高性能計算基盤

－ High Performance Computing Platforms－
#8

Analog Computing Implementations in Machine Learning

Renyuan Zhang

2020/06/25

NAIST

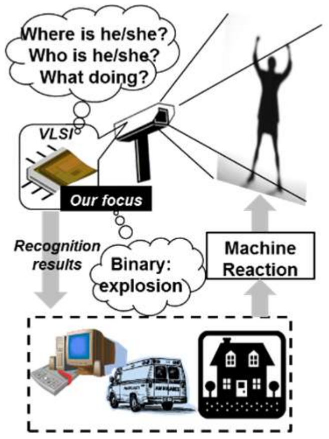
#1
Why consider AI (VLSI) chips?

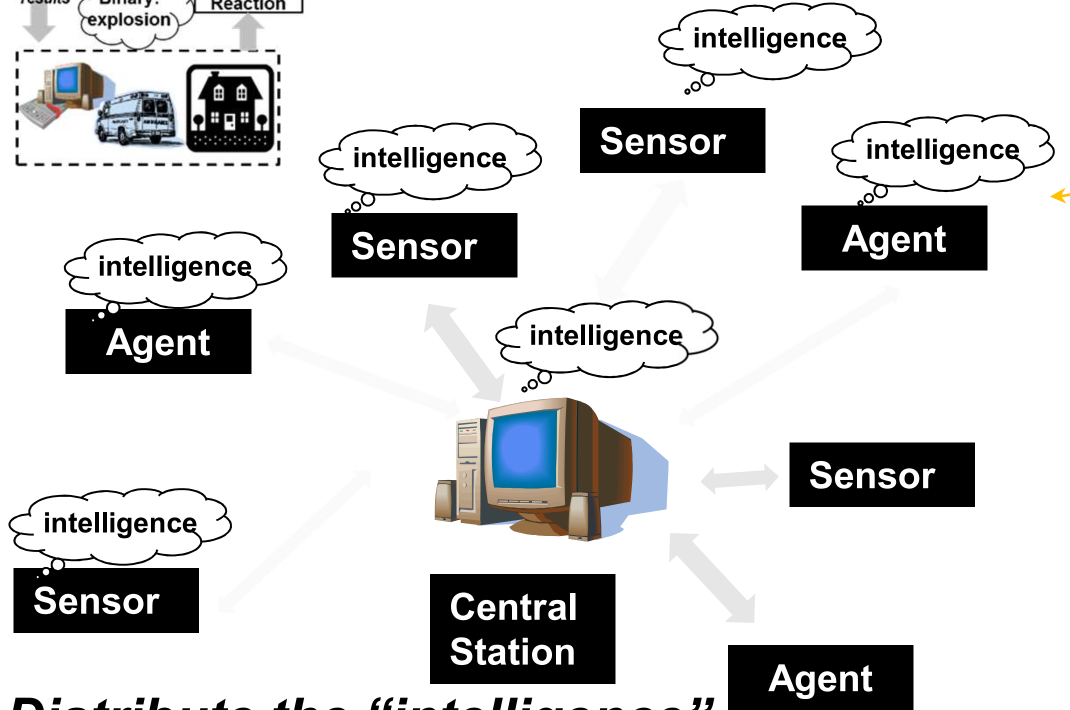
Efficient Processor in IoT

Smart chips: VLSI implementations of machine learning (on-chip learning)

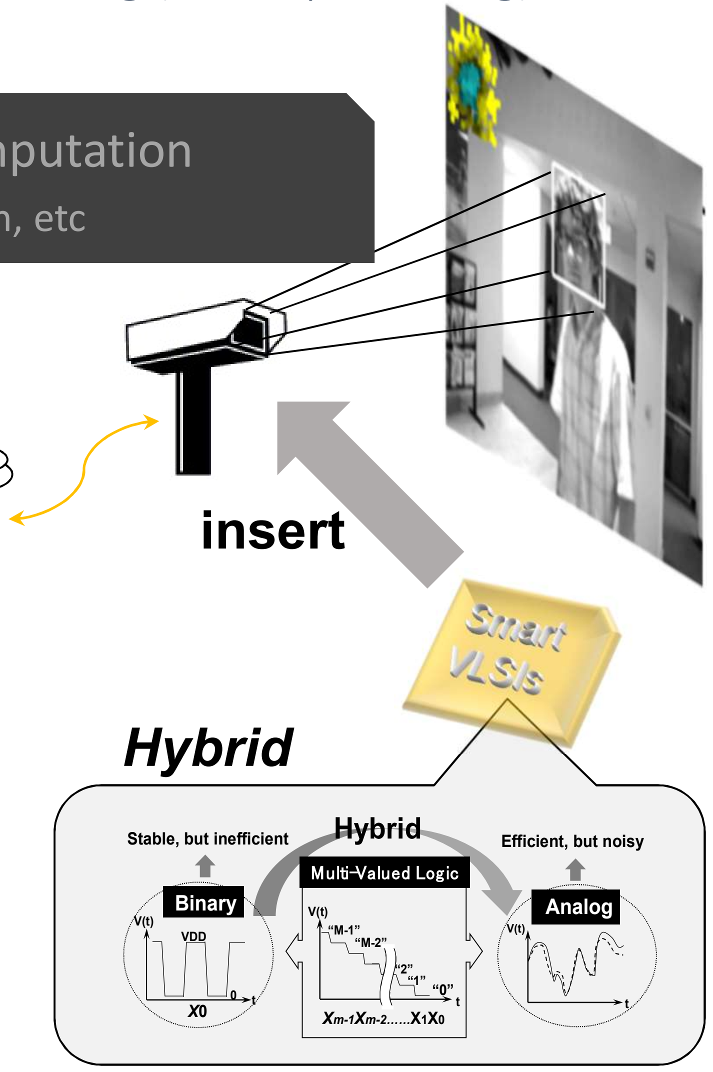
Challenge: ultra-efficient computation

Scale, speed, energy, interconnection, etc





Distribute the "intelligence"



How smart is our life?

- **Smart life:** Through advanced equipment, human being controls the life even the world easily. But what if the environment could respond smartly and automatically to individuals' needs and wishes? [M. Grady et al., *Science*, 2012]

Life style is changing

- Natural & Wild Life (hundreds years ago): **do everything by hands**
- Mechanical life (100 years): **do something by driving machine**
- Automatic life (50 years): **Some machines operate without driving**
- Smart life (soon?):

smart and soft agent like a bunch of stewards and servants

Activate outside lights and gate lock
Set temperature at 66°
Lower volume during commercial
Order 2 gallons milk from annsgroceries.com

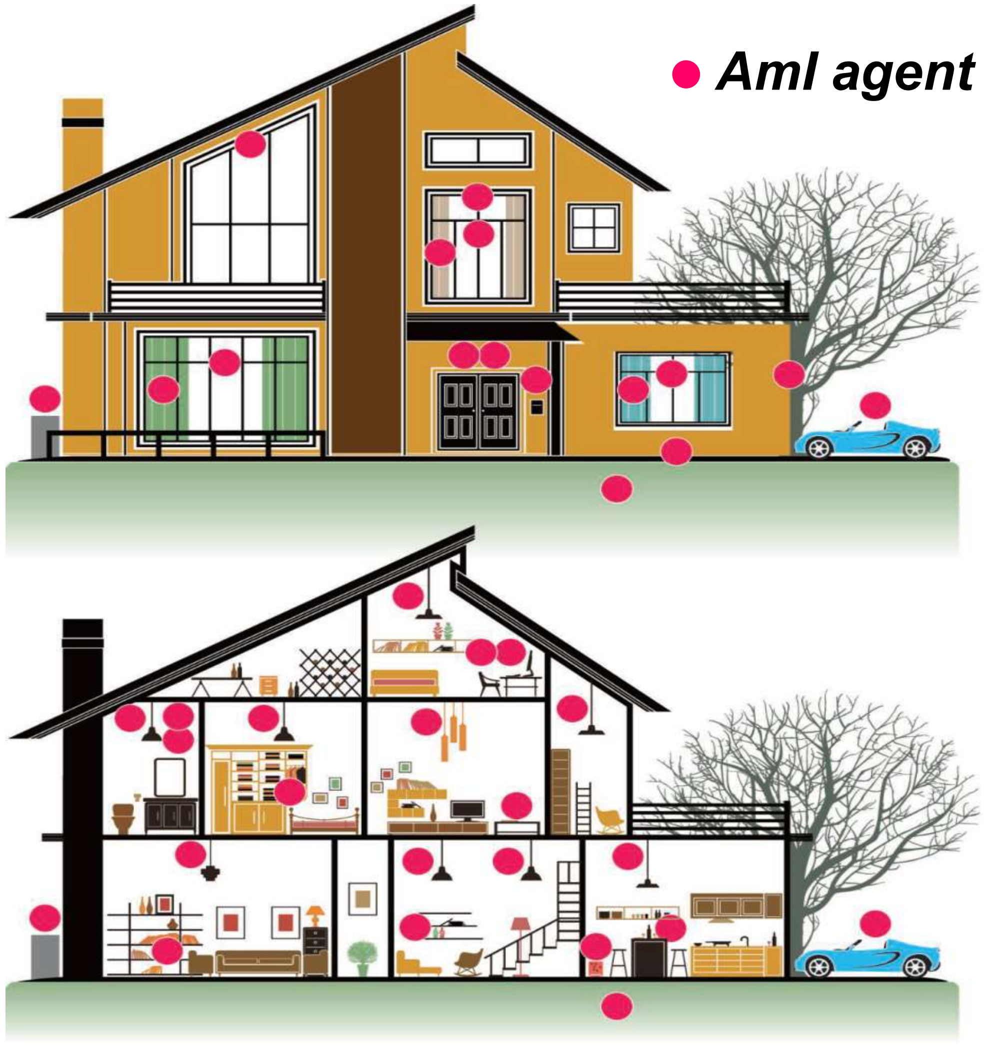
An example

- **Smart home**

D. Cook, *Science*, 2012

Outer of house

inner of house

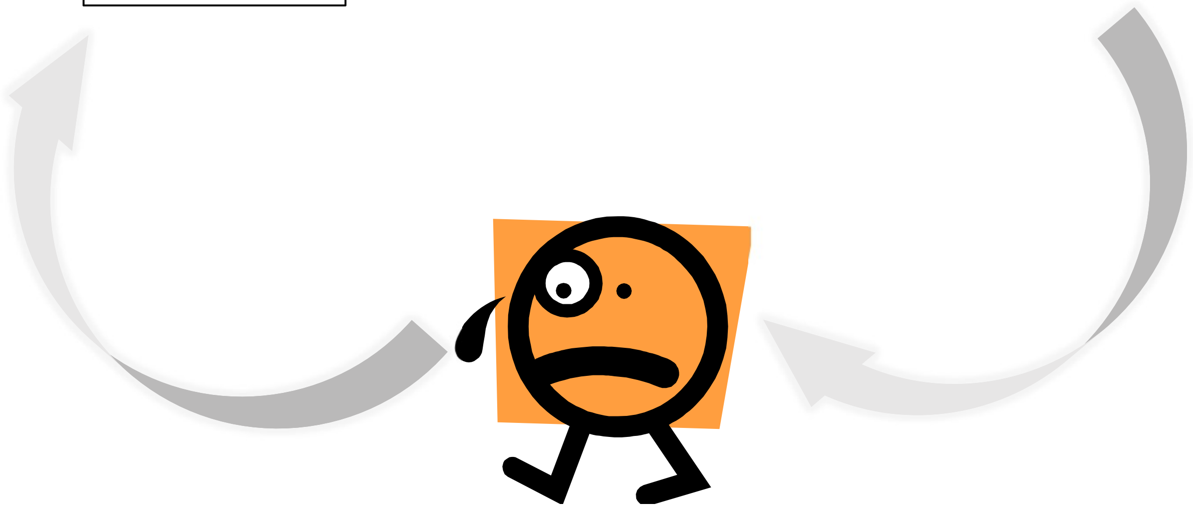


To make environment sufficiently smart

- **Key point is not automatic response, but make agents “think”.**

Evidence → Decision → Action

Evidence → Rules → Decision → Action



Human reaction

IoT hardware on VLSI side

A large number of various VLSI devices are required

	Power	**size**	**applications**
Stationary devices	Watt	Large	Displayer, Monitor, Mechanical driver, Stationary Sensor…
Nomadic devices (Body Area Network)	Mill Watt	Small/tiny	Mobile ~, Wearable ~, Body-Embedded ~
Autonomous transducers	Micro Watt	tiny	Communication devices

Central Processing Station (computer?)

H. De Man, "Ambient Intelligence: Gigascale Dreams and Nanoscale Realities", *IEEE Int. Conf. Solid-State Circuits*, vol. 1, pp 29-35, Feb. 2005.

#2
How analog computing effects AI?

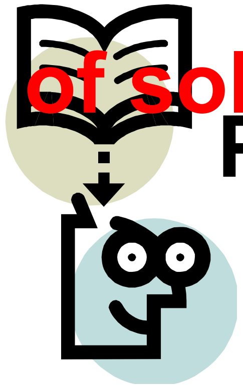
#2.1 Fully Parallel Analog VLSIs for Implementing Machine Learning

VLSI On-chip Training

- **What is “on-chip training”?**

$$253 \div 11 = ?$$

$$10010111 + 11011 = ?$$



Rules of solution have already been mastered

Possible

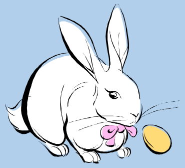
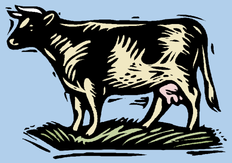
but hard

Faster

Reliably

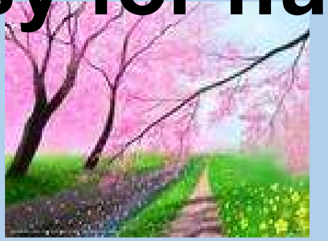
VLSI On-chip Training

- **What is “on-chip training”?**

Which is caw?

Easy for human, difficult for computer.



Is it beautiful view?

No existing rules of solution for computer.

Is he a bad man?

VLSI On-chip Training

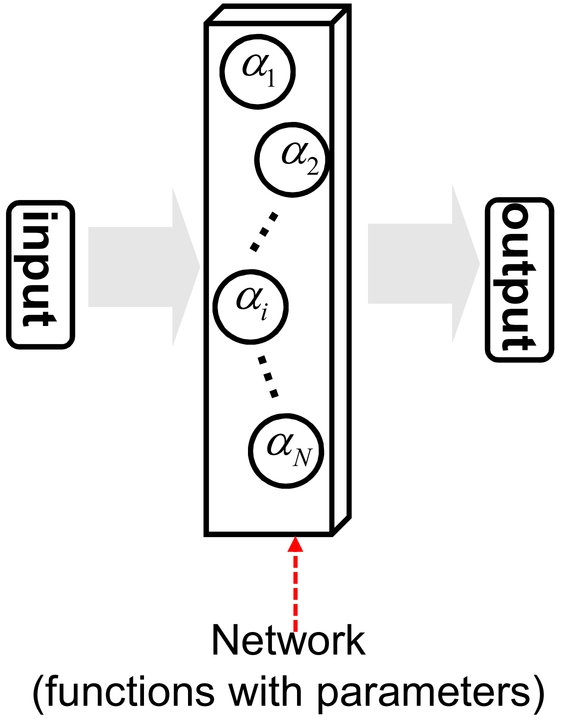
- **What is "on-chip training"?**

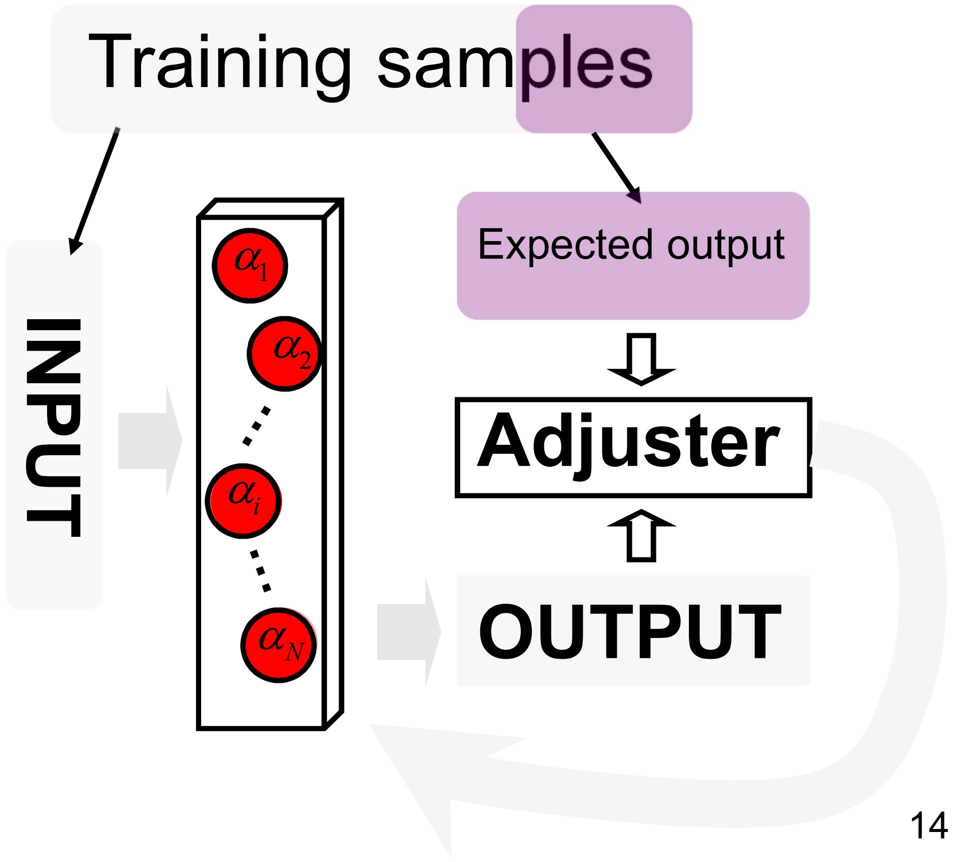
Give some learning examples, make VLSI system pursue the solving rules by itself.

	Processing elements	Element size	Energy use	Processing speed	Style of computation	Fault tolerant	learns	Intelligent, conscious
	10^{14} synapses	$10^{-6}m$	$30W$	$100Hz$	Parallel, distributed	yes	yes	usually
	10^{8} transistors	$10^{-6}m$	$30W$ (CPU)	$10^{9}\,Hz$	Serial, centralized	no	A little	Not (yet)

What is machine learning?

[Wikipedia] To learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.





Why analog?

Fast calculation (real-time)

Smaller chip area

Fully parallel cellular architecture can be built

Non-boolean

Non-clock based

Error-tolerant

General review

- **VLSI implementation strategies**

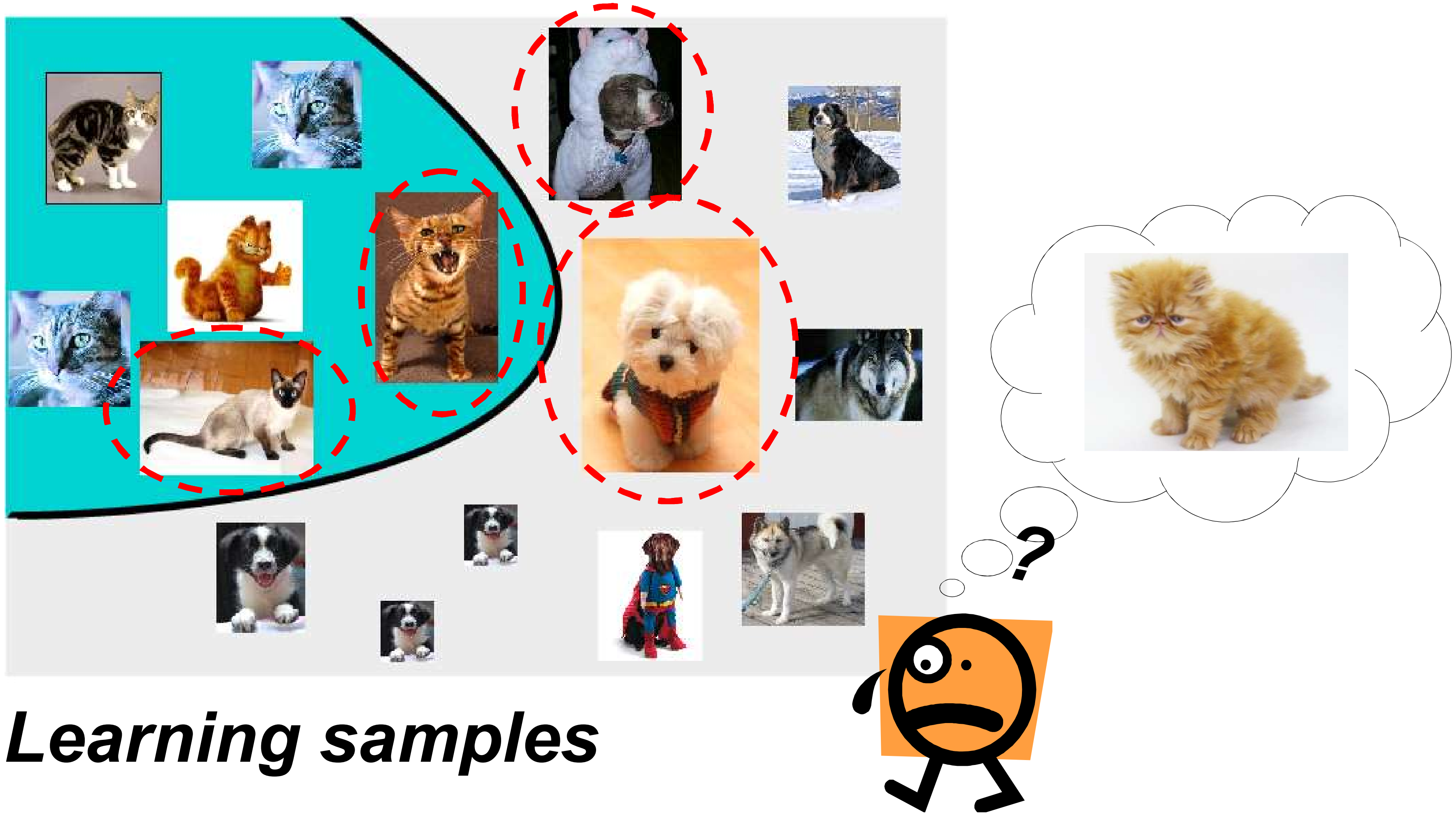
	Hardware	Step control	Speed	Chip size	Flexibility
Fully-serial	Digital	*Clock-based iteration*	☹	😐	🙂
Partially-parallel	Analog	*Clock-based iteration*	🙂	🙂	☹
Hyper-Parallel	Analog	*Non-clock Freely-feedback*	🙂🙂	🙂	😐

Outline

Analog implementations of them:

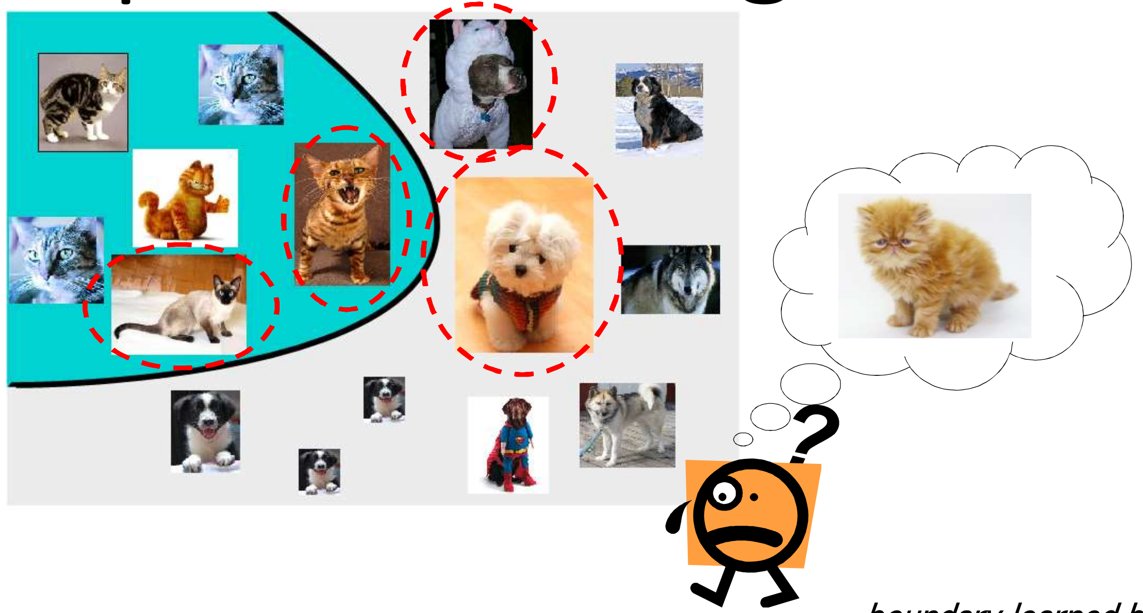
- ➢**Support Vector Machine**
- ➢K-Quasi-Centers clustering
- ➢On-line learning strategies
- ➢Data domain description
- ➢Intel Project

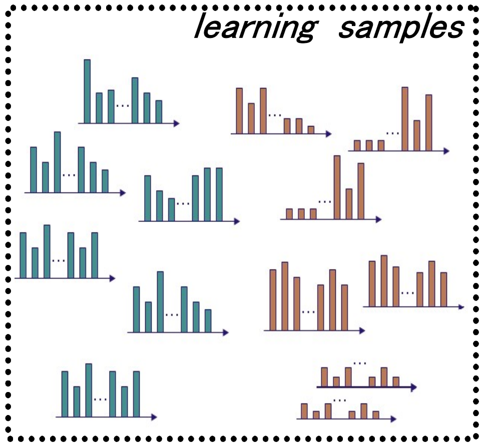
Support Vector Machine (SVM)

