Neuromorphic Computing

1. Introduction

Ben Abdallah Abderazek, Khanh N. Dang
E-mail: {benab, khanh}@u-aizu.ac.jp

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:
Neural Network Generations

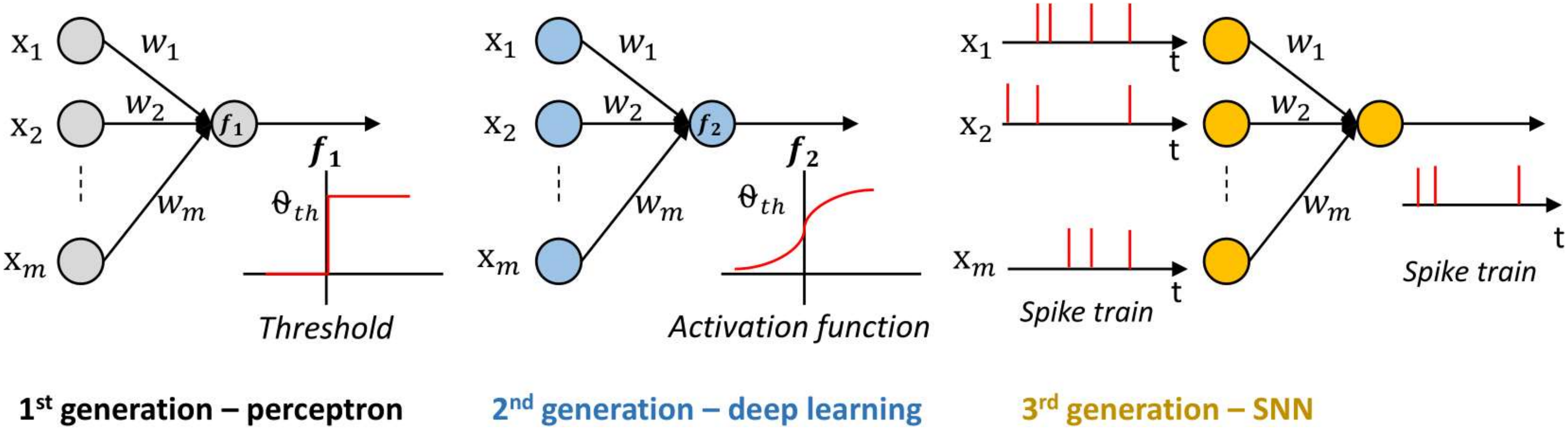


Fig. 1.2: Neural network generations

1. Neuromorphic Computing:

Conventional ANN vs Spiking Neural Network (Neuromorphic)

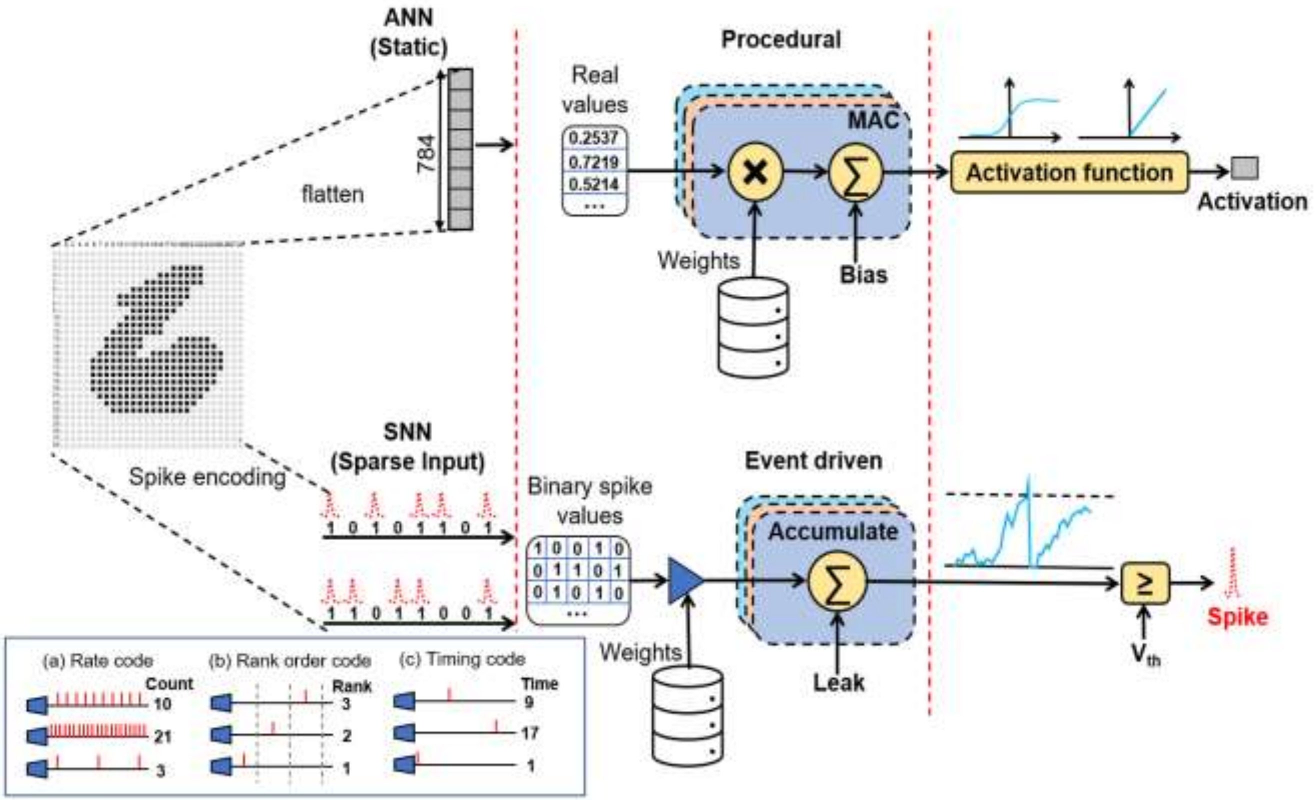
Artificial Neural Network (ANN) is a brain inspired computing paradigm modeled after the computational principles of the brain’s neural network.

Approaches:

- **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:

Conventional ANN vs Spiking Neural Network (Neuromorphic)

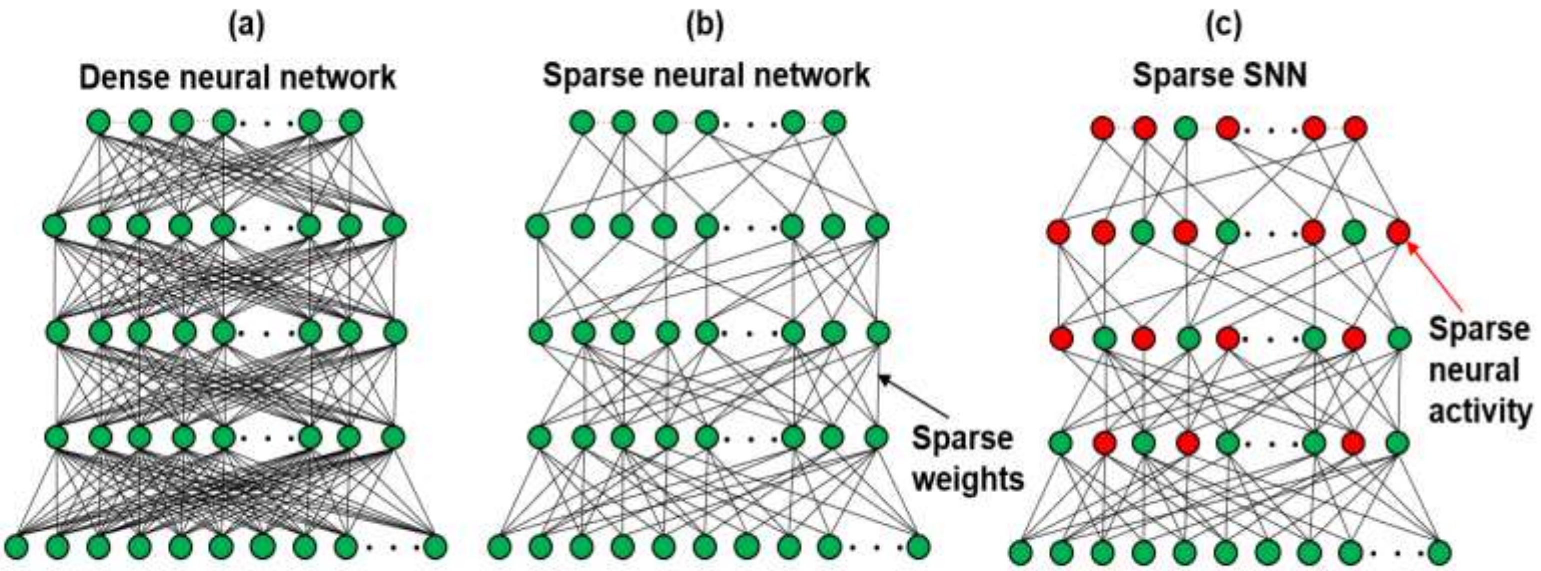


Conventional ANN vs Spiking Neural Network

- 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:

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:

What is Neuromorphic Computing?

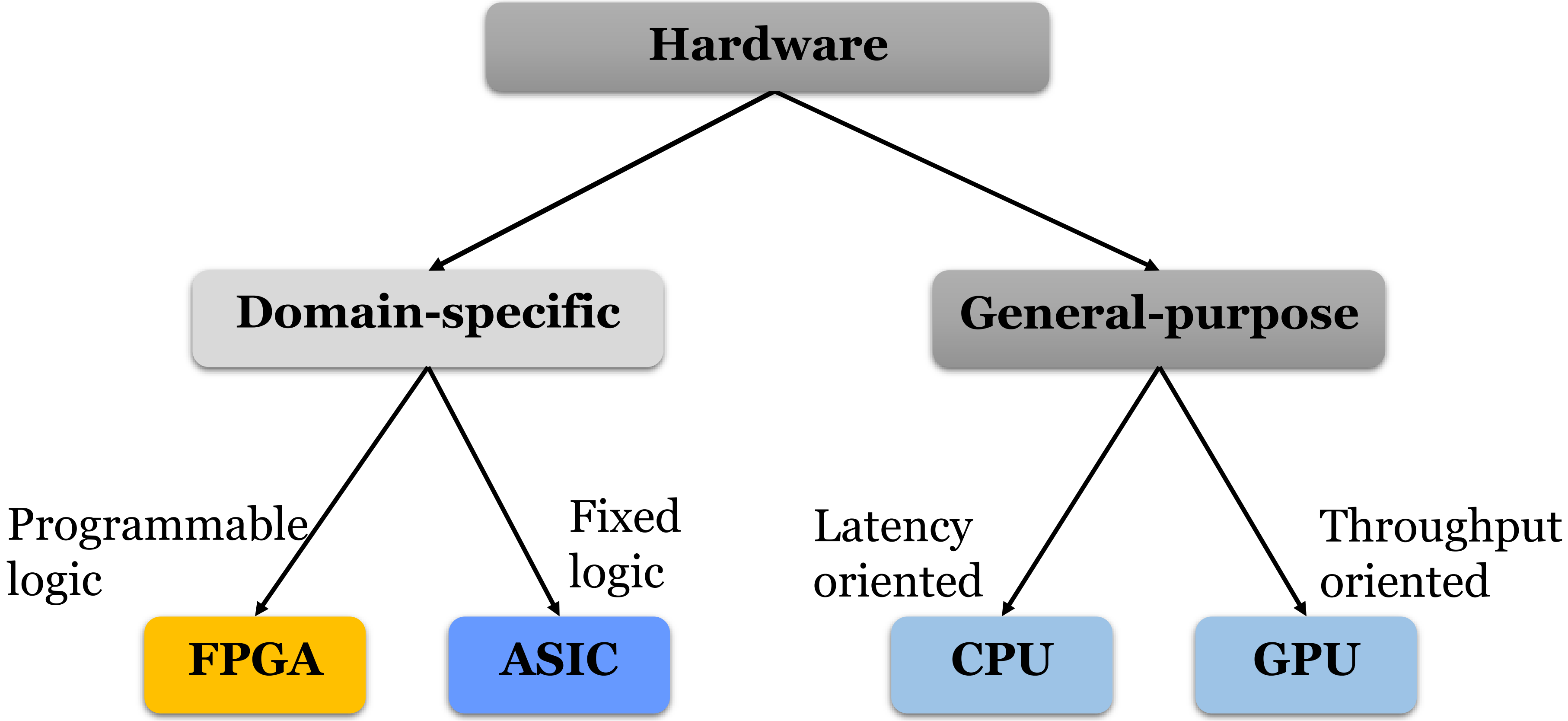
- **Neuromorphic Computing** is the use of **hardware (VLSI)** to simulate the biological architecture of the **human nervous system** (brain, complex network of nerves, etc.),

