Implementing Neural Models in Silicon

Tutorial, BICS 2004

August 29 2004

Leslie S Smith, Department of Computing Science and Mathematics, University of Stirling.

lss@cs.stir.ac.uk

Overview

- Why silicon implementation
- Real neurons (a very quick introduction)
- What to implement
- Implementation Technologies
- Spiking systems
- Synapses
- Concluding thoughts

Why implement neural models in silicon?

- To gain better (and possibly brain-like) performance for some system
- To study how some neural model performs

- Brain-like performance would be a real improvement in some areas
  - Sensing, motor control, higher level capabilities
- Silicon implementation can allow models to run rapidly enough to be applied to real data in real time
  - Clearly important for sensing and motor control: but also for making decisions in changing circumstances
- Why not just wait for workstations to get fast enough?
  - Real sensory systems are highly parallel, multiple channels of information flow
  - It might be a long wait!

Computational Neuroscience and Silicon Implementation

- Modelling is critical element in testing hypotheses in computational neuroscience

- What are the advantages of silicon over software?

  - Speed
  - Allows for testing models in computational neuroscience against real data

- Are these advantages enough to overcome the disadvantages

(e.g. difficulty in modifying systems, delays in manufacture, etc.)

- Answer: sometimes we want to be able to use these models in conjunction with other equipment
  - Then real-time operation, power consumption, and even portability can matter

Real Neurons

- Real neurons are very complex.
  - See Kandel, Schwartz and Jessell: Principles of Neural Science
- There are many different types of neuron
  - Often classified by morphology (shape), extent, location in brain, type of animal in which they occur
- What they all share is excitability
  - Operation through electric charge
  - The membrane of the cell...
    - its outermost boundary
  - ...is excitable
  - That is, its properties change depending on the voltage across it.

Overall neuron structure

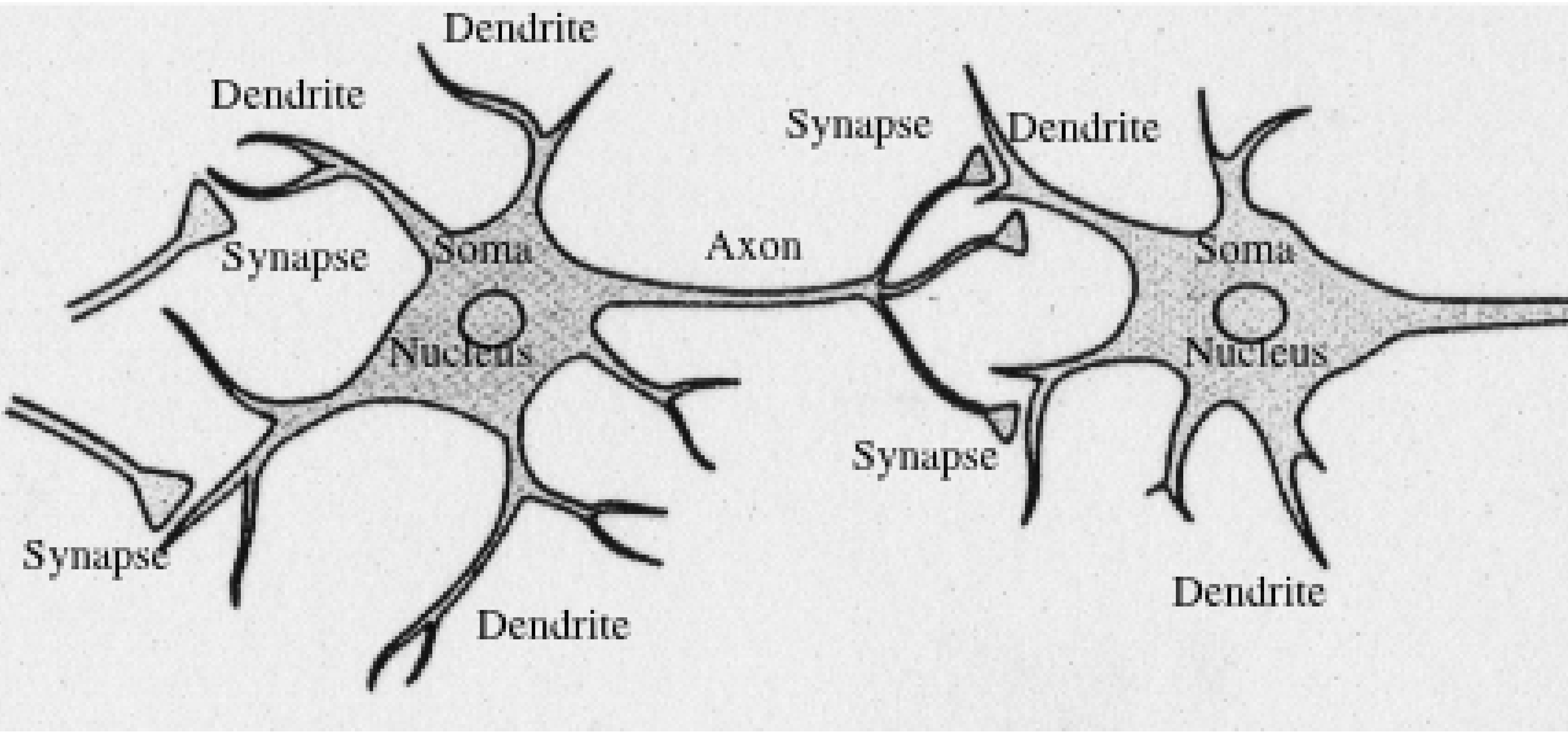
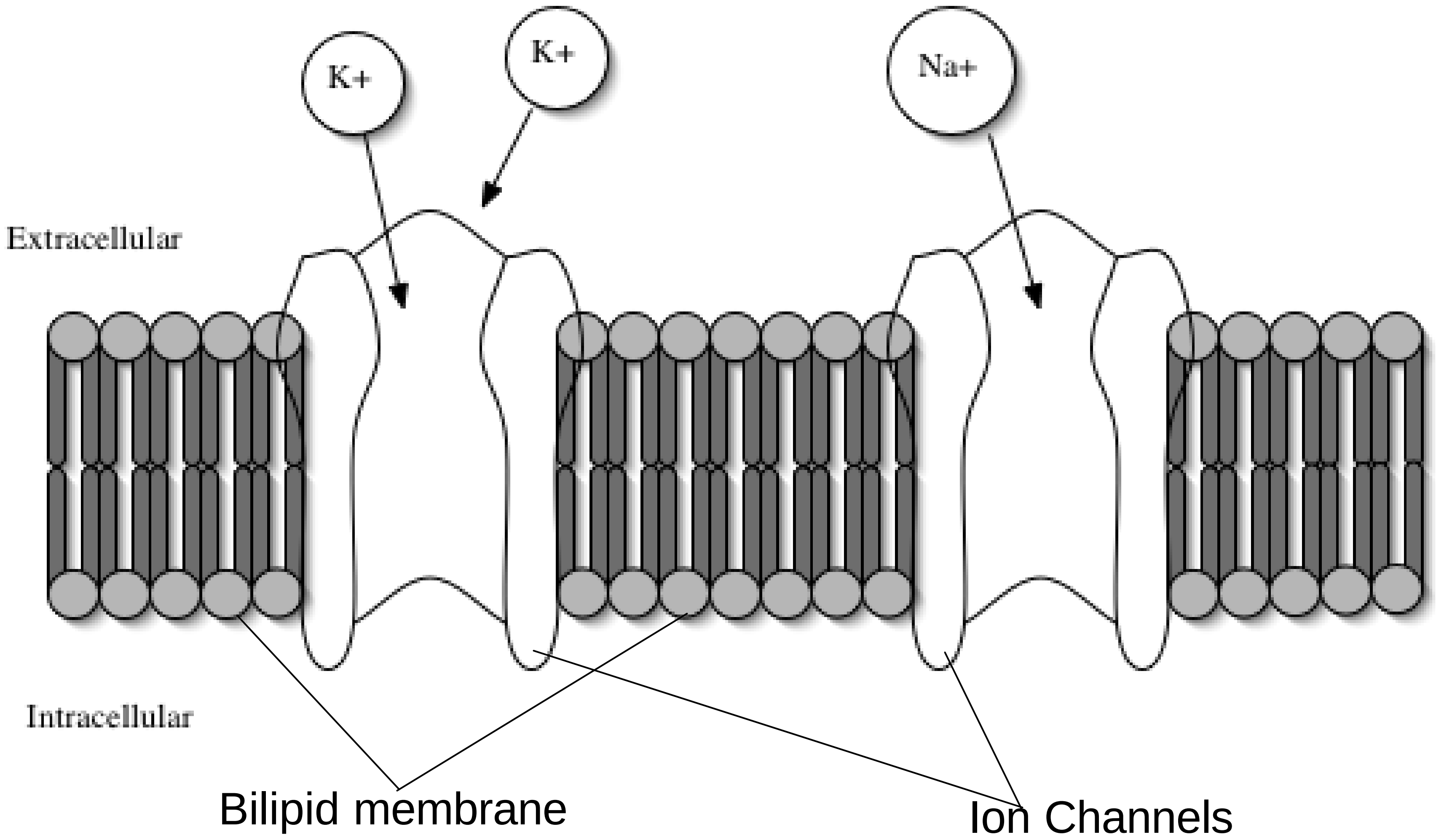


