

---

# Intelligent Signal Processing

---

Dan Hammerstrom  
Electrical And Computer Engineering

# Intelligent Computing

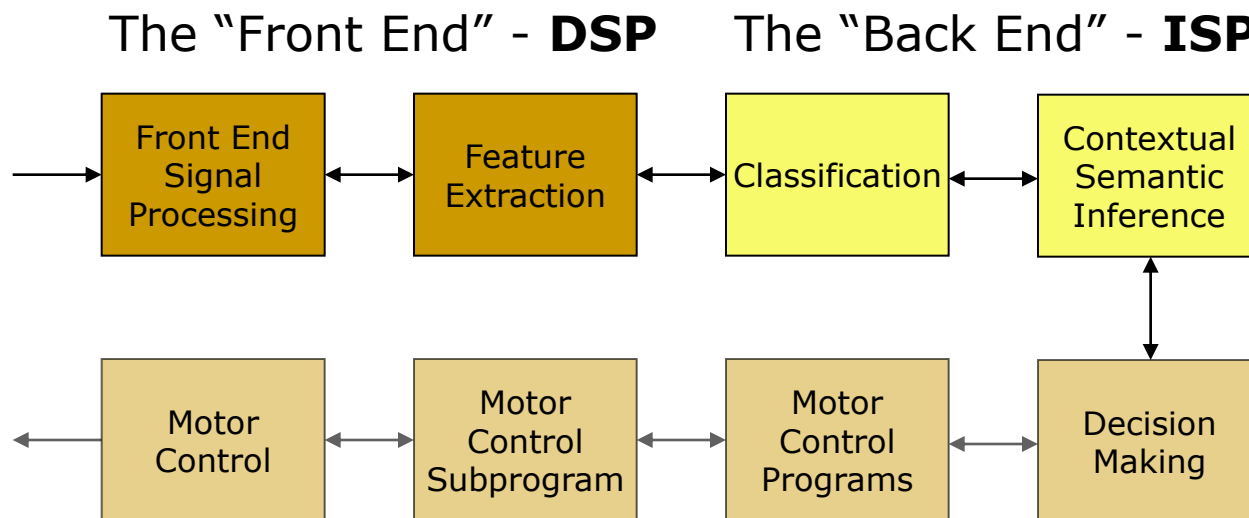
- In spite of the transistor bounty of Moore's law, there is a large class of problems that computers still do not solve well
- These problems involve the transformation of data across the boundary between the real world and the digital world
- They occur wherever a computer is sampling and acting on real world data, which includes almost all embedded computing applications
- Our lack of general solutions to these problems, outside of specialized niches, constitutes a significant barrier to computer usage and to huge markets

# Intelligent Signal Processing

- The term Intelligent Signal Processing (ISP) has been used to describe algorithms and techniques that 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

# A Prototypical Model of Intelligent Computing

- Although an over-simplification, this flow diagram characterizes most intelligent computing implementations



## The “Front End”

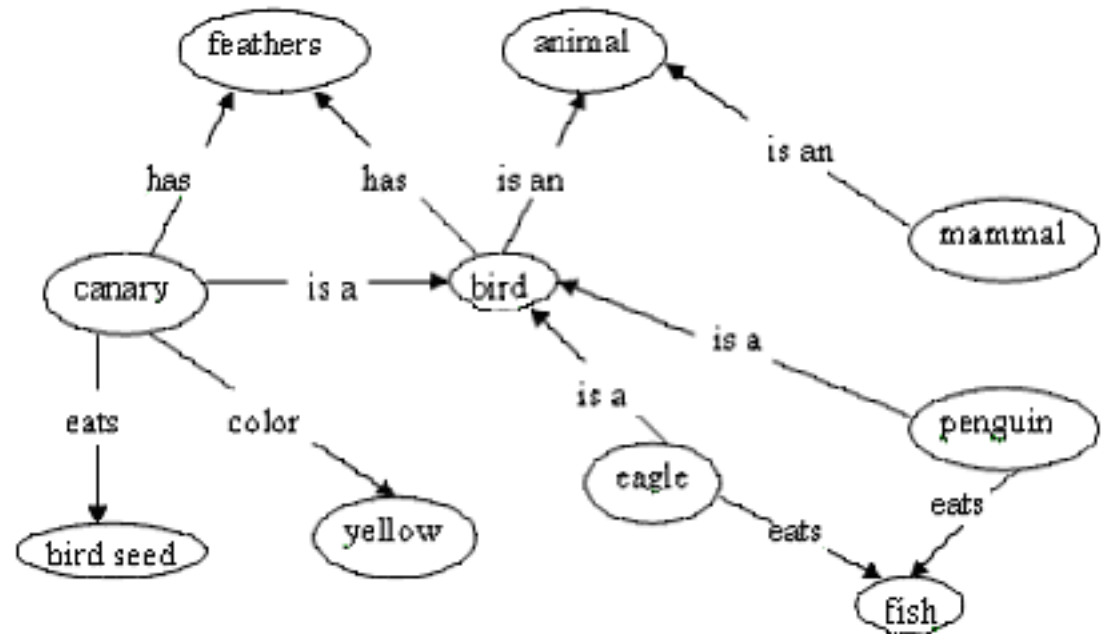
- Front end processing is well understood, it is the realm of traditional digital signal and image processing
- Front end algorithms generally apply the same computation over large arrays of elements, they are data parallel, and communication tends to be local
  - Most neuromorphic VLSI, for example, generally models front end processing
  - Another excellent example of such an architecture is the CNN (Cellular Non-linear Network) developed by Chua, Roska et al.

