A Fully Parallel Analog VLSI Architecture for Implementing Learning Algorithms

Renyuan Zhang

Electrical Engineering

The University of Tokyo

A thesis submitted for the degree of

PhilosophiæDoctor (PhD)

March 26th, 2013

1. Reviewer: Professor Tadashi Shibata

2. Reviewer: Professor Kunihiro Asada

3. Reviewer: Professor Shuichi Sakai

4. Reviewer: Professor Akira Hirose

5. Reviewer: Professor Makoto Ikeda

6. Reviewer: Professor Yoshio Mita

Day of the defense: January 29th, 2013

Abstract

The cognitive functions play very important roles in the real-world tasks such as text analysis, audio processing and visual processing. In these cognitive tasks, the human brain is much superior to traditional very large scale integrated (VLSI) processors or software programs, since the brain can learn from samples autonomously. Therefore, plenty of machine learning algorithms have been developed to realize the learning operations, which were originally implemented by the software programs. Due to the reasons of power consumption and processing performances, a number of attempts to implement the machine learning algorithms were made by using hardware including graphic processing units (GPUs), field programmable gate array (FPGA), and VLSI circuits. Since many computations in the machine learning algorithms are very complex, the implementation costs including computing time and hardware utilization are greatly concerned. Furthermore, a large amount of iterations are always required by these algorithms, the learning speed is also a critical issue. Thus, the challenge on hardware implementations of learning algorithms lies on achieving a high processing speed with the consideration of limited hardware resource.

In this thesis, a fully parallel architecture for implementing learning algorithms is proposed by using analog VLSI circuits. Several analog circuitries are designed to carry out the complex functions such as Gaussian function and Euclidean distance. These computations in the learning algorithms can be done in real time within the compact chip area. On the basis of analog computational circuitries, a generally applied architecture in fully parallel is developed to implement some

machine learning algorithms. Since the chaos of analog signals is used for learning instead of clock-based numerical iterations, the learning operation is accomplished autonomously and self-converges with a high speed. Furthermore, the chip area and inner connection explosion problem in the traditionally parallel architectures can be prevented.

To verify the proposed architecture, the support vector machine (SVM) was implemented by VLSI circuits and fabricated in a complementary metal-oxide- semiconductor (CMOS) technology. SVM is one of most important supervised machine learning algorithms, which has been widely applied in the pattern recognition tasks. In fact, a number of VLSI implementations have been developed to realize the SVM on-chip learning. Since the kernel functions in SVM theory are always expensive to carry out by using digital circuits, the analog implementations of SVM algorithm were suggested by some works. There were two problems in the previously developed works. Firstly, the traditional analog circuits applied in these works generate a highly dimensional Gaussian function through single-dimension multipliers. The error intolerably increases as the dimension increases. Therefore, these works can be hardly implemented in highly dimensional pattern classification. The second problem is the trade-off between the learning speed and the chip size. Generally, there is a trade-off between the amount of circuits and the learning speed. A high processing parallelism realizes a high speed; however, it requires a large number of circuits. The number of learning iterations, which is usually very large and does not depend on the hardware parallelism, has a marked effect on the learning speed. Therefore, conventional VLSI implementations employing clock-based iterations consume much time on these iterations indifferently to the degree of hardware parallelism. In this work, the proposed fully parallel implementation of SVM was used in the image recognition problem. An analog Gaussian generation circuit, which is robust against process variations, is developed for highly dimensional pattern vectors. The center, height, and width of the generated Gaussian function feature can all be programmed

easily. Furthermore, the chip-area-hungry part for highly dimensional Euclidean distance computations and the much smaller part for exponential computation are built separately. Only the exponential computing circuits should be duplicated for a high degree of parallelism. In this manner, a fully parallel learning SVM processor was built within the compact chip area in a standard 0.18 um CMOS technology. Upon receiving highly dimensional pattern vectors, the learning process autonomously proceeded without any clock-based control and self-converged within a single clock cycle of the system (at 10MHz). To confirm the learning/classifying performance characteristics, sixteen object images from a database are converted into 64-dimensional vectors and fed into the proposed SVM processor as learning samples. After self-learning, several other vectors were used as test patterns. The proposed SVM processor classified all the testing patterns into correct classes according to the measurement results. The processing speed, chip area and power consumption performances are improved compared with the traditional approaches.