Figure modified from Kandel

Cell membrane



Cell membrane

- Bilipid part is insulator
- It forms the dielectric of a capacitor
  - Between the relatively conducting extracellular and intracellular fluid

- Conductance is through transport of charged ions
  - Ca++, Na+, K+
  - Through ion channels
    - Proteins which allow ions through selectively

Ion Channels

- Ion channels come in many forms
  - Gerstner refers to them as the “zoo of ion channels”
- Unbalanced movement of ions across the membrane alters the potential difference between the inside of the neuron and the outside of the cell
- The permeability of an ion channel depends on the configuration of the protein that forms the channel
- This configuration can change because of the potential across the ion channel
  - Voltage-sensitive ion channels
- In addition, many ion channels change their configuration (and hence permeability) over time, or as a result of passing ions through, or because they can be blocked by particular ions.

Reversal potential

- Most ion channels could pass ions in either direction

- Two factors influence the direction
  - The potential across the membrane
  - The relative concentration of the ion species at either end of the channel

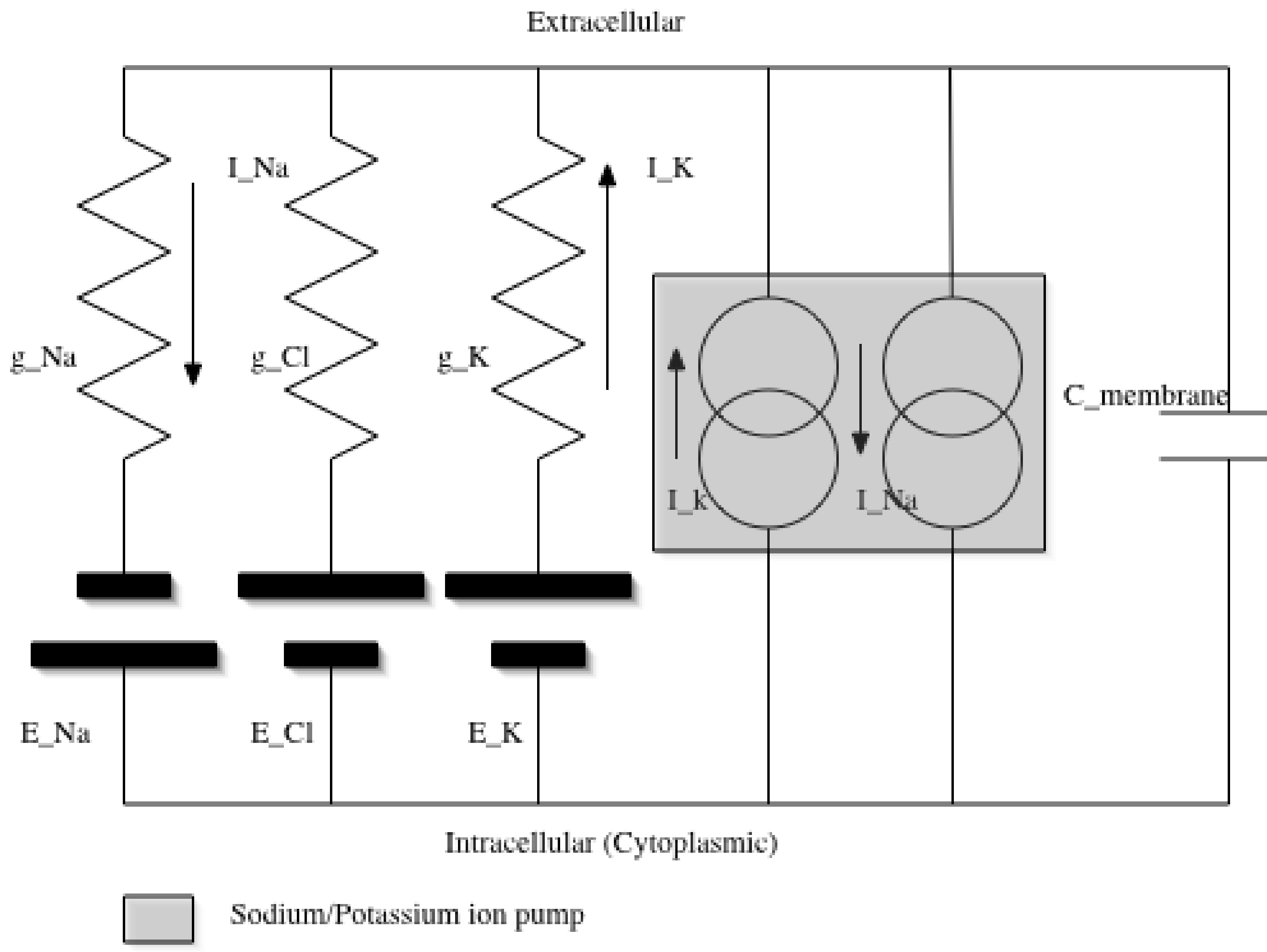
- Because there are many ion species making up the potential, these are (nearly) independent variables

- The reversal potential is the potential difference at which these balance out (giving a net 0 current).

Neuron at rest

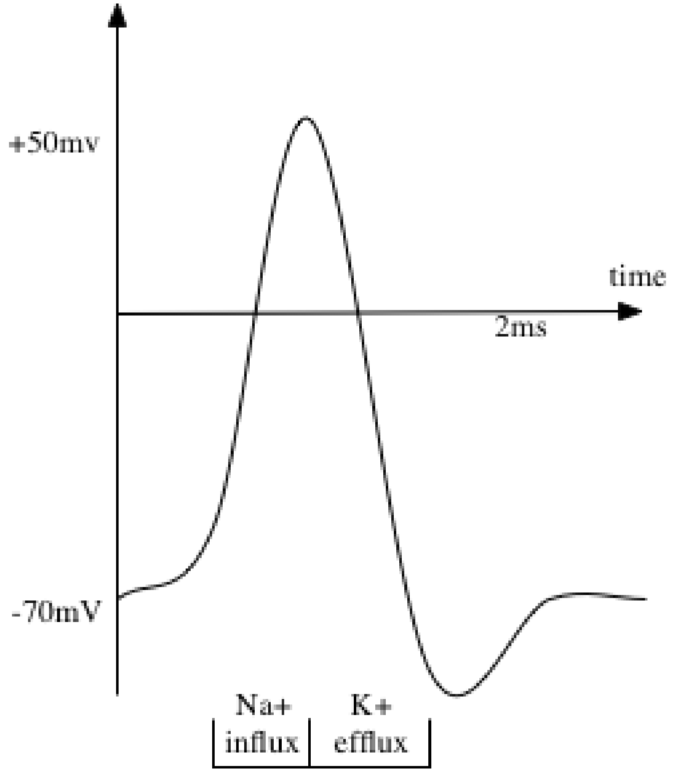
Conductance g
Current I

Na/K pump maintains neuron at about -65mV relative to extracellular fluid



Spikes (Action potentials)