- Neuromorphic Engineering is a new emerging field that involves biology, physics, mathematics, and computer science and engineering to design hardware models **of neural and sensory systems.**

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

1. Neuromorphic Computing:

Neural Algorithms Computing in Hardware



Hardware

Domain-specific

General-purpose

Programmable logic

Fixed logic

Latency oriented

Throughput oriented

FPGA

ASIC

CPU

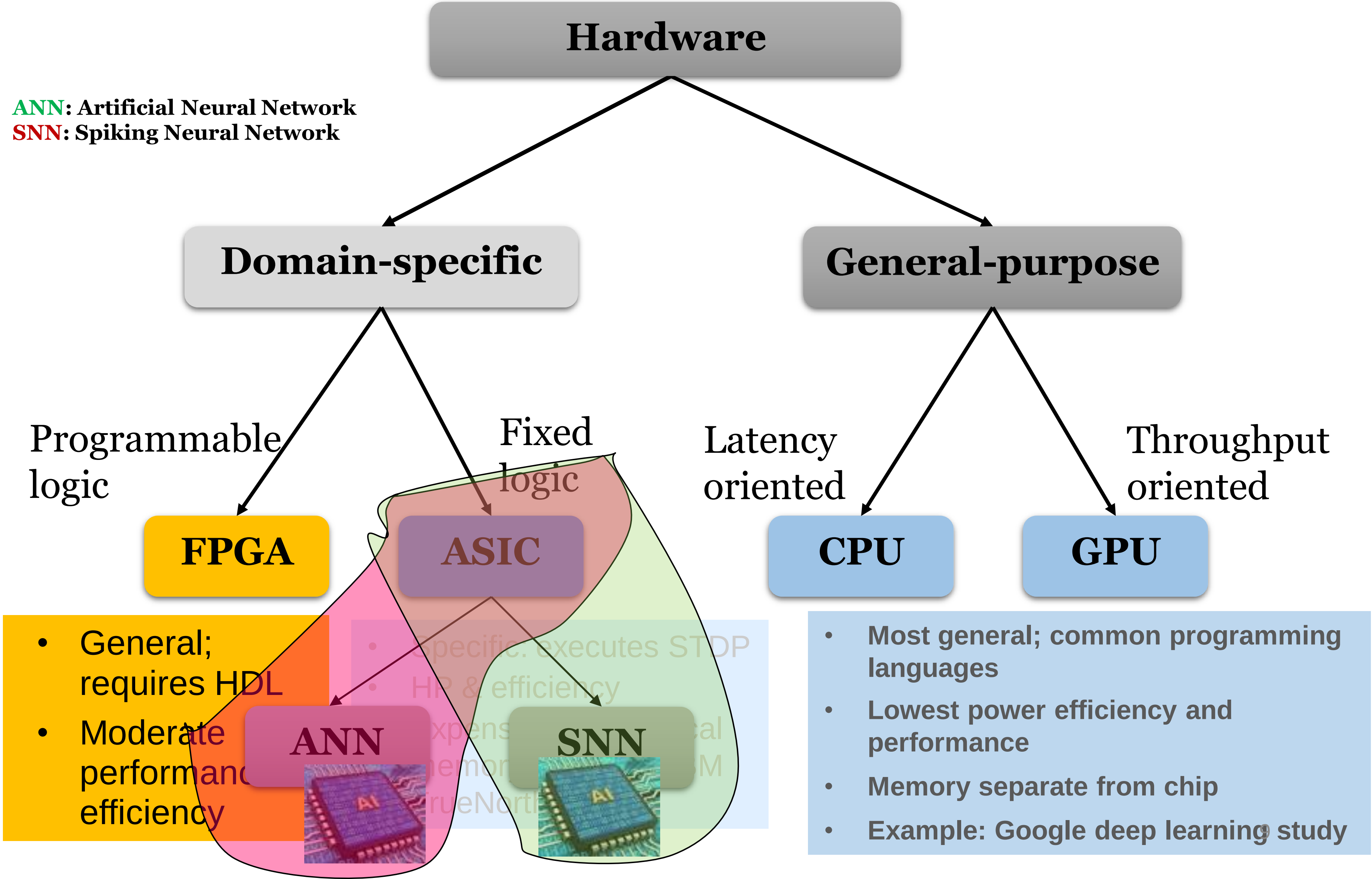
GPU

- 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

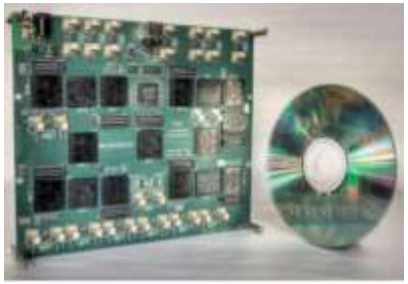
- General; requires HDL
- Moderate performance efficiency

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

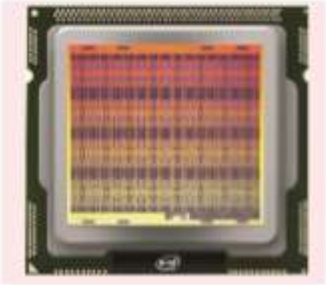
- Most general; common programming languages
- Lowest power efficiency and performance
- Memory separate from chip
- Example: Google deep learning study

1. Neuromorphic Computing:

Examples of Neuromorphic Chips/Systems


Neurogrid

IBM TrueNorth


Intel Loihi

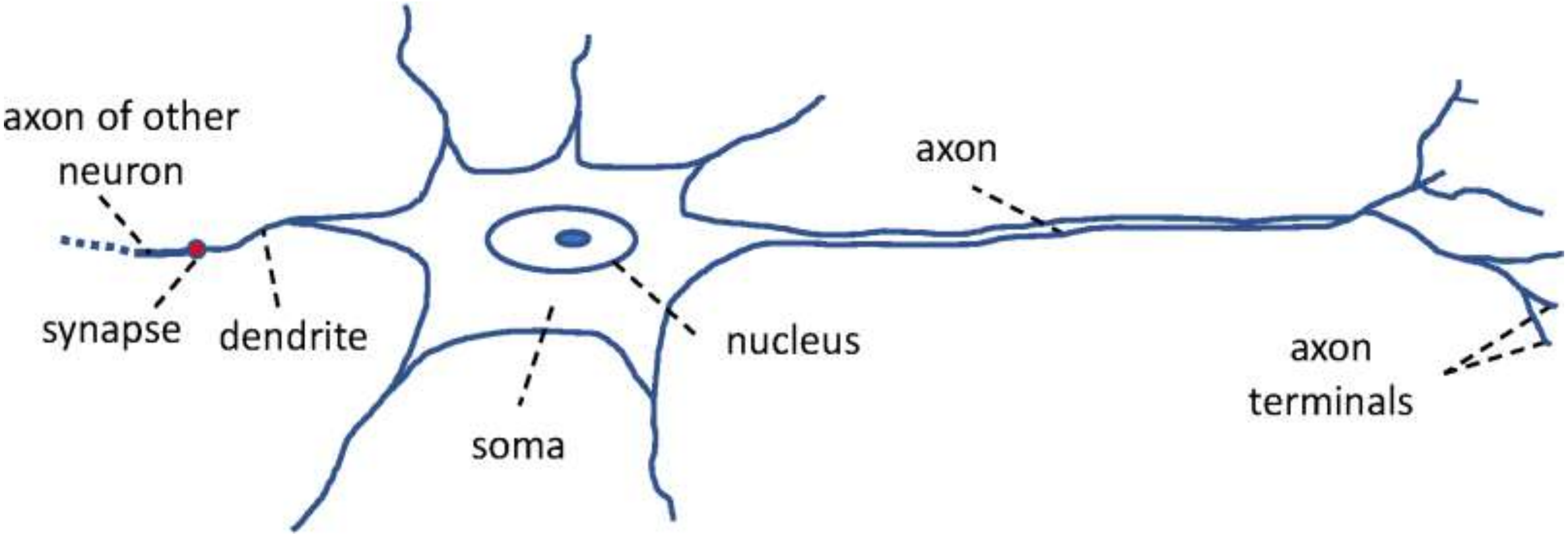
Examples of Neuromorphic Chips/Systems (not yet commercial)

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:

Neuron Excitability



(a)

Neurons information processing steps:

- **Synapses**: Connection between neurons
- **Dendrites**: Receive inputs
- **Cell body**: sums currents from dentures
- **Axon**: sends to action potential

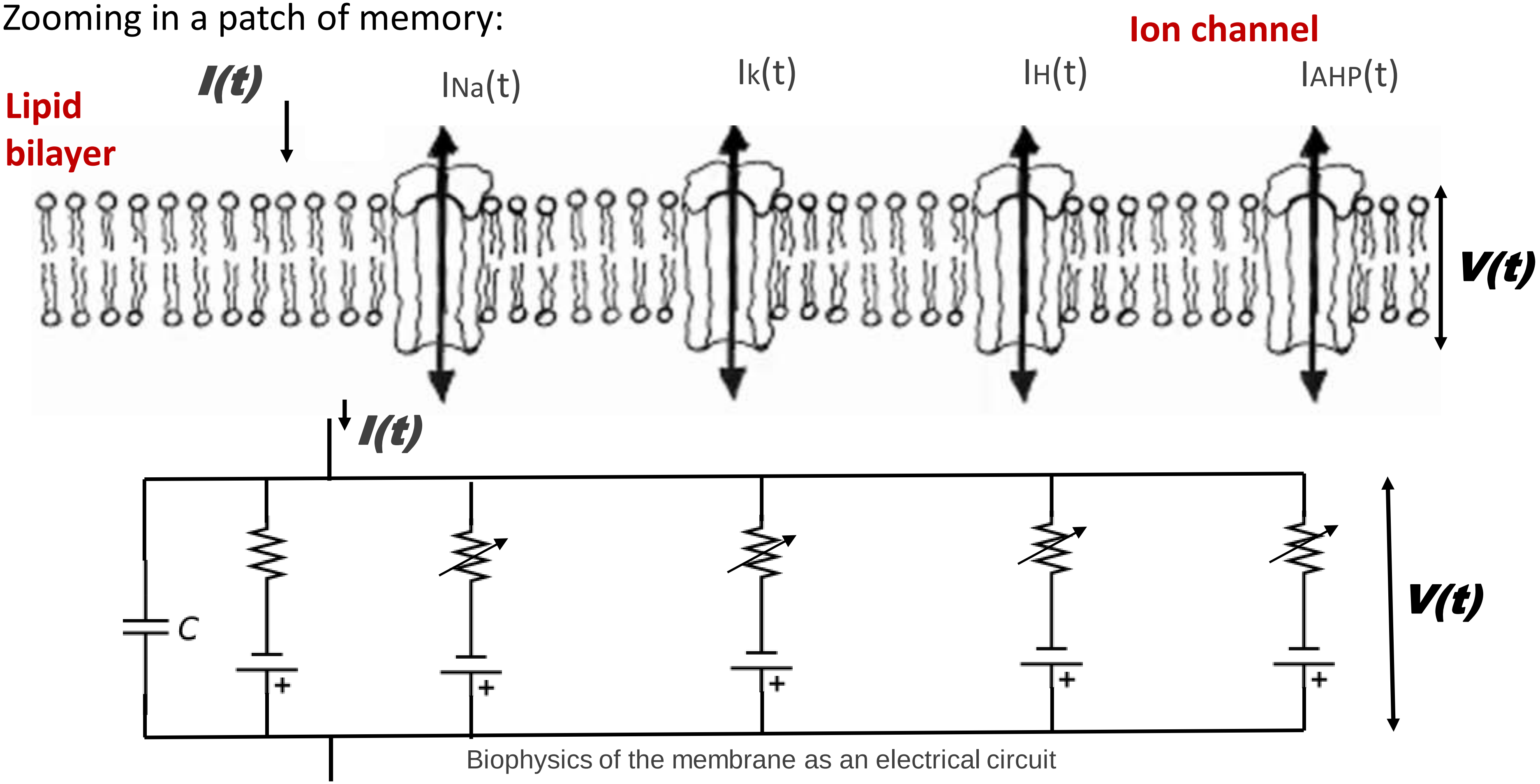
How are action potentials generated given the current flowing into the soma (cell body) from dendrites and synapses?

