Back to Agenda

Probabilistic CMOS Technology for Embedded Cognitive Applications*

Bilge Akgul
Lakshmi Narasimhan Chakrapani
Krishna V Palem

Center for Research on Embedded Systems and Technology
Georgia Institute of Technology

***This work was supported in part by DARPA under contract #F30602-02-2-0124, by the DARPA ACIP program under contract #FA8650-04-C-7126 through a subcontract from USC-ISI and by an award from Intel corporation.**

Contributions

- Computing platforms based on devices with probabilistic behavior
    - Computing platforms are not only noise tolerant but harness statistical behavior to compute
- Orders of magnitude savings in energy and performance at the application level
    - Enable the implementation of complex cognitive and probabilistic applications
- Higher quality-of-solution for cognitive and probabilistic applications
    - By harnessing naturally probabilistic substrates

✓Application analysis and optimization methodology

✓Metrics for evaluating and characterizing computing platforms based on *Probabilistic* CMOS technology

✓Experimental methodology for performance evaluation of computing platforms based on *Probabilistic* CMOS technology

Outline

Role of Probability in Cognitive Applications
Current Implementation Methodologies and Chief Concerns
Exploiting Technology Trends – *Probabilistic* CMOS Technology
Probabilistic System on a Chip (PSOC) Architectures
Optimizing Probabilistic System on a Chip (PSOC) Architectures
PSOC Architectures for Conventional Signal Processing Applications
Next Steps

Outline

Role of Probability in Cognitive Applications

- Probabilistic Algorithms find widespread use in cognitive applications
  - Probabilistic models of human reasoning
    - Bayesian model, randomized neural network model
  - Ability to generate good execution instances for arbitrarily chosen problem instances
    - Good execution for *all* problem instances
  - Ability for rapid and uniform exploration of search space
    - Heuristic optimization, heuristic search techniques

- Historical benefits
  - Rapid execution
  - Good quality of solution

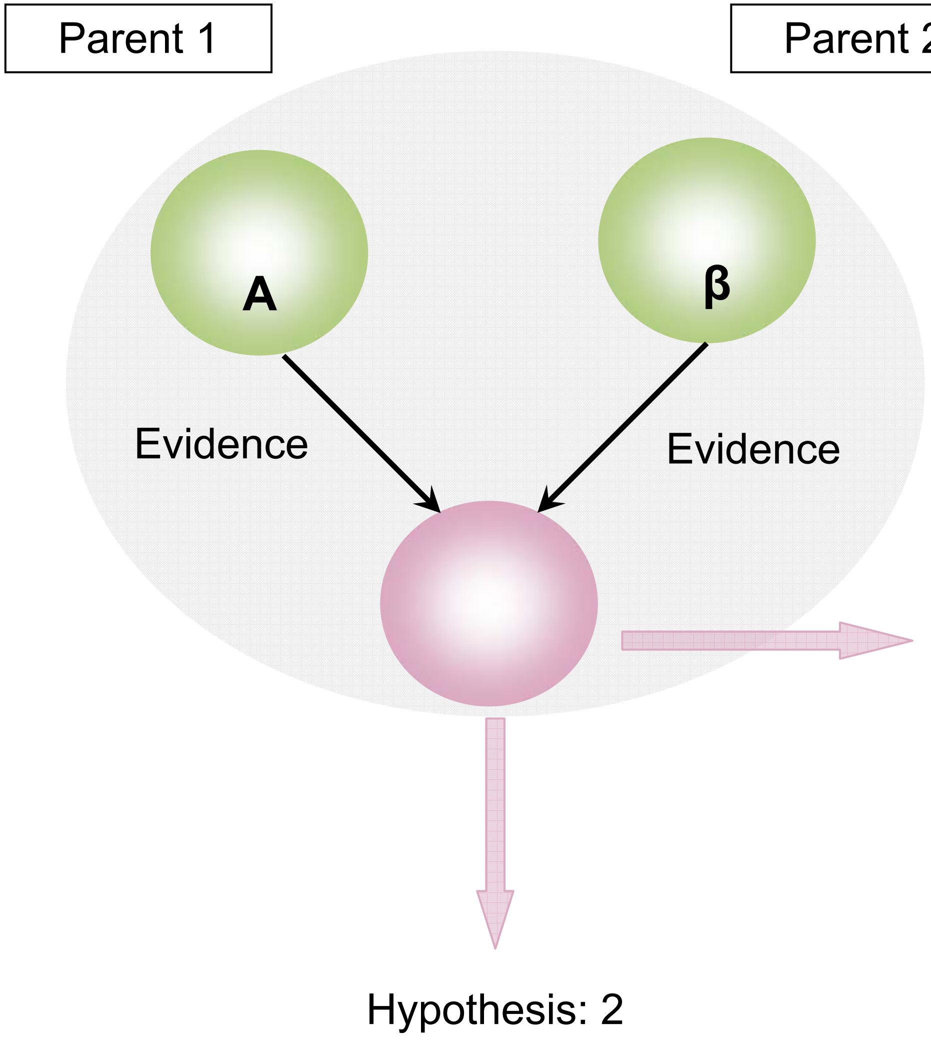
Outline

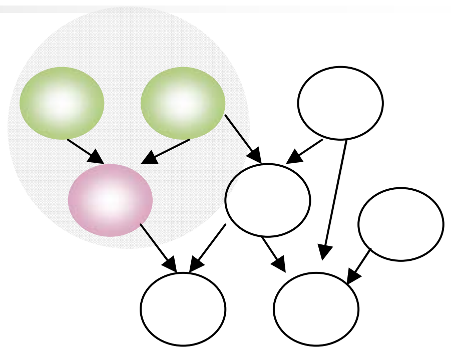
- Role of Probability in Cognitive Applications
  - Probabilistic model of human reasoning
  - Good execution instances for arbitrary problem instances
  - Rapid exploration of search space

Probabilistic Models of Human Reasoning: An Example

- Bayesian inference finds widespread use in probabilistic cognitive applications
  - Probabilities are interpreted as “degree of belief” in a hypothesis
  - Infer a hypothesis based on “evidence”
  - Can be performed using a Bayesian Network
- A Bayesian network is a directed acyclic graph
  - Nodes represent variables, edges represent dependence relationship between the variables
- A node *infers* a value based on the values of its parents and an associated conditional distribution
  - Different network topologies and conditional distributions can solve different problems

Bayesian Network





Evidence supplied by parents	Hypothesis: 1	Hypothesis: 2
A and α	0.9	0.1
A and β	0.1	0.9
B and α	0.5	0.5
B and β	0.3	0.7

Outline

- Role of Probability in Cognitive Applications
  - Probabilistic model of human reasoning
  - Good execution instances for arbitrary problem instances
  - Rapid exploration of search space

Rapid Exploration of Search Space - An Example

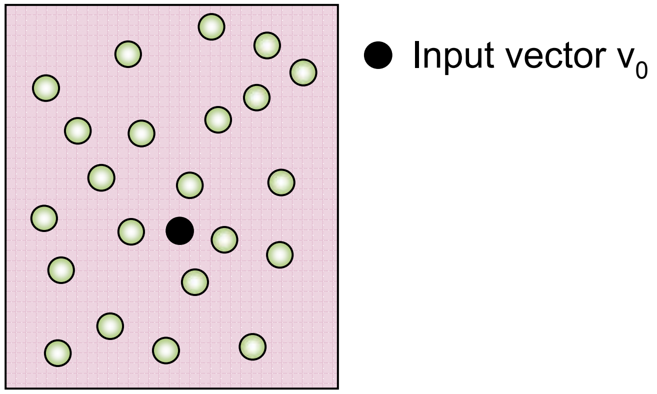
The Problem

- **Inputs**
  - A *dictionary* of *n* vectors
  - An input vector v_0
  - A distance metric $f(v, v_0)$
- **Problem**
  - find *k* vectors from *n* which are closest to v_0

Rapid Exploration of Search Space - An Example

The Problem

- **Inputs**
  - A *dictionary* of *n* vectors
  - An input vector v_0
  - A distance metric $f(v, v_0)$
- **Problem**
  - find *k* vectors from *n* which are closest to v_0

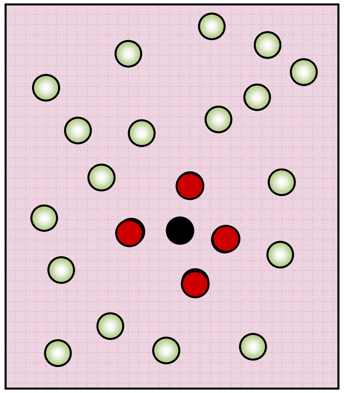


Dictionary of *n* vectors

Rapid Exploration of Search Space - An Example

The Problem

- **Inputs**
  - A *dictionary* of *n* vectors
  - An input vector v_0
  - A distance metric $f(v, v_0)$
- **Problem**
  - find *k* vectors from *n* which are closest to v_0



Dictionary of *n* vectors

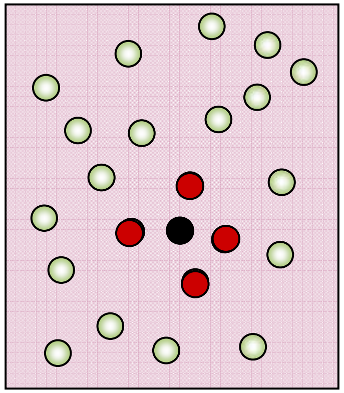
● Input vector v_0

● 4 vectors which are closest to v_0 (for $k = 4$)

Rapid Exploration of Search Space - An Example

The Problem

- **Inputs**
  - A *dictionary* of *n* vectors
  - An input vector v_0
  - A distance metric $f(v, v_0)$
- **Problem**
  - find *k* vectors from *n* which are closest to v_0

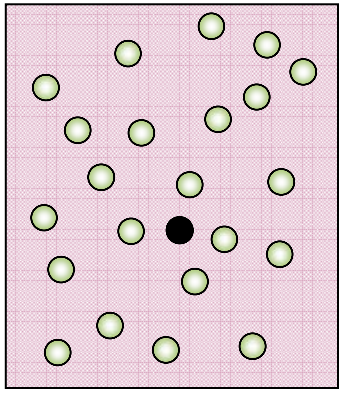


● Input vector v_0

● 4 vectors which are closest to v_0 (for $k = 4$)

Dictionary of *n* vectors

A Probabilistic Algorithm



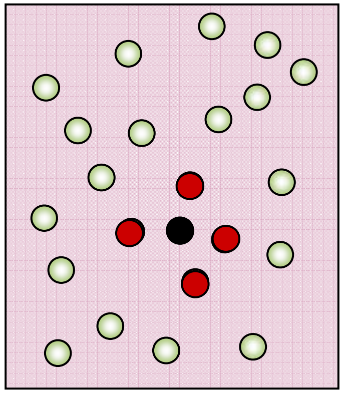
● Input vector v_0

Dictionary of *n* vectors

Rapid Exploration of Search Space - An Example

The Problem

- **Inputs**
  - A *dictionary* of *n* vectors
  - An input vector v_0
  - A distance metric $f(v, v_0)$
- **Problem**
  - find *k* vectors from *n* which are closest to v_0

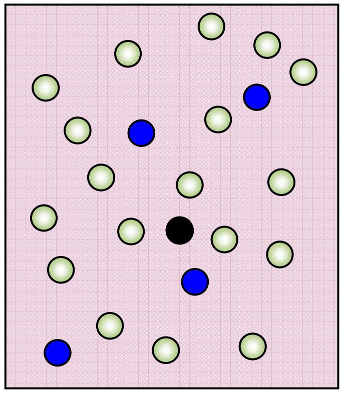


● Input vector v_0

● 4 vectors which are closest to v_0 (for *k = 4*)

Dictionary of *n* vectors

A Probabilistic Algorithm



● Input vector v_0

● Randomly selected vector

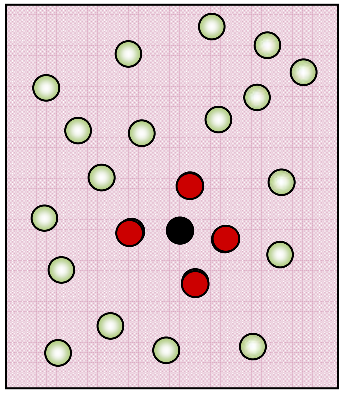
Dictionary of *n* vectors

- Select *m* vectors at random

Rapid Exploration of Search Space - An Example

The Problem

- **Inputs**
  - A *dictionary* of *n* vectors
  - An input vector v_0
  - A distance metric $f(v, v_0)$
- **Problem**
  - find *k* vectors from *n* which are closest to v_0

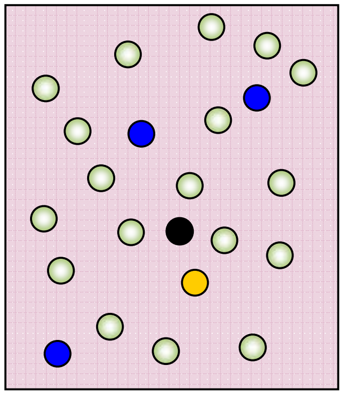


● Input vector v_0

● 4 vectors which are closest to v_0 (for $k = 4$)

Dictionary of *n* vectors

A Probabilistic Algorithm



● Input vector v_0

● Randomly selected vector

● d^{th} closest vector v_d

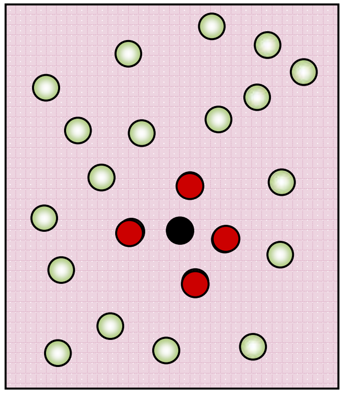
Dictionary of *n* vectors

- Select *m* vectors at random
- Select d^{th} closest vector to v_0, (v_d) from the set *m*

Rapid Exploration of Search Space - An Example

The Problem

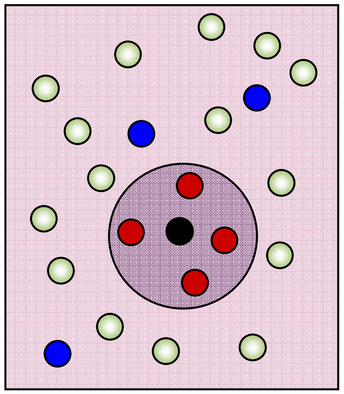
- **Inputs**
  - A *dictionary* of *n* vectors
  - An input vector v_0
  - A distance metric $f(v, v_0)$
- **Problem**
  - find *k* vectors from *n* which are closest to v_0