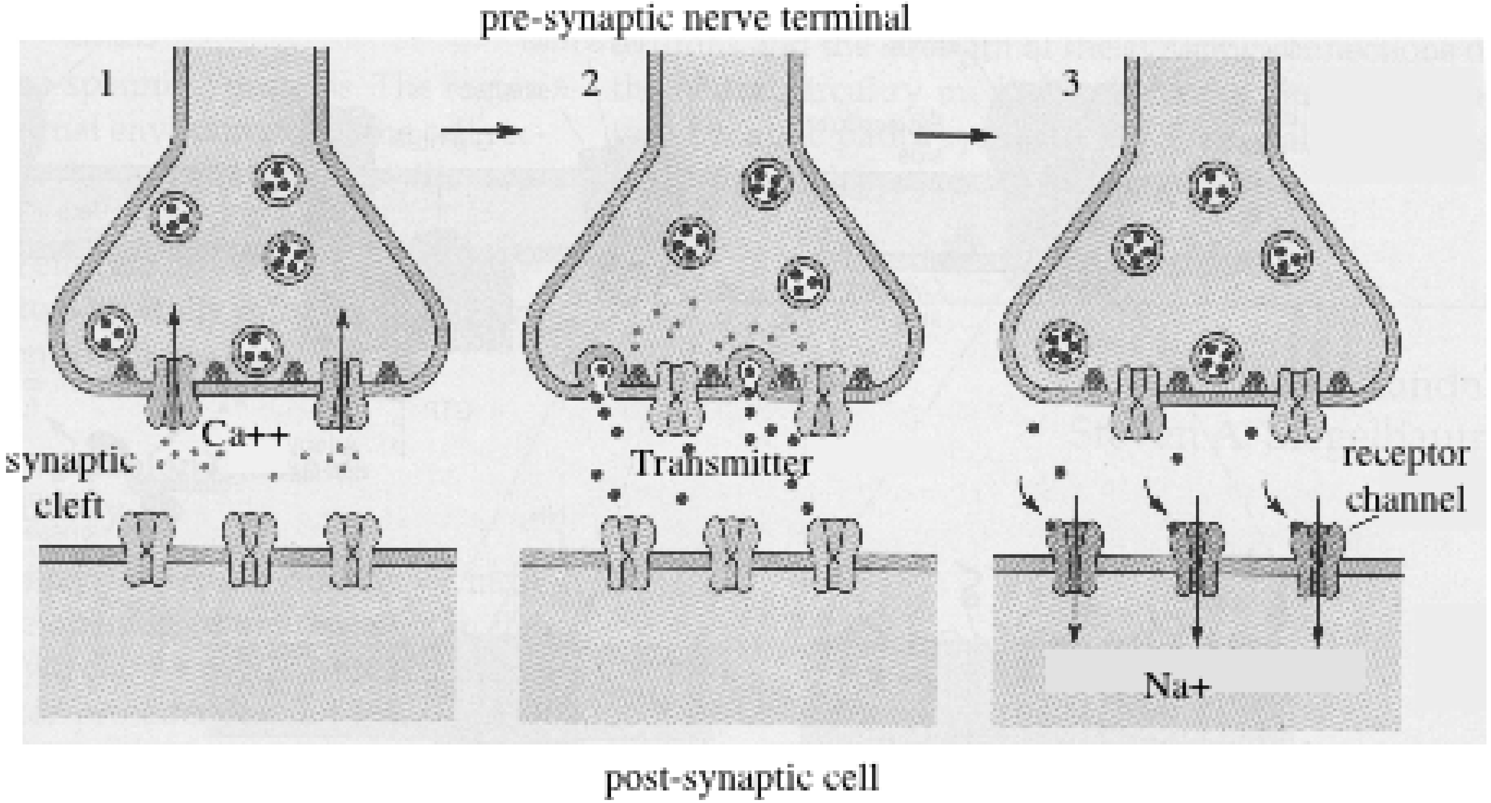
- Most real neurons communicate using action potentials
- When the potential at a particular part of the neuron (the axon hillock) exceeds some value Na+ channels open
- These inactivate, and K+ channels open
- Spikes are actively transmitted along the axon
- When they arrive at axon terminals (where axons join on to other neuron's dendrites) they affect the postsynaptic neuron's potential



Synapses

- Synapses are the connections between neurons
  - From a presynaptic neuron to a postsynaptic neuron
- Many form of synapse
  - Often presynaptic terminal releases neurotransmitter
  - Neurotransmitter in cleft...
    - Area between presynaptic neuron and postsynaptic neuron
  - ... results in opening of ion channels
  - Which then allow ions to flow
  - Altering the post-synaptic potential
  - (or injecting current into post-synaptic cell)

Example Synapse



From Kandel

Plasticity and Adaptation

- Neural systems alter in response to their inputs

- This happens over many time-scales
  - Very short: e.g. dynamic synapses
    - Need time to replenish neurotransmitter
  - Short: e.g. due to diffusable neuromodulators
  - Medium: e.g. due to morphological changes
  - Long: same as above

- Adaptation takes place in many different ways
  - Early proliferation of synapses which then die back
  - LTP and LTD at synapses
  - Changes in axon myelinisation
  - Others too

What to implement?

- Real neurons are very complex
  - And not amenable to direct implementation in silicon
  - Multiple ion species, multiple neurotransmitters and neuromodulators, plasticity are all very unlike what can be straightforwardly produced in silicon

Candidates for implementation

- Simple neuron models
  - But how simple is simple?
- Overall system models
  - Which system
  - Implemented at what level?
- Patches of membrane

Implementing simple neuron models

- The simplest models are time-free
  - They treat their input as a static vector
  - (or as a sequence of vectors)
  - An produce vector...
    - Or sequence of vector
  - ...outputs.
- Little current research on implementing such models
  - There has been in the past:
  - But:
    - Little market for them: PC speed has allowed software techniques to rule
- See references 4 and 51 for lots of details of these types of chip