As a generally applied methodology, the proposed fully parallel architecture is also used to implement the unsupervised machine learning algorithms. On the basis of K-means mechanism, which is an important pattern clustering algorithm, a hardware efficient version is developed and named as K-Quasi-Centers (KQCs) method. From viewpoint of clustering results, the suggested scheme of clustering method has similar convergence performance to the original K-means algorithm. By implementing this modified clustering algorithm, the proposed analog fully parallel architecture can be applied to solve the unsupervised pattern clustering problem. The proof-of-concept processor was designed for 64-dimensional vectors categorization. In order to verify the performances of the proposed processor, sixteen images of two kinds of objects selected from the real image database were converted into feature vectors and fed into our KQC clustering processor. According to the circuit simulation results, all the images were correctly categorized into their respective classes even with

several different random initializations, and the categorization results self-converged with higher speed than conventional approaches.

From the above image processing applications, the proposed architecture performs a high processing speed and acceptable accuracy. However, the processing capacity of VLSI implementations is seriously limited by the chip size. One of the reasonable solutions to increase the number of learning samples is applying the on-line learning strategy, which was originally developed by software programs. In this work, the efficiency and importance of each learning sample are evaluated after the learning operation. The most inefficient sample is discarded to make the learning processor accept a new sample on-line. Employing the updated samples, the learning operation is repeated again. In this manner, a fixed VLSI processor can be used for the learning operation of a very large scale even unpredictable sample space. However, since on-line learning results in a large number of machine learning operations, this strategy is difficult to realize using software or traditional VLSI processors. Employing the proposed fully parallel architecture, the learning operations are accomplished with a high speed. Thus, this on-line learning strategy is efficient for the proposed architecture particularly. In order to verify the on-line learning performances, both SVM and KQC algorithms were implemented employing the analog fully parallel architecture for the image classification and clustering problems, respectively. From the circuit simulation results, the learning results are all correct with the consideration of on-line received samples. Furthermore, a visual tracking system was built by the combination of FPGA boards and the analog SVM processor developed by this work. Employing the on-line learning SVM, the object tracking performances were improved compared with those of conventional approaches.

Besides the pattern classification and clustering problems, another important task of machine learning is called data domain description. It was found that the data domain description has an enhanced

capacity for pattern recognitions. For instance, the SVM classification algorithm is originally for the two-class classification problems; but in the real-world applications, various numbers of classes might be required, even only a single class of learning samples is available in some applications. To solve these problems, a data domain description theory (also called one-class classification) was developed as an extension of SVM theory, which is named support vector domain description (SVDD). The SVDD algorithm has been applied in some classification problems, even unsupervised clustering problems by software programs. In this work, the SVDD algorithm has been implemented by our proposed analog fully parallel architecture. The proof-of-concept chip was built for the 64-dimensional pattern recognition. A multiple chip topology was proposed for multi-class recognition problems. For expending the classes, the number of chips can be freely increased. As an example, a three-class classification system employing three SVDD chips was built for real image recognition. After the on-chip learning session, several test images were fed in the system. From the chip measurement results, all the test patterns were correctly recognized.

As the extension of analog VLSI implementations for soft-computing tasks, we discuss how CMOS supporting circuitries can interface the fabric of nano devices with digital computing world. Using CMOS ring oscillators to emulate the nano oscillator behavior, how to produce the associative memory function and to use it for image recognition is demonstrated by circuit simulation.

Acknowledgements

First of all, I would like to sincerely express my thanks to my supervisor, Professor Tadashi Shibata, for his extremely valuable guidance and powerful supports during the past three years. His enthusiasm in teaching and research and his enduring encouragement led me to become a full fledged person. It is my great honor and fortunate to have taken him as my supervisor, and the experiences in this laboratory will be the treasure in my whole life.

I would like to thank Professor Kunihiro Asada, Professor Shuichi Sakai, Professor Akira Hirose, Professor Makoto Ikeda and Professor Yoshio Mita, for their preview of the thesis and their extremely valuable comments to my research. Their remarkable suggestions were indispensable for making my dissertation study successful.

I am very thankful to Professor Yoshio Mita for his instructive suggestions and comments. His kind supports and helps are quite important to this thesis.

I express my appreciation to Ms. Kimiko Mori for her help on so many documentaries, and to Ms. Motoko Inagaki for her helpful support.

I wish to express my sincere appreciation to those who have helped me in the past three years. I want to express my gratitude to all the lab members. During these three years, I received the most encouragement from them. I am so lucky to study and work with all of them.

I would like to thank my friends, Dr. Hongbo Zhu, Mr. Pushe Zhao, Mr. Ruihan Bao, Mr. Wenjun Xia and Mr. Zheye Wang for their help on my research and in my daily life. Whenever I need help, they are always there. Their support is of great importance and really precious to me. Being with them made my college life an amazing experience.

Thanks to the GCOE program for providing me with a platform to communicate with those excellent students. The activities gave me precious opportunities to learn from other people and to express my own idea. I received very important support from GCOE program.

