurogrid

IBM TrueNorth



Intel Loihi

- The Neurogrid chip is a specialized neuromorphic hardware platform that simulates **large-scale neural networks**.
- Developed by the Brains in Silicon group at Stanford, it uses a combination of analog and digital computation to replicate the activity of biological neurons and their synaptic connections

- IBM TrueNorth is a neuromorphic chip designed to emulate the brain's neural architecture. **It features 1 million digital neurons and 256 million synapses.**
- It uses an event-driven, low-power design for efficient, scalable, and parallel processing, ideal for applications like visual recognition and sensory processing.

- Intel's Loihi is a neuromorphic chip that mimics biological neural networks. **It features 128 neuromorphic cores and on-chip learning capabilities**, making it highly efficient for adaptive AI applications.
- This chip is known for its low power and advanced spiking neural network models, enabling real-time processing and learning.

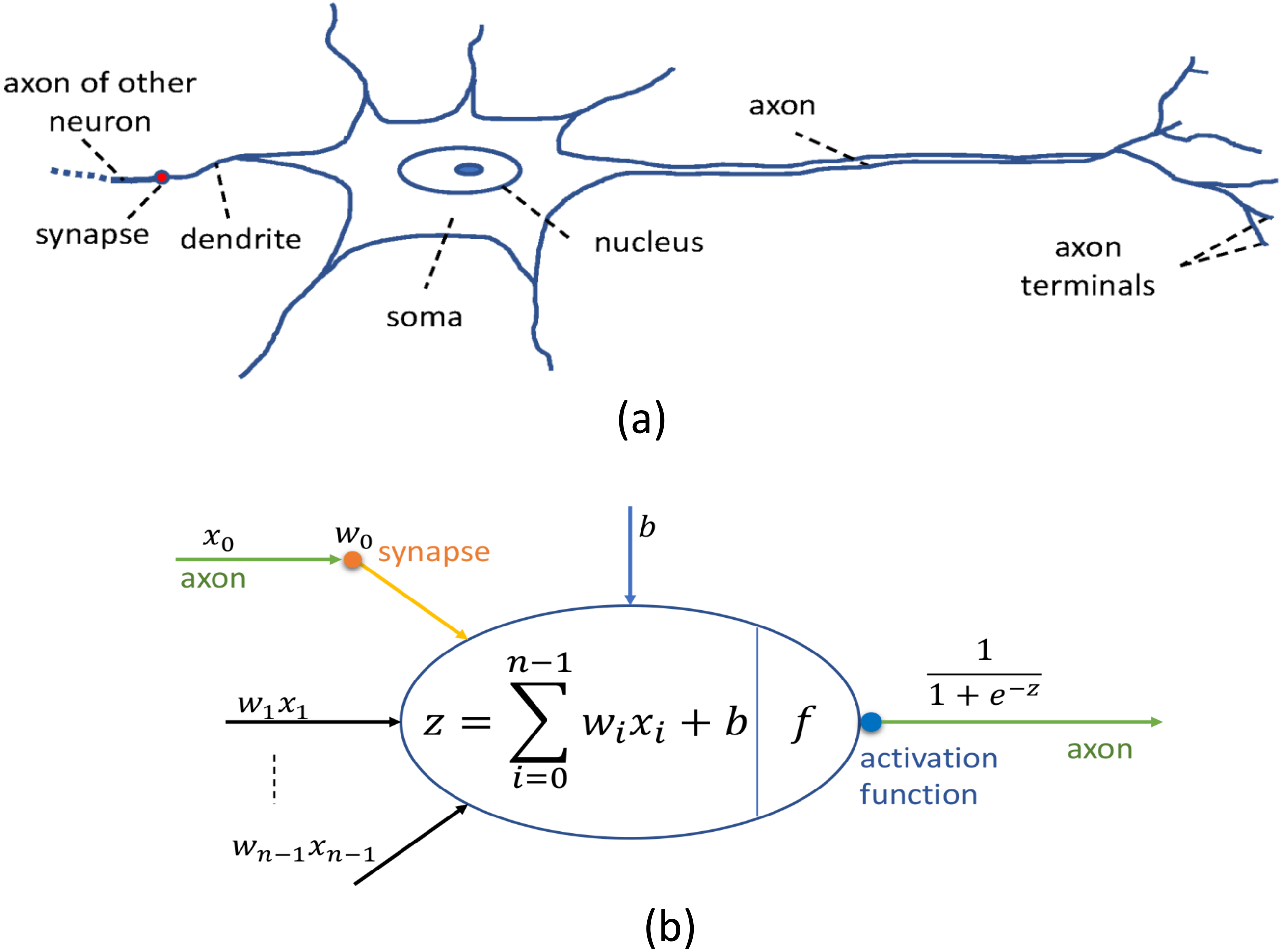
Examples of Neuromorphic Chips/Systems (not yet commercial)

Lecture Contents

1. Neuromorphic Computing Approaches
2. Hardware Models of Spiking Neurons
3. Synaptic Dynamics
4. Synaptic Plasticity Mechanisms and Learning
5. Synthesizing Real-Time Neuromorphic Systems
6. Conclusions

2. Hardware Models of Spiking Neurons:

Analogy



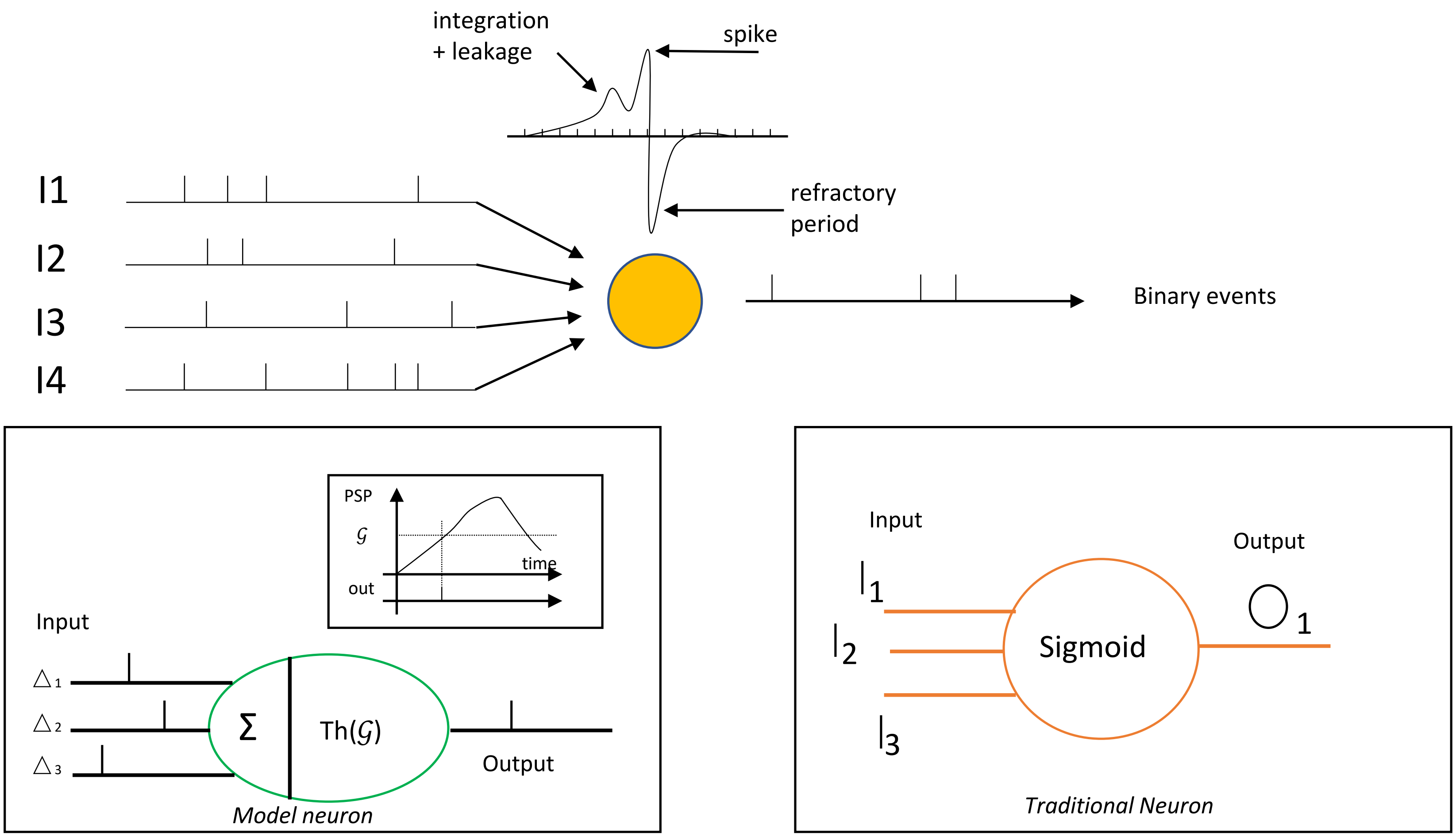
(a)

(b)

Fig. 1.1: (a) Biological neuron (b) Corresponding Mathematical model.

2. Hardware Models of Spiking Neurons:

Biological vs Traditional Neuron Models



Biological versus traditional neuron models

2. Hardware Models of Spiking Neurons:

Leaky Integrate and Fire (LIF) Neuron Model

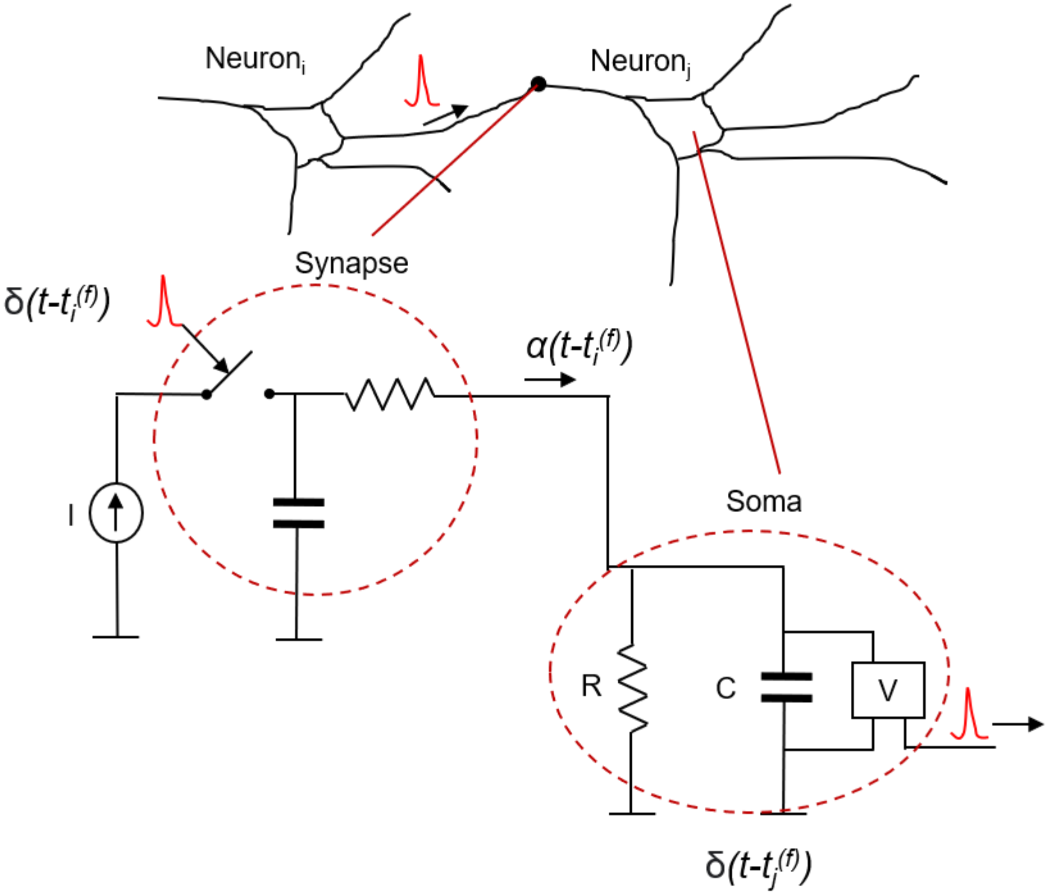


Fig. 2.11: Schematic diagram of the LIF model.