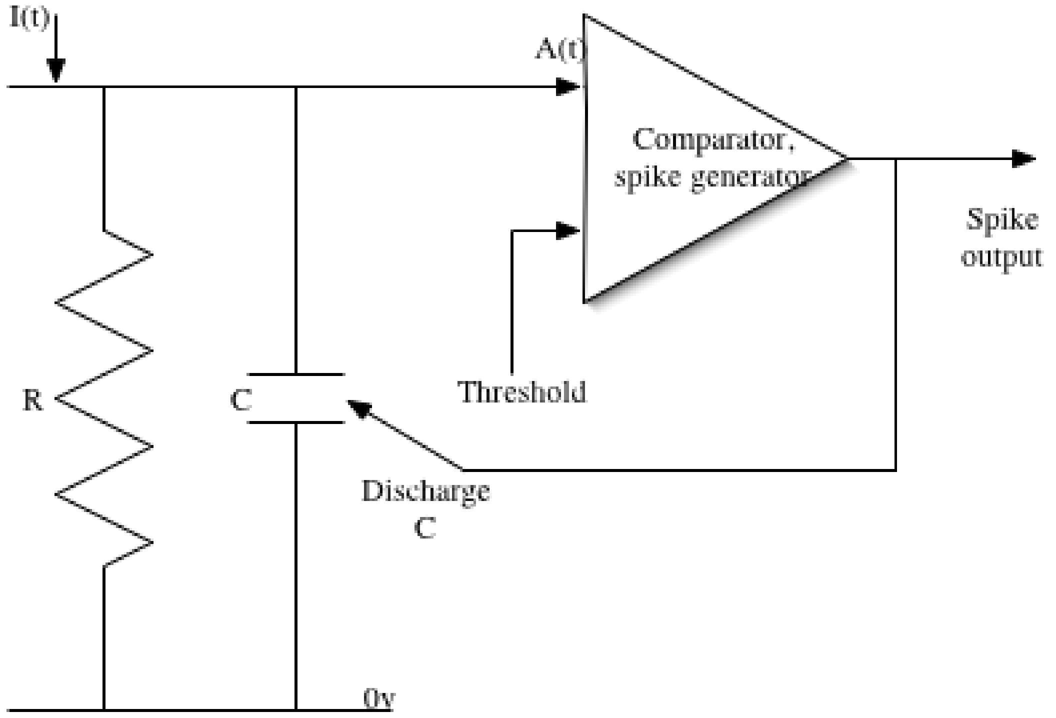
Implementing (slightly less) simple models

- One popular model which permits modelling time is the leaky integrate-and-fire model
  - Models neuron as a single point (point-neuron)
  - Models output spikes as events (shape is irrelevant)

Equivalent circuit:

Membrane -> capacitance
Ion channels -> resistor

All synapses inject current at the same point



Other neuron models

- Gerstner's spike response model
  - Models shape of post-synaptic potential
  - Models dynamic threshold
    - Allows refractory period and relative refractory period to be included

- Other non-linear leakage models may be included
  - Can allow (e.g.) realistic spike generation

- More detailed models
  - Compartmental modelling
    - Model neuron morphology as a connected set of segments of neuron
    - Often includes many different forms of leakage to model different ion channel types
    - Morphologically realistic modelling is possible
    - (Generally software)

Adaptation

- Almost all models which include adaptation change the synapse
- Many use a single-parameter characterisation of the synapse
  - The weight

- Hebbian adaptation
  - Neurons that fire together have the excitatory synapses between them strengthened

- Spike-time dependent plasticity (STDP)
  - When the presynaptic neuron fires just before the post-synaptic neuron, strengthen the synapse
  - When the presynaptic neuron fires just after the post-synaptic neuron, weaken the synapse

Modelling subsystems

- Subsystem usually has an identifiable function
  - Some aspect of vision/audition/olfaction/overall control
  - Mead’s silicon retina, Silicon cochlea, sensorimotor systems for “artificial insects”.
- Normally, individual neurons are either
  - Not modelled
  - Or modelled in a very simplistic way
  - Because the subsystem contains a very large number of neurons
- Exception: some modelling of insects and invertebrate sensory or sensorimotor systems where the neural system is mapped out precisely

Modelling patches of membrane

- Idea is is modelling membrane with ion channels
  - Normally lumped
- Aim is (usually) understanding membrane operation
  - Or aiming to develop computation based on excitable membranes

Usually based on membrane equivalent circuit

- but using a larger number of conductances
- And (often) including non-linearities in conductance operation

Implementation Techniques

- Discrete components
  - Long history: back to 1960’s (and indeed earlier):
  - models of neuron membranes and neurons (refs 42-46)
  - Models of subsystems (avian retina: (49))
- VLSI implementations
  - Many different possible technologies
    - Analog vs Digital
    - If Analog
      - Subthreshold vs supra-threshold
      - Different operating areas of the transistors in the circuit
    - If Digital
      - Custom ASIC or FPGA
- How should we compare the different technologies?

Comparison criteria

- Degree of Implementation
  - Does each entity being emulated have a corresponding piece of circuitry?
- Speed
  - Silicon implementation is faster than software, but some implementation techniques are faster than others
- Real-time system
  - Does the technique lend itself to the production of real-time implementations
- Power consumption
  - For some applications, low power is important.

