Portland State UNIVERSITY

Virtualization - An Architecture Methodology For Massively Parallel Adaptive Computing

Dan Hammerstrom
Electrical And Computer Engineering

Portland State UNIVERSITY

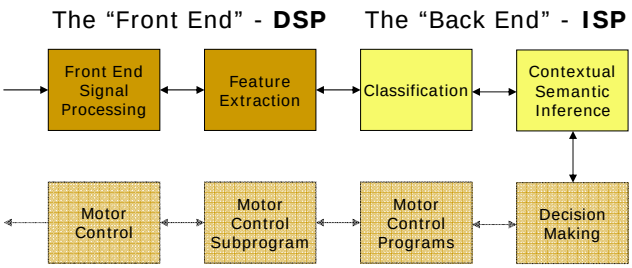
Research Objectives

- We are studying various candidate modular building blocks that will allow us to create very large, scalable structures for Intelligent Signal Processing (ISP)
  - We are looking at models that map cleanly to current and future hardware technologies
- Due to their massive parallelism, simple computations, asynchrony, fault tolerance, and relatively low precision, neural models are appealing
  - They bring certain efficiencies to the process of “inference,” a key computation in ISP
  - They also allow for a wider range of architecture options and significant implementation optimizations

Intelligent Signal Processing

- The term Intelligent Signal Processing (ISP) has been used to describe algorithms and techniques which involve the creation, efficient representation, and effective utilization of large complex models of semantic and syntactic relationships
- ISP augments and enhances existing Digital Signal Processing (DSP) by incorporating contextual and higher level knowledge of the application domain into the data transformation process

- Although an over-simplification, this flow diagram is useful in discussing the functional components of this problem domain
  - Real applications tend not to partition so cleanly
  - Notice the bidirectional arrows between each block – data flow both ways
- Most IC applications have components in each of these blocks

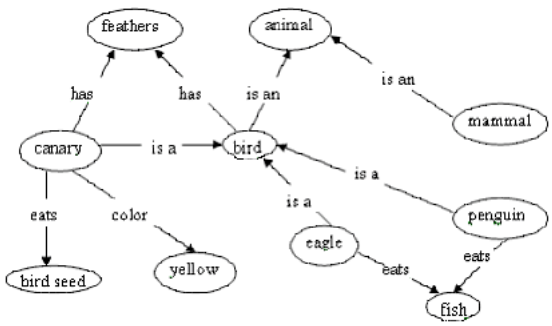


The “Front End”

- We understand the front end pretty well, it is the realm of traditional digital signal processing (DSP)
- Front end algorithms tend to involve applying the same computation over a large array of elements, they are data parallel and communication tends to be local
  - Most neuromorphic VLSI, for example, has been used primarily for this kind of front end processing
  - Another example of such an architecture is CNN (the Cellular Non-linear Network) developed by Chua, Roska et al.
    - A 2D array of programmable, “analogic” processors, with local 2D grid interconnect

But Then There’s The “Back-End” …

- What does the “Back End” do?
- Simplistically it captures information on “high order” relationships of “abstract” entities
- Which can be represented by a graph structure
- At one time these were “rules,” and “is-a” hierarchies
- But, more recently they have become probabilistic relationships -> “Bayesian” Networks
- Inference is then performed over these structures



- Back-end algorithms have fundamentally different characteristics from Front-end algorithms
- For example, they are more sparsely connected and more sparsely activated
- Also, inference requires more data communication in both up and down the hierarchy
- These can be huge networks – consider the processing of natural languages

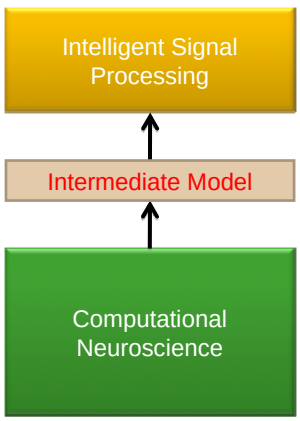
The Ultimate ISP Processor: Cerebral Cortex

- Neocortex is a folded, 2D, planar sheet
  - In humans it is, when stretched out, about 2/3s of a square meter, is 3-4 millimeters thick, and consists of roughly 30B neurons
  - It has a fairly consistent 6 layer structure and is remarkably uniform, not only across all different parts of human cortex, but across almost all mammalian species
- Although we are a long ways from understanding the details of how it does what it does, some of the basic computations are beginning to take shape

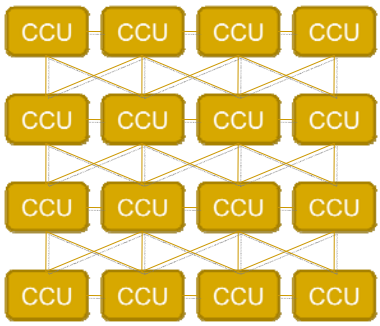
- Nature has, so it appears, produced a general purpose computational device in Cortex that is a fundamental component of higher level intelligence
- Consequently in collaboration with the neuroscience community, several groups are looking to create increasingly more sophisticated abstract models of cortex
- And then to apply these models to real applications

Computational Model

- It has been difficult to develop high level models of cognitive processing, such as ISP when starting from Computational Neuroscience
- Consequently a number of researchers have proposed various "intermediate representations"
- The best example is James Albus' Cortical Computation Unit (CCU)



- Similar modules have been adopted by a number of researchers including Lansner, Granger, Hecht-Nielsen, Anderson, and George/Hawkins (Numenta)
- CCU modules are then generally connected into a variety of "limited support" hierarchical structures
- In our own work we use something we call a "Bayesian Memory" (BM) as a CCU



