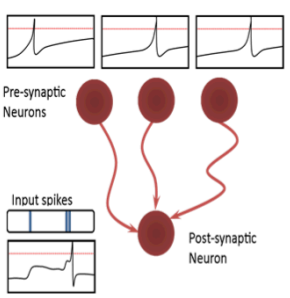
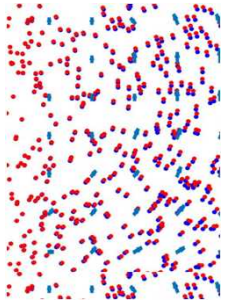
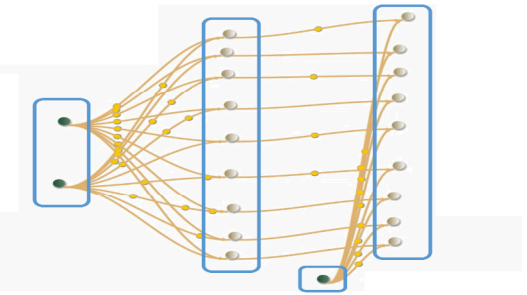
SAND2016-10419C







Spiking Network Algorithms for Scientific Computing

William Severa, Ojas Parekh, Kristofor D. Carlson, Conrad D. James, James B. Aimone

Center for Computing Research

Sandia National Laboratories

Albuquerque, NM

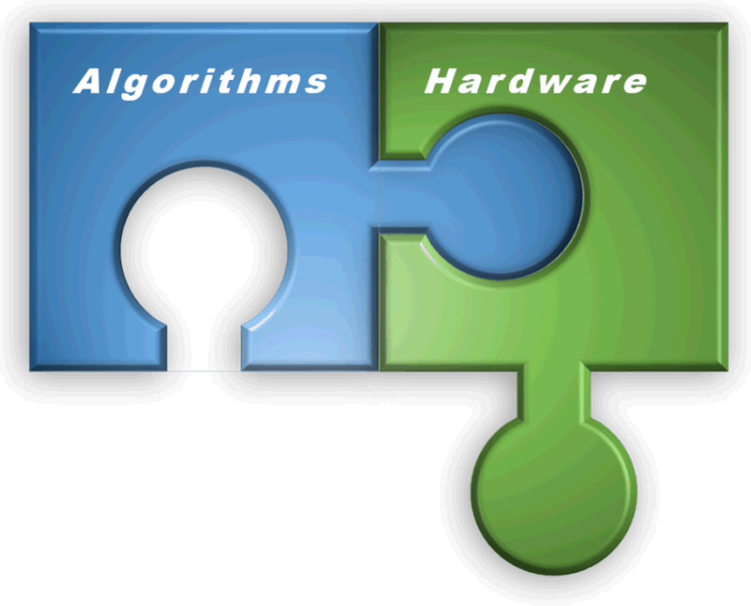
Sandia National Laboratories

Exceptional service in the national interest

Sandia National Laboratories is a multi-mission laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

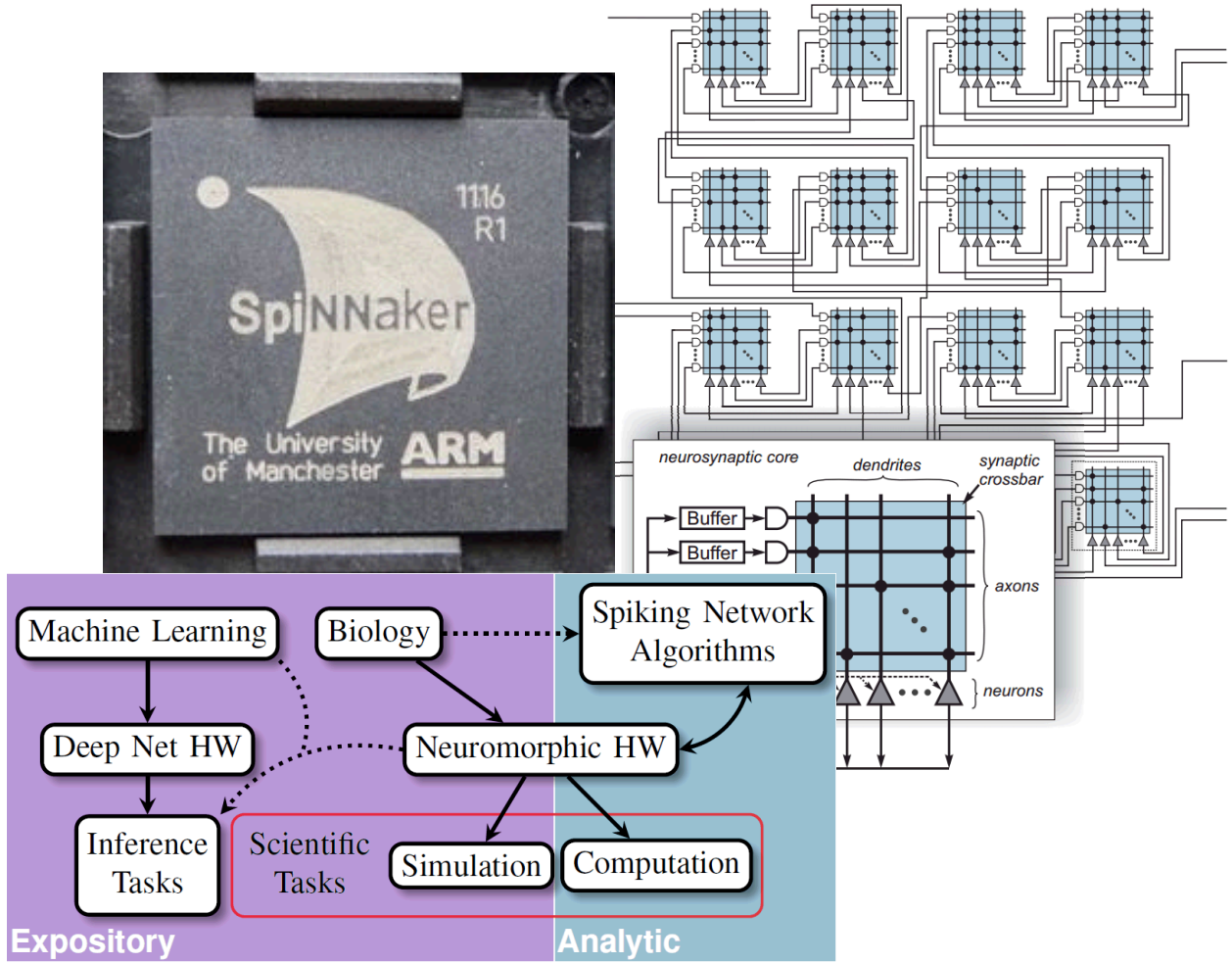
How to Make Good on Neural Computing's Promises

