

---

# Silicon Kernel Learning “Machines”

---

**Gert Cauwenberghs**

Johns Hopkins University

[gert@jhu.edu](mailto:gert@jhu.edu)

520.776 Learning on Silicon

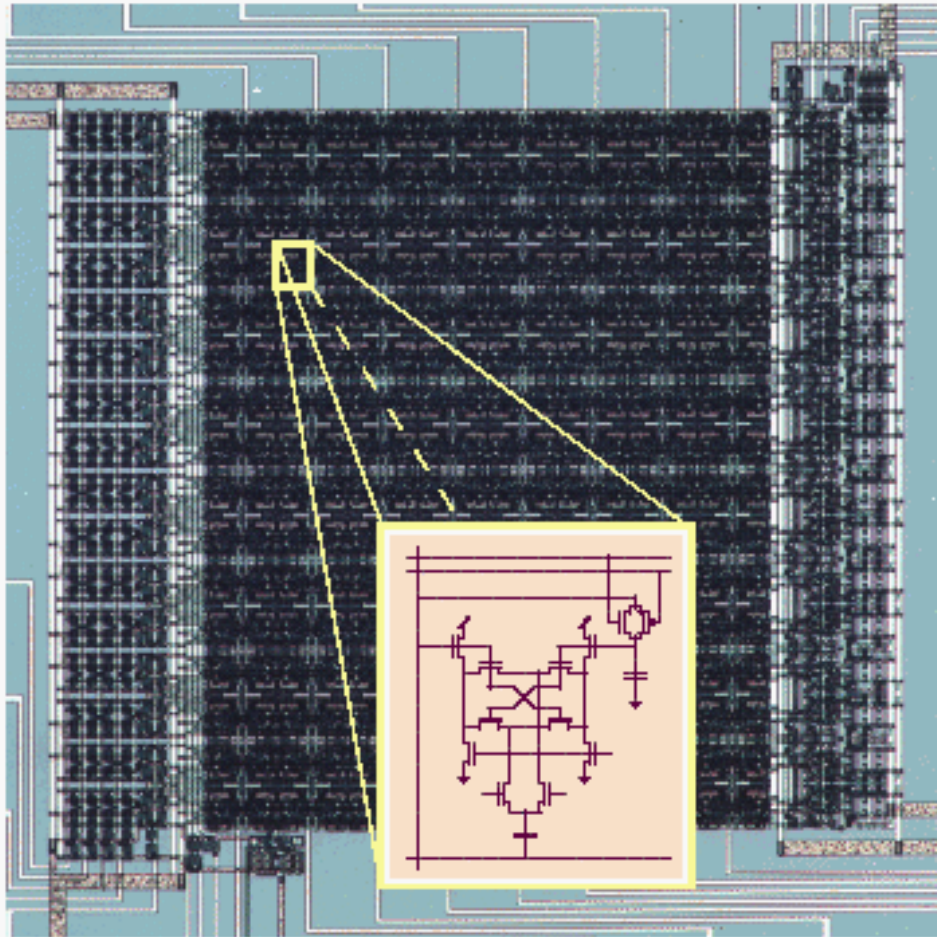
<http://bach.ece.jhu.edu/gert/courses/776>

# Silicon Kernel Learning “Machines”

## OUTLINE

- **Introduction**
  - *Kernel Machines* and array processing
  - Template-based pattern recognition
- **Kerneltron**
  - Support vector machines: learning and generalization
  - Modular vision systems
  - CID/DRAM internally analog, externally digital array processor
  - On-line SVM learning
- **Applications**
  - Example: real-time biosonar target identification

# Massively Parallel Array Kernel “Machines”

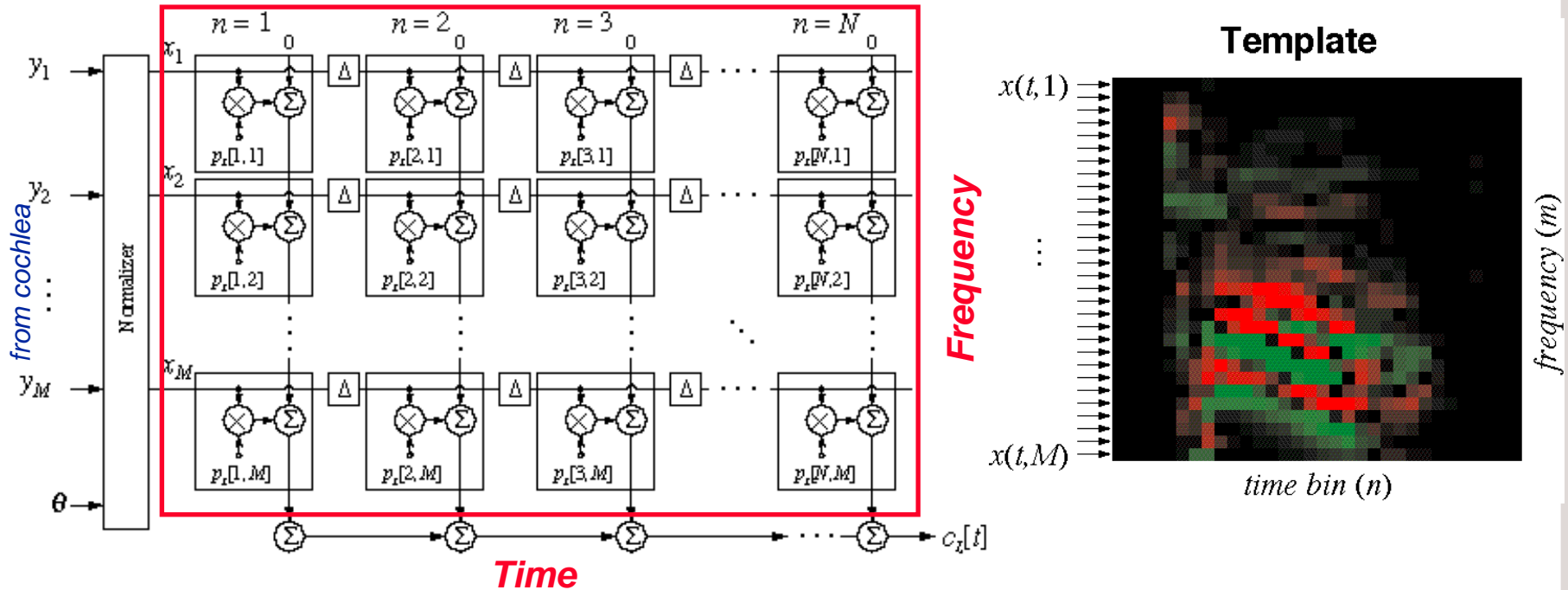


- **“Neuromorphic”**
  - distributed representation
  - local memory and adaptation
  - sensory interface
  - physical computation
  - internally analog, externally digital
- **Scalable**
  - throughput scales linearly with silicon area
- **Ultra Low-Power**
  - factor 100 to 10,000 less energy than CPU or DSP

*Example: VLSI Analog-to-digital vector quantizer  
(Cauwenberghs and Pedroni, 1997)*

# Acoustic Transient Processor (ATP)

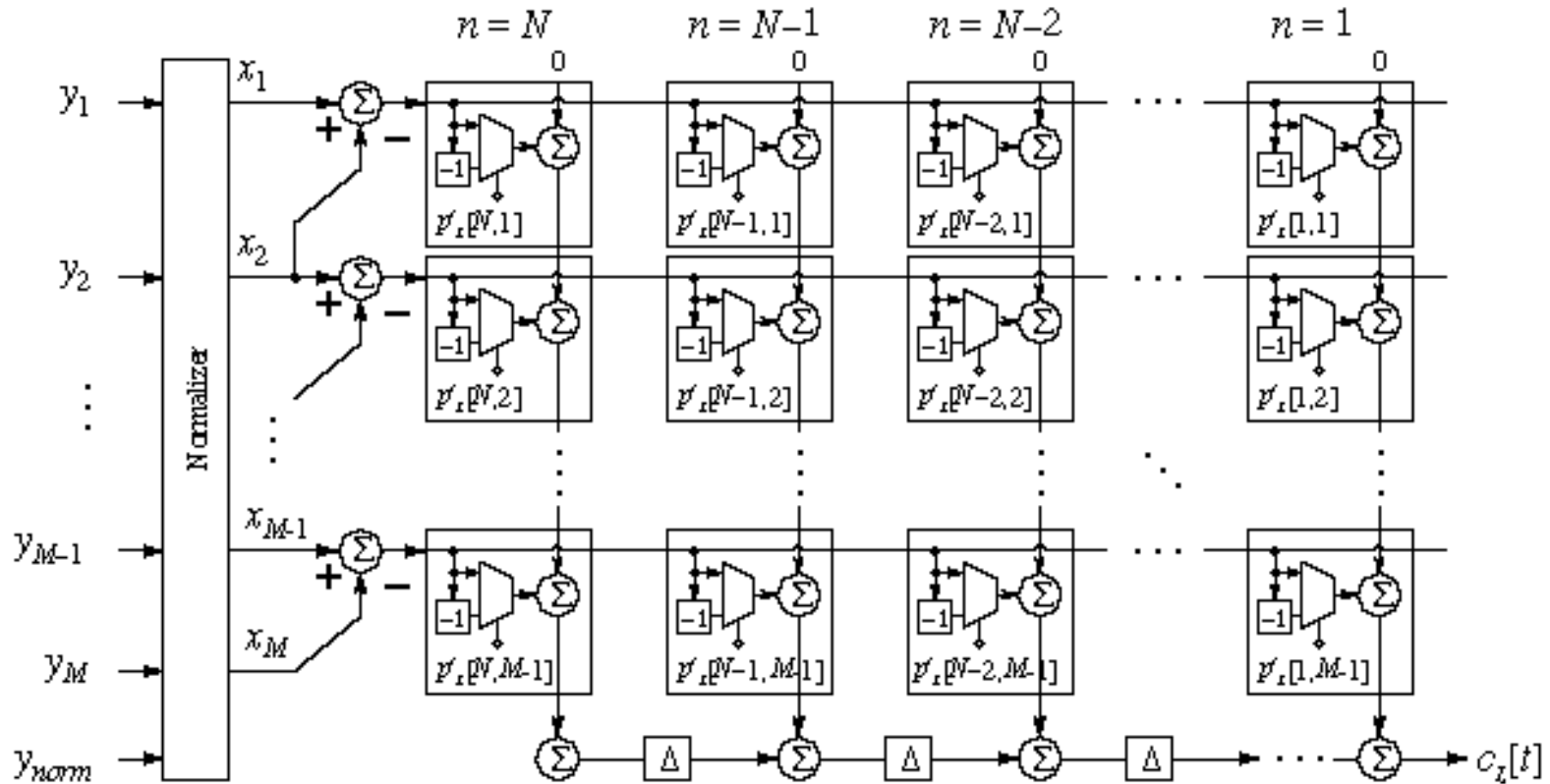
with Tim Edwards and Fernando Pineda



- Models time-frequency tuning of an auditory cortical cell (*S. Shamma*)
- Programmable template (matched filter) in time and frequency
- Operational primitives: *correlate, shift and accumulate*
- Algorithmic and architectural simplifications reduce complexity to one bit per cell, implemented essentially with a DRAM or SRAM at high density...

# Acoustic Transient Processor (ATP) Cont'd...

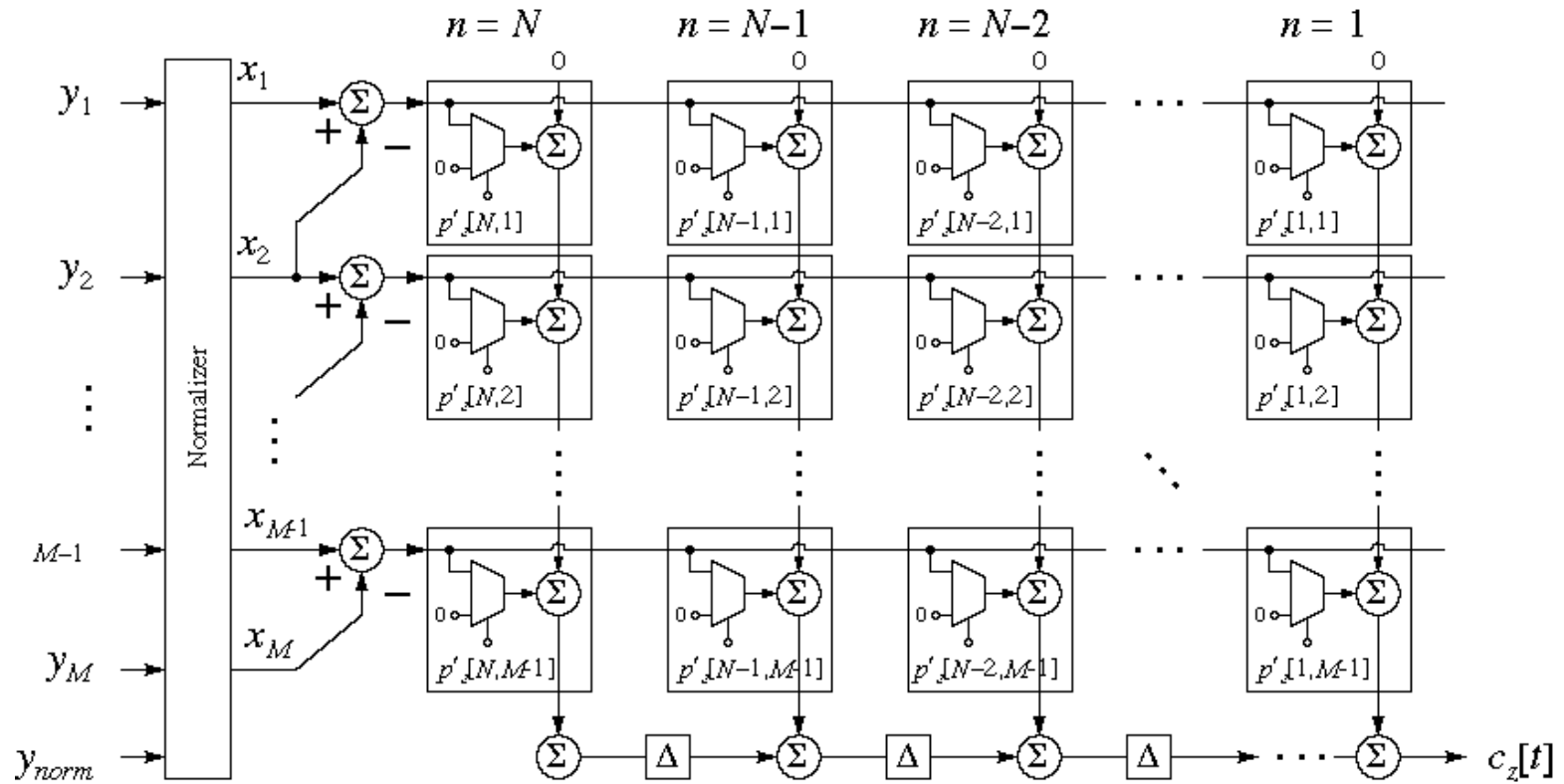
## Algorithmic and Architectural Simplifications (1)



- Channel differenced input, and binarized  $\{-1,+1\}$  template values, give essentially the same performance as infinite resolution templates.
- Correlate and shift operations commute, implemented with one shift register only.