The **inter-spike interval (ISI)** refers to the time period between consecutive action potenti

2. Hardware Models of Spiking Neurons:

Spike Coding Schemes

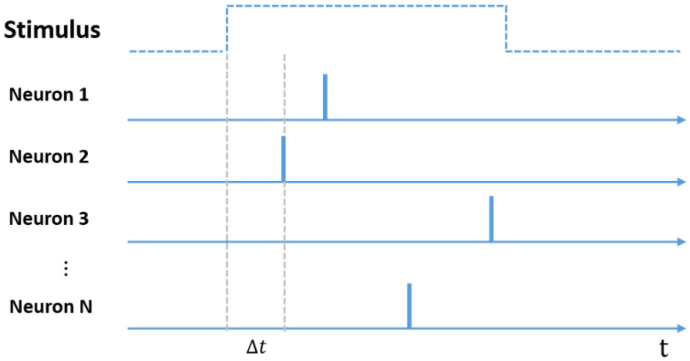


Fig. 2.2: Time to first spike

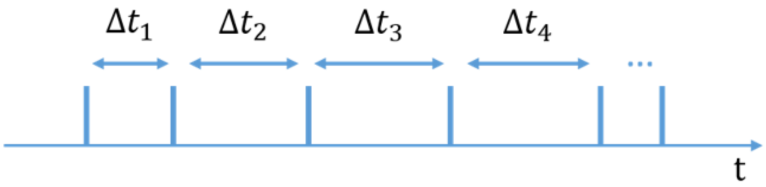


Fig. 2.3: Inter-spike-interval

- The time to first spike (TTFS) is a measure used in NC to indicate the time it takes for a neuron to fire its first action potential (spike) in response to a stimulus.

- This metric is particularly relevant in studies involving neural coding and SNNs, where the timing of the first spike can carry important information about the stimulus

- The inter-spike interval (ISI) refers to the time period between consecutive action potentials (spikes) generated by a neuron.

- It is a crucial measure in neuroscience used to analyze the firing patterns of neurons.

- By studying the distribution and variability of ISIs, we can gain insights into neuronal activity, coding mechanisms, and the overall health of neural circuits

2. Hardware Models of Spiking Neurons:

Spike Coding Schemes

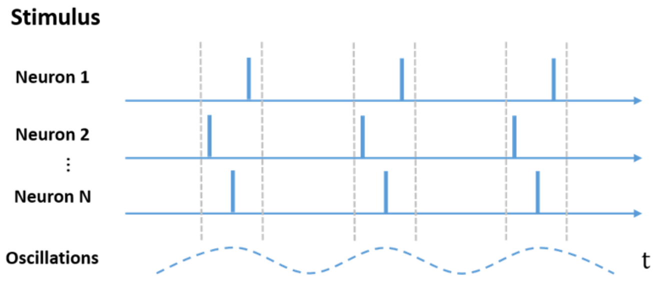


Fig. 2.4: Phase coding

- A phase coding scheme is used in neural coding where information is represented by the phase of action potentials (spikes) rather than their rate or timing alone.
- In this scheme, the timing of spikes is aligned with a reference phase, and the information is encoded in the phase relationship between spikes and this reference.
- This approach is helpful in SNNs and NC, as it can enhance the efficiency and robustness of information transmission.
- Phase coding can be more resilient to noise and improve the accuracy of NNs.

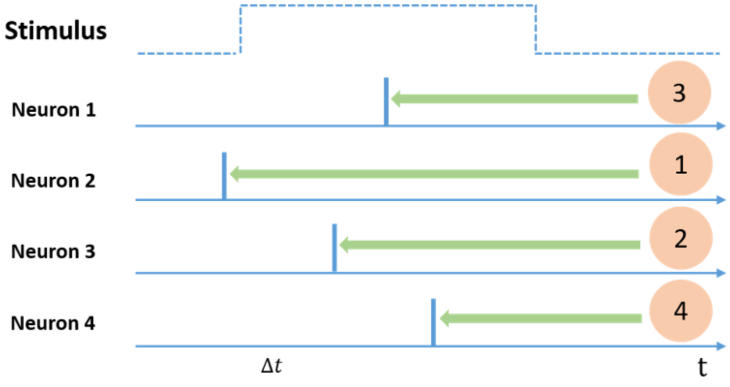


Fig. 2.5: Rank order

- Rank coding is a neural coding scheme where information is encoded based on the rank order of neuron firing rather than the precise timing of spikes.
- In this scheme, the relative timing of spikes among a group of neurons is used to represent information, which can be particularly useful in SNNs and NC.

2. Hardware Models of Spiking Neurons:

Spike Coding Schemes

Exercise 1

Consider a neuromorphic chip where neurons encode information using rate coding and temporal coding schemes.

1. **Rate Coding:** The information is represented by the firing rate of the neuron. If the firing rate of a neuron is 20 spikes per second, what is the average inter-spike interval (time between spikes)?

2. **Temporal Coding:** The information is represented by the precise timing of individual spikes. Given a pattern of spikes at the following times (in milliseconds): 0 ms, 5 ms, 10 ms, 20 ms, and 40 ms, calculate the inter-spike intervals and identify any noticeable patterns.

2. Hardware Models of Spiking Neurons:

Spike Coding Schemes

Exercise 1 Solution

1. Rate Coding:

- Firing rate = **20 spikes** per second.
- To find the average inter-spike interval, we take the inverse of the firing rate:

Average inter-spike interval = 1/(firing rate)

Average inter-spike interval = 1/ (20 spiked/second) = 0.05 seconds=50 milliseconds

Therefore, the average inter-spike interval is **50 milliseconds**.

2. Hardware Models of Spiking Neurons:

Spike Coding Schemes

Exercise 1 Solution

2. Temporal Coding:

- Given spike times: 0 ms, 5 ms, 10 ms, 20 ms, and 40 ms.
- Inter-spike intervals:

Interval 1:5ms – 0ms = 5ms
Interval 2:10ms – 5ms = 5ms
Interval 3:20 – 10ms = 10 ms
Interval 4:40 – 20ms = 20 ms

- **Noticeable patterns**: The intervals are 5 ms, 5 ms, 10 ms, and 20 ms. This pattern shows increasing intervals between spikes, representing varying information in a temporal coding scheme.
- Understanding both codings lets you appreciate how information is encoded and transmitted in neuromorphic systems.

2. Hardware Models of Spiking Neurons:

AER (Address Event Representation) Protocol

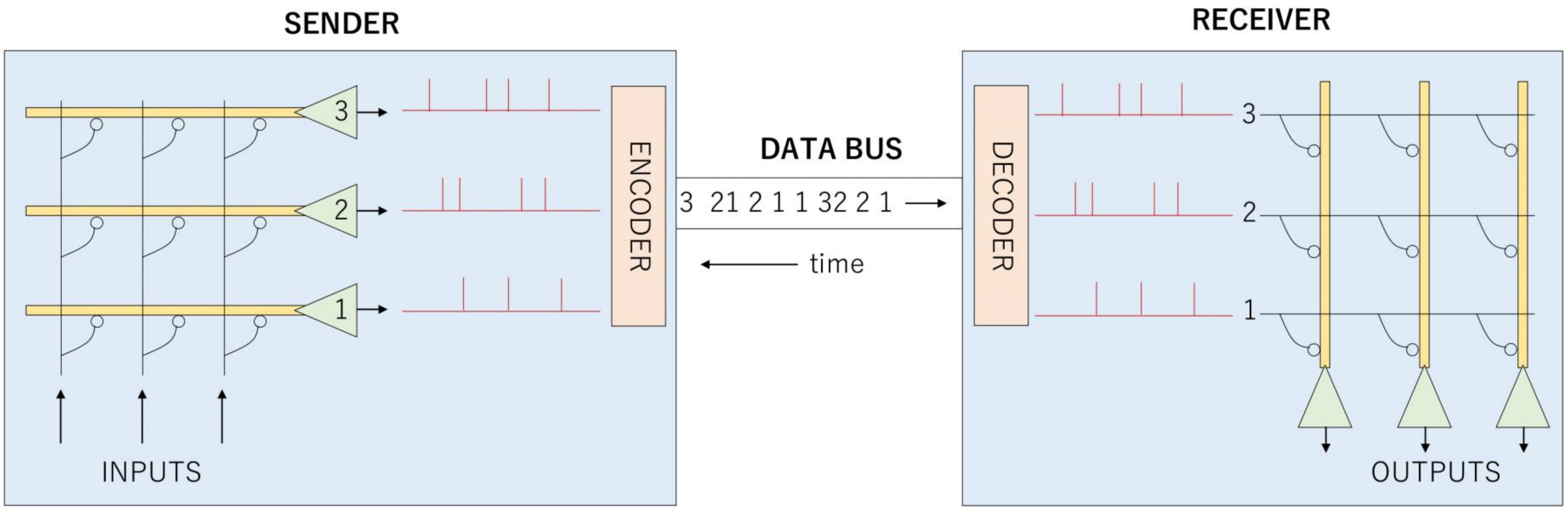


Fig. 2.15: AER (Address Event Representation) protocol

2. Hardware Models of Spiking Neurons:

AER (Address Event Representation) Protocol

Exercise 2

You have a neuromorphic chip with an Address Event Representation (AER) system. The chip has a 64x64 pixel array, where each pixel can generate an event when there is a change in light intensity. Each event packet contains the address of the pixel (row and column) and the timestamp of the event.

1. If a pixel at **row 10, column 25** generates an event at timestamp 1050, what is the AER packet format for this event?
2. How would the system handle multiple events occurring simultaneously? Provide an example with two events at the same timestamp.
3. Explain the advantages of using AER in a neuromorphic system compared to a frame-based system.

2. Hardware Models of Spiking Neurons:

AER (Address Event Representation) Protocol

Exercise 2 Solution

1. AER Packet Format:

- The address of the pixel is given by its row and column.
- The AER packet format typically includes the address and timestamp. Assuming the row and column are each represented by 6 bits (since 64 = 2^6), the packet can be structured as follows:
  - Row (6 bits): 10 -> 001010
  - Column (6 bits): 25 -> 011001
  - Timestamp (e.g., 32 bits): 1050 -> 00000000000000000000010000011010

Therefore, the AER packet format would be:

Row (6 bits)	Column (6 bits)	Timestamp (32 bits)
001010	011001	00000000000000000000010000011010

2. Hardware Models of Spiking Neurons:

AER (Address Event Representation) Protocol

Exercise 2 Solution

2. Handling Multiple Events Simultaneously:

- If multiple events occur simultaneously, the AER system can still process them sequentially. The events are buffered and transmitted in quick succession.
- Example: Two events at the same timestamp (1050) for pixels at (10, 25) and (30, 45):

- Event 1:
  - Row: 10 -> 001010
  - Column: 25 -> 011001
  - Timestamp: 1050 -> 00000000000000000000010000011010

- Event 2:
  - Row: 30 -> 011110
  - Column: 45 -> 101101
  - Timestamp: 1050 -> 00000000000000000000010000011010

The AER packets would be:

Event 1: | 001010 | 011001 | 00000000000000000000010000011010 |
Event 2: | 011110 | 101101 | 00000000000000000000010000011010 |

2. Hardware Models of Spiking Neurons:

AER (Address Event Representation) Protocol

Exercise 2 Solution

2. Advantages of Using AER in a Neuromorphic System:

- **Efficiency:** AER only transmits data when an event occurs, reducing the amount of data processed and transmitted compared to frame-based systems that send the entire frame regardless of changes.
- **Low Power Consumption:** Since AER systems process fewer data points, they consume less power.
- **Low Latency:** AER provides real-time processing capabilities as events are processed as they occur, allowing for faster response times in dynamic environments.
- **Sparse Data Representation:** AER efficiently handles sparse data, where only a small fraction of the pixels change over time, optimizing storage and computation.