- Neural-inspired computing and neuromorphic hardware are poised to provide a Beyond Moore's Law pathway
- Human brain provides an existence proof for high-capability, low-energy neural computing
- Biology has provided strong inspiration
- Machine learning has provided an appealing application
- Algorithms must match hardware to take full advantage



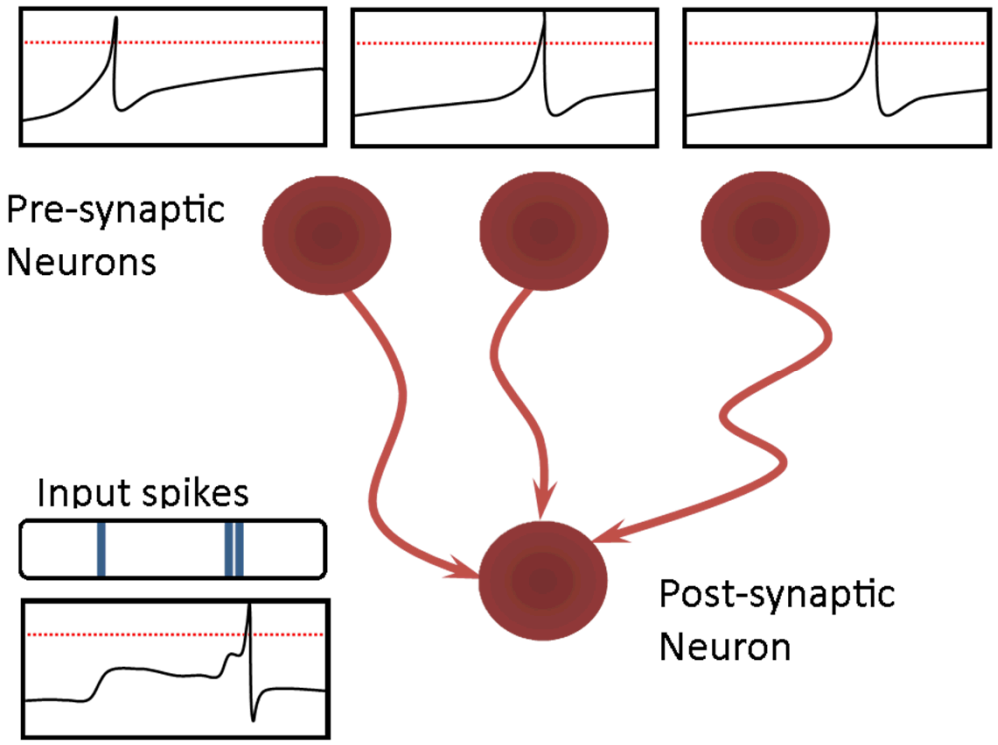
Neuromorphic Hardware Runs SNNs

- Inspiration from Neuroscience and Machine Learning
- Sophisticated circuitry required for collocated memory in spiking neural networks
- Hardware is (mostly) agnostic to the networks on it
- Low energy consumption; high performance communication
- Idea: Create abstract SNN algorithms using neurons as fundamental computing units
- Unlike biology, in silicon neurons, we can control precision
- Connections are `hand-crafted', not learned
- Expand neural network applications beyond neuroscience and machine learning



Spiking Basics

- Leaky Integrate and Fire neurons roughly approximate biological neurons
- Neurons are connected via synapses and communication is sent in single-state signals called spikes
- Spikes require time to propagate
- Time Dimension/Spikes are the main differentiator between Spiking Neural Networks and more basic Artificial Neural Networks
- Incoming spikes adjust an internal potential by some weight; if potential reaches a threshold, the neuron sends out spikes
- If potential is sub-threshold, it decays according to a leakage constant