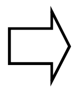


Learning samples

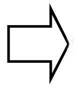
SVM for pattern recognition

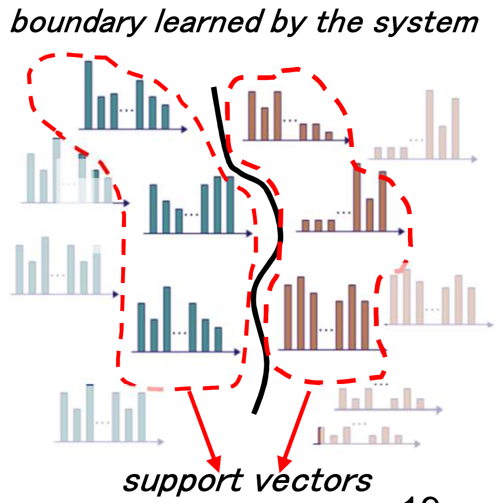




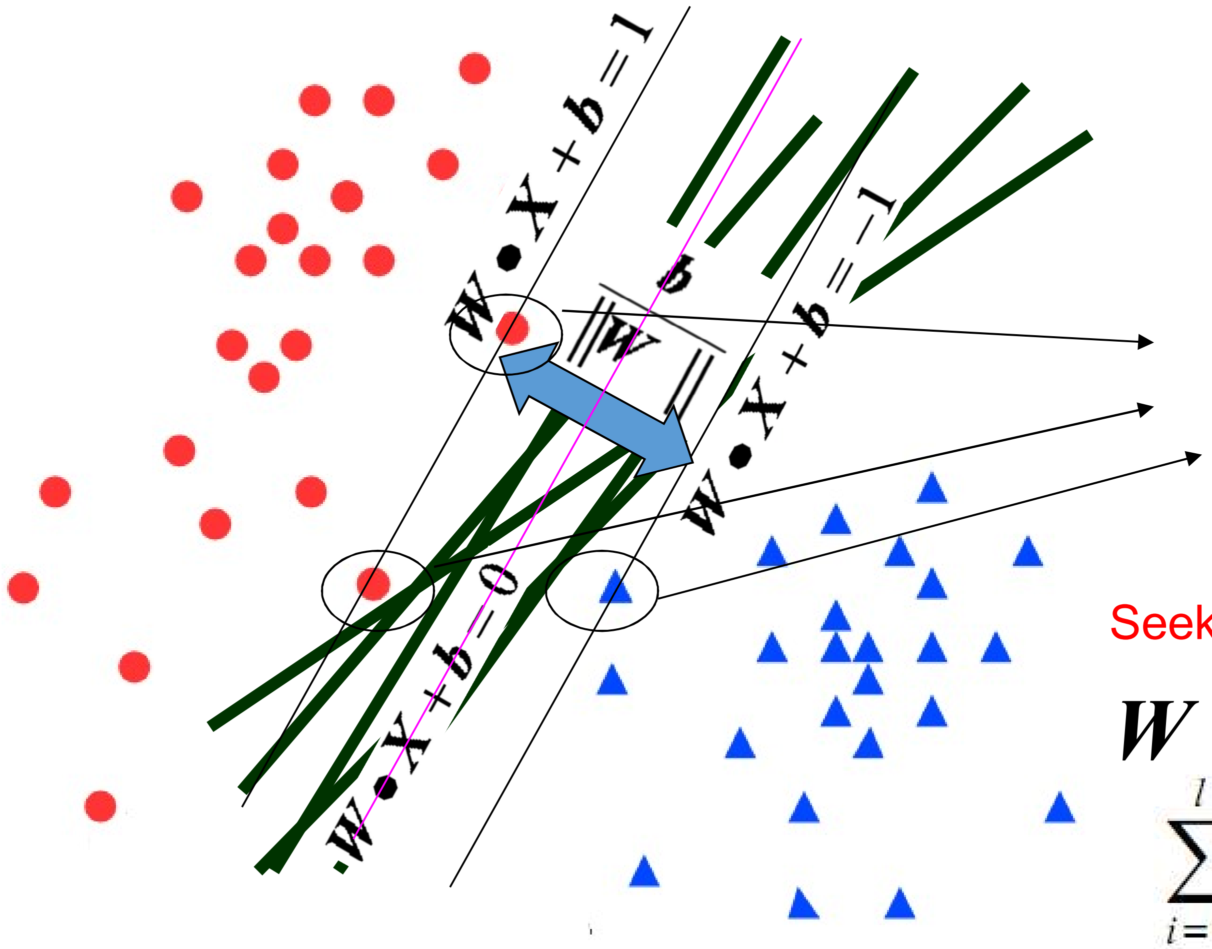


SVM learning & classifying processor





SVM for pattern recognition



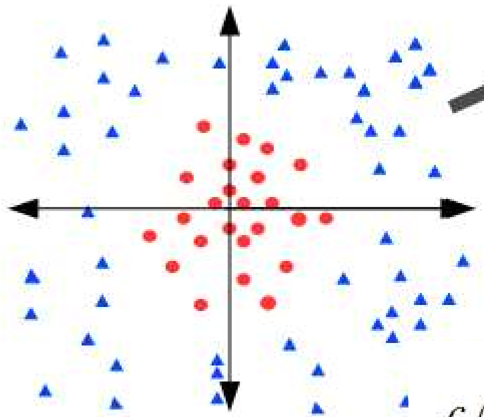
Seeking the hyperplane

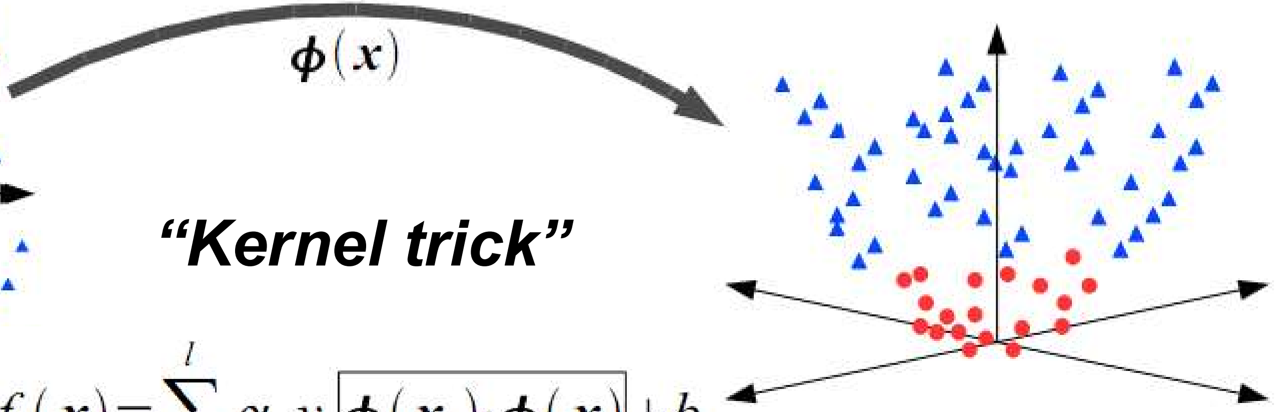
$$W \bullet X + b = 0$$

$$\sum_{i=0}^{l} \alpha_i y_i \boldsymbol{x}_i \cdot \boldsymbol{x} + b$$

SVM with "Kernel trick"

Non-linear





"Kernel trick"

$$f(\boldsymbol{x})=\sum_{i=1}^{l}\alpha_i y_i \boxed{\boldsymbol{\phi}(\boldsymbol{x}_i)\cdot\boldsymbol{\phi}(\boldsymbol{x})}+b$$

$$\downarrow$$

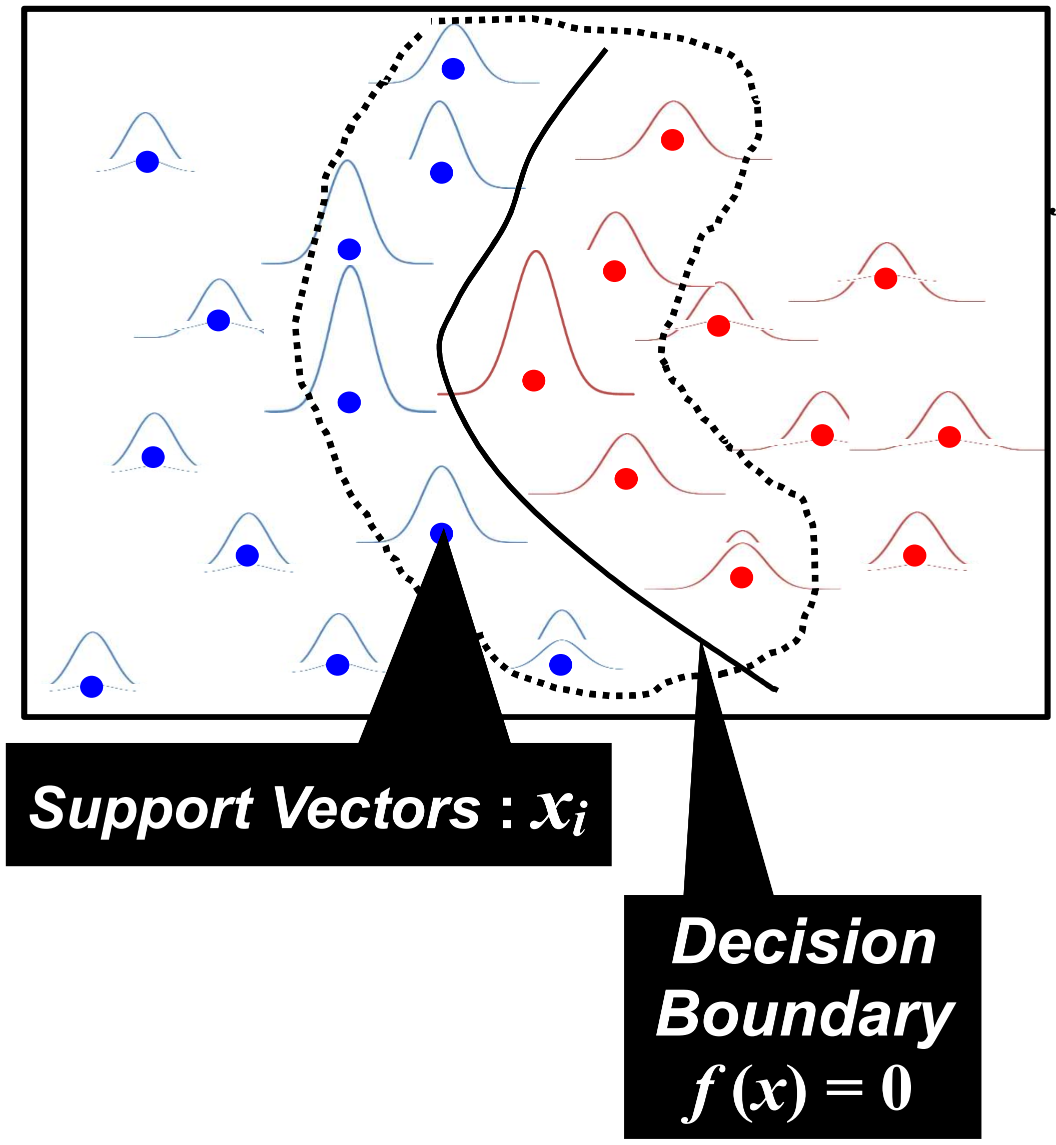
$$f(\boldsymbol{x})=\sum_{i=1}^{l}\alpha_i y_i \boxed{K(\boldsymbol{x}_i,\boldsymbol{x})}+b$$

K = ? Linear X1-X2 ? or quadratic ?

Gaussian Kernel

$$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\exp(-\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2/2\sigma^2)$$

SVM learning operation



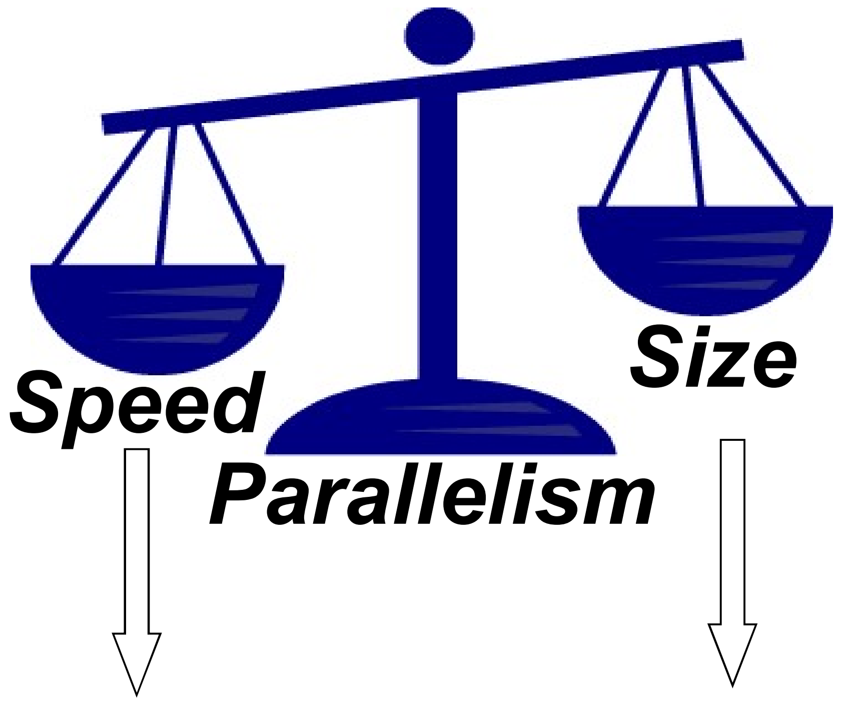
■ ***Classification***

$$f(x) = \sum \underbrace{\alpha_i}_{\text{Weight}} \underbrace{y_i}_{\text{Class (1 or -1)}} \underbrace{e^{-|\mathbf{X}_i - \mathbf{X}|^2/\sigma}}_{\text{Kernel}}$$

■ ***Learning***
= determining weights

$$\alpha_i \leftarrow 1 - y_i \sum_{j \neq i} y_j \alpha_j e^{-|\mathbf{X}_i - \mathbf{X}_j|^2/\sigma}$$

Analog Implementation of SVM



1. Complex computation: $A \cdot e^{-\frac{(x_{i1}-x_{j1})^2+(x_{i2}-x_{j2})^2+(x_{i3}-x_{j3})^2+(x_{i4}-x_{j4})^2+(x_{i5}-x_{j5})^2+(x_{i6}-}{\sigma}}$

$\longrightarrow$ ***Analog circuits to generate Gaussian function***

2. Learning operation: large number of numerical iterations

$\longrightarrow$ ***Fully parallel architecture to avoid clock-based iterations***

Analog Implementation of SVM

(with Gaussian function kernel)

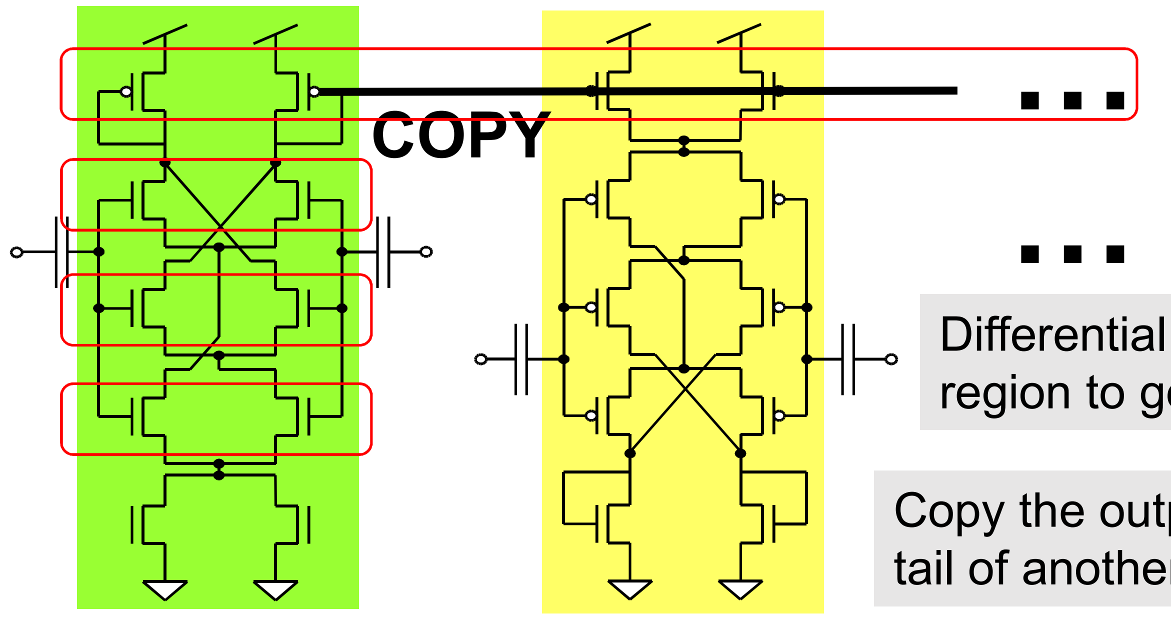
	Parallelism	Chip area	Learning speed	Dimensi ons	Chip measurement
S.-Y. Peng et al. (2008)	*Fully parallel*	$\propto N^2$	*One clock*	2	*N.A*
K. Kang et al. (2010)	*Row-parallel*	$\propto N$	$\propto N \times i$	2	*Available*
Target of this work	*Fully Parallel*	$\propto N$	*One clock*	*64*	*Available*

***N**: number of learning samples*   *i: number of numerical iterations*

[1] S.-Y. Peng, B. A. Minch, and P. E. Hasler: *Proc. Int. Symp. Circuits Syst.*, 2008, pp. 860 - 863.
[2] K. Kang and T. Shibata: *IEEE Trans. Circuits Syst*., Vol. 57, no. 7, pp. 1513 – 1524 (2010).

Gaussian-generation circuit

Traditional Gaussian-generation circuit: Bump circuit *



Differential pairs in sub-threshold region to generate a "bump"

Copy the output from an element to the tail of another, to expend the dimensions

$$e^{-(x_{i1}-x_{j1})^2} \times e^{-(x_{i2}-x_{j2})^2} \times \cdots$$

Error by mismatch 5%

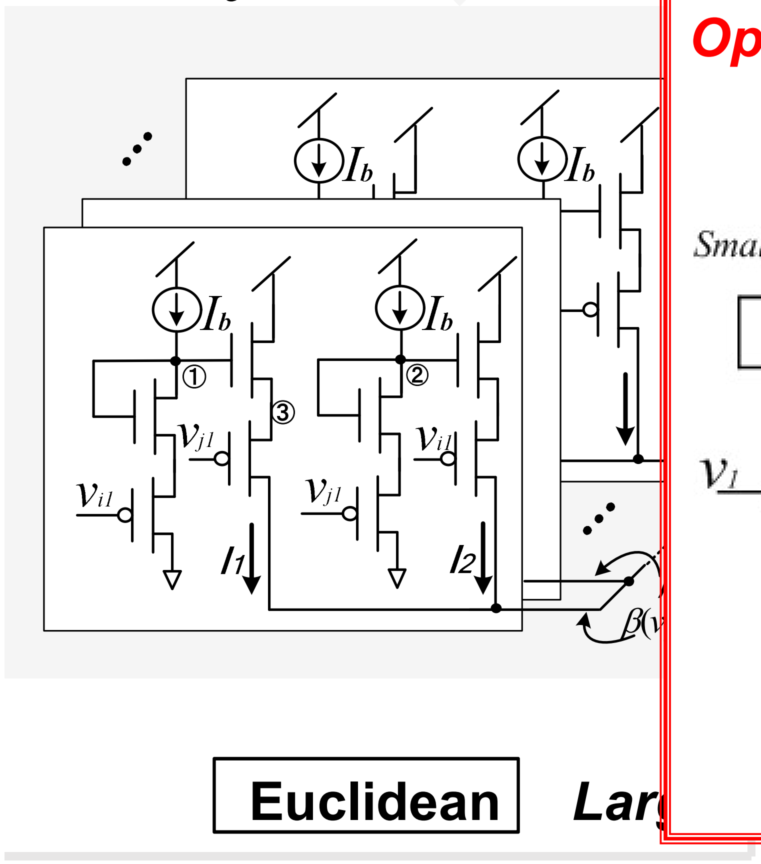
Error by mismatch 5% → **2270%! (64D)**

*K. Kang and T. Shibata: *IEEE Trans. Circuits Syst.*, Vol. 57, no. 7, pp. 1513 – 1524 (2010).

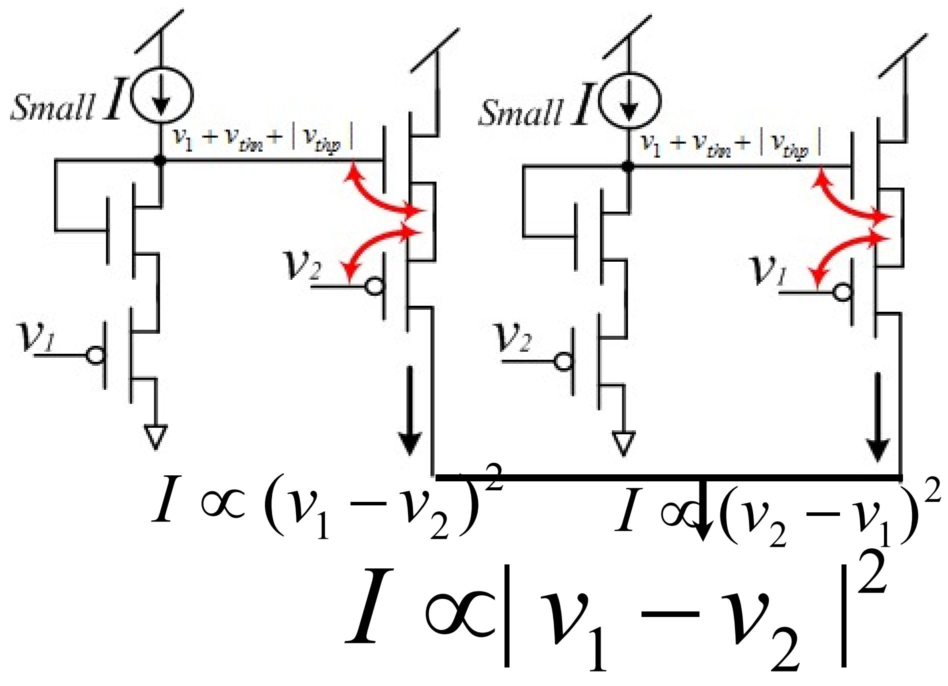
Gaussian-generation circuit

Our Proposed Gaussian-generation circuit

$$I_c \cdot e^{-\frac{(v_{i1}-v_{j1})^2+(v_{i2}-v_{j2})^2+(v_{i3}-v_{j3})^2+......+(v_{i64}-v_{j64})^2}{\sigma}}$$



Operational principle



$$I \propto (v_1 - v_2)^2 \qquad I \propto (v_2 - v_1)^2$$

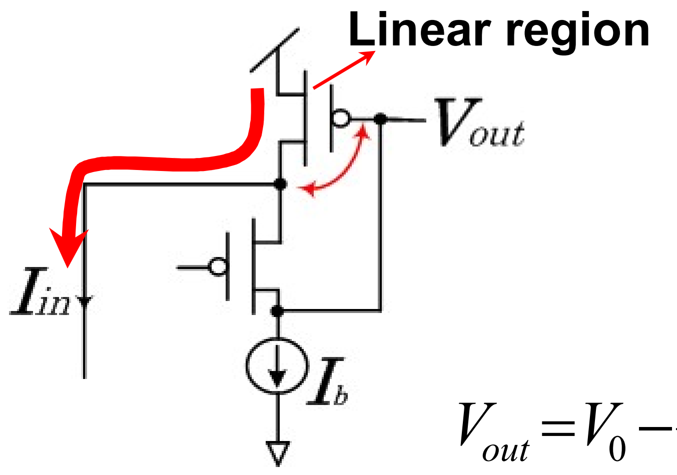
$$I \propto |v_1 - v_2|^2$$

Gaussian-generation circuit

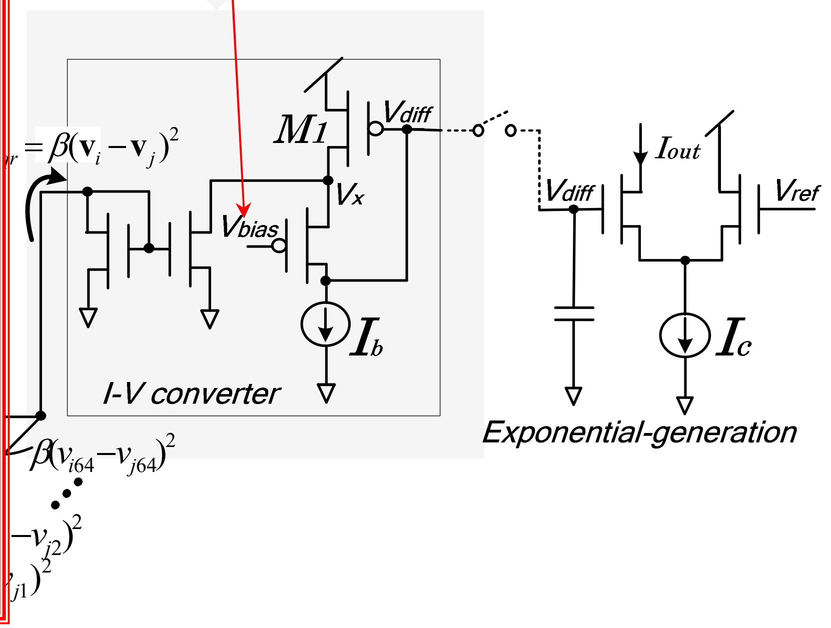
Our Proposed Gaussian-generation circuit

$$-\frac{(v_{i1}-v_{j1})^2+(v_{i2}-v_{j2})^2+(v_{i3}-v_{j3})^2+......+(v_{i64}-v_{j64})^2}{\sigma}$$

Operational principle



$$V_{out}=V_0-\frac{I_{in}}{\sigma}$$



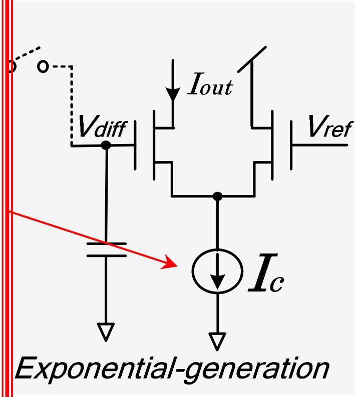
Euclidean ***Large***

Exp. ***Small***

Gaussian-generation circuit

Our Proposed Gaussian-generation circuit

$$I_c \cdot e^{-\frac{(v_{i1}-v_{j1})^2+(v_{i2}-v_{j2})^2+(v_{i3}-v_{j3})^2+......+(v_{i64}-v_{j64})^2}{\sigma}}$$



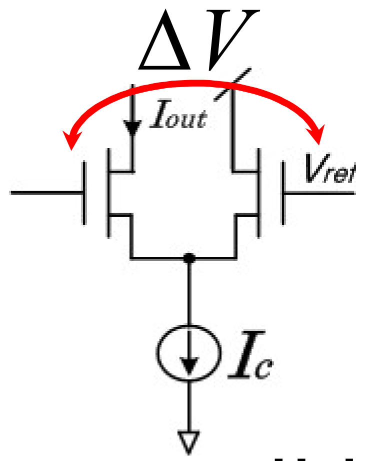
Exponential-generation

s to self-tune I_c

Euclidean ***Large***

Exp. ***Small***

Operational principle



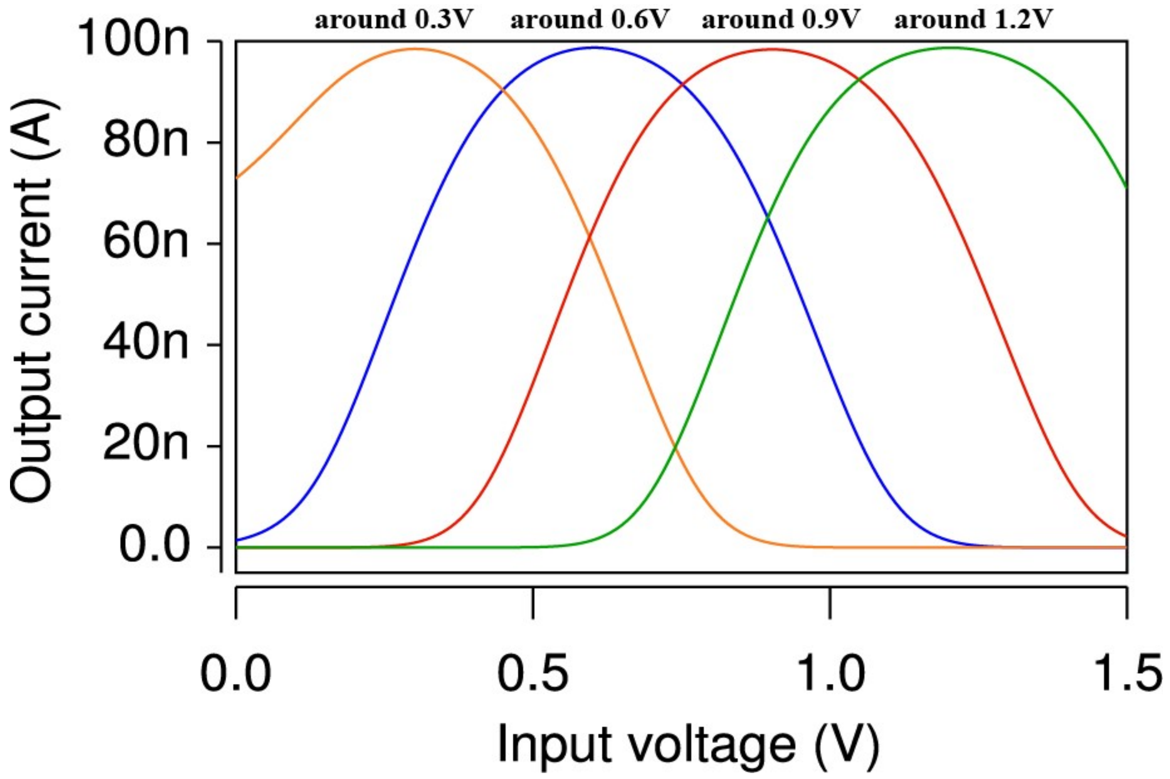
$$I_{out} \approx \frac{I_c}{2} \cdot e^{\frac{q}{kT}\Delta V}$$