- ● Input vector v_0
- ● 4 vectors which are closest to v_0 (for $k = 4$)

Dictionary of *n* vectors

A Probabilistic Algorithm



- ● Input vector v_0
- ● Randomly selected vector
- ● d^{th} closest vector v_d
- ● Vectors which are closer to v_0 than v_d is to v_0

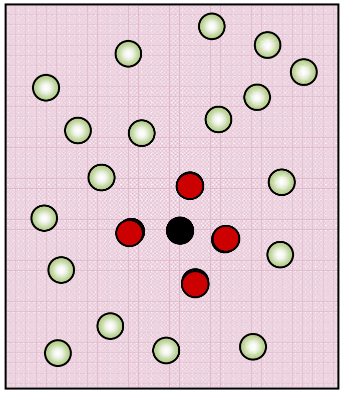
Dictionary of *n* vectors

- Select *m* vectors at random
- Select d^{th} closest vector to v_0, (v_d) from the set *m*
- Select and output all vectors *v* from dictionary such that $f(v,v_0) < f(v_d,v_0)$
  - **Thresholding instead of Sorting !**

Rapid Exploration of Search Space - An Example

The Problem

- **Inputs**
  - A *dictionary* of *n* vectors
  - An input vector v_0
  - A distance metric $f(v, v_0)$
- **Problem**
  - find *k* vectors from *n* which are closest to v_0

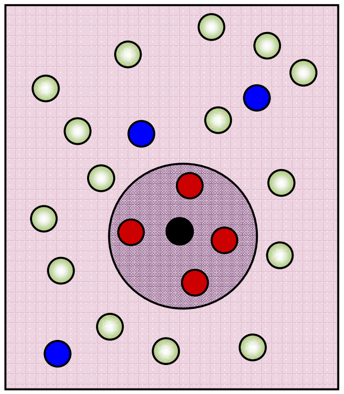


● Input vector v_0

● 4 vectors which are closest to v_0 (for $k = 4$)

Dictionary of *n* vectors

A Probabilistic Algorithm



● Input vector v_0

● Randomly selected vector

● d^{th} closest vector v_d

● Vectors which are closer to v_0 than v_d is to v_0

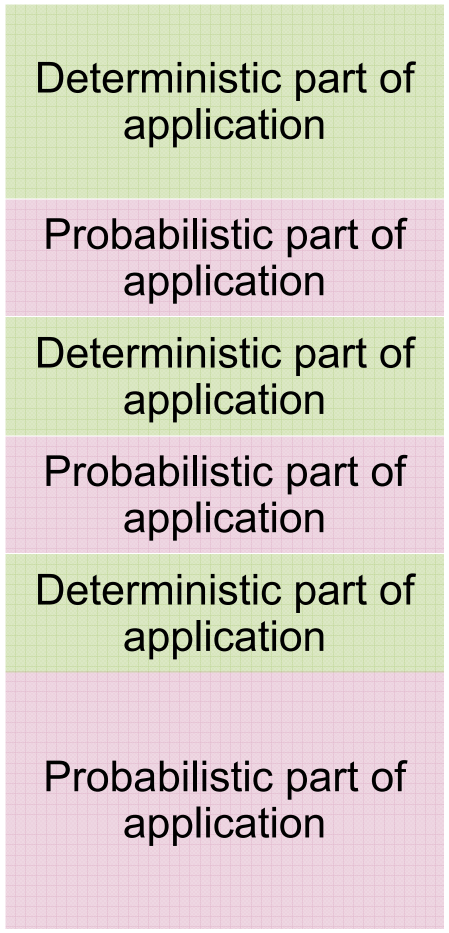
Dictionary of *n* vectors

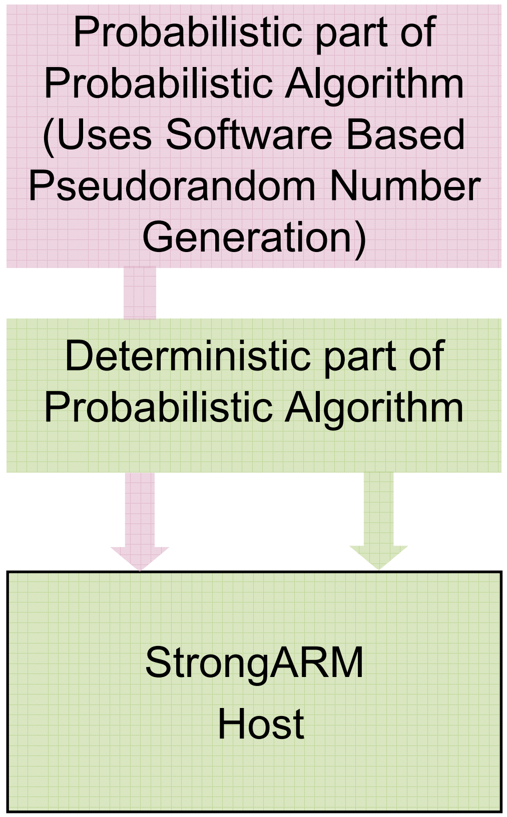
- Select *m* vectors at random
- Select d^{th} closest vector to v_0, (v_d) from the set *m*
- Select and output all vectors *v* from dictionary such that $f(v,v_0) < f(v_d,v_0)$
  - **Thresholding instead of Sorting !**

- *d* is a design parameter that determines probability of error
- *m* can be calculated in *O(d* log *m)* time
- Fast algorithm, a single pass is enough to solve the problem
- Algorithm is erroneous if it returns more than or less than *k* vectors
- Probability of error $e^{-\left(\frac{1}{d!}\right)\left(\frac{mk}{n}\right)^{d}}$ is very low
- Applications in finger print matching, facial recognition, data mining, document similarity

Outline

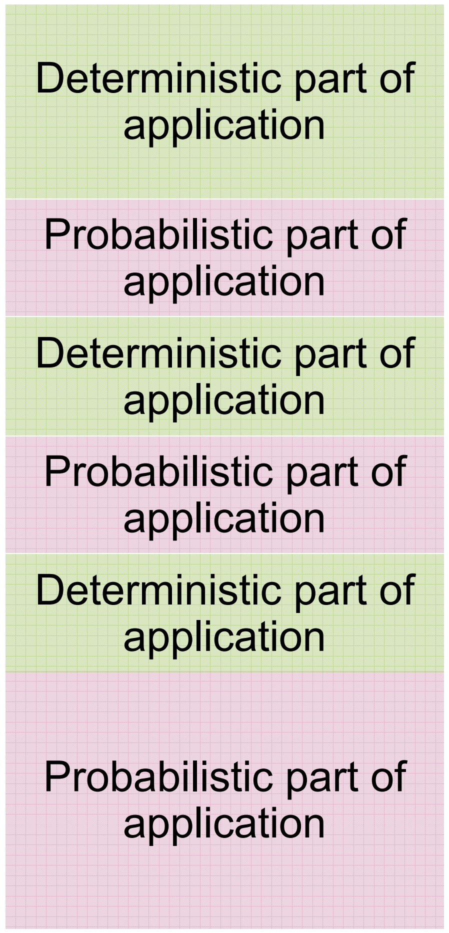
Current Day Implementation Methodologies

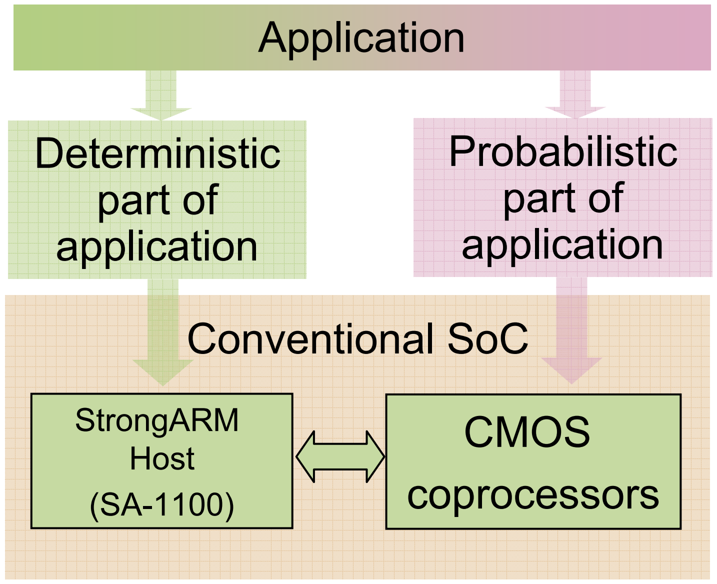

Deterministic part of application
Probabilistic part of application
Deterministic part of application
Probabilistic part of application
Deterministic part of application
Probabilistic part of application


Probabilistic part of Probabilistic Algorithm (Uses Software Based Pseudorandom Number Generation)
Deterministic part of Probabilistic Algorithm
StrongARM Host

Probabilistic algorithm on a host processor

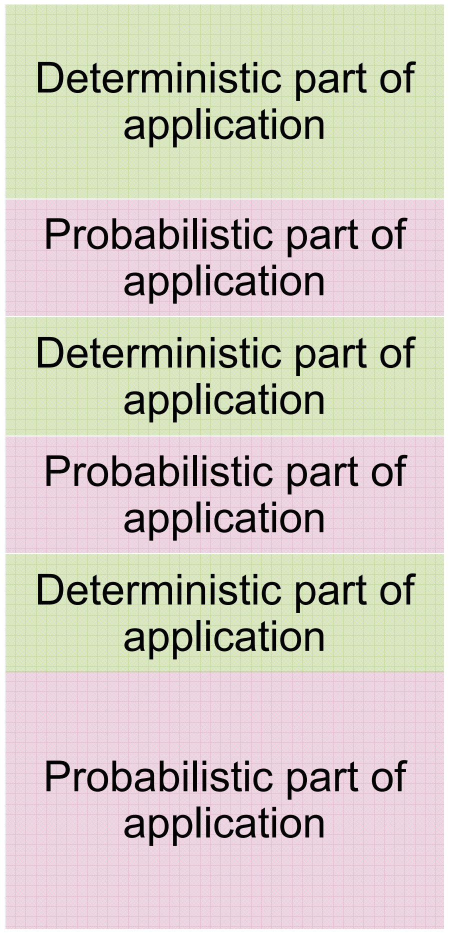
Current Day Implementation Methodologies

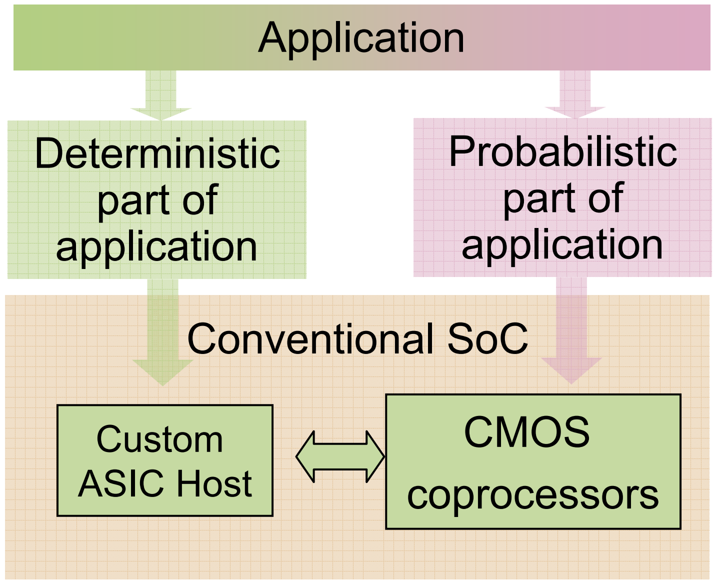

Deterministic part of application
Probabilistic part of application
Deterministic part of application
Probabilistic part of application
Deterministic part of application
Probabilistic part of application



Conventional System on a Chip

Current Day Implementation Methodologies


Deterministic part of application
Probabilistic part of application
Deterministic part of application
Probabilistic part of application
Deterministic part of application
Probabilistic part of application


Application
Deterministic part of application
Probabilistic part of application
Conventional SoC
Custom ASIC Host
CMOS coprocessors

Fully Custom System on a Chip

Chief Concerns

- Cost-benefit tradeoff for probabilistic cognitive applications

- High performance should not be offset by high cost of randomization
    - "Select a vector uniformly at random"
        - What is the cost of selecting a vector at random ?

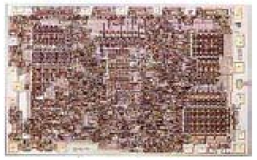
- (Intended) good quality of solution should not be compromised by bad quality of randomization
    - "Select a vector uniformly at random"
        - Truly uniformly at random ?