## But Then There's The “Back-End” ...

- In the early days of computing, “Artificial Intelligence” focused on the representation and use of contextual and semantic information
- Knowledge was generally represented by a set of rules
  - Logic operations (e.g., based on first order predicate calculus) were used to manipulate the rules and “infer” the properties and state of an environment from input data
  - By using inferred state, planning and decisions were possible
- However, these systems were “brittle,” exhibiting limited flexibility, generalization, and graceful degradation
- And they were unable to adapt dynamically (i.e., learn) within the context of most real world applications

## What Does The “Back End” (i.e., ISP) Do?

- Simplistically it captures information on “high order” relationships of “abstract” entities
- This information can often be represented graphically
- These used to be “rules,” but more recently they have become probabilistic relationships
- Inference is then performed over these structures
- This is not cognition, but it is a necessary component



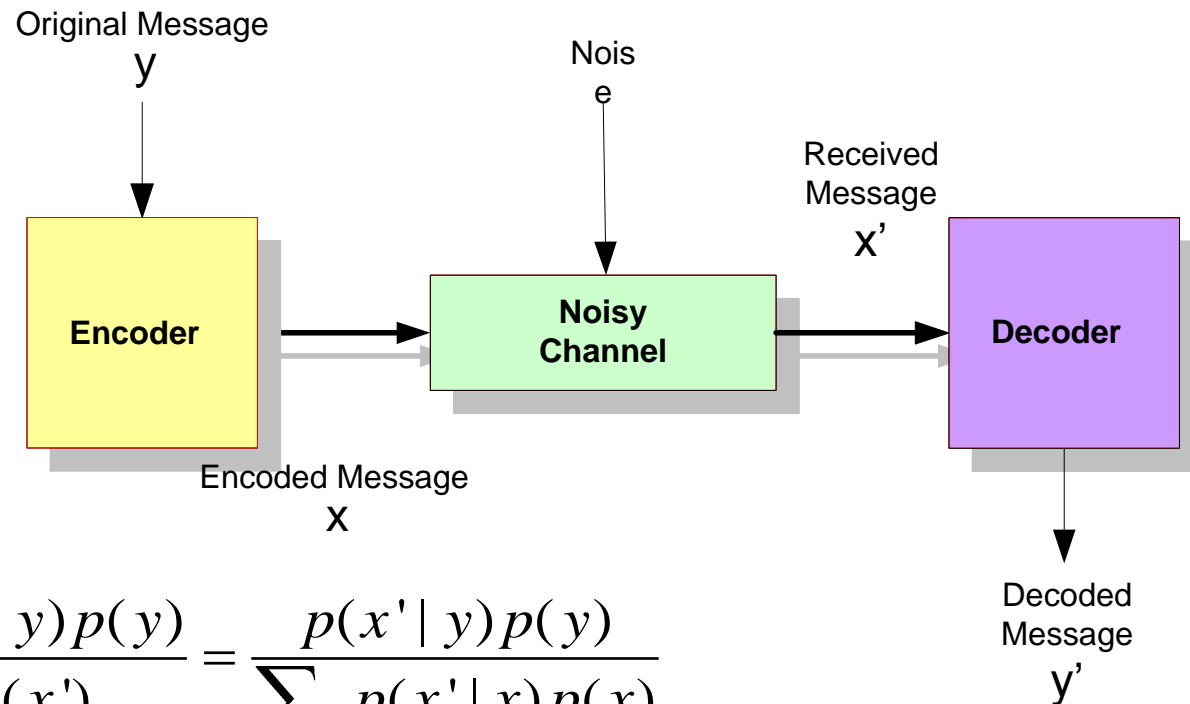
The ISP Toolbox –  
Still mostly empty after  
all these years ...