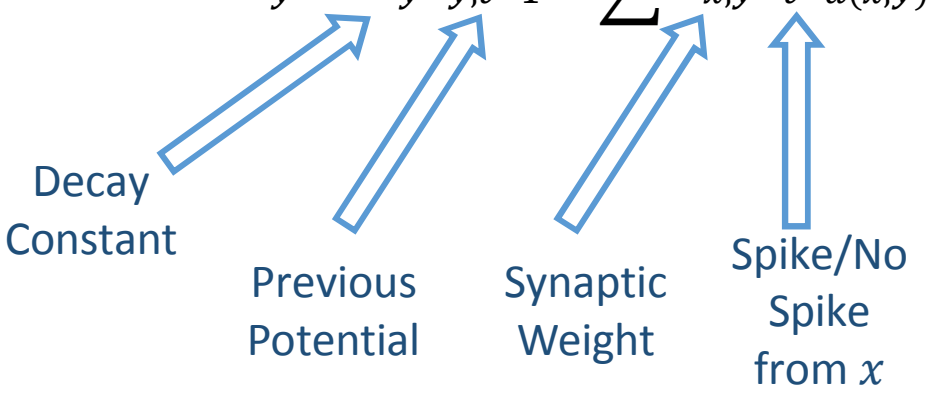
Technical Model

At each time step, each neuron integrates incoming spikes and determines whether or not to spike independently/

For pre-synaptic neurons x, post-synaptic neuron y, the neuron y spikes if and only if

$$\hat{V}_y = m_y V_{y,t-1} + \sum w_{x,y} x_{t-d(x,y)}$$

exceeds the threshold.

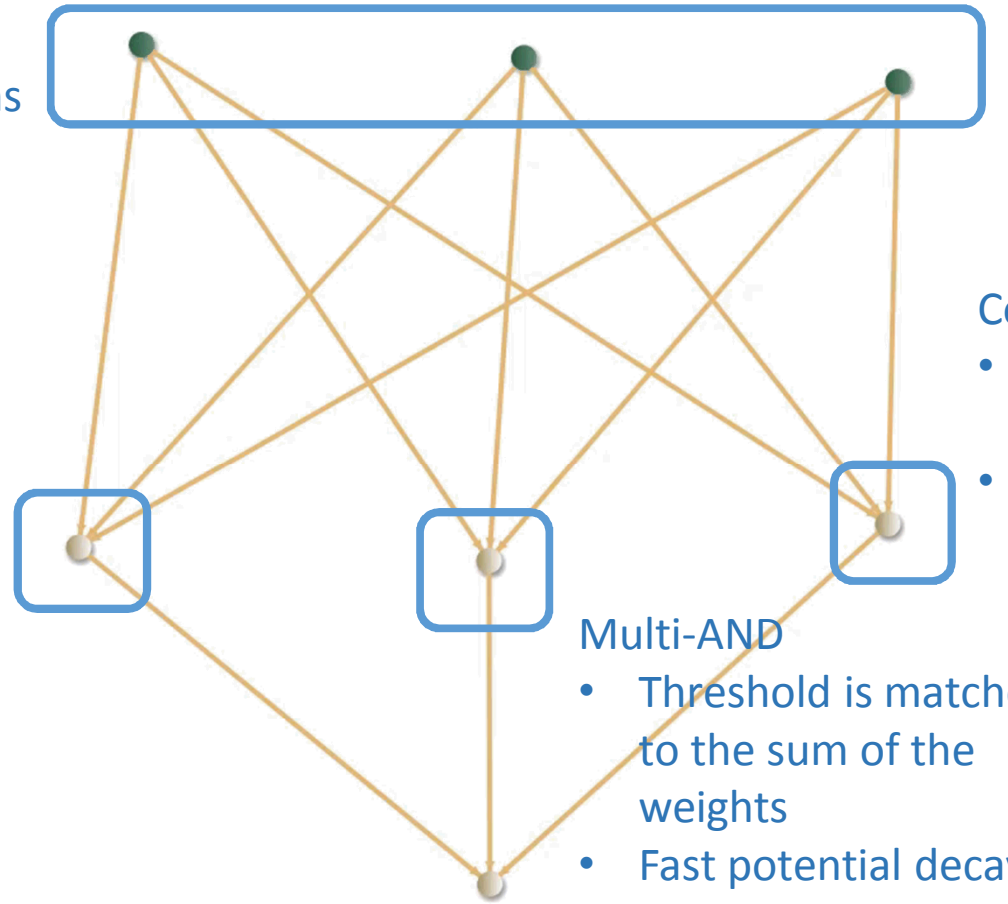


More advanced neuron models exist but have vague computational benefits.

Neural/Synaptic Properties Have Computational Benefits

- Weight and Threshold → Set Behavior and Gate Activity
- Delay → Adjust Time Sensitivity
- Parallelism → Simultaneous Multiple Conditions
- Decay → Temporal Error Correction
- Time dimension → Time/Neuron Tradeoff and Control Dynamics

Very Basic Examples



Multi-OR
- Threshold is low enough for any spike to trigger
- Fast potential decay; no memory

Counter (≥3)
- Threshold is matched to preset sum; fires if reached
- Slow potential decay; spikes persist over time