- Most CCU models operate as a kind of Bayesian node that is sending hypotheses and evidence up and down a hierarchical network – using traditional Bayesian Belief Propagation techniques
  - These models conceptually connect upwards into traditional Bayesian Networks
- We have preliminary data to indicate that each CCU can, in turn, be approximately implemented by a bidirectional associative array based on spiking neural models
  - And they conceptually connect downward to Computational Neuroscience
- The use of associative models, incidentally, creates an interesting range of architectural / hardware possibilities

Architecture Space Exploration

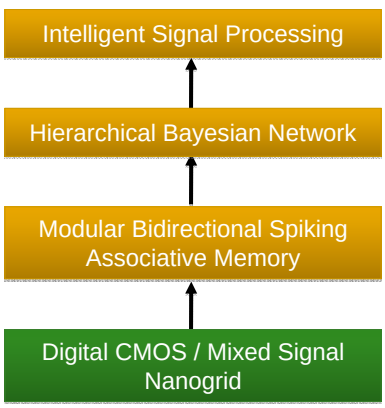
- Implementing each CCU as a modest sized associative network opens up the possibility of a wide range of hardware optimizations, including mixed signal nanogrid implementation

The questions addressed in this talk then are:

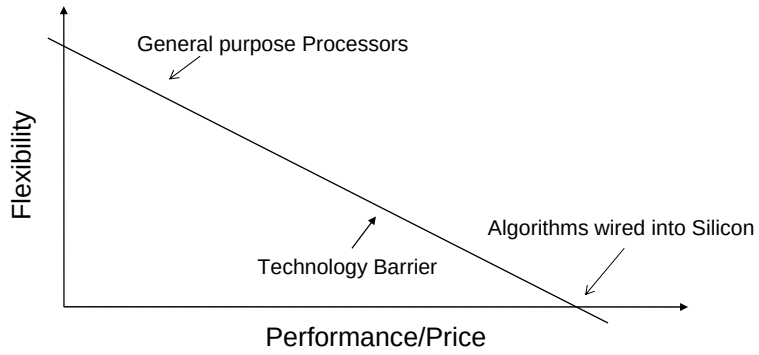
- Given current and intermediate term computing structures (which is still mostly silicon), what is the best hardware architecture?
- And what would be required to create a “human scale” model?

Architecture Space Exploration

- So the big picture is given in the graph to the right
- There is a large range of implementation options
  - Multi-core arrays
  - Neuromorphic VLSI
  - Nano-grids
  - …
- How does one go about exploring such a large space?



- To start, let's define a general flexibility / Performance-Price trade-off
- Where do we belong on this spectrum?

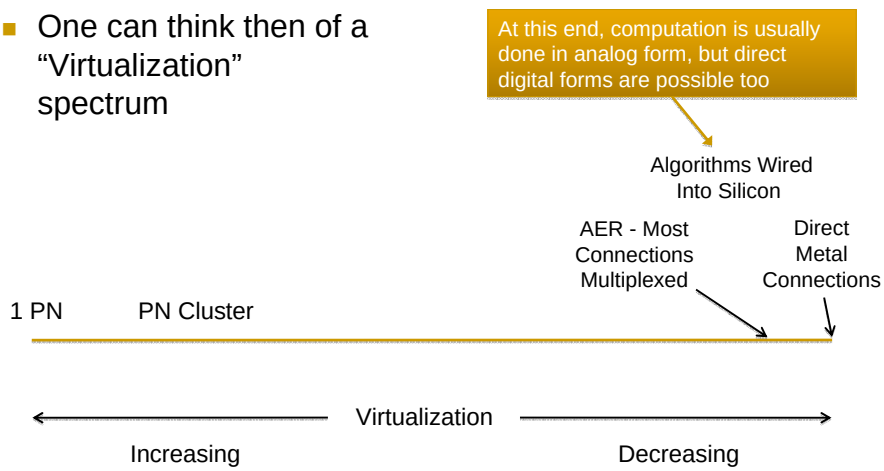


Virtualization

- We believe that **virtualization** is the most important architecture trade-off for these models
  - And it is quantifiable, so one can talk about the degree of virtualization
- To understand virtualization we introduce two graphs: the computational model is represented by the c-graph, which is emulated on a hardware structure called the p-graph
  - These graphs, in essence, show the structure of the "data flow" and the computations on the data both in the model and in the hardware
  - The nodes in the *p-graph* represent the physical computational entities and the arcs represent physical communication paths
  - In a computer cluster the *p-graph* is the physical network of processors and their interconnect
  - The *c-graph* can be thought of as a "virtual" network being emulated by a physical network (the *p-graph*)

- The **degree of virtualization** then is roughly the "amount" of multiplexing of physical resources by the computation and communication tasks
  - Multiplexing implies several virtual (*c-graph*) nodes sharing a single physical compute element (*p-graph*), and/or multiple *c-graph* arcs mapped into a single *p-graph* arc
  - A single *p-graph* node may emulate many *c-graph* nodes
  - And each *p-graph* connection may contain many "virtual" *c-graph* connections
  - At one end of the spectrum is a single fast processor (a *p-graph* of one node) emulating the entire *c-graph* (maximum virtualization)
  - At the other end of the virtualization spectrum, the *c-graph* is implemented intact on an identical *p-graph* (the algorithm is wired directly into hardware, there is no virtualization)