(b)

2. Hardware Models of Spiking Neurons:

Biophysical description

Zooming in a patch of memory:



Biophysics of the membrane as an electrical circuit

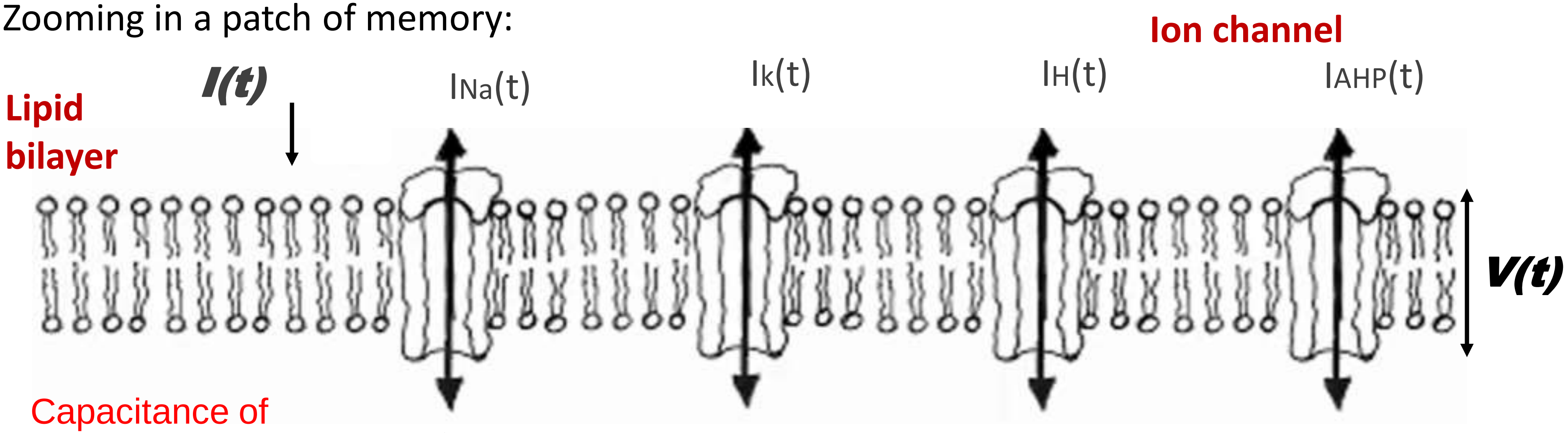
I(t): Current of membrane
V(t): Membrane potential (Difference in electrical potential between inside and outside of the cell.)

C: Capacitance of the membrane
gL: Conductance of the membrane
EL: Equilibrium potential of *Leak*

2. Hardware Models of Spiking Neurons:

Biophysical description

Zooming in a patch of memory:



Capacitance of the membrane

Na: sodium ion

K: potassium ion

$$\boxed{I = C dV/dt + g_L(V-E_L) + I_{Na} + I_K + I_H + I_{AHP}}$$

Leak conductance of the membrane

Kirchhoff's current rule

Leak equilibrium potential

I(t): Current of membrane

V(t): Membrane potential (Difference in electrical potential between inside and outside of the cell.)

C: Capacitance of the membrane

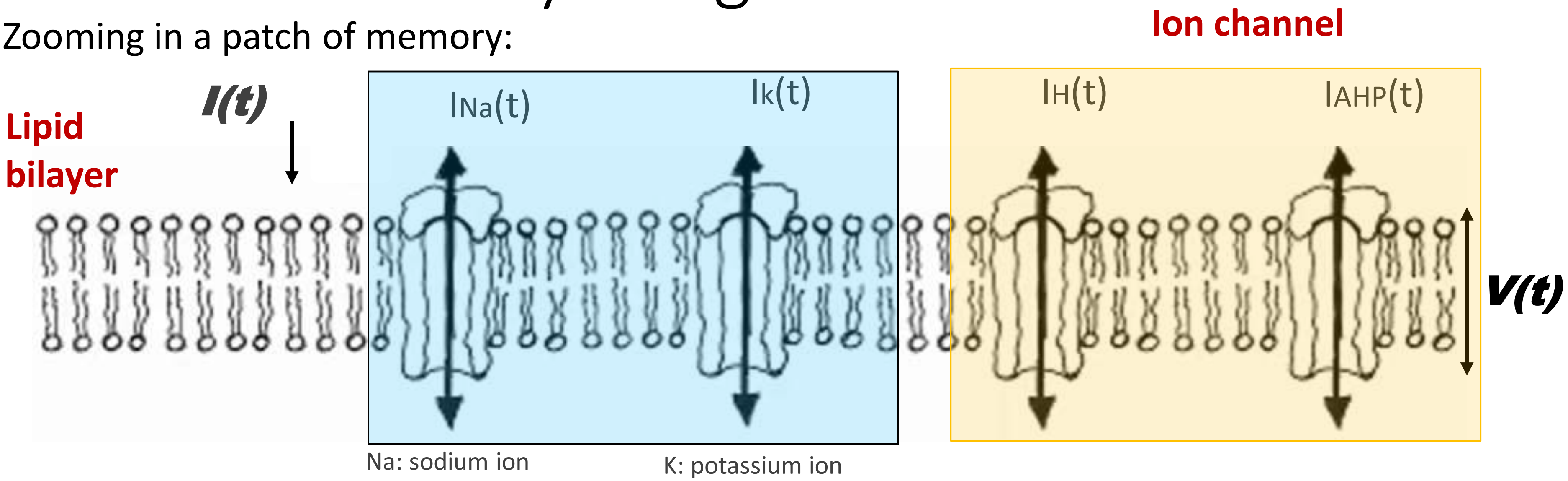
gL: Conductance of the membrane

EL: Equilibrium potential of *Leak*

2. Hardware Models of Spiking Neurons:

Leaky Integrate-and-Fire

Zooming in a patch of memory:



$$I = C dV/dt + g_L(V-E_L) + I_{Na} + I_K + I_H + I_{AHP}$$

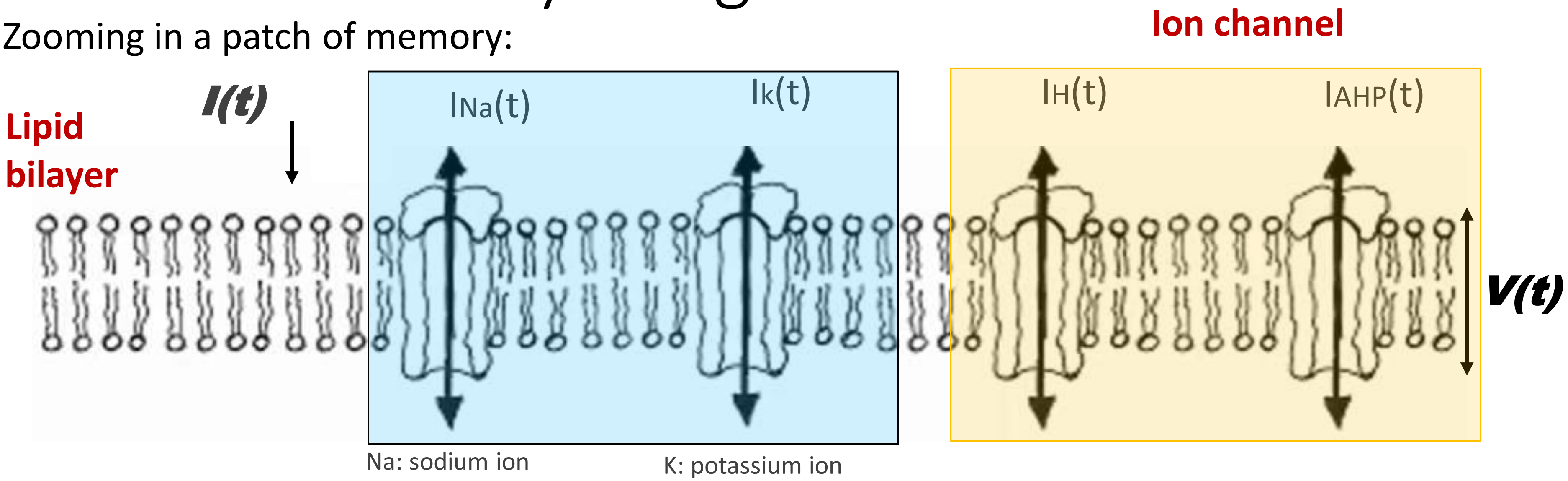
Replace by a threshold for spike emission follows by a reset to a fixed value/potential.

Ignore action on ion channels for now.

2. Hardware Models of Spiking Neurons:

Leaky Integrate-and-Fire

Zooming in a patch of memory:

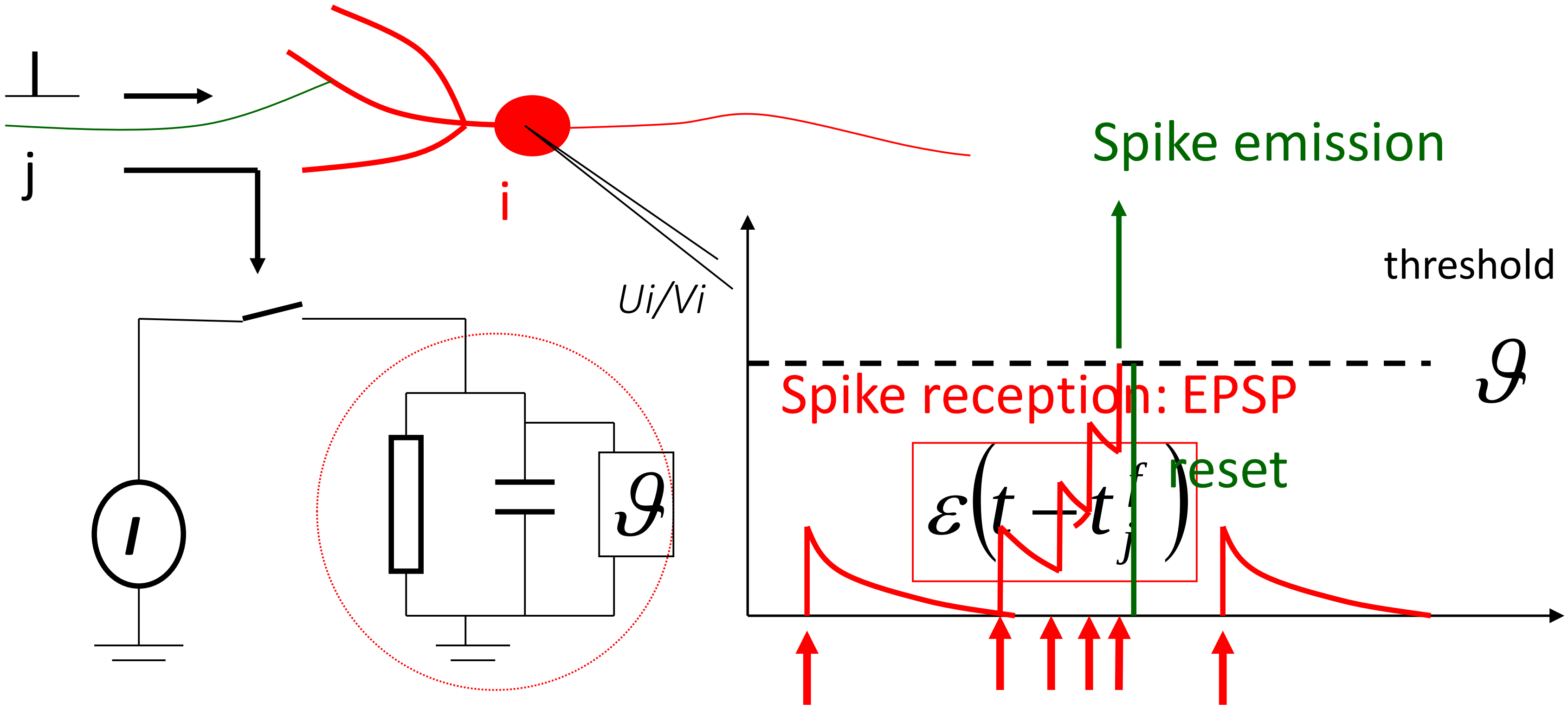


$$I = C dV/dt + g_L(V-E_L) + \boxed{I_{Na} + I_K +} \boxed{I_H + I_{AHP}}$$

$$C_m dV/dt = -g_L(V-E_L) + I$$

$$\boxed{\text{If } V(t) = V_{th} \text{ then } V(t+\Delta) = E_L}$$

Integrate-and-fire Model



Spike emission

threshold

Spike reception: EPSP

reset

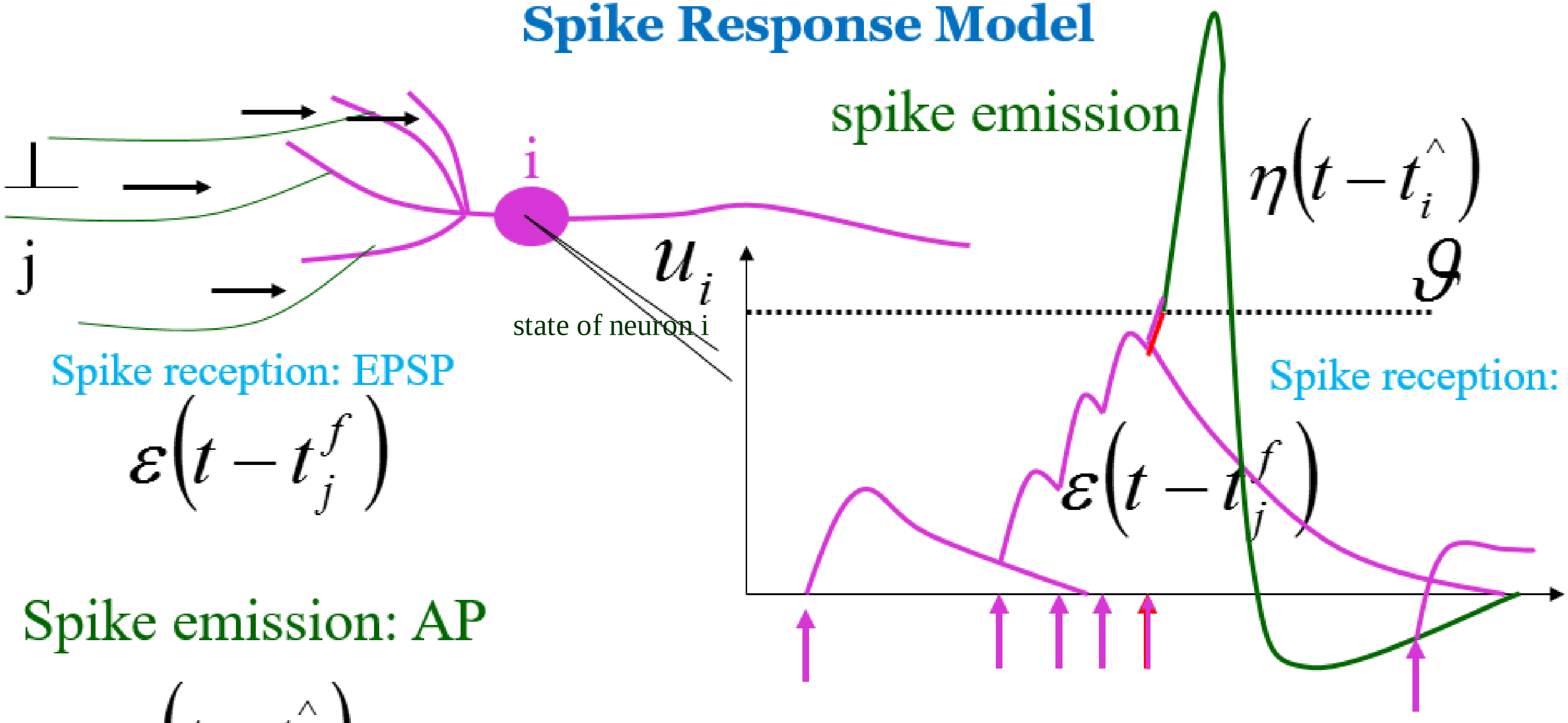
$$\tau \cdot \frac{d}{dt} \text{Vi} = -V(t) + RI(t)$$ linear

$$V(t) = \vartheta \Rightarrow$$ Fire+reset threshold

More details on ''Spiking Neuron Models'', Cambridge press, 2002

Spiking Neuron Model

Spike Response Model



Spike reception: EPSP

$$\varepsilon(t - t_j^f)$$

Spike emission: AP

$$\eta(t - t_i^{\wedge})$$

reset of the membrane potential (action potential)

EPSP: Excitatory postsynaptic potentials

$$u_i(t) = \eta(t - t_i^{\wedge}) + \sum_j \sum_f w_{ij}\, \varepsilon(t - t_j^f)$$

$$\boxed{u_i(t) = \vartheta \Rightarrow \text{Firing:} \quad t_i^{\wedge} = t}$$

Spike Resonse Model (SRM), Gerstner (1996)

2. Hardware Models of Spiking Neurons:

Spike Coding Schemes

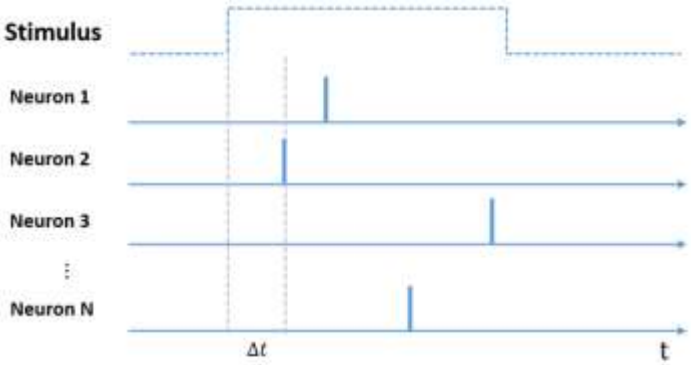


Fig. 2.2: Time to first spike

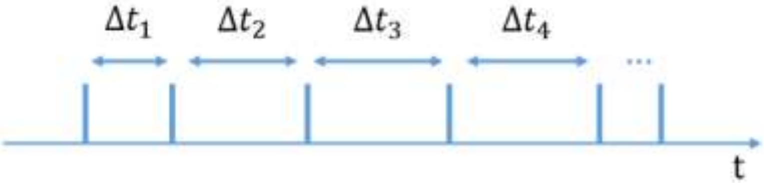


Fig. 2.3: Inter-spike-interval

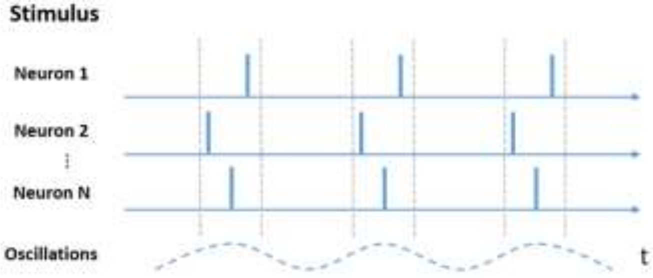


Fig. 2.4: Phase coding

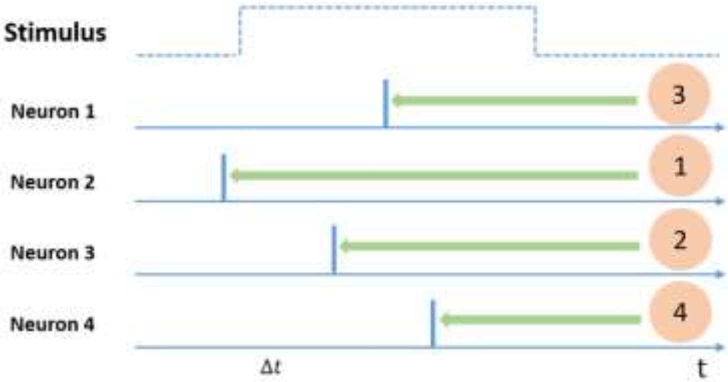


Fig. 2.5: Rank order

2. Hardware Models of Spiking Neurons:

Neurons Communication Scheme

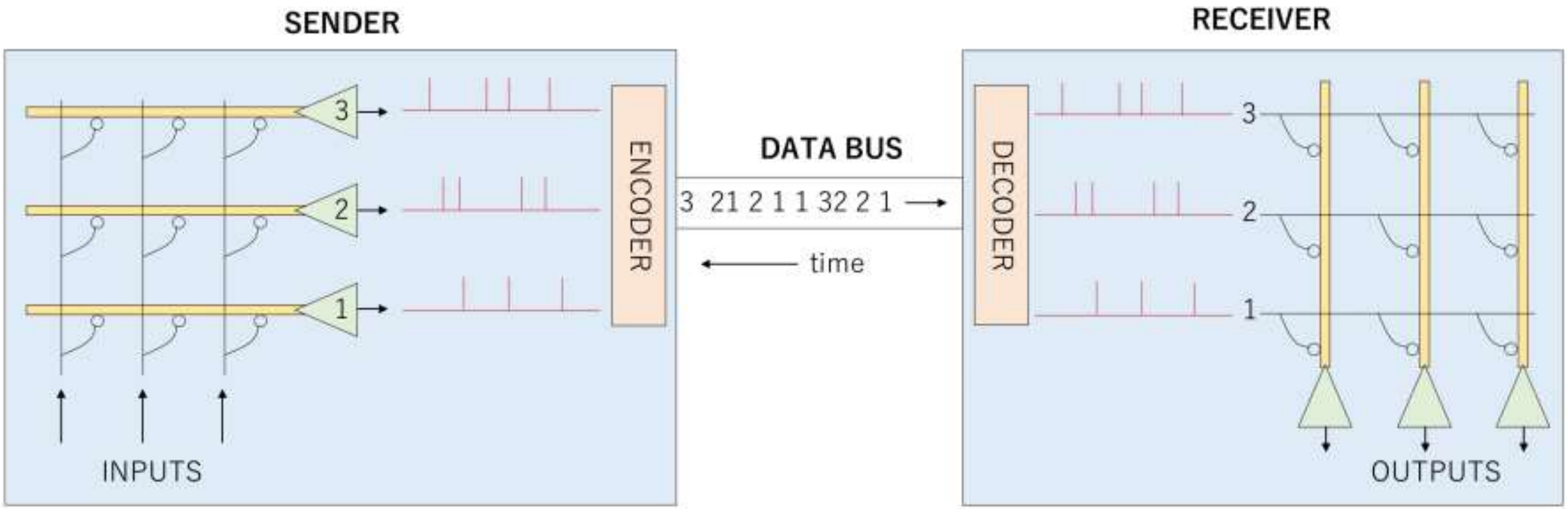


Fig. 2.15: AER (Address Event Representation) protocol

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

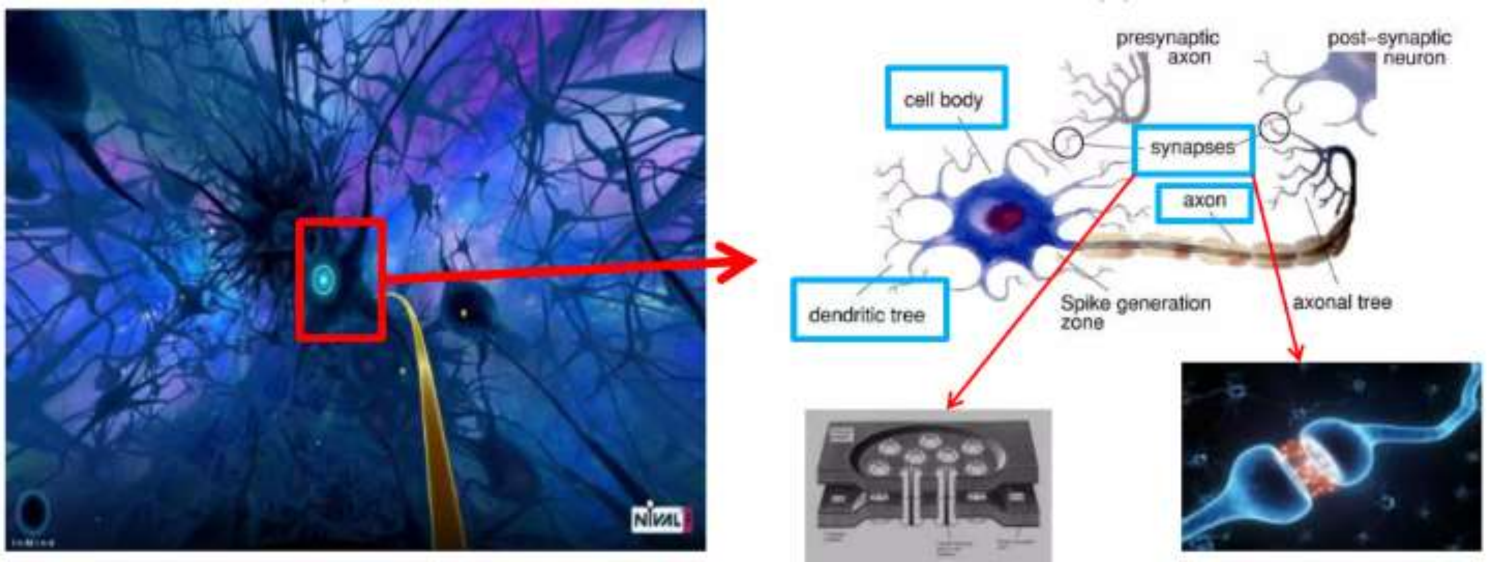


Fig. Complex Structure of a Neural Network [M.Bertrand,2015].