Outline

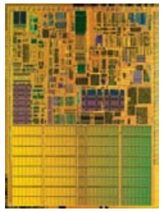
Role of Probability in Cognitive Applications
Current Implementation Methodologies and Chief Concerns
Exploiting Technology Trends – *Probabilistic* CMOS Technology
Probabilistic System on a Chip (PSOC) Architectures
Optimizing Probabilistic System on a Chip (PSOC) Architectures
PSOC Architectures for Conventional Signal Processing Applications
Next Steps

Exploiting Technology Trends

- The characteristics of VLSI systems are changing
  - Shrinking feature sizes, Increasing transistor densities



Intel 4004, 10 microns 2300 transistors (1971)

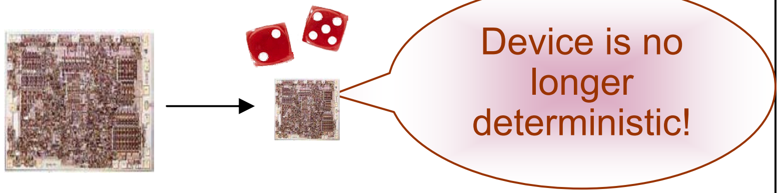


Intel Pentium M, 0.90 microns 170 Million transistors (2003)

⇨

parametric variations and *noise* yield probabilistic devices

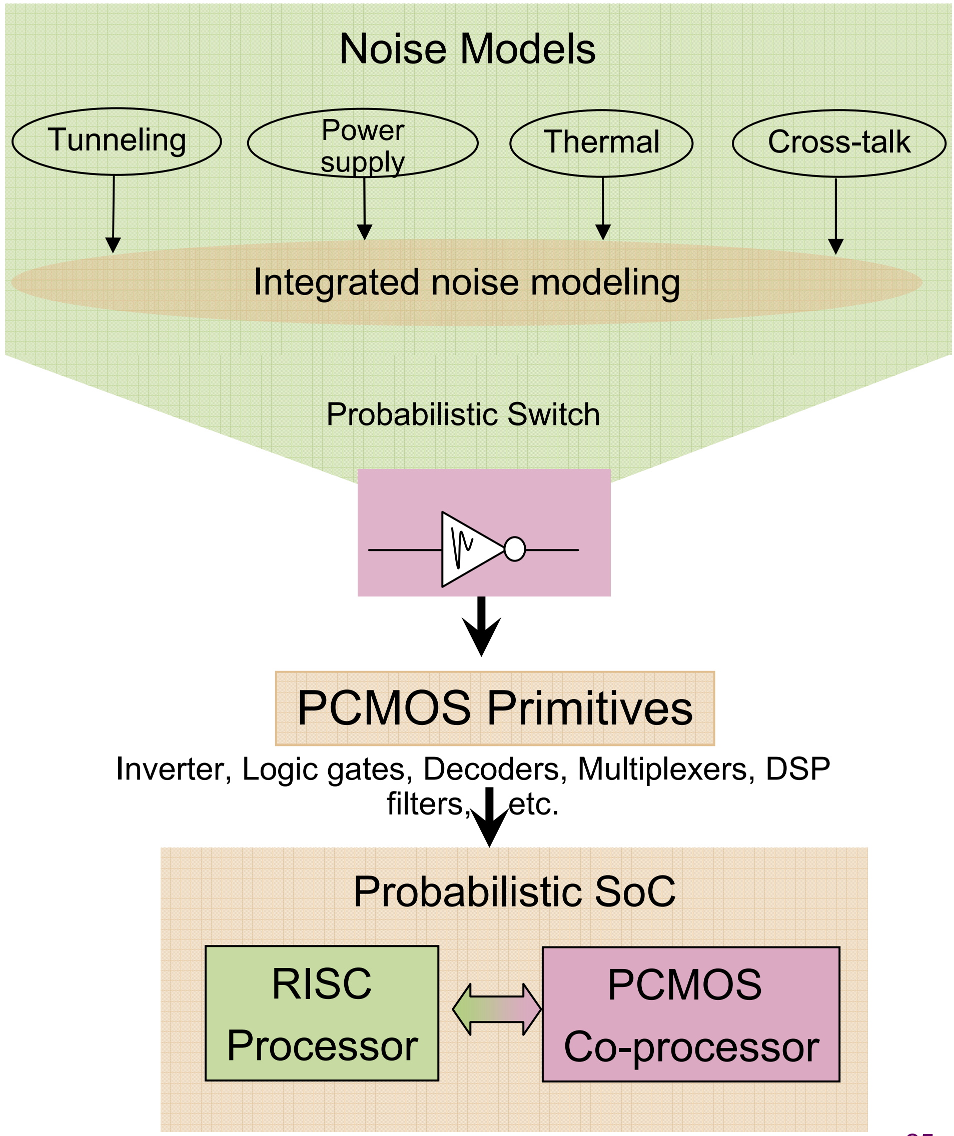
Shrinking feature sizes and increasing densities increase power density



"Relaxing the requirement of 100% correctness for devices and interconnects may dramatically reduce costs of manufacturing, verification, and test. Such a paradigm shift is likely forced in any case by technology scaling, which leads to more transient and permanent failures of signals, logic values, devices, and interconnects" – ITRS Road Map

Central Idea

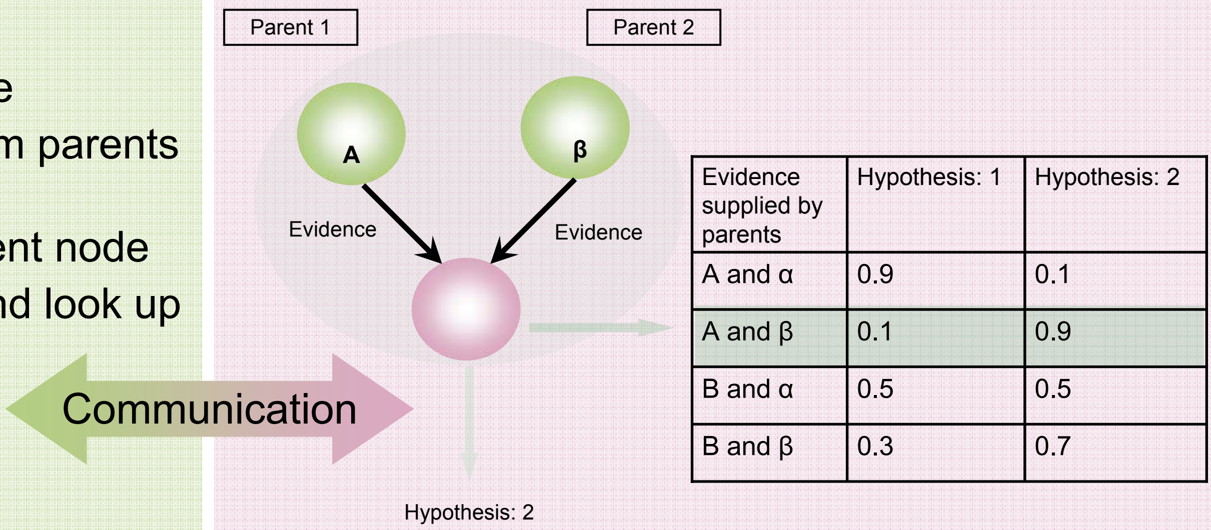
- Model and use noise susceptible *switches* as architecture building blocks
  - Behavior of such a switch is *probabilistic* yielding Probabilistic CMOS (PCMOS)
- Use PCMOS building blocks to realize *probabilistic primitives* of probabilistic applications
  - Non-deterministic behavior of devices is useful, not harmful
  - Orders of magnitude improvement in energy and performance
  - High quality of randomization



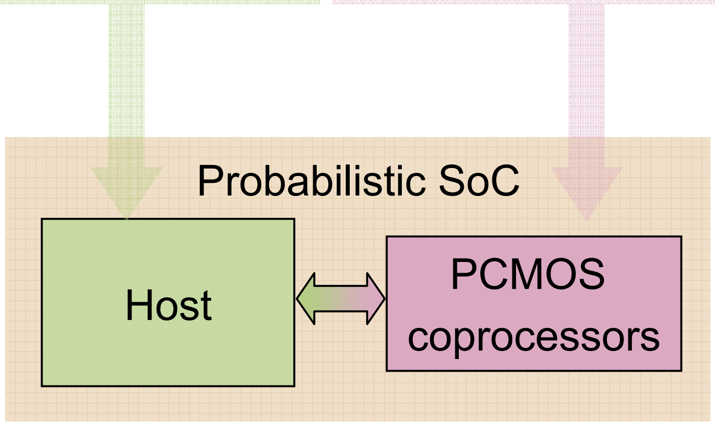
Outline

An Example PSOC Design

- For all nodes do
  - Choose current node
  - Collect evidence from parents
  - Select Table which corresponds to current node
  - Supply evidences and look up hypothesis
  - Update hypotheses
- End For



Evidence supplied by parents	Hypothesis: 1	Hypothesis: 2
A and α	0.9	0.1
A and β	0.1	0.9
B and α	0.5	0.5
B and β	0.3	0.7



An Example PSOC Design

Consider the core probabilistic step of Bayesian Network

Evidence supplied by parents	Hypothesis: 1	Hypothesis: 2
A and α	0.9	0.1
A and β	0.1	0.9
B and α	0.5	0.5
B and β	0.3	0.7

Evidence supplied by parents	Hypothesis: 1	Hypothesis: 2
A and α	0.9	0.1
A and β	0.1	0.9
B and α	0.5	0.5
B and β	0.3	0.7

An Example PSOC Design

Consider the core probabilistic step of Bayesian Network

Evidence supplied by parents	Hypothesis: 1	Hypothesis: 2
A and α	0.9	0.1
A and β	0.1	0.9
B and α	0.5	0.5
B and β	0.3	0.7

Evidence supplied by parents	Hypothesis: 1	Hypothesis: 2
A and α	0.9	0.1
A and β	0.1	0.9
B and α	0.5	0.5
B and β	0.3	0.7

0.1 0.9 P3 P4 P5 P6 P7

An Example PSOC Design

Consider the core probabilistic step of Bayesian Network

Evidence supplied by parents	Hypothesis: 1	Hypothesis: 2
A and α	0.9	0.1
A and β	0.1	0.9
B and α	0.5	0.5
B and β	0.3	0.7

Evidence supplied by parents	Hypothesis: 1	Hypothesis: 2
A and α	0.9	0.1
A and β	0.1	0.9
B and α	0.5	0.5
B and β	0.3	0.7

PCMOS

Switch	Switch	Switch	Switch	Switch	Switch	Switch

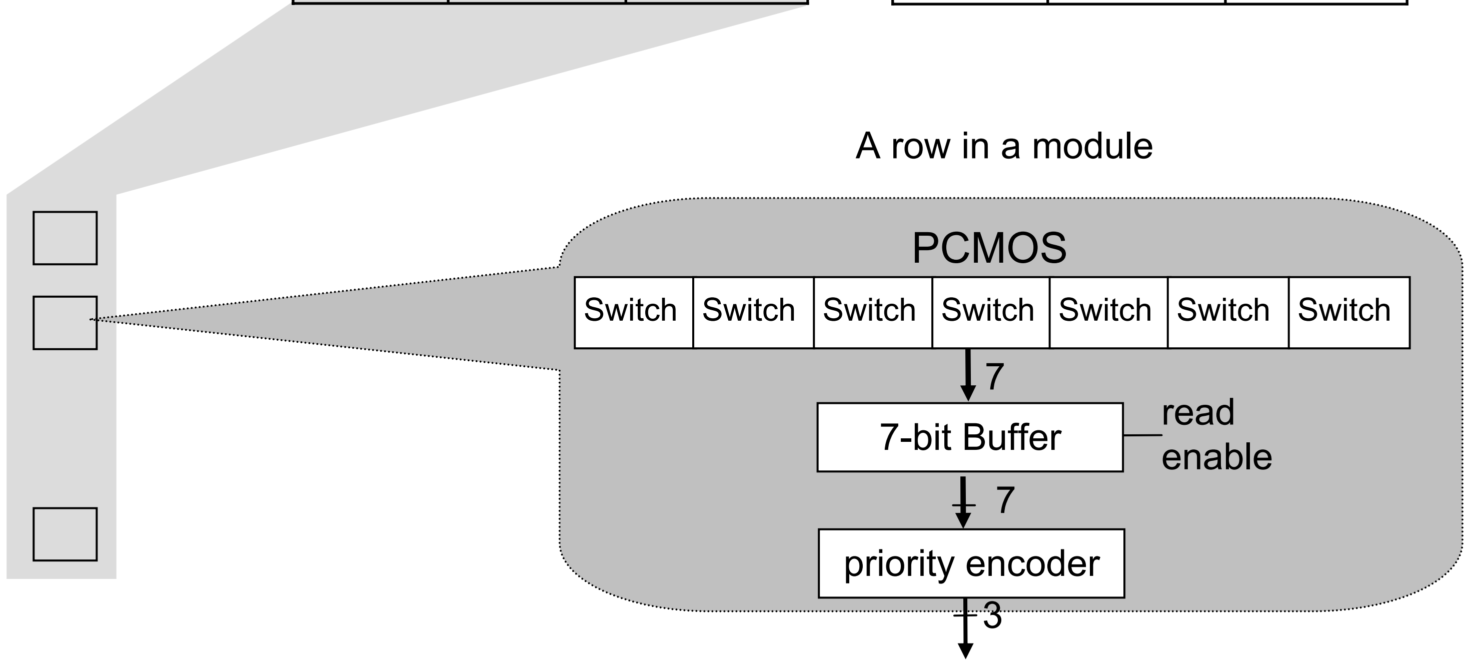
An Example PSOC Design

Consider the core probabilistic step of Bayesian Network

Evidence supplied by parents	Hypothesis: 1	Hypothesis: 2
A and α	0.9	0.1
A and β	0.1	0.9
B and α	0.5	0.5
B and β	0.3	0.7

Evidence supplied by parents	Hypothesis: 1	Hypothesis: 2
A and α	0.9	0.1
A and β	0.1	0.9
B and α	0.5	0.5
B and β	0.3	0.7

A row in a module

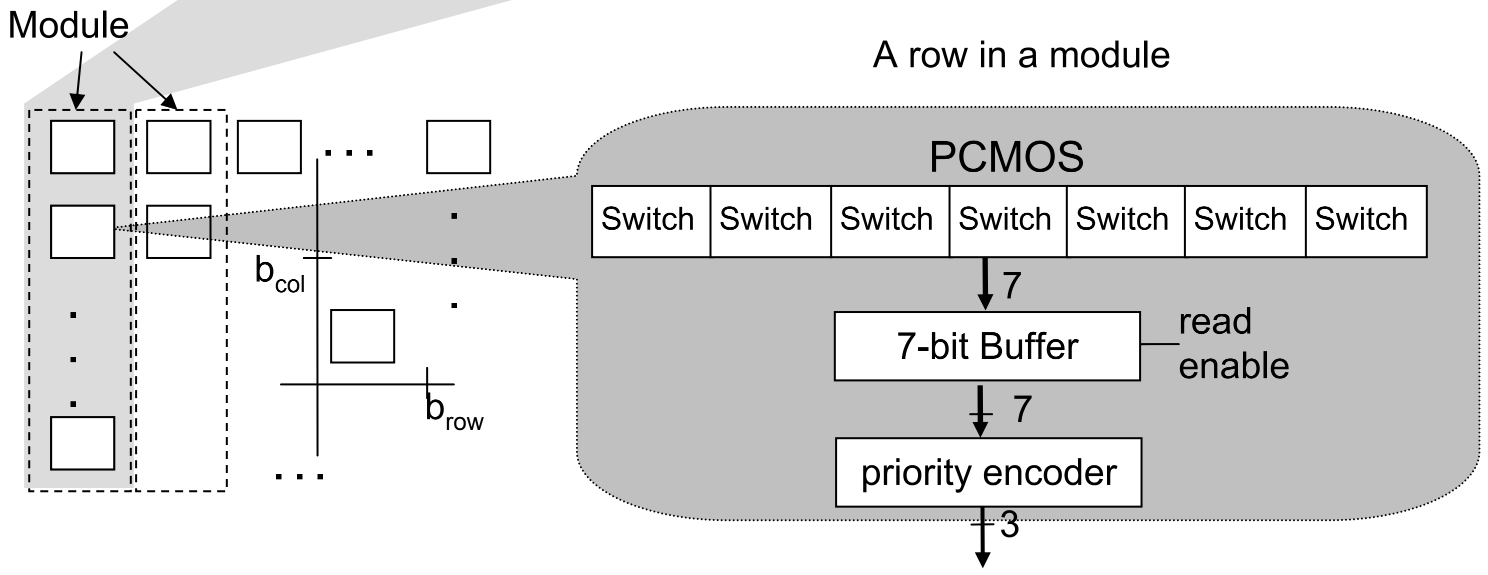


An Example PSOC Design

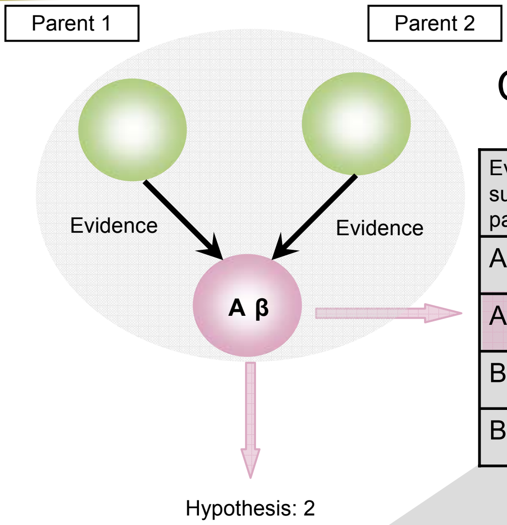
Consider the core probabilistic step of Bayesian Network