- One can think then of a "Virtualization" spectrum



- Neural models are massively parallel so we have a large range of virtualization available to us, much more so than with traditional computational models
- However, since it promises the highest performance, why not just put ourselves at the direct implementation (little or no virtualization) end of the spectrum?
- And this is, in fact, where most neuromorphic hardware sits
- But, does that always give the best performance/price for any model?
- Not necessarily!
- Factors such as differential hardware costs and the dynamic behavior of the models have a big impact on the “sweet spot” of virtualization

Portland State UNIVERSITY

An Example: Virtual Connections

- Even with connectivity patterns that are much sparser than cerebral cortex, the use of dedicated metal lines for each connection quickly overwhelms the available connection resources in a typical silicon process
- Consequently the neuromorphic aVLSI community, which basically wires individual neurons into silicon multiplexes many connections over most metal lines, using the Address Event Representation (AER)
  - “Programmable Connections in Neuromorphic Grids,” Joseph Lin, Paul Merolla, John Arthur, and Kwabena Boahen, *49th IEEE Midwest Symposium on Circuits and Systems*, 2006

- This is necessary, since most neural systems have huge connectivity requirements, you can quickly run out of metal!
- Assume a rectangular array of silicon neurons where
  - each neuron receives input from its *N* nearest neighbors
  - each such connection consists of a single metal line
  - the number of metal layers is much less than *N*
  - The area required by the metal interconnect is of $O(N^3)$
    - Literally several square meters for billions of connections
- And this is especially wasteful when you consider that neuron connection bandwidth is a few bits per second and a metal line is 8-9 orders of magnitude faster

- But if we multiplex connections, then don't we need to store and respond to large numbers of connection addresses?
- Interconnect (metal) vs. RAM for addresses, address RAM vs. interconnect, it is a trade-off that cannot be avoided!
- Imagine a 1M node network and assume the weight matrix is 50% full, which is 500×10^9 connections
  - If they are 1-bit connections, then we have 8 per byte, or 62.5 Gbytes
  - But if they are 32 bit addresses, that is 2 Terabytes
- However, is connectivity all that we can multiplex? Actually, no

- So, given a certain set of hardware capabilities, and a model that has a certain structure and certain computations associated with it, AND, which has certain dynamic characteristics,
- Can we find the “sweet spot” hardware architecture for this model, given some implementation substrate (in our case CMOS and CMOS/nanogrids)?
  - “Optimal” is a tricky term and we do not claim optimality and prefer to use the term “sweet spot”
- And what does this analysis teach us about implementing such models in silicon, and will it result in a useful architectural “methodology”?

Caveats!

1. Our computational model is not special, we assumed that it was reasonably cortical-like, but not everybody will agree with our assumptions
2. Our hardware assumptions are not special, we made reasonable assumptions about currently existing, or relatively near-term projected CMOS and the existence of a implementable CMOL-like nano-grid structure
3. The specific architectural results only hold for the specific model and specific hardware assumptions that were made – different assumptions will change the results and the location of the sweet spot in the Virtualization spectrum
4. The goal of this work is an architectural exploration methodology that, hopefully, will be useful across a broader selection of models and hardware
   - As far as we know, there has been no study of hardware implementation trade-offs of biologically inspired models
5. System operates in real time: Mead, “Time is its own representation”

Virtualization

Multiple Aspects

- Definition:
  - The **degree** of time-multiplexing of neural components / algorithmic components via hardware resources
- A tool / concept for hardware designers
- Study / Investigate / Estimate: Trade-Offs
  - Performance – Time
  - Price – Area and Power
  - Efficiency – Utilization, Silicon Real-Estate Usage
  - Scalability – How large? How Many?
  - Single chip / Multi-chip

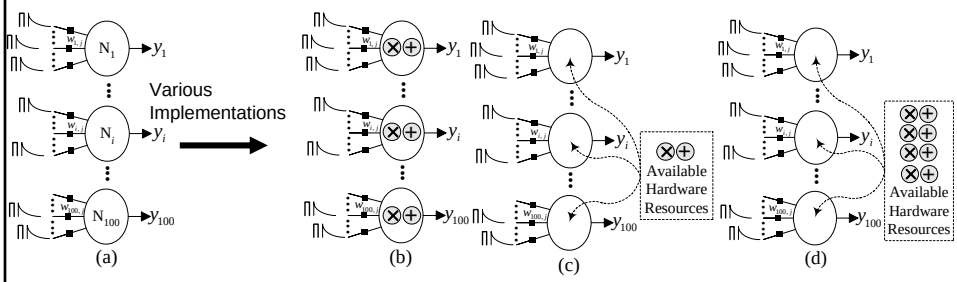
Most of this work was done by my student Mazad Zaveri

Portland State UNIVERSITY

Is Virtualization A Useful Concept?

- For existing technology, compare various designs:
  - PC vs. FPGA vs. Digital vs. Mixed-Signal vs. Analog vs. …
  - Find the “missing” designs for an application
  - Explore new candidate hardware
  - Study new implementations and trade-offs
- For future technology:
  - Ballpark estimates of performance/price and other trade-offs
  - Compare CMOS to emerging technologies (Nano, CMOL, etc.)
  - Transition from CMOS to an emerging technology - what maps efficiently and what does not?
  - Study hypothetical CMOL implementations of new applications, to efficiently leverage emerging technology