Finally, I would like to thank my father, mother. Without their warm concerns and helps, it would be impossible for me to complete my study. I feel their warm care and support, which encourages me all the time.

Contents

List of Figures

LIST OF FIGURES

LIST OF FIGURES

List of Tables

1

Introduction

1.1 Background

For a long time, people have been trying to grasp how human come to understand the world. Human brain builds rich causal models, make strong generalizations, and construct powerful abstractions, whereas the input data are sparse, noisy, and ambiguous–in every way far too limited (1). Via these models, human brain is much superior to the conventional artificial systems in the cognitive functions. Thus, it is very attractive but challenging to understand how the brain builds these models, even construct similar models by artificialities. Considering the situation of a child learning the meanings of words, any parent knows, and scientists have confirmed, that typical 2-yearolds can learn how to use a new word such as "horse" or "hairbrush" from seeing just a few examples (2, 3). Within the infinite landscape of all possible objects, there is an infinite but still highly constrained subset that can be called "horse" and another for "hairbrush" in children's mind. This natural phenomenon is called "learning the knowledge".

In order to achieve the learning functions by artificialities, the theory of machine learning (ML) has been introduced (5). Machine learning is the study of computer algorithms capable of learning to improve their performance of a task on the basis of their own previous experience (7). The training examples come from some generally unknown probability distribution and the learner has to extract from them something more general, something about that distribution, that allows it to produce useful predictions in new cases (4). As an engineering field,

1. INTRODUCTION

ML has become steadily more mathematical and more successful in applications over the past decades. These applications of ML were not only limited in the fields of mathematics (8) but also in the surprisingly wide fields of engineering (9), science (11), even business (10). Essentially, the tasks of machine learning cover the following types of problems: pattern recognition, regression estimation and density estimation (6). The focus of this thesis is the problem of pattern recognition, which mainly includes the tasks of pattern classification (20), clustering (46) and data domain description (67).

To solve these problems, various mathematical algorithms have been developed for many years. Most of these algorithms were originally implemented by software programs. However, in some specific fields of applications, the hardware implementations of machine learning algorithms are required (12, 13, 16) on the basis of VLSI circuits. Specialized machine learning hardware (which can either support or replace software) offers appreciable advantages in these situations as can be traced as follows (14):

1. Speed: Specialized hardware can offer very high computational power at limited price and thus can achieve several orders of speed-up, especially in the neural domain where parallelism and distributed computing are inherently involved.

2. Cost: A hardware implementation can provide margins for reducing system cost by lowering the total component count and decreasing power requirements. This can be important in certain high-volume applications, such as ubiquitous consumer-products for real-time image processing,that are very price-sensitive.

3. Graceful degradation: An intrinsic limitation of any sequential uni-processor based application is its vulnerability to stop functioning due to faults in the system. As it was suggested by some works (15), even with some powerful general purpose equipment such as multi-core-CPU computers, an efficient error-tolerance mechanism is still present. Especially in most of machine learning applications, the performances are not greatly sensitive to the computational accuracy.

Many types of hardware, which include general purpose graphic processing units (GPUs), FPGAs, digital and analog VLSI circuits, were applied to implement machine learning algorithms. In this thesis, the analog VLSI circuits for implementing learning algorithms are interested, since the analog elements benefit by exploiting simple physical effects to carry out some of complex functions directly. This property is helpful to generate a high parallelism for implementing on-chip learning operations. However, the accuracy and reliability of analog circuits should be carefully considered.

1.2 Learning Algorithms in This Thesis

Among various machine learning algorithms, this thesis focuses on support vector machine (SVM), K-means-like clustering and support vector domain description (SVDD) particularly since they are very typical and important algorithms to solve pattern recognition problems.

SVM (21) was originally developed in computer science field to solve binary classification problems, which is one of most important supervised machine learning algorithms. The main idea of SVM is to separate the classes with a plane that maximizes the margin between them. The objective function is then optimized by solving a large-scale Quadratic Programming (QP) problem with linear and box constraints (29). However, the learning sample patterns in the real-world applications are usually in the form of high dimensional vectors, even non-linear separable. Thus, the so-called "kernel tricks" are necessary to map the learning sample vectors nonlinearly into a higher-dimensional feature space via kernel functions, and construct a separating hyperplane with maximum margin there. This yields a nonlinear decision boundary in input space. By the use of a kernel function, it is possible to compute the separating hyperplane without explicitly carrying out the map into the feature space. Among plenty of kernel functions, the Gaussian function was found one of most powerful kernel in SVM learning problems. The SVM algorithm with Gaussian function kernels is the interest of this work.