# Acoustic Transient Processor (ATP) Cont'd...

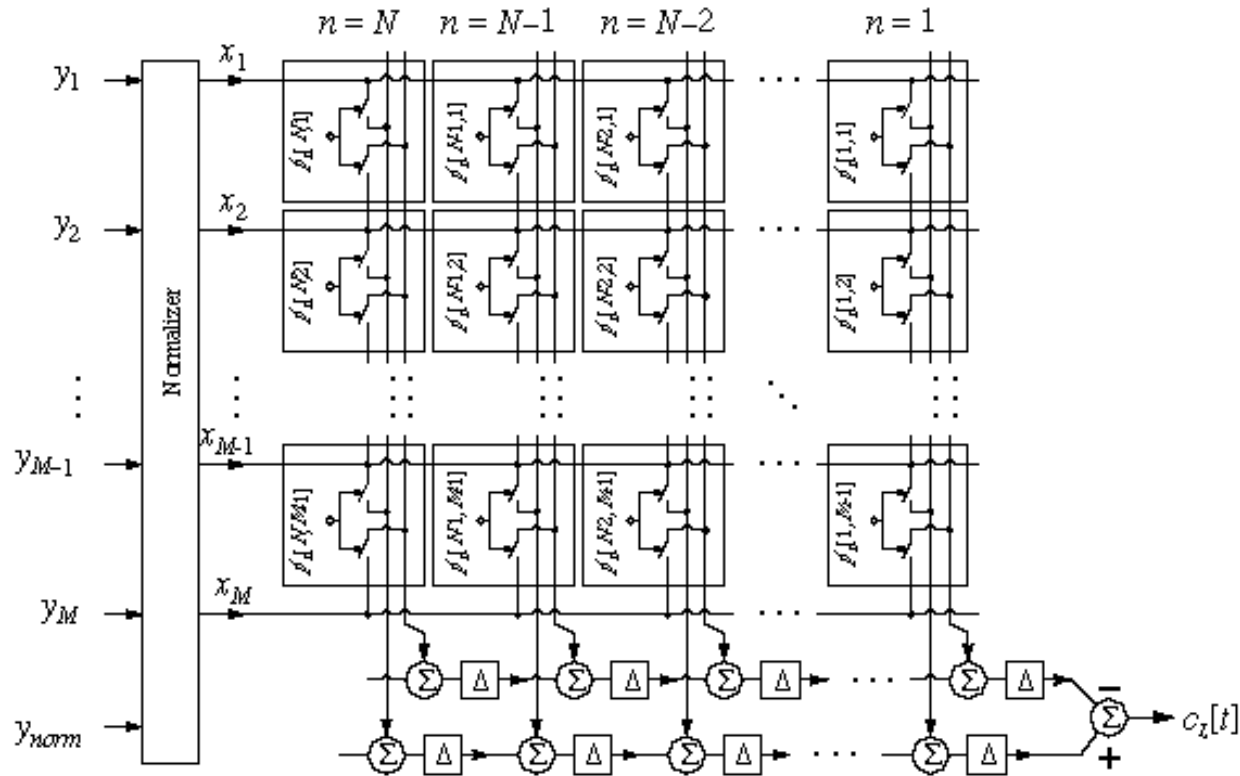
## Algorithmic and Architectural Simplifications (2)



- Binary  $\{-1,+1\}$  template values can be replaced with  $\{0,1\}$  because of normalized inputs.
- Correlation operator reduces to a simple one-way (on/off) switching element per cell.

# Acoustic Transient Processor (ATP) Cont'd...

## Algorithmic and Architectural Simplifications (3)



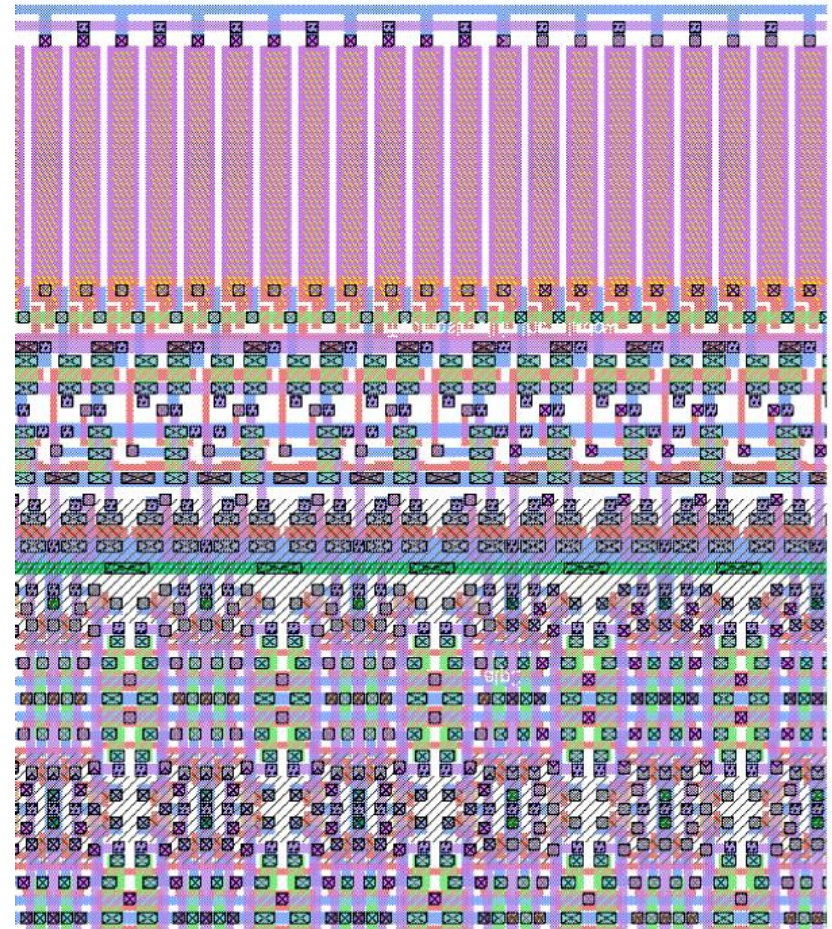
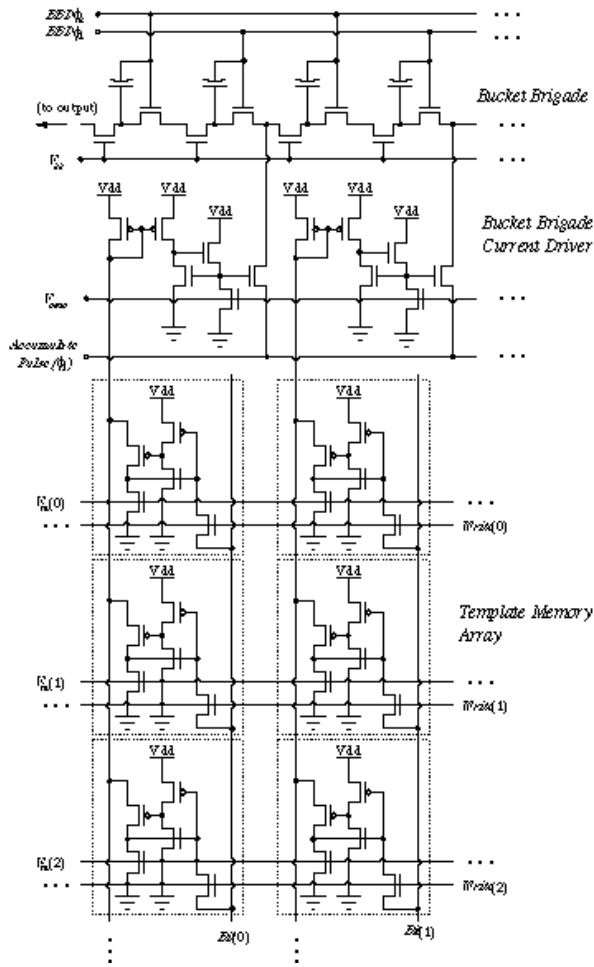
- Channel differencing can be performed in the correlator, rather than at the input. The cost seems like a factor of two in complexity. Not quite:
- Analog input is positive, simplifying correlation to single-quadrant, implemented efficiently with current-mode switching circuitry.
- Shift-and-accumulate is differential.

# Acoustic Transient Processor (ATP)

## Memory-Based Circuit Implementation

Shift-and-Accumulate

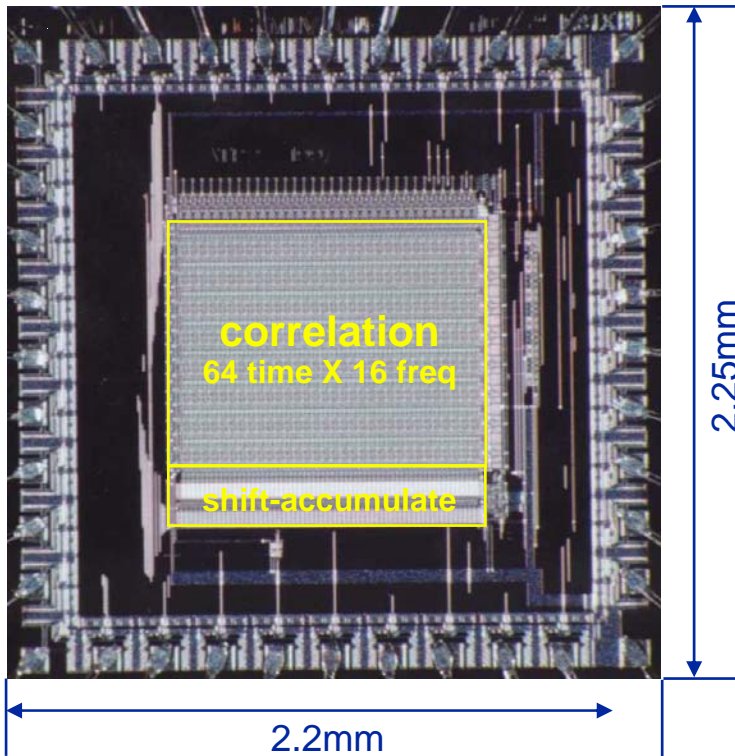
Correlation





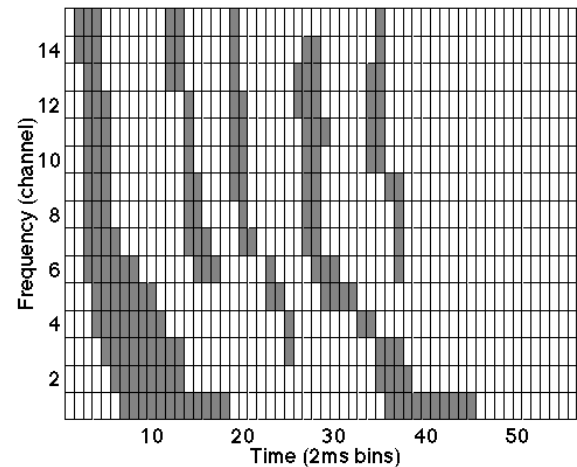
# Acoustic Transient Processor (ATP)

with Tim Edwards and Fernando Pineda

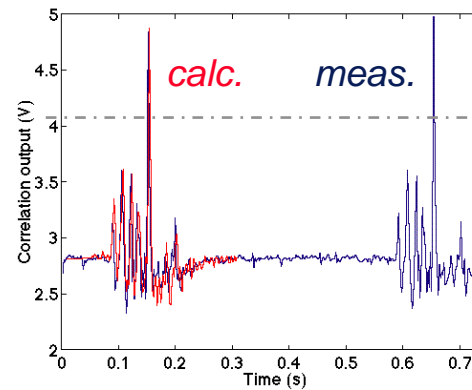


- 2.2mm X 2.2mm in 1.2 $\mu$ m CMOS
- 64 time X 16 frequency bins
- 30  $\mu$ W power at 5V

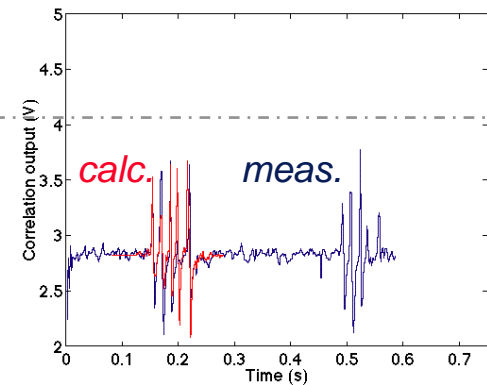
“Can” template



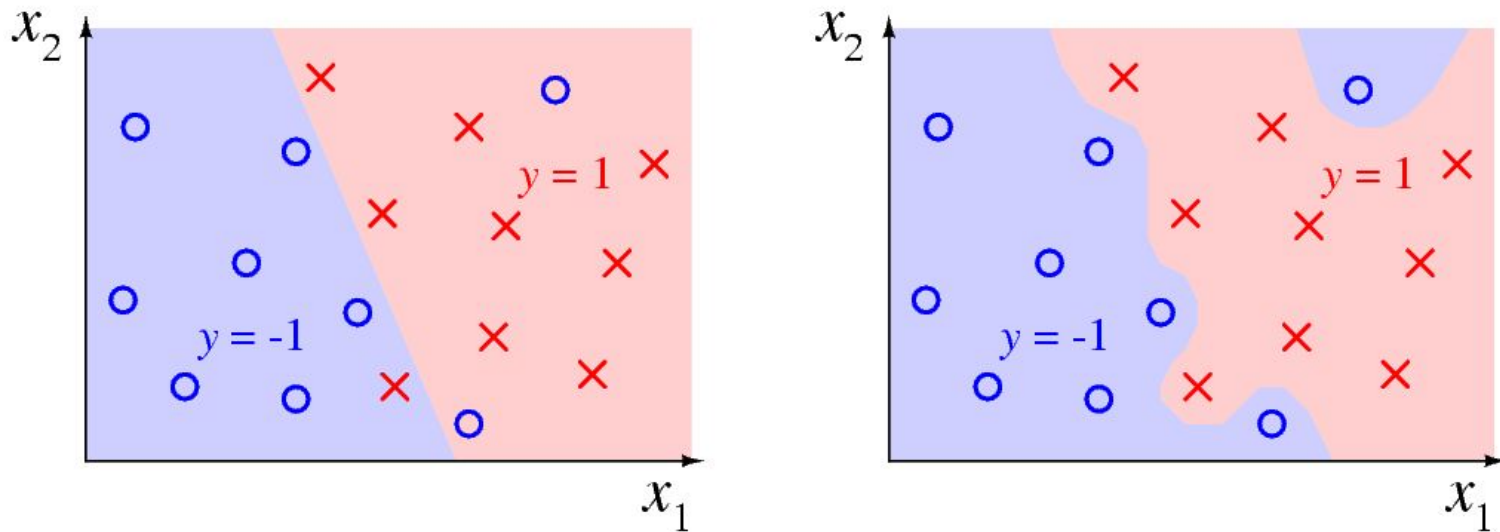
“Can” response



“Snap” response



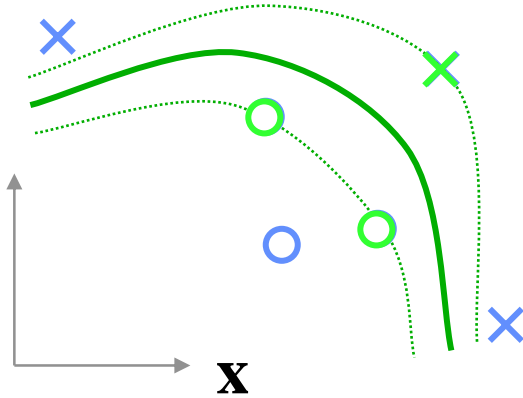
# Generalization and Complexity



- Generalization is the key to supervised learning, for classification or regression.
- *Statistical Learning Theory* offers a principled approach to understanding and controlling generalization performance.
  - *The complexity of the hypothesis class of functions determines generalization performance.*
  - *Support vector machines control complexity by maximizing the margin of the classified training data.*

# Kernel Machines

Mercer, 1909; Aizerman et al., 1964  
Boser, Guyon and Vapnik, 1992

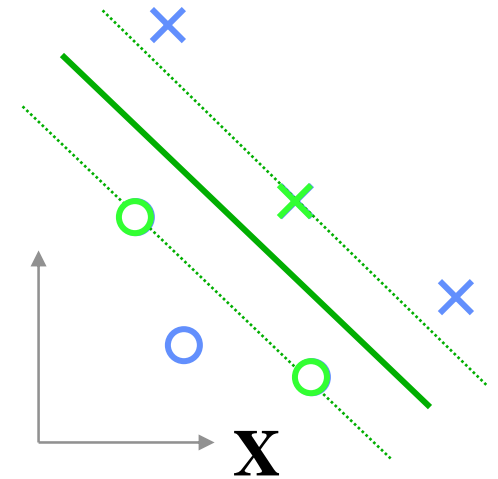


$$\Phi(\cdot)$$



$$\mathbf{X}_i = \Phi(\mathbf{x}_i)$$

$$\mathbf{X} = \Phi(\mathbf{x})$$



$$\mathbf{X}_i \cdot \mathbf{X} = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})$$



$$y = \text{sign}\left(\sum_{i \in S} \alpha_i y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) + b\right)$$

$$y = \text{sign}\left(\sum_{i \in S} \alpha_i y_i \mathbf{X}_i \cdot \mathbf{X} + b\right)$$

$$K(\cdot, \cdot)$$



$$\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) = K(\mathbf{x}_i, \mathbf{x})$$

Mercer's Condition

$$y = \text{sign}\left(\sum_{i \in S} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right)$$

# Some Valid Kernels

Boser, Guyon and Vapnik, 1992

- Polynomial (Splines etc.)