Comparison of technologies

Implementation Technology	Degree of Implementation	Speed	Real-time	Power consumption
Subthreshold AVLSI	High	High	Yes	Very low
Above threshold AVLSI	High	V. High	Yes	Medium
DVLSI	Low	High	Possible	Medium-high
FPGA	Low-medium	Medium	Possible	Medium-high
Workstation	Minimal	Low	Not usually	High
DSP	Low	Med-High	Possible	High

On ASICs

- ASICs (aVLSI, sub/above threshold, DVLSI) have their own difficulties
  - Design
    - Design of DVLSI is quite difficult, and skilled.
    - Design of above threshold aVLSI is harder
    - Design of subthreshold aVLSI is harder again.
  - Fabrication
    - Unless you have industrial muscle...
      - (or a lot of money)
    - ...you wait a long time to get fabricated chips back for test
  - Testing
    - Need to design for testing
      - Make the design easy to test
    - Need to build a rig to text the system
  - And if it doesn’t work...long design cycle
  - (Though focussed ion beam systems can sometimes help)

On FPGAs

- Field-programmable gate arrays
  - Digital
  - Quickly reprogrammable
  - Faster than workstations
    - But not as fast as ASICs
  - Really somewhere between hardware and software implementations.

- Field-programmable analog arrays
  - An area of growing interest
  - May well provide a new technology for neuron implementation

- Digital Signal Processing
  - Add-on to a workstation
  - Quickly reprogrammable

Analog or Digital

- Signal coding
  - Digital: discrete values, valid at specific instants
  - Analog: continuous values in continuous time

- Most VLSI design and fabrication systems are designed for digital
  - Result is that analogue designer often is not using the best systems

- Digital systems use transistors only in on or off states
- Analogue systems rely on
  - (above threshold) Linear part of the transistor characteristic
  - (below threshold) Exponential part of transistor characteristic

Analog systems

- Issues are
  - Bandwidth
    - Maximum frequency at which system can run
  - Slew rate
    - maximum rate at which voltages can change
  - Noise level
    - thermal noise, electrical interference
  - Drift
    - slow signal change, due e.g. to temperature change
- Advantages are
  - Simple multipliers and adders
  - Simple function implementation
    - For some functions (e.g.squashing functions)
    - May not be entirely accurate
      - Often unimportant
- Disadvantages
  - Lack of accuracy, noise, drift

Digital Systems

- Issues are
  - Sampling rate
    - Maximal bandwidth is 0.5 * sampling rate
  - Representation length
    - Defines the accuracy to which values are held, and hence of operations
- Advantages
  - High speed multipliers, adders, functions
- Disadvantages
  - Multipliers, adders, functions are relatively large
  - Power consumption
  - Inputs and outputs need to be digitised

Degrees of implementation

- Analogue systems
  - Small multipliers, adders, functions are usually fully implemented
  - 1:1 correspondence with neural circuit elements
- Digital systems
  - Multipliers, functions are large
  - Insufficient room for full implementation
  - Usually functional units are shared between different parts of the neurons being modelled: partial implementation.
  - If shared between P units, then functional units must run at P*speed of signal

- Favours analogue systems

Matching

- Analogue systems often rely on similar transistors having similar characteristics
- But process variation across chip may cause this not to be the case

- Digital designers usually care little about I_{ds}/V_{gs} characteristic
  - But it is critical in analogue designs

  - Above threshold: defines gain
  - Below threshold: defines function characteristic

- Mismatch for subthreshold can be tackled by setting a bias voltage, but this means more off-chip circuitry

Memory

- Memory is required for
  - Constant values (thresholds, delays, characteristics for ion channels)
  - Alterable values (synaptic weights)
- Digital memory is well developed
  - sRAM, dRAM, EEPROM (flash)
- Analog memory can be more problematic
  - Can use digital memory + Digital/analog convertor (DAC)
  - Can sometime share DAC between different memories
- Not a new problem: motor-driven potentiometers in perceptron, or Widrow's memistor
- Floating gate technology
  - Similar technique to EEPROM, but adapted to holding analogue values
  - Use Fowler-Nordheim tunneling/hot electron injection to move up/down gradually
  - Can be difficult to implement reliably

Spiking systems

- Signals are trains of pulses
- Pulses may be modulated in many ways
  - Pulse height/width/frequency(timing) modulation
- Systems can be low power
  - Usually, power is used primarily when a pulse occurs
- Pulse-based neurons have some analogue and some digital characteristics
- Analogue:
  - Synapses, accumulation of activity level
- Digital:
  - Pulse is an event characterized purely by time of occurrence
- Real neural systems are pulse based