Lecture Contents

1. Neuromorphic Computing Approaches
2. Hardware Models of Spiking Neurons
3. Synaptic Dynamics
4. Synaptic Plasticity Mechanisms and Learning
5. Synthesizing Real-Time Neuromorphic Systems
6. Conclusions

3. Synaptic Dynamics:

Complex Structure of a Neural Network

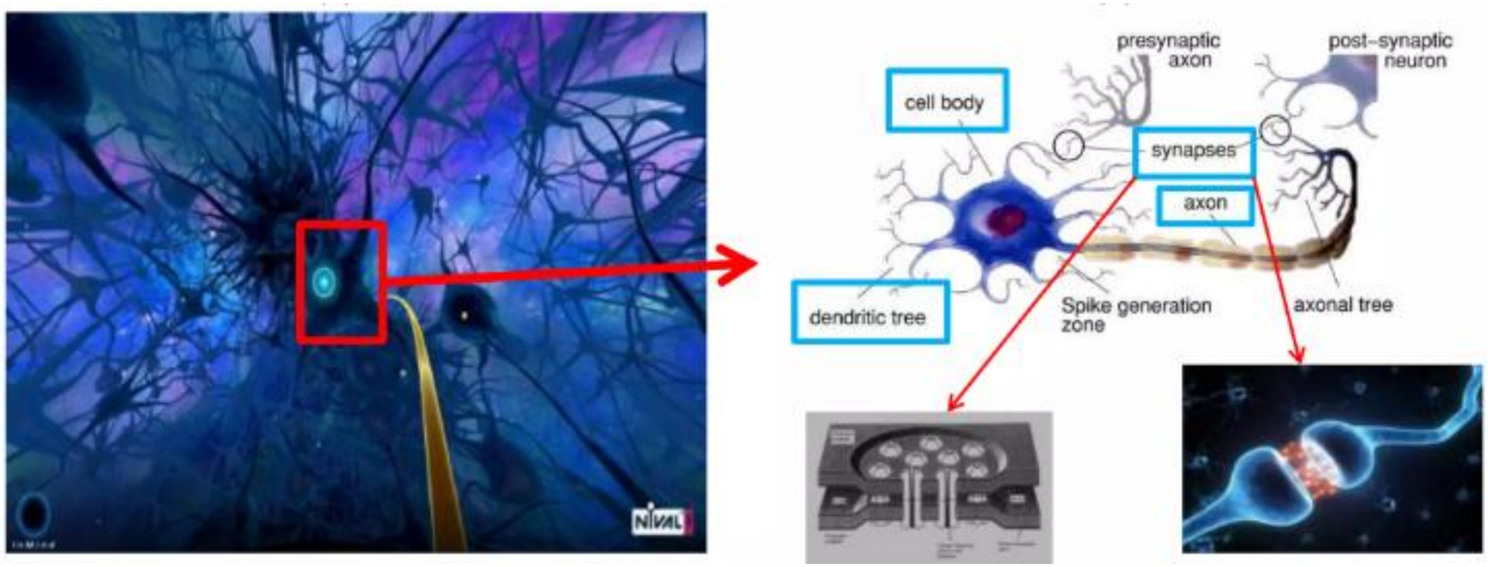


Fig. Complex Structure of a Neural Network [M.Bertrand,2015].

- A typical neural network has four main regions: The cell body, the **dendrites**, The **axon**, and the **presynaptic terminals**.

- Each region has a distinct role in the generation of signals and the communication between neurons.

- Neurons can communicate through electrical synapses or chemical synapses alone or via both types of interactions.

3. Synaptic Dynamics:

What is Synaptic Dynamics?

- Connections between neurons are not static, but change in amplitude and timing.
- Synaptic dynamics is the time-dependent changes in synaptic currents that change the strength of coupling between neurons.
- Both presynaptic and postsynaptic contribute to the changes of synaptic currents.
- Synaptic dynamics realizes **adaptive learning**.

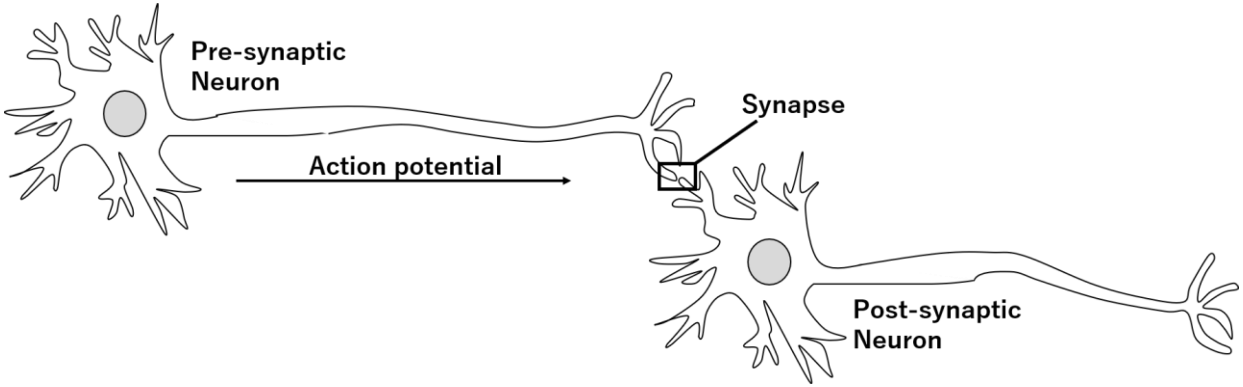


Fig. 2.1: Two neurons communicating via a synapse.

Lecture Contents

1. Neuromorphic Computing Approaches
2. Hardware Models of Spiking Neurons
3. Synaptic Dynamics
4. Synaptic Plasticity Mechanisms and Learning
5. Synthesizing Real-Time Neuromorphic Systems
6. Conclusions

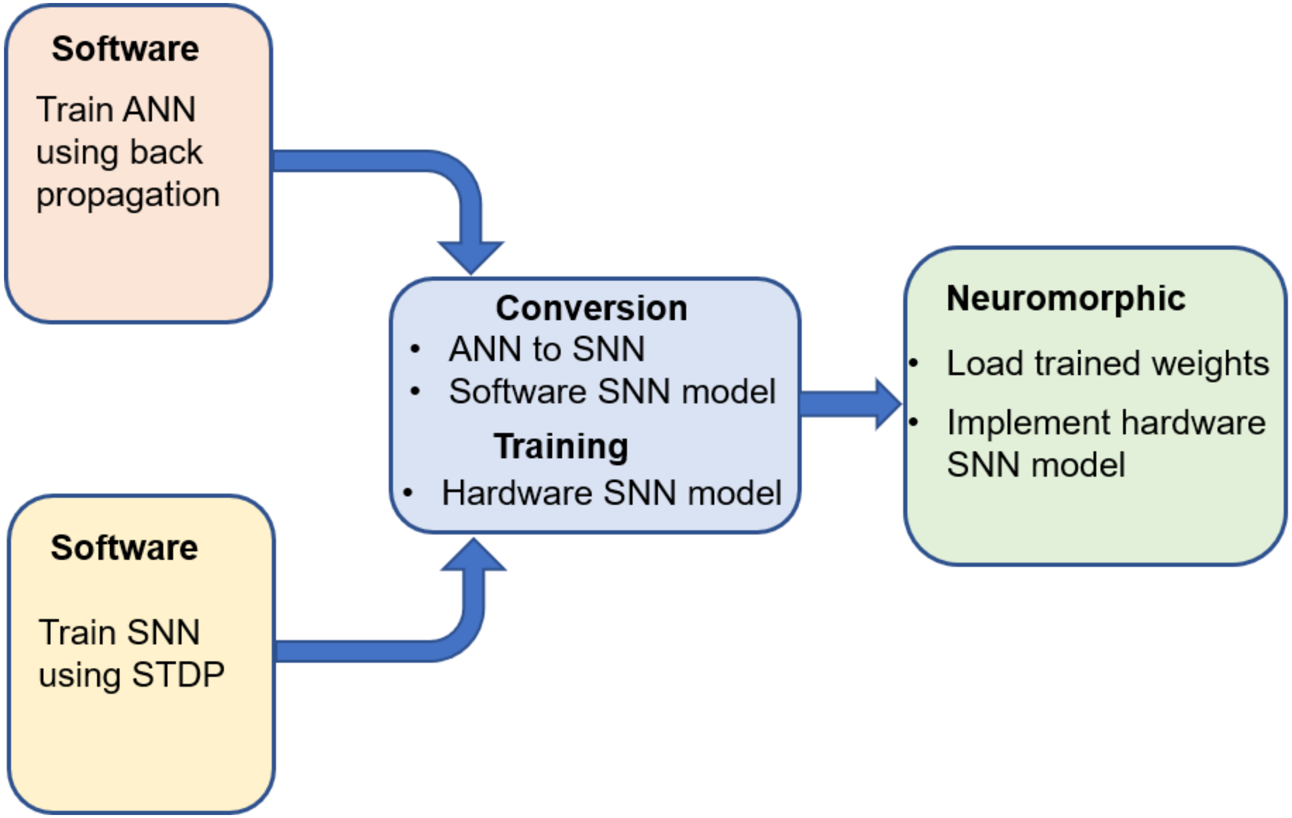
4. Synaptic Plasticity Mechanisms & Learning:

Learning Methods

- **Synaptic Plasticity:** STDP is a mechanism that models synaptic plasticity, which is the ability of synapses to strengthen or weaken over time in response to activity levels.
- **Temporal Coding:** STDP captures the importance of the precise timing of spikes in neural coding. The timing relationship between pre-synaptic and post-synaptic spikes determines whether the synaptic connection is strengthened or weakened.
- **Hebbian Learning:** STDP follows the principle of Hebbian learning, often summarized as "cells that fire together wire together." When a pre-synaptic neuron consistently fires shortly before a post-synaptic neuron, the synaptic connection is strengthened (long-term potentiation), and when it fires shortly after, the connection is weakened (long-term depression).