- 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**.

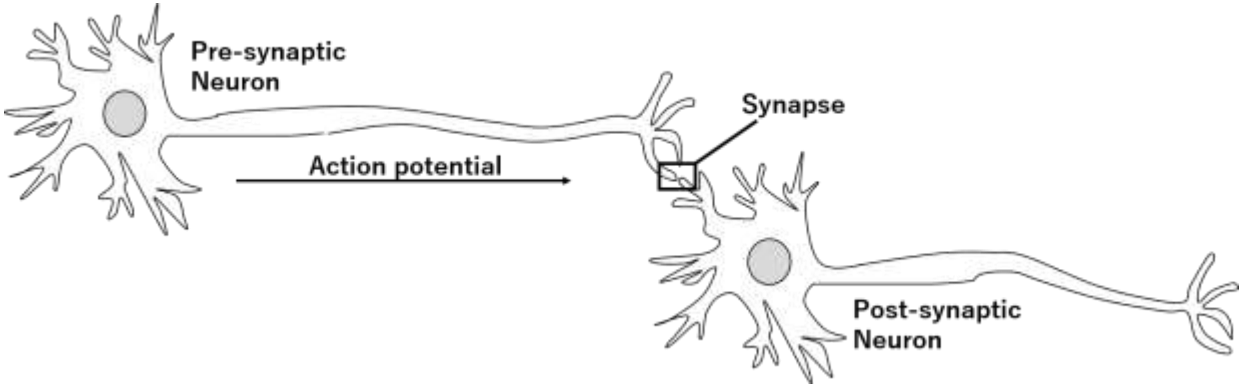


Fig. 2.1: Two neurons communicating via a synapse.

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

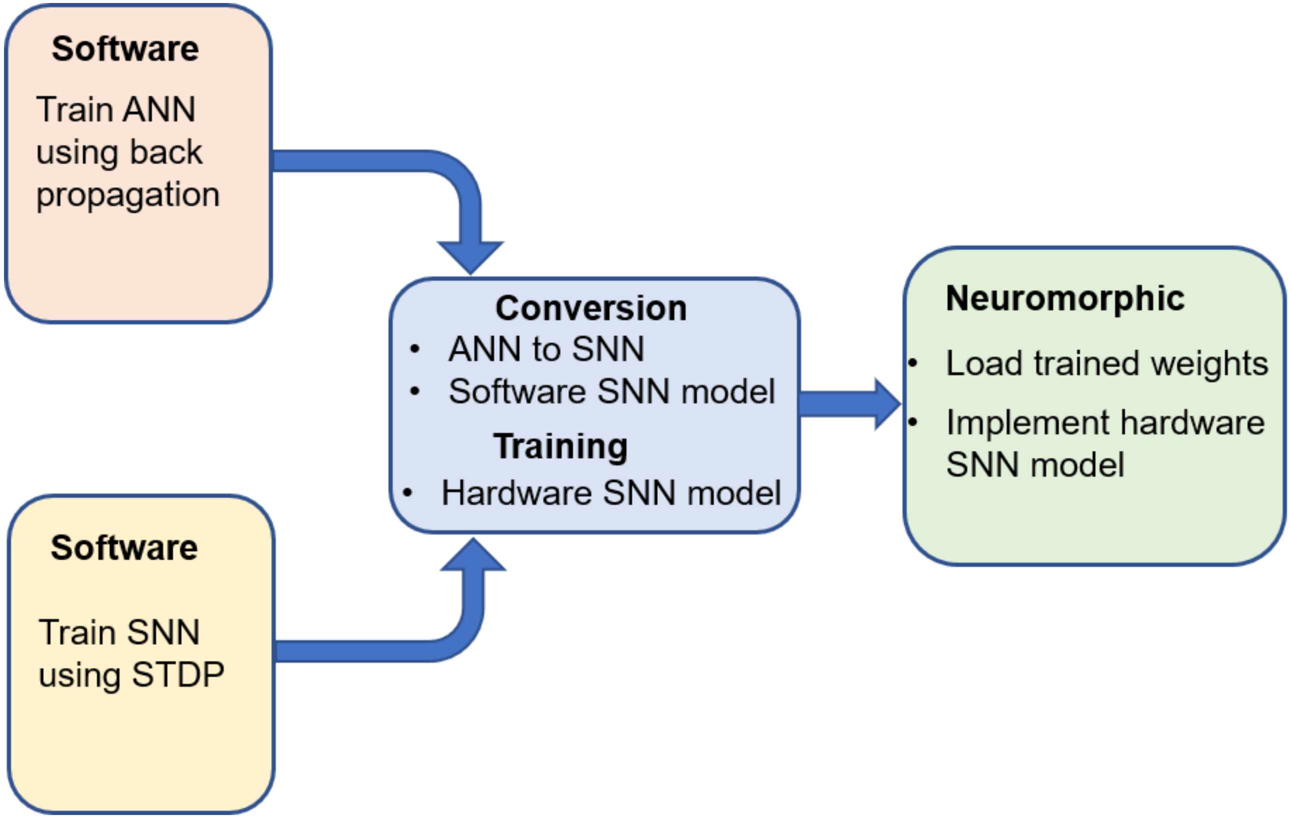
- Spiking neural network (SNN) processes and communicates sparse binary signals (spikes) in a highly parallel and event-driven manner.

- The learning phase (minimizes a particular cost (loss)), is a complex process of acquiring the parameters to output the correct inference results.

- The cost function optimization is performed with a gradient-descent-based optimization or other classical optimization methods (i.e., genetic algorithm).

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

4. Synaptic Plasticity Mechanisms & Learning: Learning Methods

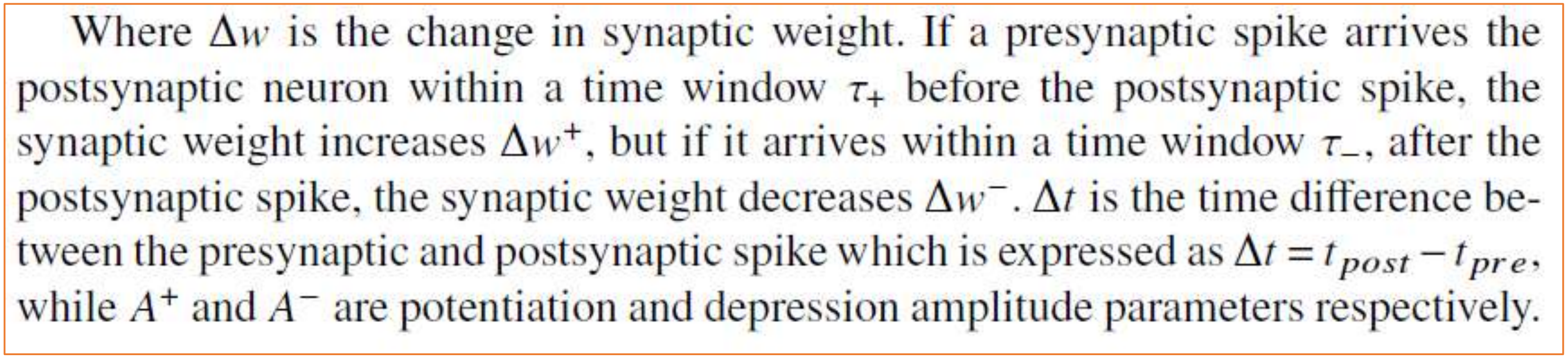


Neuromorphic Learning Framework

4. Synaptic Plasticity Mechanisms & Learning:
Spike-timing-dependent plasticity (STDP)

- Adjusts the strength of connections (synapses) between neurons in the brain.
  - ✓ 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)

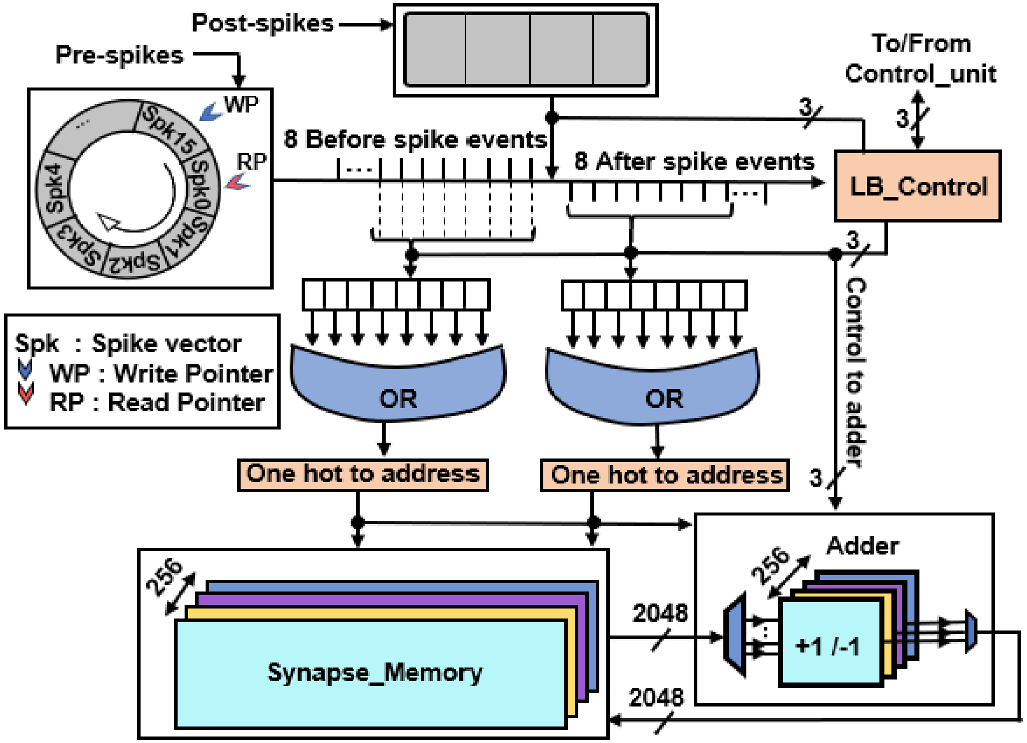


Fig. 1.5: STDP Architecture.

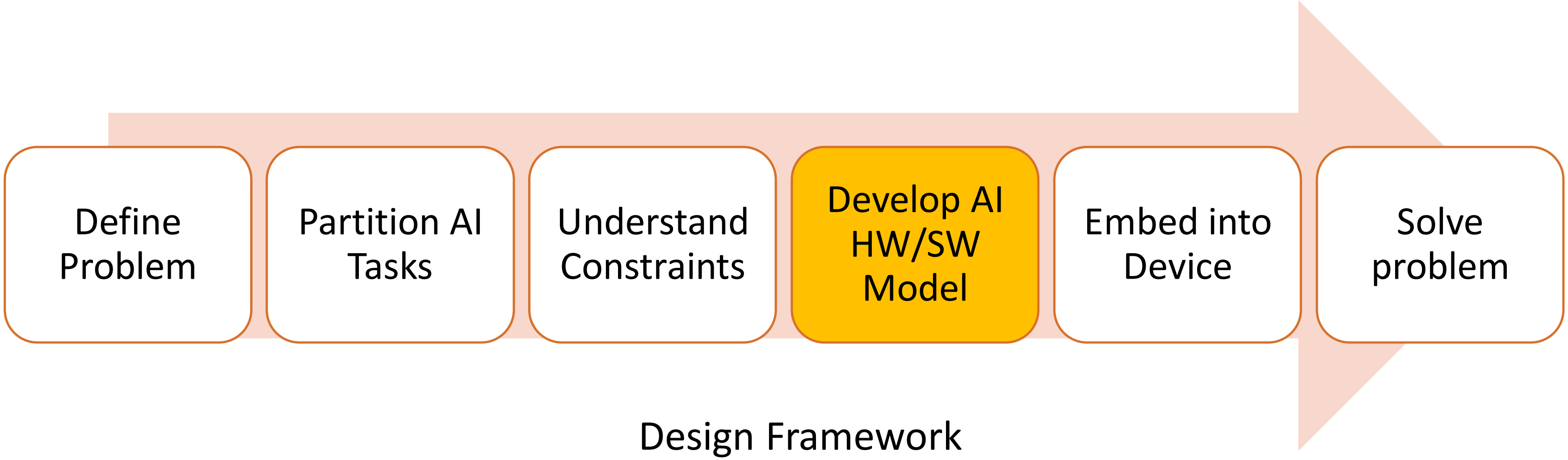
- The STDP unit Follows the *spike* or *pulse* model assumption for cortical neurons where information lies in spike timings, and not in spike shapes.
- 16 presynaptic traces are required to initiate the learning process. The PWU mechanism enables fast parallel on-chip learning.