Unbalance of differential pair

Gaussian-generation circuit

Performance of Our Proposed Gaussian-generation circuit

Center value of Gaussian function feature can be dynamically programmed



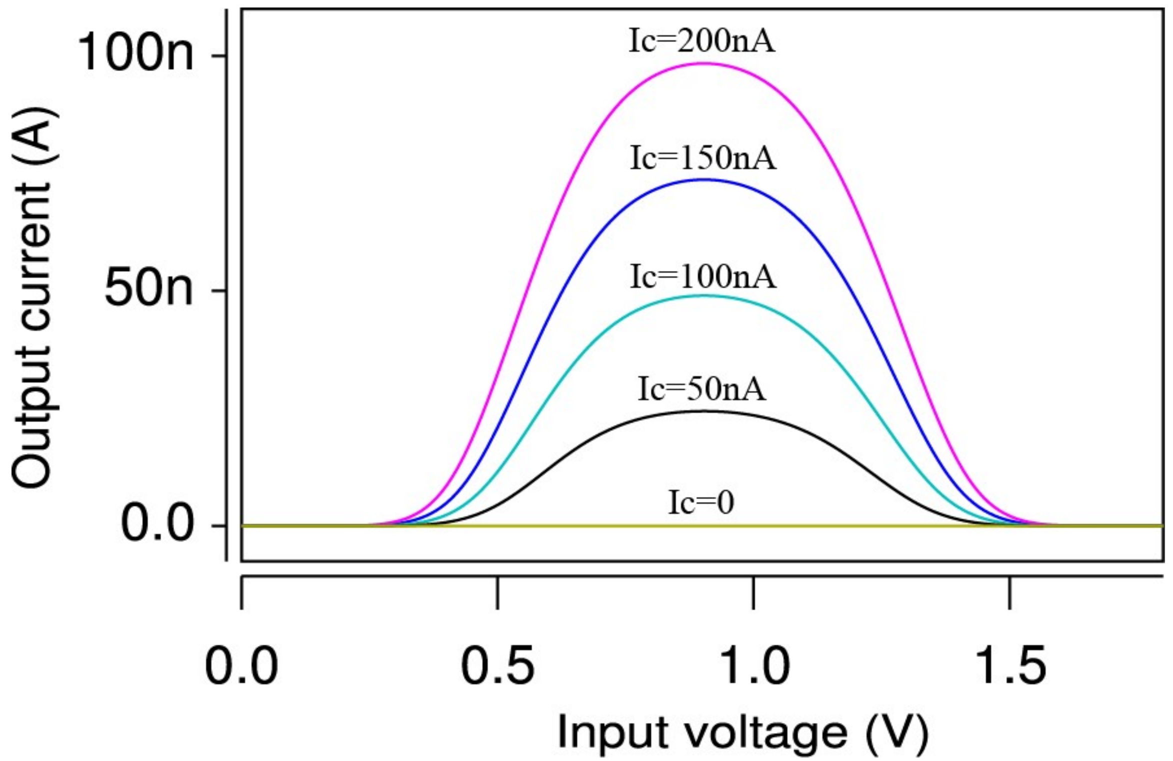
Simulation

$$I_c \cdot e^{-\frac{(v_{i1}-v_{j1})^2+(v_{i2}-v_{j2})^2+(v_{i3}-v_{j3})^2+\ldots\ldots+(v_{i64}-v_{j64})^2}{\sigma}}$$

Gaussian-generation circuit

Performance of Our Proposed Gaussian-generation circuit

Peak-height of Gaussian function feature can be dynamically programmed



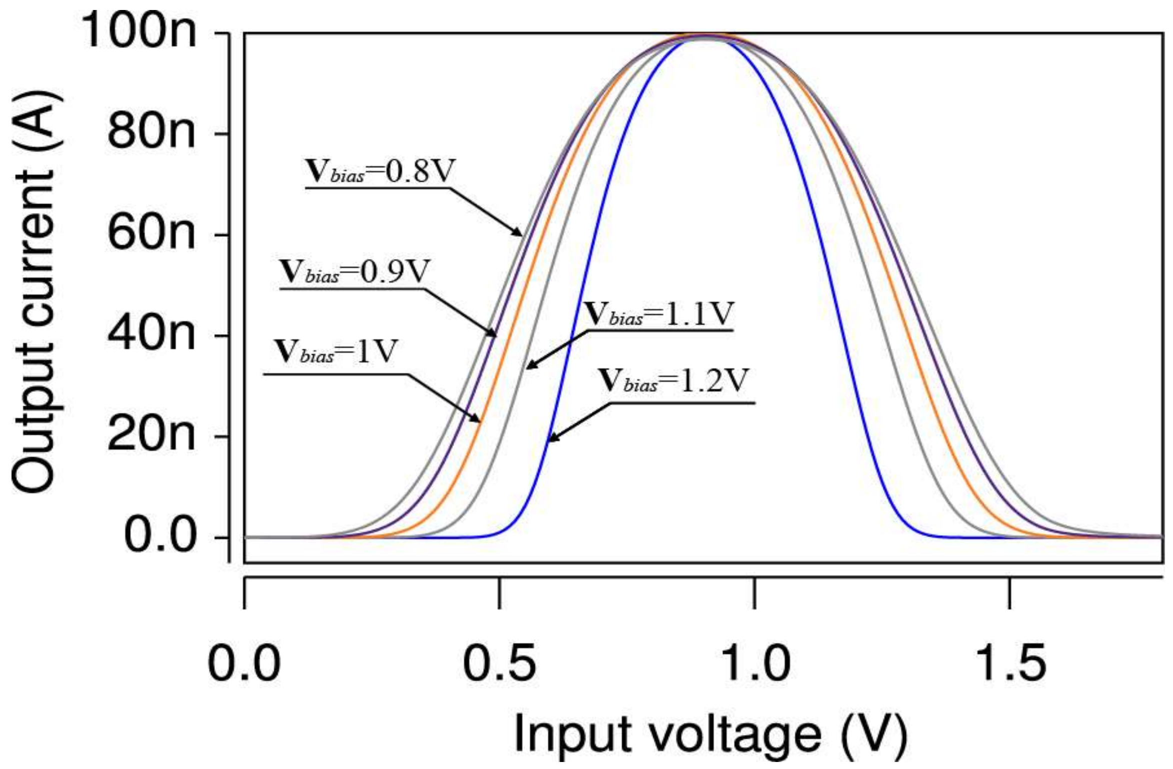
Simulation

$$\boxed{I_c \cdot e^{-\frac{(v_{i1}-v_{j1})^2+(v_{i2}-v_{j2})^2+(v_{i3}-v_{j3})^2+......+(v_{i64}-v_{j64})^2}{\sigma}}}$$

Gaussian-generation circuit

Performance of Our Proposed Gaussian-generation circuit

Width of Gaussian function feature can be dynamically programmed



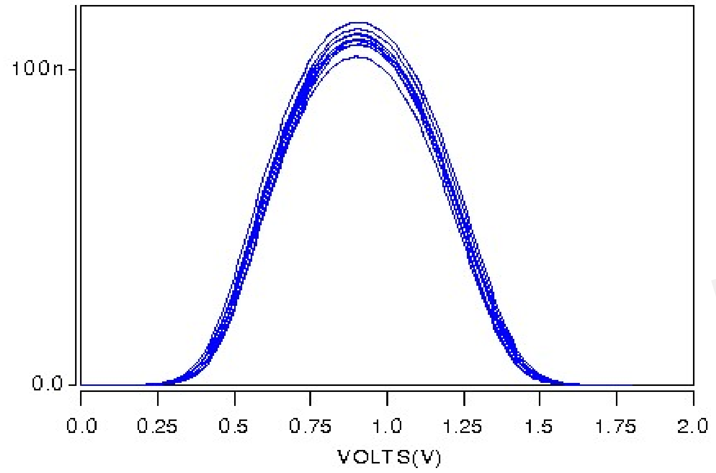
Simulation

$$I_c \cdot e^{-\frac{(v_{i1}-v_{j1})^2+(v_{i2}-v_{j2})^2+(v_{i3}-v_{j3})^2+......+(v_{i64}-v_{j64})^2}{\sigma}}$$

Gaussian-generation circuit

Performance of Our Proposed Gaussian-generation circuit

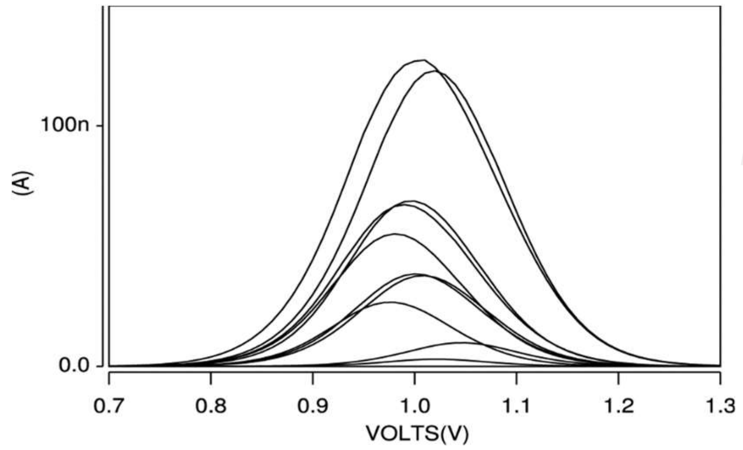
Reliability of proposed Gaussian generation circuit:



Monte Carlo Simulation

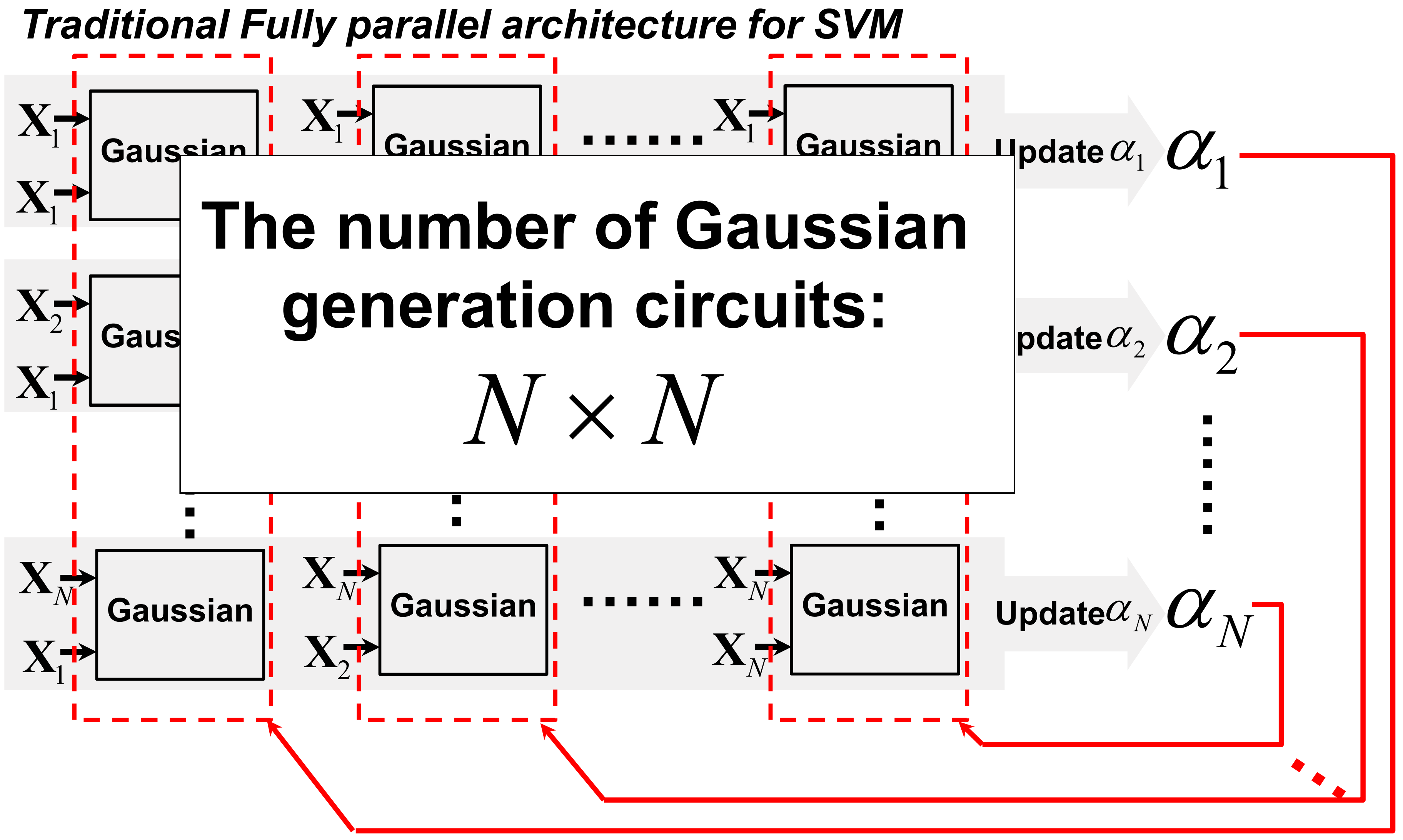
5% variation on Vth

Reliability of traditional Gaussian generation circuit:



Monte Carlo Simulation

Fully parallel architecture for SVM



The number of Gaussian generation circuits:

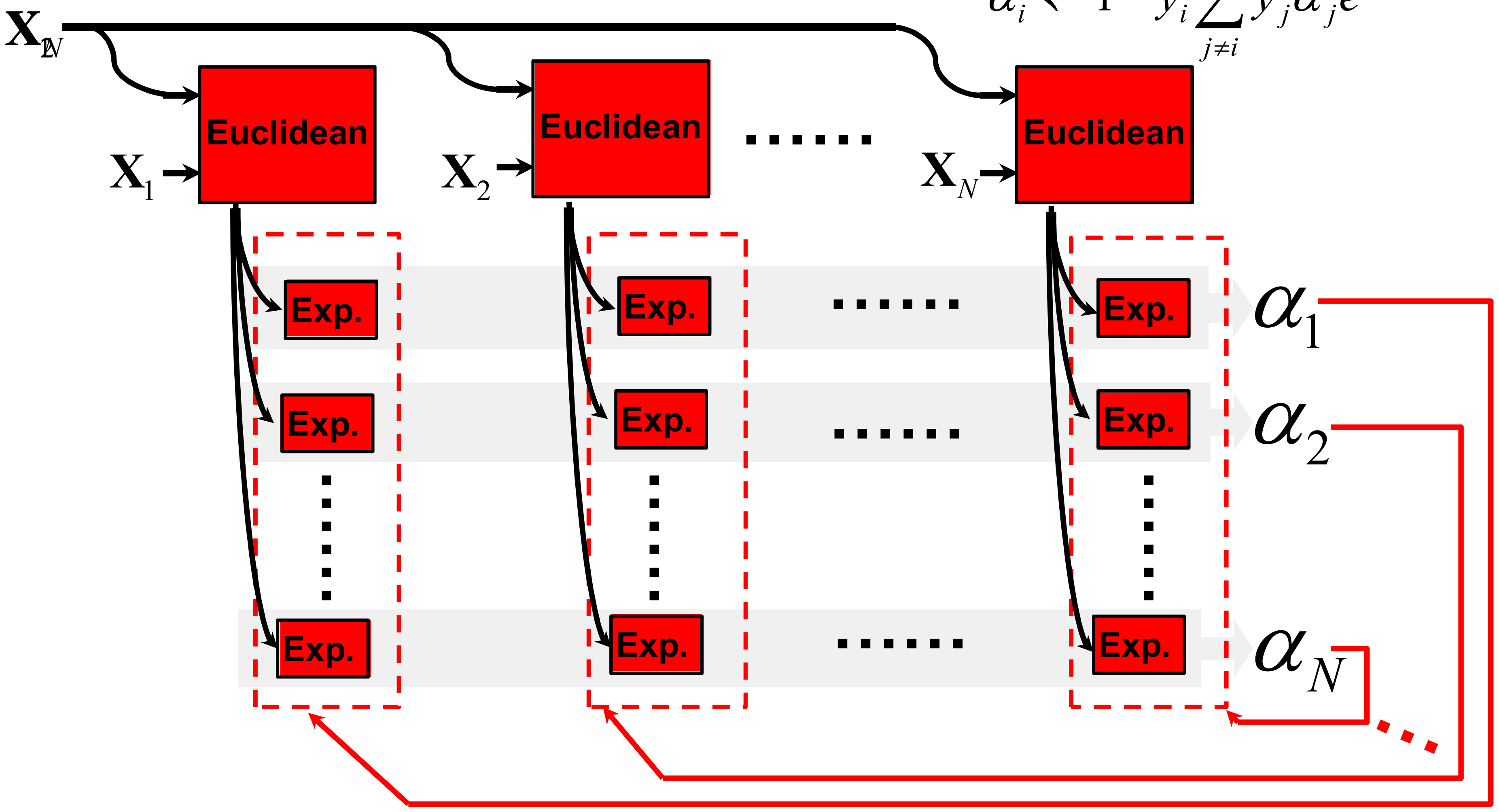
$$N \times N$$

S.-Y. Peng, B. A. Minch, and P. E. Hasler: *Proc. Int. Symp. Circuits Syst.*, 2008, pp. 860 - 863.

Fully parallel architecture for SVM

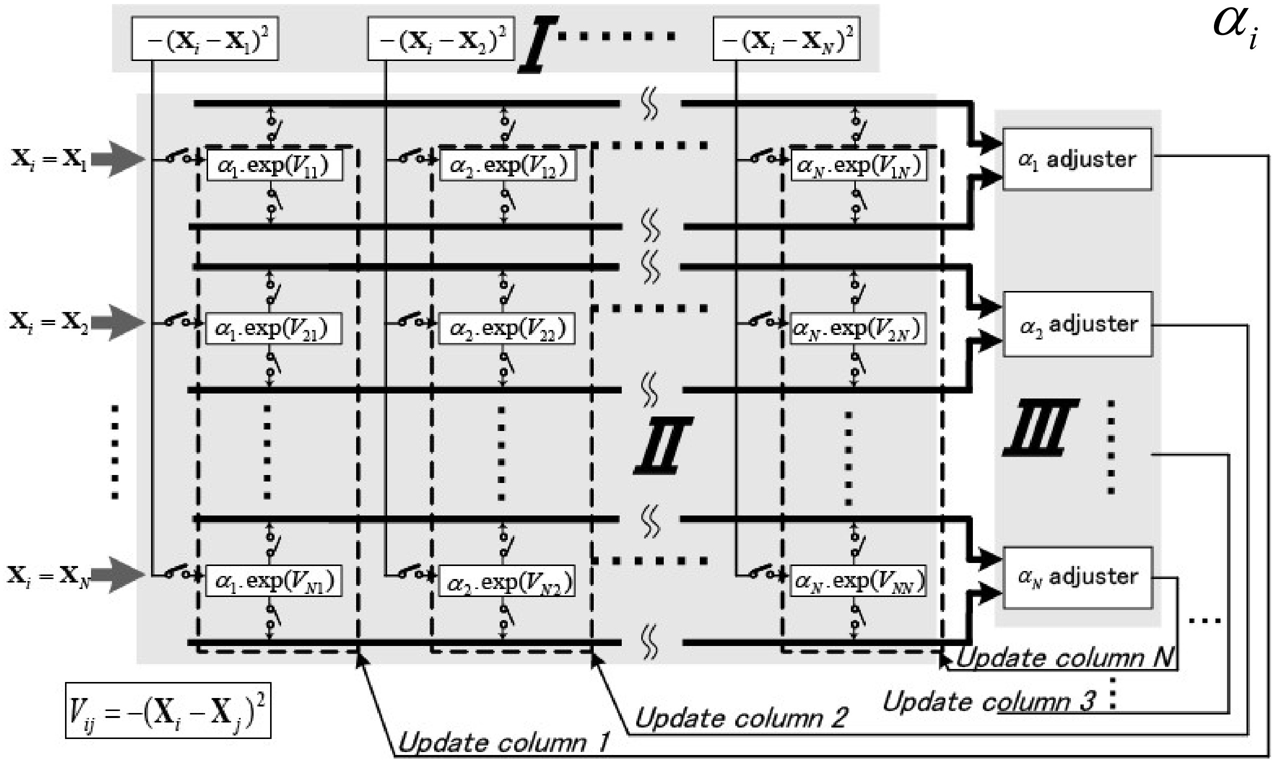
Proposed fully parallel learning architecture for SVM

$$\alpha_i \leftarrow 1 - y_i \sum_{j \neq i} y_j \alpha_j e^{-|\mathbf{X}_i - \mathbf{X}_j|^2/\sigma}$$



Fully parallel architecture for SVM

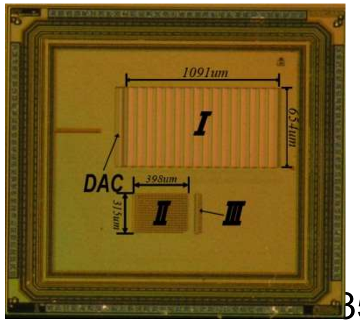
Organization of SVM processor



$$\alpha_i \leftarrow 1 - y_i \sum_{j\neq i} y_j \alpha_j e^{-|\mathbf{X}_i-\mathbf{X}_j|^2/\sigma}$$

Non-iterative based

Freely analog feedback



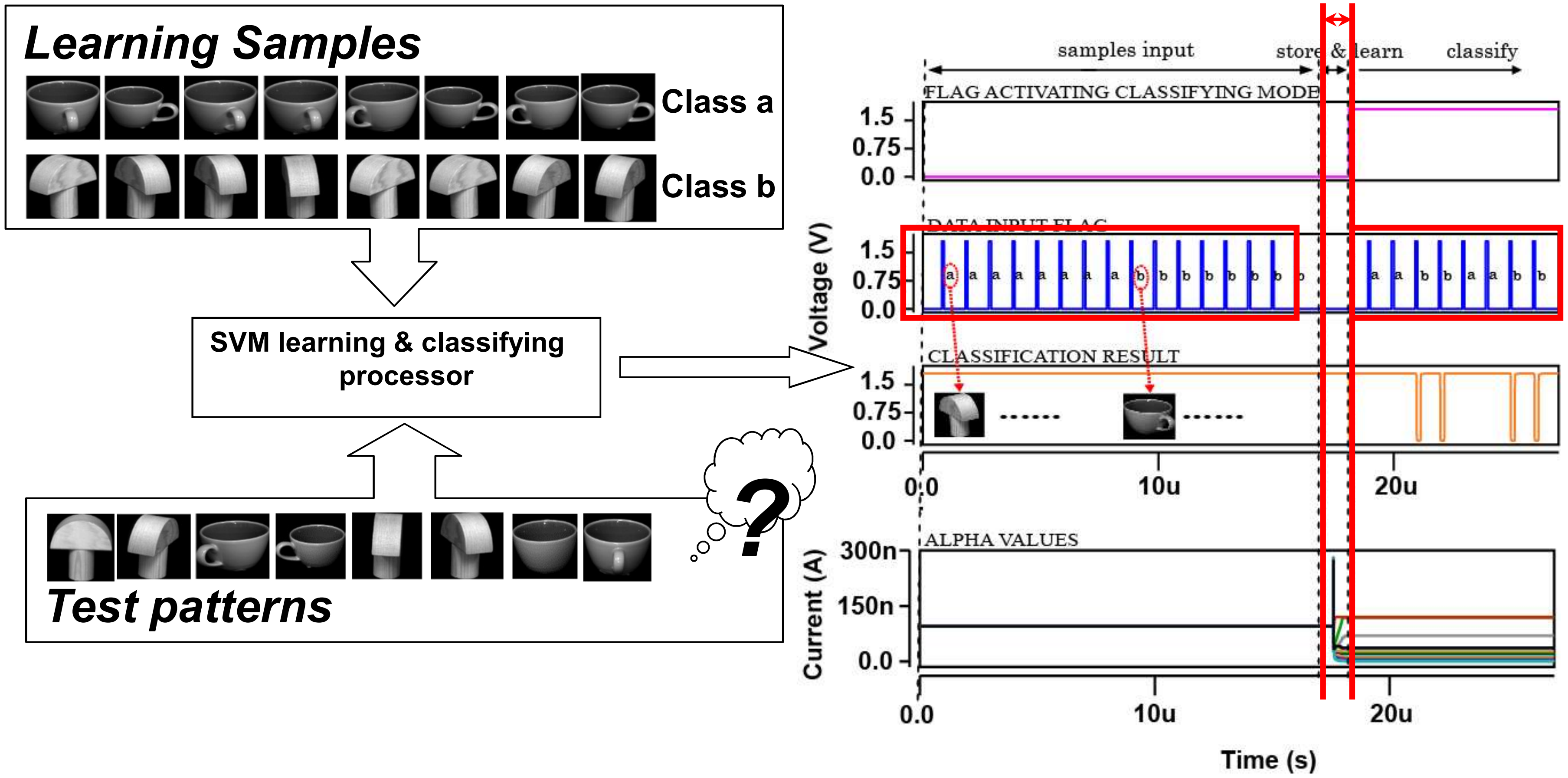
✓Fast learning and self-converging

✓Compact chip-area

Verification of SVM processor

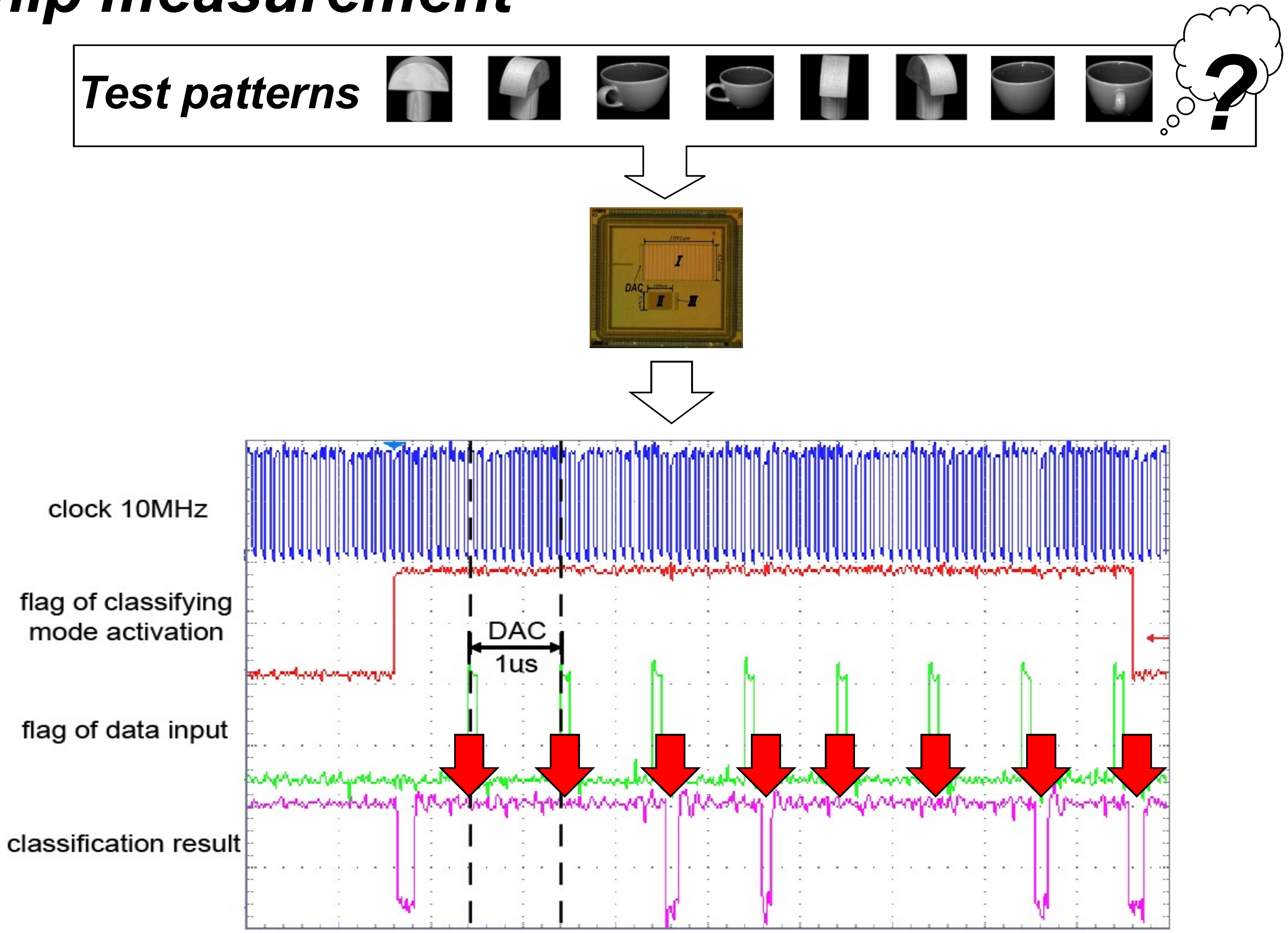
[illegible] DAC session

Simulation



Verification of SVM processor

Chip measurement



Performance of SVM Processor

Comparisons

	[1] ISCAS^{2008}	[2] TCAS^{2010}	This work
Technology	Simulation	$0.18\mu m$ CMOS	$0.18\mu m$ CMOS
Operation	Learning/Classifying	Learning/Classifying	Learning/Classifying
Learning parallelism	Fully parallel	Row parallel	Fully parallel
Kernel function	Gaussian	Gaussian	Gaussian
No. of kernels	$4 \times 4 + 4$	12	18.8*
Input vector	Analog voltage	Digital (8 bits)	Digital (8 bits)
Number of samples	4	12	16
Number of dimensions	2	2	$1 \sim 64$
Learning time (ns)	N/A	$12 \times l \times 60^{\sharp}$	100
Classifying speed (vectors/s)	N/A	8.7×10^5	10^6

*The number of kernels is evaluated from the viewpoint of chip area.