Evidence supplied by parents	Hypothesis: 1	Hypothesis: 2
A and α	0.9	0.1
A and β	0.1	0.9
B and α	0.5	0.5
B and β	0.3	0.7

Evidence supplied by parents	Hypothesis: 1	Hypothesis: 2
A and α	0.9	0.1
A and β	0.1	0.9
B and α	0.5	0.5
B and β	0.3	0.7



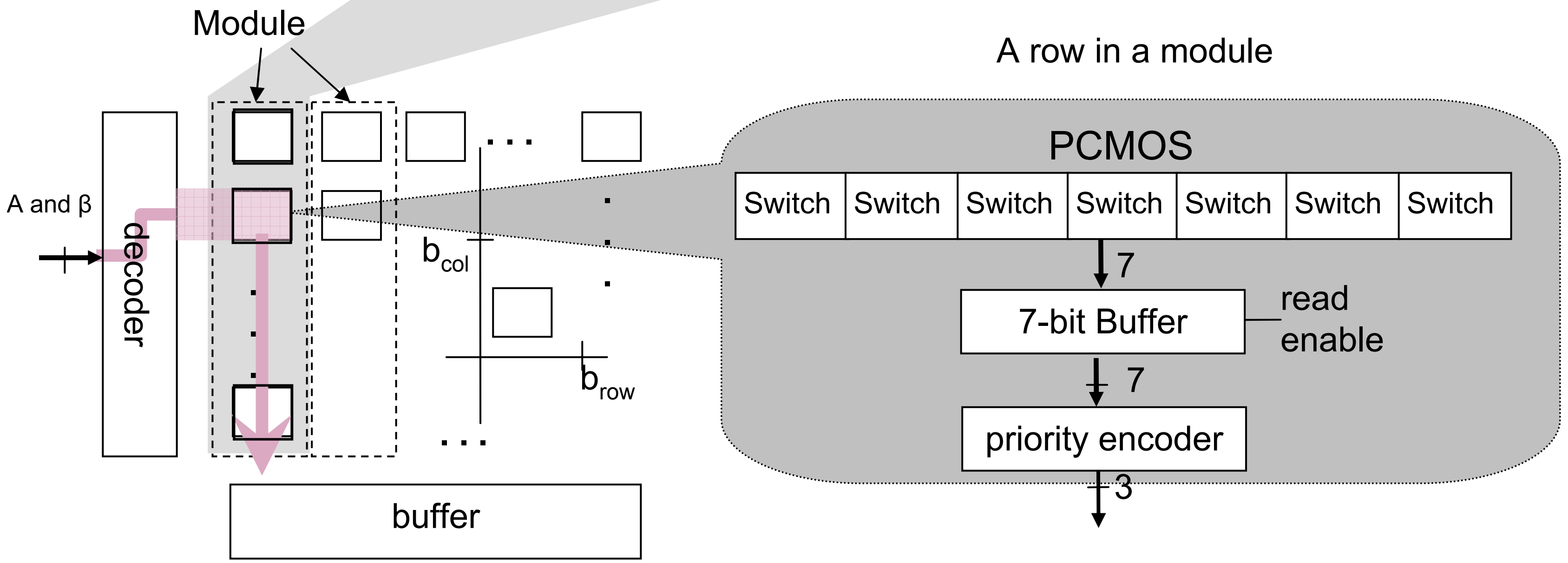
An Example PSOC Design



Consider the core probabilistic step of Bayesian Network

Evidence supplied by parents	Hypothesis: 1	Hypothesis: 2
A and α	0.9	0.1
A and β	0.1	0.9
B and α	0.5	0.5
B and β	0.3	0.7

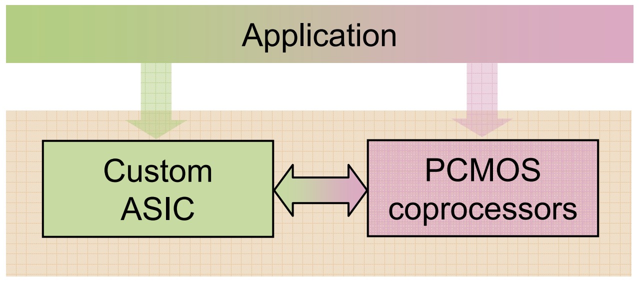
Evidence supplied by parents	Hypothesis: 1	Hypothesis: 2
A and α	0.9	0.1
A and β	0.1	0.9
B and α	0.5	0.5
B and β	0.3	0.7



Metrics

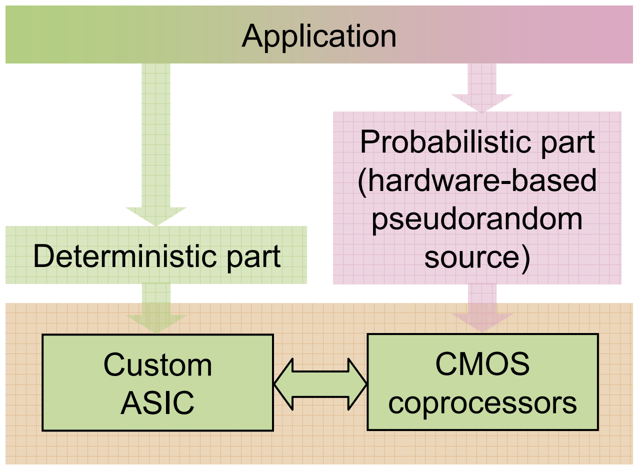
- EPP = Energy (Joules) x Performance (seconds)
  - We are interested in both energy and performance
  - Invariant under voltage scaling techniques
    - Energy decreases and time increases proportionally
- Architecture Gain (Baseline, I) = Γ_I = $EPP_{Baseline}$ / EPP_I
  - **I** is a particular choice of technology *implementation* and baseline is the host
  - Baseline is a CMOS based Custom design

EPP_I



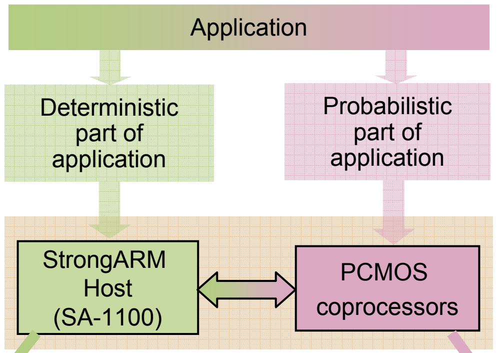
Probabilistic System-on-a-chip architecture with custom ASIC host

$EPP_{Baseline}$



Conventional System-on-a-chip with custom ASIC host

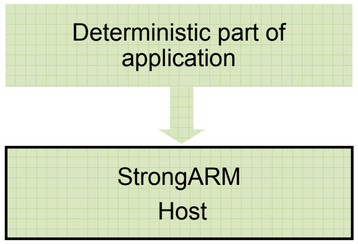
Summary of Results for Architecture *Gain*



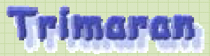
Application
Deterministic part of application
Probabilistic part of application
StrongARM Host (SA-1100)
PCMOS coprocessors

Probabilistic system-on-a-chip architecture

vs

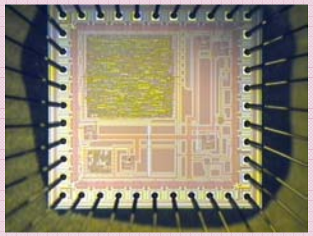


Deterministic algorithm on a host processor



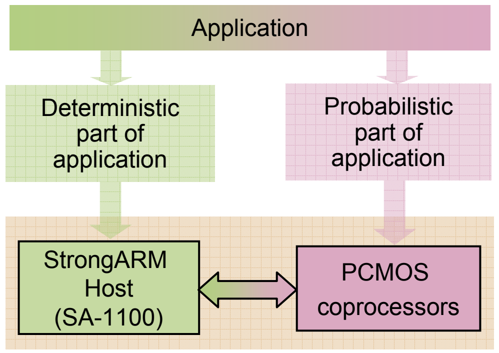
A. Sinha and A. Chandrakasan, “Jouletrack a web based tool for software energy profiling,” 38th DAC, pp. 220–225, 2001.

Chip measurements



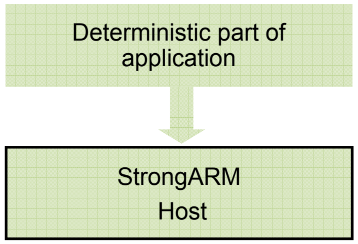
HSpice Simulations

Summary of Results for Architecture *Gain*


Application
Deterministic part of application
Probabilistic part of application
StrongARM Host (SA-1100)
PCMOS coprocessors

Probabilistic system-on-a-chip architecture

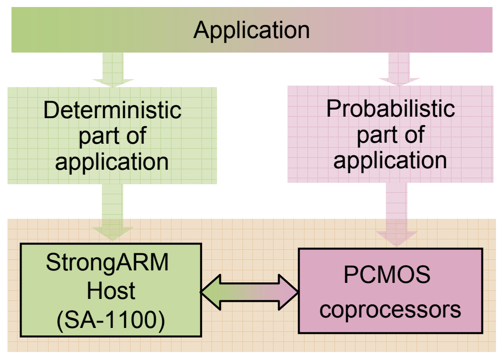
vs



Deterministic algorithm on a host processor

Algorithm	Applications	Min EPP Gain	Max EPP Gain
Bayesian	SPAM Filters, *Windows Printer Trouble Shooting*, Battlefield Planning	**12.5**	**291**

Summary of Results for Architecture *Gain*

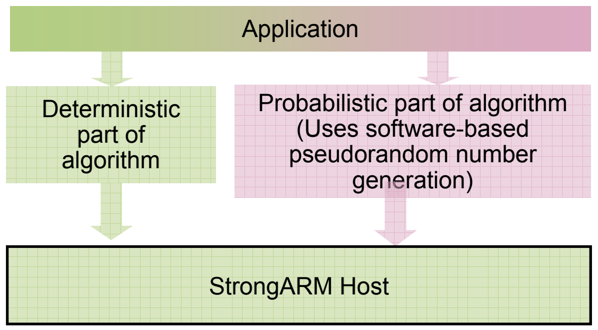


Application
Deterministic part of application
Probabilistic part of application
StrongARM Host (SA-1100)
PCMOS coprocessors

Probabilistic system-on-a-chip architecture

vs

Implementation Alternatives (providing *pseudo*random bits)



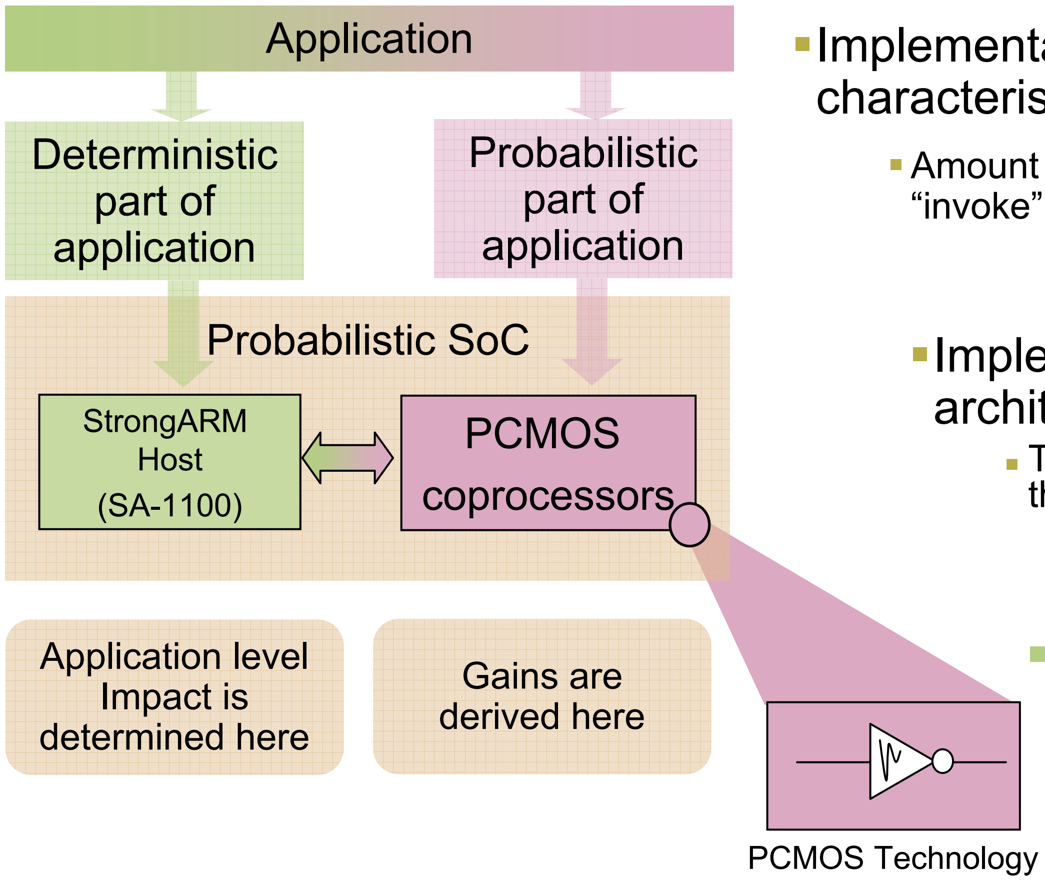
Application
Deterministic part of algorithm
Probabilistic part of algorithm (Uses software-based pseudorandom number generation)
StrongARM Host

Probabilistic algorithm on a host processor

Algorithm	Applications	Min EPP Gain	Max EPP Gain
Cellular Automata	Pattern generation and classification *(String classification)*	83	110
Randomized Neural Network	Image and pattern classification, *Optimization of NP-hard problems (vertex cover)*	226.5	300
Hyper-Encryption	*Security applications*	1.06	1.06
Bayesian Network	*Hospital patient management*	3	7.5

Outline

Addressing Chief Concerns – Cost-Benefit Analysis of Randomization



Application
Deterministic part of application
Probabilistic part of application
Probabilistic SoC
StrongARM Host (SA-1100)
PCMOS coprocessors
Application level Impact is determined here
Gains are derived here
PCMOS Technology

- Implementation independent algorithmic characteristics
  - Amount of opportunity in the algorithm to “invoke” PCMOS based primitives
- Implementation dependent architecture characteristics
  - The “fit” of the architecture primitives to the application primitives
- Implementation dependent technology characteristics
  - Energy and performance efficiency of PCMOS

- Amount of opportunity in an application is captured through Probabilistic Flux F
  - Flux is the ratio of the number of core probabilistic steps to the total number of operations during its execution.