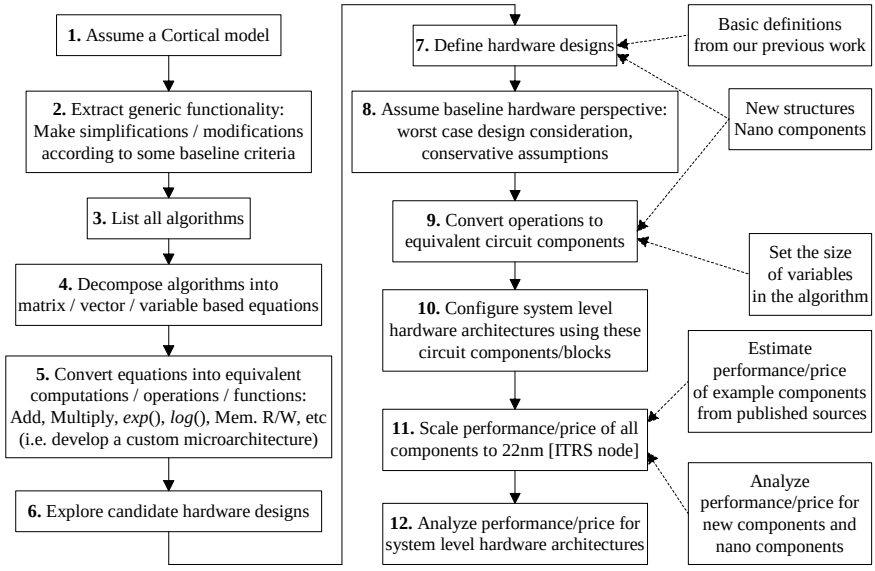
Virtualization: Simplified Example



	Fully Parallel Minimum Virtualization	Fully Sequential Max. Virtualization	Semi-Parallel Intermediate degree
Area requirements	x	x/100	x/25
Timing	x	x*100	X*25
Utilization Efficiency			
Input Activity — High Act.	HH	HH	HH
Input Activity — Low Act.	L	HH	H

How do we use these ideas?

- An architecture assessment methodology for exploring the hardware design space

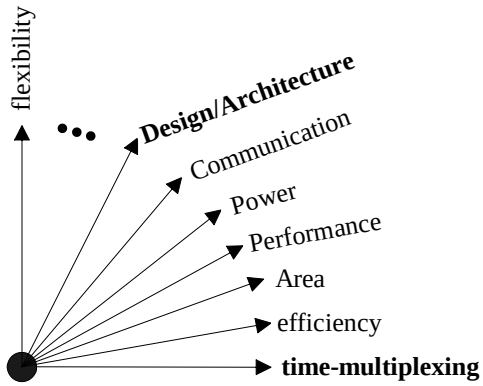


Aspects / Dimensions of Virtualization

Various dimensions of virtualization

- Time-multiplexing:
  - Sequential to Parallel
- Design / Architecture type
  - Data precision and type
  - FXP, Floating-pt, …
  - Analog vs. digital vs. mixed signal
- Flexibility:
- Performance
- Price – Area & Power
- Purpose:
  - General or application-specific
- Implementation:
  - Fine grained to coarse grained

- As the degree of virtualization changes:
  - One traverses a multi-dimensional space
  - An implementation is a single point in this space

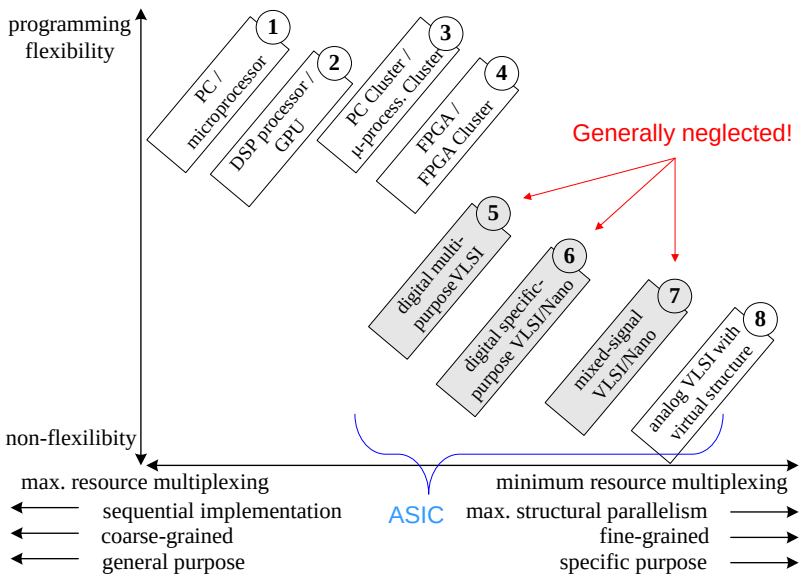


Axes that designer can control:

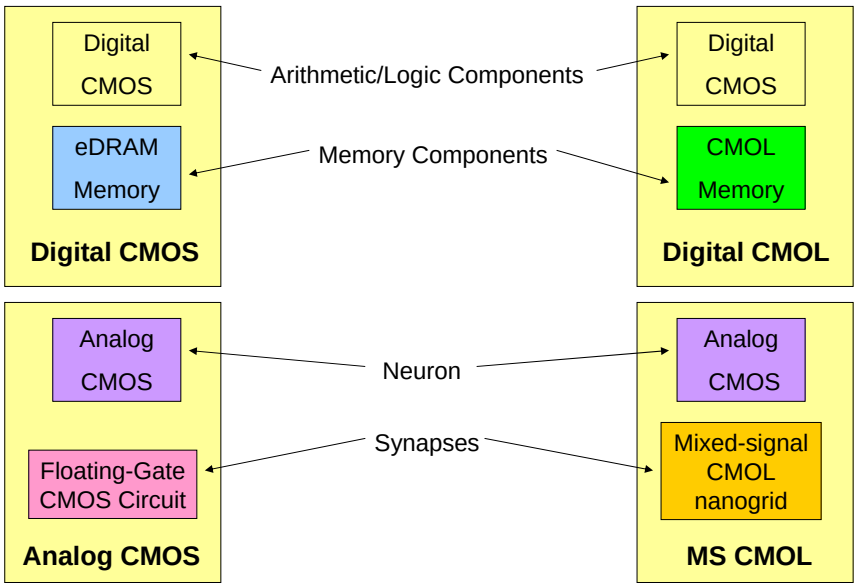
Design/Architecture &

Number of processors (time-multiplexing)