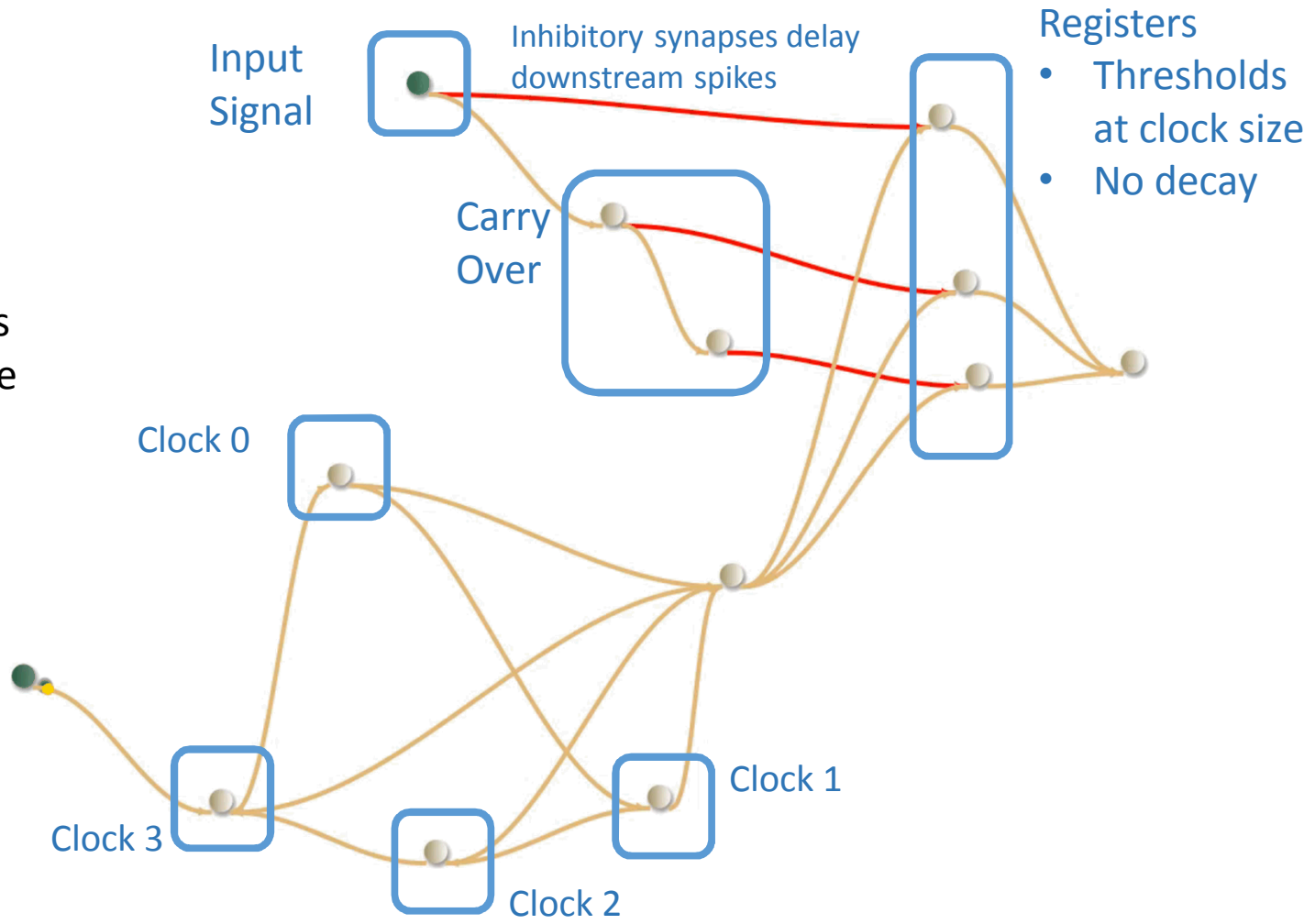
Multi-AND
- Threshold is matched to the sum of the weights
- Fast potential decay; no memory

But Better Building Blocks Needed

- Using neurons to create logic gates is circular and almost always inefficient
- Instead, we need modules that are inherently neural and appropriate for spiking neurons
- Some are easy to construct, like basic filters and sampling structures
- The best will leverage unique neuron and network properties, especially control of dynamics and connectivity

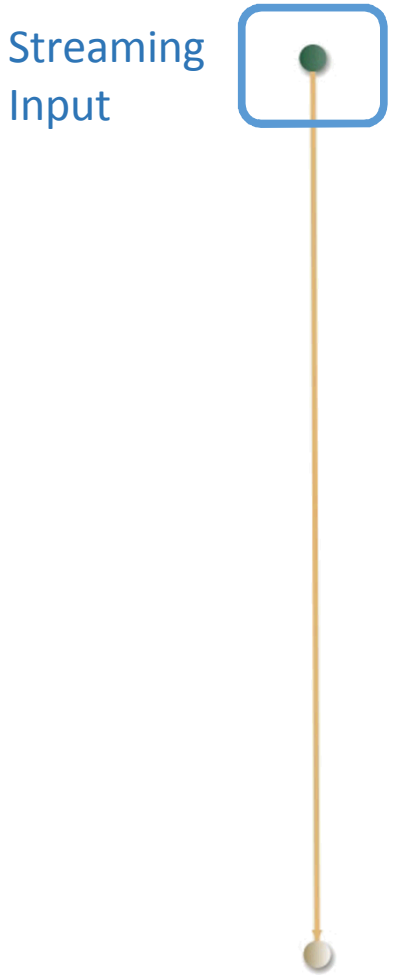
Another Building Block: Counting

- Time dimension also provides useful dynamics
- Spikes are single-state
- Here, store integer values in the dynamics of the system
- 4 position clock, 3 registers → 64 values
- Value of a register is read by the relative spike time compared to the clock
- Provides a fully spiking counter
- Can easily be modified to more sophisticated functions



A Simple Filter

- Neurons' decay can be used as a filter for activity levels
- Conversely, decay can be used to correct timing errors
- By tuning decay rate and thresholds, the post-synaptic neuron fires on, for example:
  - Two spikes that are consecutive
  - Two separated by one time step only
- All other spike patterns are filtered out