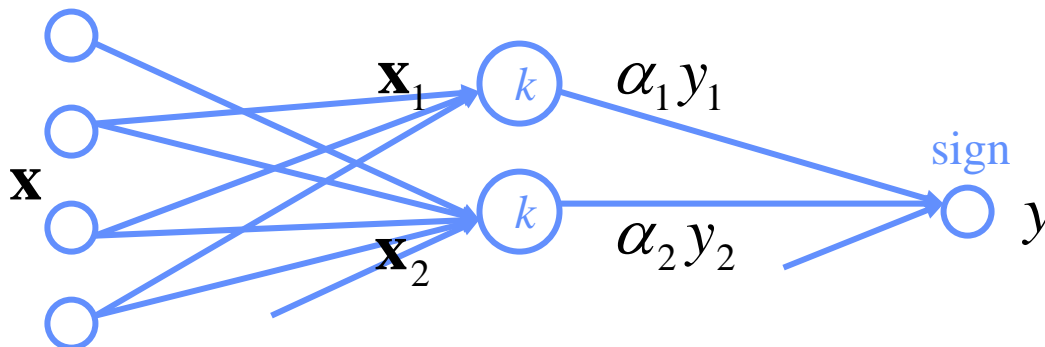
$$K(\mathbf{x}_i, \mathbf{x}) = (1 + \mathbf{x}_i \cdot \mathbf{x})^v$$

- Gaussian (Radial Basis Function Networks)

$$K(\mathbf{x}_i, \mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}\|^2}{2\sigma^2}\right)$$

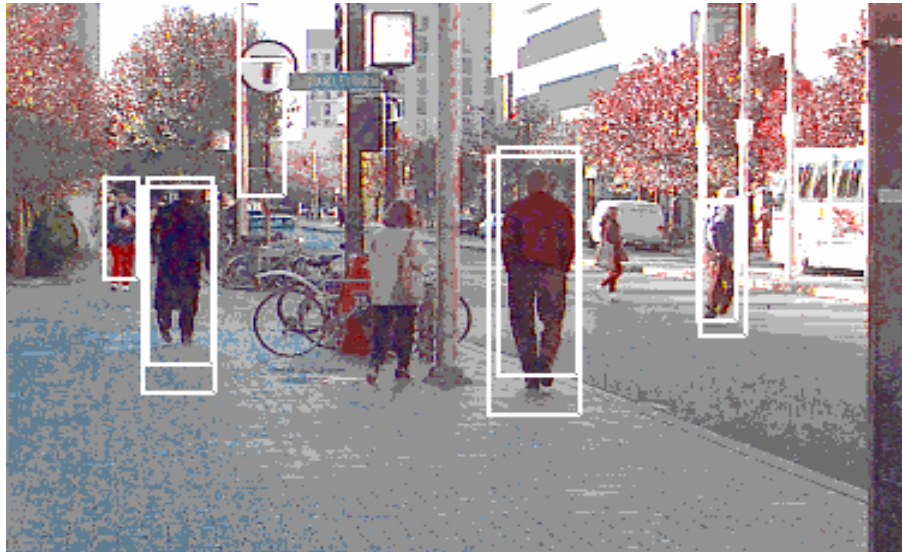
- Sigmoid (Two-Layer Perceptron)

$$K(\mathbf{x}_i, \mathbf{x}) = \tanh(L + \mathbf{x}_i \cdot \mathbf{x}) \quad \text{only for certain } L$$

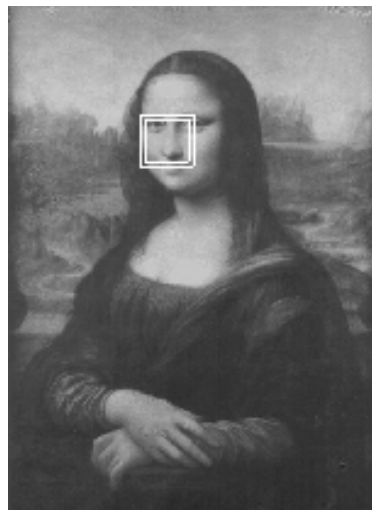


# Trainable Modular Vision Systems: The SVM Approach

Papageorgiou, Oren, Osuna and Poggio, 1998



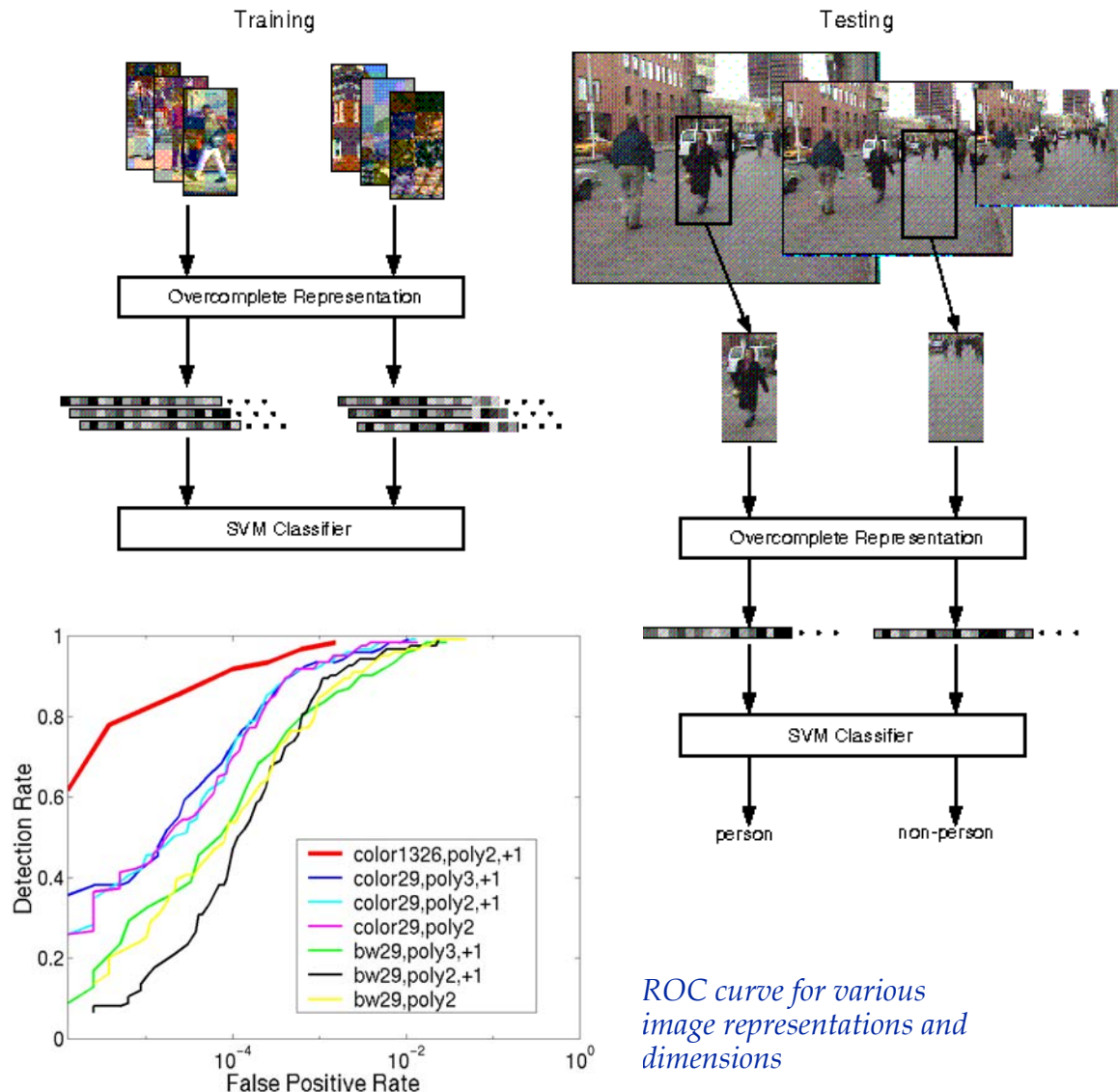
- Strong mathematical foundations in *Statistical Learning Theory* (Vapnik, 1995)
- The training process selects a small fraction of prototype *support vectors* from the data set, located at the *margin* on both sides of the classification boundary (e.g., barely faces vs. barely non-faces)



*SVM classification for pedestrian and face object detection*

# Trainable Modular Vision Systems: The SVM Approach

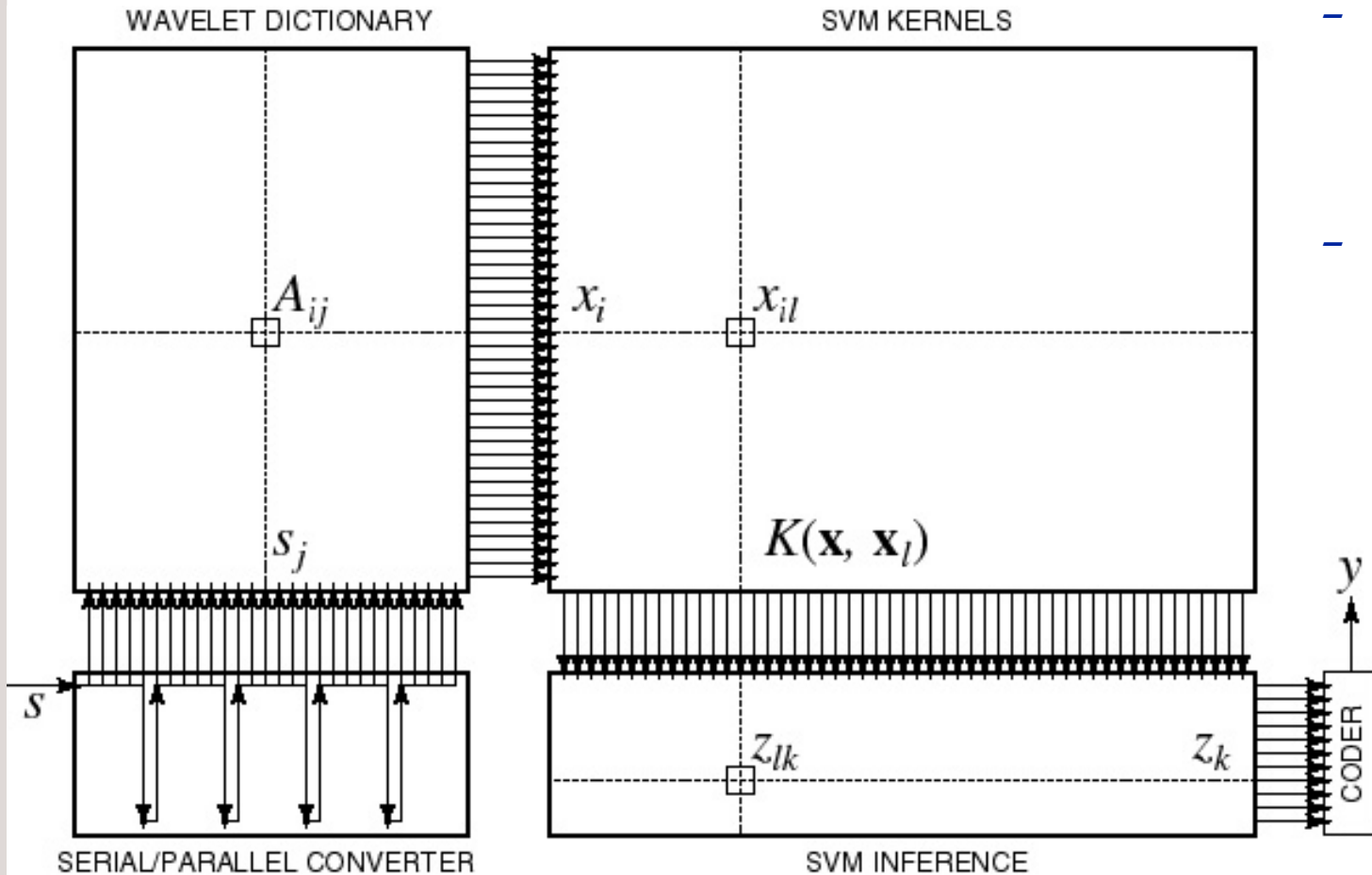
Papageorgiou, Oren, Osuna and Poggio, 1998



- The number of support vectors and their dimensions, in relation to the available data, determine the generalization performance
- Both training and run-time performance are severely limited by the computational complexity of evaluating kernel functions

*ROC curve for various image representations and dimensions*

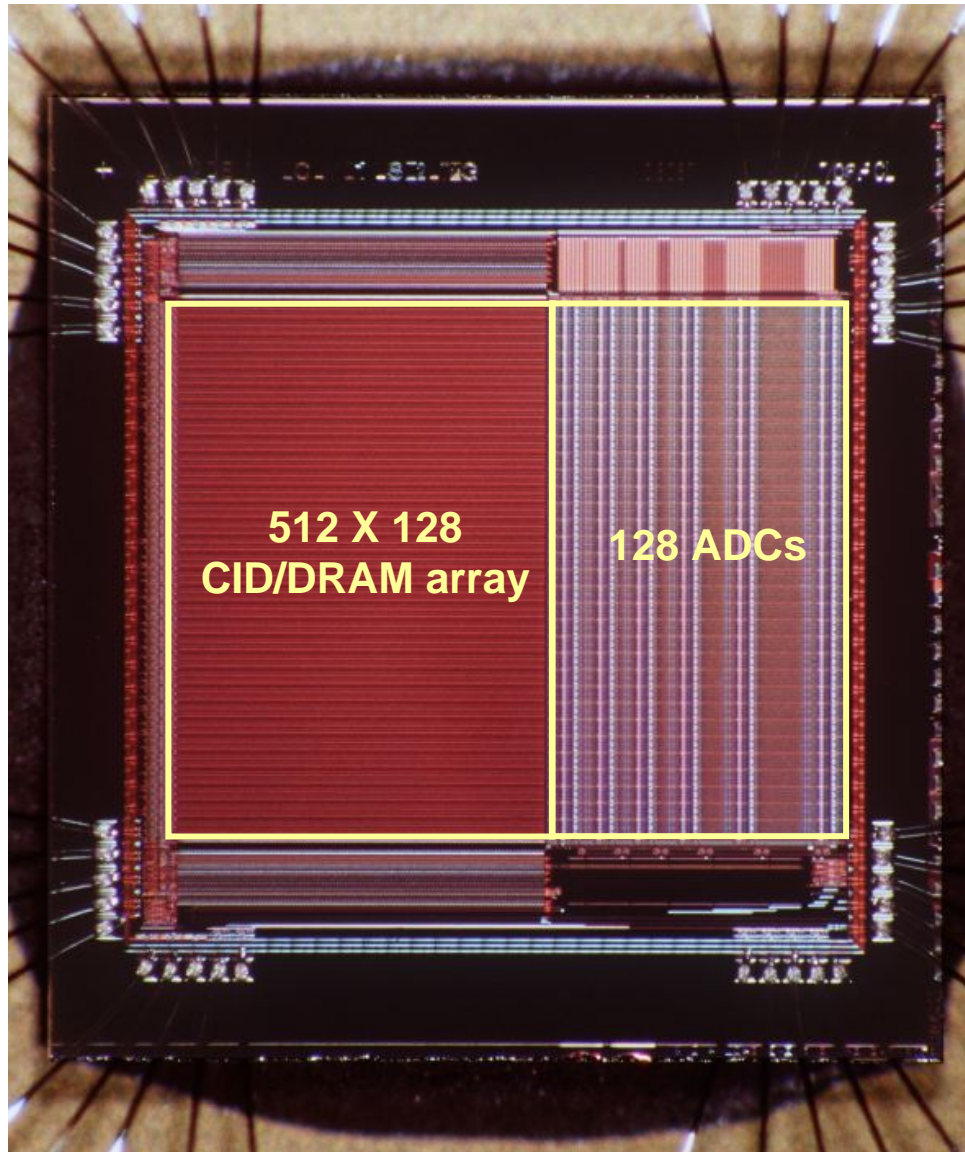
# Scalable Parallel SVM Architecture



- Full parallelism yields very large computational throughput
- Low-rate input and output encoding reduces bandwidth of the interface