Hardware Virtualization Spectrum (More Detail)



Assumptions: Design Configurations



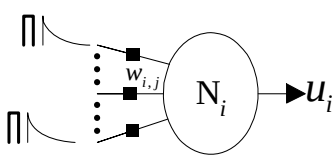
Portland State UNIVERSITY

Assumptions: The Model

- Human cortex
  - Neurons = 2×10^{10}
  - Synapses = 1.6×10^{14}
  - Synapses per neuron = 8000
- Area of cortex = 2.4×10^{5} mm^2
  - Square shape (Each side = ~490 mm)
- Neuron model:
  - Spike Response Model *[Gerstner]*

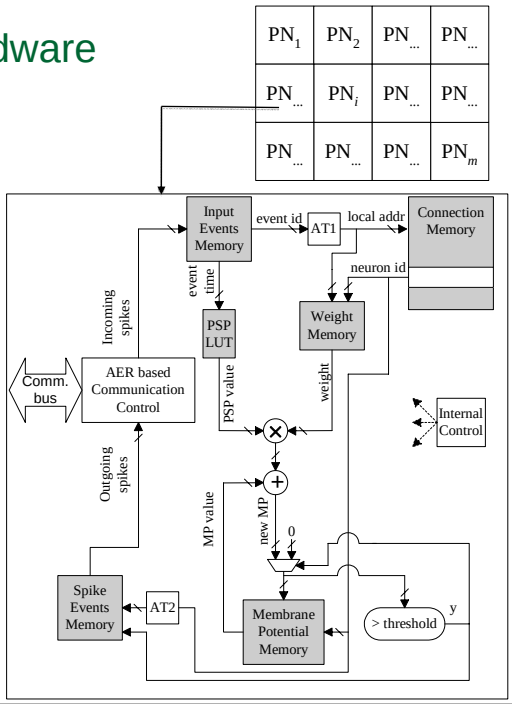
$$u_i(t) = \sum_{j=1}^{N} w_{ij}\varepsilon_{ij}(t - t_j) + \eta_i(t - t_i)$$

$$\varepsilon_{ij}(t) = [\exp(-\frac{t-\tau_a}{\tau_m}) - \exp(-\frac{t-\tau_a}{\tau_s})]H(t-\tau_a)$$



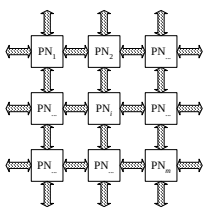
Assumptions: The Hardware

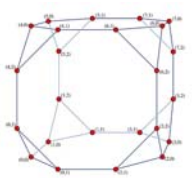
- Cortex Emulation - large-scale spiking network
  - *m* Processing Nodes (PN)
  - Each PN implements
    - $2x10^{10}$ / *m* neurons
    - $1.6x10^{14}$ / *m* synapses
    - $2.4x10^{5}$ / *m area of cortex*
    - Inter-neuron Connectivity mapped by Connection Memory
  - PN operation
    - Digital operation *[Gao07]*
    - Event-driven processing
    - Event (model-time) resolution (1ms)
- System uses asynchronous logic (no clock) and is generally assumed to operate in real-time



Assumptions: p-graph

- Communication (for *m* PNs)
  - based on estimated total connections for *m* PNs
  - Spikes transmitted as AER [Gao07]
- Communication structures *[Wittie81]*
  - Nearest Neighbor Mesh (NNM)
    - Average-message-delay = $ceil(m^{0.5})/2 \times t_{msg}$
  - Cube Connected Cycles (CCC)
    - Average-message-delay = $(7D/4) \times t_{msg}$; where $(D \times 2^D) >= m$
- Factors considered in estimating connections
  - Percentage of global (inter-cortical or corticocortical) connections (= 2/3) *[Lansner07]*
  - Axonal sharing/terminal arborization (=1/80) *[Modha07]*
  - Message merging *[Modha07]*
    - Concurrent events in sender PN going to same receiver PN (=1/100) *[Lansner07]*
  - Percentage of activity (at neuron outputs → synapses) (=0.01) *[Lennie03]*
- Total-messages = (1/80)×(1/100)×(0.01)×total-connections
- Total-delay = Total-messages × Average-message-delay





Assumptions: Technology

- CMOS, digital 22nm (from ITRS), analog 90nm
- Nanogrid structures fabricated on top of CMOS
  - Feature size of 4.5nm (9nm inter-metal pitch)
  - Currently researchers are making wires out of silicon and other materials that are ~15 nm in diameter, eventually going to < 10 nm, with lengths up to 10 µm
  - A CMOL (Cmos / MOLecular) circuit combines a CMOS stack, and two layers of parallel nanowires, manufactured by **Nanoimprint Lithography**
  - In CMOL, the interfacing is provided by pins that are distributed over the circuit area, which are aligned in a skewed manner with the underlying CMOS circuitry