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

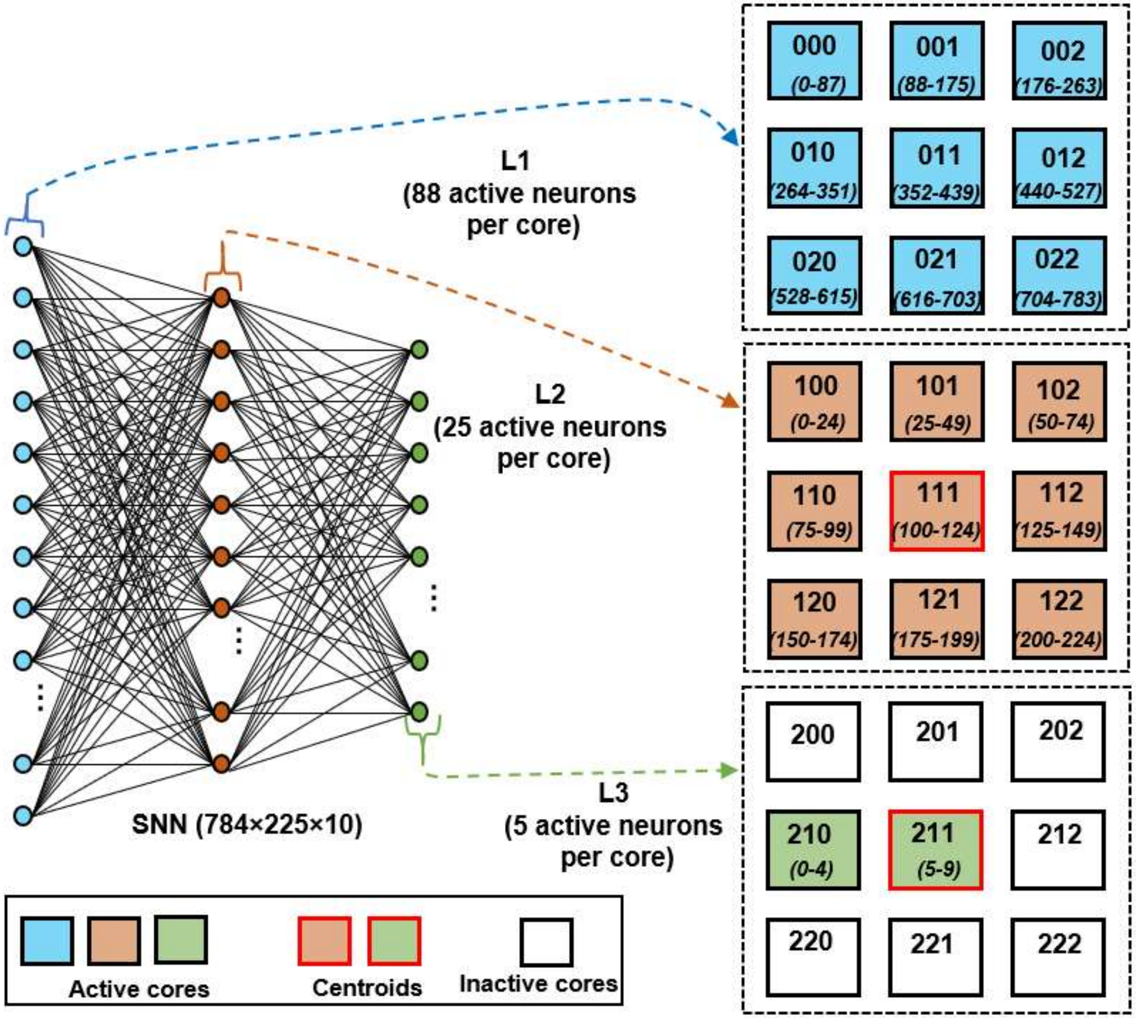


Design Framework

Define Problem→ Partition AI Tasks → Understand Constraints → Develop AI HW/SW Model -> Embed into Device -> Solve probler