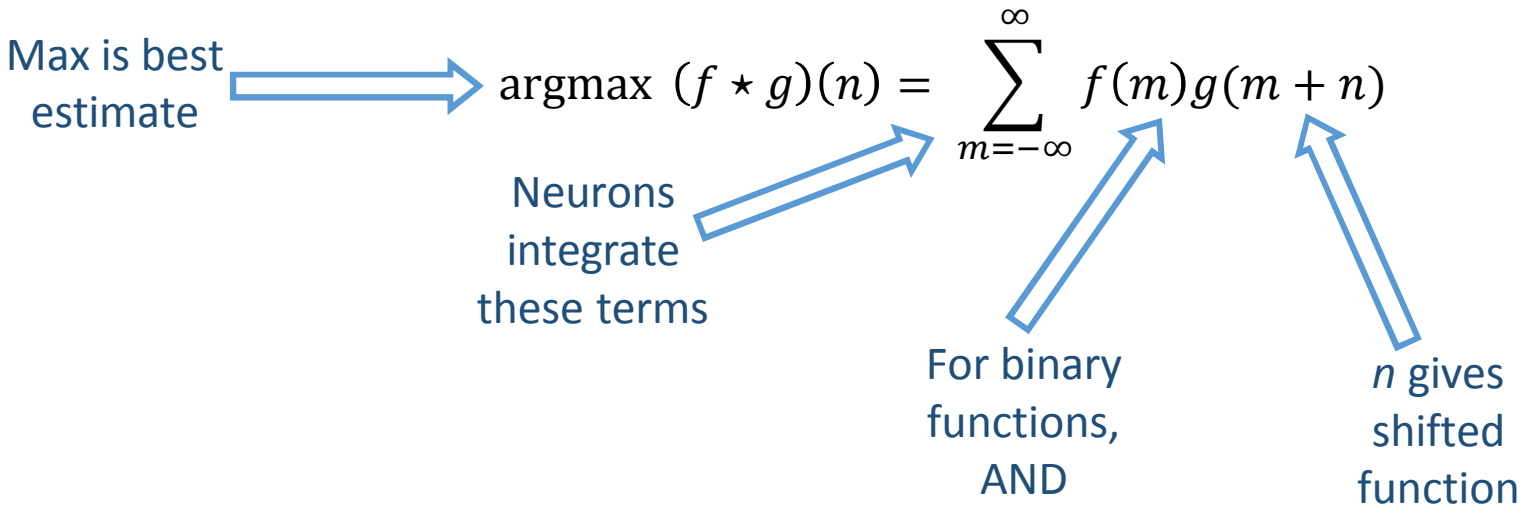


A Velocimetry Application

- A motivating application is the determination of the local velocity in a flow field
- The maximal cross-correlation between two sample images provides a velocity estimate
- SNN algorithms are straightforward; exemplify core concepts
  - Highly parallel
  - Different neural representations
  - Modular, precise connectivity
  - Time/Neuron tradeoff

One-Dimensional Cross Correlation

Essentially, finding the maximum cross-correlation is process in which we shift the images (functions) and find where the overlap is greatest. Then, the amount by which the functions were shifted provides the best estimate of the motion.



Sliding Inner Product Finds Overlap After Shift

Example:

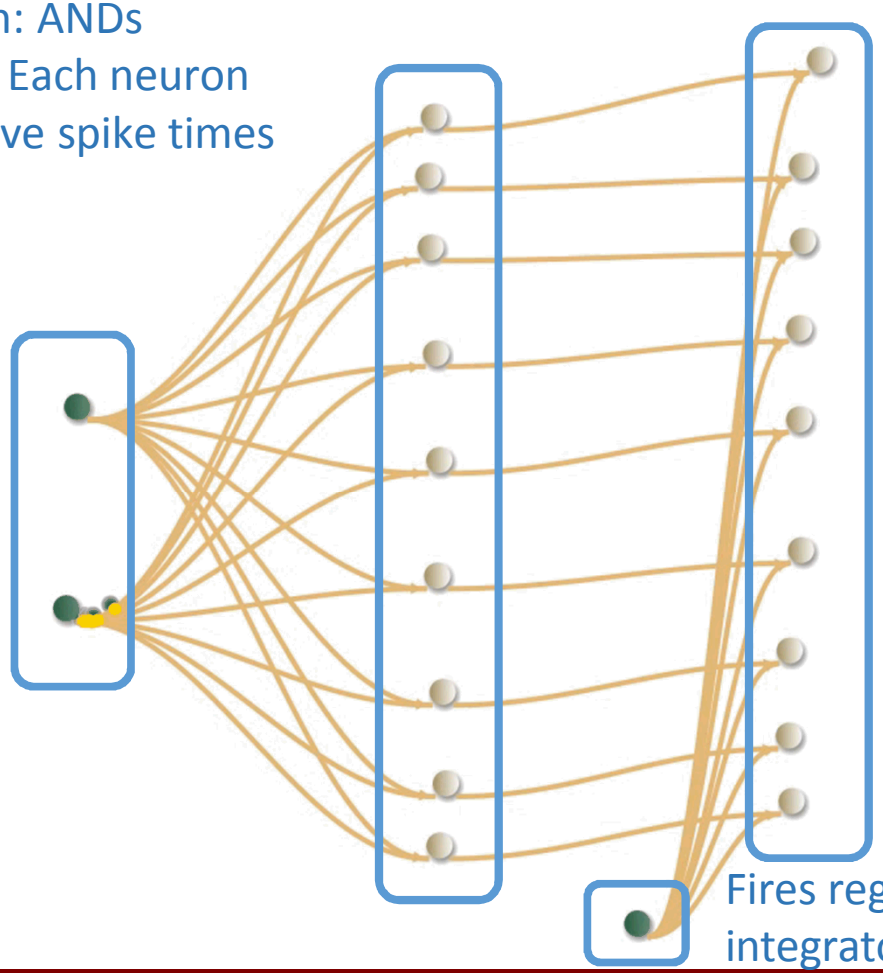
```
                                  f: 0,0,1,0,1,0,1,0
g:                      1,0,1,1,1,0,0,1

n: 7
Cross-Correlation: 0
```