CMOL*

- A CMOL (Cmos / MOLecular) circuit combines a CMOS stack, and two layers of parallel nanowires, manufactured by **Nanoimprint Lithography**
- In CMOL, the interfacing is provided by pins that are distributed over the circuit area, which are aligned in a skewed manner with the underlying CMOS circuitry
- This arrangement allows individual access to each nanowire, even if the half-pitch (F_{nano}) of the nanowire crossbar is much less than that (F_{CMOS}) of semiconductor metal width and pitch
- This trick allows access to each nanodevice via a pair of CMOS cells located at the Manhattan distance $|\Delta x| + |\Delta y| < 1/\sin\alpha + 1$
- CMOS then provides current drive, I/O, and signal restoration

* Developed by K. Likharev at SUNY Stony Brook

Assumptions: The Model

- All electro-chemical activities in a spiking neuron cost energy
  - Major energy-consuming activities *[Lennie03]*
    - Postsynaptic Potentials (PSP)
    - Firing/Resting potential
    - Propagation of spikes along axons
- Based on metabolic constraints of adult human (100W total, 20W brain)
  - Only 1% of all neurons can be "substantially active simultaneously" *[Lennie03]*
- Our interpretation
  - Only 1% of all neurons can emit a spike at any instant
  - Only 1% of all synapses have active PSPs at any instant
    - Worst case: PSPs at all synapses start at the same time

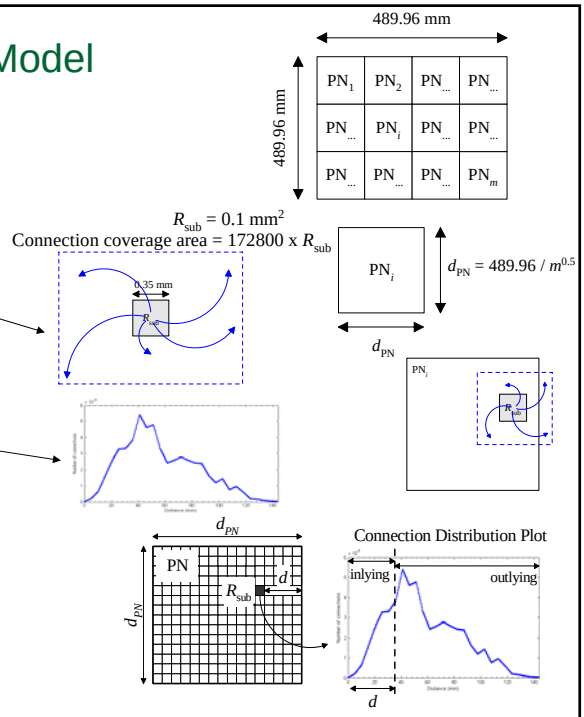
Portland State UNIVERSITY

Assumptions: The Model

- Distributing global (cortex) activity to PNs (load allocation)
  - Global Activity (i.e. in cortex is 1%)
    - Global: No. of active synapses ($S_c = 0.01 \times 1.6 \times 10^{14}$)
    - Global: No. of neurons emitting spike ($N_c = 0.01 \times 2 \times 10^{10}$)
  - PN Activity (there are *m* PNs)
    - Number of active synapses per PN_i is ($S_{PN\text{-}i}$)
      - $S_{PN\text{-}i}$ is sampled randomly from a uniform distribution of range (0 to $2S_c/m$), such that global activity is constant
        - $(S_{PN\text{-}1} + S_{PN\text{-}i} + S_{PN\text{-}m}) = S_C$
    - Number of neurons emitting spikes per PN_i is ($N_{PN\text{-}i}$)
      - $N_{PN\text{-}i}$ is sampled randomly from a uniform distribution of range (0 to $2N_c/m$), such that global activity is constant
        - $(N_{PN\text{-}1} + N_{PN\text{-}i} + N_{PN\text{-}m}) = N_C$

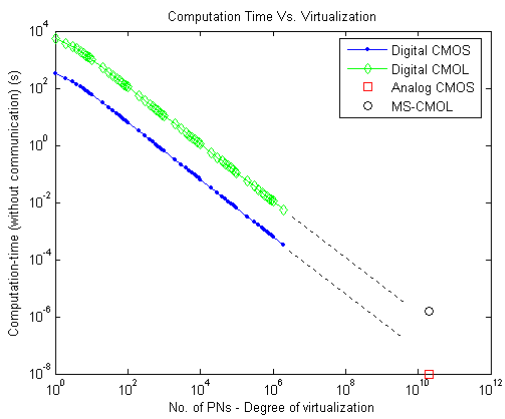
Assumptions: The Model

- Derive inter-PN connections
  - Each PN divided onto sub-regions R_{sub} (0.1 mm^2)
  - For each sub-region the connection coverage area is known *[Schüz06]*
  - The distribution of corticocortical connections Vs. distances is known *[Braitenberg98]*
  - Estimate total connections in each PN
    - Aggregate outlying connections from all sub-regions



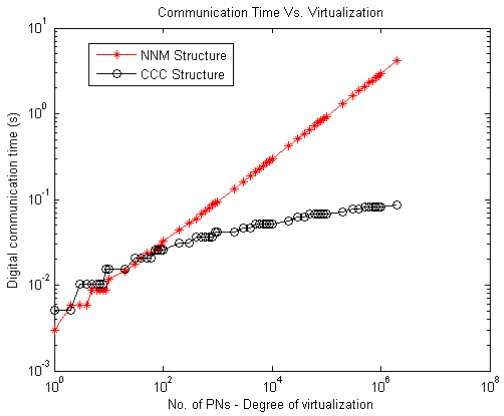
Results: Computation Time

- Digital CMOS/CMOL: Computation-time decreases linearly with increasing number of Processing Nodes (PNs)
  - Digital CMOL slower than digital CMOS by 15x
- Simple Extrapolation to 2x10^{10} PNs
  - Comparable timings for Digital CMOS and Analog CMOS
    - Assuming analog design is 10^5 times faster than model-time *[Schemmel04]*