# The *Kerneltron*: Support Vector “Machine”

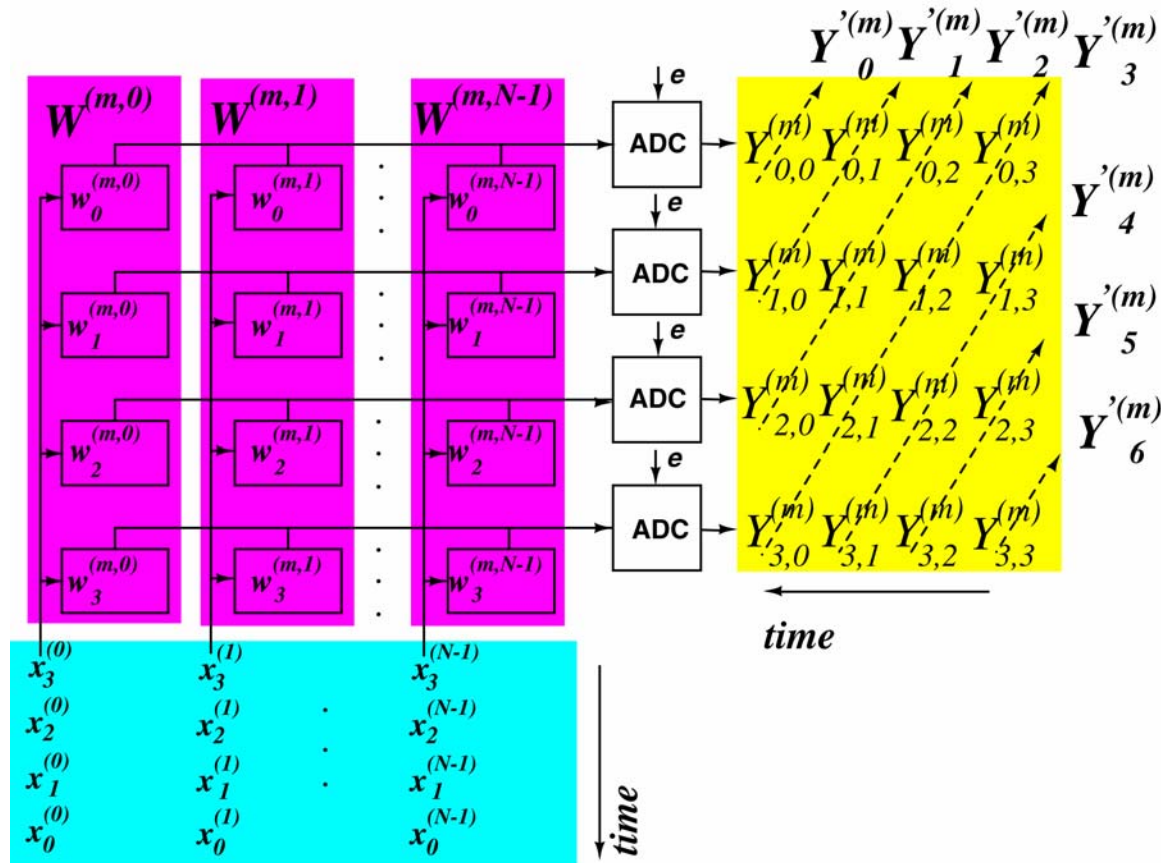
Genov and Cauwenberghs, 2001



- 512 inputs, 128 support vectors
- 3mm X 3mm in 0.5um CMOS
- Fully parallel operation using “computational memories” in hybrid DRAM/CCD technology
- Internally analog, externally digital
- Low bit-rate, serial I/O interface
- Supports functional extensions on SVM paradigm

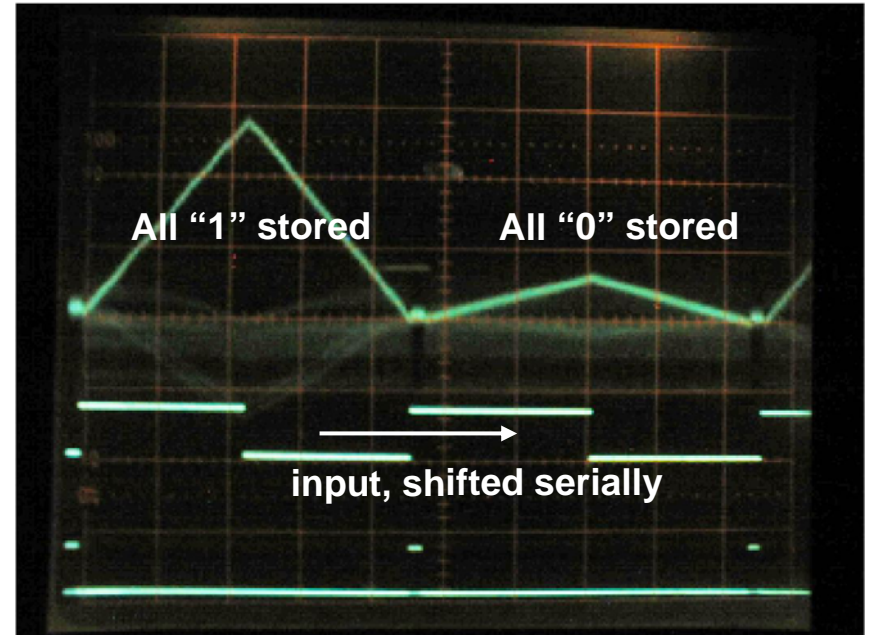
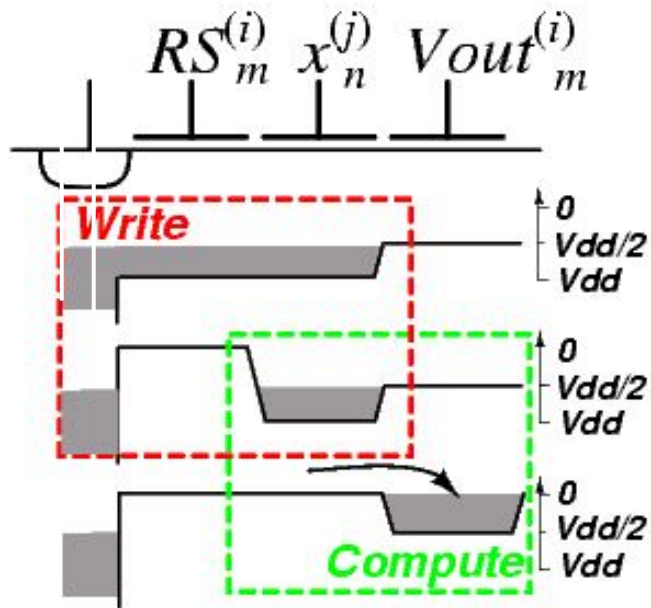
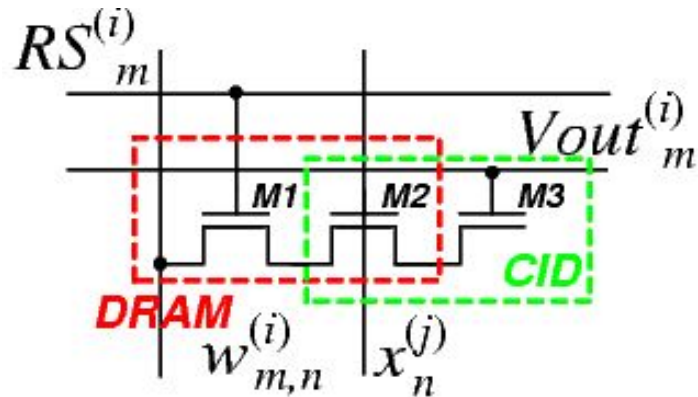


# Mixed-Signal Parallel Pipelined Architecture



- Externally digital processing and interfacing
  - *Bit-serial input, and bit-parallel storage of matrix elements*
  - *Digital output is obtained by combining quantized partial products*

# CID/DRAM Cell and Analog Array Core

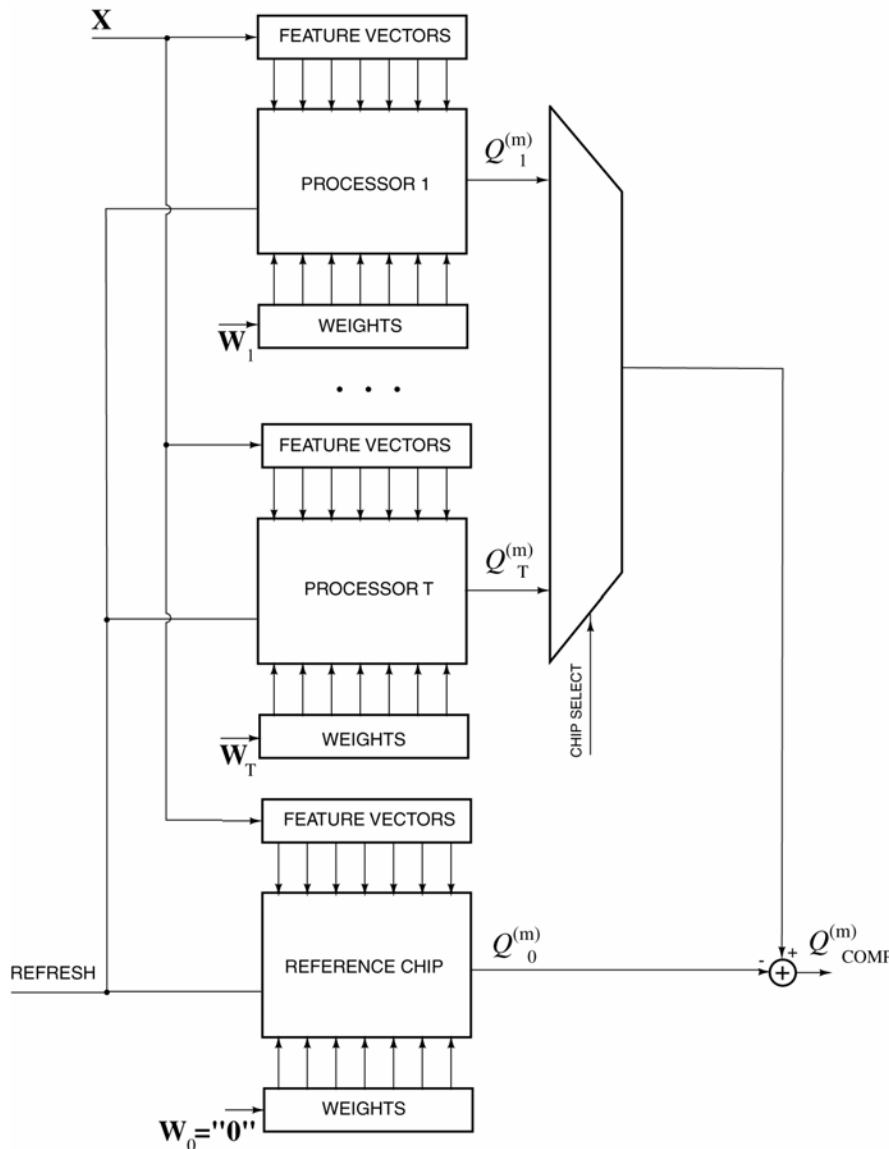


*Linearity of parallel analog summation*

- Internally analog computing
  - *Computational memory integrates DRAM with CID*

# Feedthrough and Leakage Compensation

*in an extendable multi-chip architecture*



$x_j^{(n)}$	$w_i^{(m,n)}$	$y_{i,j}^{(m,n)}$
0	0	0
0	1	0
1	0	$\epsilon$
1	1	$1+\epsilon$

$$Y_{i,j}^{(m)} = \sum_{n=0}^{N-1} w_i^{(m,n)} x_j^{(n)} = \sum_{n=0}^{N-1} y_{i,j}^{(m,n)}$$

$$\begin{aligned}
 Y_{i,j}^{(m)} &= (1 + \epsilon) \sum_{n=0}^{N-1} w_i^{(m,n)} x_j^{(n)} \\
 &\quad + \epsilon \sum_{n=0}^{N-1} (1 - w_i^{(m,n)}) x_j^{(n)} \\
 &= \sum_{n=0}^{N-1} w_i^{(m,n)} x_j^{(n)} + \epsilon \sum_{n=0}^{N-1} x_j^{(n)}
 \end{aligned}$$

# Oversampled Input Coding/Quantization

- **Binary** support vectors are stored in *bit-parallel* form
- Digital inputs are *oversampled* (e.g. *unary* coded) and presented *bit-serially*

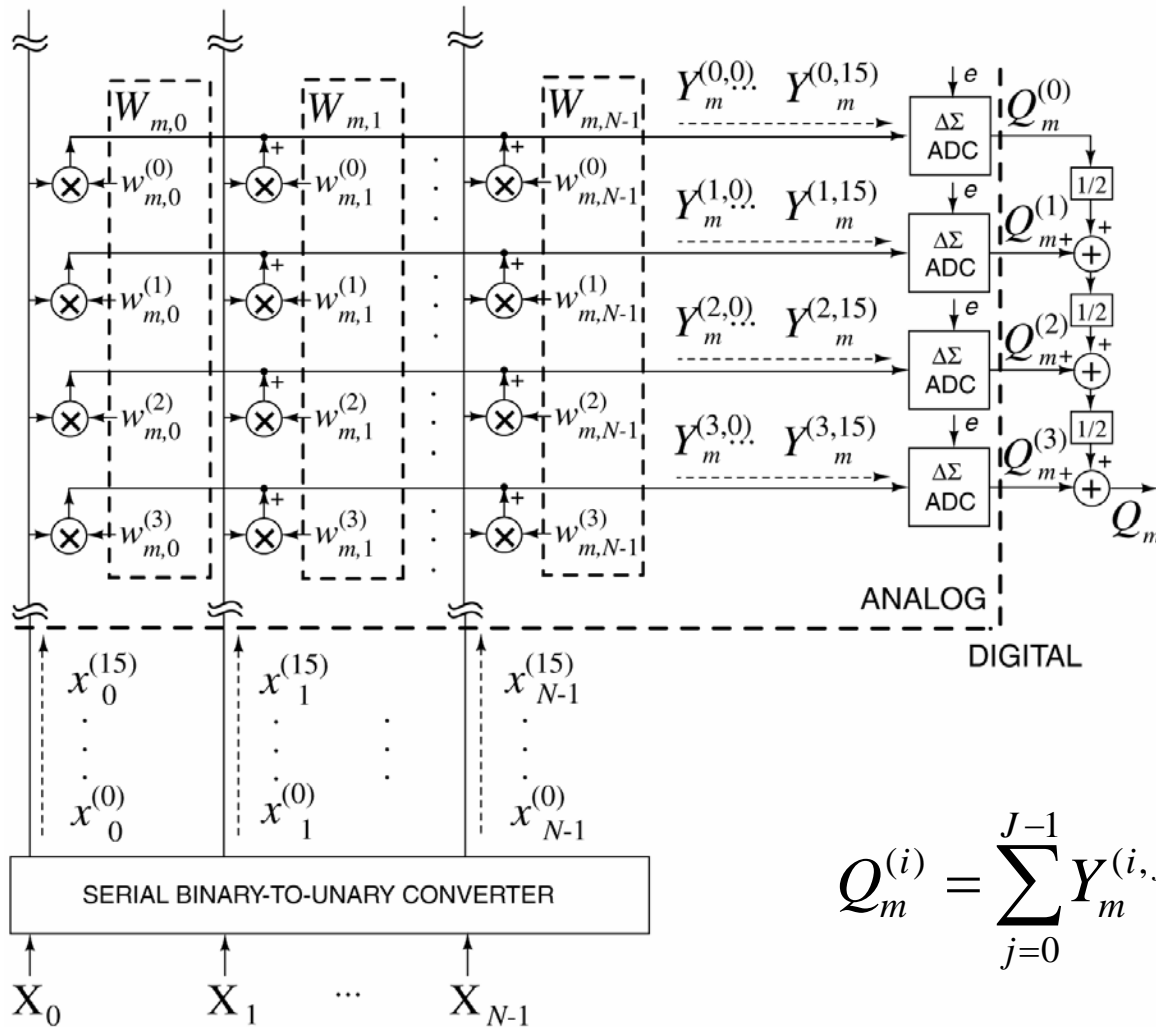
$$W_{mn} = \sum_{i=0}^{I-1} 2^{-i-1} w_{mn}^{(i)} ; \quad X_n = \sum_{j=0}^{J-1} x_n^{(j)} \quad \leftarrow \text{Data encoding}$$

$$Y_m = \sum_{n=0}^{N-1} W_{mn} X_n = \sum_{i=0}^{I-1} 2^{-i-1} Y_m^{(i)}, \quad \text{where} \quad \leftarrow \text{Digital accumulation}$$

$$Y_m^{(i)} = \sum_{j=0}^{J-1} Y_m^{(i,j)} \quad \text{and} \quad \leftarrow \text{Analog delta-sigma accumulation}$$

$$Y_m^{(i,j)} = \sum_{n=0}^{N-1} w_{mn}^{(i)} x_n^{(j)} \quad \leftarrow \text{Analog charge-mode accumulation}$$