4. Synaptic Plasticity Mechanisms & Learning:

Learning Methods



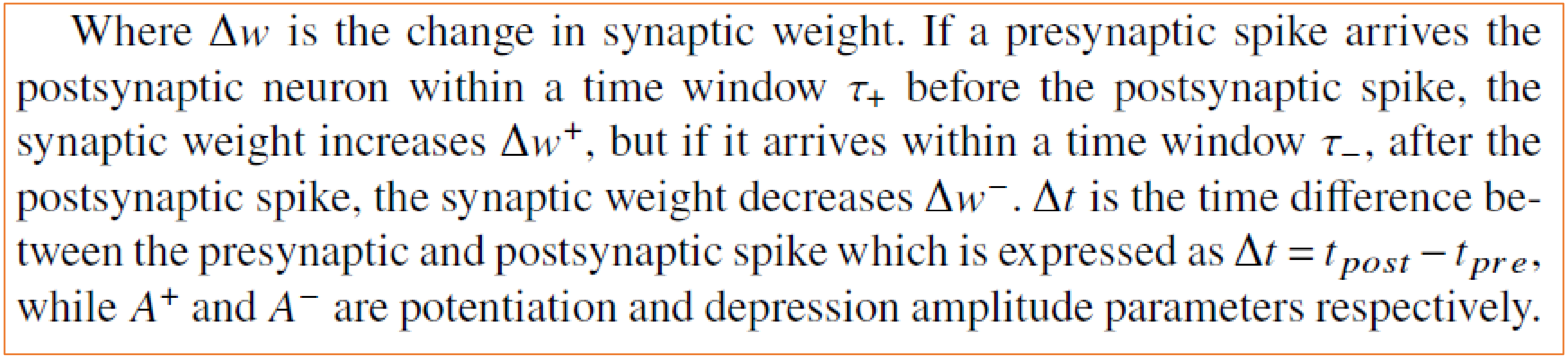
Neuromorphic Learning Framework

4. Synaptic Plasticity Mechanisms & Learning:

Spike-timing-dependent plasticity (STDP)

STDP: adjusts the connection strengths based on the relative timing of a particular neuron's output and input action potentials.

$$\Delta w = \begin{cases} \Delta w^{+} = A^{+} e^{\left(\frac{-\Delta t}{\tau_{+}}\right)}, & \text{if } \Delta t > 0 \\ \Delta w^{-} = -A^{-} e^{\left(\frac{\Delta t}{\tau_{-}}\right)}, & \text{if } \Delta t \leq 0 \end{cases}$$



Where Δw is the change in synaptic weight. If a presynaptic spike arrives the postsynaptic neuron within a time window τ_{+} before the postsynaptic spike, the synaptic weight increases Δw^{+}, but if it arrives within a time window τ_{-}, after the postsynaptic spike, the synaptic weight decreases Δw^{-}. Δt is the time difference between the presynaptic and postsynaptic spike which is expressed as $\Delta t = t_{post} - t_{pre}$, while A^{+} and A^{-} are potentiation and depression amplitude parameters respectively.

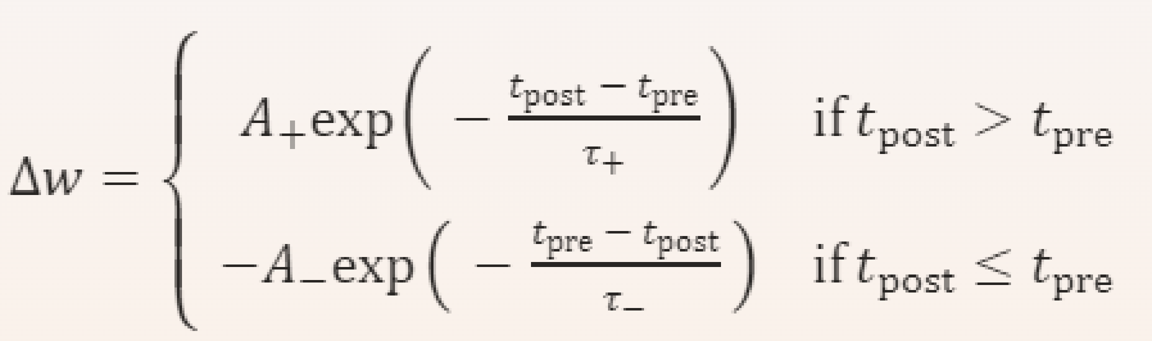
4. Synaptic Plasticity Mechanisms & Learning:

Spike-timing-dependent plasticity (STDP)

Exercise 3

Consider a neuromorphic chip implementing the STDP learning algorithm. In this algorithm, the weight of the synaptic connection between two neurons is adjusted based on the timing difference between the pre-synaptic and post-synaptic spikes.

1. If a pre-synaptic spike occurs at time $t_{pre} = 10$ ms and a post-synaptic spike occurs at time $t_{post}=12$ ms, calculate the change in synaptic weight Δw using the STDP rule. Assume the STDP learning rule is defined as:



$$\Delta w = \begin{cases} A_{+}\exp\left(-\frac{t_{\text{post}} - t_{\text{pre}}}{\tau_{+}}\right) & \text{if } t_{\text{post}} > t_{\text{pre}} \\ -A_{-}\exp\left(-\frac{t_{\text{pre}} - t_{\text{post}}}{\tau_{-}}\right) & \text{if } t_{\text{post}} \leq t_{\text{pre}} \end{cases}$$

Where A+ = 0.005, A− = 0.005 , τ+=20 ms and τ−=20 ms.

A+ and A- are potentiation and depression amplitude parameters, respectively.

4. Synaptic Plasticity Mechanisms & Learning:

Spike-timing-dependent plasticity (STDP)

Exercise 3 Solution

1. **Calculating the change in synaptic weight Δw**

- Given: tpre=10 ms, tpost=12 ms, A+=0.005, A−=0.005, τ+=20 ms, τ-=20 ms

  Since tpost > tpre, we use the first case of the STDP rule:

$$\Delta w = A_{+} \exp\left(- \frac{t_{\text{post}} - t_{\text{pre}}}{\tau_{+}} \right)$$

A+ and A- are potentiation and depression amplitude parameters, respectively.

Substituting the values:

$$\Delta w = 0.005 \exp\left(- \frac{12\ \text{ms} - 10\ \text{ms}}{20\ \text{ms}} \right)$$

$$\Delta w = 0.005 \exp\left(- \frac{2\ \text{ms}}{20\ \text{ms}} \right)$$

$$\Delta w = 0.005 \exp\left(- 0.1 \right)$$

$$\Delta w \approx 0.005 \times 0.9048$$

$$\Delta w \approx 0.004524$$

Therefore, the change in synaptic weight Δw is approximately 0.004524.

4. Synaptic Plasticity Mechanisms & Learning:

Spike-timing-dependent plasticity (STDP)

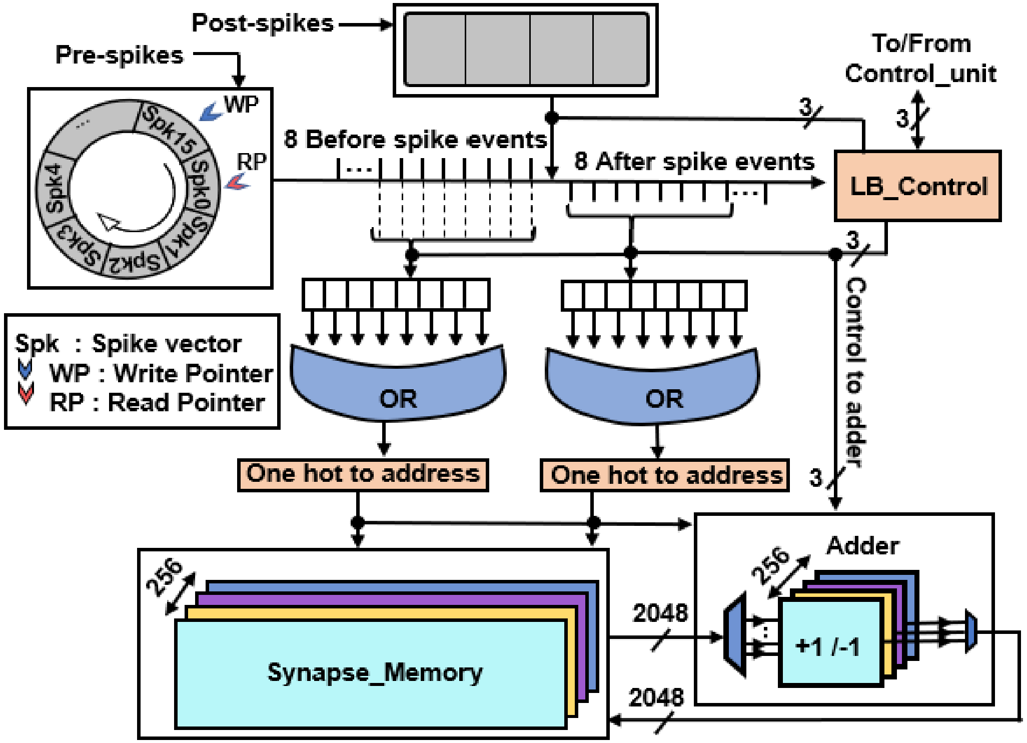


Fig. 1.5: STDP Architecture.

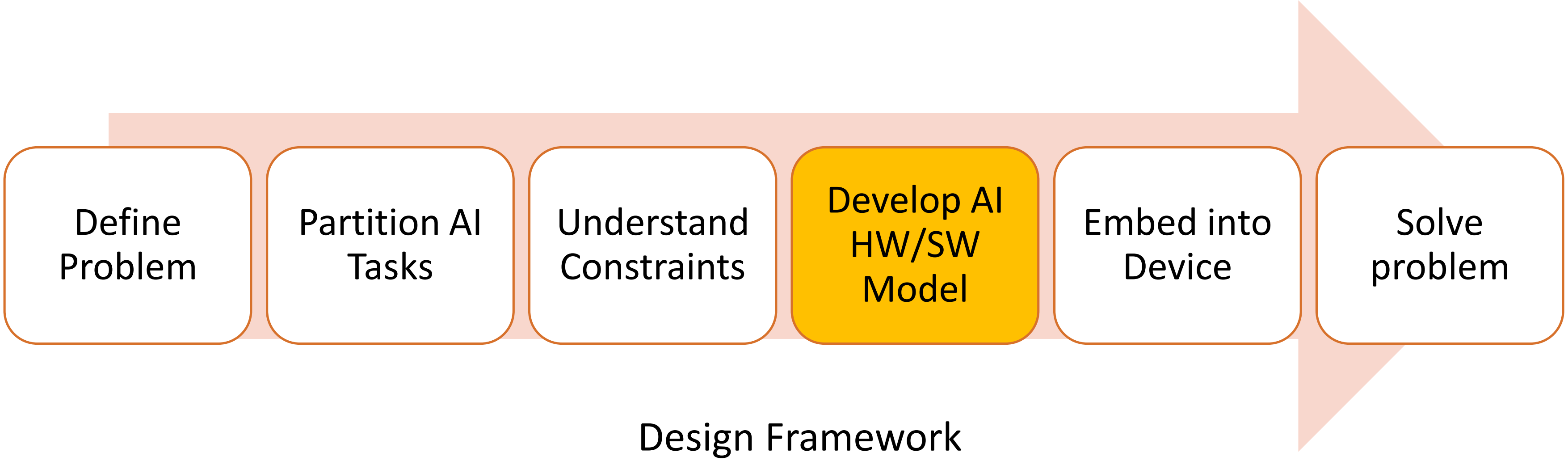
- The STDP unit Follows the *spike* or *pulse* model assumption for cortical neurons where information lies in spike timings, and not in spike shapes.
- 16 presynaptic traces are required to initiate the learning process. The PWU mechanism enables fast parallel on-chip learning.