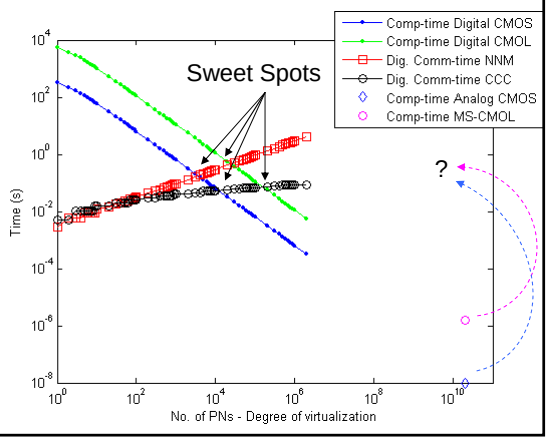
Results: Communication Time

- Total messages that needed to be transferred
  - 0.14×10^{12} msgs/sec (for time-step simulation approach: 13.5×10^{7} msgs per 1 ms)
- Communication time increases with increasing number of PNs
  - Increases linearly - NNM structure
  - Increases sub-linearly - CCC structure

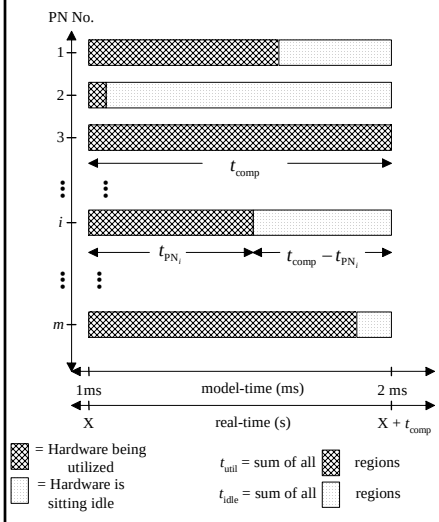


Results: Time (Computation + Communication)

- Digital CMOS/CMOL
  - Computation-time decreases linearly
  - But communication-time increases (NNM & CCC structure)
  - Communication-time dominates after several hundred/thousand PNs.
    - Sweet Spots
- Analog CMOS and MS-CMOL
  - Communication is generally digital – AER based
  - Final speed may be limited by
    - digital communication
    - number of partitions (dies/chips)
  - So what are the trade-offs?
    - How many partitions?

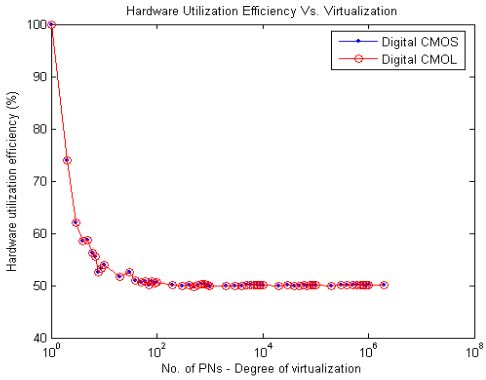


Results: Utilization



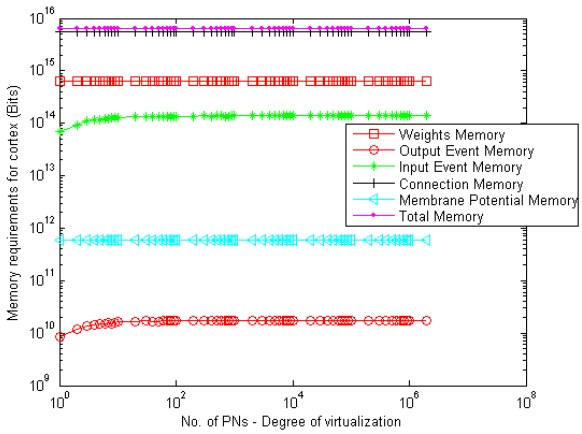
$$HUE = \frac{t_{util}}{t_{util} + t_{idle}}$$

- Hardware Utilization Efficiency (HUE) decreases to 50% after about 1000 PNs.
  - This means that 50% of the time the compute hardware is sitting idle



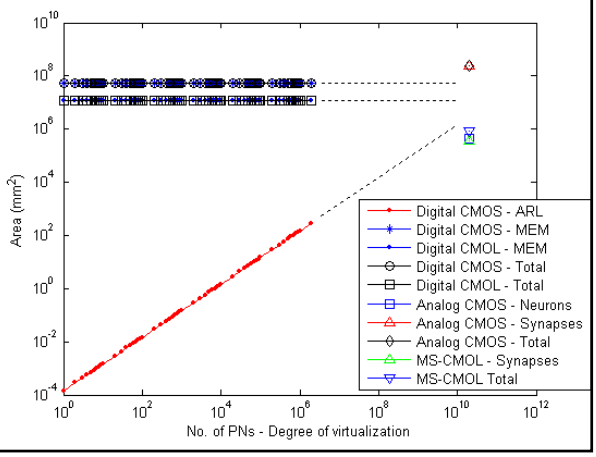
Results: Memory Requirement

- The weight memory and connection memory dominate the total memory requirement for Digital CMOS/CMOL
  - Both directly proportional to the number of synapses in cortex (1.6×10^{14})
    - Each weight is assumed 4 bits
    - Each global connection/address assumed to be 35 bits = ceil($\log_2(2 \times 10^{10})$) – no variable length addressing assumed