# Oversampling Architecture

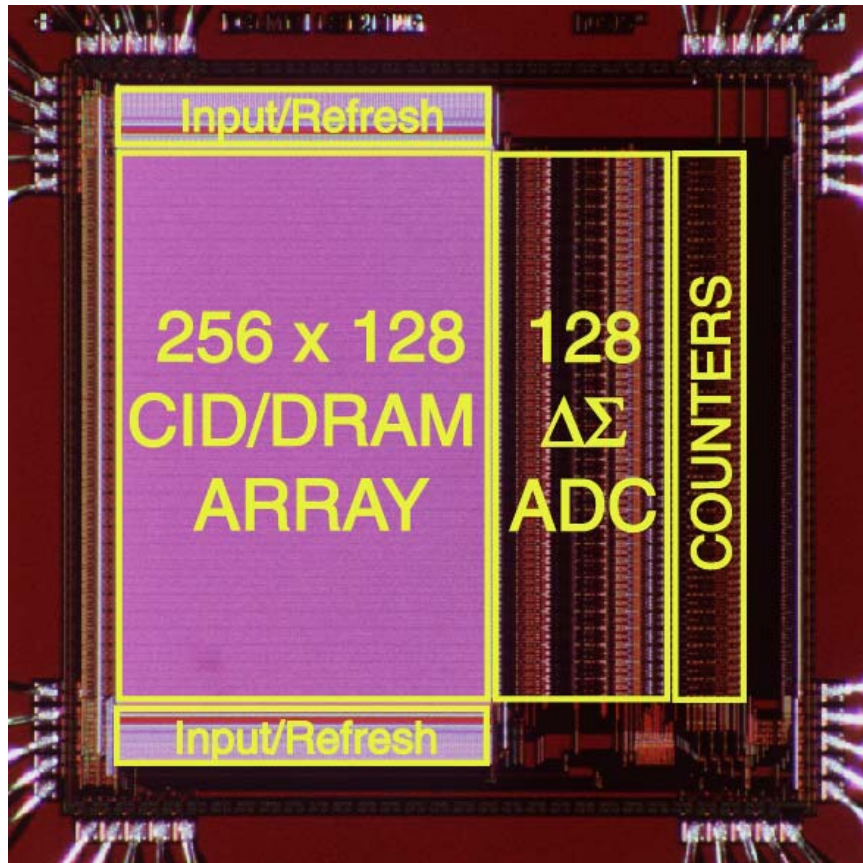


- Oversampled input coding (e.g. unary)
- Delta-sigma modulated ADCs accumulate and quantize row outputs for all unary bit-planes of the input

$$Q_m^{(i)} = \sum_{j=0}^{J-1} Y_m^{(i,j)} + e$$

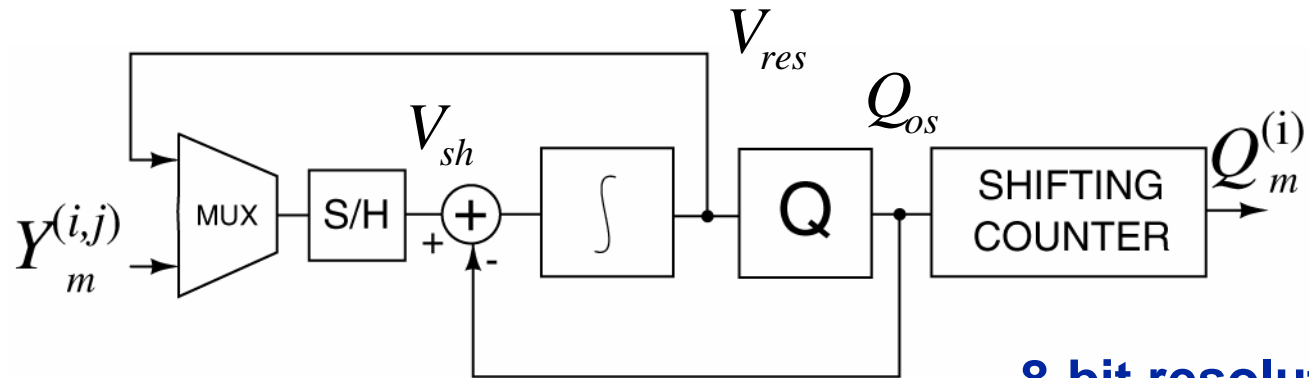
# Kerneltron II

Genov, Cauwenberghs, Mulliken and Adil, 2002

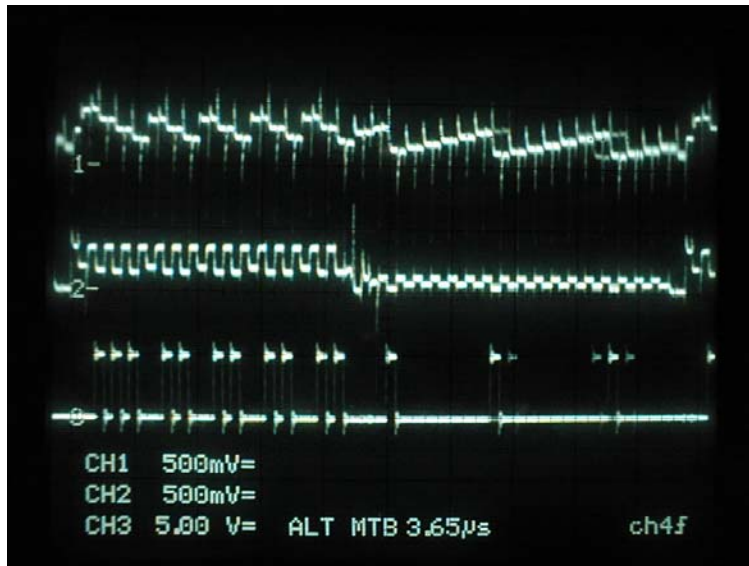


- 3mm x 3mm chip in 0.5 $\mu$ m CMOS
- Contains 256 x 128 cells and 128 8-bit delta-sigma algorithmic ADCs
- 6.6 GMACS throughput
- 5.9 mW power dissipation
- 8 bit full digital precision
- Internally analog, externally digital
- Modular; expandable
- Low bit-rate serial I/O

# Delta-Sigma Algorithmic ADC



**8-bit resolution in  
32 cycles**

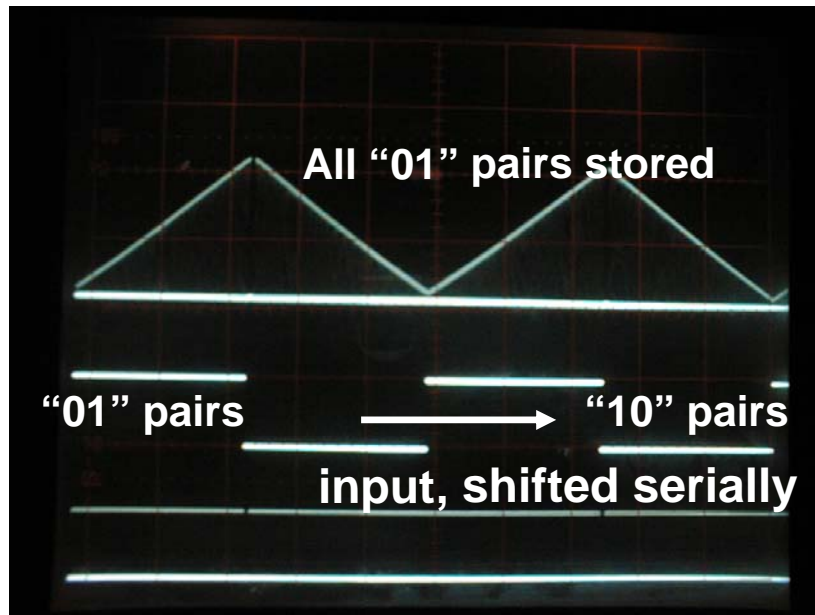
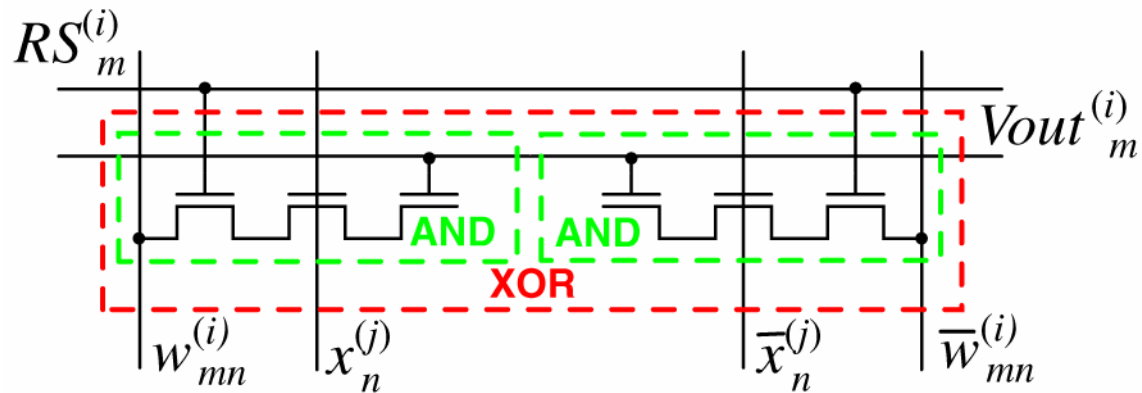