Lecture Contents

1. Neuromorphic Computing Approaches
2. Hardware Models of Spiking Neurons
3. Synaptic Dynamics
4. Synaptic Plasticity Mechanisms and Learning
5. Synthesizing Real-Time Neuromorphic Systems
6. Conclusions

5. Synthesizing Real-Time Neuromorphic Systems:

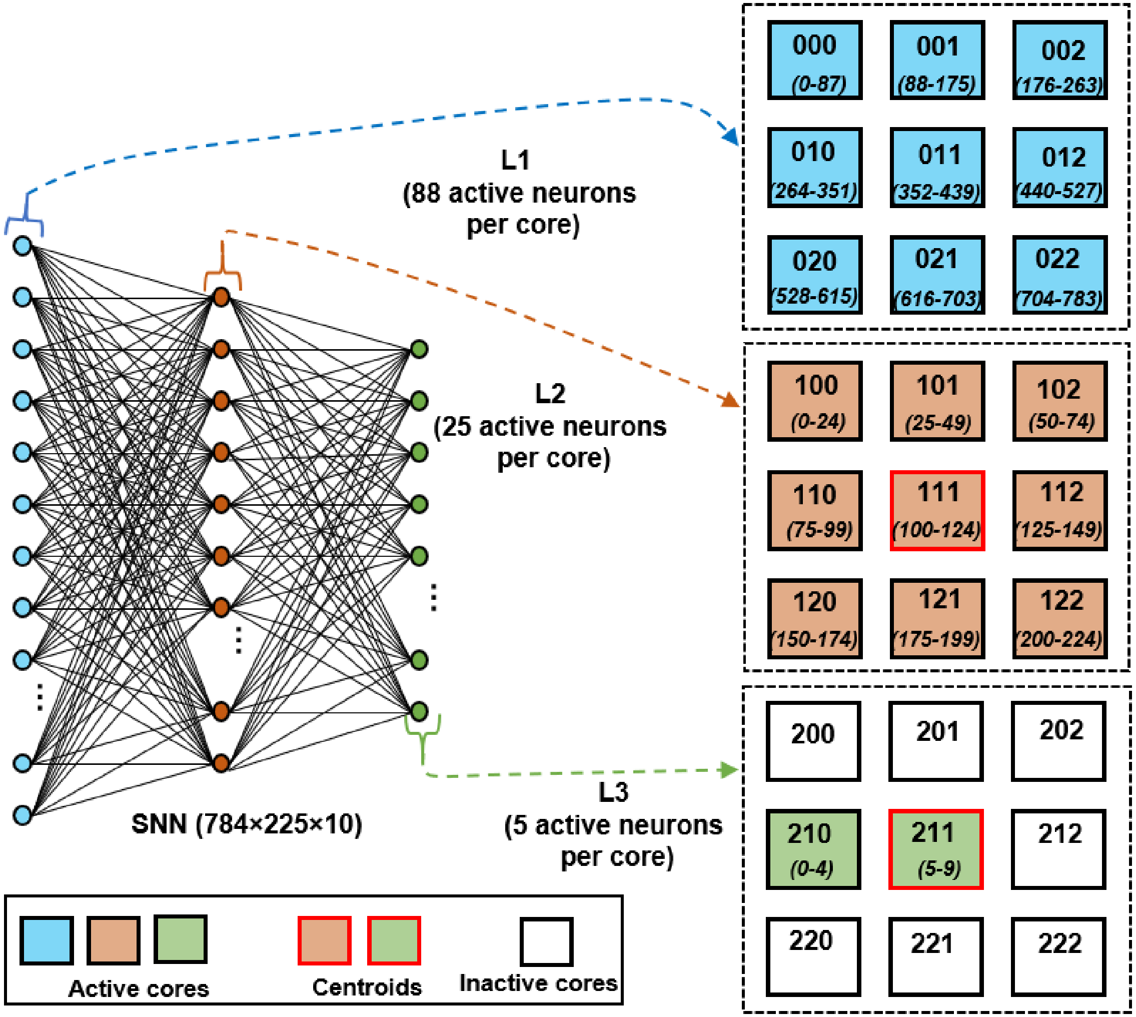
A framework for a Real Neurocomputing Design



Design Framework

5. Synthesizing Real-Time Neuromorphic Systems:

Application mapping



L1
(88 active neurons per core)

L2
(25 active neurons per core)

L3
(5 active neurons per core)

SNN (784×225×10)

000 (0-87)
001 (88-175)
002 (176-263)
010 (264-351)
011 (352-439)
012 (440-527)
020 (528-615)
021 (616-703)
022 (704-783)

100 (0-24)
101 (25-49)
102 (50-74)
110 (75-99)
111 (100-124)
112 (125-149)
120 (150-174)
121 (175-199)
122 (200-224)

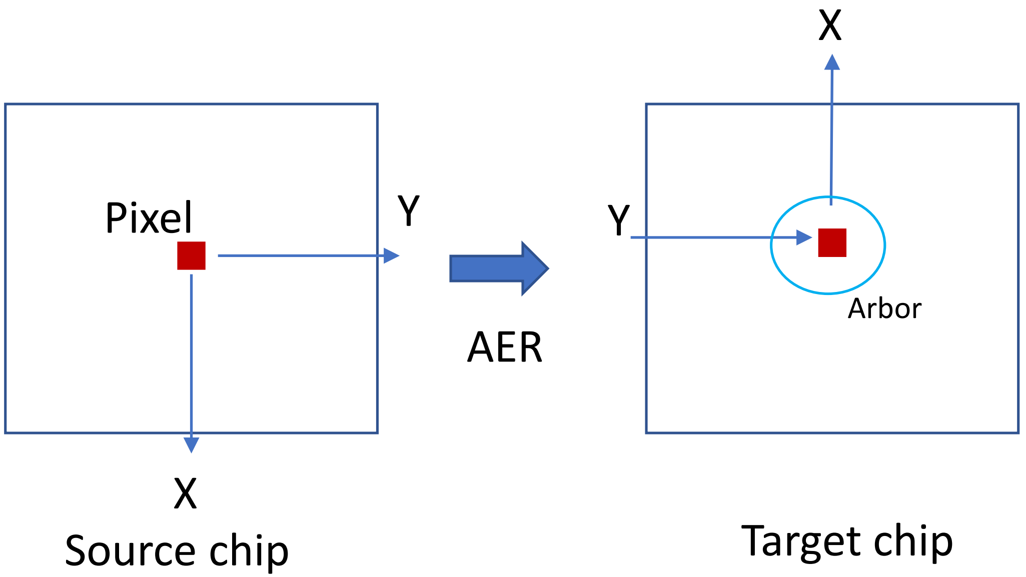
200
201
202
210 (0-4)
211 (5-9)
212
220
221
222

Active cores
Centroids
Inactive cores

Application mapping example on a 3 × 3 × 3 Neuromorphic Chip

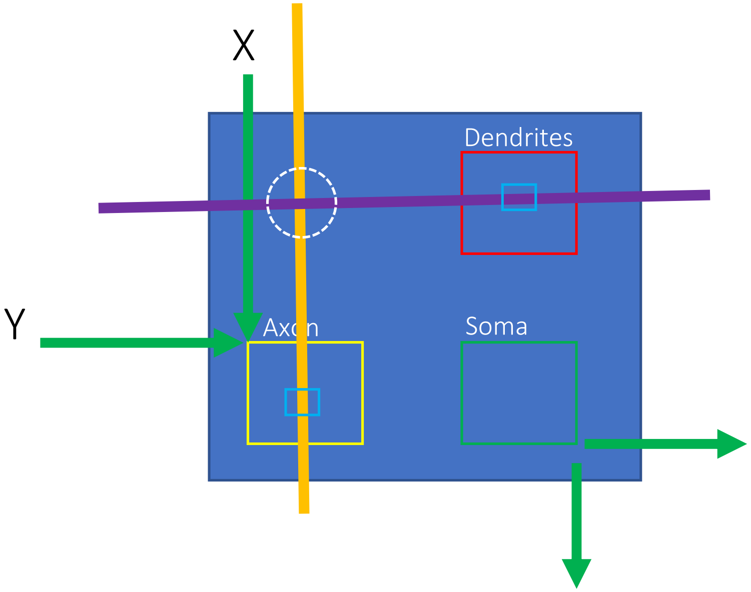
5. Synthesizing Real-Time Neuromorphic Systems:

Connecting Neuromorphic Chips



5. Synthesizing Real-Time Neuromorphic Systems:

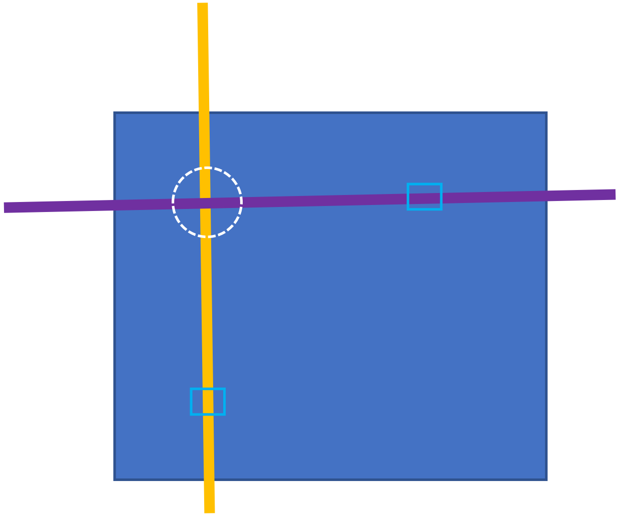
Inside the Pixel



Inside the Pixel

5. Synthesizing Real-Time Neuromorphic Systems:

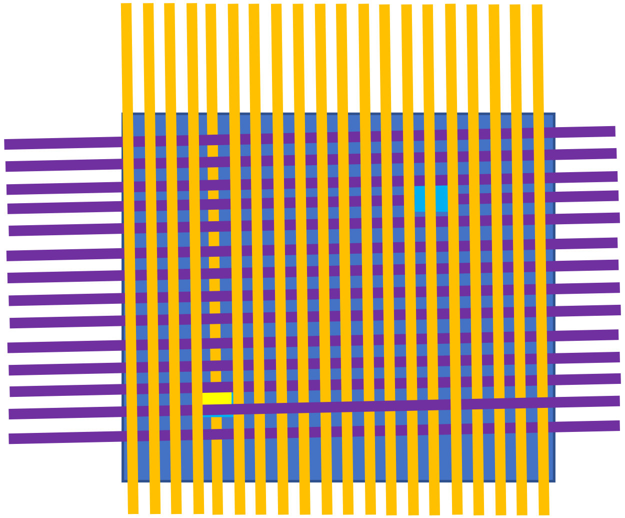
Using Crossbars



Using Crossbars

5. Synthesizing Real-Time Neuromorphic Systems:

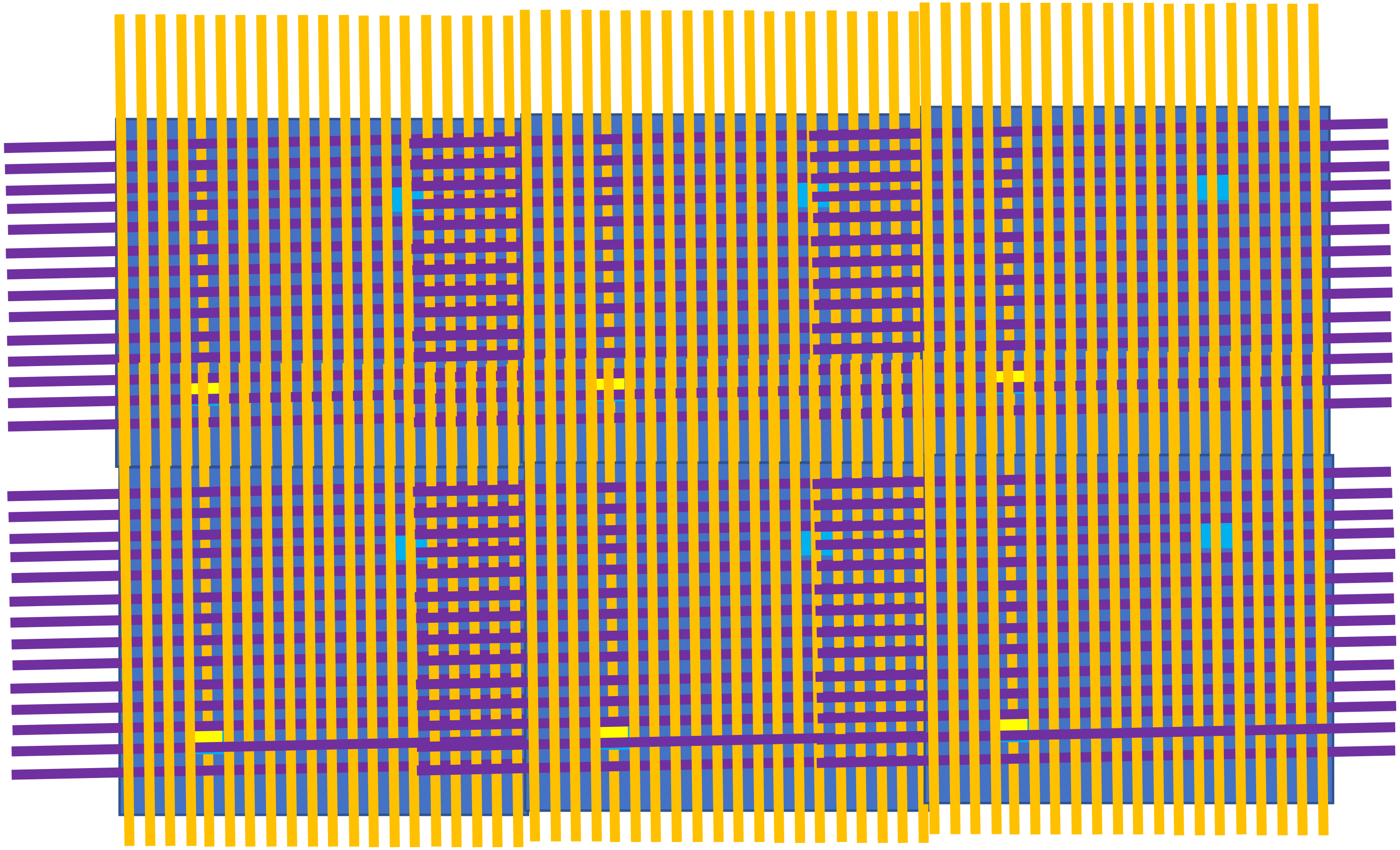
Using Crossbars



Using Crossbars

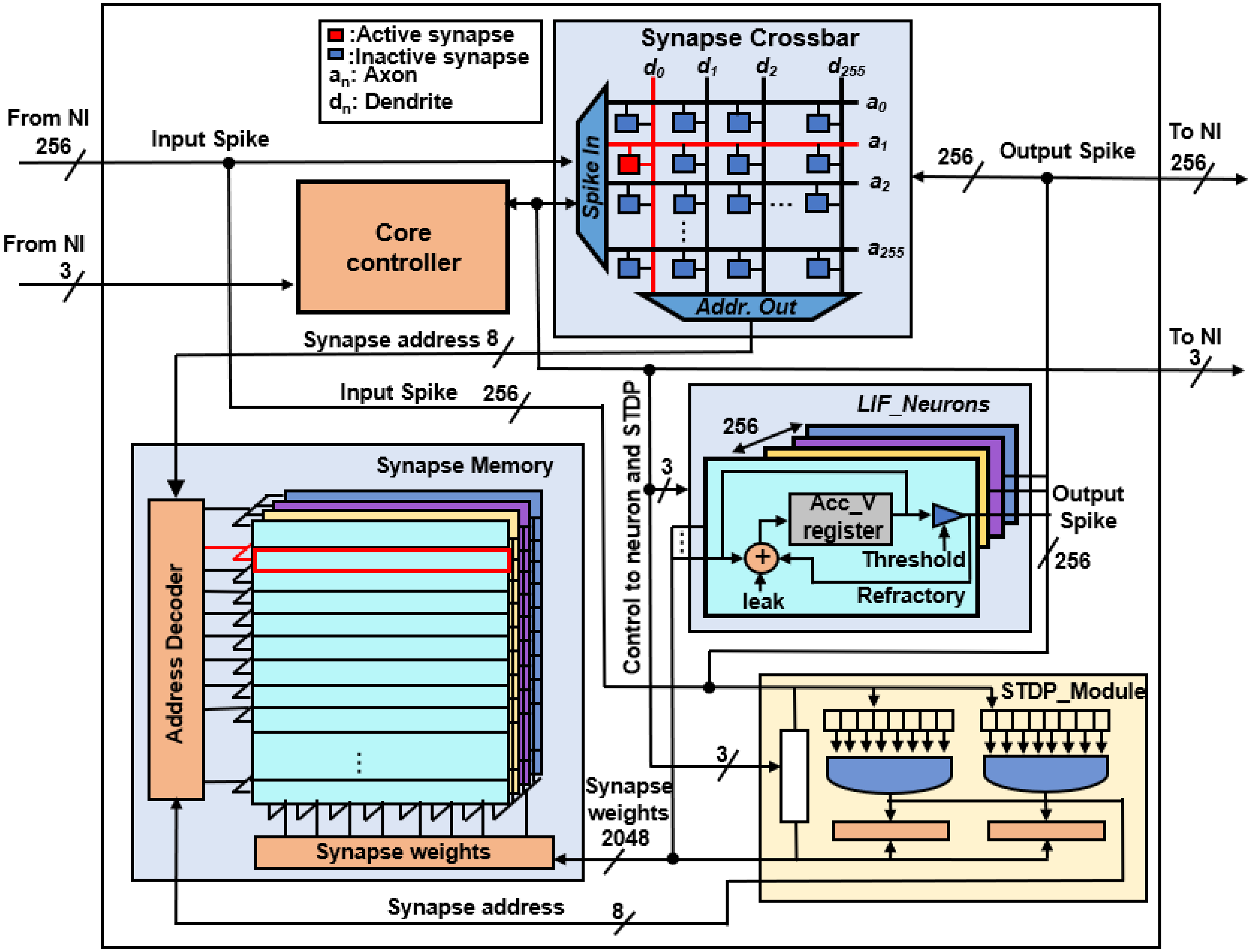
5. Synthesizing Real-Time Neuromorphic Systems:

Using Crossbars



5. Synthesizing Real-Time Neuromorphic Systems:

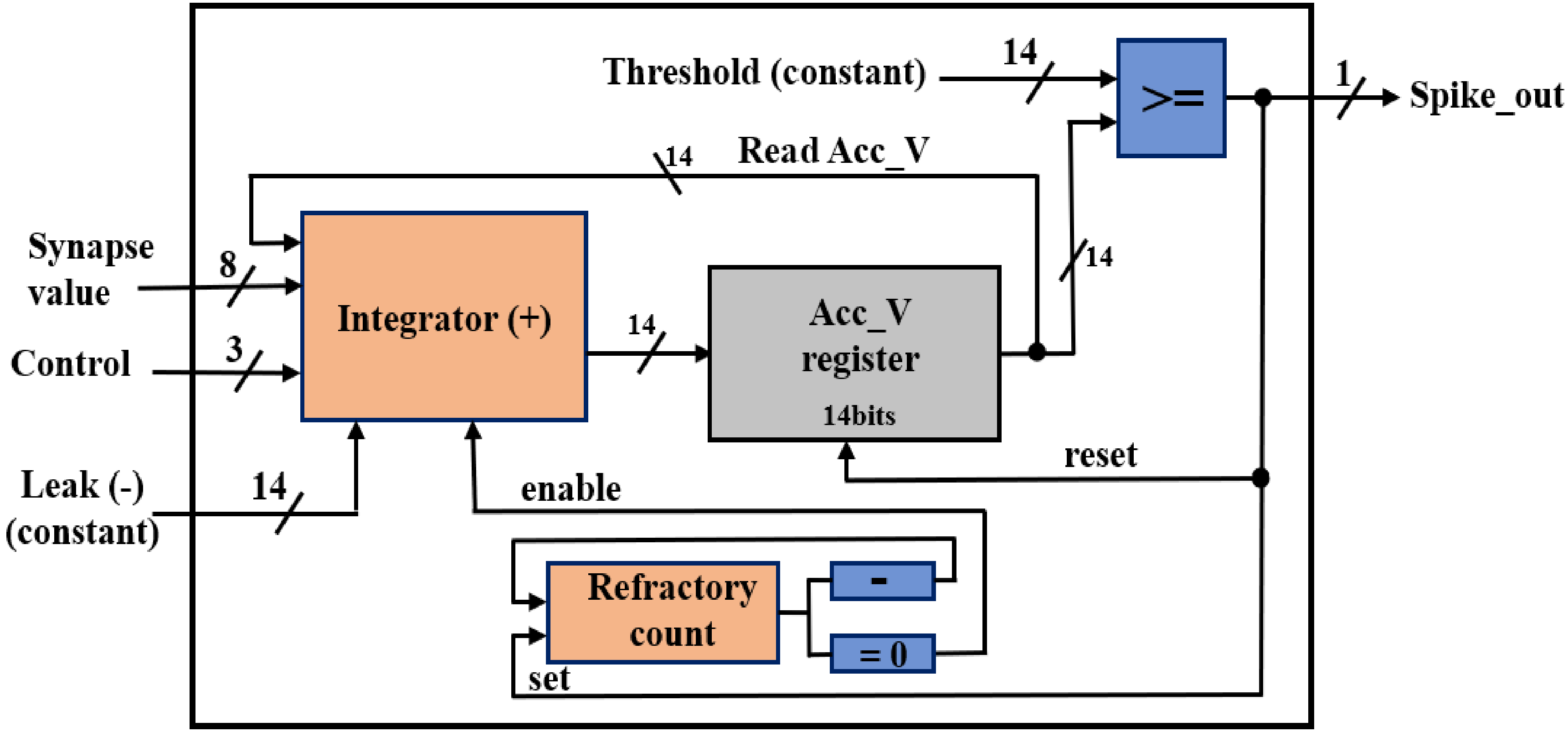
Spiking Neuro-Processing Core



Architecture of Spiking Neuro-Processing Core.

5. Synthesizing Real-Time Neuromorphic Systems:

LIF Neuron Module



Architecture of LIF Neuron

5. Synthesizing Real-Time Neuromorphic Systems:

LIF Neuron Module

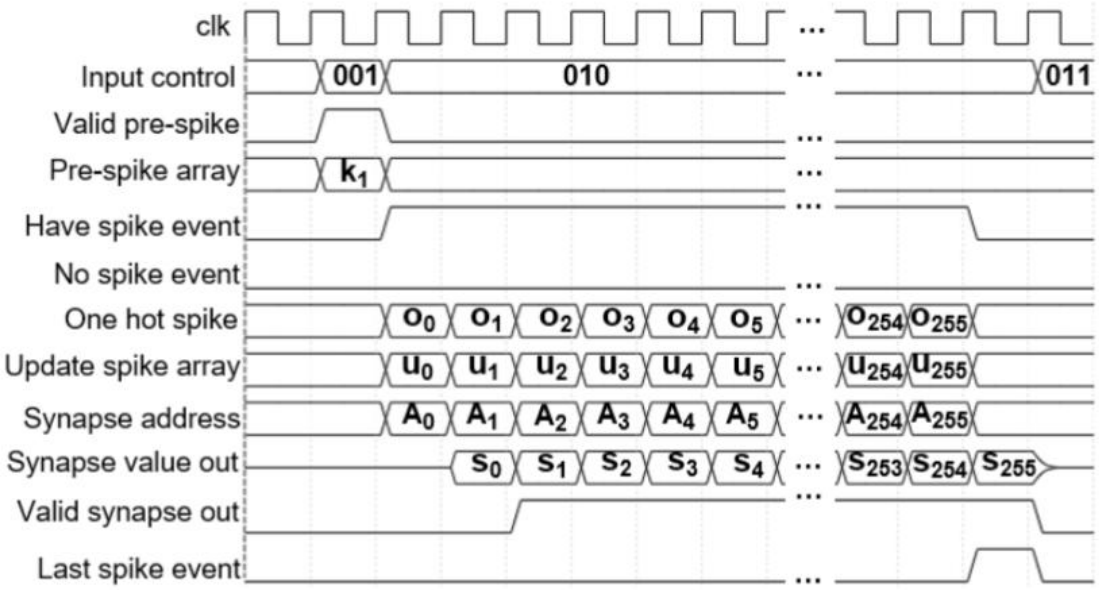

clk
Input control
Valid pre-spike
Pre-spike array
Have spike event
No spike event
One hot spike
Update spike array
Synapse address
Synapse value out
Valid synapse out
Last spike event
001
010
011
k_1
o_0 o_1 o_2 o_3 o_4 o_5 … o_{254} o_{255}
u_0 u_1 u_2 u_3 u_4 u_5 … u_{254} u_{255}
A_0 A_1 A_2 A_3 A_4 A_5 … A_{254} A_{255}
S_0 S_1 S_2 S_3 S_4 … S_{253} S_{254} S_{255}

Illustration of neuron update operation at the crossbar

1. An input presynaptic spike array is stored and checked for spike events. If present, the ***Have spike*** event signal becomes high. Afterwards, the one hot operation to get the synapse address begins, updating the one hot spike array for every spike event: from ***O0 to O255***.