# Bayesian Networks

- We now have **Bayesian networks**
  - A major contribution to this effort was the work of Judea Pearl
    - Pearl, J., *Probabilistic Reasoning In Intelligent Systems – Networks of Plausible Inference*, Morgan Kaufman, 1988 & 1997
  - These systems are far less brittle and they also more faithfully model aspects of animal behavior, since animals learn from their surroundings and appear to do a kind of probabilistic inference from learned knowledge as they interact with their environment
  
- Bayesian nets express the structured, graphical representations of probabilistic relationships between several random variables
  - The graph structure is an explicit representation of conditional dependence (encoded by network edges)
  
- We propose that Bayesian Inference, in its various forms, will be an increasing important computation in future systems

**The Inference Problem:**  
Choose the most likely  $y$ ,  
based on  $P[y|x']$

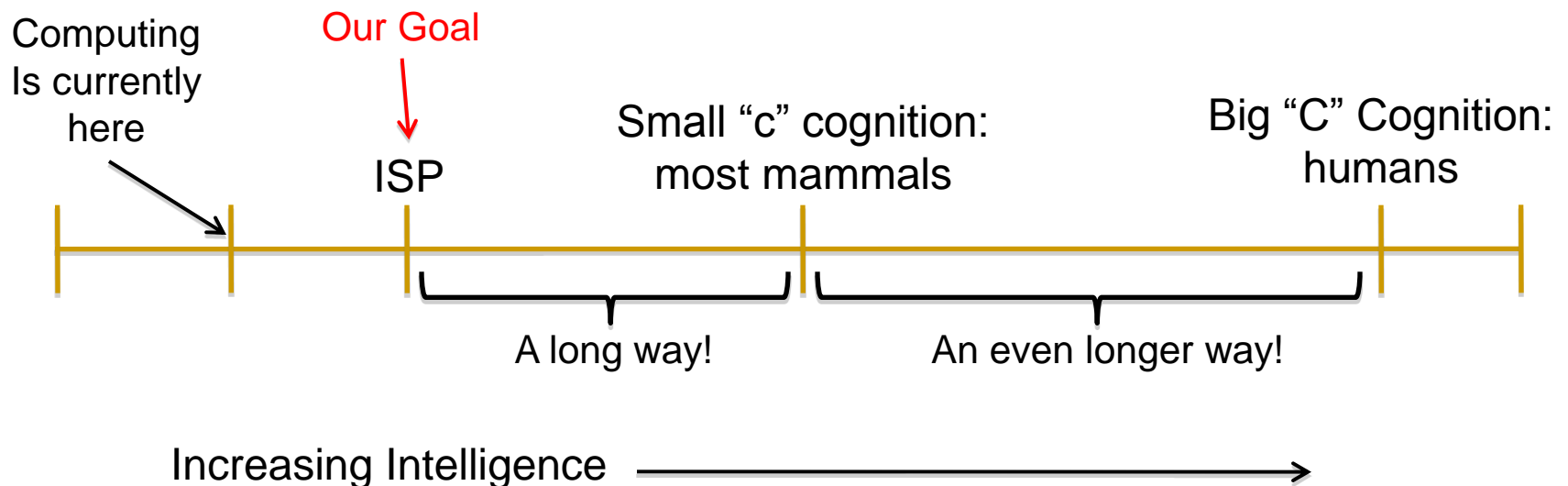


$$p(y | x') = \frac{p(x' | y)p(y)}{p(x')} = \frac{p(x' | y)p(y)}{\sum_x p(x' | x)p(x)}$$

We need to “infer” the most likely original message given the data we received and our knowledge of the statistics of channel errors and the messages being generated

# The Scope of Our Project

- Conceptually one can think of “computational intelligence” as a spectrum
- And though not universally accepted, it has been hypothesized that this spectrum is more or less continuous from one end to the other