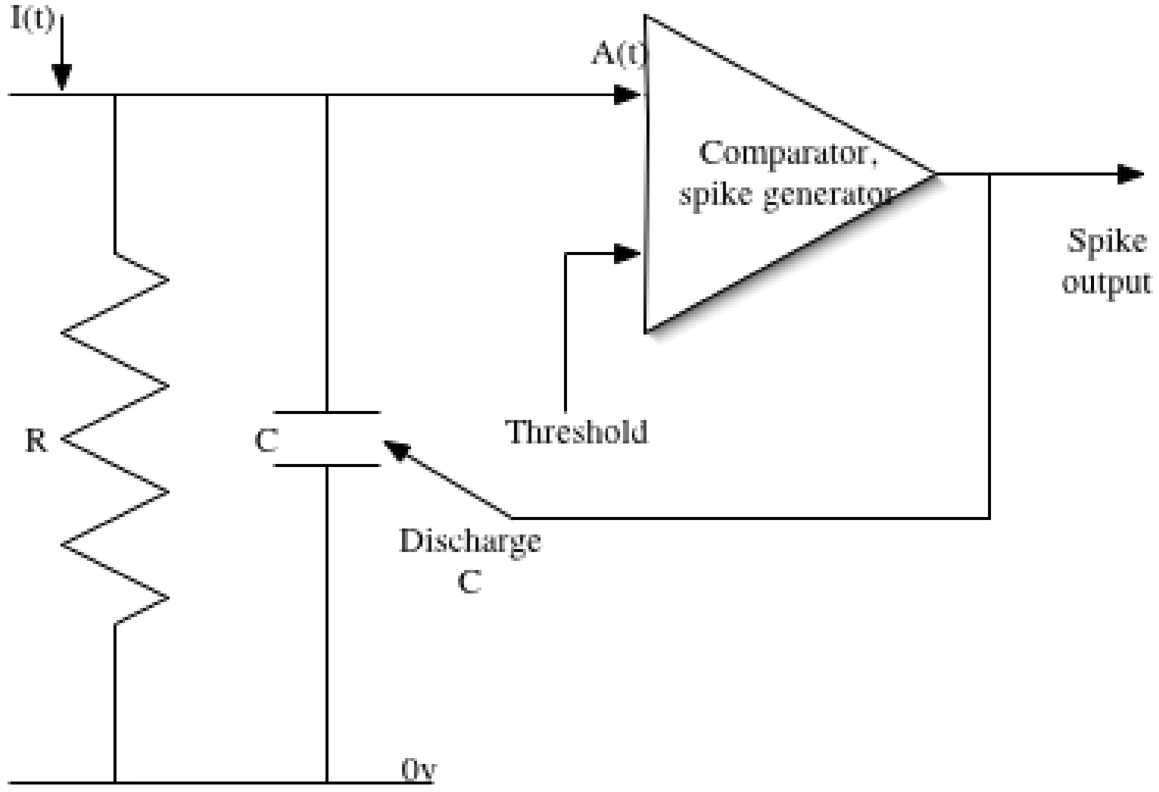
Implementing spiking systems: aVLSI

Example: LIF neuron

Unfortunately, cannot implement this circuit directly in aVLSI.

Values of C implementable lead to huge values for R.

(possible in sub-threshold aVLSI, but difficult to repeat precisely)



LIF in aVLSI

- How to solve this problem:
  - Switched capacitor techniques
    - Can permit implementation of a large resistor using switching and a small capacitor
    - Problematic: can have noise implications ion sensitive analogue circuitry due to switching transients
  - Follower-integrator circuit (in subthreshold aVLSI)
    - Can develop a delay line, and this can be useful in achieving the right order of magnitude of time constants

LIF in dVLSI

- Essentially, need to do numerical solution of the equations describing the system
- Leads to same issues as those that arise in software simulation:
  - Timestep
    - Long or short, fixed or variable
  - Accuracy
    - How many bits to use
    - Integer, or fixed point, or floating point
      - Integer and fixed point circuits are much more compact
      - But less accurate
- This can provide the appropriate time constants for the leak.
- Delays can be produced by counting cycles, and comparing result to some predefined value.

Synapses

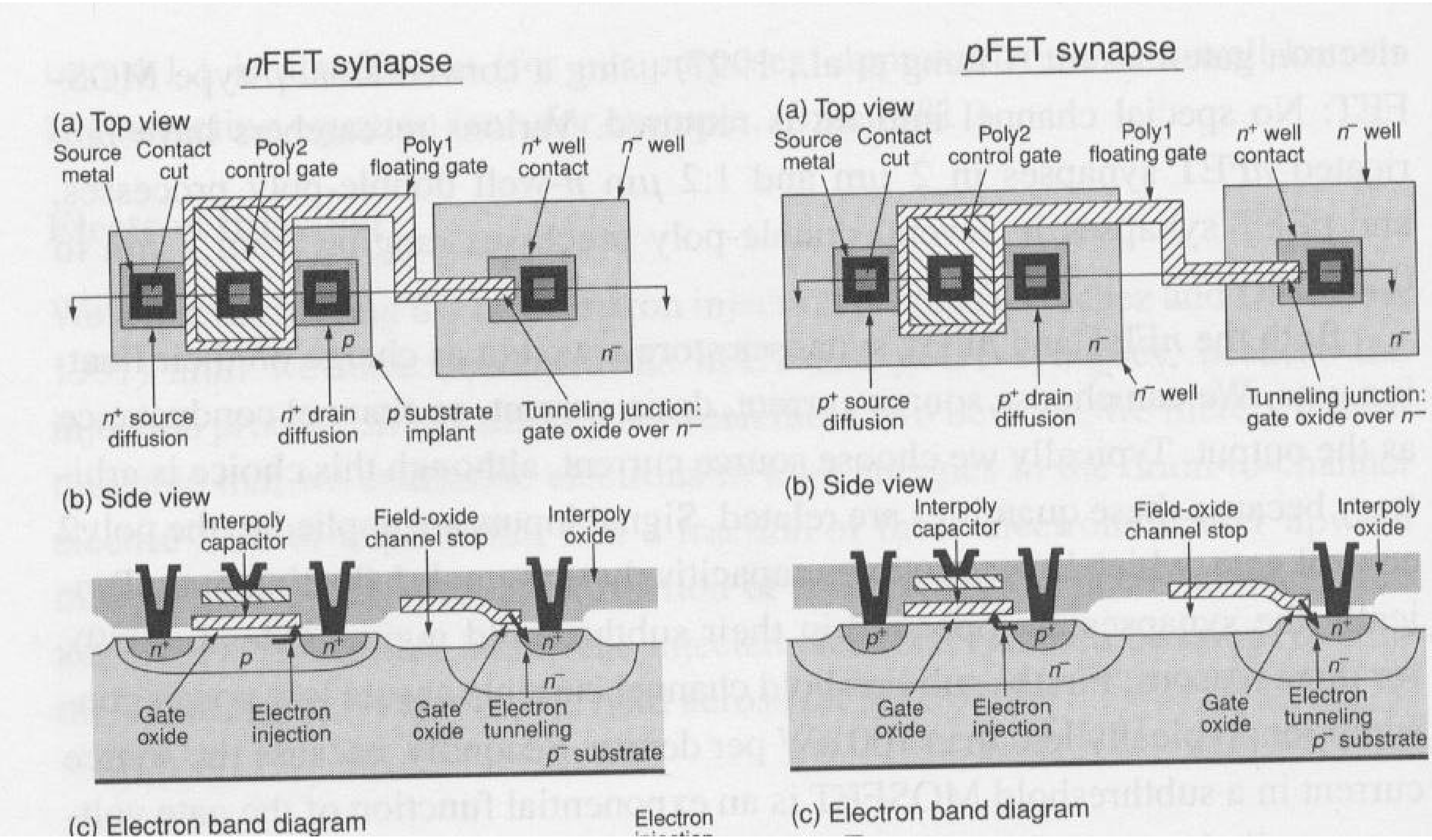
- Synapses are by far the commonest element in real neural circuitry
- So any implementation needs to be able to implement synapses efficiently
  - And be able to implement a lot of them