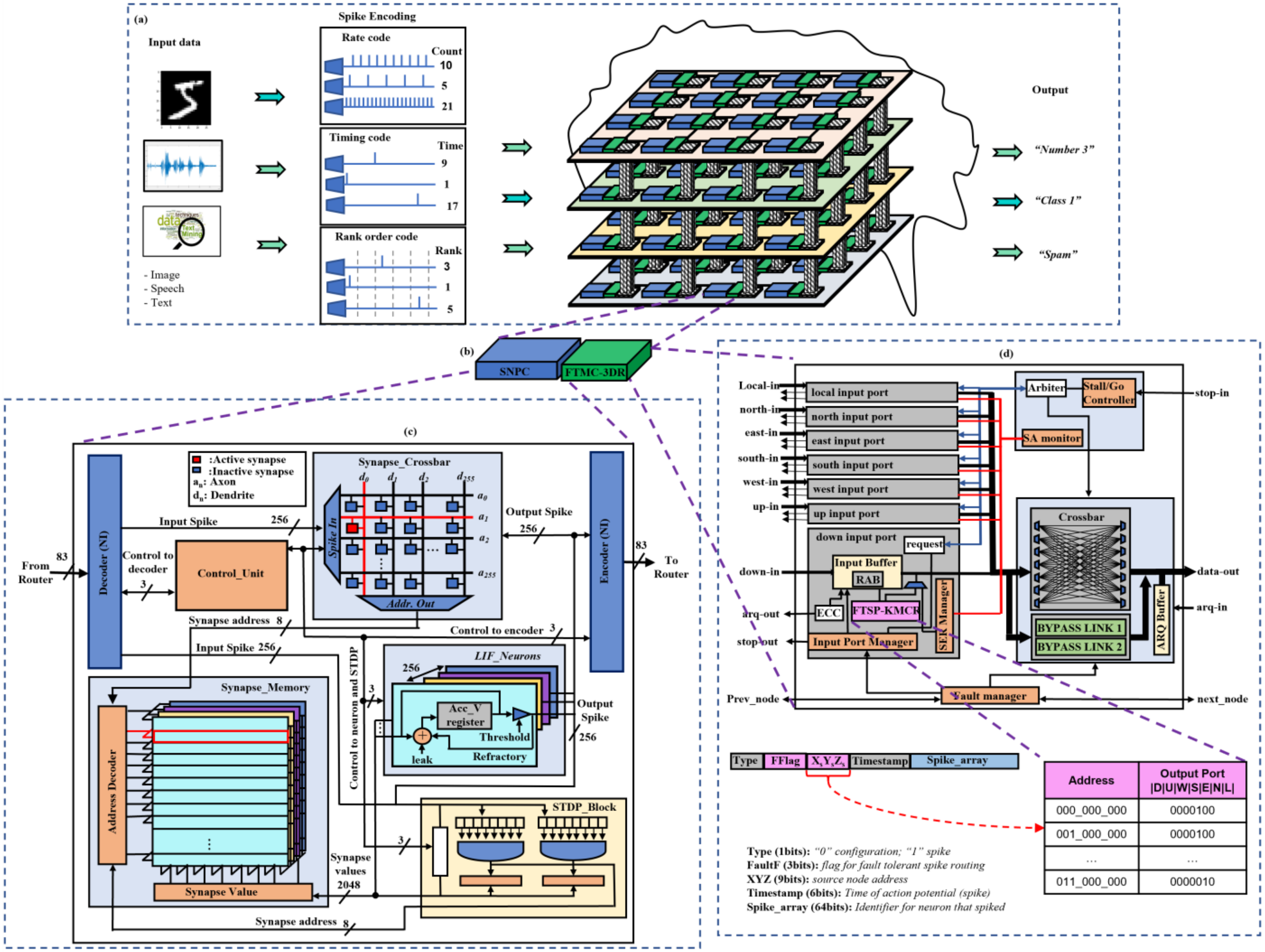
2. The stored presynaptic spike array is also updated after each spike event is processed: from ***U0 to U255.***

3. The synapse address is then used to fetch the synapse values from the synapse memory, and sent to the postsynaptic neurons.

4. When the last spike event in the array has been processed, the crossbar sends a signal to the control unit signaling that all spike events have been processed

5. Synthesizing Real-Time Neuromorphic Systems:

NASH Neuromorphic Architecture



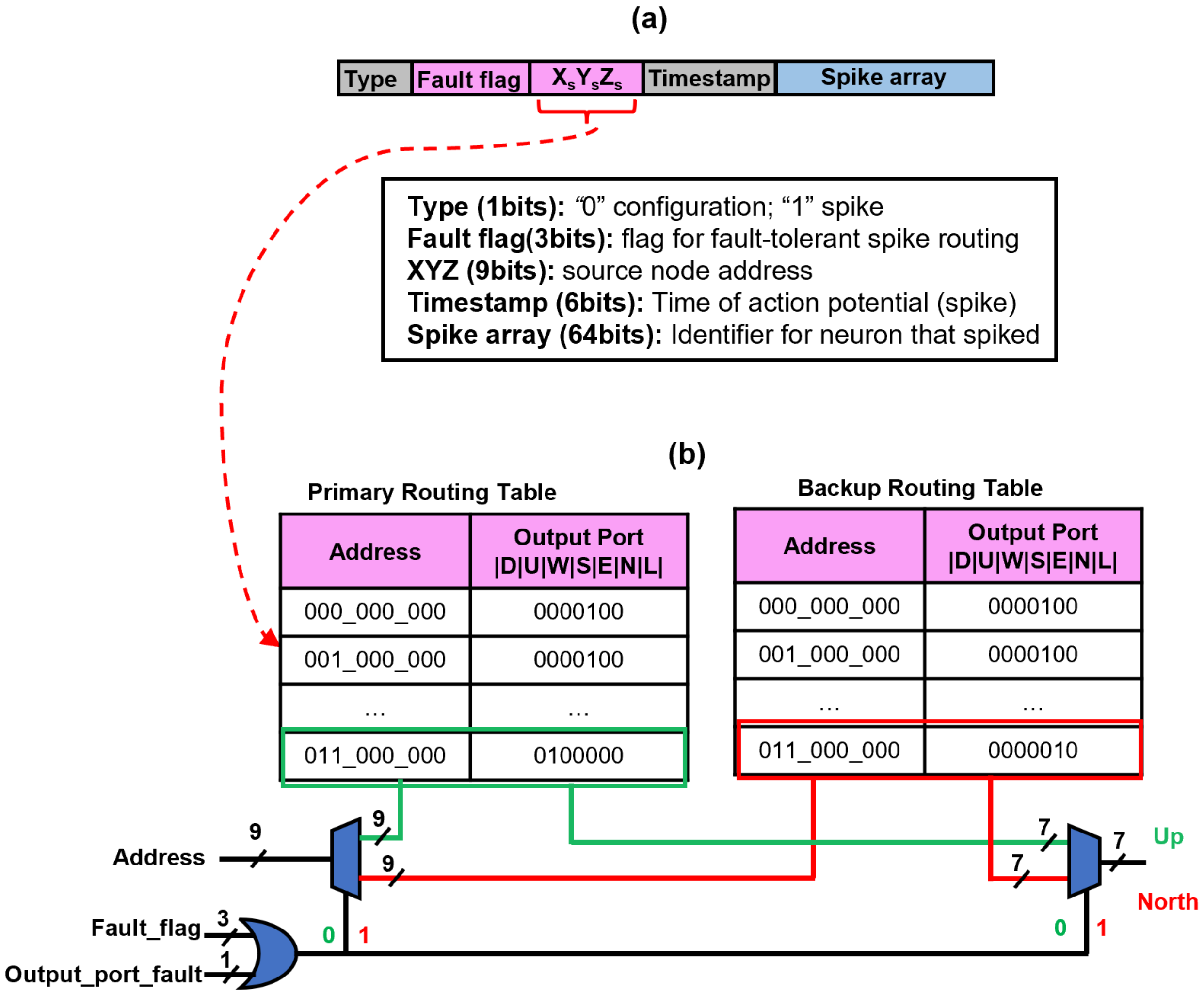
Address	Output Port |D|U|W|S|E|N|L|
000_000_000	0000100
001_000_000	0000100
…	…
011_000_000	0000010

Type (1bits): *"0" configuration; "1" spike*
FaultF (3bits): *flag for fault tolerant spike routing*
XYZ (9bits): *source node address*
Timestamp (6bits): *Time of action potential (spike)*
Spike_array (64bits): *Identifier for neuron that spiked*

Organization of the NASH Neuromorphic Chip

5. Synthesizing Real-Time Neuromorphic Systems:

Spiking Neuron Packet Format



(a)

Type	Fault flag	$X_sY_sZ_s$	Timestamp	Spike array

Type (1bits): "0" configuration; "1" spike
Fault flag(3bits): flag for fault-tolerant spike routing
XYZ (9bits): source node address
Timestamp (6bits): Time of action potential (spike)
Spike array (64bits): Identifier for neuron that spiked

(b)

Primary Routing Table

Address	Output Port |D|U|W|S|E|N|L|
000_000_000	0000100
001_000_000	0000100
…	…
011_000_000	0100000