CREST at GEORGIA TECH

DARPA

Addressing Chief Concerns

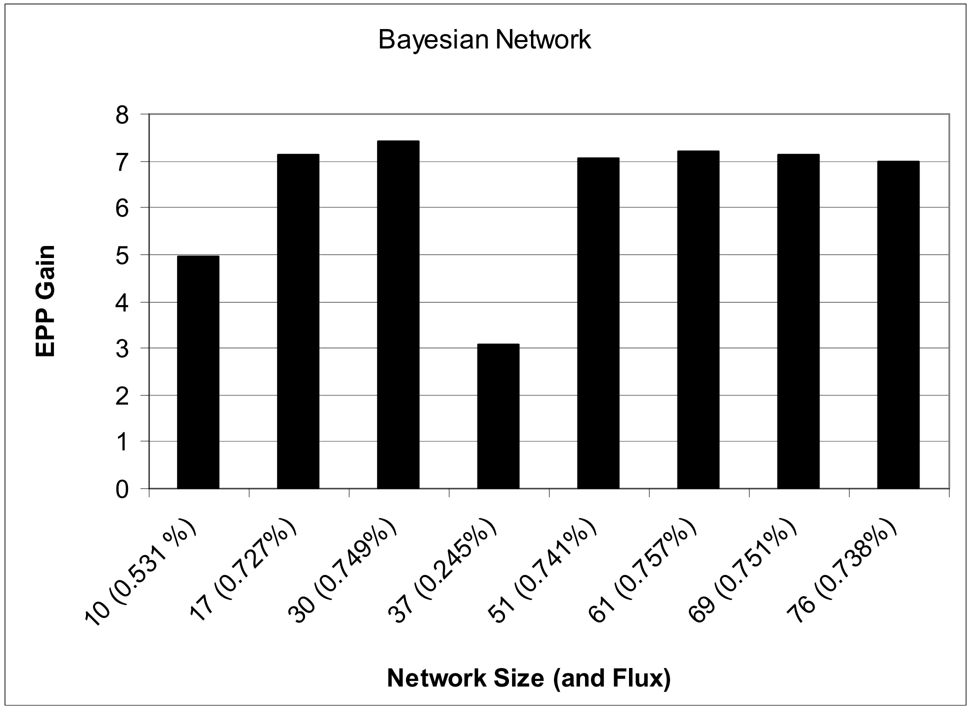
$$\begin{tikzcd}[column sep=small]
\text{Recall: } \Gamma_I = \text{EPP}_{\text{Baseline}} / \text{EPP}_I \arrow[d] & \\
\boxed{\text{Energy}_{\text{Baseline}} \times \text{Time}_{\text{Baseline}}} \arrow[d] & \\
\boxed{\text{Energy}_{(\text{det,Baseline})} + \text{Energy}_{(\text{prob,Baseline})}} \arrow[d] \arrow[dr] & \\
\boxed{\text{Number of times deterministic part “invokes” the host} \times \text{Energy per invocation}} & + \;\boxed{\text{Number of times probabilistic part is “invoked”} \times \text{Energy per invocation}}
\end{tikzcd}$$

$$\boxed{\text{Cycles}_{(\text{det,Baseline})} \times \text{Energy}_{(\text{cycle,Host})}} + \boxed{\text{Flux} \times \text{Cycles}_{(\text{det,Baseline})} \times \text{Energy}_{(\text{Flux,Baseline})}}$$

Using similar approach for Time and Approximating

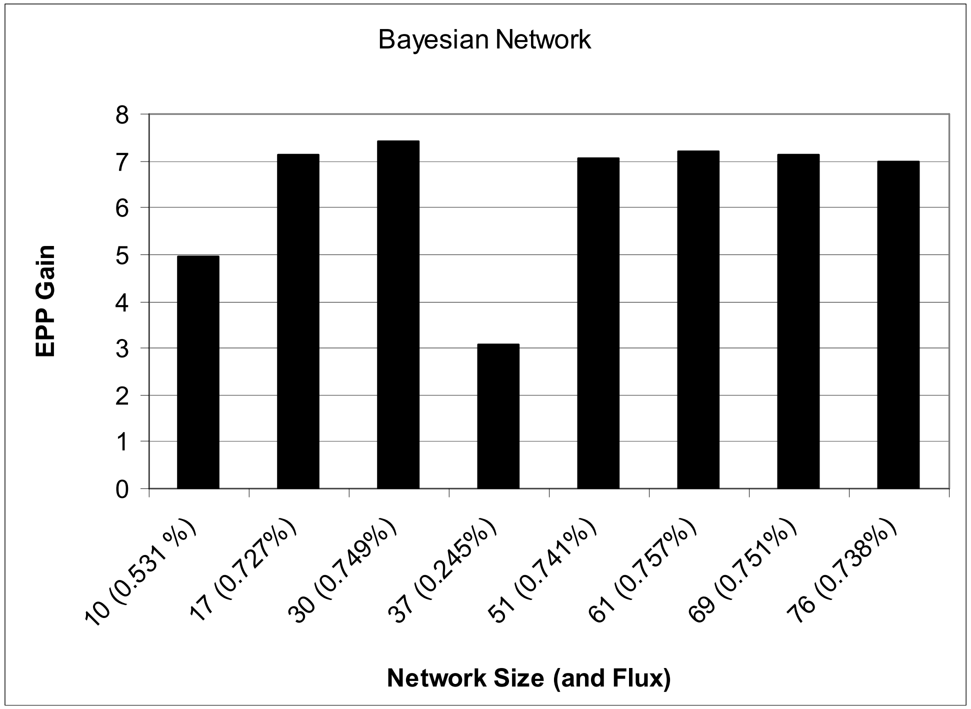
$$\Gamma_I \cong \left(1 + \frac{\text{Flux} \times \text{Energy}_{(\text{Flux,Baseline})}}{\text{Energy}_{(\text{cycle,Host})}}\right)\left(1 + \frac{\text{Flux} \times \text{Time}_{(\text{Flux,Baseline})}}{\text{Time}_{(\text{cycle,Host})}}\right)$$

Implementation Independent *Algorithmic* Characteristics



$$\Gamma_I = \left(1 + \frac{\text{Flux} \times \text{Energy}_{(\text{Flux,Baseline})}}{\text{Energy}_{(\text{cycle,Host})}}\right)\left(1 + \frac{\text{Flux} \times \text{Time}_{(\text{Flux,Baseline})}}{\text{Time}_{(\text{cycle,Host})}}\right)$$

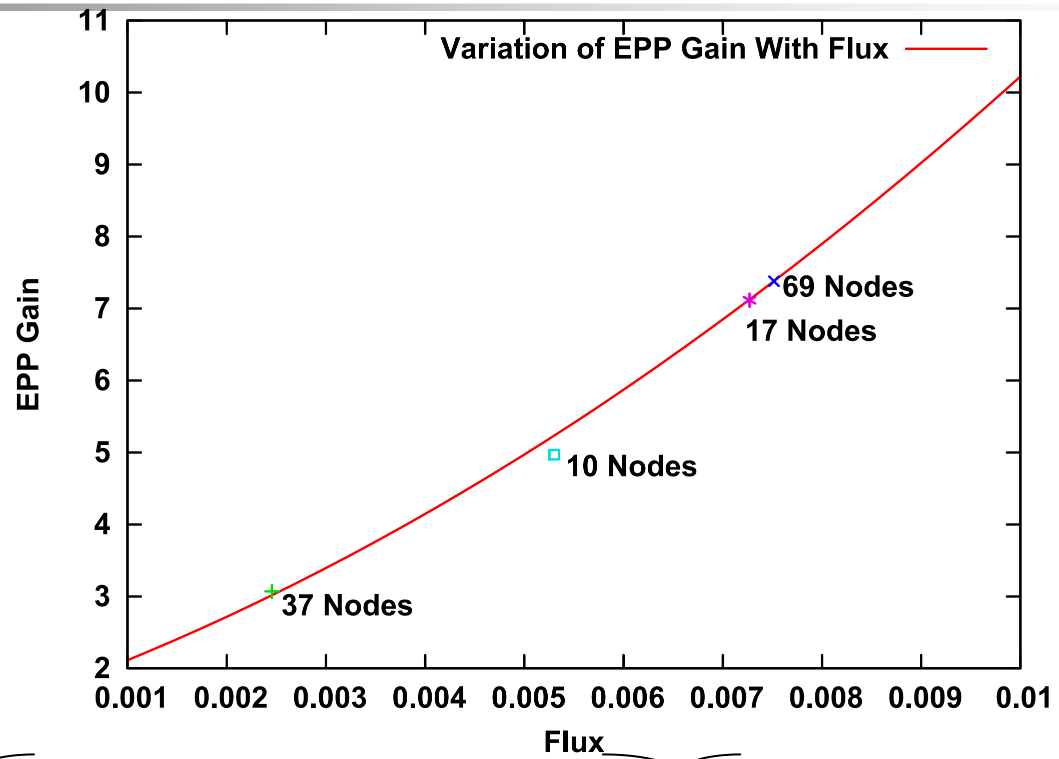
Implementation Independent *Algorithmic* Characteristics



$$\Gamma_I = \left(1 + \frac{\text{Flux} \times \cancel{\text{Energy}_{(\text{Flux,Baseline})}}}{\cancel{\text{Energy}_{(\text{cycle,Host})}}}\right)\left(1 + \frac{\text{Flux} \times \cancel{\text{Time}_{(\text{Flux,Baseline})}}}{\cancel{\text{Time}_{(\text{cycle,Host})}}}\right)$$

- Energy consumed for implementing the “core probabilistic step”, energy efficiency of the (deterministic) host processor are held invariant

Implementation Independent *Algorithmic* Characteristics

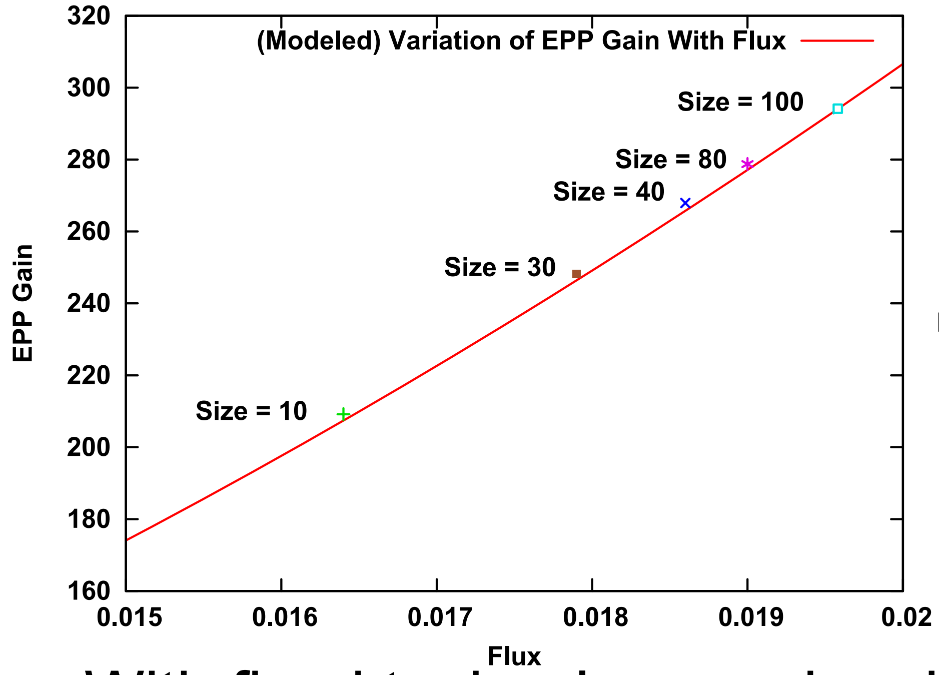


$$\Gamma_I = \left(1 + \frac{\text{Flux} \times \cancel{\text{Energy}_{(\text{Flux,Baseline})}}}{\cancel{\text{Energy}_{(\text{cycle,Host})}}}\right)\left(1 + \frac{\text{Flux} \times \cancel{\text{Time}_{(\text{Flux,Baseline})}}}{\cancel{\text{Time}_{(\text{cycle,Host})}}}\right)$$

- Energy consumed for implementing the “core probabilistic step”, energy efficiency of the (deterministic) host processor are held invariant

Implementation Independent *Algorithmic* Characteristics

- Similar trend can be demonstrated for other applications as well
    - Consider the randomized neural network application

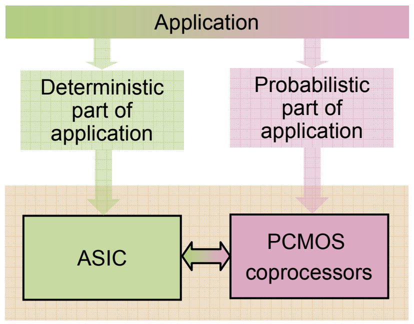


$$\Gamma_I \cong \left[1 + \frac{\text{Flux} \times \cancel{\text{Energy}}_{(\text{Flux,Baseline})}}{\cancel{\text{Energy}}_{(\text{cycle,Host})}}\right]\left[1 + \frac{\text{Flux} \times \cancel{\text{Time}}_{(\text{Flux,Baseline})}}{\cancel{\text{Time}}_{(\text{cycle,Host})}}\right]$$