Results: Area

- Digital CMOS/CMOL
  - Total area dominated by memory (eDRAM or CMOL-MEM)
  - Extrapolation to 2×10^{10}: Arithmetic/logic (ARL) area for digital designs comparable to Analog CMOS (neuron area)
- Analog CMOS (fully parallel)
  - Area dominated by synapses
    - Assume one I&F CMOS circuit per neuron *[Indiveri06]*
    - Assume one floating-gate per synapse *[Hasler97]*
- MS-CMOL
  - Area of Neurons and synapses is equal
  - Assume one I&F CMOS circuit per neuron
  - Each synapse is nano-memristor *[Wei Lu09]*



Summary

Design	Comm. Struct.	No. of PNs @ Sweet Spots	Single PN Area (mm^2)	Complete Cortex (all PNs) Area MEM (mm^2)	Area ARL (mm^2)	Comp. Time (sec)	Comm. Time (sec)	Final Area with Comm. (all Silicon) (mm^2)
Digital CMOS	NNM	4000	$2\xi10^4$	$4.2\xi10^7$	0.5	0.2	0.2	~ $4.2\xi10^7$
	CCC	20000	$3\xi10^3$	$4.2\xi10^7$	3	0.05	0.05	$6.12\xi10^9$
Digital CMOL	NNM	30000	$4\xi10^2$	$1.2\xi10^7$	5	0.5	0.5	~ $1.2\xi10^7$
	CCC	200000	60	$1.2\xi10^7$	30	0.07	0.07	$7.74\xi10^9$

		No. of Partitions	Area (mm^2)	Area Synapse (mm^2)	Area Neuron (mm^2)	Comp. Time (sec)	Comm. Time (sec)	Final Area with Comm. (all Silicon) (mm^2)
Analog CMOS	NNM	20000	$12.5\xi10^3$	$2.5\xi10^8$	$4.5\xi10^5$	$1\xi10^{-8}$	0.5	~2.510^8
	CCC	20000	$12.5\xi10^3$	$2.5\xi10^8$	$4.5\xi10^5$	$1\xi10^{-8}$	0.05	2.5510^{10}
Mix-Sig CMOL	NNM	20000	40	$3.5\xi10^5$	$4.5\xi10^5$	$2\xi10^{-6}$	0.5	~810^5
	CCC	20000	40	$3.5\xi10^5$	$4.5\xi10^5$	$2\xi10^{-6}$	0.05	$8.12\xi10^7$

Human Sized Cortex Approximation

Configuration	Time (sec) Total	Area (mm2) Total	Perf/Price	Normalized Perf/Price	Number of 300mm wafers
Digital CMOS - NNM	4.00E-01	4.20E+07	5.95E-08	7.5	660
Digital CMOL - NNM	1.00E+00	1.20E+07	8.33E-08	10.4	189
Analog CMOS - NNM	5.00E-01	2.50E+08	7.99E-09	1.0	3937
Mixed Signal CMOL - NNM	5.00E-01	8.00E+05	2.50E-06	313.1	13

- Assumes $2x10^{10}$ neurons and $1.6x10^{14}$ synapses
- Remember our simplified model, we're not claiming functional equivalence!
- The mixed signal CMOL is the most promising
- Execution time is simulating 1ms of model time in 50ms

Portland State UNIVERSITY

Conclusions

- **Digital CMOS and Digital CMOL**
  - Memory dominates total area
  - Multiplexing of Arithmetic/logic components past a certain point has diminishing returns
  - Cube-Connected-Cycles communication
    - Full silicon (passive network) implementation infeasible
    - Switching (active network) implementations needed
  - Nearest-Neighbor-Mesh communication
    - Full silicon (passive network) implementation feasible

Conclusions

- **Analog CMOS**
  - Cortex-Scale implementation needs to be partitioned to chips/wafers
    - If partitioned
      - Speed advantage of Analog CMOS is lost
      - Final time dominated by inter-communication between partitions
  - Synapse storage (Floating-gate) dominates the total area
  - Digital CMOS and Digital CMOL implementations are denser
- **Mixed-Signal CMOL**
  - Cortex-Scale implementation needs to be partitioned to chips/wafers
    - If partitioned
      - Speed advantage of Mixed-signal CMOL is lost
      - Final time dominated by inter-communication between partitions
  - Area for (analog CMOS) Neurons and (nano-memristor) Synapses is comparable
  - MS CMOL (nano wire & memristor grid) based synapses will be 700x denser than analog floating-gate synapses
  - MS CMOL design will be 35x denser than Digital CMOL design

- Assumptions concerning sparse activation, computation, and global (inter-PN) communication play an important role in results
- Digital computation requires a number of simplifying assumptions such as piece-wise linear approximations of the synaptic impulse response
- This adds “approximation” noise to the model computation – what is the effect of that noise vs. other kinds of noise in such a large model?

- As the computational models mature and large scale simulations are developed, we will continue to fine-tune our analyses
- As well as study issues related to dynamic learning and model "approximation" noise
- We are not aware of any other such an analysis, which explores most of the known hardware space, has been done for these kinds of models
- More analyses like this are needed to guide research in models, hardware components, and architecture

email: strom@ece.pdx.edu
my page: http://www.ece.pdx.edu/~strom