## Learning / Adaptation

- Animals learn in real time from their surroundings, and appear to do a kind of probabilistic inference from learned knowledge as they interact with their environments, but learning is difficult for standard Bayesian Networks
- Therefore, a desirable characteristic of ISP is incremental, integrative adaptation / learning during system operation

## Problem: Scaling

- The scaling limitations of both symbolic and traditional neural network approaches constitute a major shortcoming - Inference in Bayesian Nets is NP-Hard
- It is very likely that sheer size is a major component of the “secret sauce of cognition”
  - “Size matters!” (*Godzilla*, the movie, 1998)
  - Consider the differences: hundreds of rules or thousands of nodes vs. millions / billions of neurons
- We need models that learn and scale
- And, consequently, we need hardware/software platforms that allow self-organization and adaptation, and which scale to very large sizes

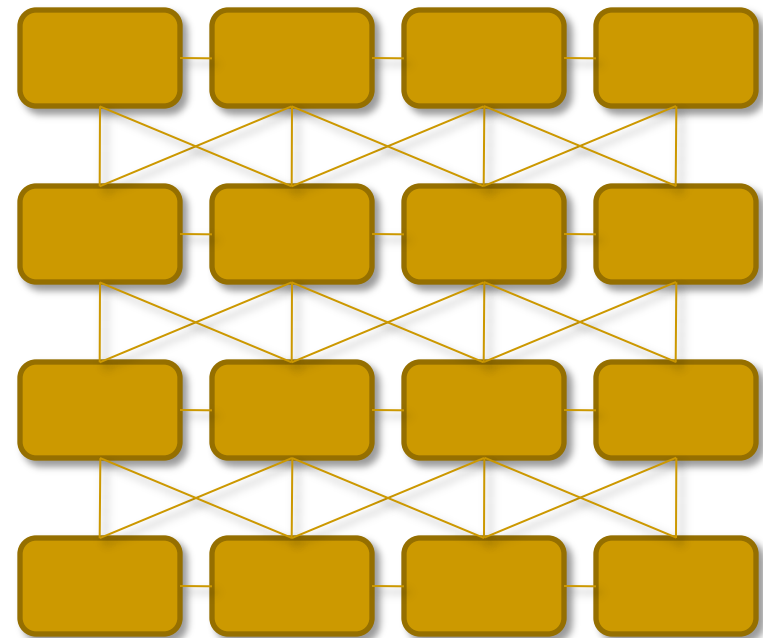
## Solution: Distributed Representation Bayesian Networks

- In a distributed data representation each coding unit participates in multiple distinct representations
  - A representation has a more “statistical” aspect to it by virtue of the ensemble of vectors in its representation
- An important variation is the sparse distributed code
  - Where all cells in the code have an equal response probability across all representations, but a low response probability for any single representation
  - The dimensionality of the space spanned by the input vectors is not necessarily reduced
- How do we build Bayesian Networks with distributed representations?
  - Use a regularly structured network, where the nodes create their own representations

From *Big Brain* by Gary Lynch and Rick Granger (Palgrave MacMillan 2008):