Not only the supervised but also the unsupervised machine learning algorithm is interested in this thesis. The K-means algorithm is widely used in pattern

recognition, data mining, and image segmentation as a very powerful learning tool. It is a typical unsupervised clustering method. Using this algorithm, learning samples given in the form of multidimensional vectors can be partitioned into a limited number (K) of clusters according to their feature similarity without supervision. In order to implement the K-means like clustering algorithms by a fully parallel architecture, a modified scheme of the original K-means is proposed in this thesis, which is named K-Quasi-Centers (KQCs) method. This KQCs method has very similar mathematical properties to the original K-means, but it is friendly to implement by the fully parallel architecture.

The general problem of hardware implementations of learning algorithms is the trade-off between the hardware utilization and processing capacity. In this work, an on-line-learning strategy is applied to design the analog on-chip learning processors with the consideration of limited hardware resource. In this manner, both proposed SVM and KQCs learning processors can be used for the on-chip learning of large (even unpredictable) scale sample space.

In addition, the problem of data domain description is considered in this work. In domain description the task is not to distinguish between classes of samples like in classification problems or to produce a desired outcome for each input sample like in regression problems, but to give a description of a set of samples. This description should cover the class of samples represented by the training set, and ideally should reject all other possible samples in the sample space. Here, the data domain description employing the support vector mechanism is interested, which is called support vector domain description (SVDD), or one-class SVM classification (OCSVM) (67).

1.3 Applications in This Thesis

Machine learning techniques were applied to different fields as natural language processing, medical diagnosis, bio-informatics, efficient search engine design, mobile learning, classifying DNA sequences, speech and handwriting recognition, object identification in computer vision, game playing, robot locomotion, stock market analysis, chem-informatics, detecting credit card fraud in financial institutions (16). In this thesis, we are particularly interested in the image recognition

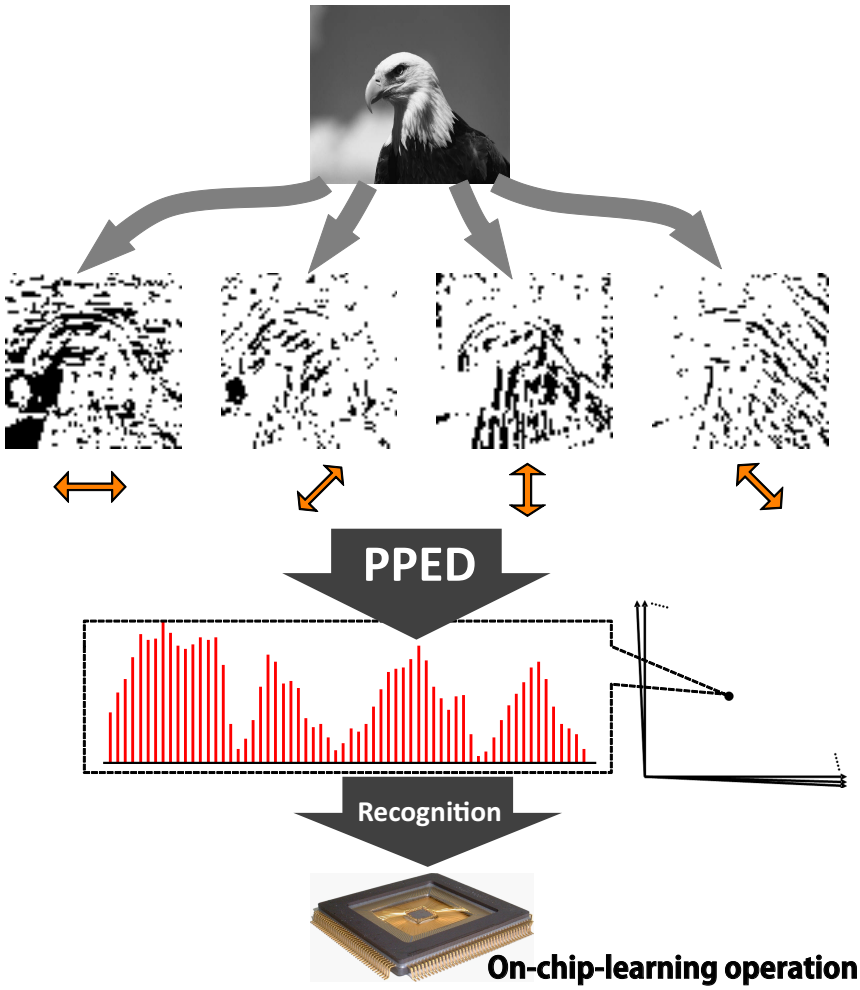


Figure 1.1: Preliminary processes for the image recognition tasks in this thesis.