Time Multiplexed Cross Correlation

Feature Detectors
- For direct cross-correlation: ANDs
- Graduated input delays → Each neuron sensitive to different relative spike times
- *Rate Coding*

Time-coded Inputs
- Neuron fires at time t if function is 1 at time t
- Two input functions, two neurons
- *Temporal Coding*



Fires regularly; forces integrator to fire

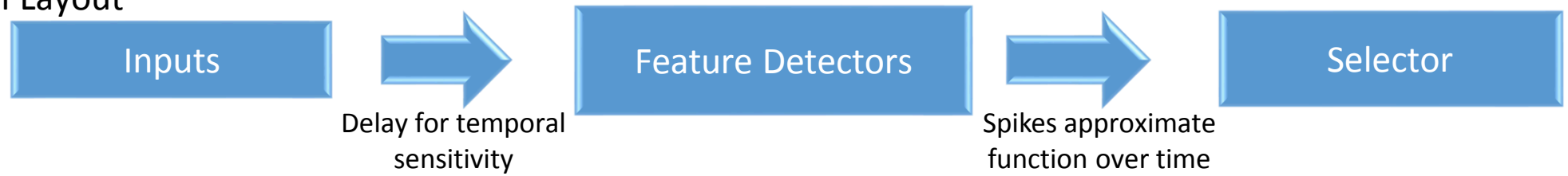
Integrators
- Counts spikes from previous layer
- No decay, high threshold
- First to fire is maximum with sufficient threshold
- *Latency Coding*

Temporal Coding: $O(n)$ neurons; $O(n)$ runtime

Parallelize inputs and corresponding timesteps to achieve $O(n^2)$ neurons; $O(1)$ runtime

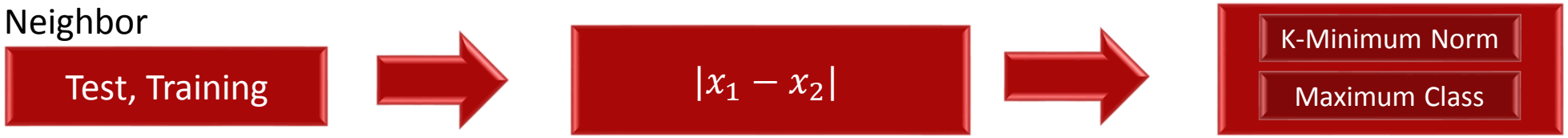
Basic Building Blocks Expand Capabilities