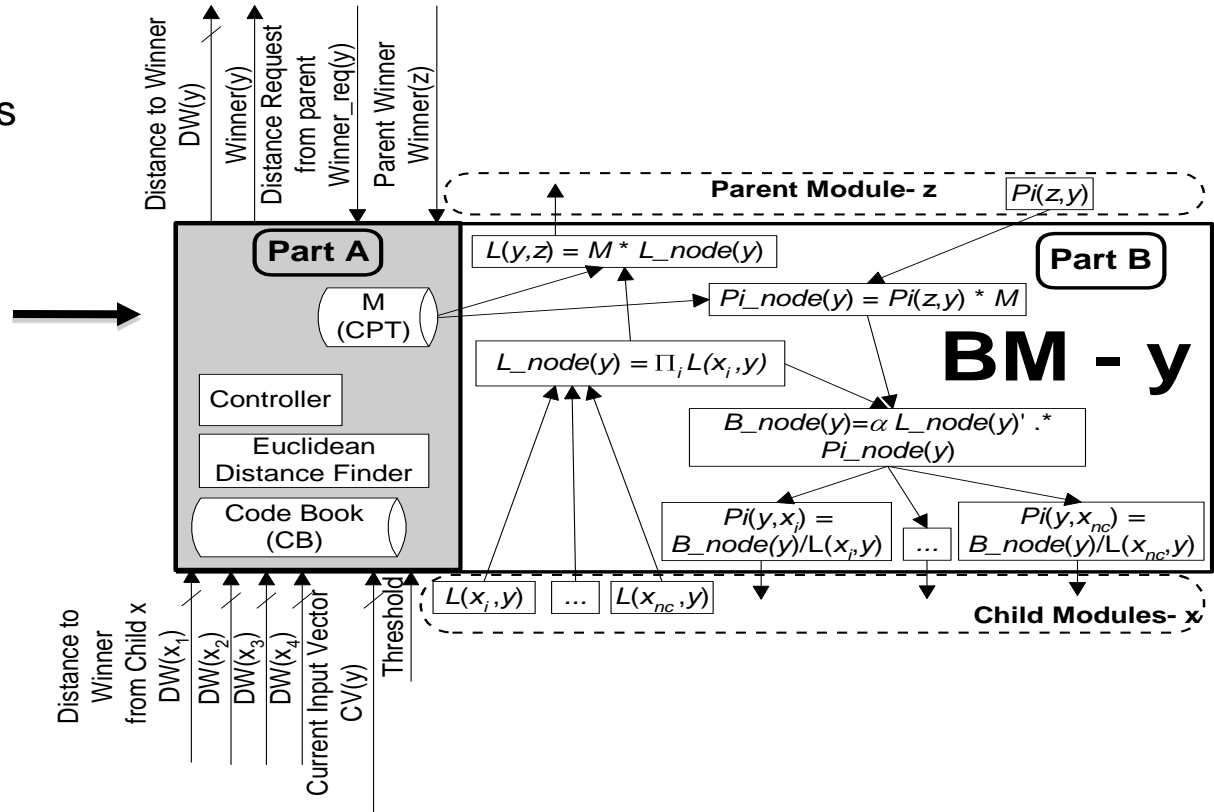
- “... the ‘front end’ circuits of the brain ... specialize in their own particular visual and auditory inputs, the rest of the brain converts these to random-access encodings in association areas throughout cortex. ... these areas take initial sensory information and construct grammars”
- “These are not grammars of linguistic elements, they are grammatical organizations (nested, hierarchical, sequences of categories) of percepts – visual, auditory, ...”
- “Processing proceeds by incrementally assembling these constructs ... these grammars generate successively larger ‘proto-grammatical fragments,’ eventually constituting full grammars”
- “They thus are not built in the manner of most hand-made grammars; they are statistically assembled, to come to exhibit rule-like behavior, of the kind expected for linguistic grammars”

- Each node sees only a subset of the nodes in the previous layer
- And each node's subset has some but not complete overlap with its neighbors, which is one way to distribute the representation
- One can implement exact BBP, but due to the small granularity of nodes, various kinds of approximate inference may be possible
- In fact some researchers have speculated that approximate inference is actually better!



# “Bayesian Memory”

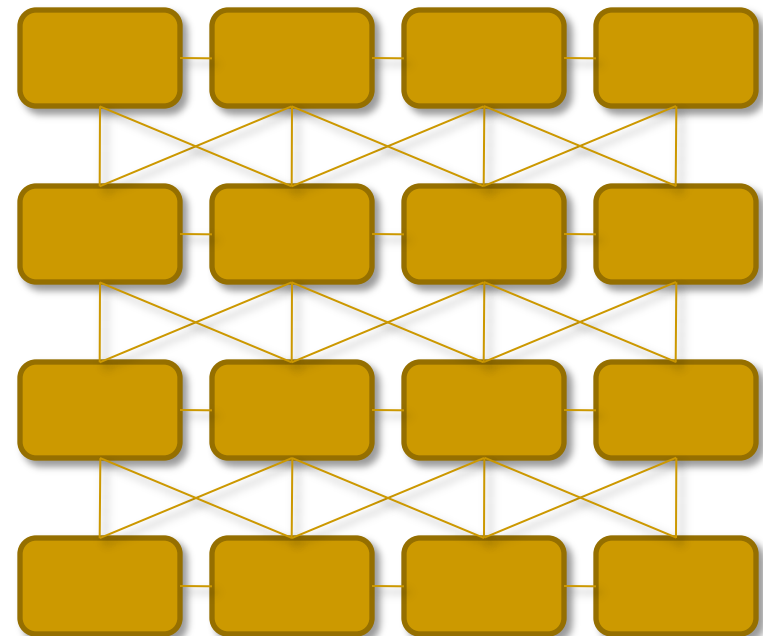
- One such approximation is Bayesian Memory
- The Bayesian Memory Building Blocks can be implemented directly in a functional manner
- Part A is a maximum entropy, dimension reducing vector quantizer
- And Part B implementing Bayesian Belief Propagation or an approximation