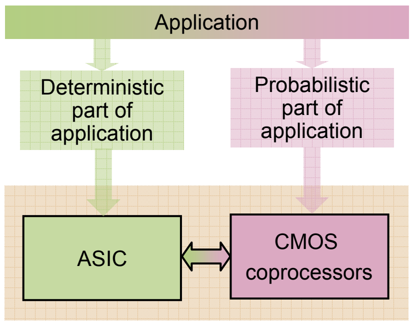
- With fixed technology and architecture parameters gains increase with Flux
    - Recall - Flux quantifies “amount of opportunity”

Insights and optimizations

- Lessons learned from analysis of gains
  - Algorithm
    - Increase opportunity (increase flux)
  - Architecture and Technology
    - Increase gains due to PCMOS
    - Create efficient architectures that leverage PCMOS
  - Increase the efficiency of the host processor
    - Design Custom-ASIC host processor


Application
Deterministic part of application
Probabilistic part of application
ASIC
PCMOS coprocessors

vs


Application
Deterministic part of application
Probabilistic part of application
ASIC
CMOS coprocessors

Algorithm	EPP Gain (before optimization – StrongARM Host)	EPP Gain (after optimization – ASIC Host)
Hyper Encryption	1	9.48
Probabilistic Cellular Automata	1.06	561

Addressing Chief Concerns - Quality of Randomization

- Impact of probabilistic bits on application level ***quality of solution*** is important
  - How "good" (how "random") is PCMOS ?
  - What is "random" ?
  - *Work-in-progress to test quality-sensitive applications*

- Quality of randomness tests from National Institute of Standards and Technology (NIST)
  - Compare and evaluate the random sequences generated by PCMOS through measurements
  - *Demonstrate application-level impact*

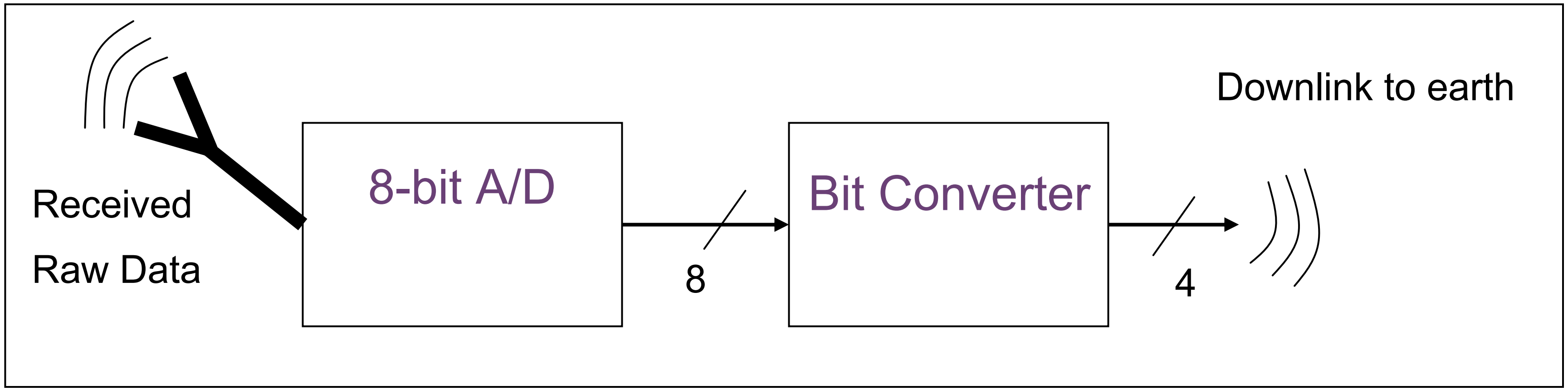
Test	PCMOS	PRNG
Frequency	Pass (0.98)	Fail (0.84)
Block-frequency	Pass (1.00)	Pass (0.98)
Cumulative sum	Pass (0.98)	Fail (0.86)
Runs	Pass (0.98)	Pass (0.96)
FFT	Pass (1.00)	Pass (1.00)
Approximate entropy	Pass (0.98)	Fail (0.92)
Long-run	Pass (1.00)	Pass (1.00)
Rank	Pass (1.00)	Fail (0.00)
Non-overlapping template	Pass (0.9375)	Pass (0.9375)
Overlapping template	Fail (0.8889)	Fail (0.00)
Lempel-Ziv	Fail (0.8125)	Fail (0.0625)
Linear complexity	Pass (1.00)	Pass (1.00)
Universal Statistical	Fail (0.725)	Fail (0.8889)
Serial	Pass (1.00)	Pass (1.00)

(result > 0.93) → Pass   (result < 0.93) → Fail

Outline

Role of Probability in Cognitive Applications
Current Implementation Methodologies and Chief Concerns
Exploiting Technology Trends – *Probabilistic* CMOS Technology
Probabilistic System on a Chip (PSOC) Architectures
Optimizing Probabilistic System on a Chip (PSOC) Architectures
PSOC Architectures for Conventional Signal Processing Applications
Next Steps

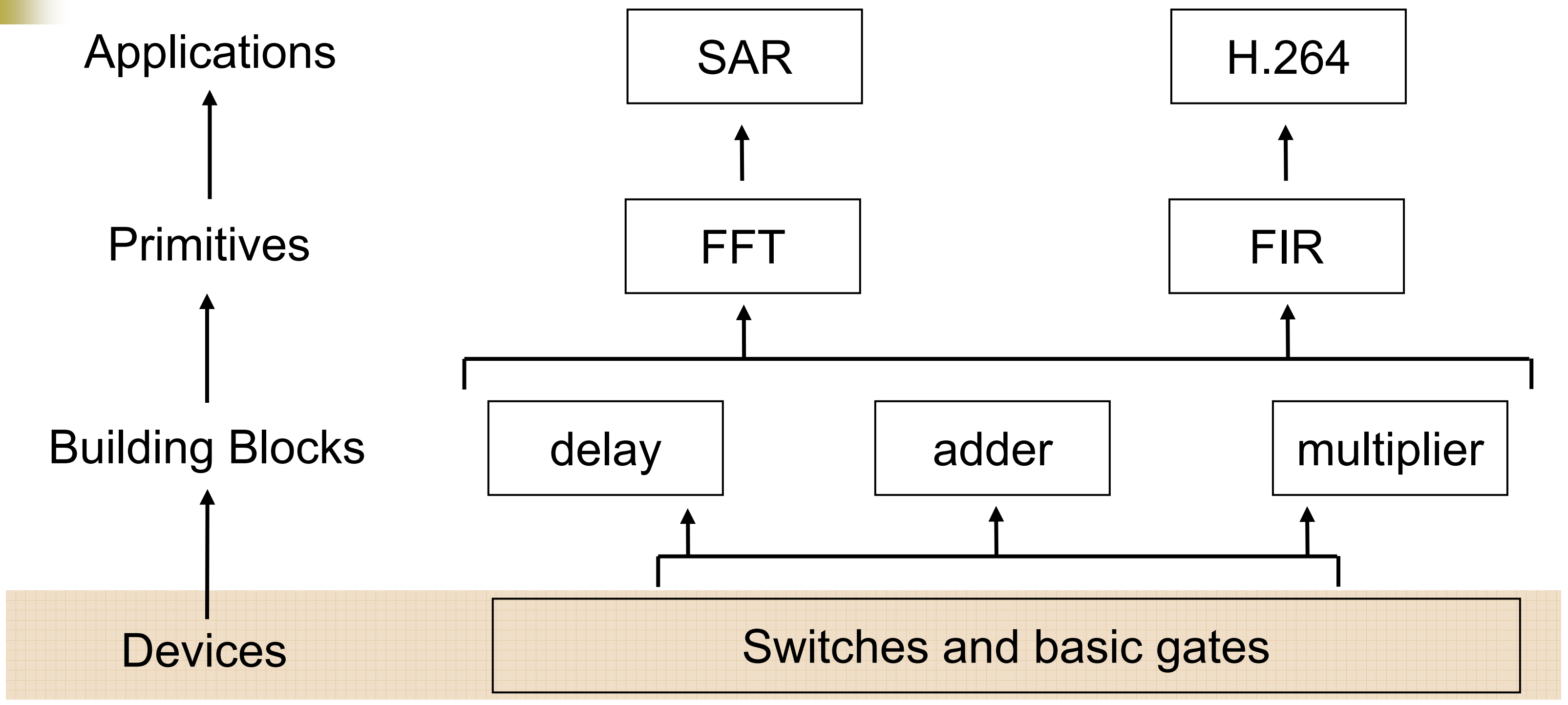
A/D Conversion of Raw Data



- Bit converter is used to compress data to send to earth because of the limited bandwidth and power constraints
- 8 bit A/D and Compression of data create *Quantization Noise*[1]
- Fewer bits used for A/D and Compression, more *Quantization Noise*
- However as long as *Quantization Noise* is less than noise created by PCMOS, quantization noise will never be seen
- Conclusion: With PCMOS we can use a more efficient A/D

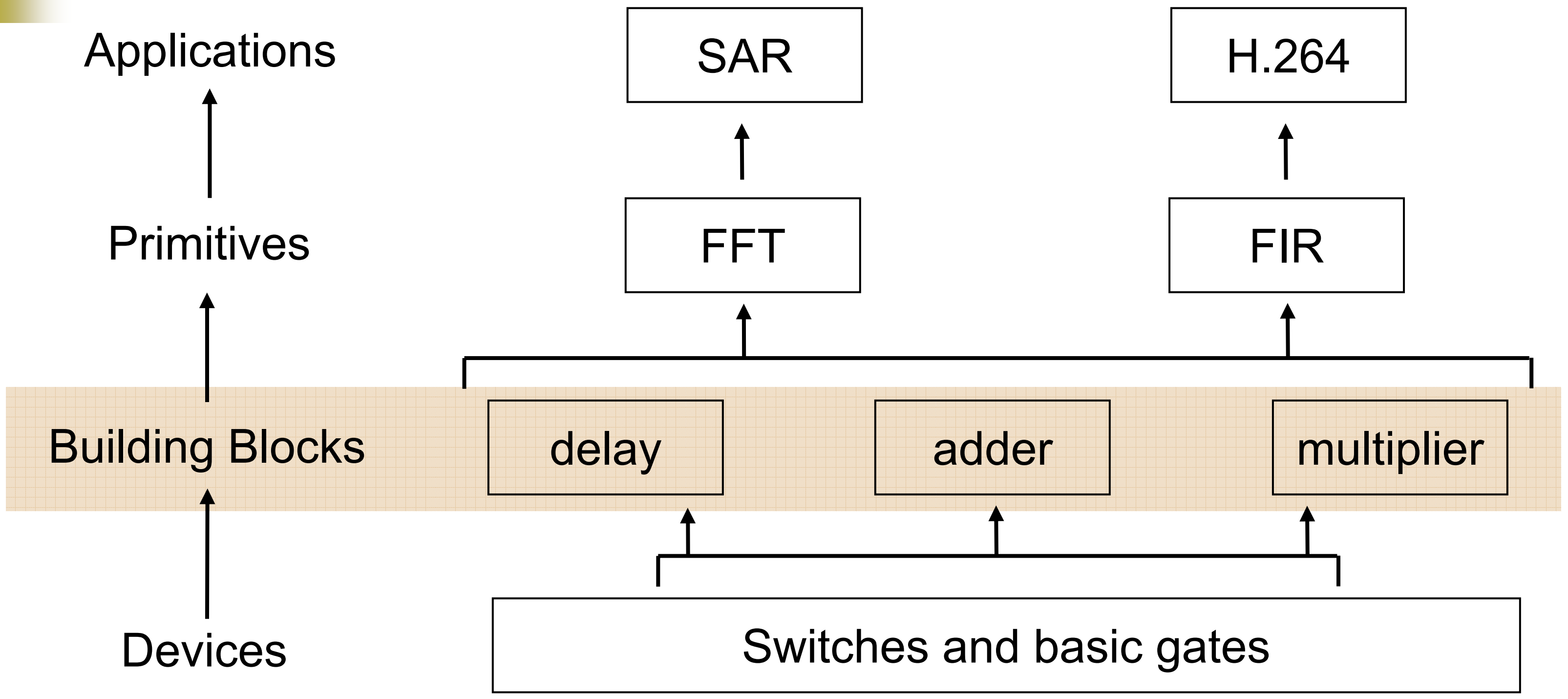
R. Kwok and W. T. Johnson. "Block Adaptive Quantization of Magellan SAR Data". *IEEE Transactions on Geoscience and Remote Sensing,* vol 27, July, 1989.