←  $V_{res}$  *Residue voltage*

←  $V_{sh}$  *S/H voltage*

←  $Q_{os}$  *Oversampled digital output*

# Signed Multiply-Accumulate Cell

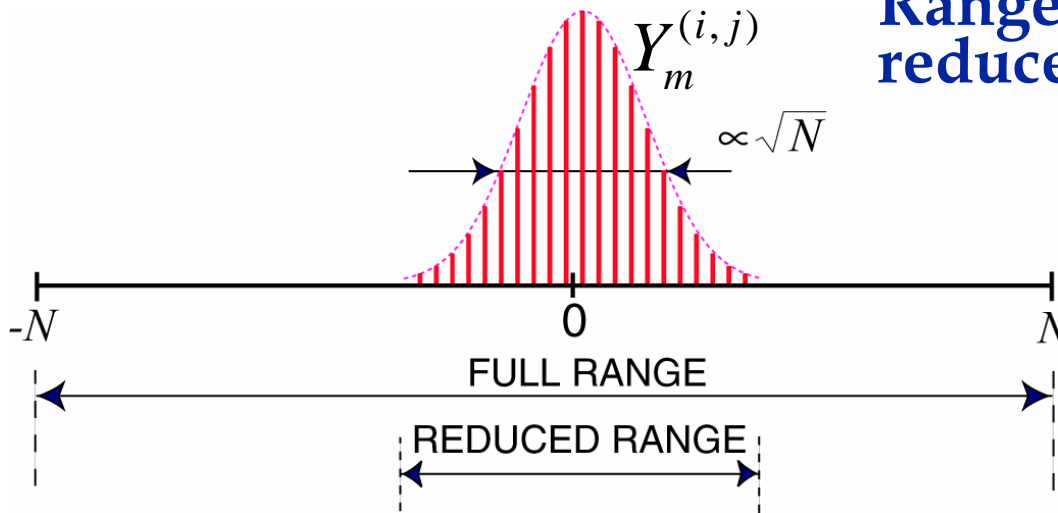


*Linearity of parallel analog summation in XOR CID/DRAM cell configuration*



# Stochastic Architecture

Range of  $Y_m^{(i,j)}$  is reduced by  $\sqrt{N}$

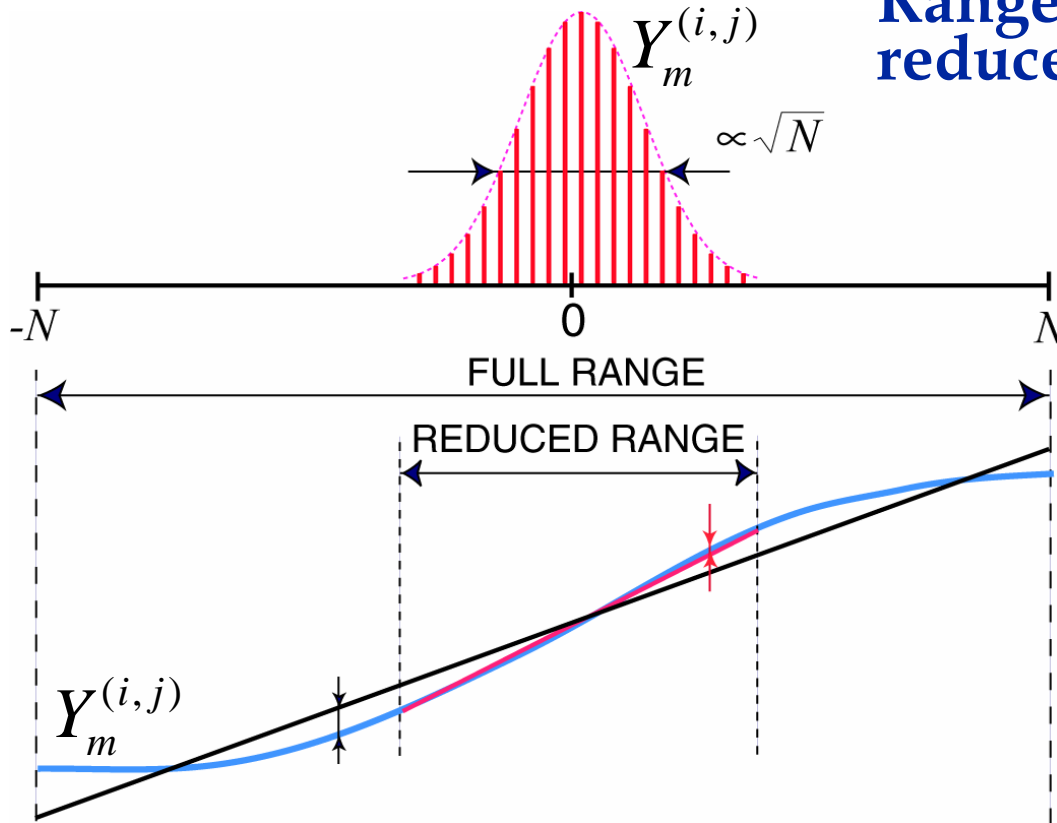


- ADC resolution requirements are relaxed by  $\sqrt{N}$

$$Y_{i,j}^{(m)} \rightarrow N(\mu=0, \sigma=\sqrt{N})$$

# Stochastic Architecture

Range of  $Y_m^{(i,j)}$  is reduced by  $\sqrt{N}$



- ADC resolution requirements are relaxed by  $\sqrt{N}$

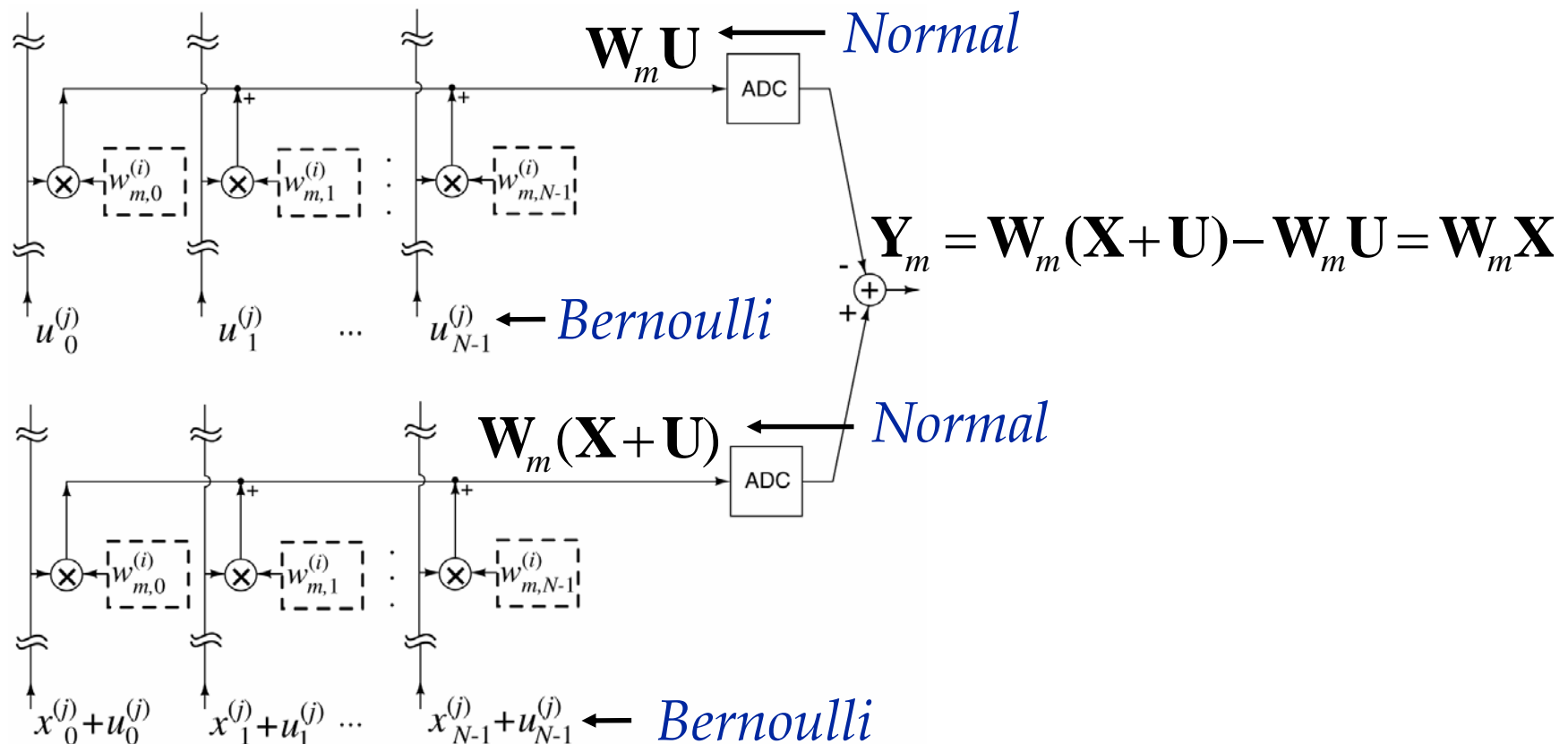
- Analog addition nonlinearity does not affect the precision of computation

$$Y_{i,j}^{(m)} \rightarrow N(\mu=0, \sigma=\sqrt{N})$$


# Stochastic Input Encoding

$$\mathbf{U} = \sum_{k=0}^{K-1} 2^{-k-1} \mathbf{u}^{(k)} - \text{random, uniform} \quad \Rightarrow \mathbf{u}^{(k)} - \text{random, Bernoulli}$$

$$\mathbf{X} = (\mathbf{X} + \mathbf{U}) - \mathbf{U}; \quad \text{If (range of } \mathbf{U}) \gg (\text{range of } \mathbf{X}), \text{ binary coefficients of } (\mathbf{X} + \mathbf{U}) \text{ are } \sim \text{Bernoulli}$$

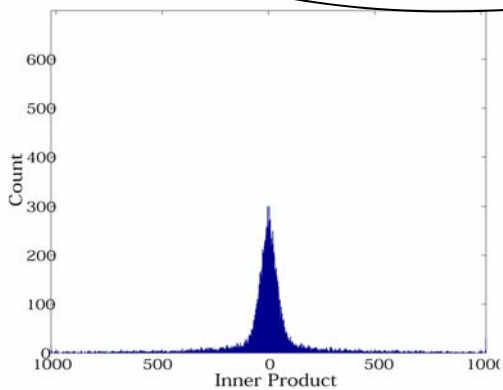


# Stochastic Encoding: Image Data

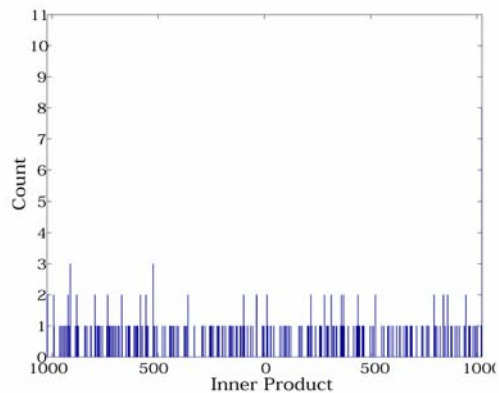
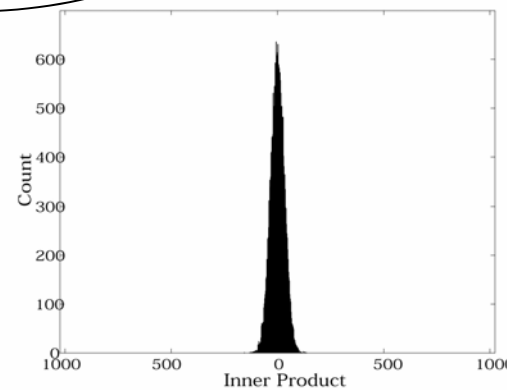
$$\mathbf{W}_m^{(i)} \cdot \mathbf{X}^{(j)} = Y_m^{(i,j)}$$


$$\mathbf{W}_m^{(i)} \cdot (\mathbf{X}^{(j)} + \mathbf{U}) = \tilde{Y}_m^{(i,j)}$$

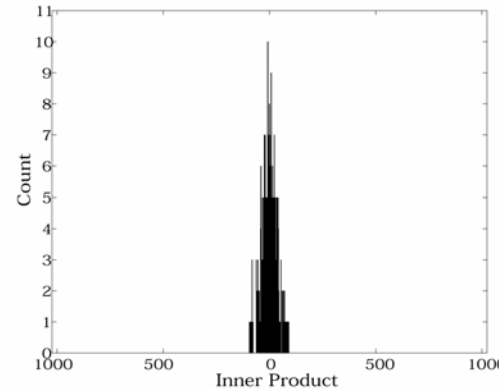
random



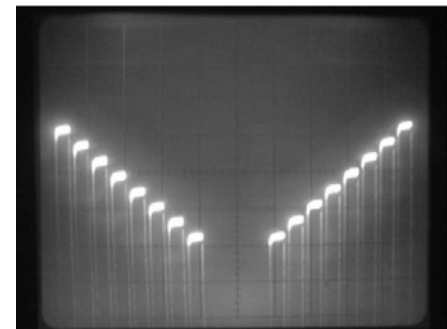
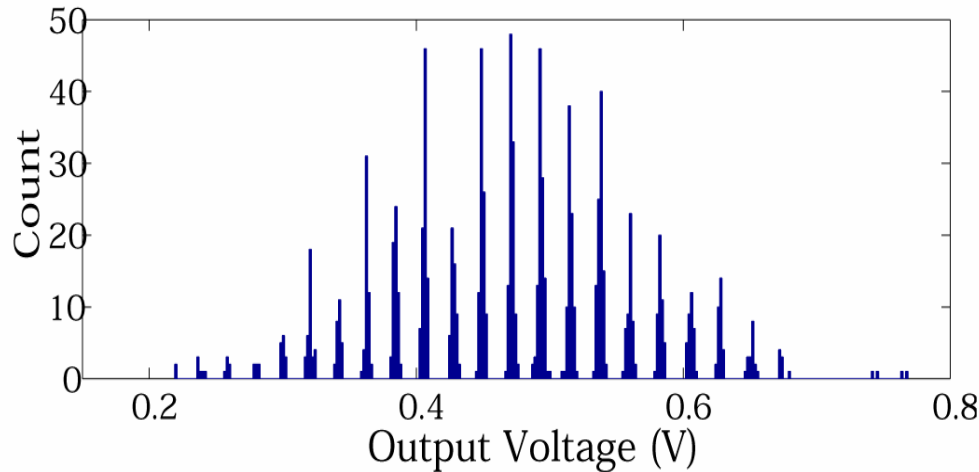
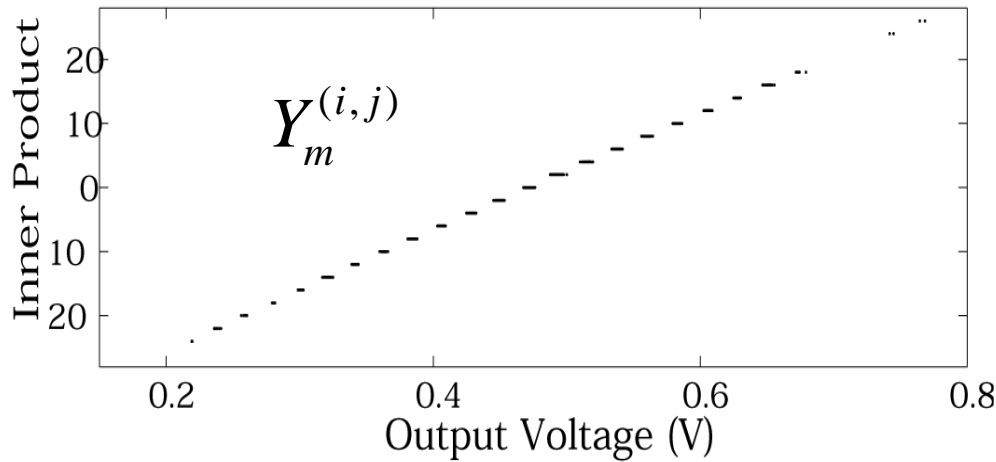
All bit  
planes



MSB  
plane



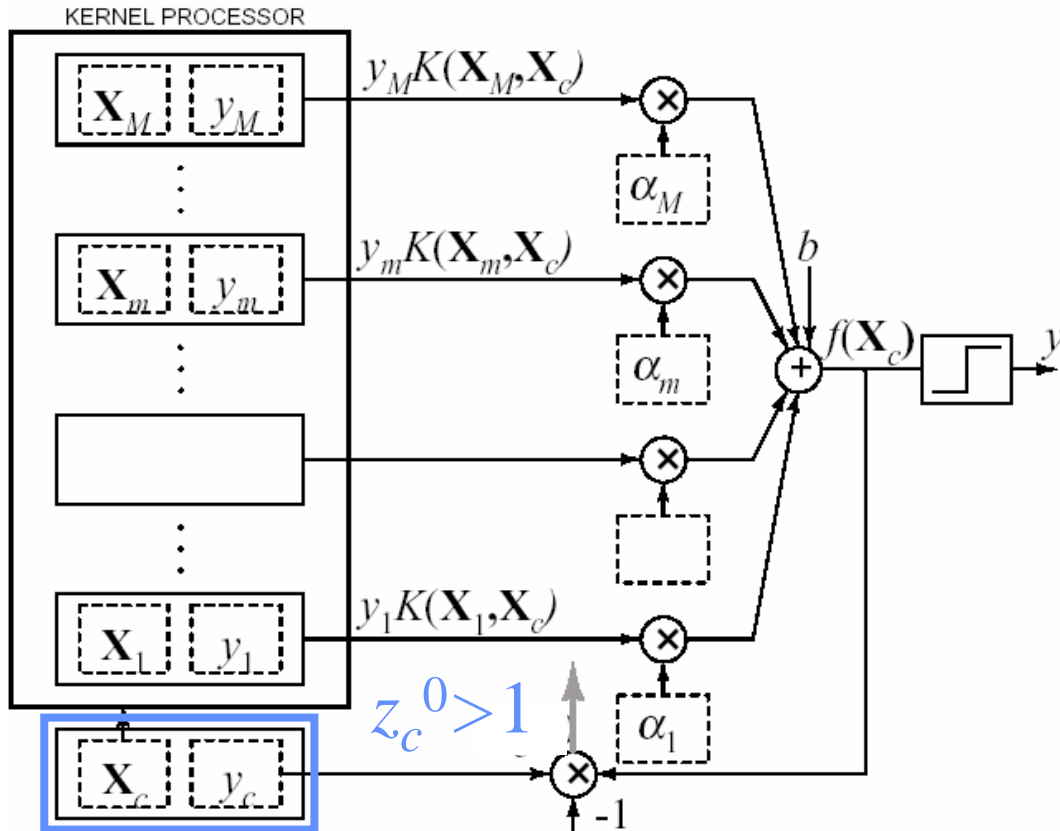
# Stochastic Encoding: Experimental Results