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

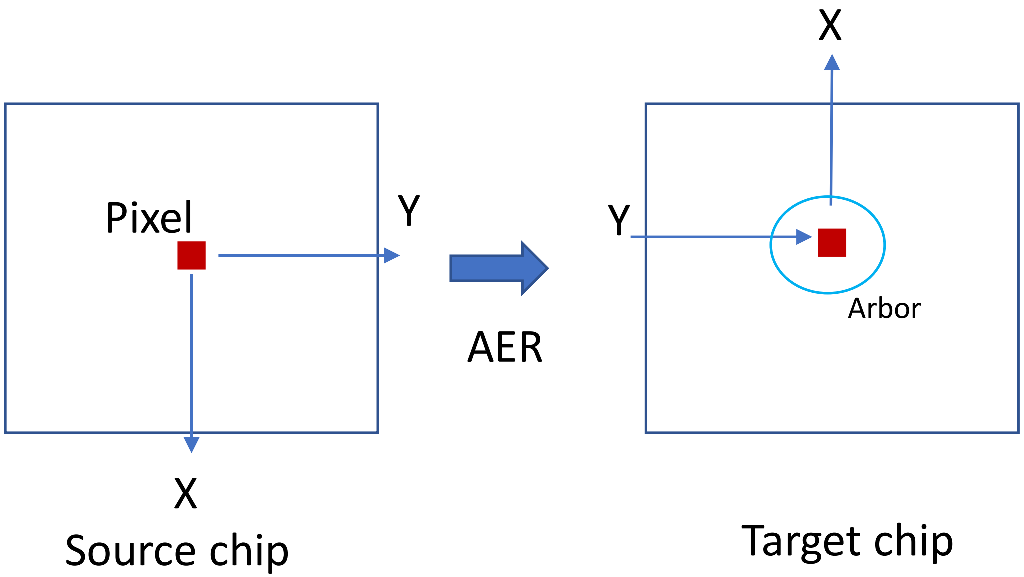
Active cores

Centroids

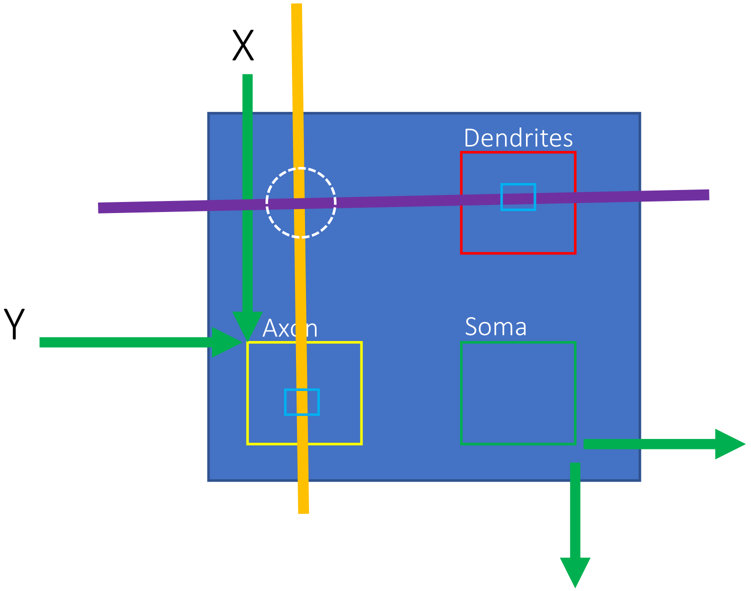
Inactive cores

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

5. Synthesizing Real-Time Neuromorphic Systems: Connecting Neuromorphic Chips

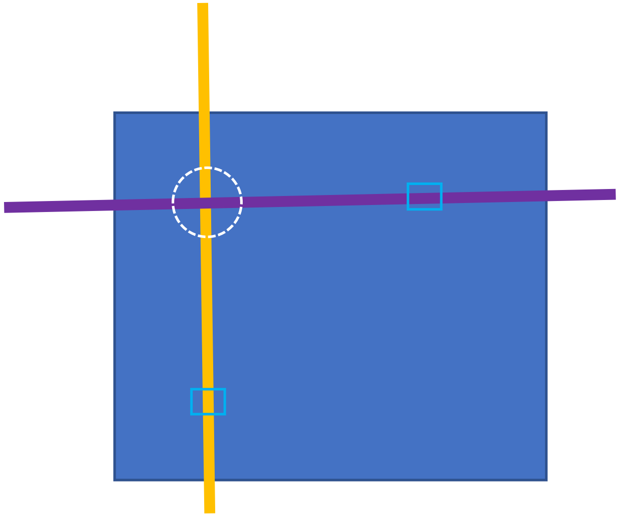


5. Synthesizing Real-Time Neuromorphic Systems:
Inside the Pixel



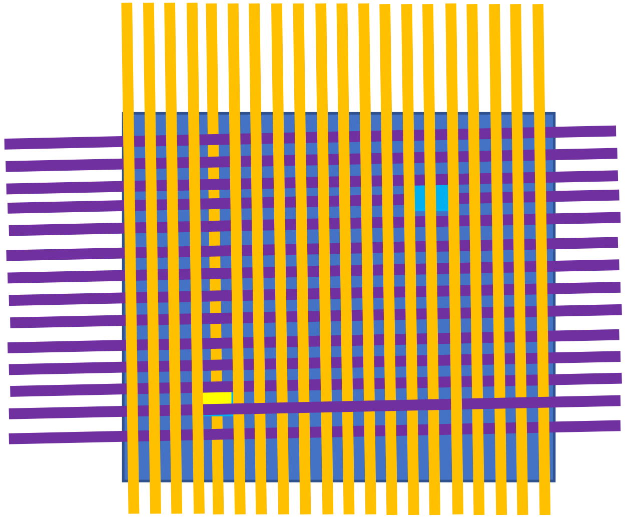
Inside the Pixel

5. Synthesizing Real-Time Neuromorphic Systems: Using Crossbars



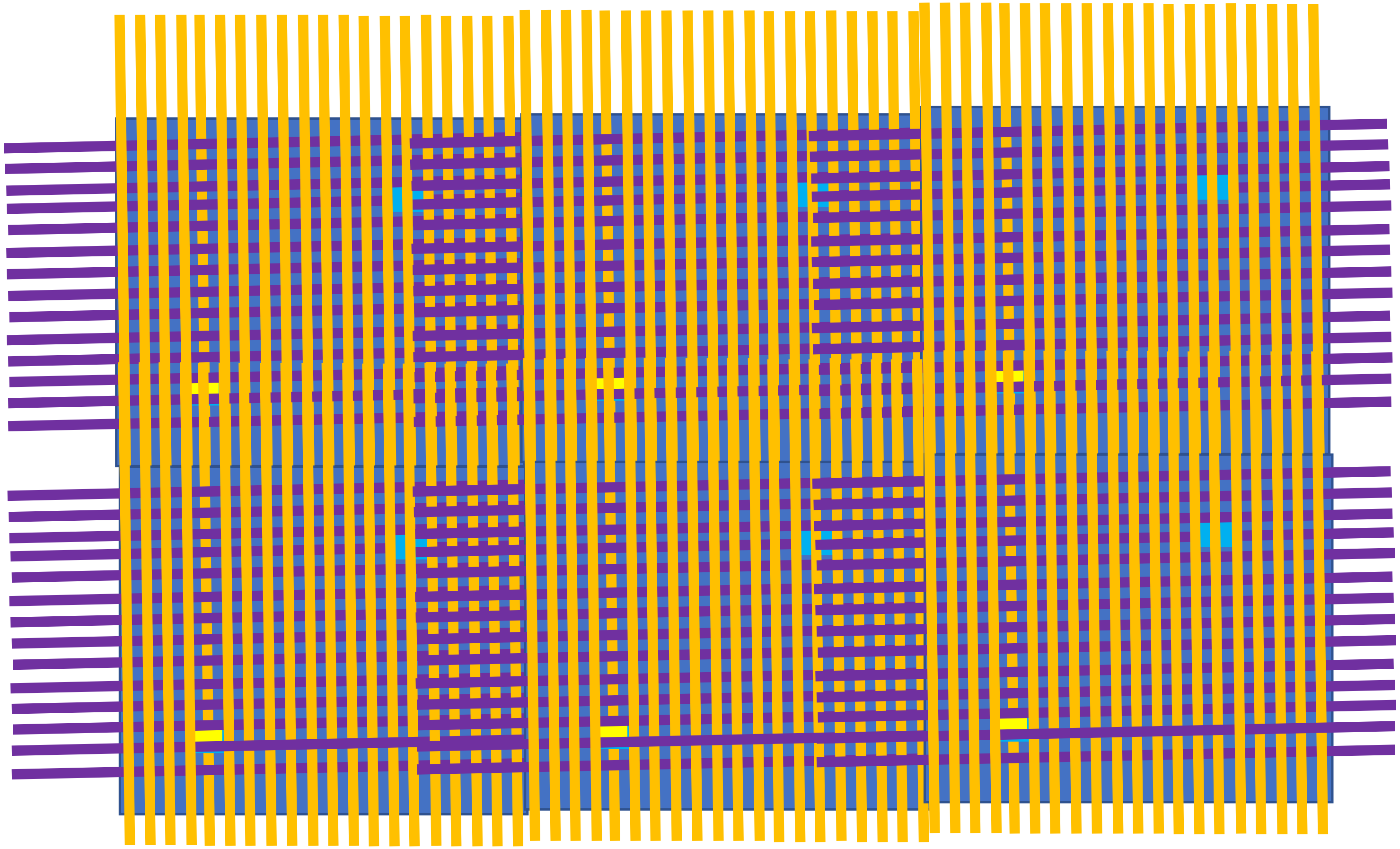
Using Crossbars

5. Synthesizing Real-Time Neuromorphic Systems: Using Crossbars



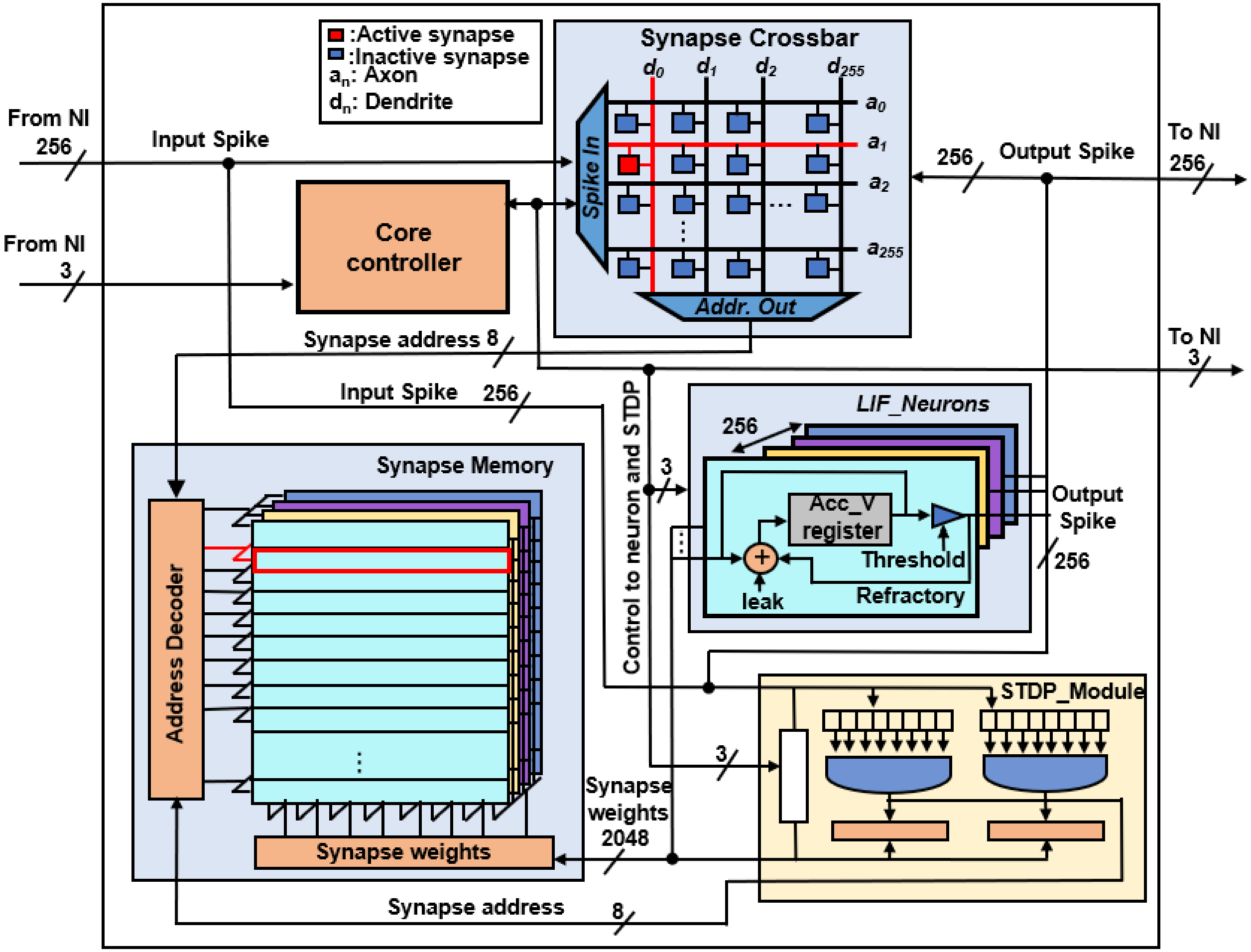
Using Crossbars

5. Synthesizing Real-Time Neuromorphic Systems: Using Crossbars



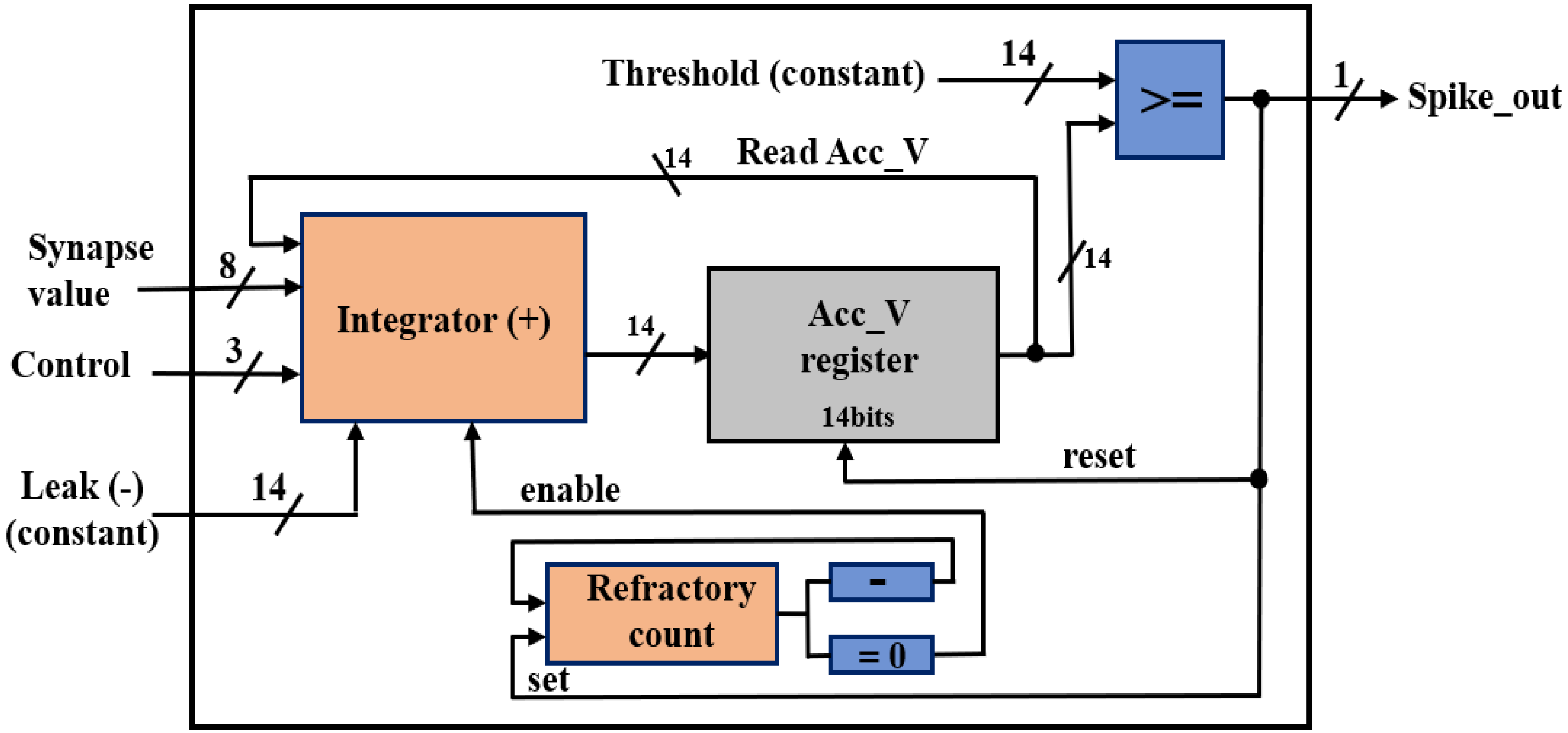
5. Synthesizing Real-Time Neuromorphic Systems:

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

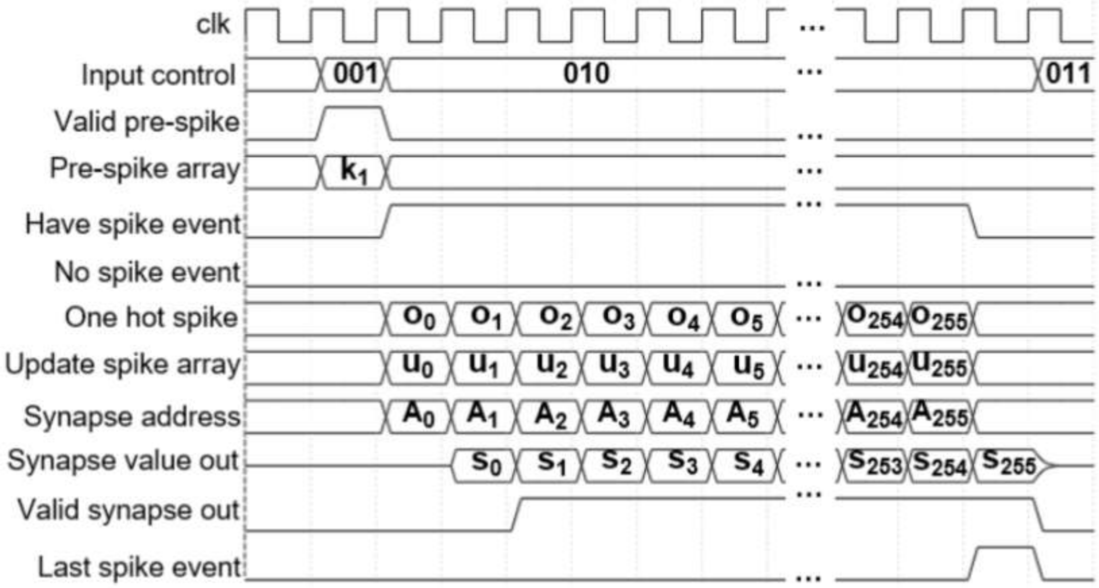

clk
Input control
Valid pre-spike
Pre-spike array
Have spike event
No spike event
One hot spike
Update spike array
Synapse address
Synapse value out
Valid synapse out
Last spike event
001
010
011
k_1
O_0 O_1 O_2 O_3 O_4 O_5 ... O_{254} O_{255}
U_0 U_1 U_2 U_3 U_4 U_5 ... U_{254} U_{255}
A_0 A_1 A_2 A_3 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