- Real synapses are very small, but can be quite complex
- Artificial synapses need to be small,
  - Or shared between a large number of emulated synapses

- What's required?
  - At minimum: a weight, connecting an input or presynaptic neuron to a postsynaptic neuron or output
  - Weight should be able to be altered
  - May also want some time-course of the effect that the synapse has

Synapses in aVLSI

- Part 1: Holding the value (or weight). (from Liu et al, Analog VLSI)



Alternatives

- Capacitative storage
  - Store as voltage on a capacitor
    - Leakage
  - Store as ratio of voltages between two capacitors
    - Better, but still leaks away
- Mix analogue and digital:
  - Digital storage, the DAC to generate value
- Altering the weight
  - For adaptive systems this is a requirement

Rest of the synapses (Liu et al)

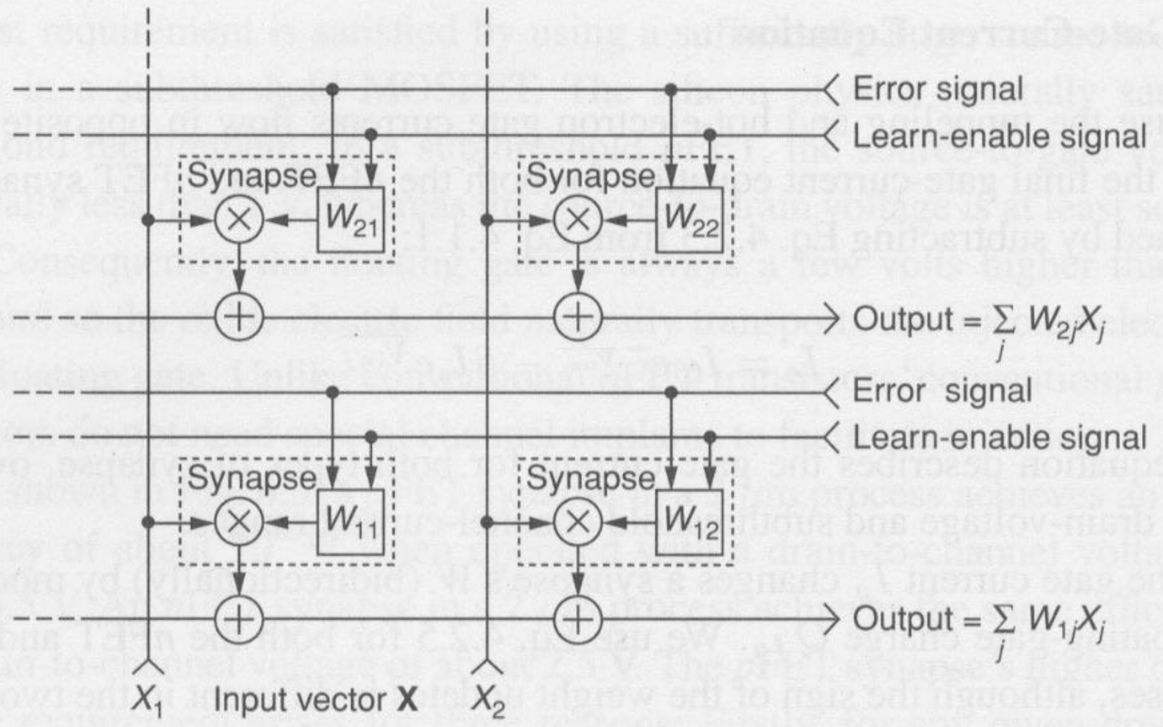


Figure 4.7

A learning-array block diagram. Each synapse multiplies its column input with its nonvolatile analog weight, and outputs a current to the row-output wire, which sums the synapse-output currents along that row. Column inputs that are coincident with the row learn-enable signals cause weight increases at selected synapses. The error signal constrains the time-averaged sum of the row-synapse weights to be a constant, bounding the row weights by forcing the synapses to compete for weight value.

Other areas

- There is interest in producing models of excitable membranes.
  - Usually lumped ionic channels (rather than emulating them individually)
  - Silicon neuron work

- There is interest in producing whole system models
  - Based on Mead, Lazzaro’s work on silicon retina, silicon cochlea
  - (Beyond the scope of this tutorial)

- There is interest in using very small feature sizes (deep-sub micron)
  - Noise becomes a serious difficulty in implementing dVLSI
  - Can one use this form of noise in a highly parallel silicon implementation?

Concluding thoughts

- Silicon implementation has great promise
  - But so far, not really lived up to its promise
- Why?
  - Workstation speed
    - Where size and power are not considerations, easier to use a workstation and software
- Yet I still think implementation in silicon will become important
  - Ubiquitous use of machines
  - Need for better sensing systems that really do work in real time
  - Limits to growth of dVLSI