$\sharp$ l is the number of iterations for convergence.

[1] S.-Y. Peng, B. A. Minch, and P. E. Hasler: *Proc. Int. Symp. Circuits Syst.*, 2008, pp. 860 - 863.
[2] K. Kang and T. Shibata: *IEEE Trans. Circuits Syst.*, Vol. 57, no. 7, pp. 1513 – 1524 (2010).

Support Vector Machine

Summary

Proposed architecture can be actually applied in SVM, with improved performances

R. Zhang and T. Shibata, SSDM, 2011.

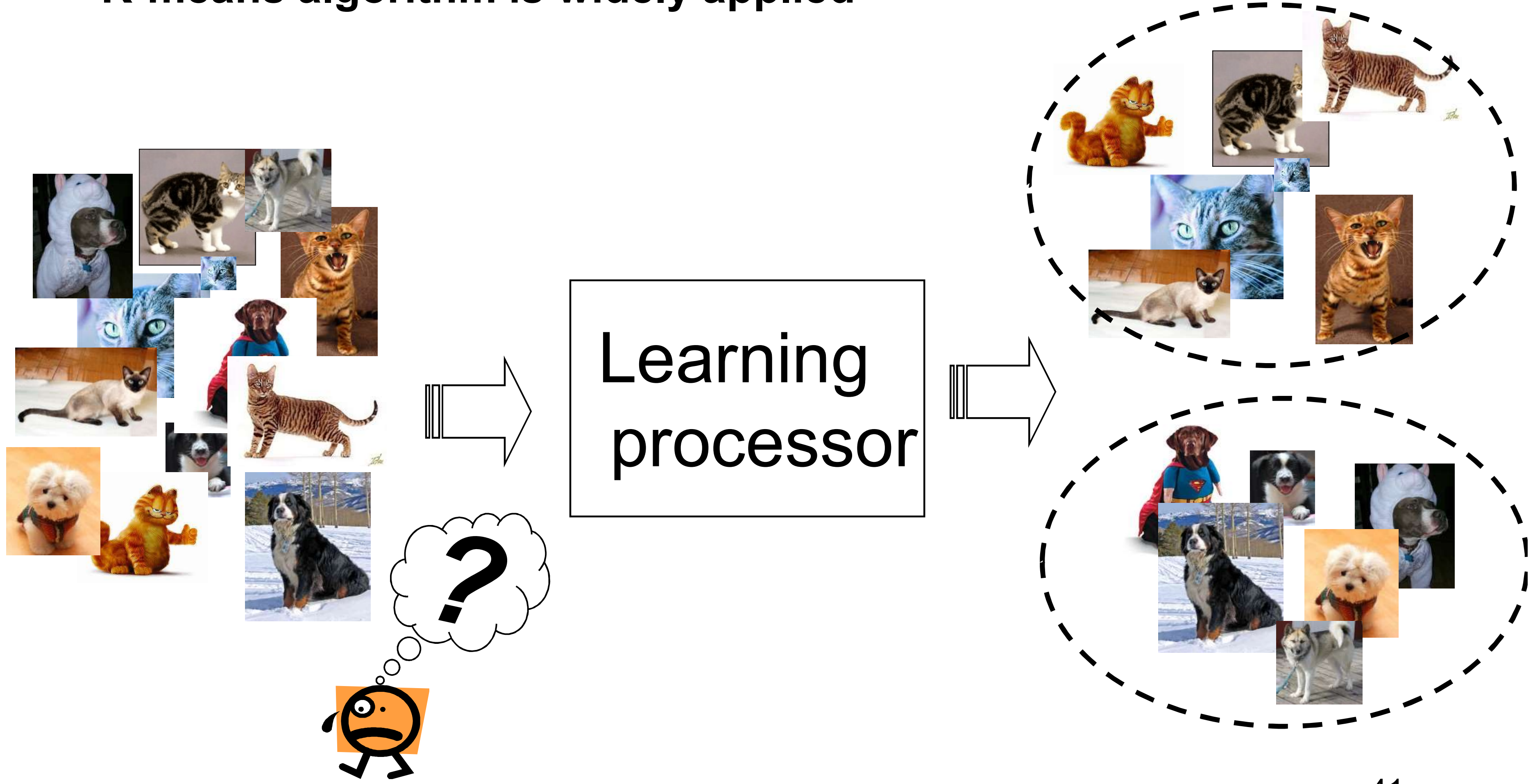
R. Zhang and T. Shibata, JJAP, 2012.

Outline

Analog implementations of them:

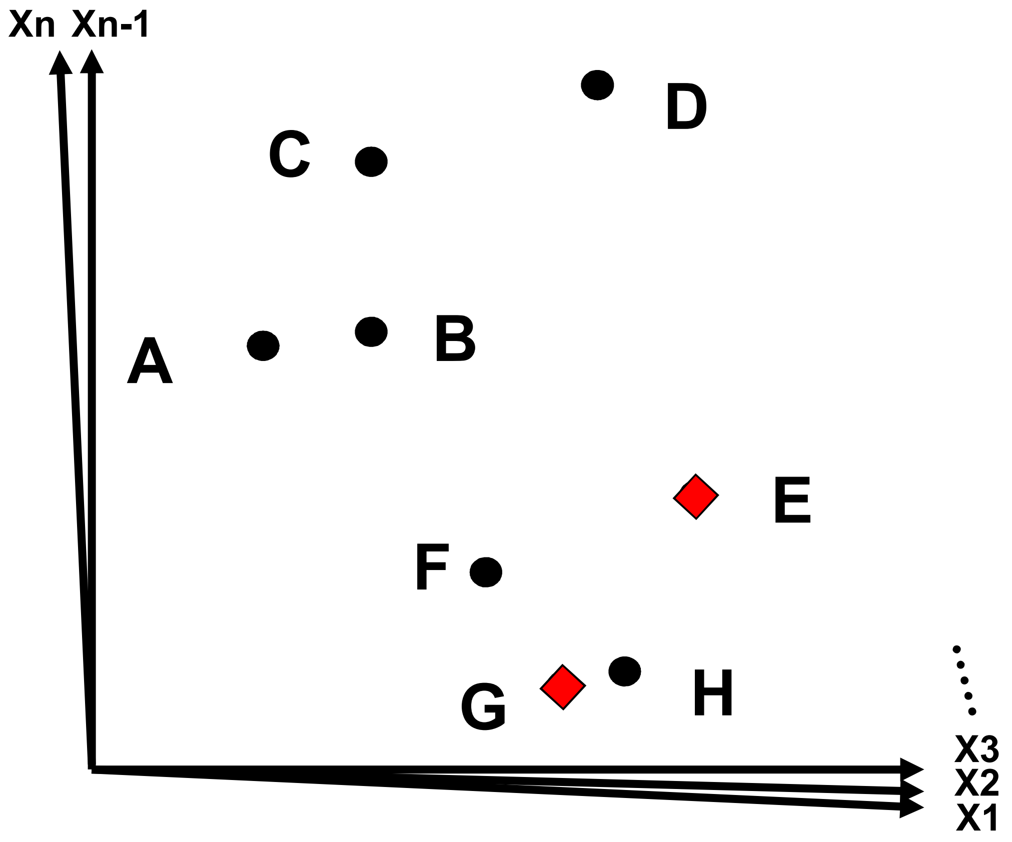
Pattern clustering

- **Patterns are clustered into categories according to the similarity**
- **K-means algorithm is widely applied**



Pattern clustering

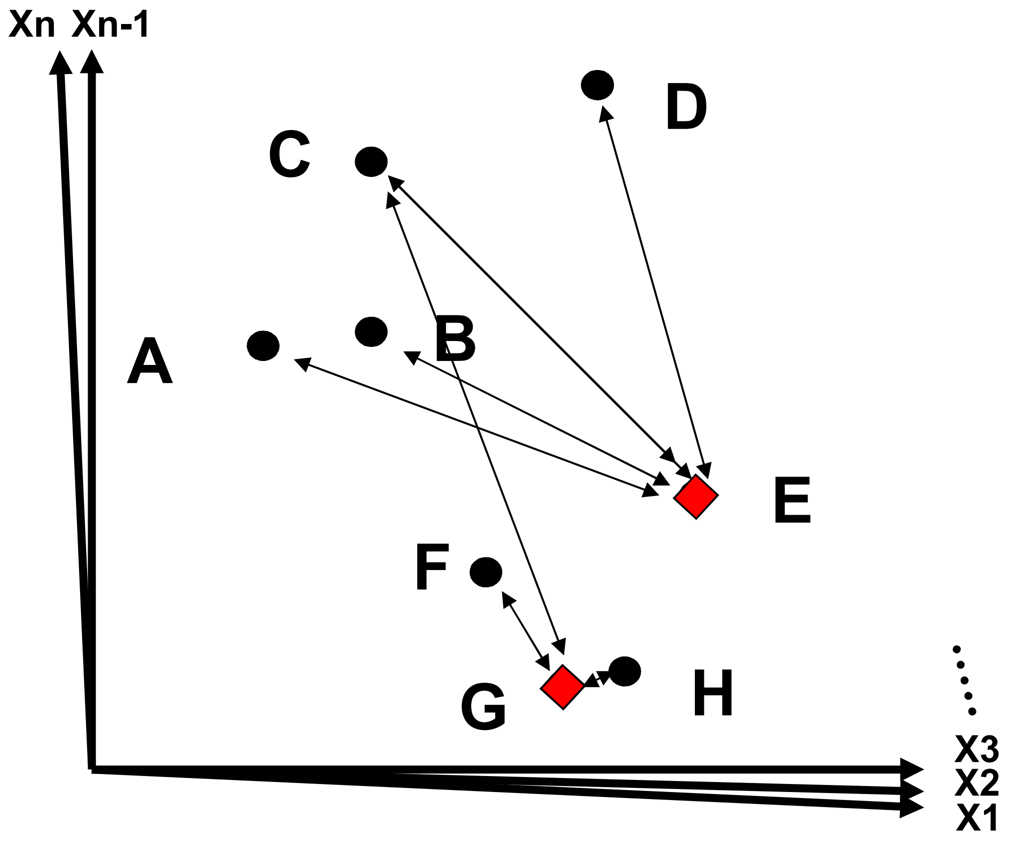
➢Original K-means



(1)Select an initial cluster center

Pattern clustering

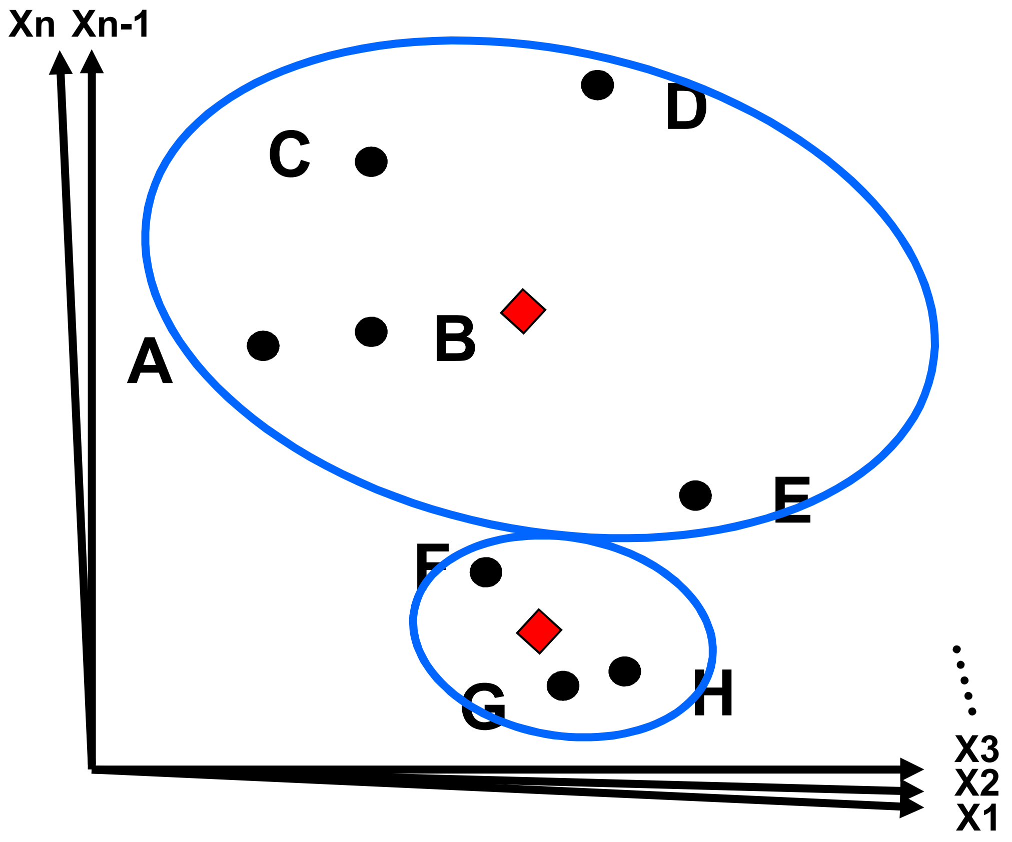
➢Original K-means



(2)Compute the distance from the cluster center

Pattern clustering

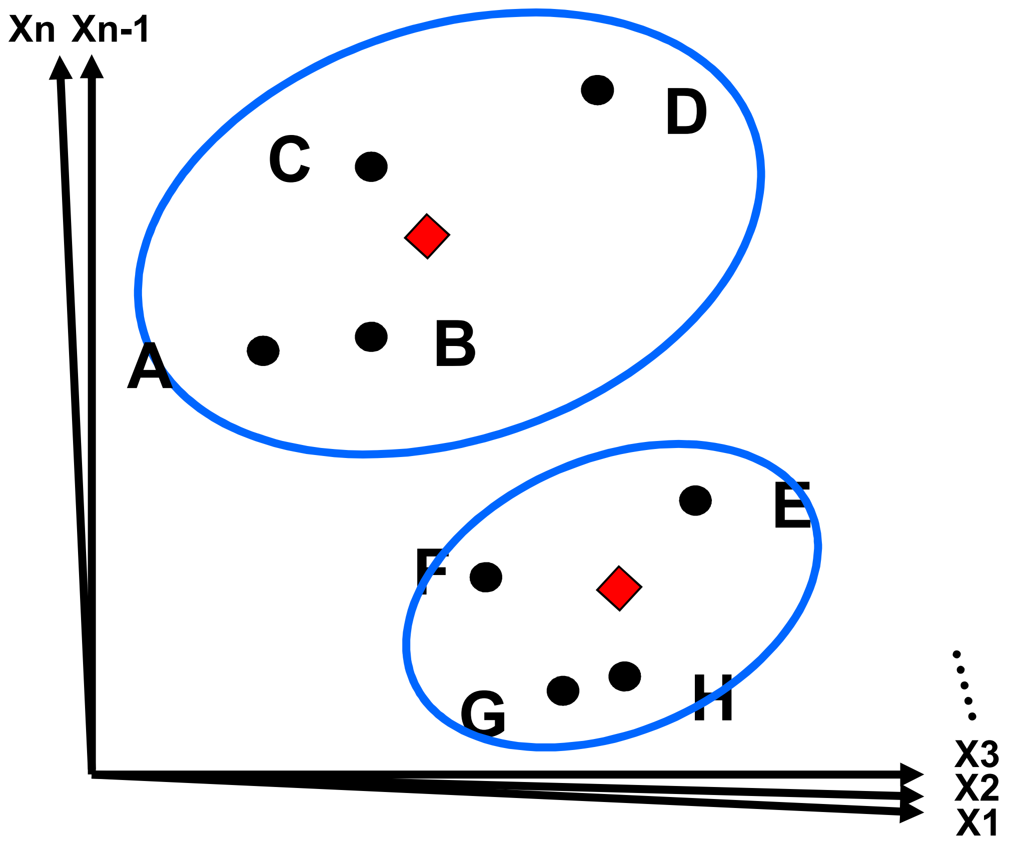
➢**Original K-means**



(3)Form cluster and update cluster center as centroid of patterns

Pattern clustering

➢Original K-means



(4) Repeat the step (2) and step(3) Until the stop condition is satisfied.

Pattern clustering

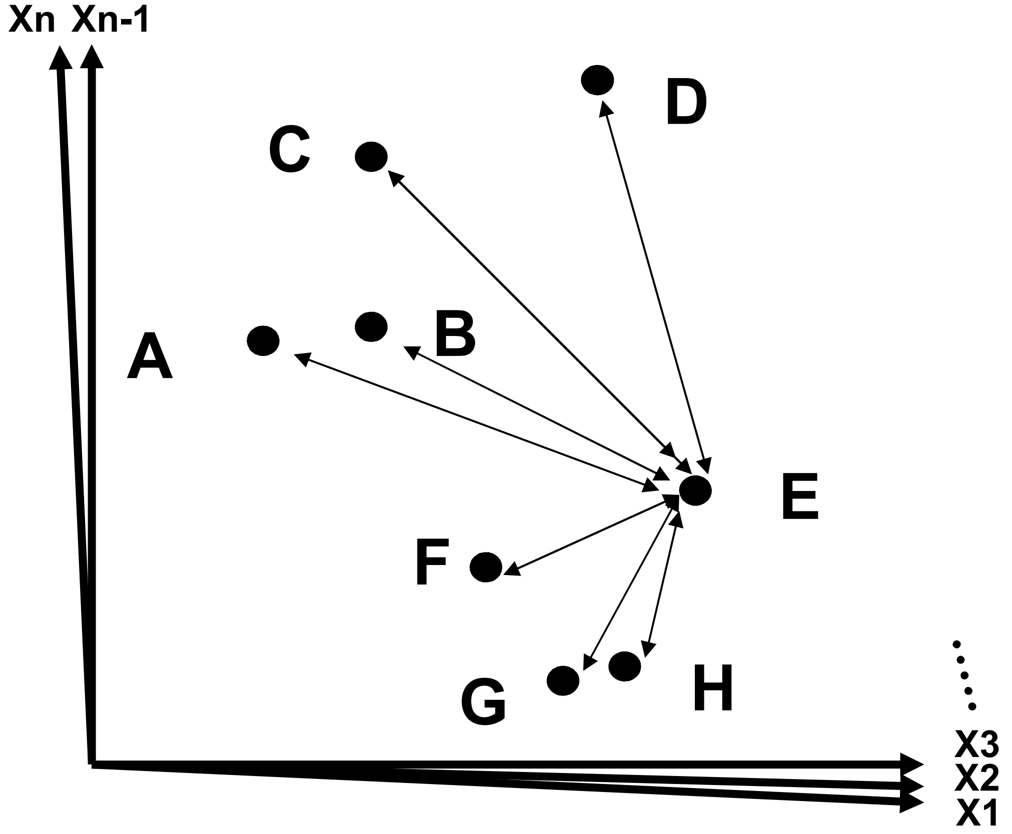
•Original K-means

**Plenty of calculations among vectors,
hardly implemented in parallel.**

K-Quasi-Centers (KQC) is proposed as the parallel-architecture-friendly version of K-means.