problems including the supervised classification and unsupervised categorization of real images. Regarding the image processing flows, some preliminary processes are necessary to efficiently represent the real images in the hardware-friendly forms, multi-dimensional vectors for instance. Figure 1.1 illustrates the preliminary processes applied in this work before the recognitions. Here, a vector generation procedure which is called projected principal-edge distribution (PPED for short) algorithm (42) is used to convert two-dimensional image data to a vector representation. The input image is first subjected to pixel-by-pixel spatial filtering operations to detect edges in four directions: horizontal (H); $+45^0$; vertical

(V); and -45^0. The number of projection sums for each direction is combined in one vector with 64 dimensions. In this manner, the main features of an input image can be extracted by a 64-dimensional vector representation, which is called feature vectors. After these preliminary processes, the feature vectors are fed into the on-chip learning processor for recognition.

1.4 Related Works

In order to implement the learning algorithms by hardware, two important issues are concerned. Firstly, the VLSI circuits for carrying out necessary computations (usually complex) should be particularly designed according to different algorithms. Secondly, a reasonable VLSI architecture is important to realize the learning operations. Many approaches have been explored to solve these problems during the past decades.

1.4.1 Error-Tolerance Computation

For the image recognition problems, the computations are always among high dimensional vectors. Furthermore, in each interested machine learning algorithm in this thesis, complex computations such as highly dimensional Gaussian function (for SVM and SVDD) and Euclidean distance calculation (For KQCs) are needed. Thus, the implementations of these computations are expensive in silicon. Fortunately, the learning performances are usually not very sensitive to the absolute accuracy of computations in the image recognition problems (64). There are many approaches being explored to carry out the error-tolerant computations by simple VLSI circuits. Here, the Gaussian generation circuits, which are widely applied but not limited in the SVM algorithm, are used as examples.

Since the exact calculation of Gaussian function by traditional digital circuits requires a long processing time and expensive hardware utilization, some digital Gaussian generation circuits used the linear piece function (LPF) to mimic the mathematical behavior of actual Gaussian (37). In their experiments, two or three pieces of linear function can generate an acceptable performances in pattern

recognition instead of actual Gaussian. The cost of hardware implementation was greatly reduced.

To make further reduction of implementation cost, several attempts of analog Gaussian generation circuits were also made on the basis of so-called "bump circuit" (38). In this type of implementations, the differential pairs operating in the sub-threshold region are employed to generate the Gaussian-like function feature. Other differential-pair-based circuits with improved performances were even designed (19, 39) considering some practical issues. However, the MOS transistors operating in sub-threshold region are usually sensitive to the process variations. In addition, this kind of circuits generate a highly dimensional Gaussian function through single-dimension multipliers (41). The error intolerably increases as the dimension increases. Therefore, these works can be hardly implemented in highly dimensional pattern classification. On the other hand, some other analog circuit designs (17, 18) based on the translinear principle are also not convenient to implement the learning algorithm due to their poor programmability on function feature. Consequently, a robust highly dimensional Gaussian generation circuit, which is compact and easy to program, is desired in this work.

1.4.2 Traditional VLSI architectures for Implementing Learning Algorithms

Here, the VLSI implementations of SVM algorithms are used as examples again. From the essential point of view, the learning operation is to pursue suitable parameters for the recognition models by iterative updates. Namely, a large amount of numerical iterations are needed by traditional approaches, which are expensive in silicon. Some early presented works carry out the learning process off-chip by using software programs (30, 31) to avoid the expensive on-line learning implementations. Recently, the feasibility to implement on-chip learning algorithms were also investigated, where the trade-off between the learning speed and the chip size was concerned. Generally, there is a trade-off between the amount of circuits and the learning speed on the basis of parallelism. A high hardware parallelism realizes a high speed; however, it requires a large number of circuits.

Table 1.1: General discussion of VLSI implementation for SVM on the basis of parallelism.

	Hardware	Step control	Speed	Chip area	Flexibility
Serial	Digital(36)	clock-based iteration	low	fair	good
Partially parallel	Analog(41)	clock-based iteration	fair	small	poor
Fully parallel	Analog(40)	analog chaos free feedback	high	large	poor

The number of learning iterations, which is usually very large and does not depend on the hardware parallelism, has a marked effect on the learning speed. Therefore, conventional VLSI implementations employing clock-based iterations consume much time on these iterations indifferently to the degree of hardware parallelism.