PCMOS Building Blocks for DSP



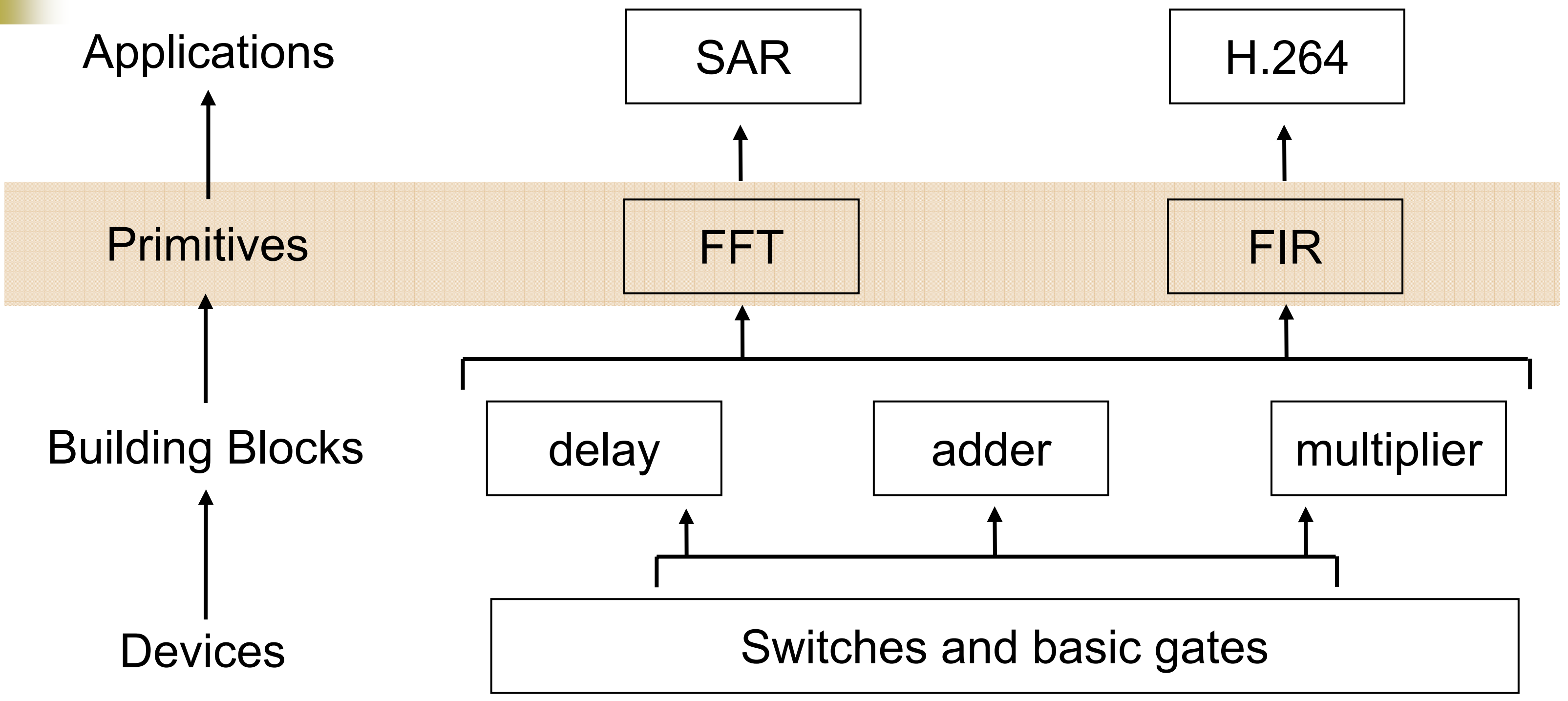
- What are the implications of error?
  - Not all applications require deterministic behavior
  - Degradation

PCMOS Building Blocks for DSP



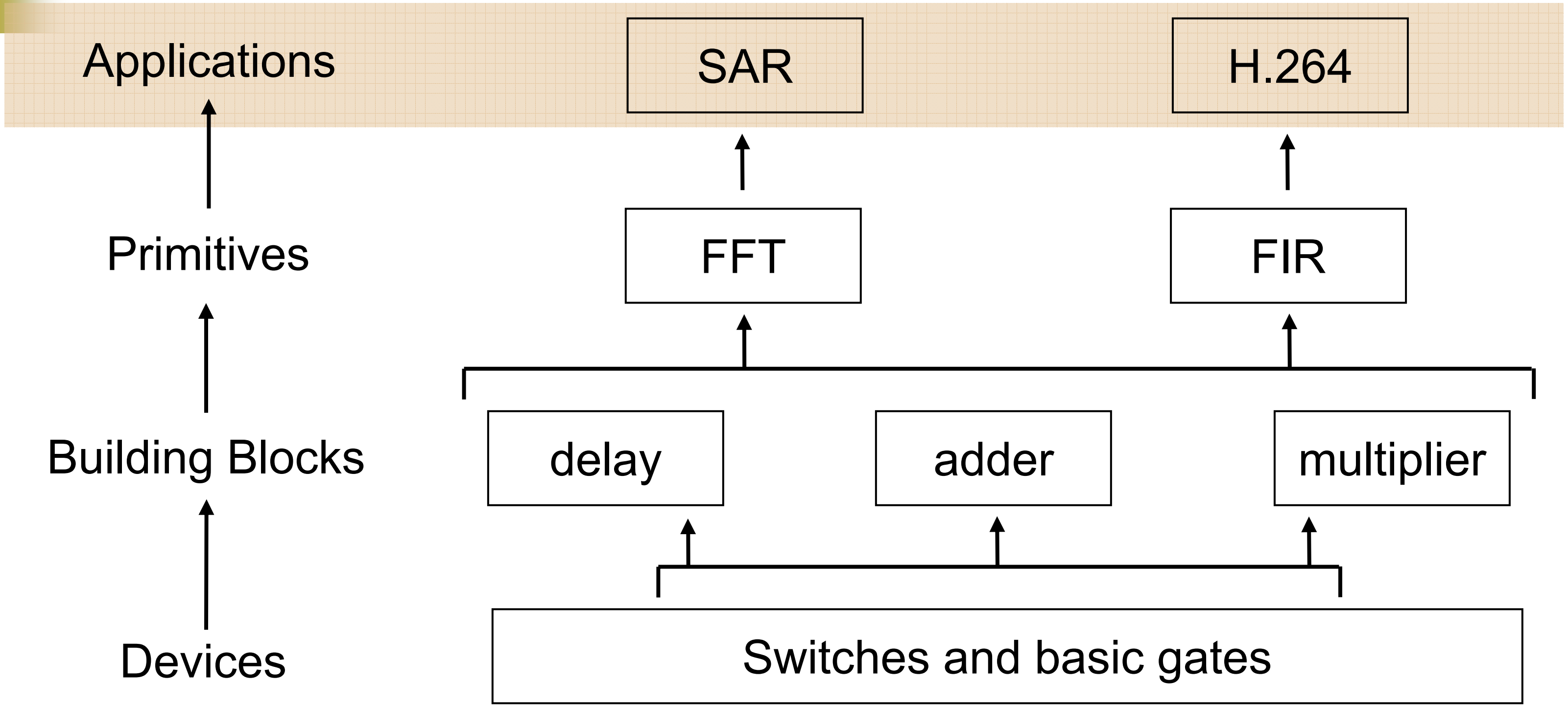
- What are the implications of error?
  - Not all applications require deterministic behavior
  - Degradation
    - Error rate or probability *p*

PCMOS Building Blocks for DSP



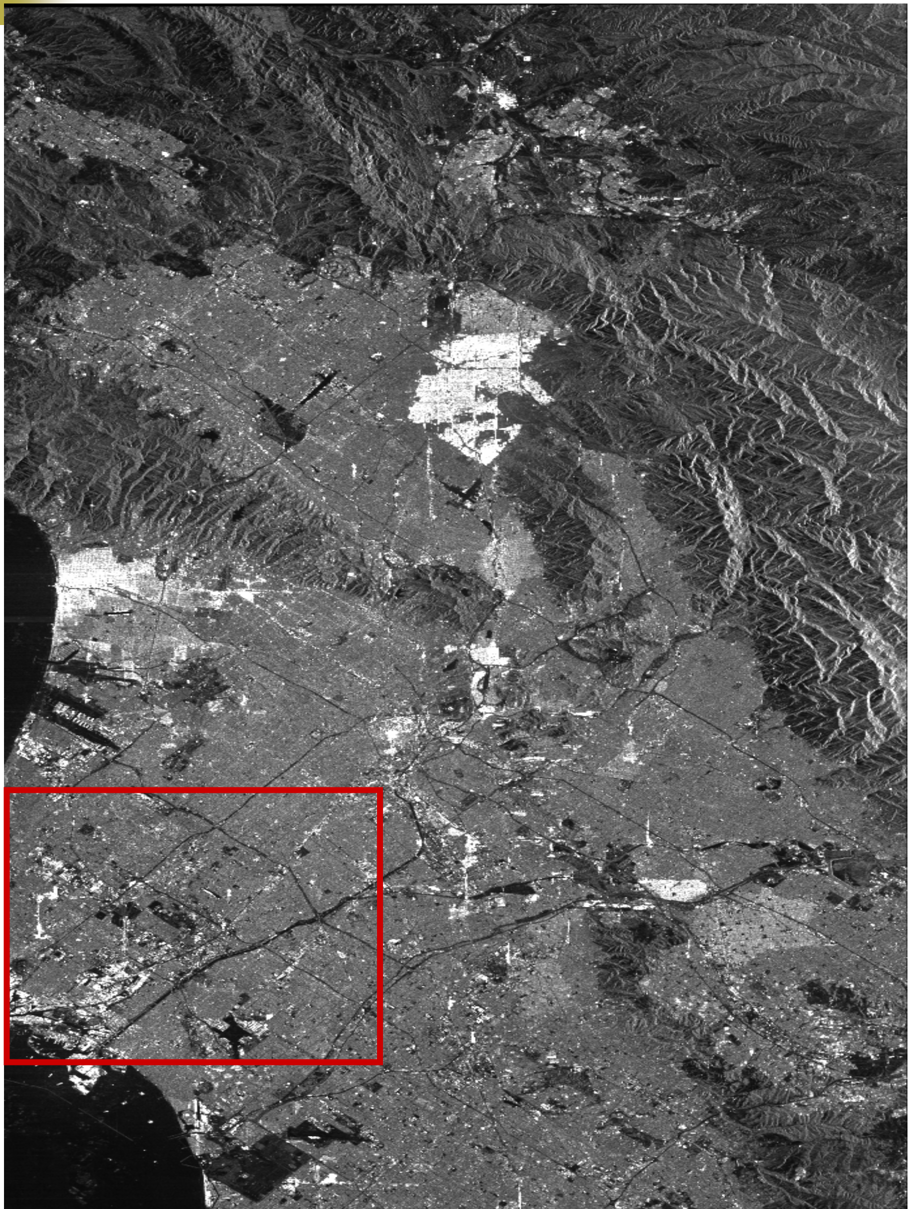
- What are the implications of error?
  - Not all applications require deterministic behavior
  - Degradation
    - Error rate or probability *p*
    - Signal to noise ratio (SNR)

PCMOS Building Blocks for DSP



- What are the implications of error?
  - Not all applications require deterministic behavior
  - Degradation
    - Error rate or probability *p*
    - Signal to noise ratio (SNR)
    - Image distortion

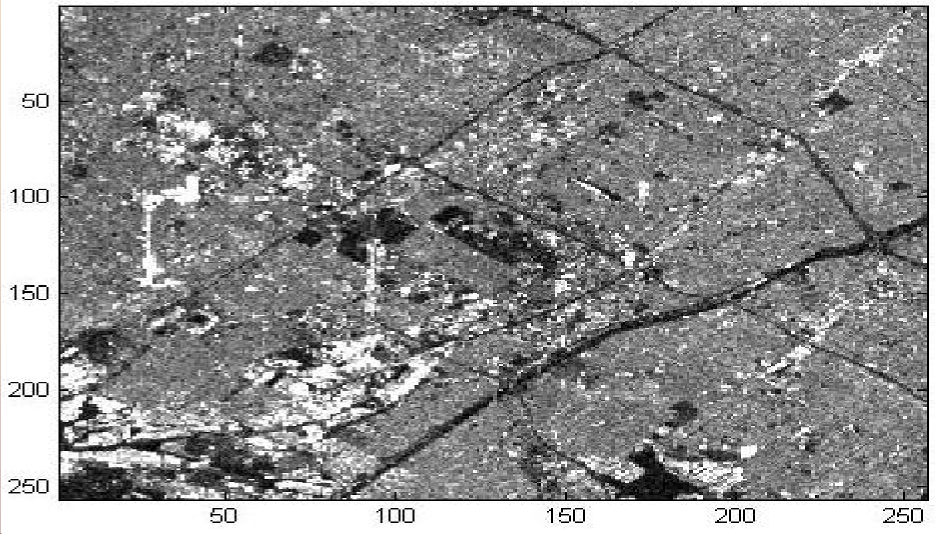
Original Image



- Original SAR image taken of Los Angeles area.
- Simulation assumes raw data has already been converted to digital and is ready for 32-bit processing
- Simulation assumes *either:*
  - A future technology generation where noise is comparable to supply voltage *OR*
  - Propagation delay errors are present due to energy savings-performance trade-off

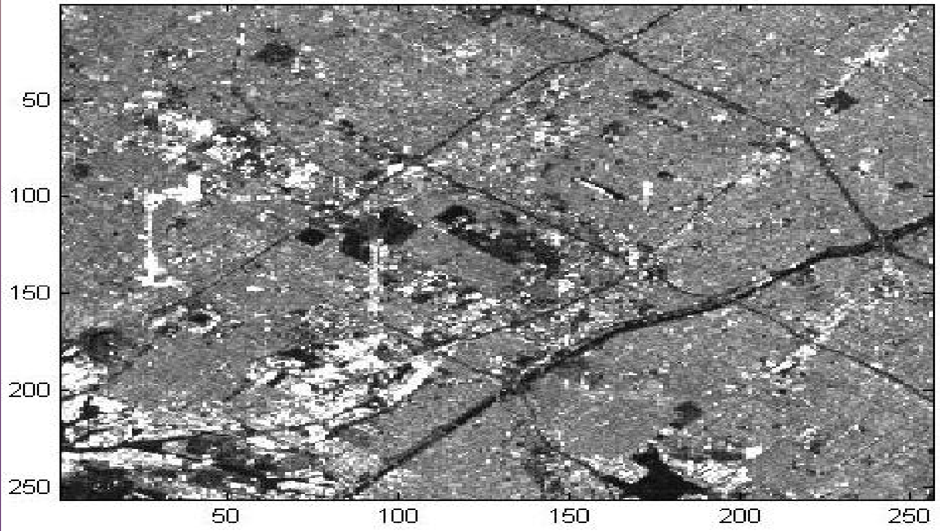
PCMOS Impact on SAR Imaging

PCMOS Result



- Radar image of Los Angeles using PCMOS SAR processor.
- 5.6X energy savings over current technology processor
- SNR = 28 dB