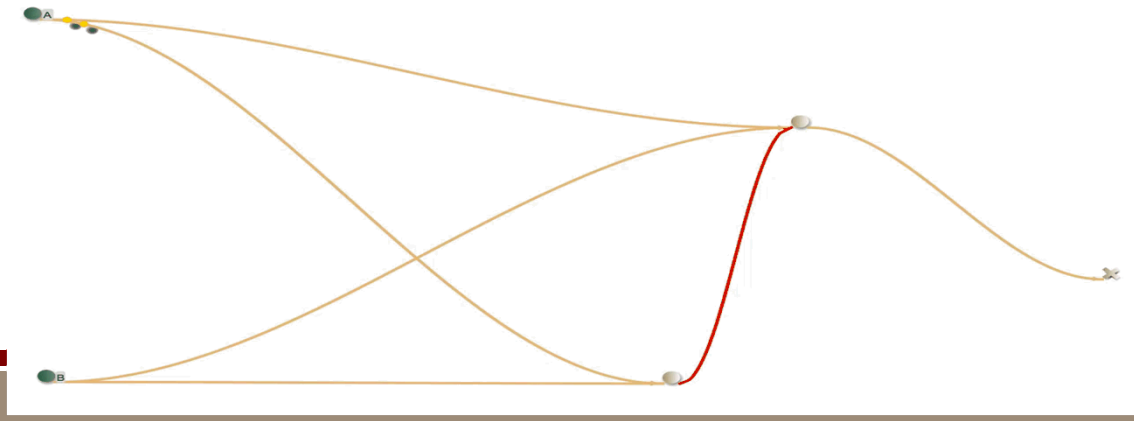
Functional Layout



Cross Correlation

K-Nearest Neighbor





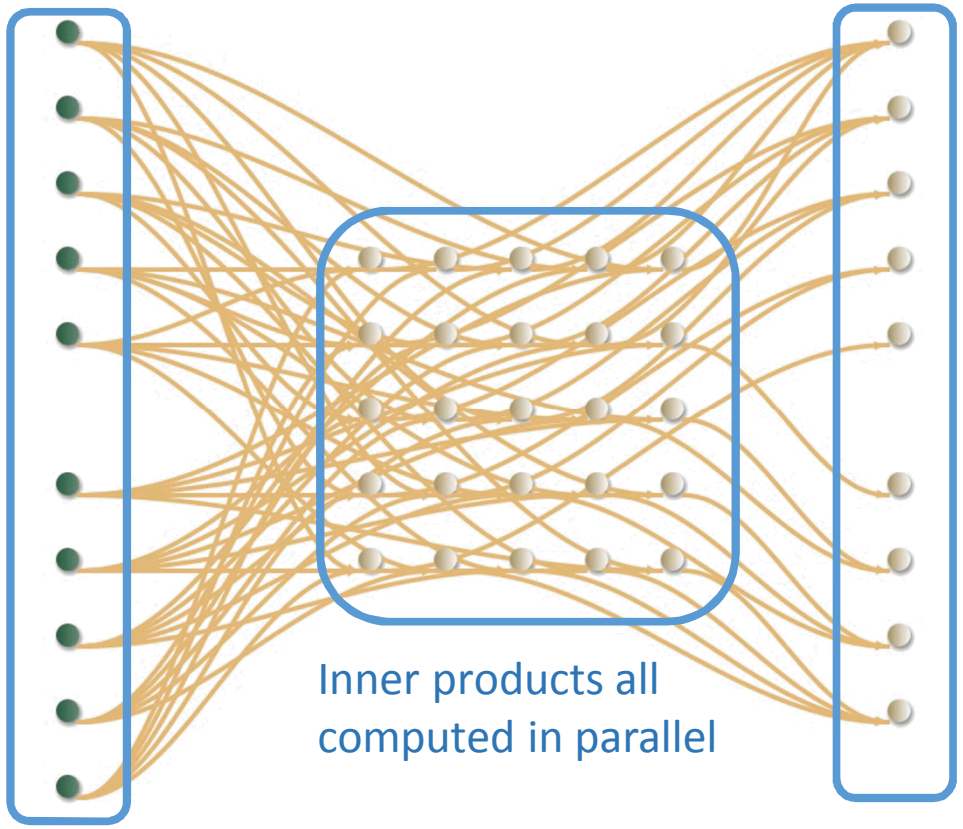
Cross-Correlation Exhibits Time/Neuron Tradeoff

- Neuromorphic hardware is massively parallel
- Exchange Time Cost ↔ Neuron Cost
- At times, can construct constant-time algorithms
- No free lunch; Complexity is unchanged
- Some representations are better suited than others
- For constant-time cross-correlation: new max method must be used $O(n^2)$
- Neurons: $O(n^2) \leftrightarrow O(n)$
- Time: $O(1) \leftrightarrow O(n)$