K-Quasi-Centers clustering

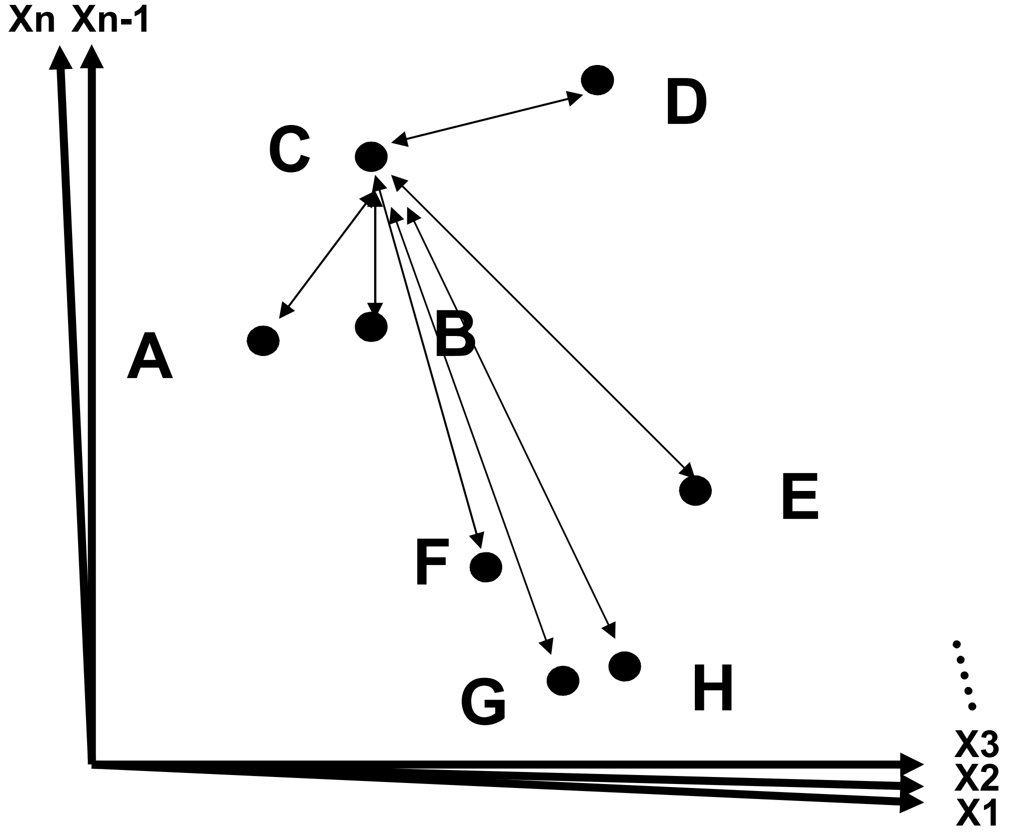
- **Our basic idea**



(1) Compute all the Euclidean distances and store them

K-Quasi-Centers clustering

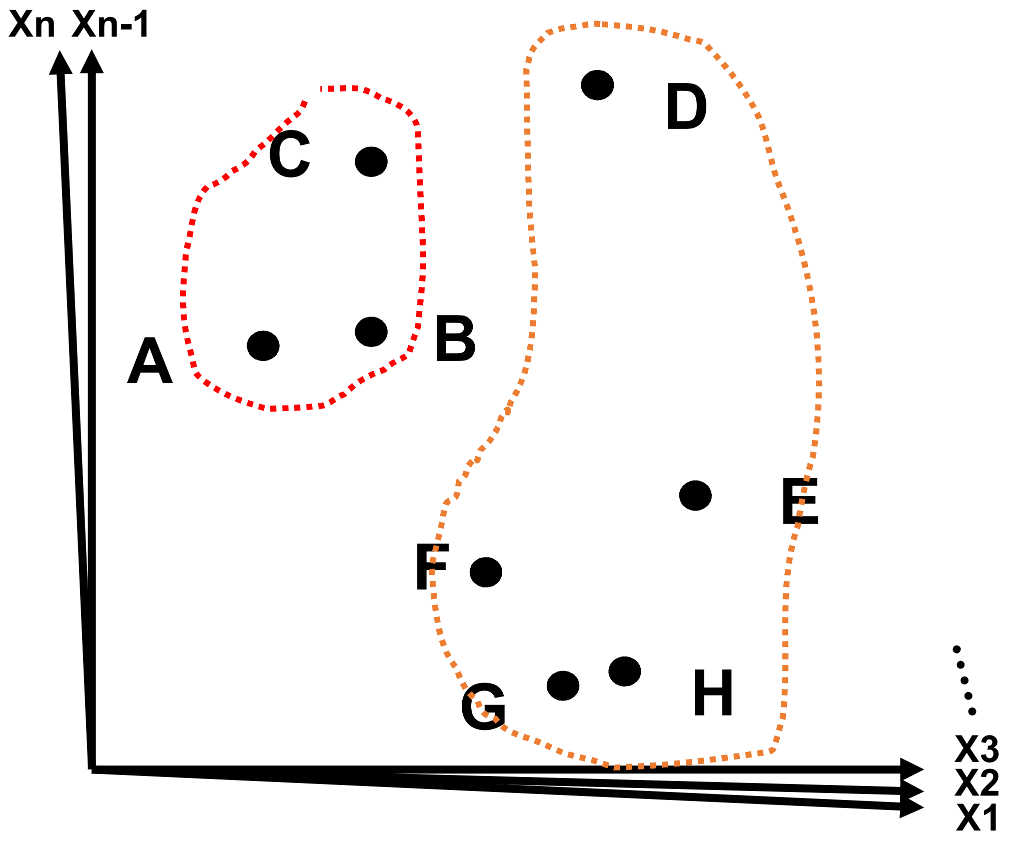
➢Our basic idea



(1) Compute all the Euclidean distances and store them

K-Quasi-Centers clustering

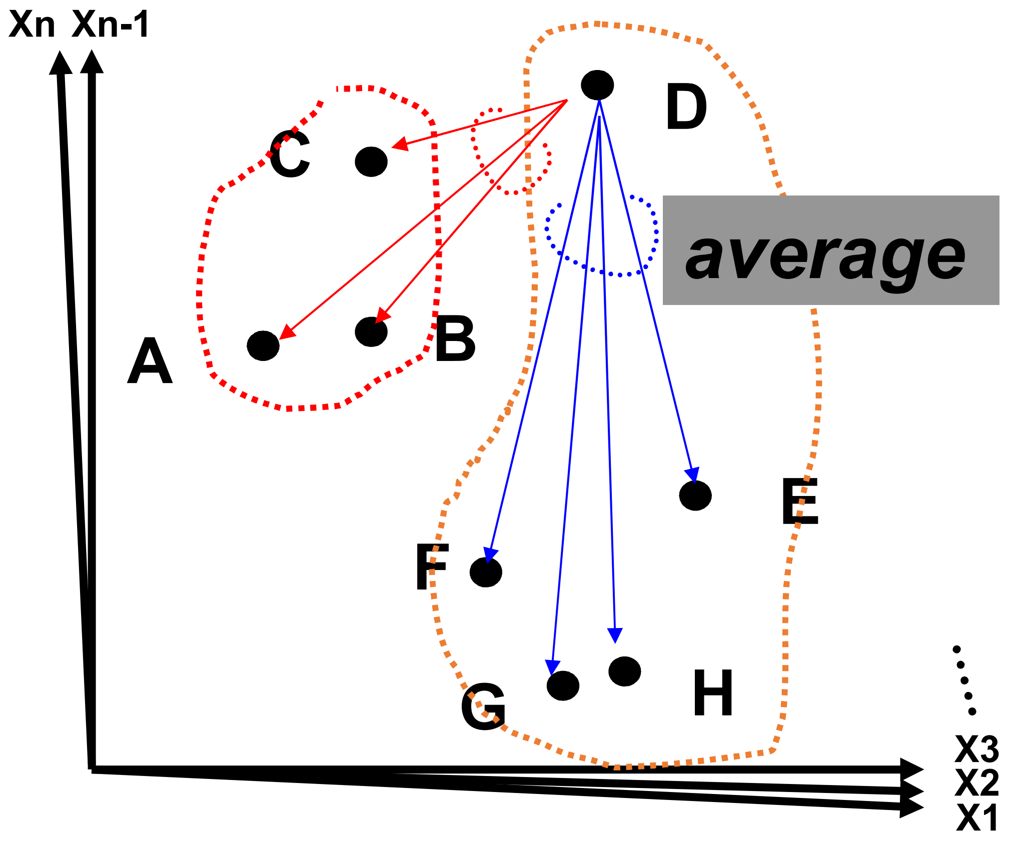
- **Our basic idea**



(2) Initialize the clustering randomly

K-Quasi-Centers clustering

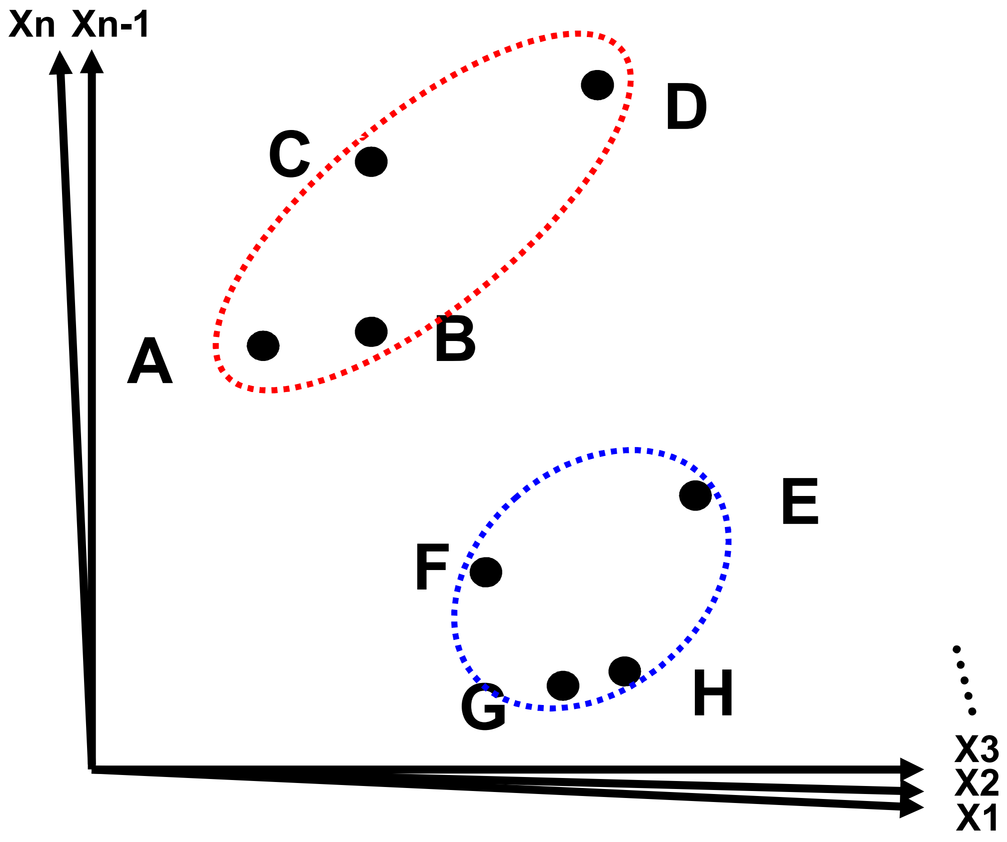
- **Our basic idea**



(3) Updating the clustering form by scalar calculation

K-Quasi-Centers clustering

- **Our basic idea**



(4) Repeating till converge

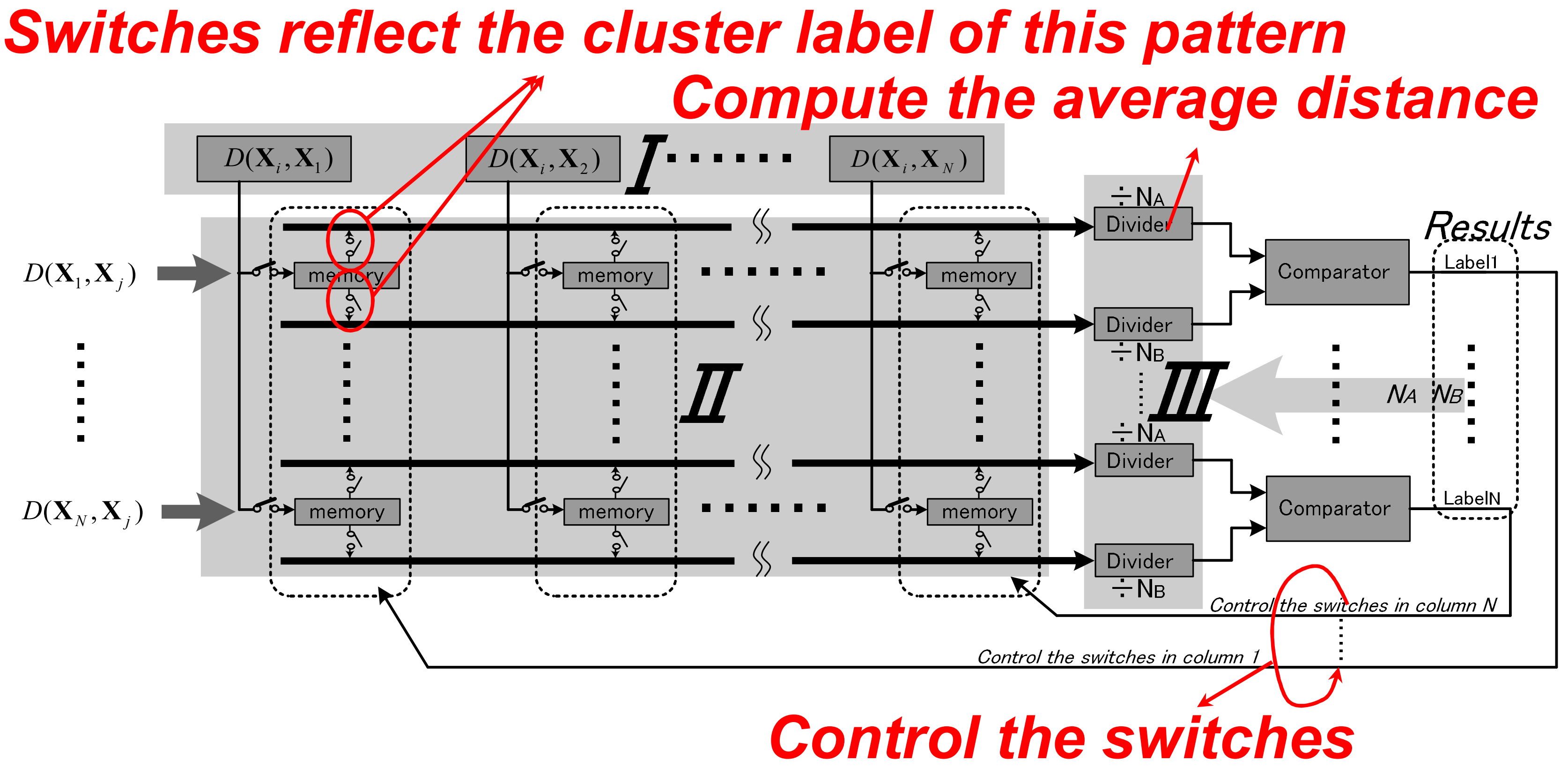
K-Quasi-Centers clustering

- **Our basic idea**

Learning based on the calculations among scalars
Possible to implement in fully-parallel

Fully parallel KQC processor

➢ **Hardware implementation (two clusters)**

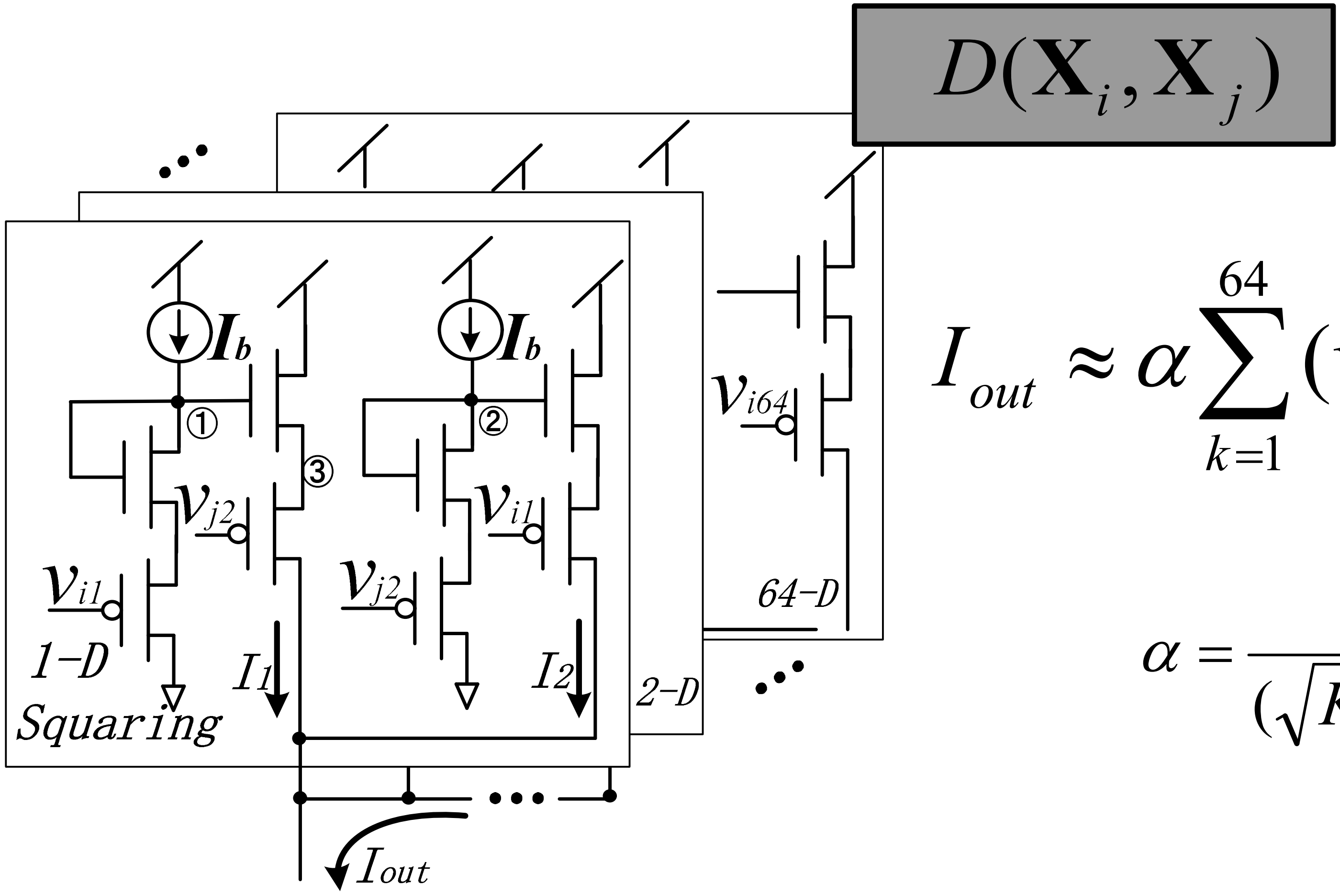


Switches reflect the cluster label of this pattern
Compute the average distance
$D(\mathbf{X}_i,\mathbf{X}_1)$
$D(\mathbf{X}_i,\mathbf{X}_2)$
$D(\mathbf{X}_i,\mathbf{X}_N)$
I
II
III
$D(\mathbf{X}_1,\mathbf{X}_j)$
$D(\mathbf{X}_N,\mathbf{X}_j)$
memory
÷NA
Divider
÷NB
Comparator
Results
Label1
LabelN
NA NB
Control the switches in column N
Control the switches in column 1
Control the switches

Fully-parallel, free and real-time feedback, self-converge

Fully parallel KQC processor

➢Euclidean distance

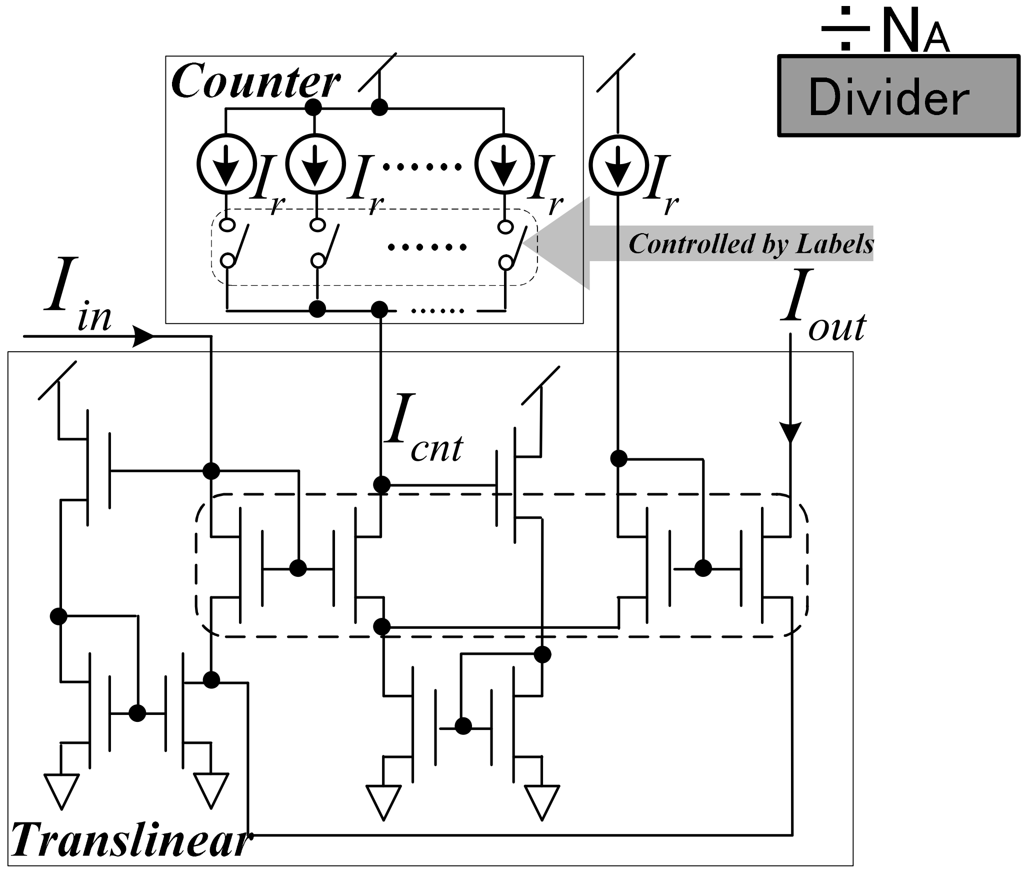


$$I_{out} \approx \alpha \sum_{k=1}^{64} (v_{ik} - v_{jk})^2$$

$$\alpha = \frac{K_n K_p}{(\sqrt{K_n} + \sqrt{K_p})^2}$$

Fully parallel KQC processor

➢ **Average**

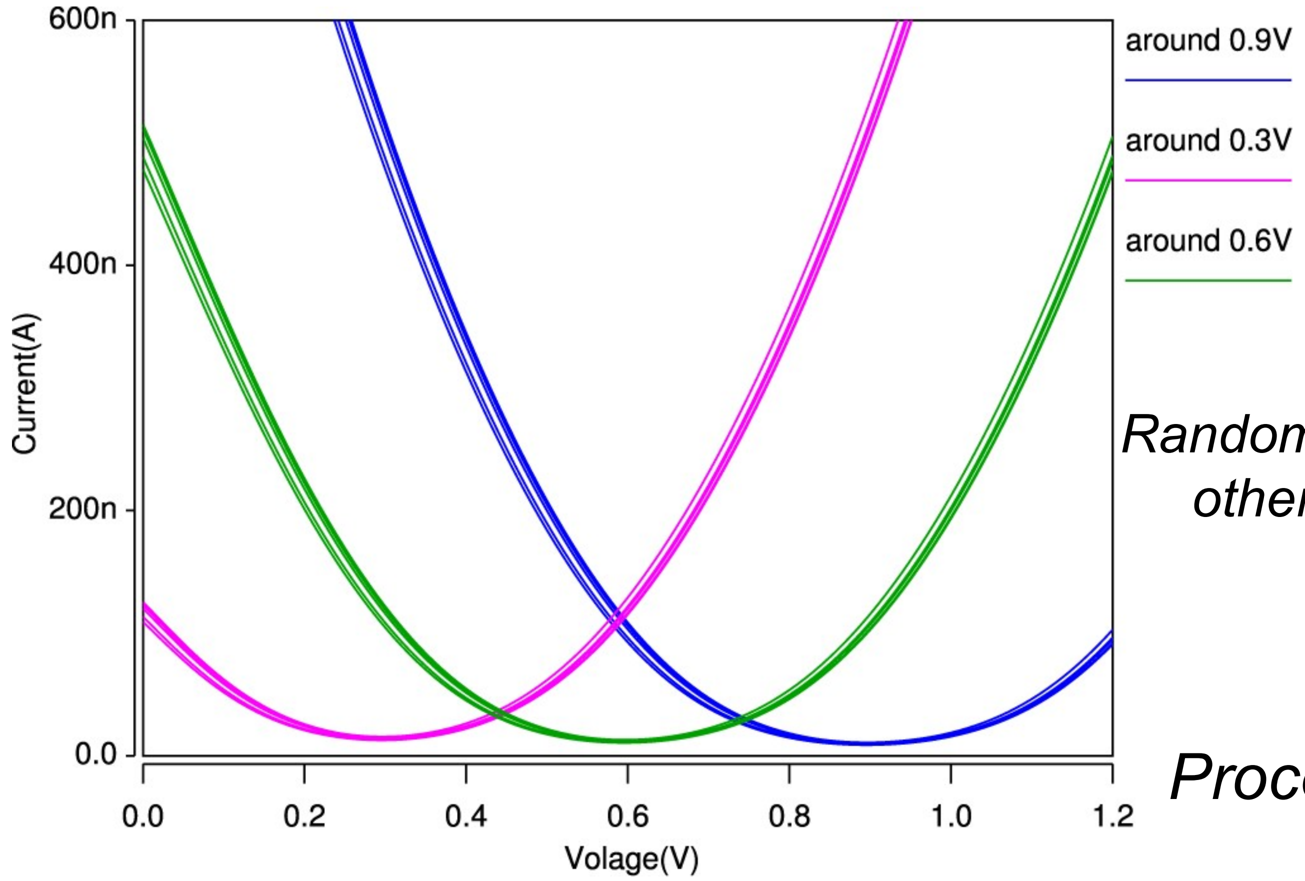


$$I_{out\_A} \approx \frac{I_{in}}{N_A}$$

or $$I_{out\_B} \approx \frac{I_{in}}{N_B}$$

Fully parallel KQC processor

➢Euclidean distance



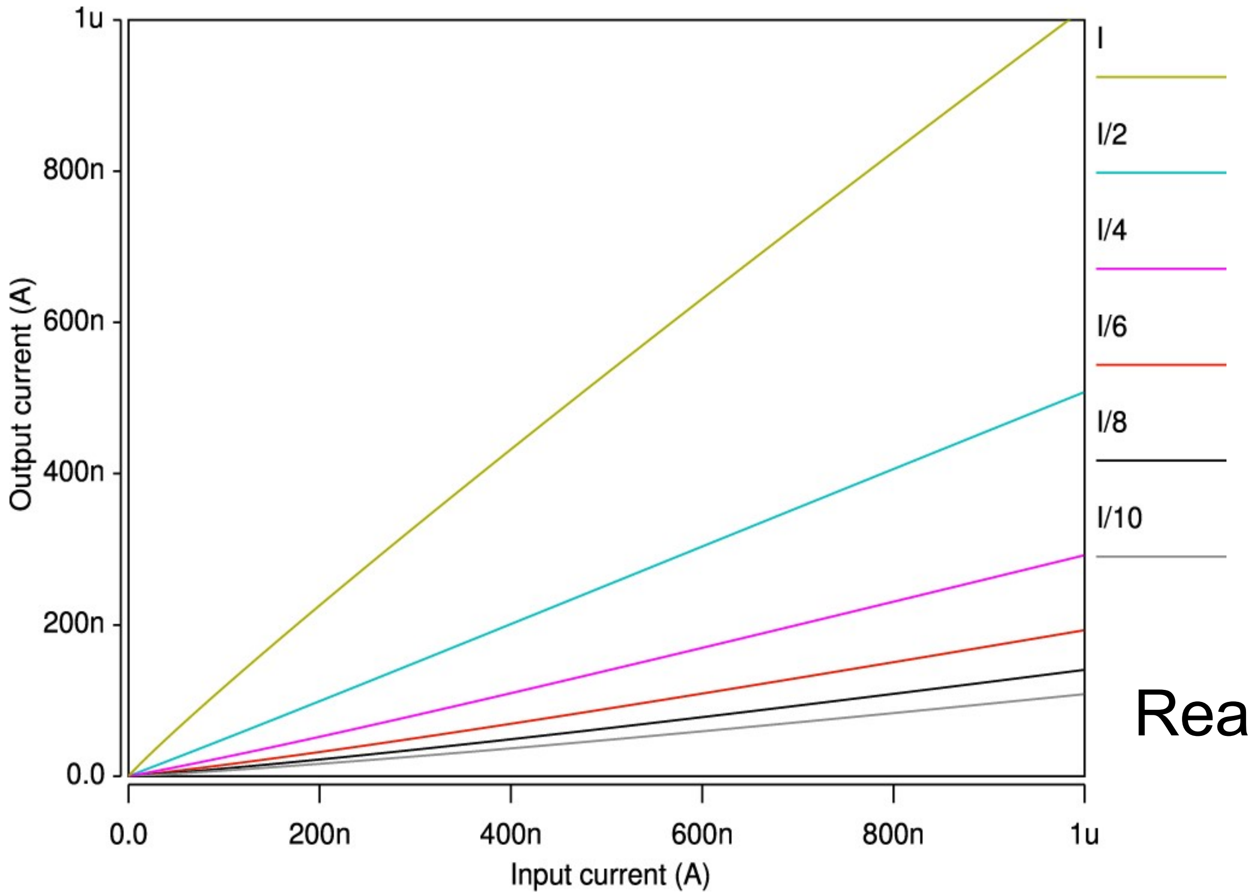
Simulation

Randomly select a dimension, others are constant

Process variation: 5%

Fully parallel KQC processor

➢Translinear (divider)