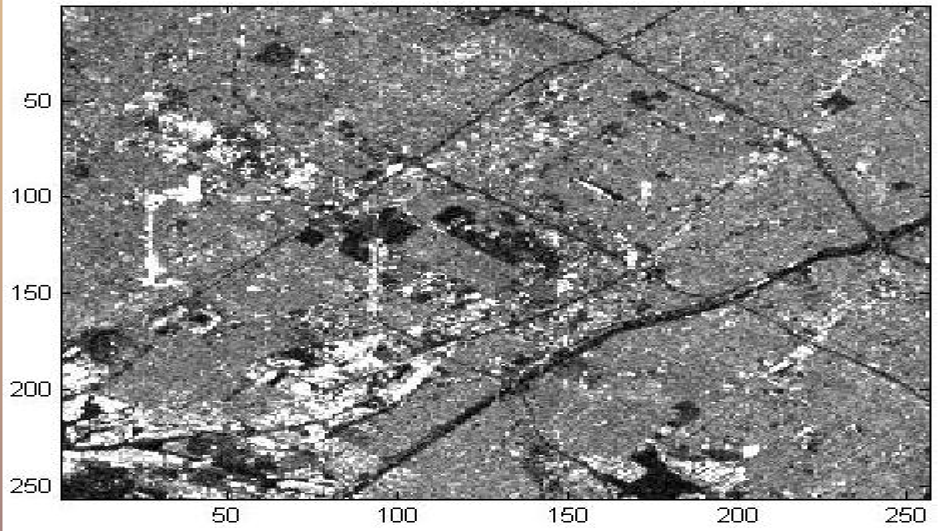
Current Technology Result



- Radar image of Los Angeles using current SAR processor
- SNR > 30 dB

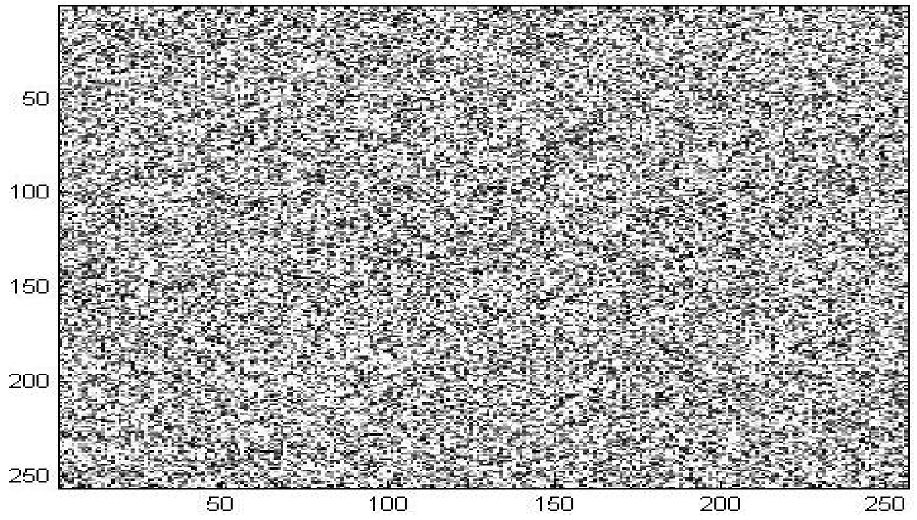
PCMOS Impact on SAR Imaging

PCMOS Result



- Radar image of Los Angeles using PCMOS SAR processor.
- 5.6X energy savings over current technology processor
- SNR = 28 dB

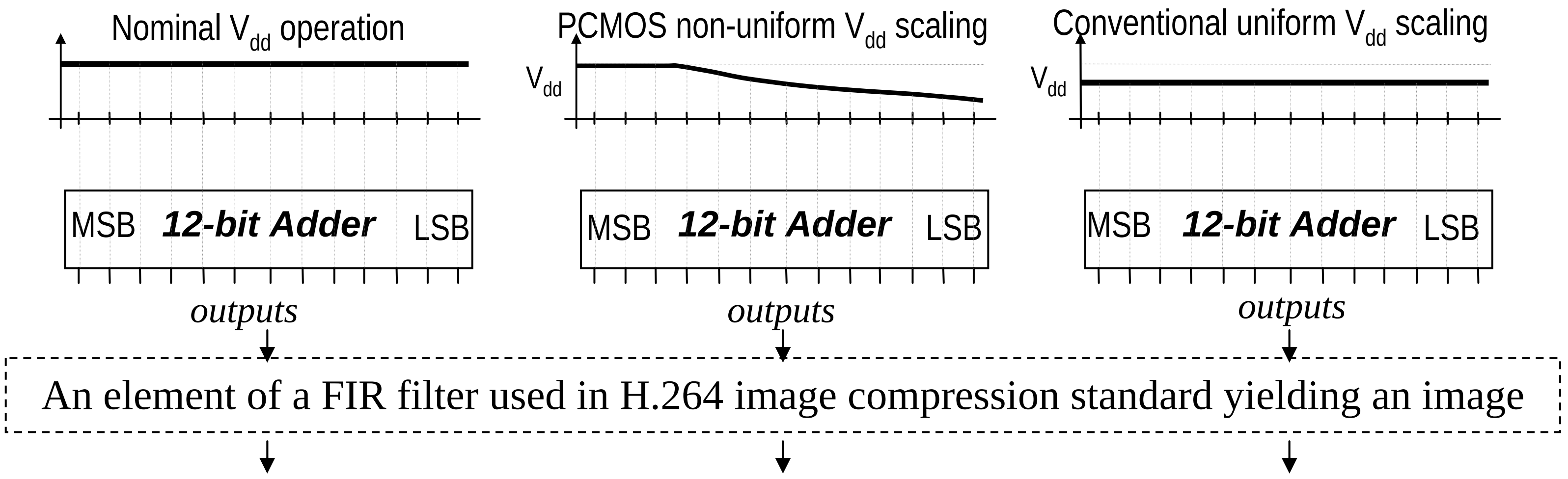
Conventional Voltage Scaling

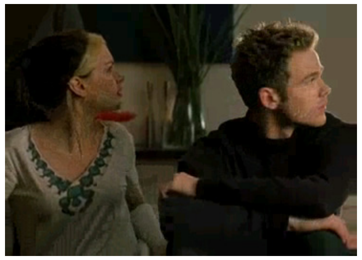


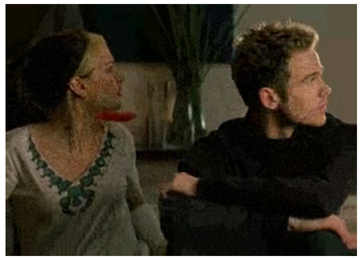
- Conventional voltage scaling technique used where supply voltage dropped uniformly
- 2.5X reduction in energy over current technology
- SNR = 0

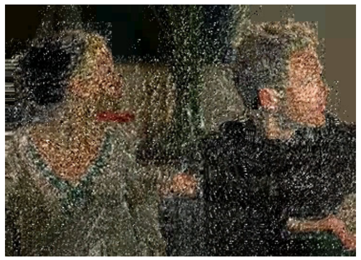
H.264 Image Decoding with PCMOS

- *FIR* sub-circuit of H.264 decoding implemented with PCMOS


Nominal V_{dd} operation
PCMOS non-uniform V_{dd} scaling
Conventional uniform V_{dd} scaling
V_{dd}
V_{dd}
MSB *12-bit Adder* LSB
MSB *12-bit Adder* LSB
MSB *12-bit Adder* LSB
outputs
outputs
outputs
An element of a FIR filter used in H.264 image compression standard yielding an image


Normal operation


Non-Uniform voltage scaling
1.66X Energy Savings


Conventional voltage scaling
1.69X Energy Savings

K. V. Palem, B. E. S. Akgul, and J. George. Variable scaling for computing elements. Invention Disclosure, Feb. 2006.

H.264 Experiment

FIR Type	SNR
1. Original	Approach $+\infty$
2. Geometric	60 dB
3. Linear	50 dB
4. Uniform	4 dB
5. Geometric	60 dB

Play Movie

FIR Type	Implementation
Original:	No scaling
Geometric:	Geometric scaling with *PCMOS*
Linear:	Linear scaling with *PCMOS*
Uniform:	Conventional (unbiased) scaling

Total energy spent by FIR filter = 4.25 nJ 3.5 nJ

- FIR filter is simulated in HSpice
  - FIR filter is used for interpolation in the H.264 decoder application
- Geometric V_{dd} scaling with PCMOS improves SNR-energy trade-off significantly
  - *Ideal for streaming to mobile devices*

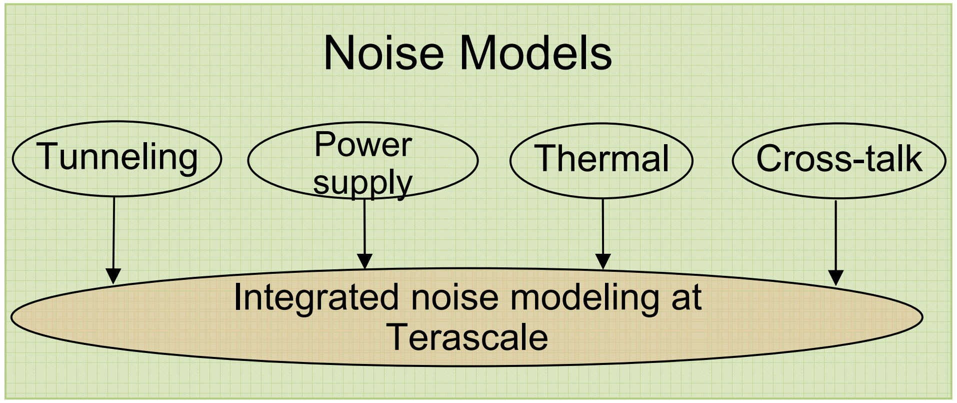
Outline

Next Steps: Programmable Co-Processors

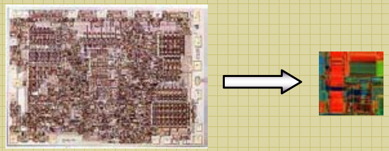
Inverter, Logic gates, Decoders, Multiplexers, DSP filters, etc.

PCMOS Primitive Libraries

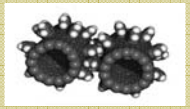
+



DSM & fabrication technologies

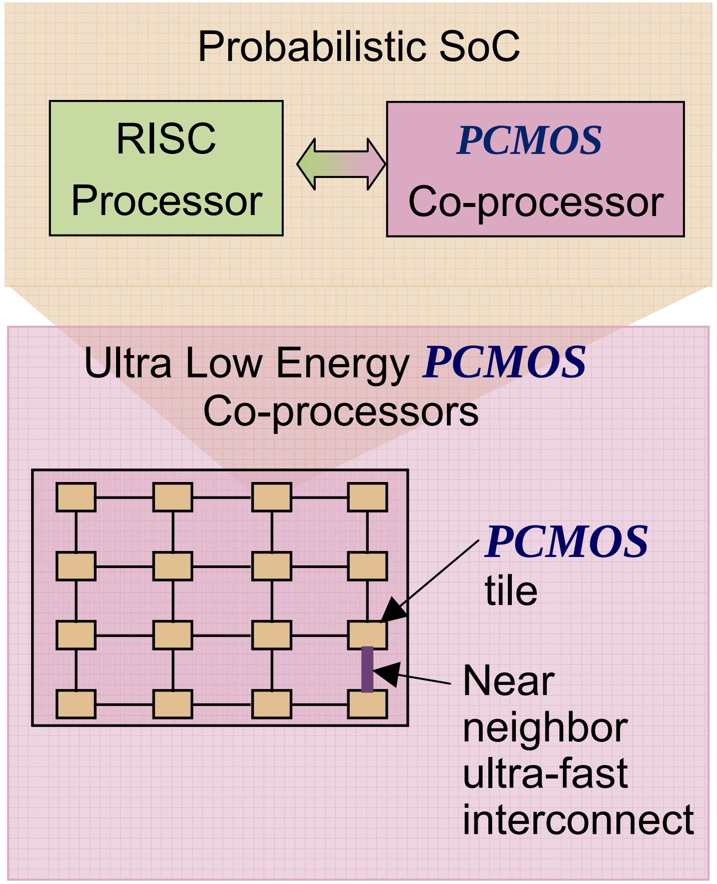


Increasing densities



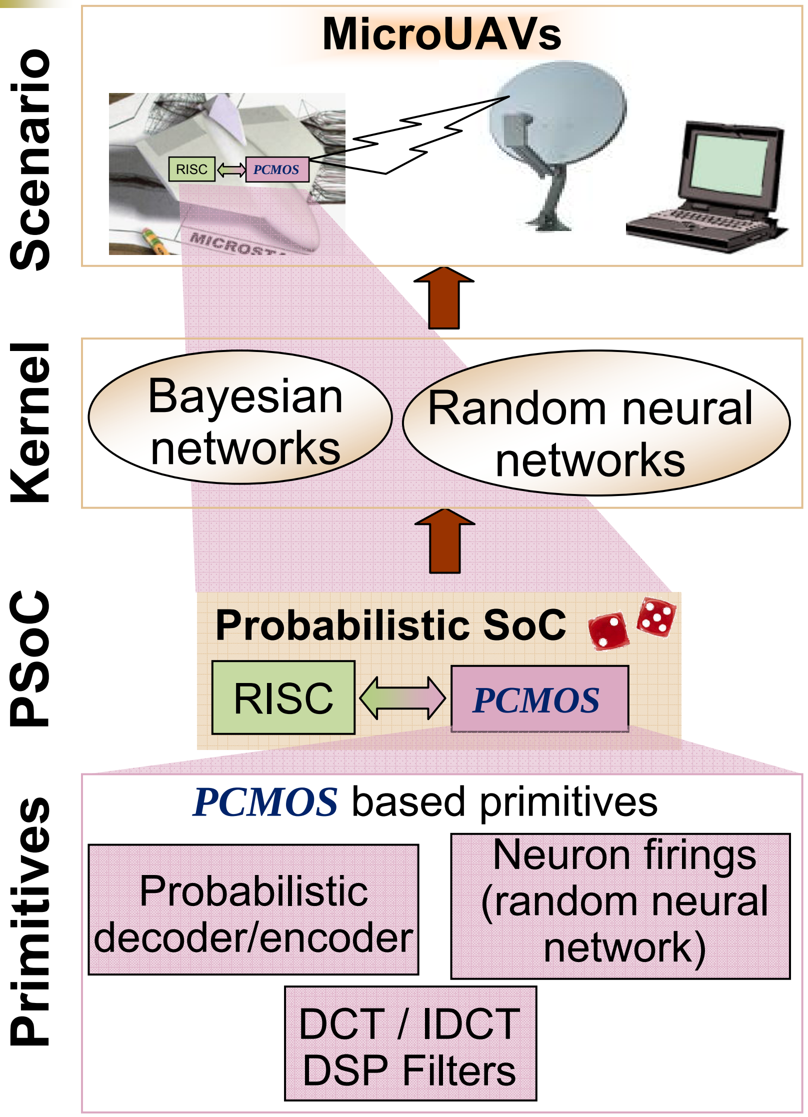
Nanotubes and other Molecular Technologies

=


Probabilistic SoC
RISC Processor
PCMOS Co-processor
Ultra Low Energy PCMOS Co-processors
PCMOS tile
Near neighbor ultra-fast interconnect

Programmable Co-processors for Cognitive Kernels

Next Steps: Application Impact



- Micro-UAVs
  - application demand
    - high-performance
    - real-time wireless communication and data-sharing
    - realize probabilistic algorithms such as pattern matching, inferencing, reasoning
  - need low energy consumption
  - benefit from error correction and redundancy mechanisms to
    - compensate device level errors
    - adjust *application quality*
- Orders of magnitude savings in *energy* x *performance* against conventional designs
  - extend battery life from hours to weeks