Referring some previously proposed works, a general discussion of VLSI implementation for SVM algorithm is made by Tab. 1.1 on the basis of parallelism. The example of digital implementation was one of typical serial architectures, which achieves very well flexibility. An attempt of partially parallel architecture was made with considerations of both learning speed and chip area. A high processing speed can be achieved within very compact chip area by this approach. Regarding the traditional fully parallel architecture in this example, the chaos of analog signals is used for the learning operation instead of numerical iteration. Therefore, the learning speed in this work is extremely high.

The circuit organization of tradition fully parallel architecture is illustrated in Fig. 1.2, which was designed to train a set of N samples $\mathbb{X}_i$s ($i = 1, 2, \ldots, N$). From the essential point of view, the learning operation of SVM is to pursue the suitable values for N parameters (α_is in this figure) by feedback updates. Here, the freely feedback of analog signals were applied, and directly effect the computational elements in real-time. That is the reason that this architecture achieved extremely high learning speed. However, it is found that a large number

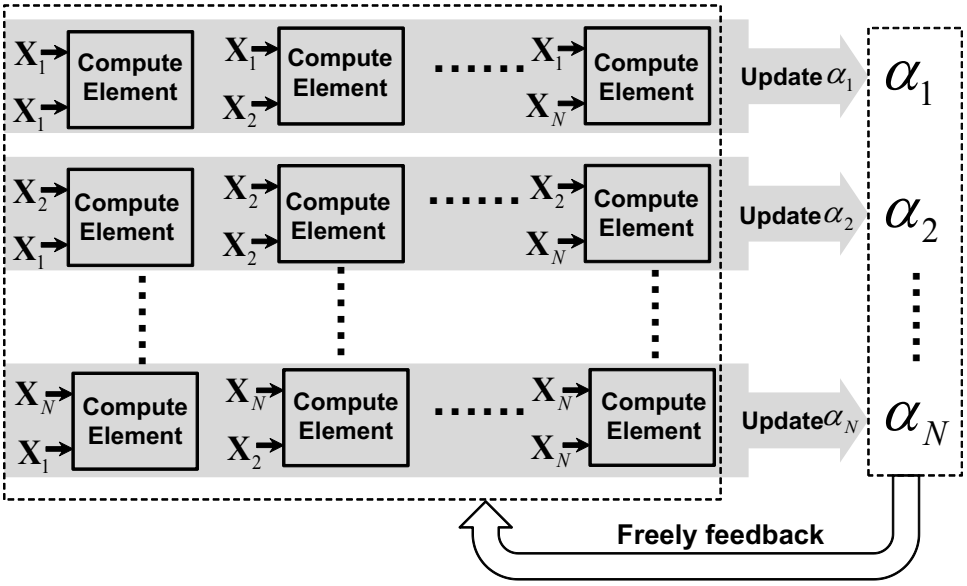


Figure 1.2: Circuit organization of tradition fully parallel architecture for implementing SVM learning algorithm.

of computational elements (in proportion to square of the number of learning samples) are needed by this architecture. Due to the chip size limitation, it is not practical especially for the highly dimensional applications. A general problem of these two parallel implementations is the flexibility. As soon as the hardware is designed, the numbers of dimensions and samples are fixed and hardly changed according to various applications.

1.5 Proposed Fully Parallel Analog VLSI Architecture

The purpose of this thesis is to propose a fully parallel architecture for implementing learning algorithm, which can achieve a high on-chip-learning speed within a compact chip area. Since we are interested in the highly dimensional pattern recognition, applying traditional fully parallel architecture is not practical in the real-world applications. Thus, a new topology is suggested as it is shown in Fig. 1.3.