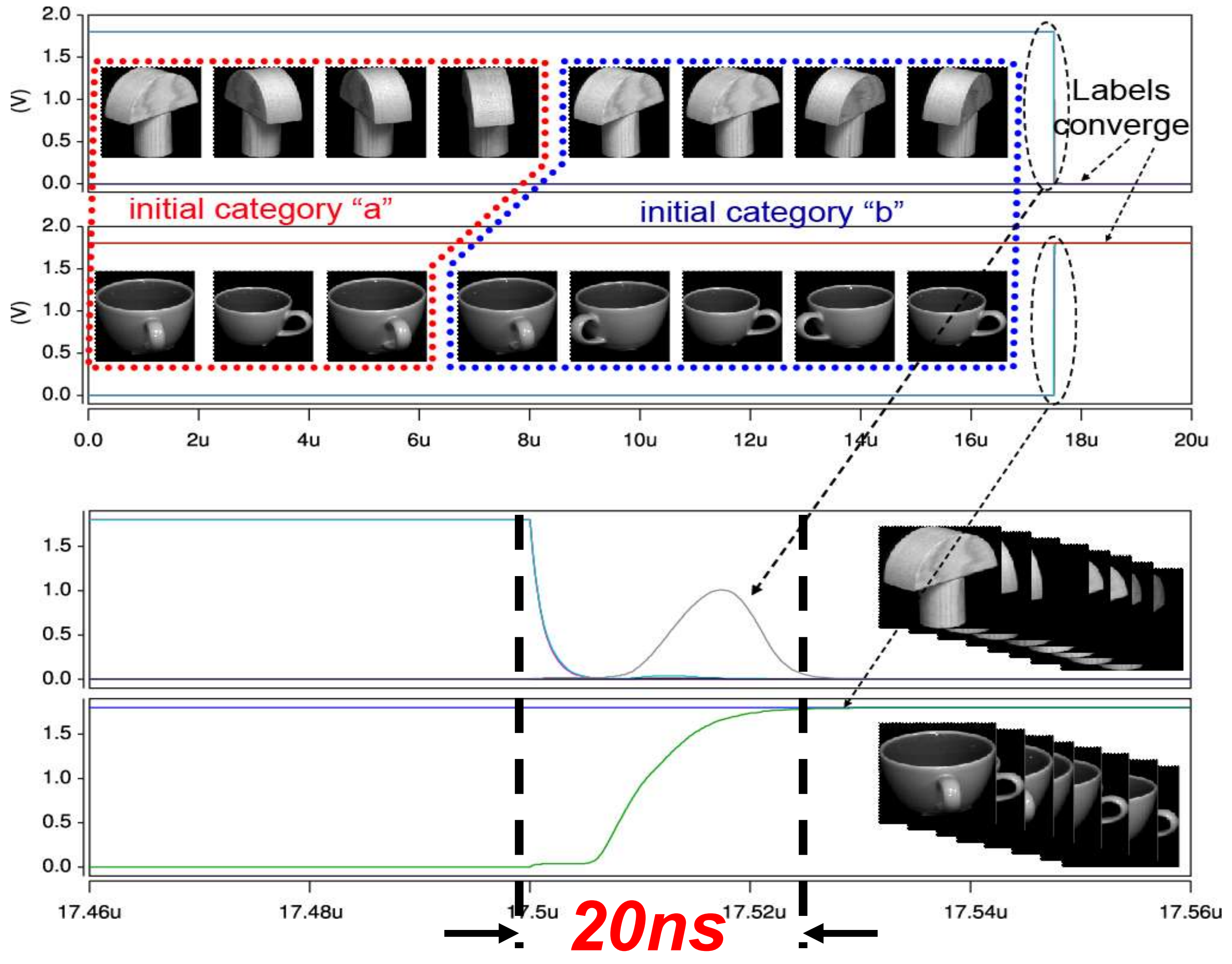


Simulation

Real-time computation

Fully parallel KQC processor

➢Image clustering

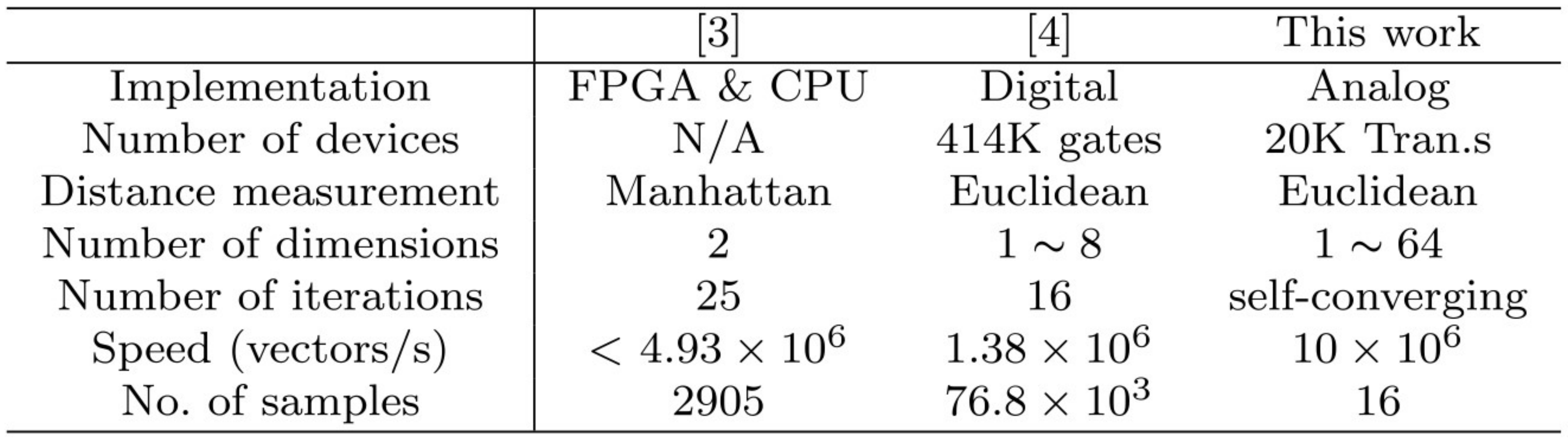


Simulation

Fully parallel KQC processor

➢Performance

Comparisons



	[3]	[4]	This work
Implementation	FPGA & CPU	Digital	Analog
Number of devices	N/A	414K gates	20K Tran.s
Distance measurement	Manhattan	Euclidean	Euclidean
Number of dimensions	2	$1 \sim 8$	$1 \sim 64$
Number of iterations	25	16	self-converging
Speed (vectors/s)	$< 4.93 \times 10^6$	1.38×10^6	10×10^6
No. of samples	2905	76.8×10^3	16

[3] H. M. Hussain et al, *NASA/ESA Conf. Adaptive Hardware and Systems*, 2011, pp. 246-255.
[4] T.-W. Chen and S.-Y. Chien, *IEEE Trans. VLSI Syst.*, vol. 11, no. 8, pp. 1336-1345 (2011).

K-Quasi-Centers clustering

Summary

Proposed architecture can be applied in pattern clustering problems, with improved performances

R. Zhang and T. Shibata, “An Analog K-means Learning Processor Employing Fully-Parallel Self-Converging Circuitry” , *Int. Analog VLSI Workshop*, 2011, pp. 91-96.

Problems

Limited hardware resource

V.S.

Huge database

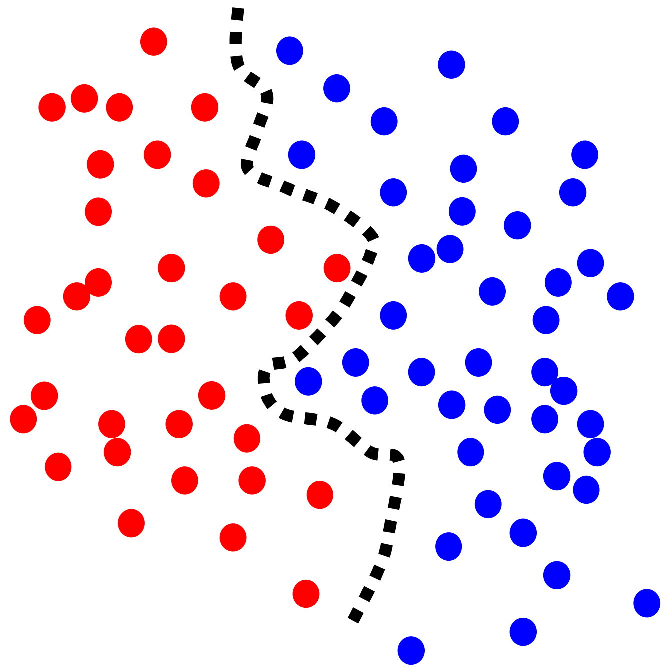
Unpredictable database

Outline

Analog implementations of them:

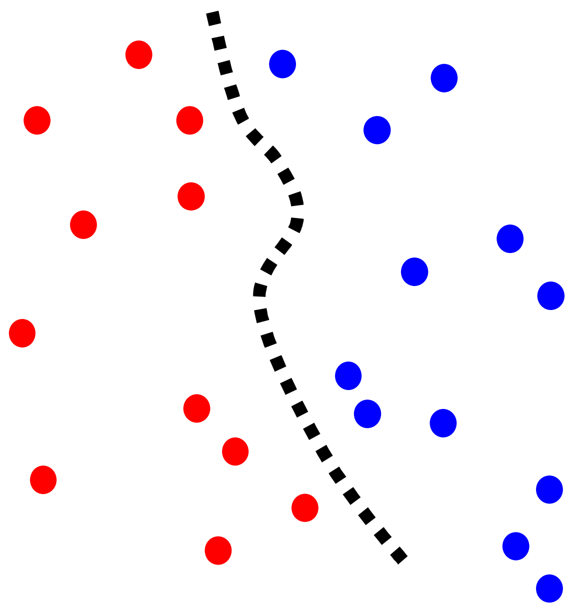
- Support Vector Machine
- K-Quasi-Centers clustering
- **On-line learning strategies**
- Data domain description
- Intel Project

On-line learning



Great performance

But difficult to train



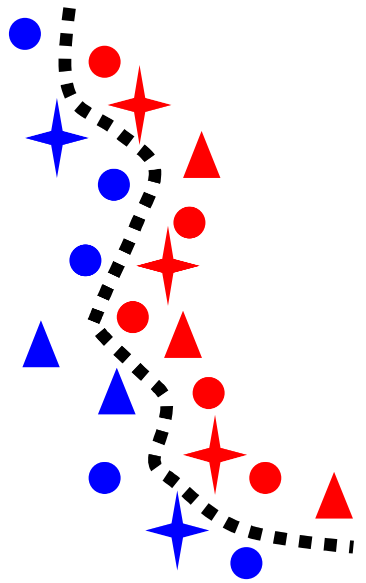
Easy to train

But performance is poor

Hardware resource is limited

On-line learning

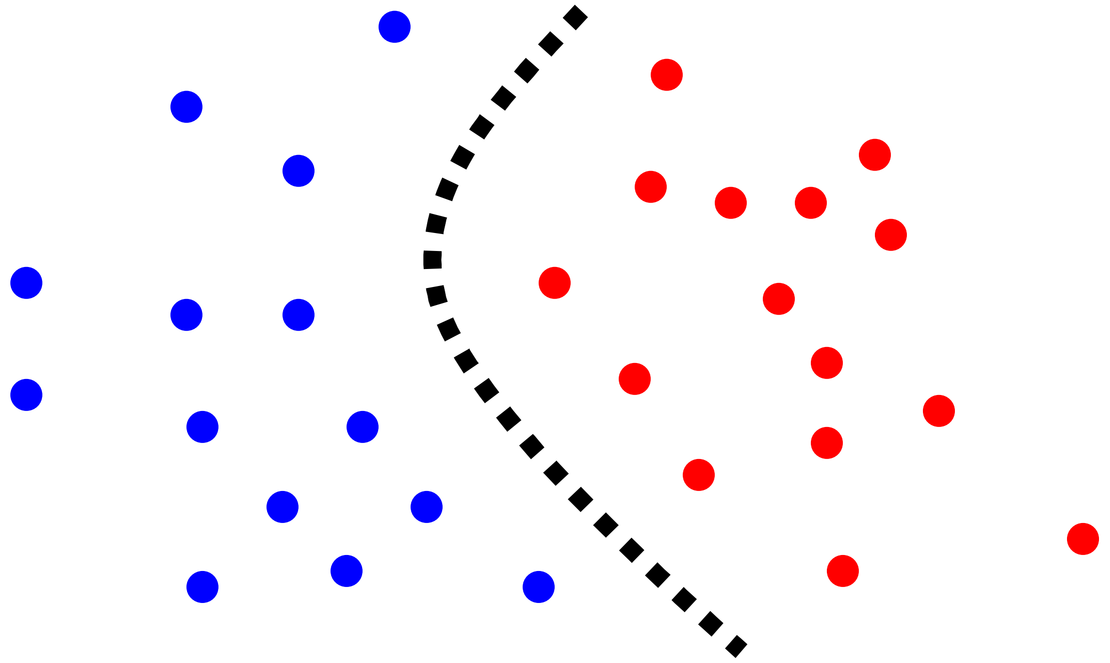
We wish:



On-line learning strategy is developed for this purpose

On-line learning

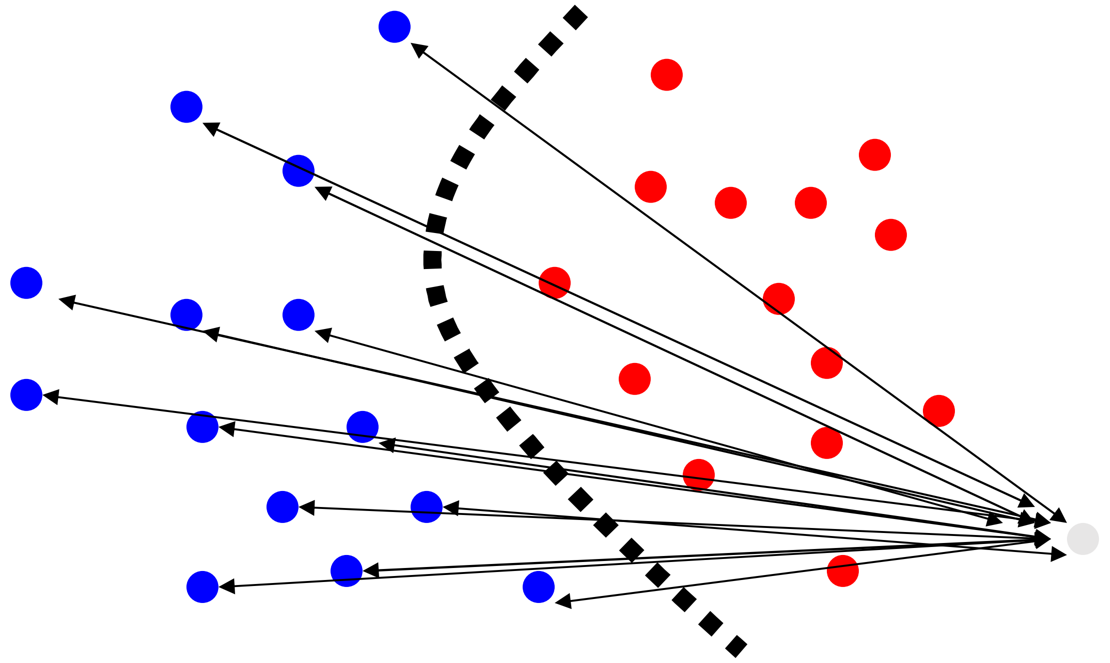
➢**Our proposal**



Half-unsupervised

On-line learning

➢**Our proposal**

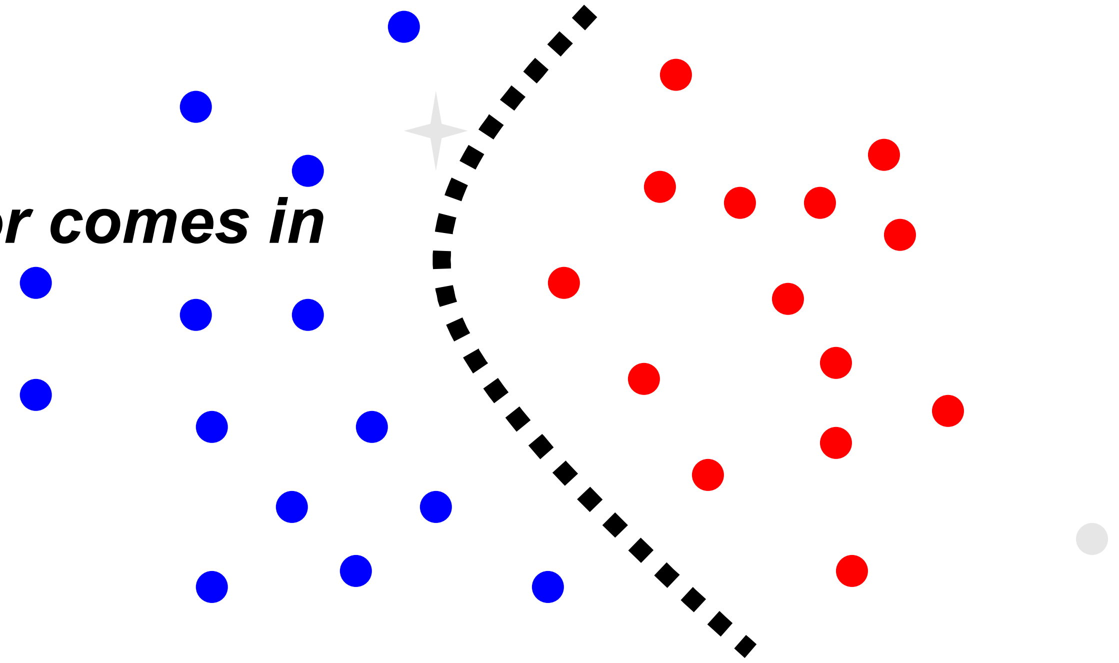


Fortunately, it is possible only in fully-parallel system

On-line learning

- **Our proposal**

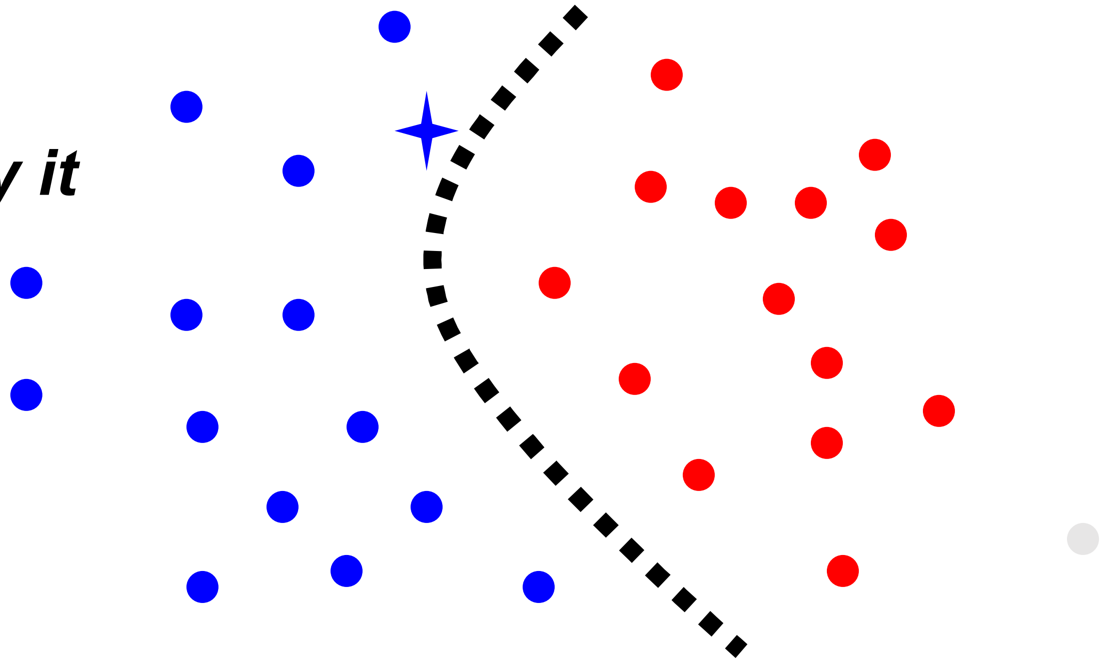
New vector comes in



On-line learning

➢**Our proposal**

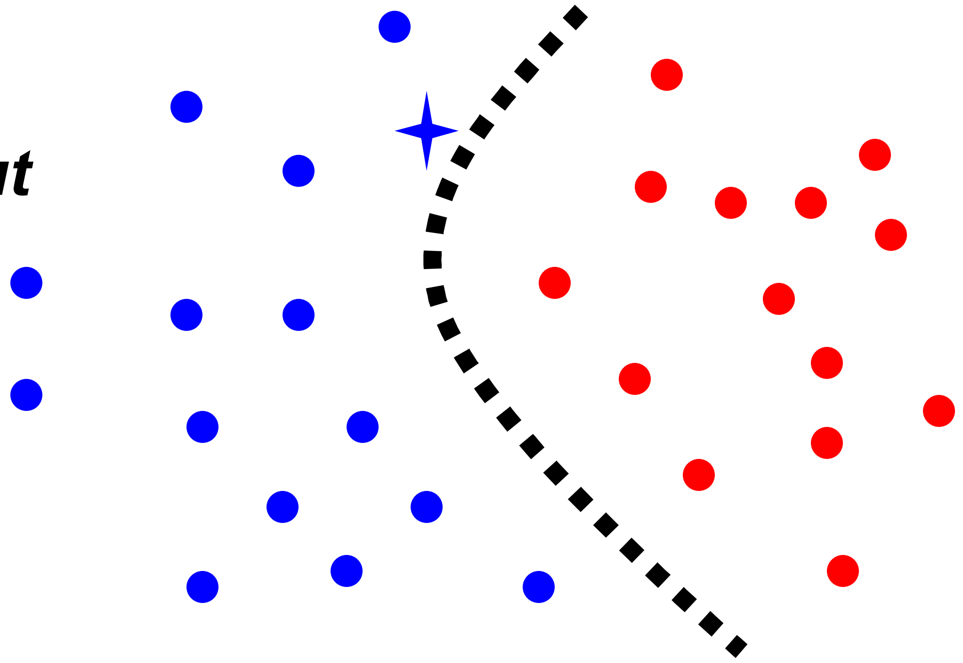
Classify it



On-line learning

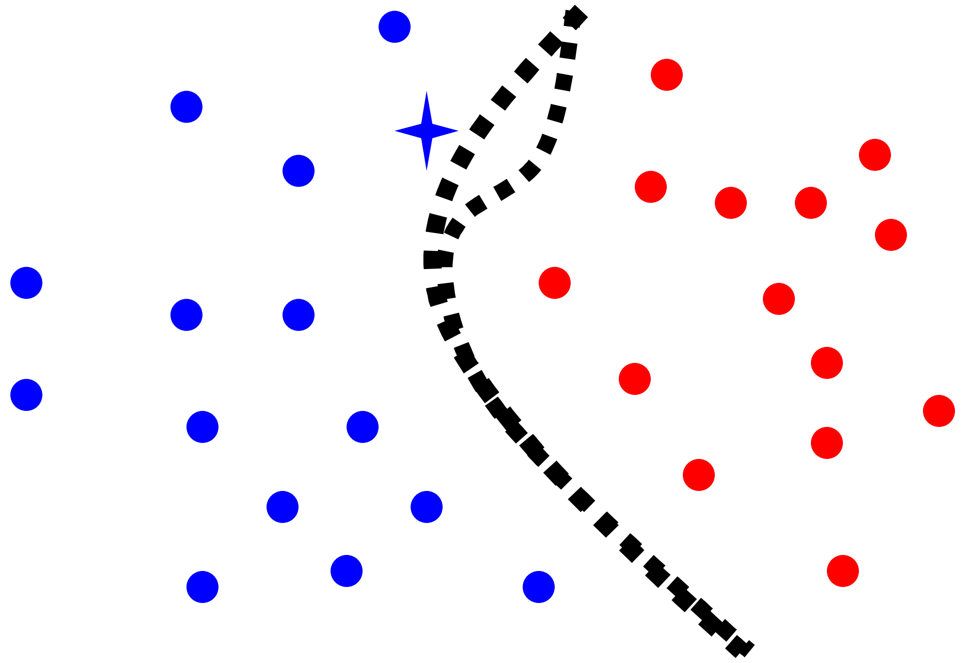
➢**Our proposal**

One-Out
One-In



On-line learning

- **Our proposal**

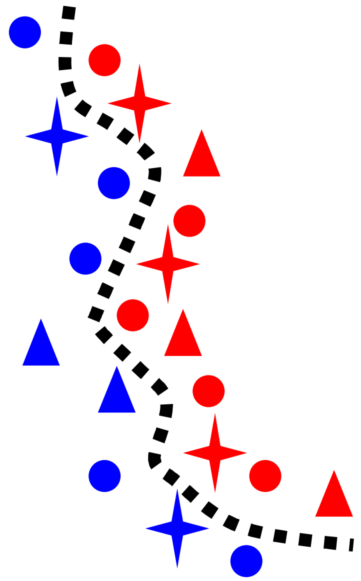


Learning again

On-line learning

➢**Our proposal**

Finally

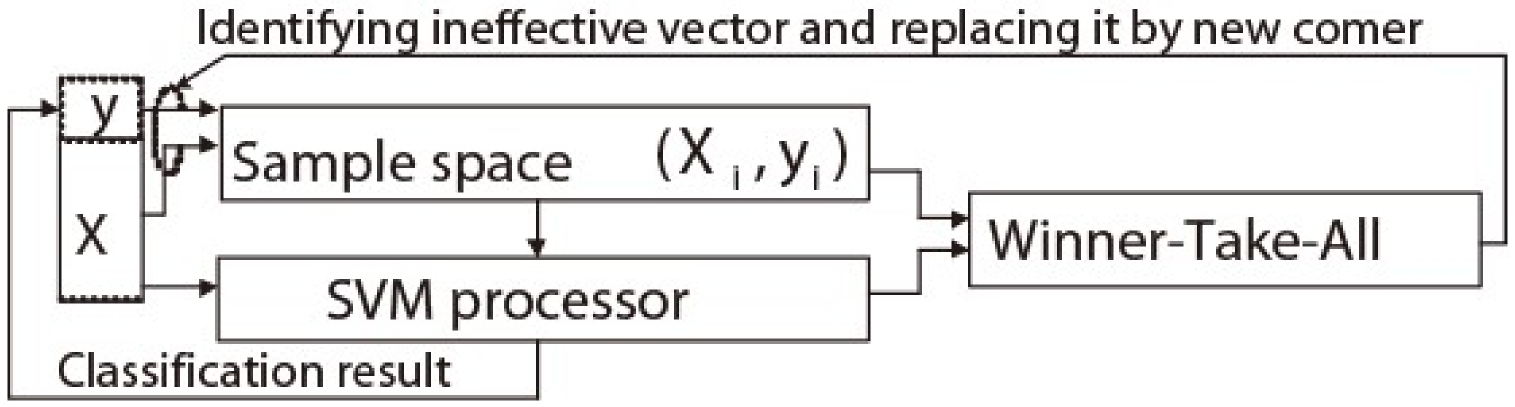


This strategy is suitable for fully parallel hardware implementation

On-line learning

Hardware implementations

On-line SVM


Identifying ineffective vector and replacing it by new comer
y
X
Sample space (X_i, y_i)
SVM processor
Winner-Take-All
Classification result

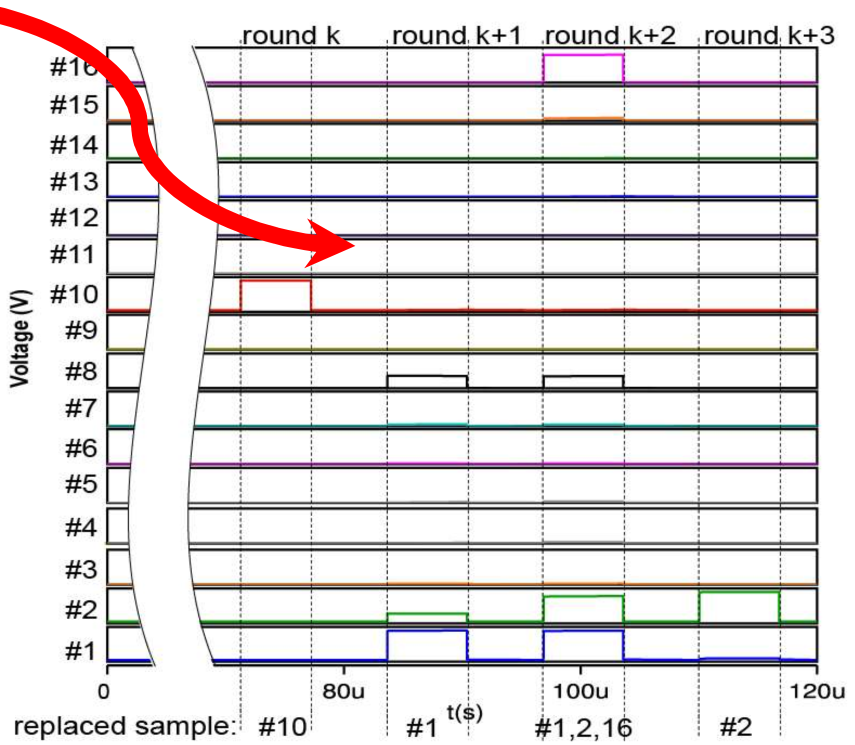
On-line learning

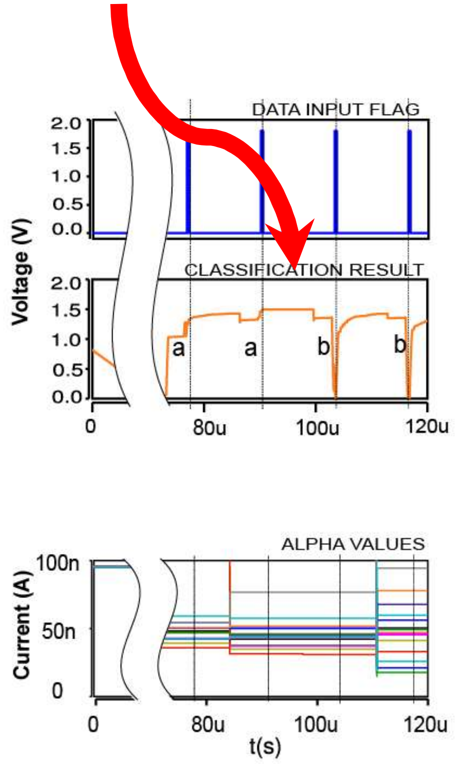
Hardware implementations

On-line SVM

Which one is useless?