*Worst-case mismatch*

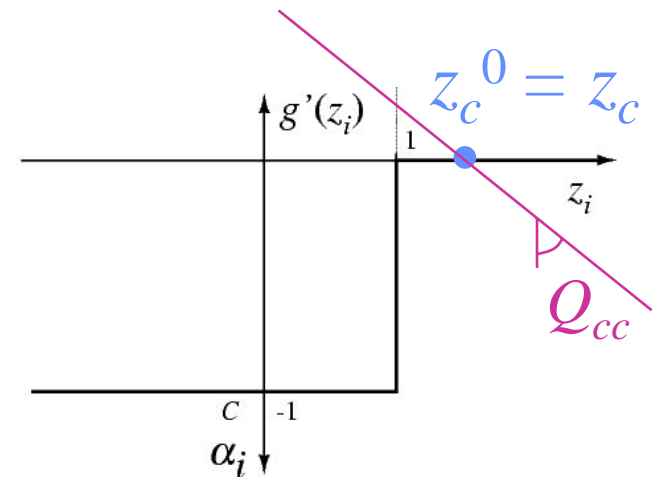
# Sequential On-Line SVM Learning

with Shantanu Chakrabarty and Roman Genov

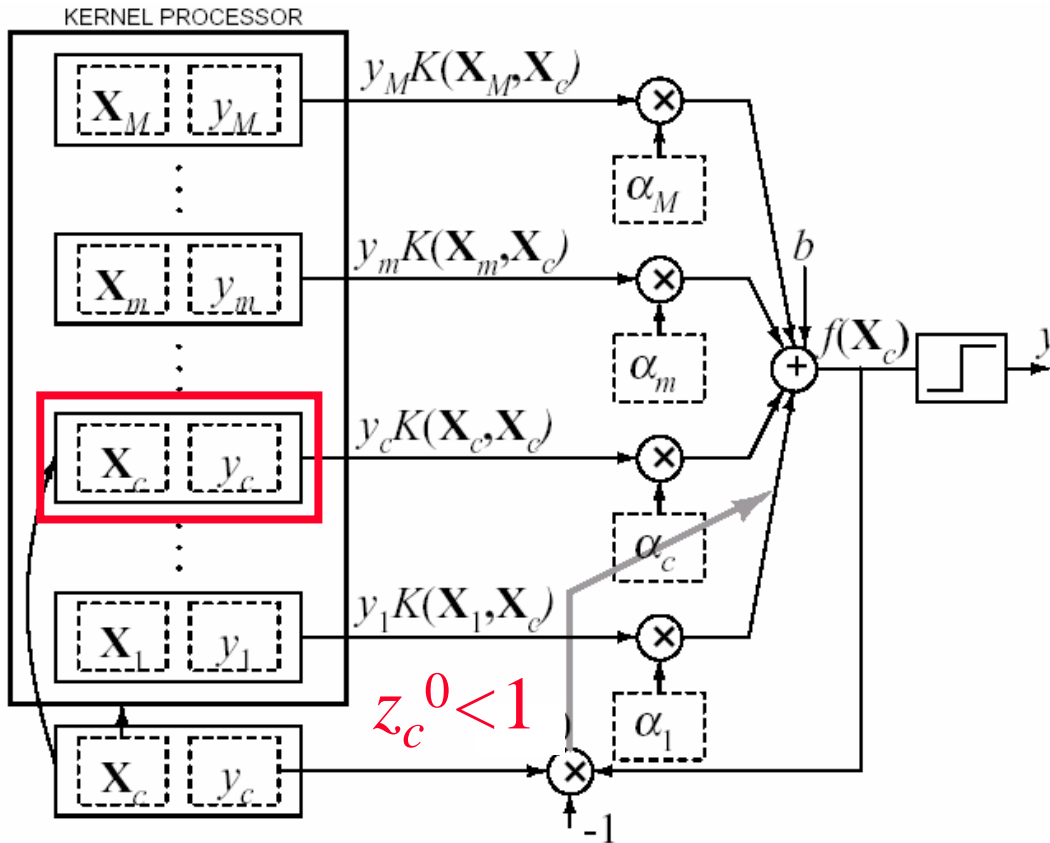


$$\alpha_c = -Cg'(z_c)$$

$$z_c \approx z_c^0 + Q_{cc} \alpha_c$$

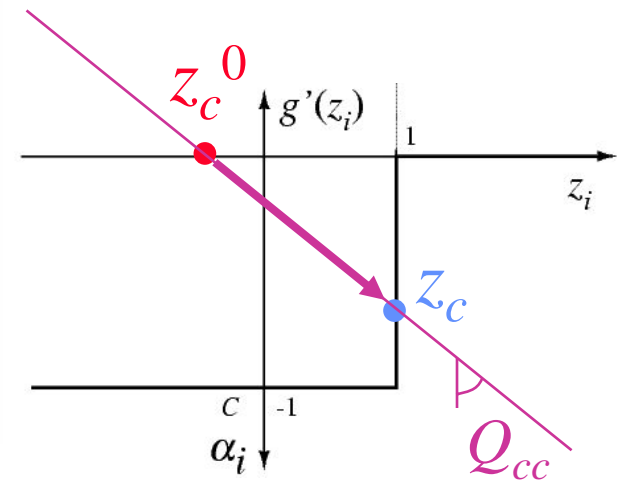


# Sequential On-Line SVM Learning



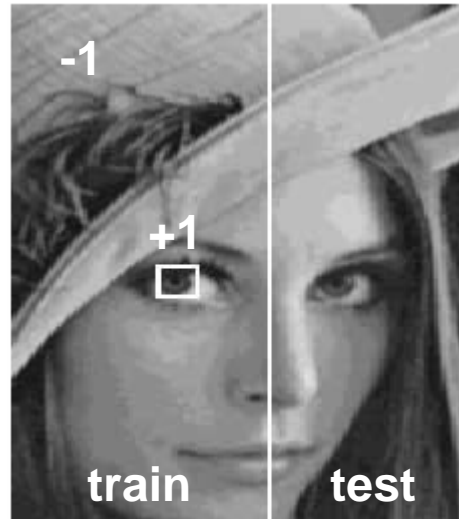
$$\alpha_c = -Cg'(z_c)$$

$$z_c \approx z_c^0 + Q_{cc} \alpha_c$$



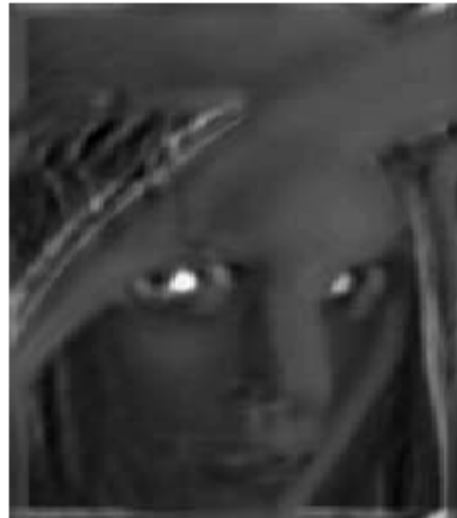
*margin/error support vector*

# Effects of Sequential On-Line Learning and Finite Resolution



- *Matched Filter Response*

- *Batch Training*
- *Floating-Point Resolution*

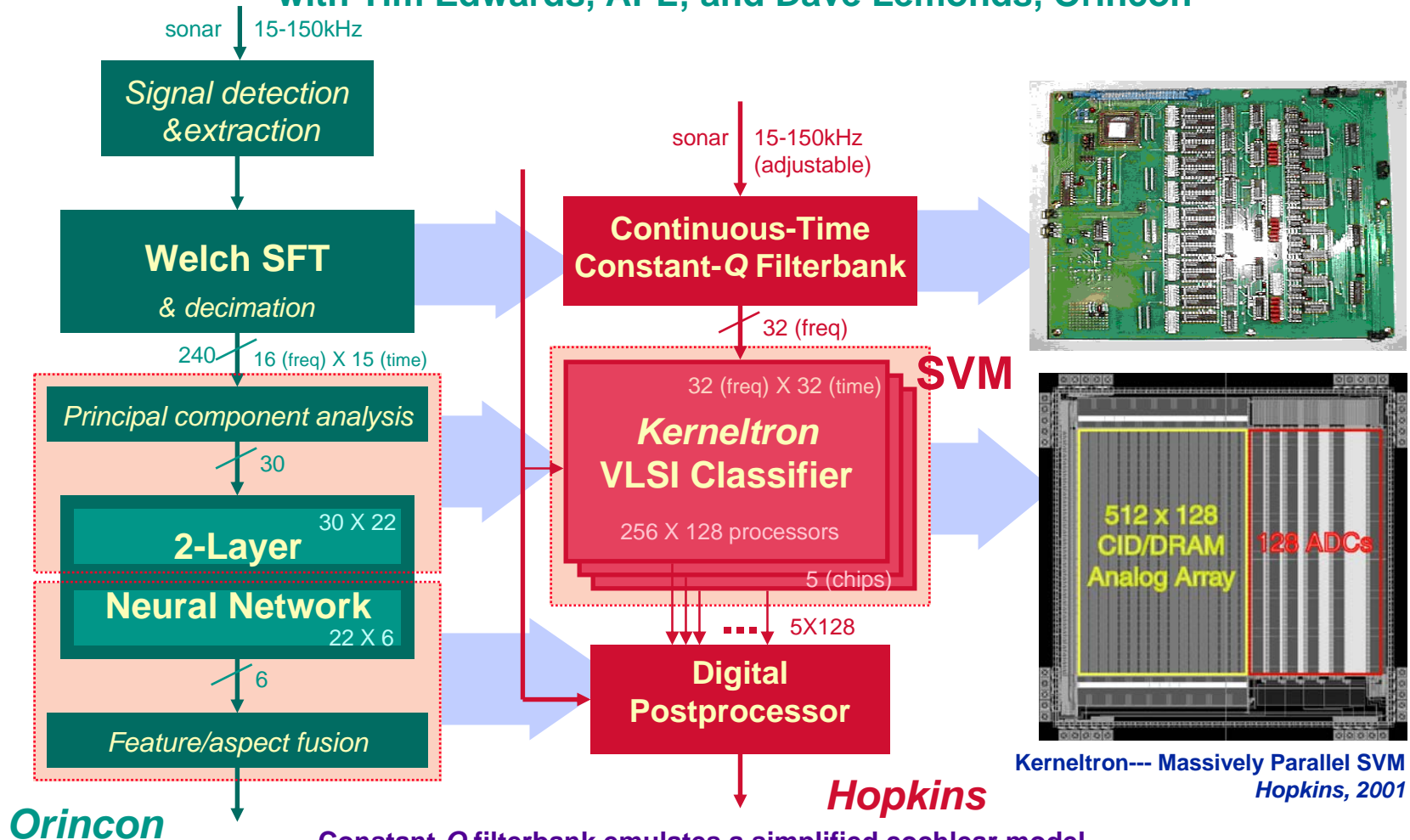


- *On-Line Sequential Training*
- *Kerneltron II*



# Biosonar Signal Processor and Classifier

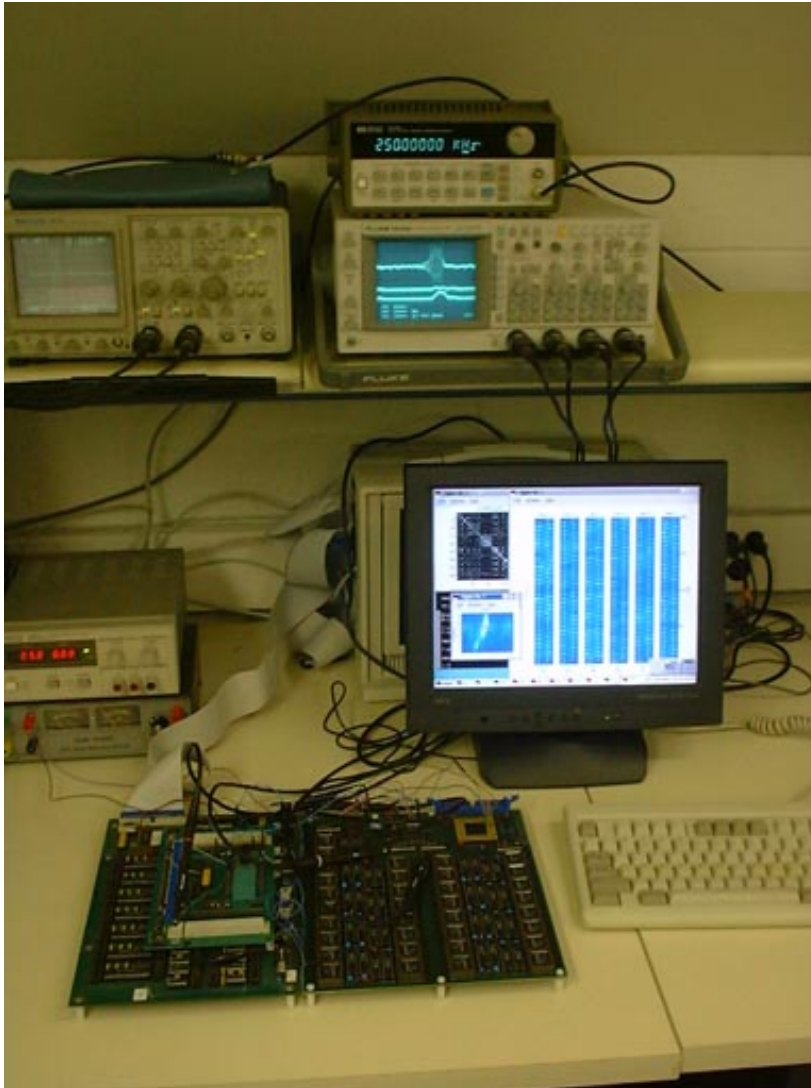
with Tim Edwards, APL; and Dave Lemonds, Orincon



Orincon

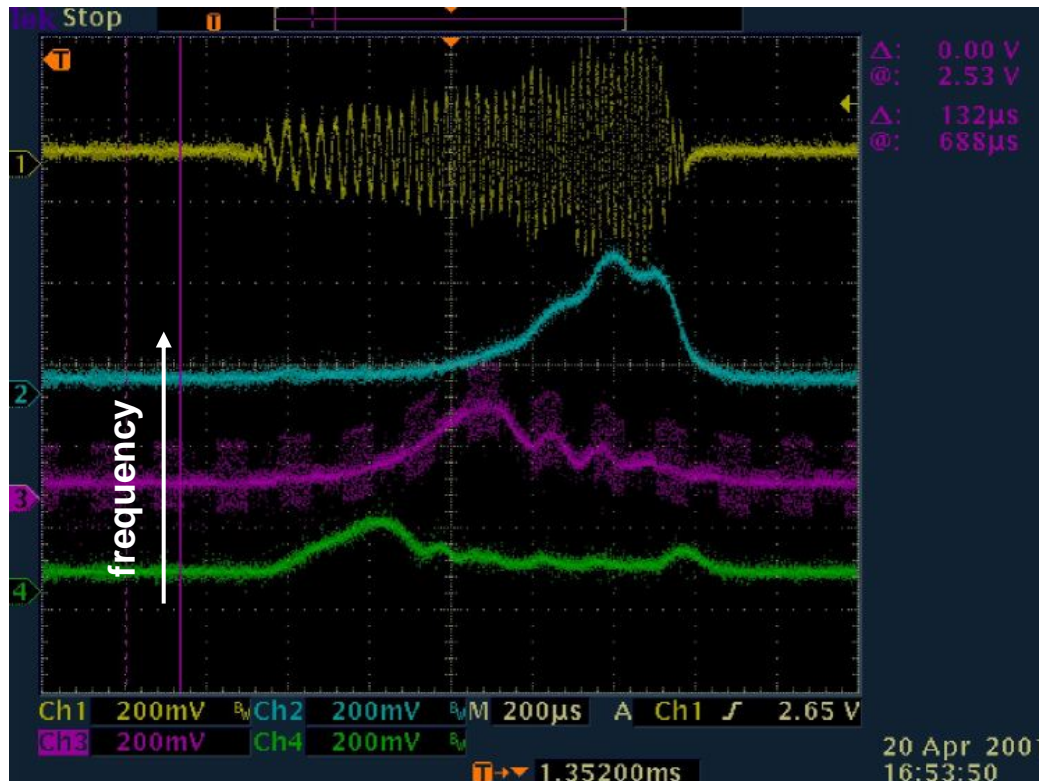
- Constant-Q filterbank emulates a simplified cochlear model
- *Kerneltron* VLSI support vector machine (SVM) emulates a general class of neural network topologies for adaptive classification
- Fully programmable, scaleable, expandable architecture
- Efficient parallel implementation with distributed memory