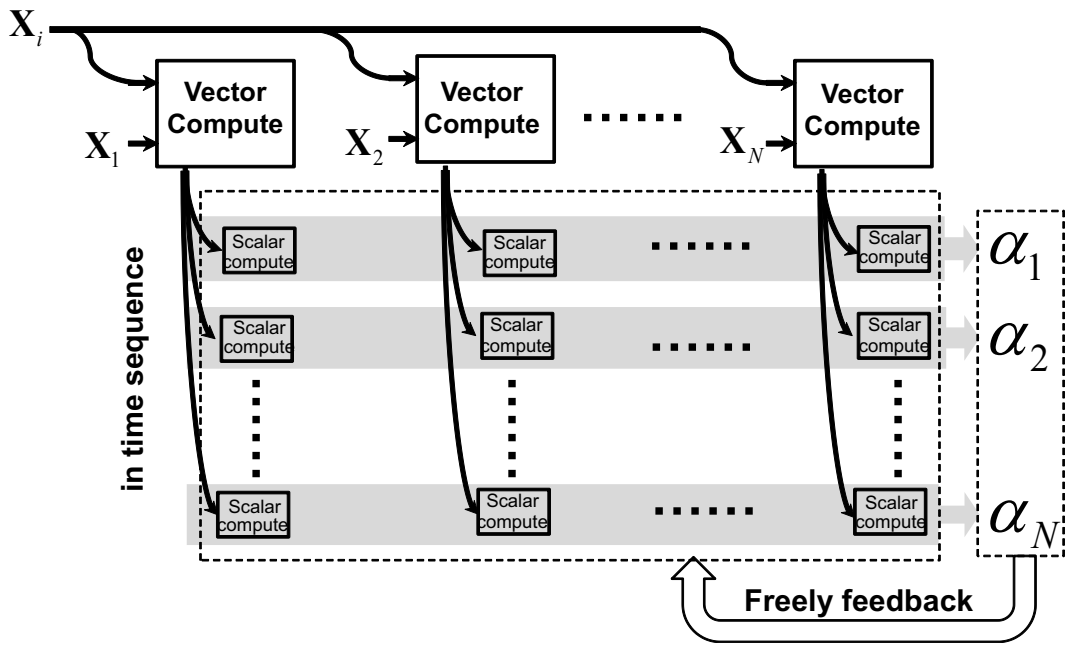


Figure 1.3: Circuit organization of proposed fully parallel architecture for implementing learning algorithm.

The chip-area-hungry part for computations among highly dimensional vectors and the much smaller part for computations among scalars are designed separately. By designing specific computational circuits and adjusting the algorithms, the learning operations (real-time updates of parameters) only effect the scalar computation elements. Namely, Only the computing circuits for scalars should be duplicated as an array for a high degree of parallelism. Before the learning, complex computations among vectors are carried out by a row-parallel circuitry, and the results are fed into the fully parallel array in time sequence. During the learning operation, only the fully parallel array is active, which is also helpful to reduce the power consumption. In this manner, a fully parallel learning processor can be built within the compact chip area. Obviously, some specific computational circuitries should be designed for this purpose, which will be described in detailed in the following parts.

1.6 Organization of This Thesis

The rest part of this thesis is organized as follows. The fully parallel analog VLSI implementation of SVM algorithm is described in Chapter 2 along with verification examples of image classification. As a general applied methodology, the proposed fully parallel architecture can also implement a pattern clustering algorithm named K-Quasi-Centers. The circuit design and verification for image clustering application are presented in Chapter 3. In order to extend the number of learning samples for both SVM and KQCs implementations, an on-line-learning strategy is proposed with the consideration of limited hardware resource in Chapter 4. Chapter 5 describes the proposed analog implementation of SVDD algorithm, which has an advanced capacity in pattern classification problems. The conclusion of this thesis is made in Chapter 6.

It should be noted that, the pattern recognition problem can be solved by using not only machine learning algorithms but also some other soft-computing technologies based on the associative memory function. Building associative memories based on the physics of nano oscillators presents a lot of potential for pattern recognition. Using CMOS ring oscillators to emulate the nano oscillator behavior, how to produce the associative memory function and to use it for image recognition using CMOS ring oscillators to emulate the nano oscillator behavior is demonstrated in the appendix of this thesis.

2

Fully Parallel Support Vector Machine Processor Employing Analog Circuitry

An analog support vector machine (SVM) processor employing a fully parallel self-learning circuitry was developed for the classification of highly dimensional patterns. To implement a highly dimensional Gaussian function, which is the most powerful kernel function in classification algorithms but computationally expensive, a compact analog Gaussian generation circuit was developed. By employing this proposed Gaussian generation circuit, a fully parallel self-learning processor based on an SVM algorithm was built for 64 dimension pattern classification. The chip real estate occupied by the processor is very small. The object images from two classes were converted into 64 dimension vectors using the algorithm developed in a previous work and fed into the processor. The learning process autonomously proceeded without any clock-based control and self-converged within a single clock cycle of the system (at 10 MHz). Some test object images were used to verify the learning performance. According to the circuit simulation results, it was shown that all the test images were classified into correct classes in real time. A proof-of-concept chip was designed in a 0.18 μm complementary metal oxide semiconductor (CMOS) technology, and the performance of the proposed SVM processor was confirmed from the measurement results of the fabricated chips.

2.1 Introduction to the Proposed SVM On-Chip Learning Processor

The human brain is very superior to traditional very large scale integrated (VLSI) processors or software programs in cognitive tasks, such as pattern recognition, since the brain can learn from samples autonomously. Thus, a number of algorithms for machine learning functions have been developed over many years.(20) Among them, the support vector machine (SVM) is known as one of the most powerful algorithms (21) and has been applied to a number of pattern classification problems, such as text categorization (22, 23, 24), audio processing (25), and image classification (26, 27, 28). However, they were mostly implemented by a software program. Recently, some attempts have been made to implement the SVM algorithm directly in silicon (30, 31) for pattern classification. However, since the learning processes of these works were not implemented on-chip, additional off-chip learning sessions are required to activate the systems.