## Associative Memory Approximates Bayesian Inference

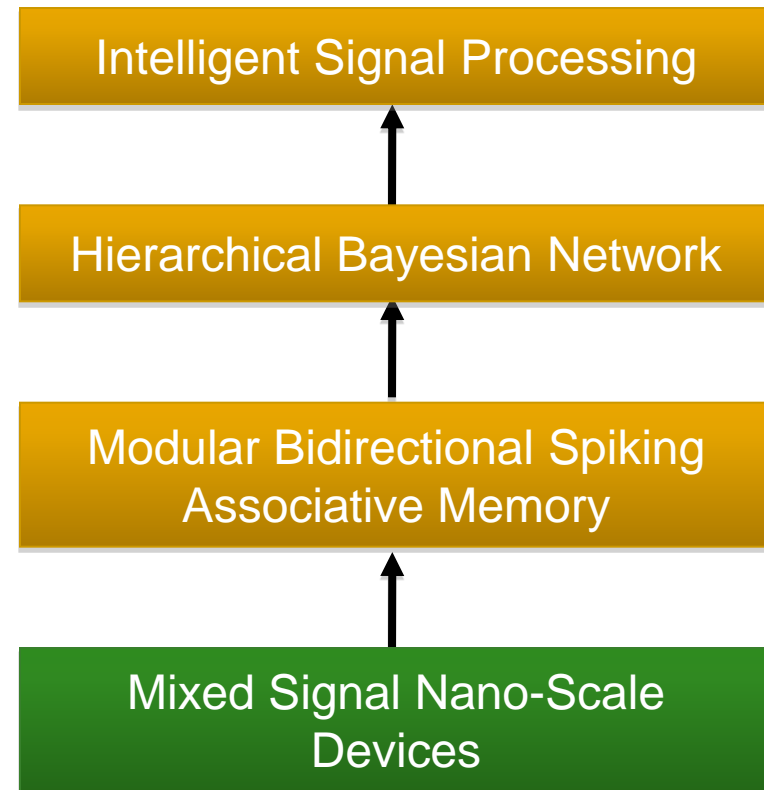
- An even more hardware friendly approximation is traditional associative memory – which can be thought of as a massive constraint satisfaction system
- We have shown that some associative memory models (e.g., Palm and Willshaw), can learn dynamically, approximate a Vector Quantizer and approximate Bayesian classification
- To build a Bayesian Memory using associative memory, two mappings are needed: a forward path and a feedback path, so we use a Bidirectional Associative Memory (BAM)
- Inference is massively parallel, but global convergence can be a problem
- We are now investigating various nano-scale technologies that primarily use analog representations

- Implement each node in the network as a modest sized associative memory
- Implement each associative memory using a nano-scale recursive feedback structure
- Inter-module connectivity is almost total, but inter-module connectivity is very sparse and can be multiplexed
- **We believe that we now have a path from ISP applications to nanoscale computation**



# One Possible Path To Implementation

- Our approach is top-down, not bottom up
- There is a large range of implementation options
  - 1000 Atom processors
  - Neuromorphic VLSI
  - Nano-grids
  - Other nano ...



# Field Adaptable Bayesian Arrays

Each Square is a single Bayesian Memory Node

Nanoscale  
Analog  
Associative  
Memory

Nanoscale  
Analog  
Associative  
Memory

Nanoscale  
Analog  
Associative  
Memory

Nanoscale  
Analog  
Associative  
Memory

Thousands of  
of nodes  
with full  
connectivity

CMOS provides  
sparse  
inter-module  
connectivity,  
I/O, signal  
amplification

## FABA – Long Term Goal

- A roughly 1 inch die containing several billion CMOS transistors and close to a trillion molecular devices
- Operating at over 10 Tera-Ops
- Extensive fault / defect tolerance
- Performs real-time, adaptive bayesian inference over very complex spatial and temporal knowledge structures
- Available in a portable, hand-held, low power devices