New comer is …?



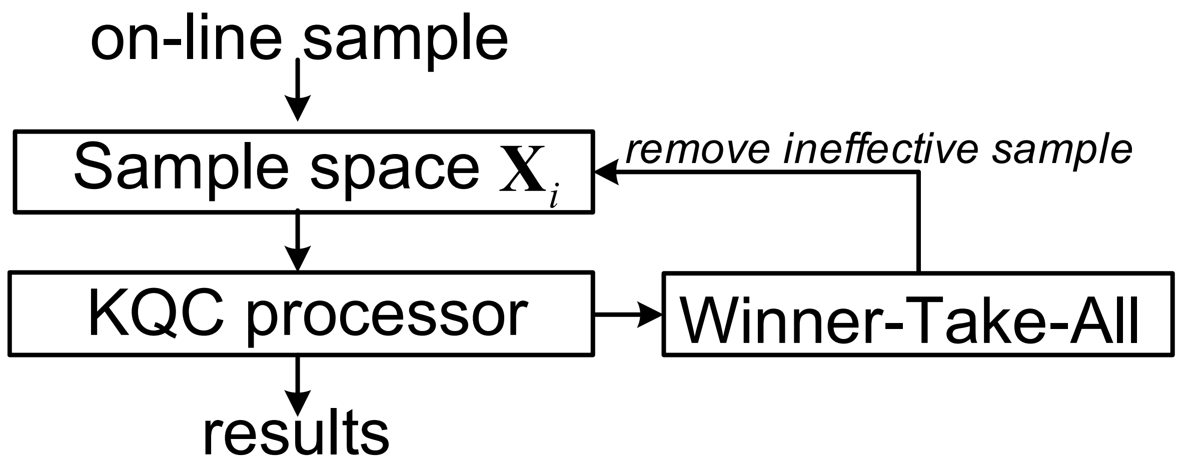


Simulation

On-line learning

Hardware implementations

On-line KQC learning

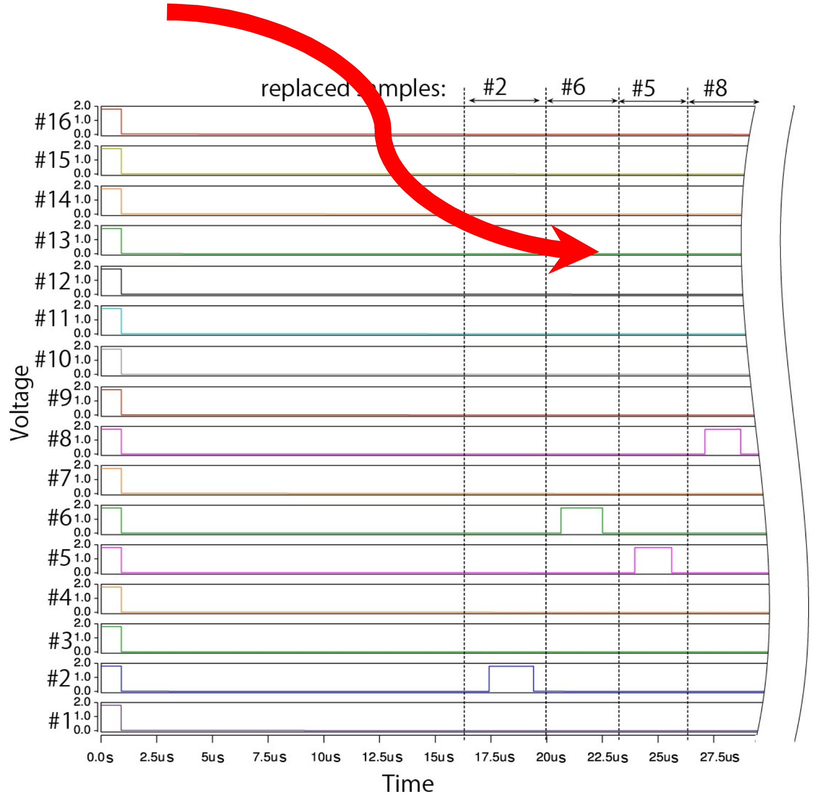


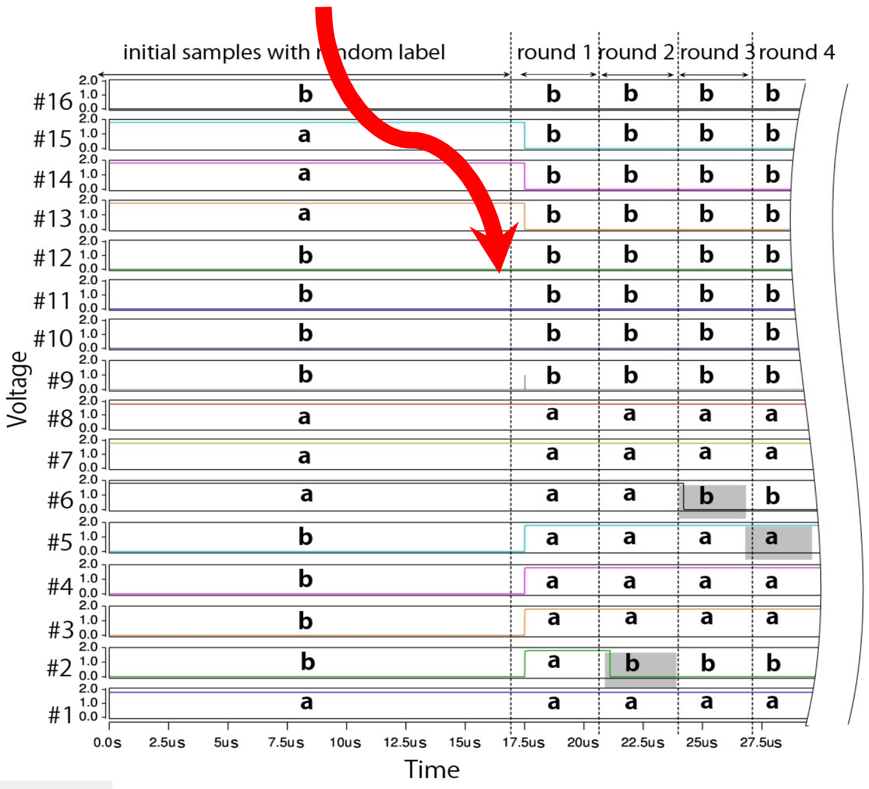
On-line learning

On-line K-means

Which one is useless?

How to cluster them?





	initial	round 1	round 2	round 3	round 4
#16	b	b	b	b	b
#15	a	b	b	b	b
#14	a	b	b	b	b
#13	a	b	b	b	b
#12	b	b	b	b	b
#11	b	b	b	b	b
#10	b	b	b	b	b
#9	b	b	b	b	b
#8	a	a	a	a	a
#7	a	a	a	a	a
#6	a	a	a	b	b
#5	b	a	a	a	a
#4	b	a	a	a	a
#3	b	a	a	a	a
#2	b	a	b	b	b
#1	a	a	a	a	a

Simulation

Application example

Is this kind of hardware feasible to be used in the real-world applications?

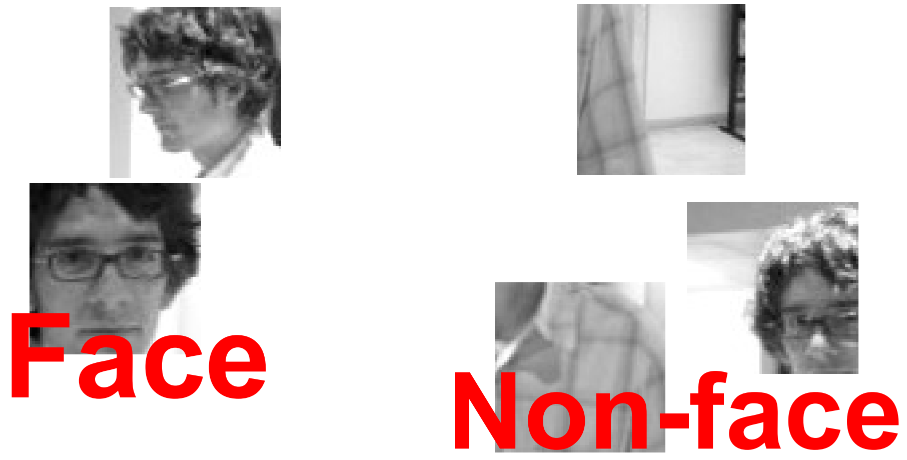
An object tracking system was built by our group-member Mr. Zhao, employing an SVM chip fabricated in this work.

Application example

SVM in tracking

Target:
find face in video

SVM:
distinguish face and background

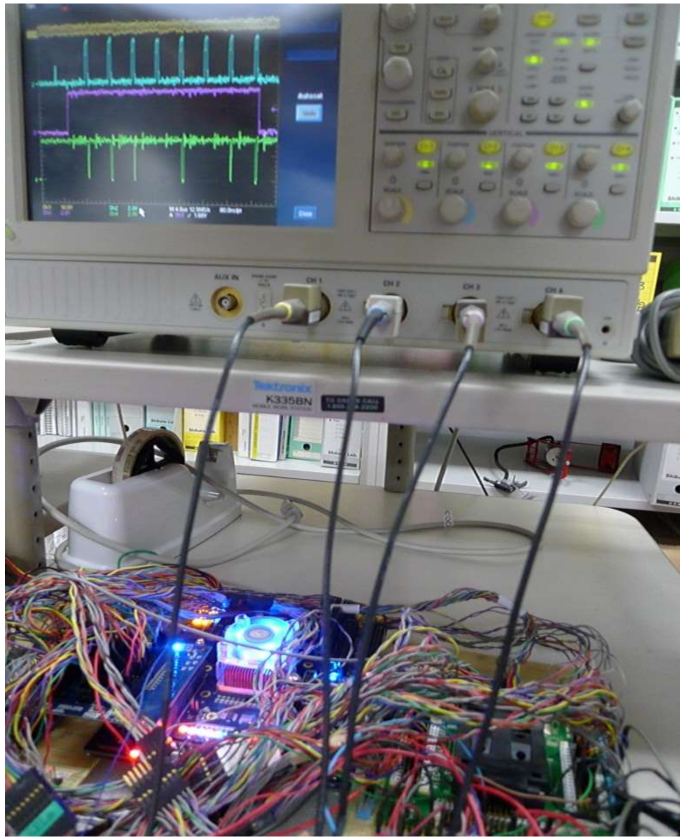


Framework of system was built by Mr. Zhao

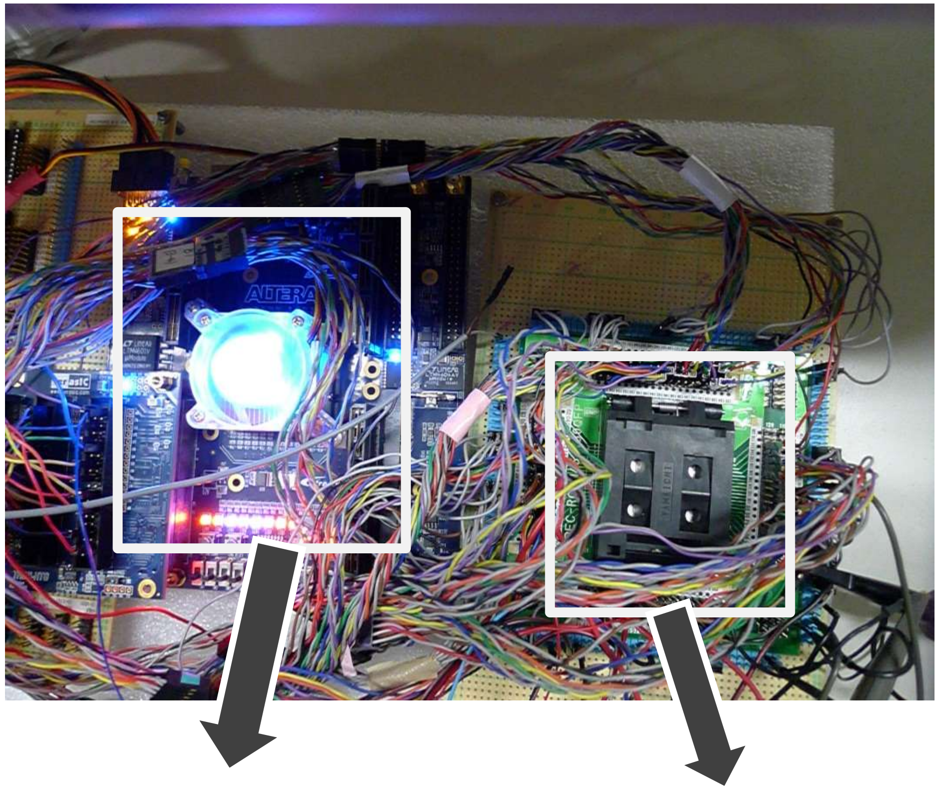
Analog SVM chip was employed

Application example

SVM in tracking



Framework of system was built by Mr. Zhao



FPGA: Tracking Framework

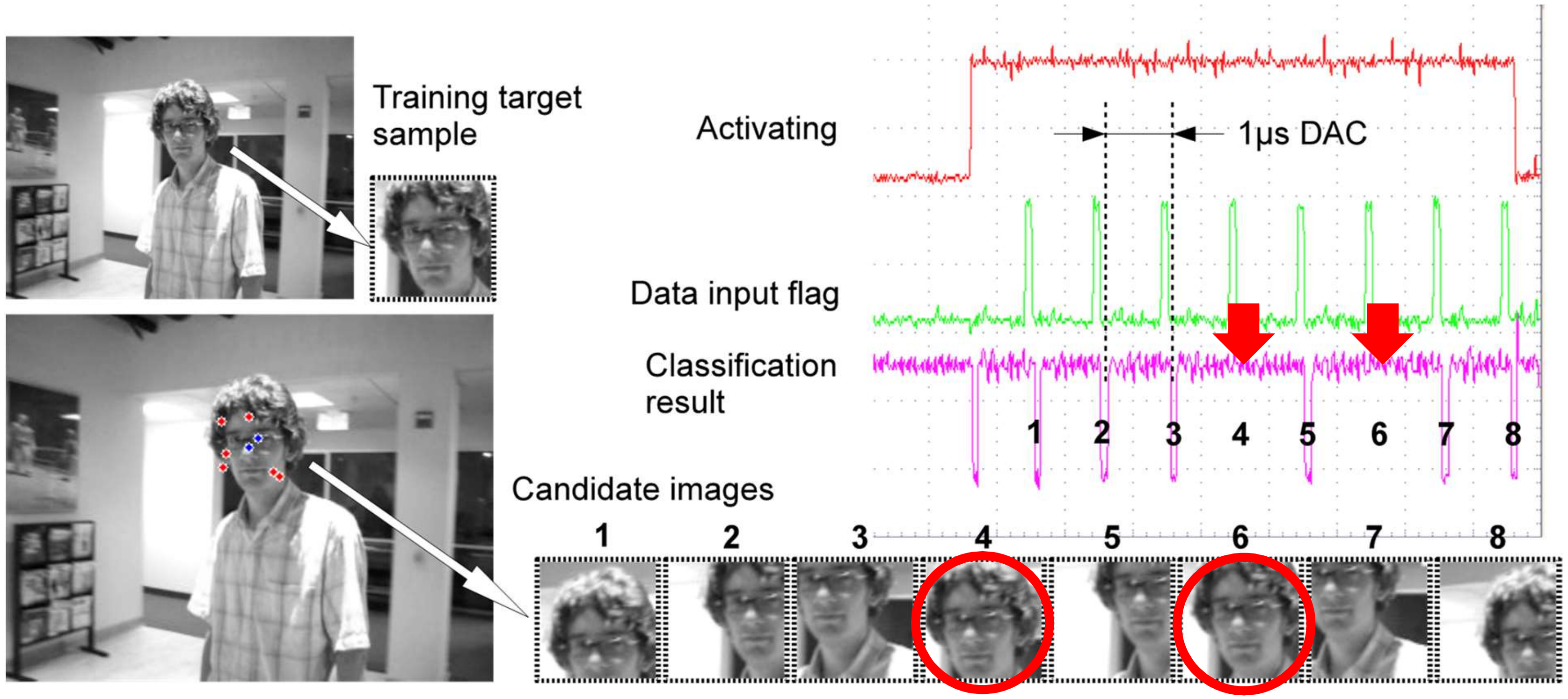
Analog SVM chip

Analog SVM chip was employed

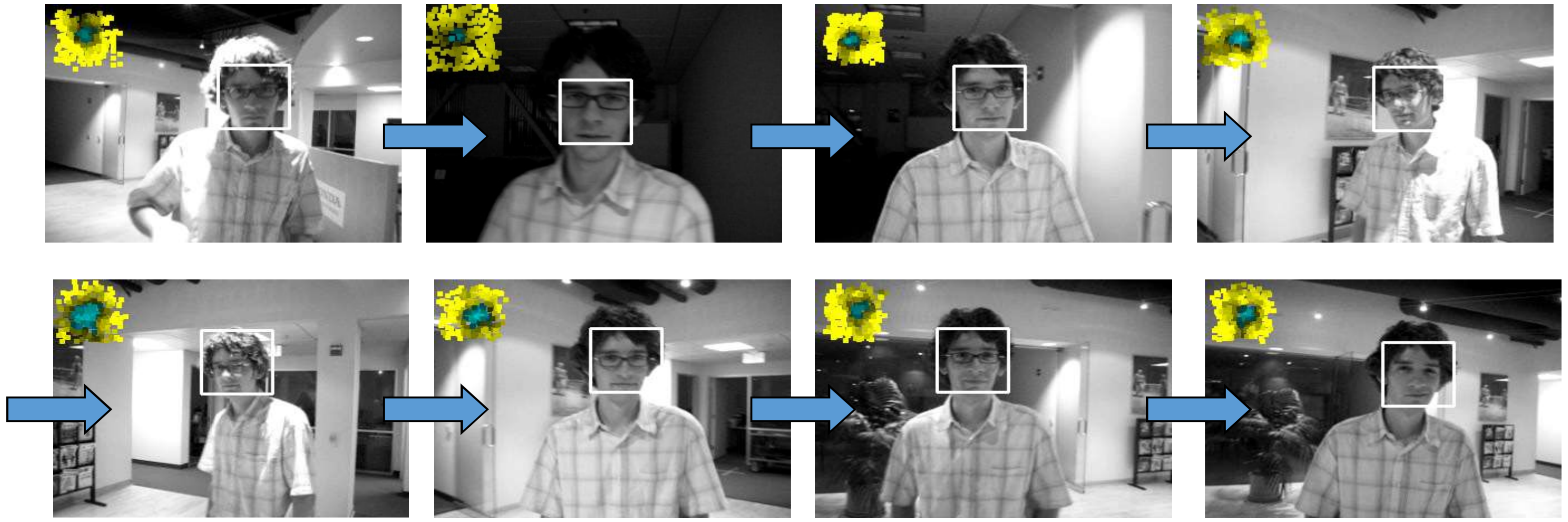
Application example

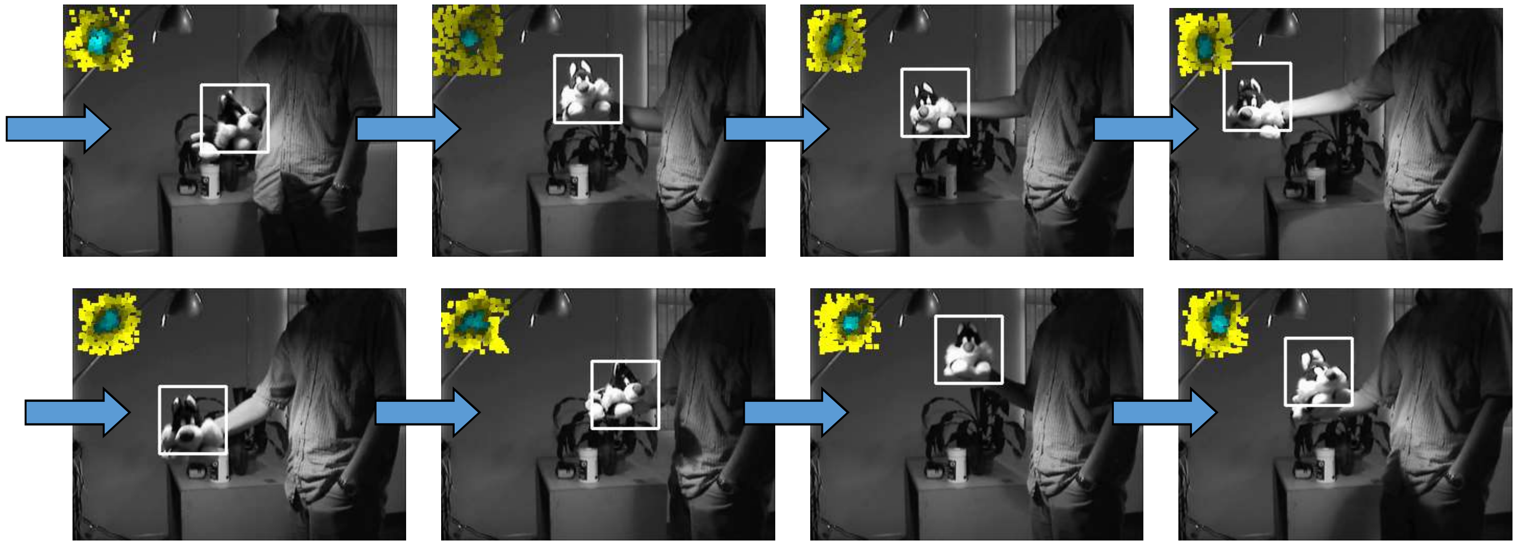
SVM in tracking

Measurement



Object Tracking With On-Line Shape Learning





P. Zhao, R. Zhang, and T. Shibata, "Real-time Object Tracking Algorithm Employing On-Line Support Vector Machine and Multiple Candidate Regeneration," ICAISC, 2012, Poland.

Summary

- ✓ *Proposed on-line learning strategy, which is on the basis of fully parallel architecture, helps to improve hardware flexibility and efficiency.*
- ✓ *Proposed hardware is feasible to use in real-world applications.*

R. Zhang and T. Shibata, LNCS, 2012.
R. Zhang and T. Shibata, *J. Analog Integrated Circuits and Signal Processing* (Springer), 2012.
P. Zhao, R. Zhang and T. Shibata, LNCS, 2012.

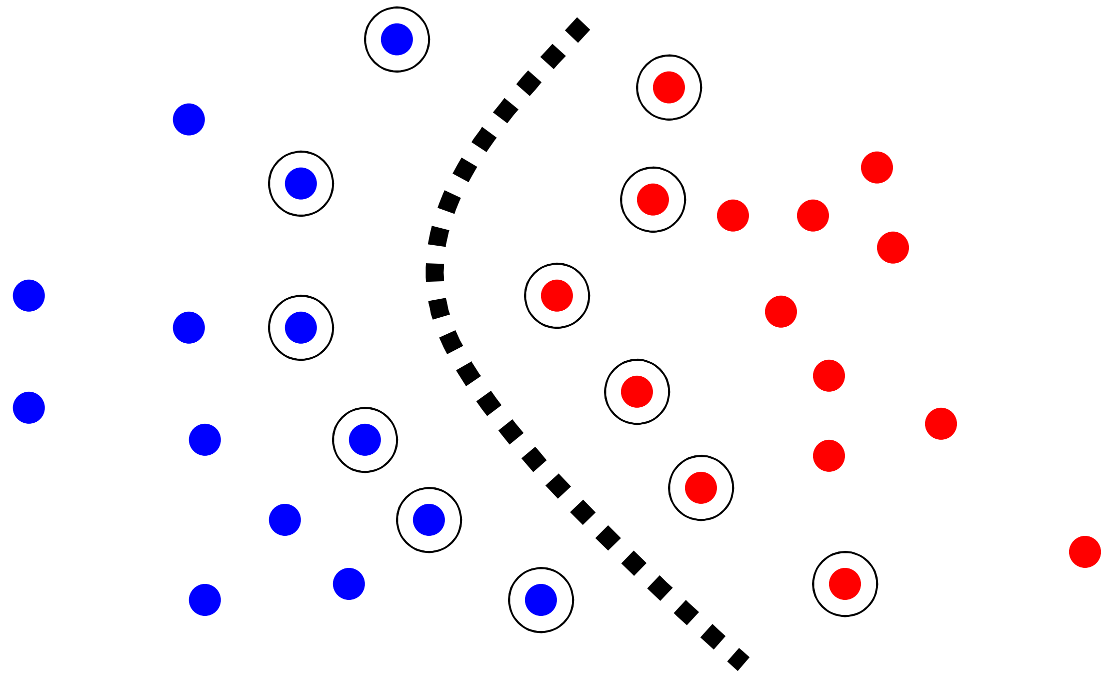
Outline

Analog implementations of them:

- Support Vector Machine
- K-Quasi-Centers clustering
- On-line learning strategies
- **Data domain description**
- Discussion on scalability

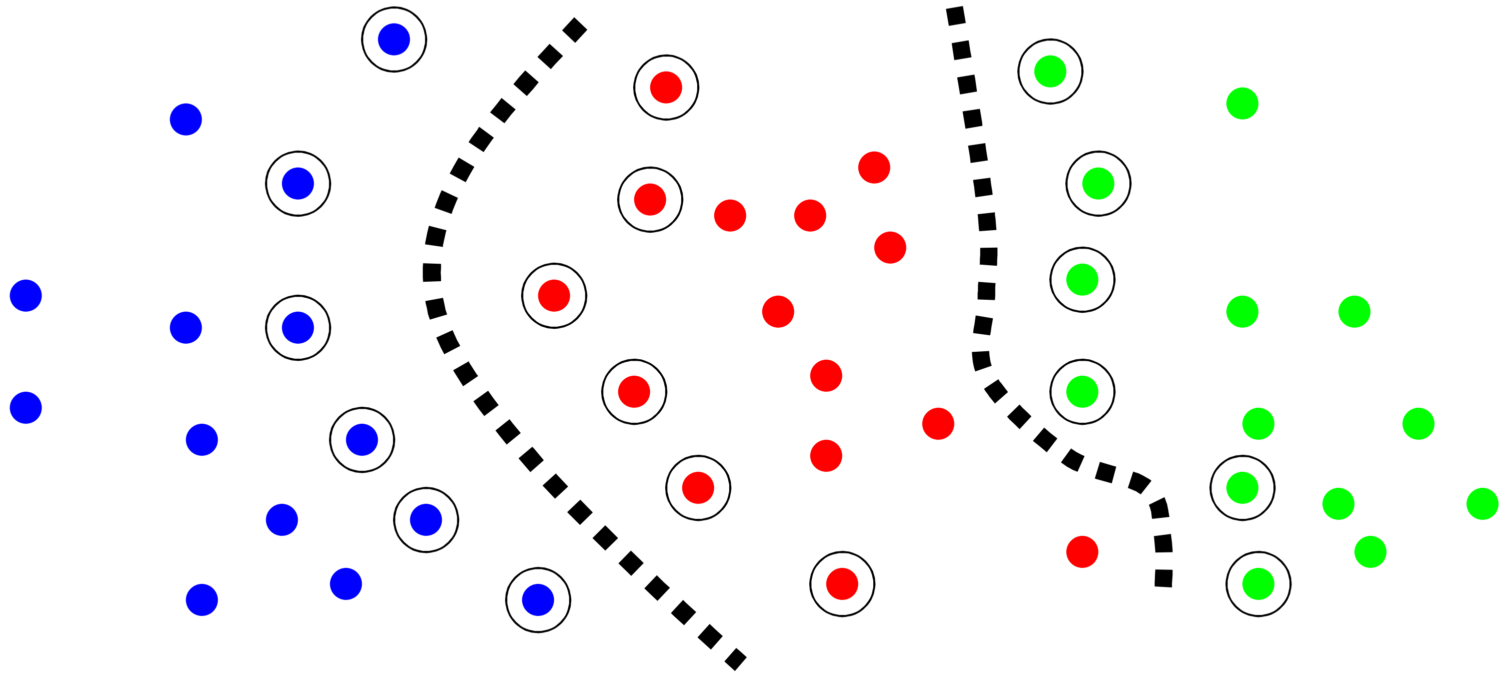
Data domain description

Standard SVM can do this well:



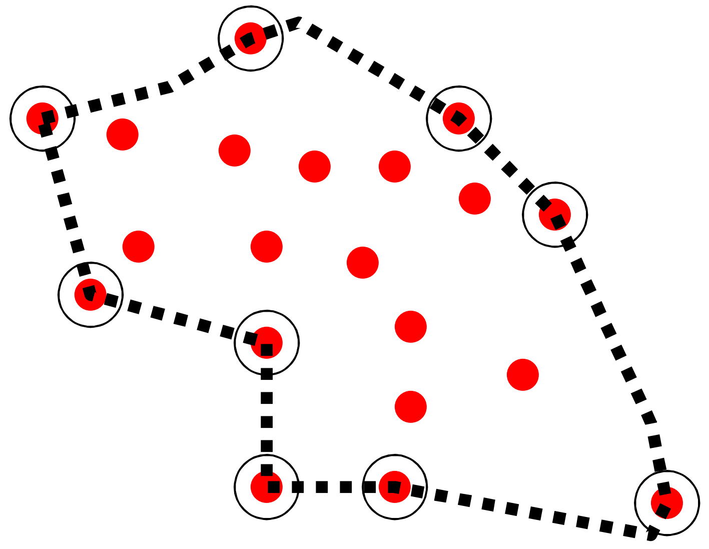
Data domain description

Standard SVM can do this somehow:



Data domain description

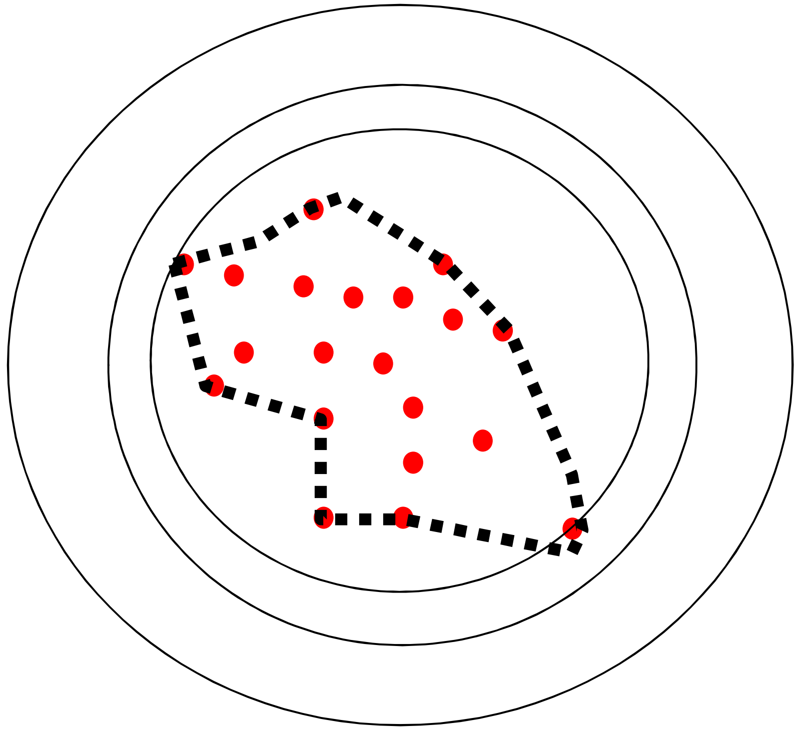
How about this?



Data domain description

Support vector domain description (SVDD) was developed

But, expensive to train it



Find a sufficiently compact and fit boundary, including almost all samples

Data domain description

Derivations in appendix 2

Algorithm:

To make: $(\mathbb{X}_i - \mathbf{a})^{\mathbf{T}}(\mathbb{X}_i - \mathbf{a}) \leq R^2 + \xi_i,$ **With minimum R**

Becomes:

$$\min L = 1 - \sum_i \alpha_i^2 - \sum_{i \neq j} \alpha_i \alpha_j K(\mathbb{X}_i, \mathbb{X}_j)$$

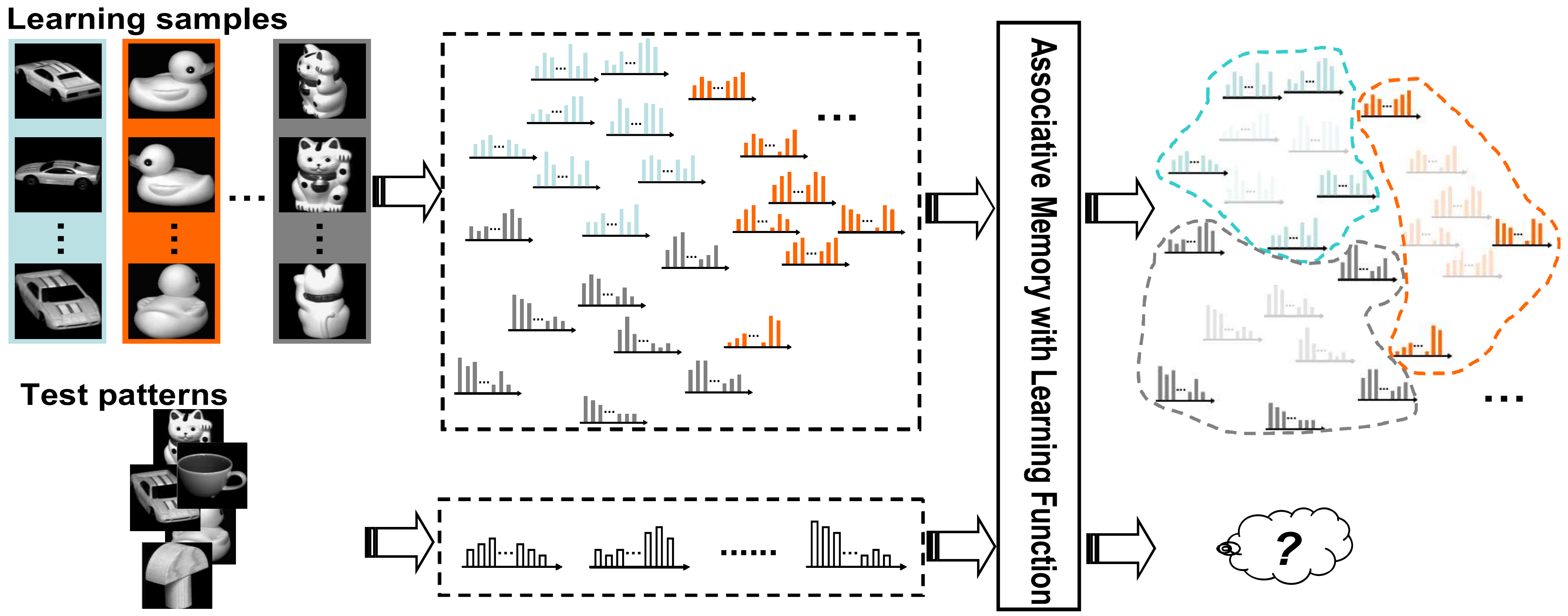
with:

$$\sum_i \alpha_i = 1$$

$$\begin{cases} \alpha_i \leftarrow \frac{1}{2}(\lambda - \sum_{j \neq i} \alpha_j K_{ij}) \\ \lambda \leftarrow \frac{1}{N}(1 + \sum_i \sum_j \alpha_j K(\mathbb{X}_i, \mathbb{X}_j)) \end{cases}$$

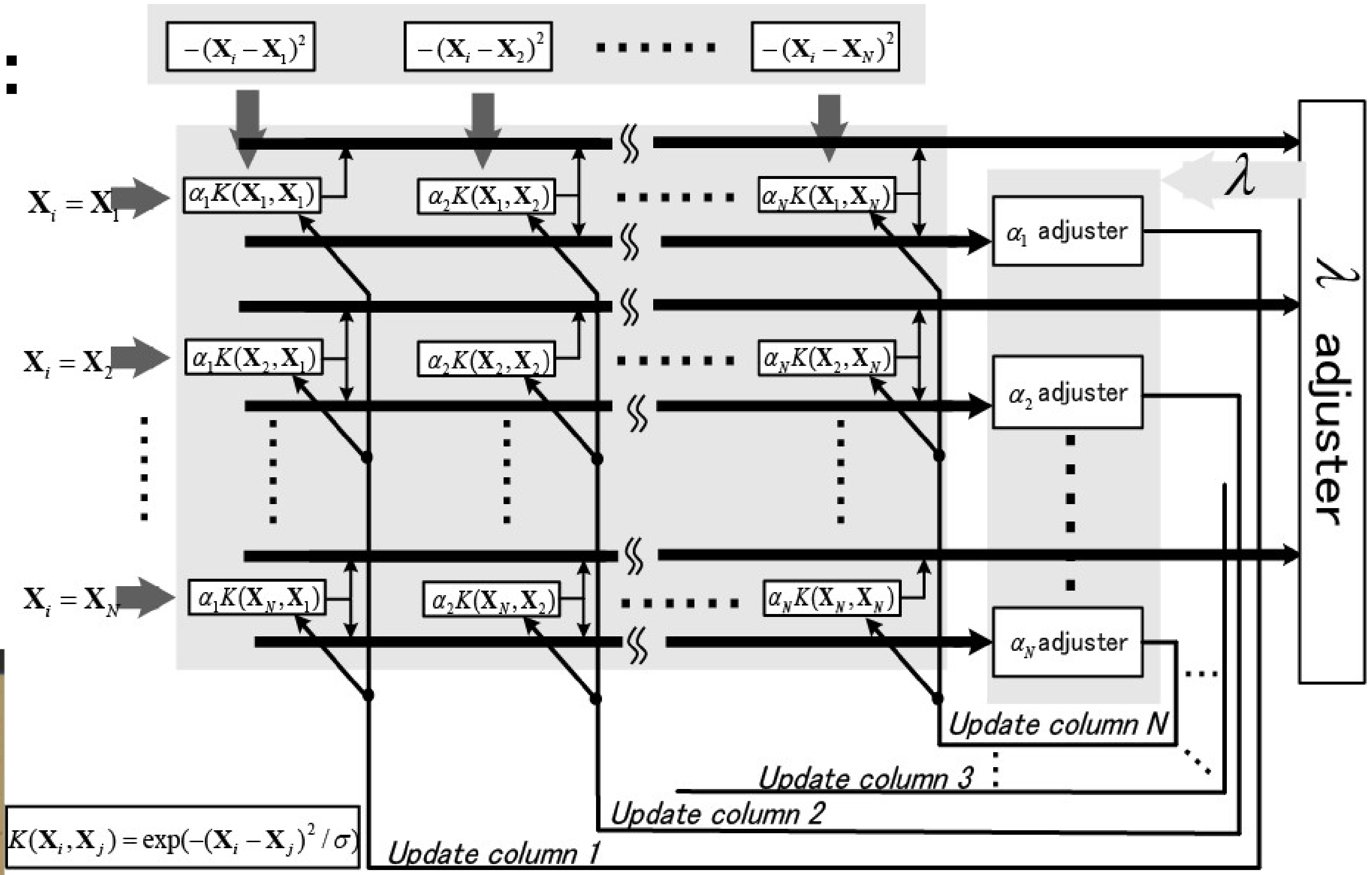
Data domain description

In the real-world applications:

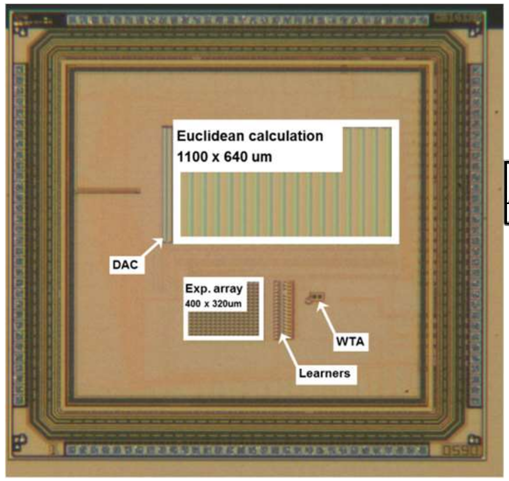


Data domain description

Architecture:



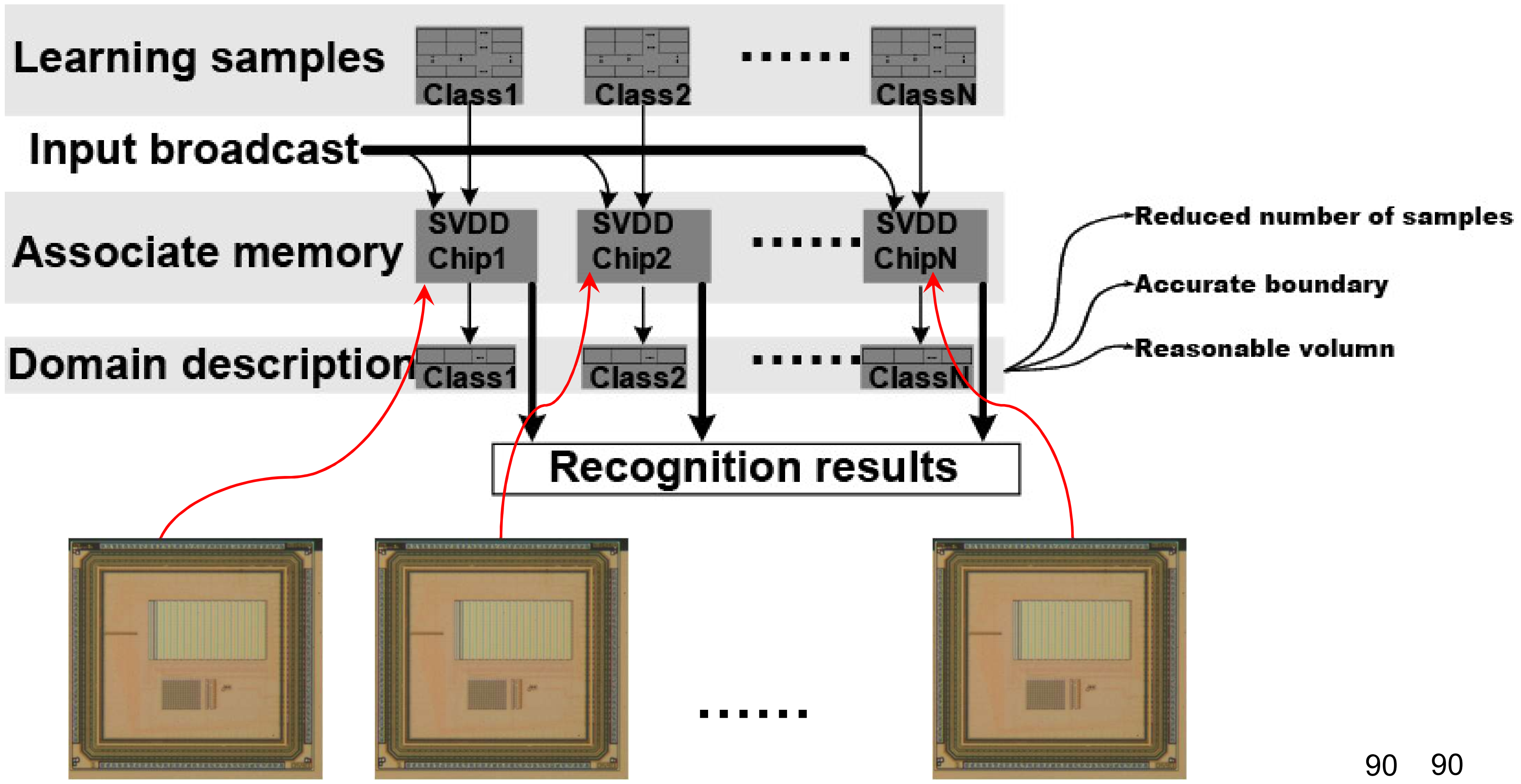
$-(\mathbf{X}_i-\mathbf{X}_1)^2$
$-(\mathbf{X}_i-\mathbf{X}_2)^2$
$-(\mathbf{X}_i-\mathbf{X}_N)^2$
$\mathbf{X}_i=\mathbf{X}_1$
$\mathbf{X}_i=\mathbf{X}_2$
$\mathbf{X}_i=\mathbf{X}_N$
$\alpha_1K(\mathbf{X}_1,\mathbf{X}_1)$
$\alpha_2K(\mathbf{X}_1,\mathbf{X}_2)$
$\alpha_NK(\mathbf{X}_1,\mathbf{X}_N)$
$\alpha_1K(\mathbf{X}_2,\mathbf{X}_1)$
$\alpha_2K(\mathbf{X}_2,\mathbf{X}_2)$
$\alpha_NK(\mathbf{X}_2,\mathbf{X}_N)$
$\alpha_1K(\mathbf{X}_N,\mathbf{X}_1)$
$\alpha_2K(\mathbf{X}_N,\mathbf{X}_2)$
$\alpha_NK(\mathbf{X}_N,\mathbf{X}_N)$
λ
α_1 adjuster
α_2 adjuster
α_N adjuster
λ adjuster
Update column N
Update column 3
Update column 2
Update column 1
$K(\mathbf{X}_i,\mathbf{X}_j)=\exp(-(\mathbf{X}_i-\mathbf{X}_j)^2/\sigma)$



$$\begin{cases} \alpha_i \leftarrow \frac{1}{2}\left(\lambda - \sum_{j\neq i}\alpha_j K_{ij}\right) \\ \lambda \leftarrow \frac{1}{N}\left(1+\sum_i\sum_j \alpha_j K(\mathbb{X}_i,\mathbb{X}_j)\right) \end{cases}$$

Data domain description

In the real-world applications:

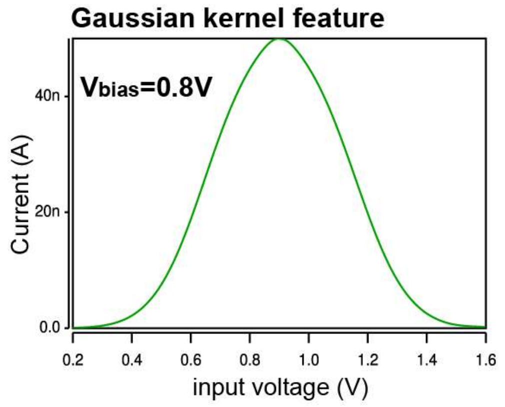


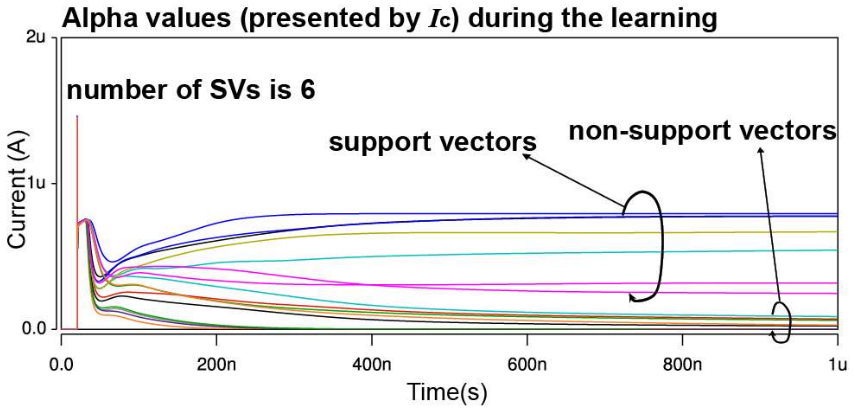
90

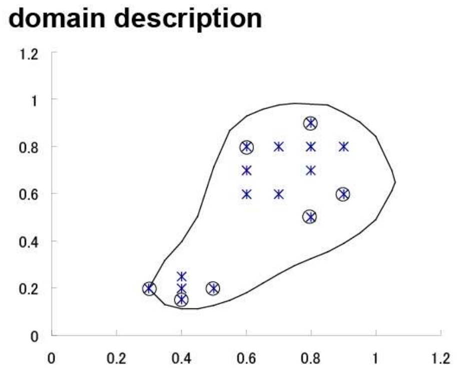
Data domain description

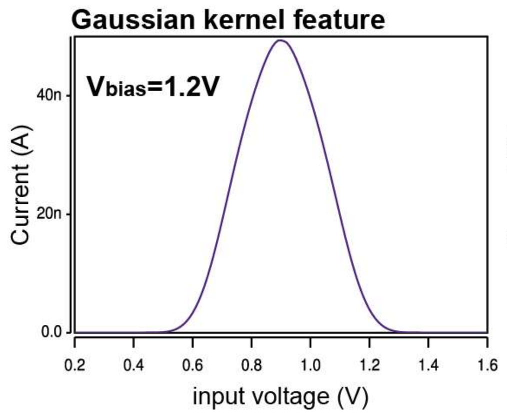
Using the fully parallel array:

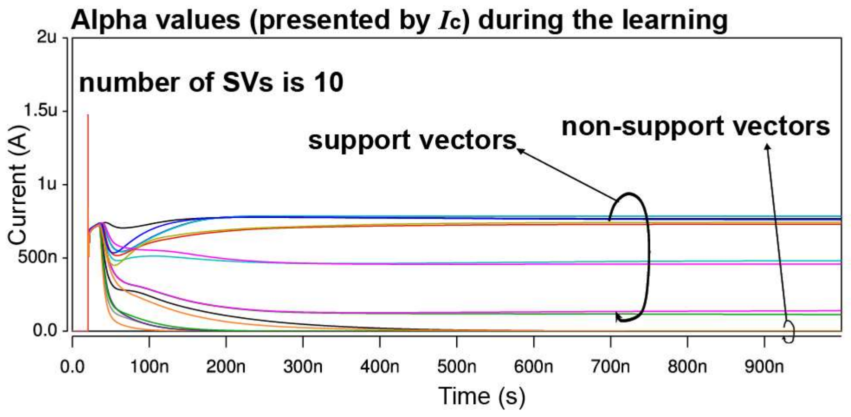
Simulation

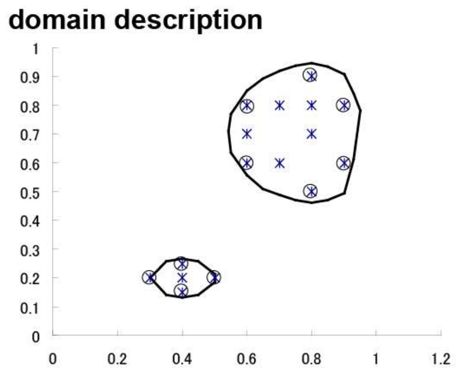




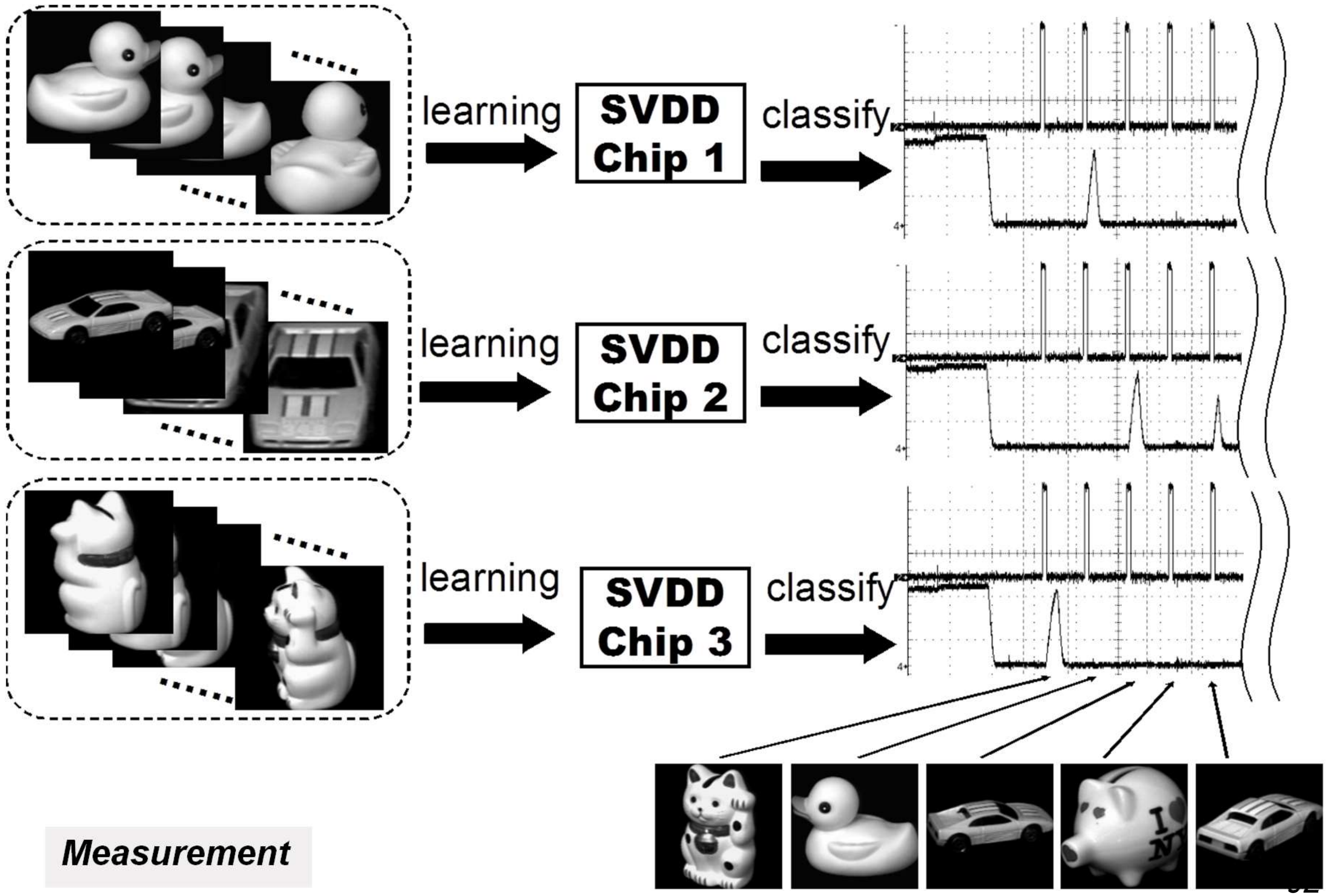








Data domain description



Measurement

Data domain description

Summary

SVDD is more similar to human perception, it can be implemented by the proposed architecture with mathematical tricks.

R. Zhang and T. Shibata, "A VLSI Hardware Implementation Study of SVDD Algorithm Using Analog Gaussian- Cell Array for On-Chip Learning", *IEEE Int. Workshop on Cellular Nanoscale Networks and their Applications*, Aug. 2012.

Last dance

Gap makes people hop

End

Thank you very much.