Inputs

- One neuron per function per dimension

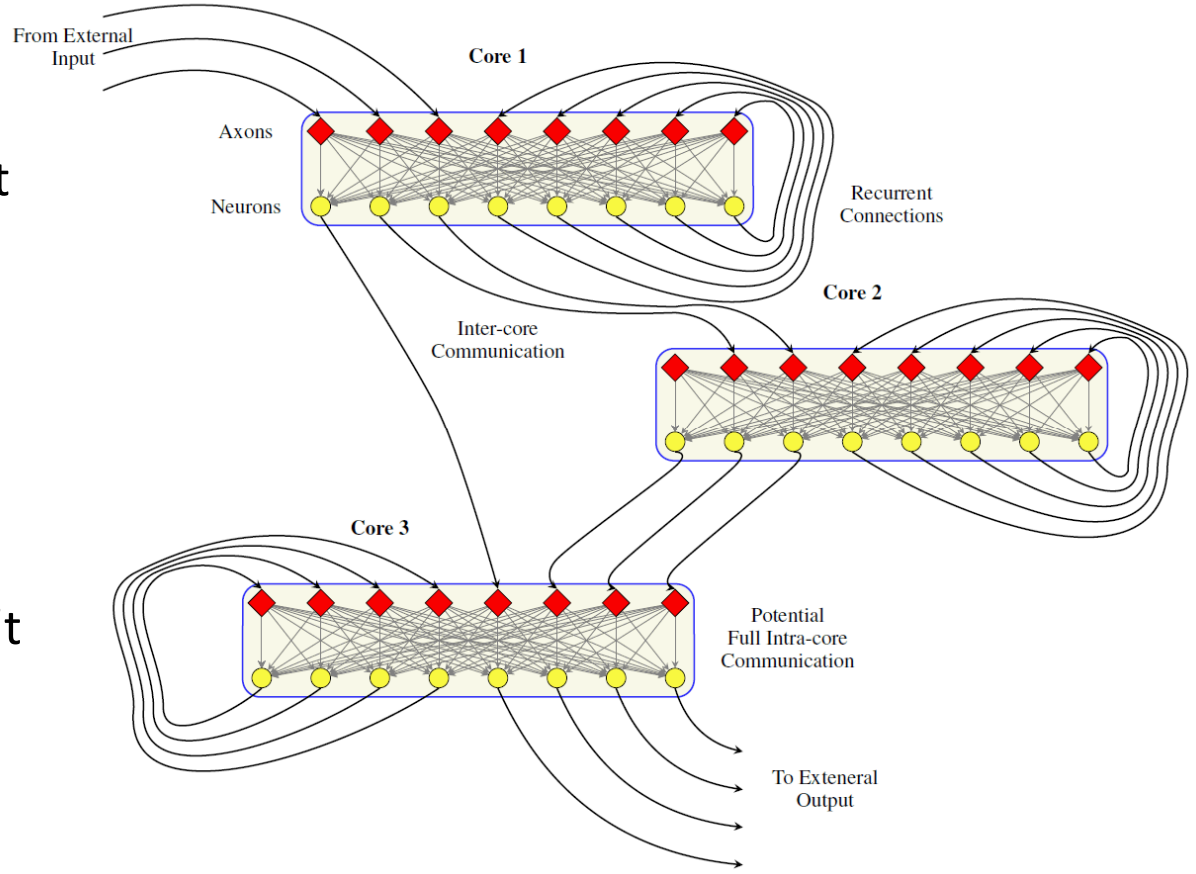
Output signal routed to Argmax



Inner products all computed in parallel

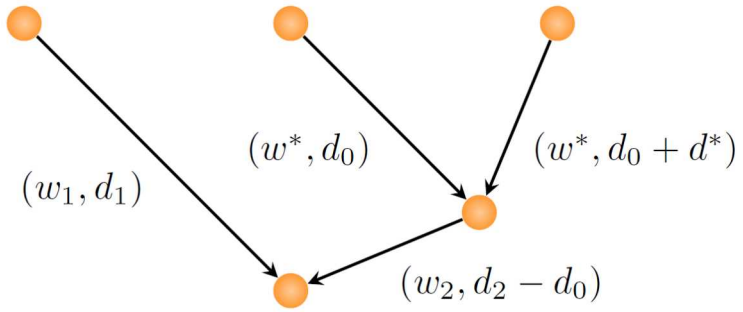
Looking to Hardware Implementation (TrueNorth)

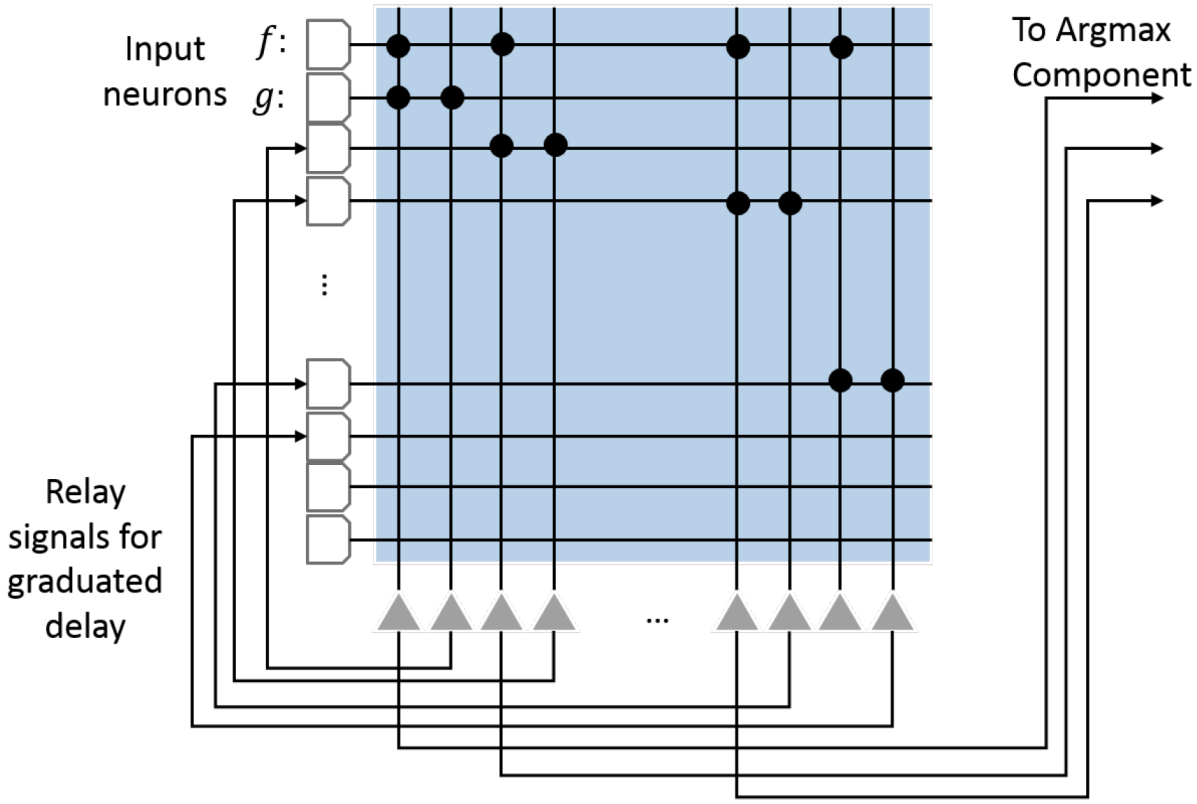
- Full recurrence only within neurosynaptic cores
- Output of neuron may only go to one input axon
- Weight is shared along input axon
- Neurons with multiple outgoing weights is expensive
- Need to maximize use of cores (Corelets are built of whole cores)
- When scaling $O(n^2)$, a million neurons isn't that many



Axon Restrictions Complicate Connectivity

- Neuron outputs have only one ‘axon’
- Shared synaptic properties (same signal to all downstream neurons)
- Need auxiliary neurons to split signal
- For delay, no added time cost
- For weight, added time cost proportional to minimum delay
- Similar techniques can address limited fan-in (in some cases)





How should we program these?

- Still an open problem
- Software simulators are reasonably well established
  - Nest
  - Neuron
  - Brian
  - CARLsim
- Sophisticated neuron models/dozens of parameters not needed
- Rarely does software graph translate directly to hardware
- No framework to ‘debug’ neural algorithms

Summary

- Spiking neural networks can perform exact computation
- Connectivity need not be driven by biology or training
- Communication and memory scheme of neuromorphic hardware leads to computational benefits
- Certain utility afforded by different Neuronal/Synaptic properties has been explored, but can be explored further
- Putting algorithms to hardware shows limitations of architectures
- Theory → Software → Hardware workflow must be improved