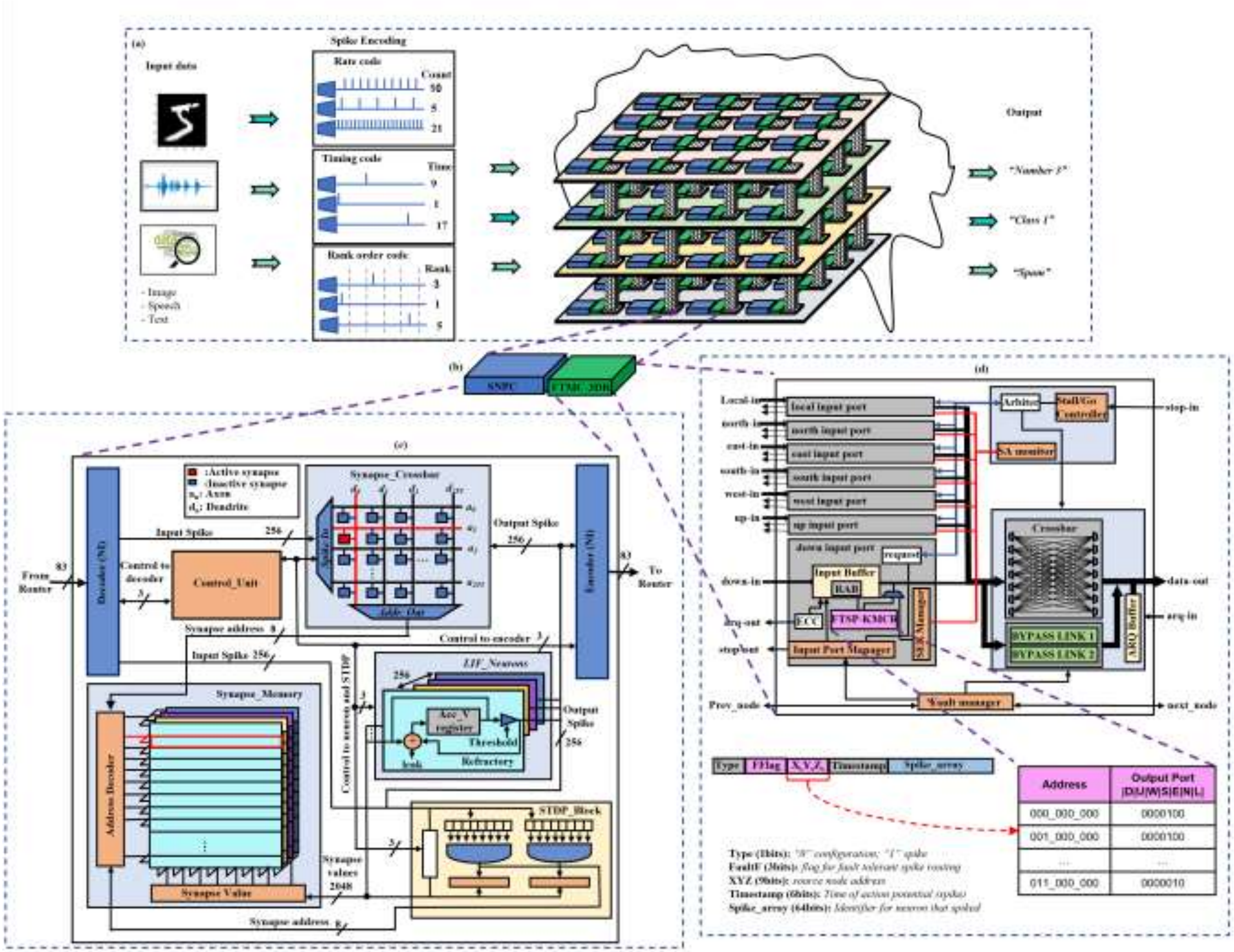
1. An input presynaptic spike array is stored and checked for spike events. If present, the ***Have spike*** event signal becomes high. Afterwards, the one hot operation to get the synapse address begins, updating the one hot spike array for every spike event: from ***O0 to O255***.

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 Architecture

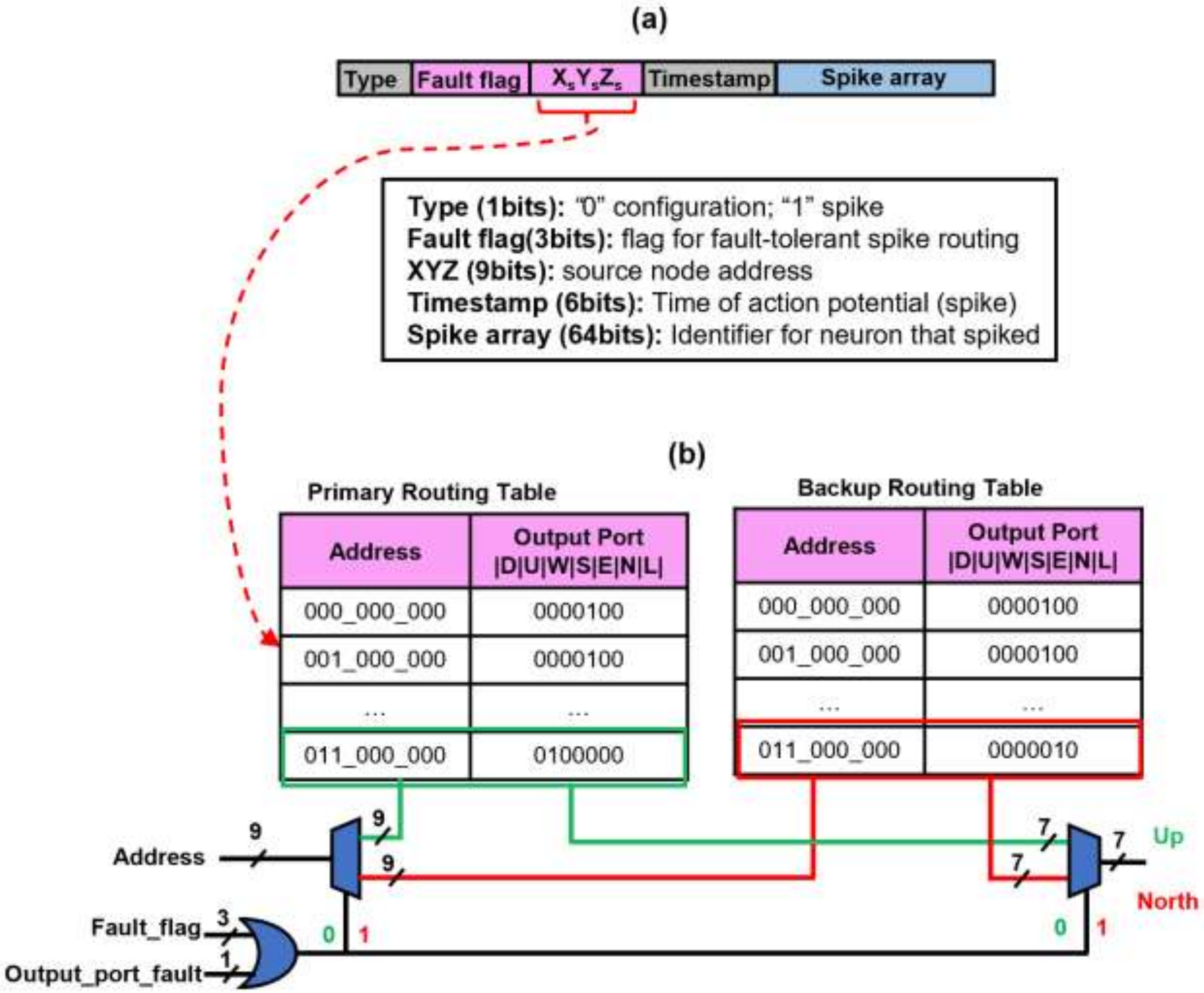


Address	Output Port [D|U|W|S|E|N|L]
000_000_000	0000100
001_000_000	0000100
...	...
011_000_000	0000010

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



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

Primary Routing Table

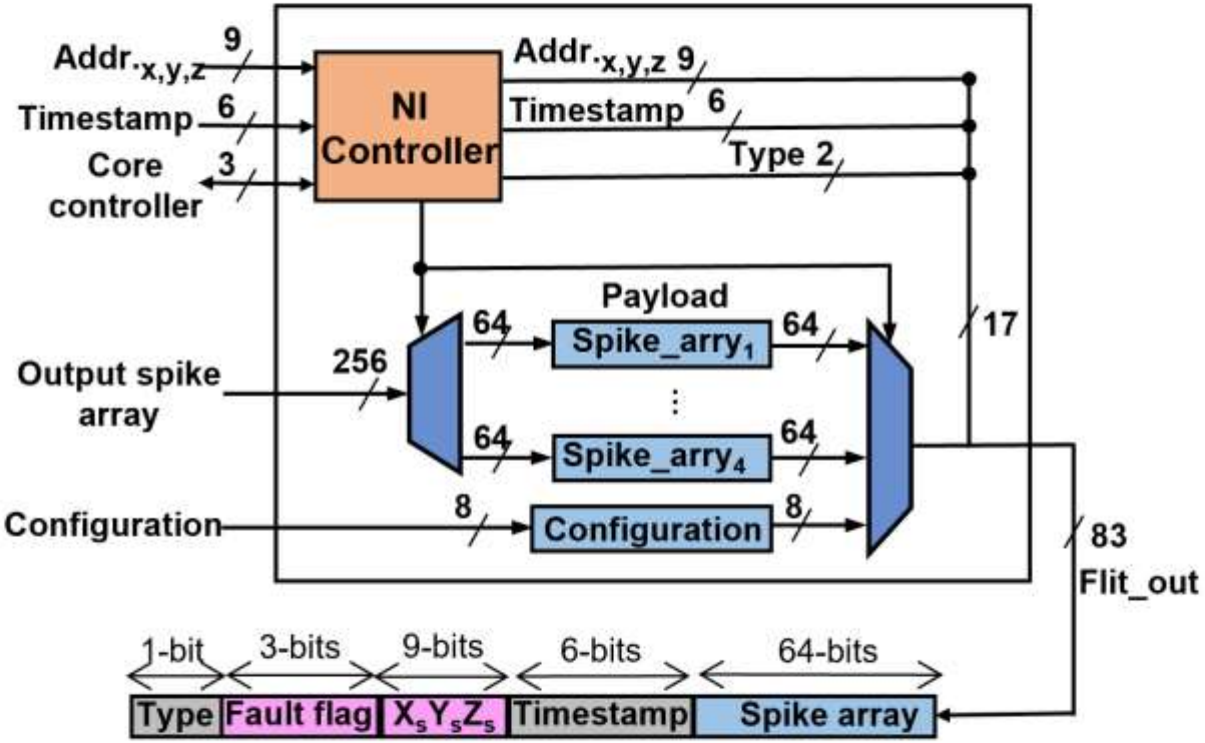
Address	Output Port |D|U|W|S|E|N|L|
000_000_000	0000100
001_000_000	0000100
...	...
011_000_000	0100000

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

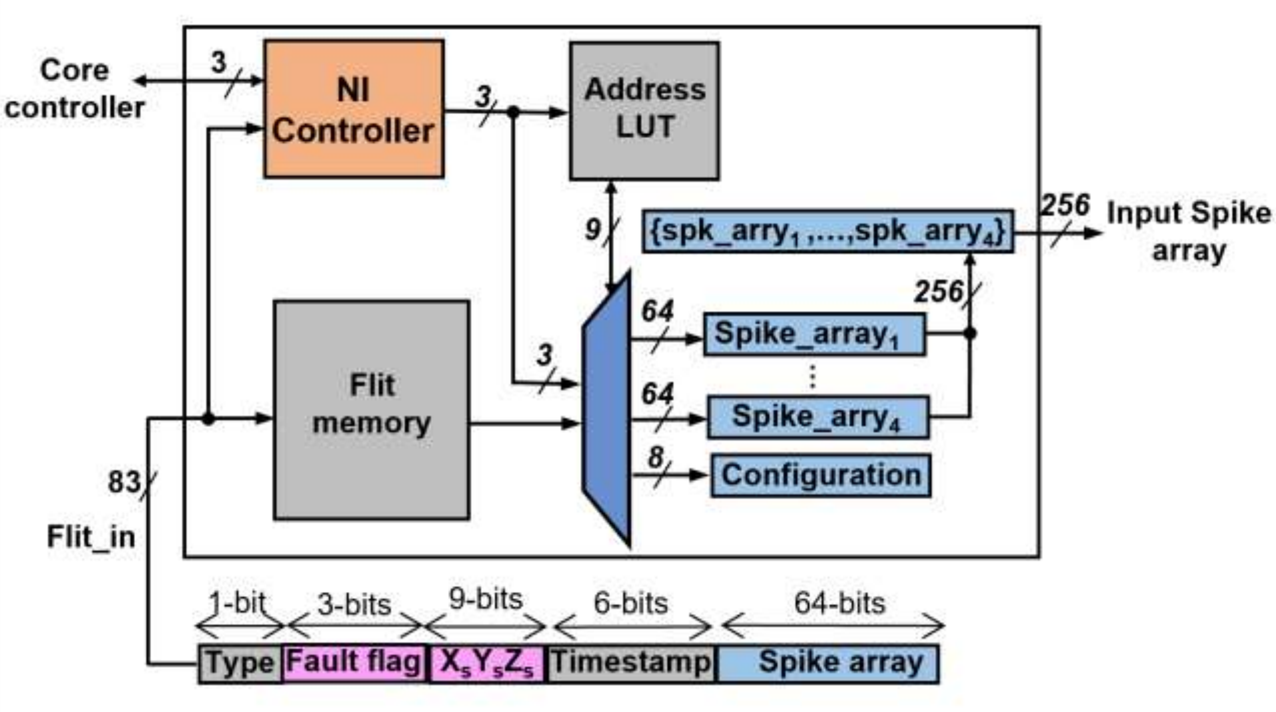


Network Interface: Encoder.

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

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:
  - 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 strength of coupling 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