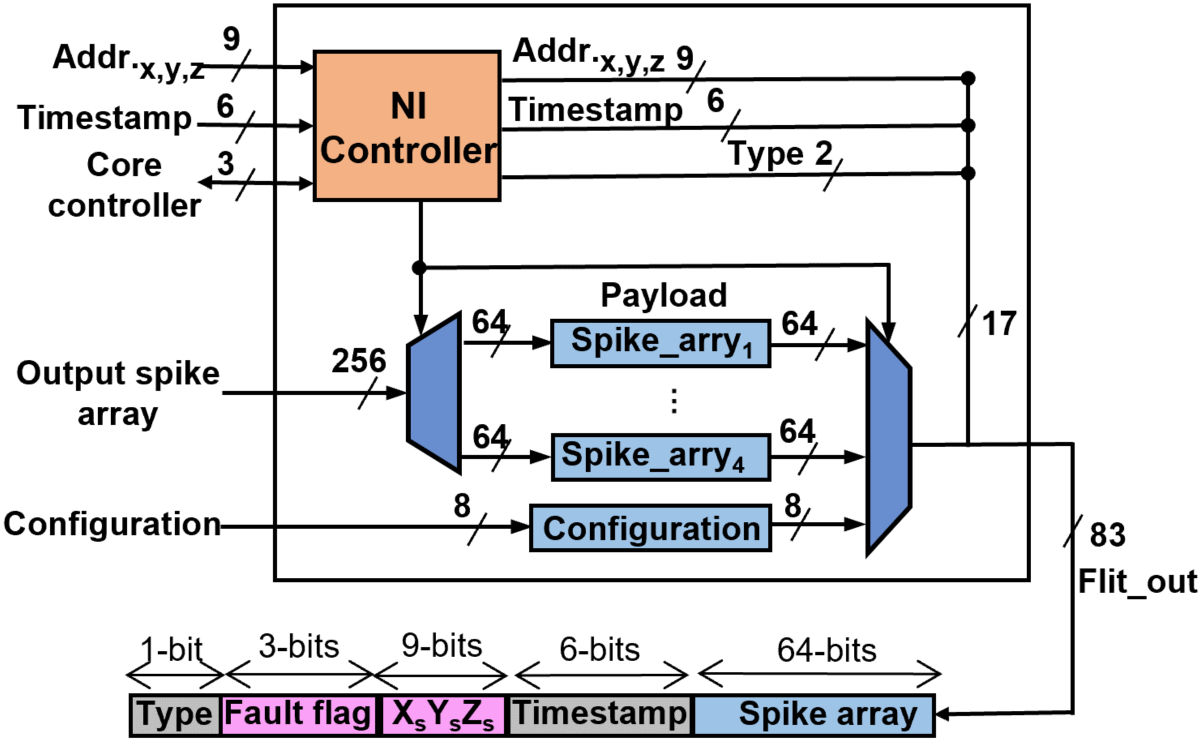
Backup Routing Table

Address	Output Port |D|U|W|S|E|N|L|
000_000_000	0000100
001_000_000	0000100
…	…
011_000_000	0000010

Spiking neuron packet format

5. Synthesizing Real-Time Neuromorphic Systems:

Network Interface (1/2): Encoder



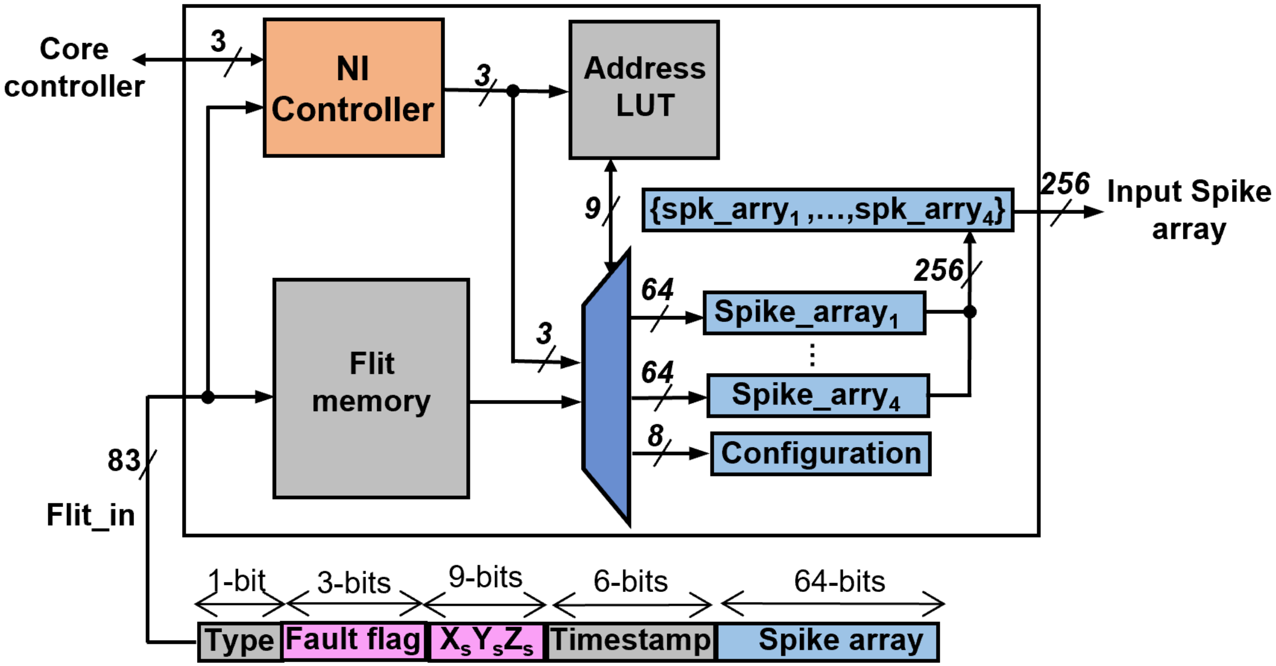
Network Interface: Encoder.

Operations of the encoder can be summarized in the following steps:

- Receive output spikes from local SNPC and packet into flits.
- After packeting, send flit to local router

5. Synthesizing Real-Time Neuromorphic Systems:

Network Interface (1/2): Decoder



Network Interface: Decoder.

Operations of the decoder can be summarized in the following steps:

- Receive spike packets from local router and unpack.
- Forward the spikes to the local SNPC as presynaptic spike train.

Lecture Contents

1. Neuromorphic Computing Approaches
2. Hardware Models of Spiking Neurons
3. Synaptic Dynamics
4. Synaptic Plasticity Mechanisms and Learning
5. Synthesizing Real-Time Neuromorphic Systems
6. Conclusions

6. Conclusions

- Neuromorphic Computing is the use of hardware (VLSI) to simulate the biological architecture of the human nervous system (brain, complex network of nerves, etc.),

- Spiking Neural Network (Neuromorphic):
  - More analogous to the brain, communicating via spikes in a sparse event-driven manner.
  - Exploits spike sparsity to achieve low power.

- Synaptic dynamics is the time-dependent changes in synaptic currents that change the coupling strength between neurons.

- There are various training/learning algorithms for SNNs:
  - Unsupervised Spike-timing-dependent plasticity (STDP)
  - ANN to SNN conversion

- Synthesizing a Neuromorphic System:
  - Define Problem→ Partition AI Tasks → Understand Constraints → Develop AI HW/SW Model → Embed into Device → Solve the Problem

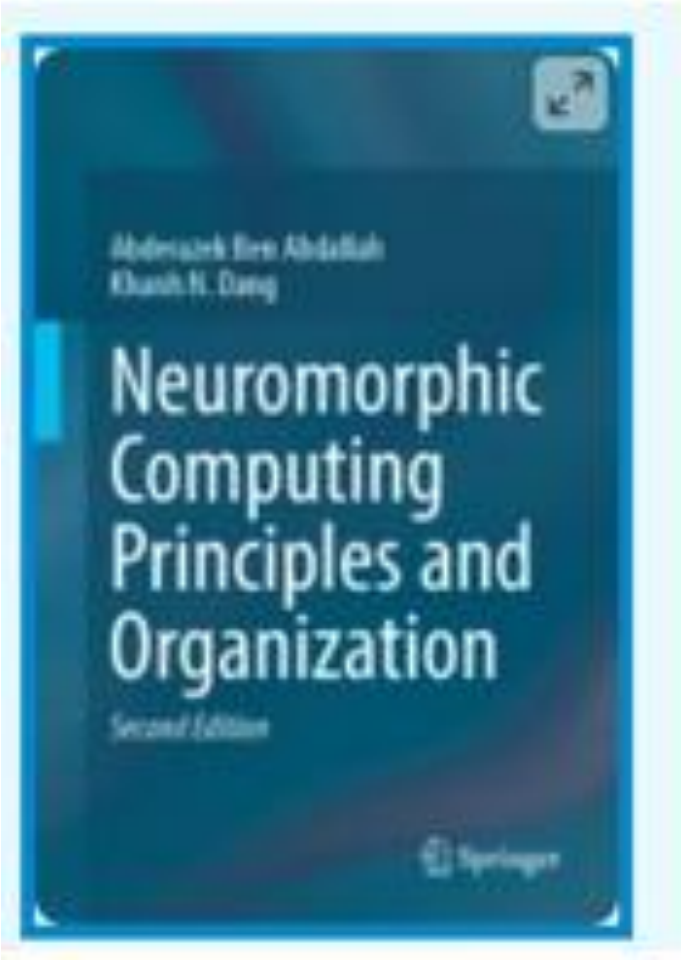
Quizzes

Please download Quiz1 material from here:

https://www.u-aizu.ac.jp/~benab/classes/kit/quiz1.pdf

Reference

Neuromorphic Computing Course, UoA
https://www.u-aizu.ac.jp/misc/neuro-eng/book/NeuromorphicComputing/



Abderazek Ben Abdallah
Khanh N. Dang

Neuromorphic Computing Principles and Organization

Second Edition

Springer

https://link.springer.com/book/9783031830884

Reference

1. Ngo-Doanh Nguyen, Khanh N. Dang, Akram Ben Ahmed, Abderazek Ben Abdallah, Xuan-Tu Tran, "**NOMA: A Novel Reliability Improvement Methodology for 3-D IC-based Neuromorphic Systems**", IEEE Transactions on Components, Packaging and Manufacturing Technology, 2024. DOI: 10.1109/TCPMT.2024.3488113
2. M. Maatar, Z. Wang, K. N. Dang and A. Ben Abdallah, "**BTSAM: Balanced Thermal-State-Aware Mapping Algorithms and Architecture for 3D-NoC-Based Neuromorphic Systems**," in IEEE Access, vol. 12, pp. 126679-126692, 2024
3. K. N. Dang, N. A. V. Doan, N. -D. Nguyen and A. B. Abdallah, "**HeterGenMap: An Evolutionary Mapping Framework for Heterogeneous NoC-Based Neuromorphic Systems**," in IEEE Access, vol. 11, pp. 144095-144112, 2023
4. N. -D. Nguyen, A. B. Ahmed, A. Ben Abdallah and K. N. Dang, "**Power-Aware Neuromorphic Architecture With Partial Voltage Scaling 3-D Stacking Synaptic Memory,**" in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 31, no. 12, pp. 2016-2029, Dec. 2023
5. W. Y. Yerima, K. N. Dang and A. B. Abdallah, "**R-MaS3N: Robust Mapping of Spiking Neural Networks to 3D-NoC-Based Neuromorphic Systems for Enhanced Reliability**," in IEEE Access, vol. 11, pp. 94664-94678, 2023
6. N. -D. Nguyen, X. -T. Tran, A. B. Abdallah and K. N. Dang, "**An In-Situ Dynamic Quantization With 3D Stacking Synaptic Memory for Power-Aware Neuromorphic Architecture**," in IEEE Access, vol. 11, pp. 82377-82389, 2023
7. W. Y. Yerima, O. M. Ikechukwu, K. N. Dang and A. Ben Abdallah, "**Fault-Tolerant Spiking Neural Network Mapping Algorithm and Architecture to 3D-NoC-Based Neuromorphic Systems**," in IEEE Access, vol. 11, pp. 52429-52443, 2023
8. O. M. Ikechukwu, K. N. Dang and A. B. Abdallah, "**On the Design of a Fault-Tolerant Scalable Three Dimensional NoC-Based Digital Neuromorphic System With On-Chip Learning**," in IEEE Access, vol. 9, pp. 64331-64345, 2021
9. Abderazek Ben Abdallah, Khanh N. Dang, "**Toward Robust Cognitive 3D Brain-inspired Cross-paradigm System**," Frontier in Neuroscience 15:690208, doi: 10.3389/fnins.2021.690208