# Real-Time Biosonar Signal Processor



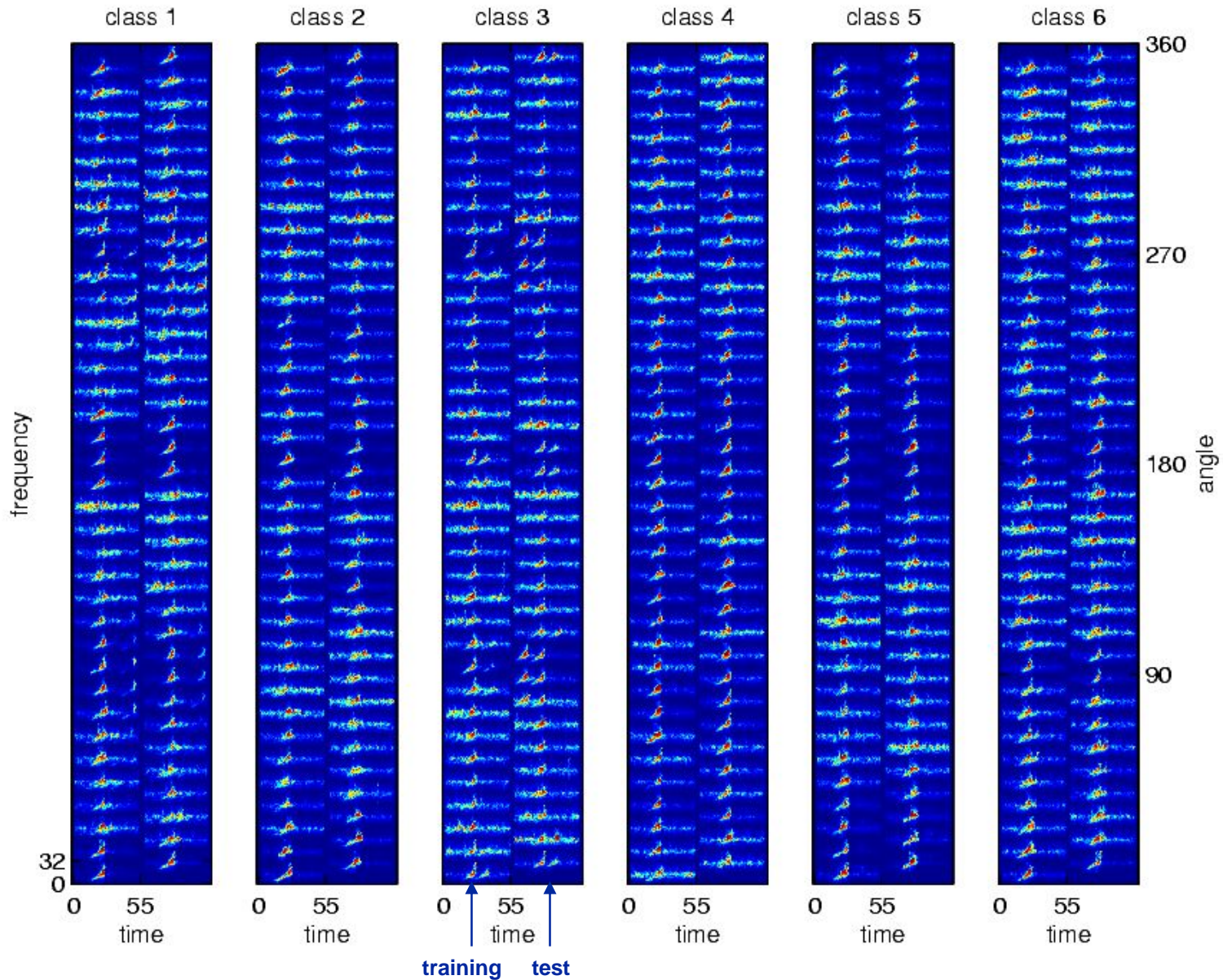
- Analog continuous-time input interface at sonar speeds
  - *250kHz bandwidth*
  - *32 frequency channels*
- Digital programmable interface
  - *In-site programmable and reconfigurable analog architecture*

# Frontend Analog Signal Processing



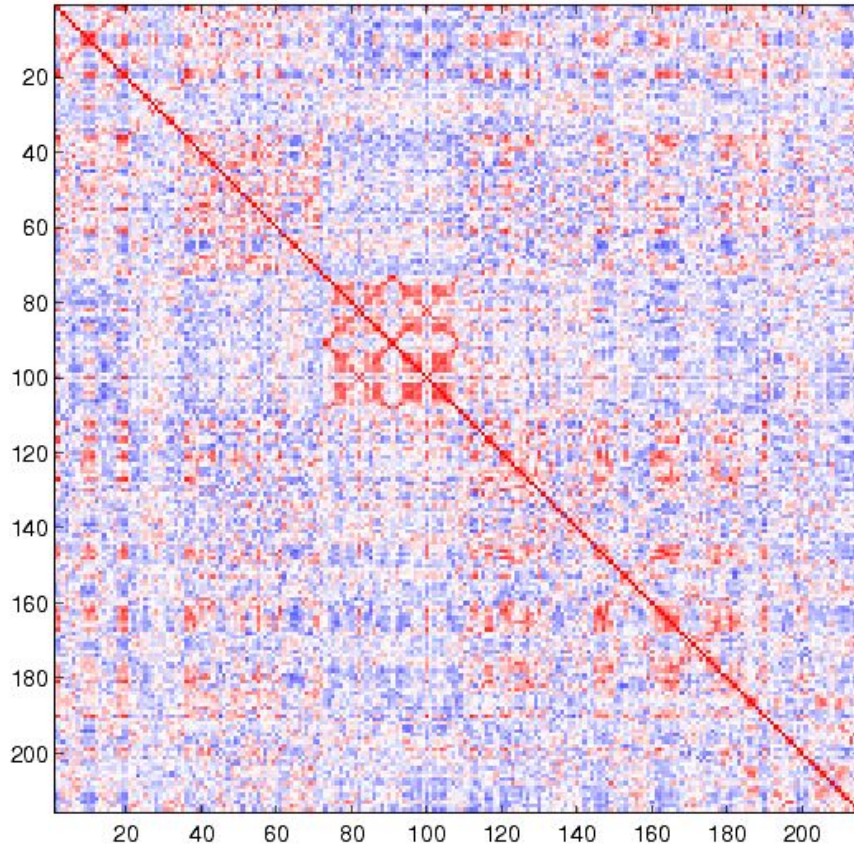
- Continuous-time filters
  - *Custom topology*
  - *Individually programmable Q and center frequencies*
  - *Linear or log spacing on frequency axis*
- Energy, envelope or phase detection
  - *Energy, synchronous*
  - *Zero-crossings, asynchronous*

# Real-Time LFM2 Frontend Data

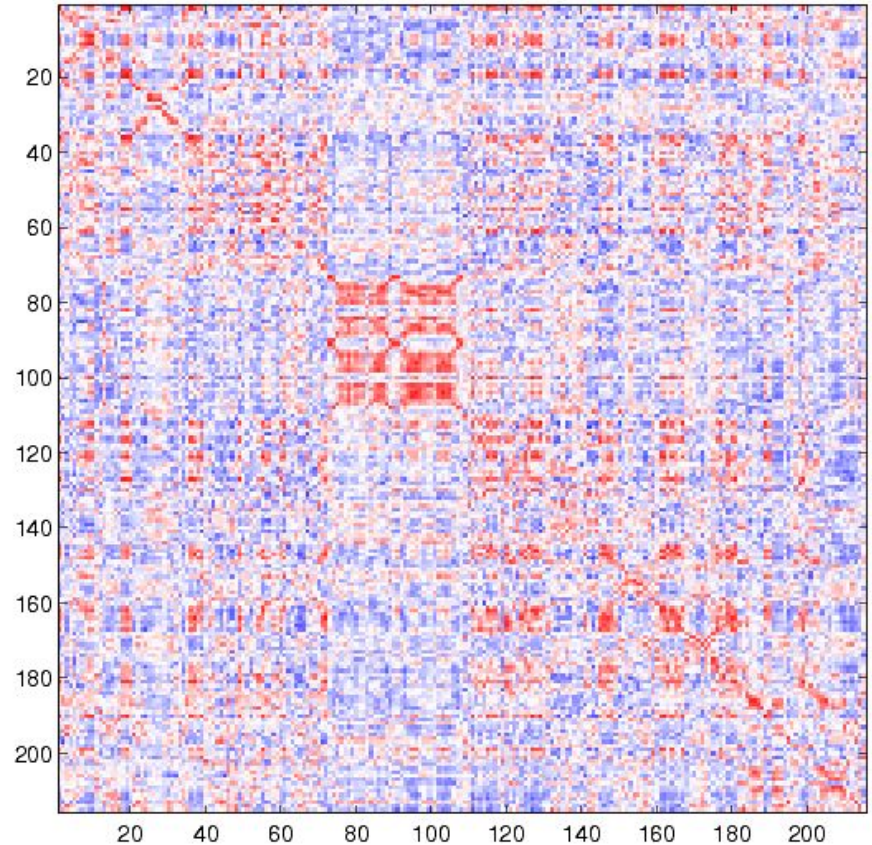


# LFM2 Kernel Computation

training vs. training



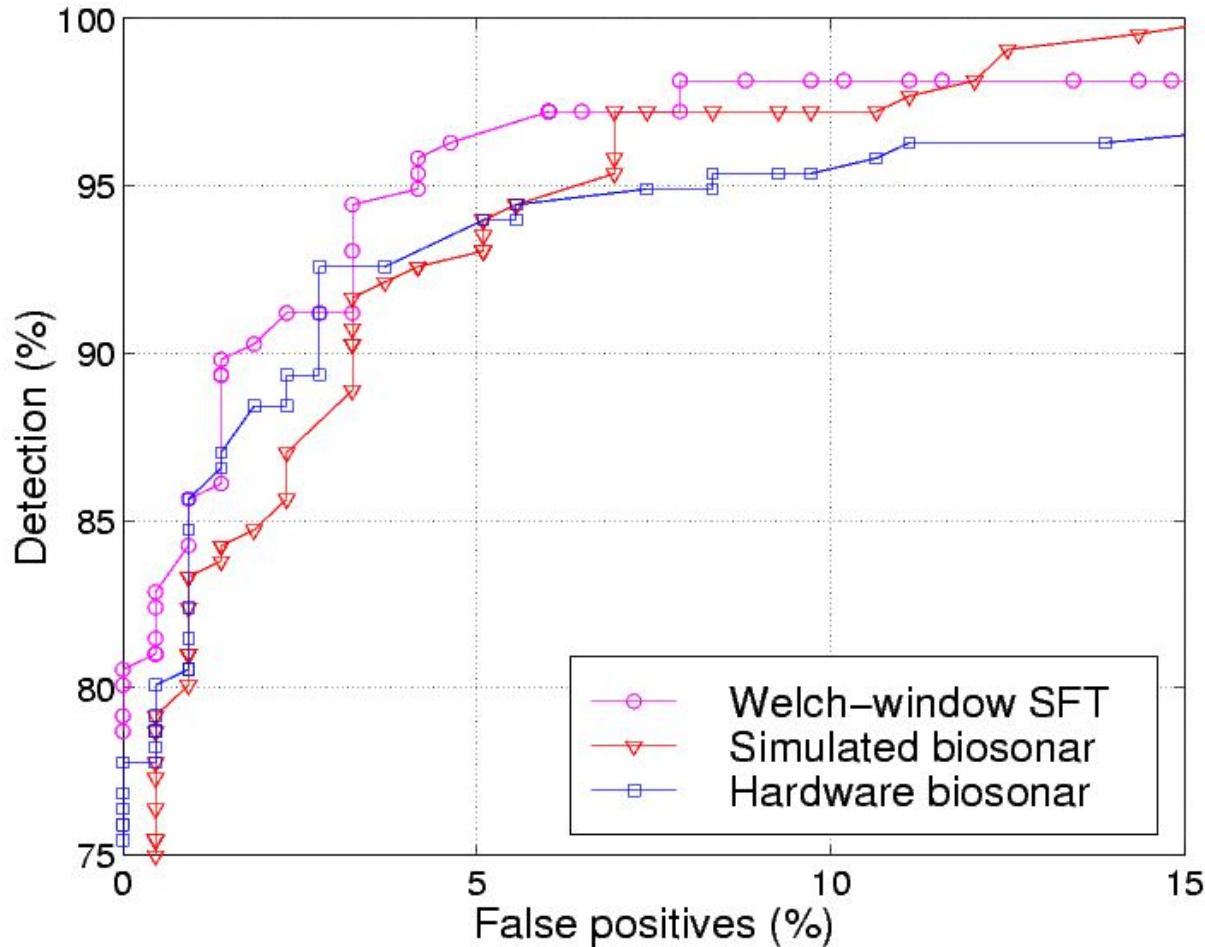
training vs. test



- *Inner-product based kernel is used for SVM classification*
- *PCA is used to enhance features prior to SVM training*

# Measured vs. Simulated Performance

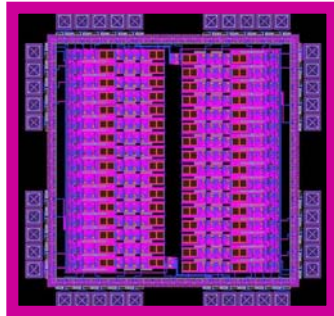
## *LFM2 mines/non-mines classification*



*ROC curve obtained on test set by varying the SVM classification threshold*

- Hardware and simulated biosonar system perform close to the Welch STF/NN classification benchmark.
  - *Classification performance is mainly data-limited.*
- Hardware runs in real time.
  - *2msec per ping.*
  - *50 times faster than simulation on a 1.6GHz Pentium 4.*

# System Integration

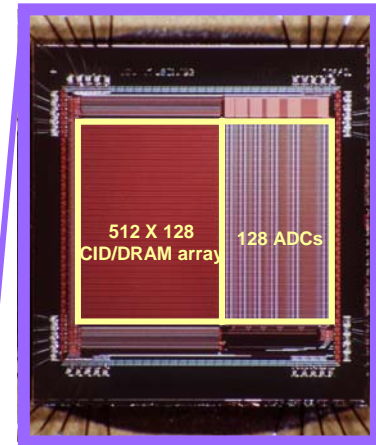


## Frontend

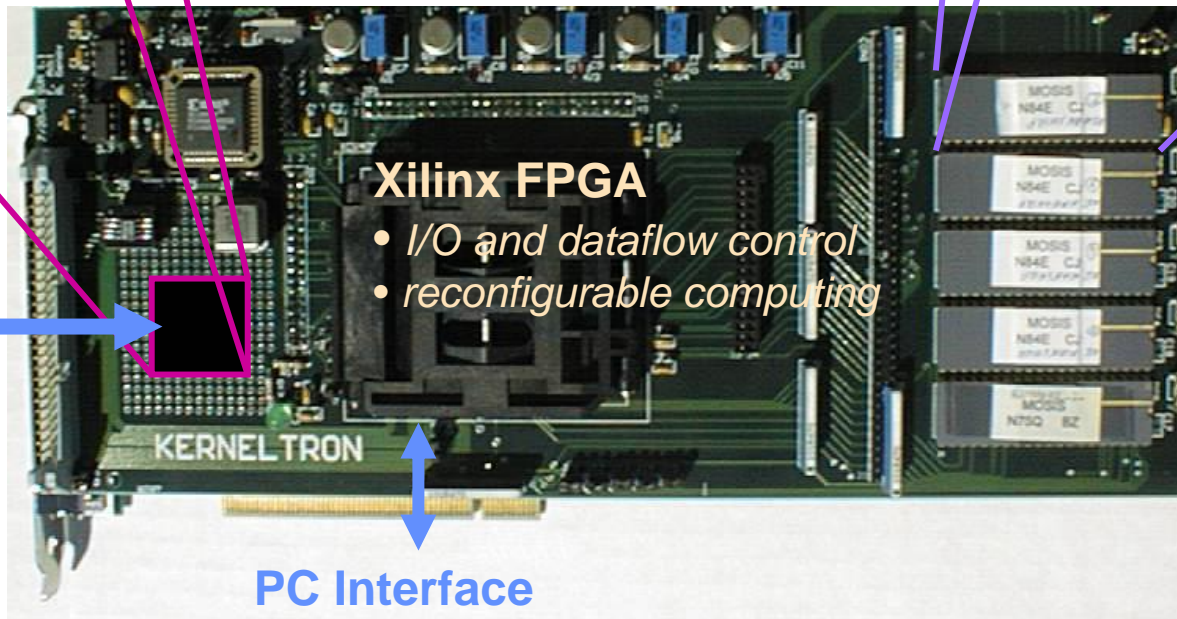
- 32 channels
- 100Hz-200kHz
- Smart A/D

## Kerneltron SVM

- $10^{10}$ -  $10^{12}$  MAC/s
- internally analog
- externally digital



Analog  
Sonar  
Input



## Xilinx FPGA

- I/O and dataflow control
- reconfigurable computing

## PC Interface

- classification output; digital sonar input
- programming and configuration

# Conclusions

- **Parallel charge-mode and current-mode VLSI technology offers efficient implementation of high-dimensional kernel machines.**
  - Computational throughput is a factor 100-10,000 higher than presently available from a high-end workstation or DSP, owing to a fine-grain distributed parallel architecture with bit-level integration of memory and processing functions.
  - Ultra-low power dissipation and miniature size support real-time autonomous operation.
- **Applications include unattended adaptive recognition systems for vision and audition, such as video/audio surveillance and intelligent aids for the visually and hearing impaired.**