Several VLSI implementations have realized on-chip learnable SVMs with linear or quadratic kernel functions (32, 33, 34). It is found that the SVM with a Gaussian kernel has an enhanced capability in classifying linearly nonseparable patterns (35). Therefore, the on-chip learnable SVM with a Gaussian function kernel was proposed (36) by employing digital circuits. On the other hand, the Gaussian function is computationally expensive when it is computed by digital VLSI circuits (37). To compute the Gaussian function with a higher speed and a lower cost in silicon, various types of analog Gaussian generation circuit have been explored and applied to pattern classification.(38, 39, 40)

On the basis of analog Gaussian generation circuits, an on-chip learnable SVM employing a fully parallel learning circuitry was developed(40). The chip processed only four learning sample vectors and its performance was evaluated by circuit simulation. Both the learning and classification speeds of this work were improved compared with those of the previously proposed works. However, the fully parallel learning circuitry requires a large chip area in proportion to the square of the number of sample data, which is not practical. To reduce the chip size, another on-chip learnable Gaussian kernel SVM processor was built (41)

by employing a row-parallel learning circuitry with a learning capacity of twelve sample vectors. However, the vectors have only two dimensions.

There were two problems in the two works mentioned above. Firstly, the traditional analog circuits applied in these works generate a highly dimensional Gaussian function through single-dimension multipliers. The error intolerably increases as the dimension increases. Therefore, these works can be hardly implemented in highly dimensional pattern classification. The second problem is the trade-off between the learning speed and the chip size. Generally, there is a trade-off between the amount of circuits and the learning speed. A high processing parallelism realizes a high speed; however, it requires a large number of circuits. The number of learning iterations, which is usually very large and does not depend on the hardware parallelism, has a marked effect on the learning speed. Therefore, conventional VLSI implementations employing clock-based iterations consume much time on these iterations indifferently to the degree of hardware parallelism.

The purpose of this work is to develop a fully parallel learning SVM processor with high learning speed and compact chip size for highly dimensional pattern classification. An analog Gaussian generation circuit, which is robust against process variations, was developed for highly dimensional pattern vectors. The center, height, and width of the generated Gaussian function feature can all be programmed easily. Furthermore, the chip-area-hungry part for highly dimensional Euclidean distance computations and the much smaller part for exponential computation are built separately. Only the exponential computing circuits should be duplicated for a high degree of parallelism. In this manner, a fully parallel learning SVM processor was built within the compact chip area in a standard 0.18 μm complementary metal oxide semiconductor (CMOS) technology. Upon receiving highly dimensional pattern vectors, the learning process autonomously proceeded without any clock-based control and self-converged within a single clock cycle of the system (at 10 MHz). To verify the learning/classifying performance characteristics, 16 object images from a database were converted into 64-dimensional (64D) vectors and fed into the proposed SVM processor as learning samples. After self-learning, several other vectors were used as test patterns. The proposed SVM processor classified all the testing patterns into correct classes according

to the simulation results. A proof-of-concept chip designed in a 0.18 μm CMOS technology was fabricated and the performance of the proposed SVM processor was confirmed from measurement results.

The remaining parts of this chapter are organized as follows: in Section 2.2, the process of image pattern classification and the SVM algorithm applied in this work are briefly introduced. In Section 2.3, the circuit architecture of the proposed self-learning SVM processor is described. In Section 2.4, the experimental verifications are shown by considering both simulation and measurement results. Finally, summary is presented in Section 2.5.

2.2 Preliminaries

2.2.1 Process of image pattern classification applied in this work

This work is carried out to classify the object images into their respective categories with a two-class configuration. A set of images with known class labels are given as learning samples. After a machine learning process, other images used as test patterns are expected to be recognized according to the learning samples.

The process of image pattern classification realized by this work is illustrated in Fig. 2.1. The main features of images are extracted as highly dimensional vectors (64D in this work) by employing the projected principle edge distribution (PPED) method (42). In this manner, the task of image recognition is converted to classify the respective highly dimensional vectors. The vectors representing learning samples are fed into the proposed SVM processor operating in the learning mode. After learning, several significant vectors (named support vectors) are identified. In the classifying mode, the vectors that represent test image patterns are fed into the SVM processor. The classification results are output in the form of a binary signal indexing the class label.

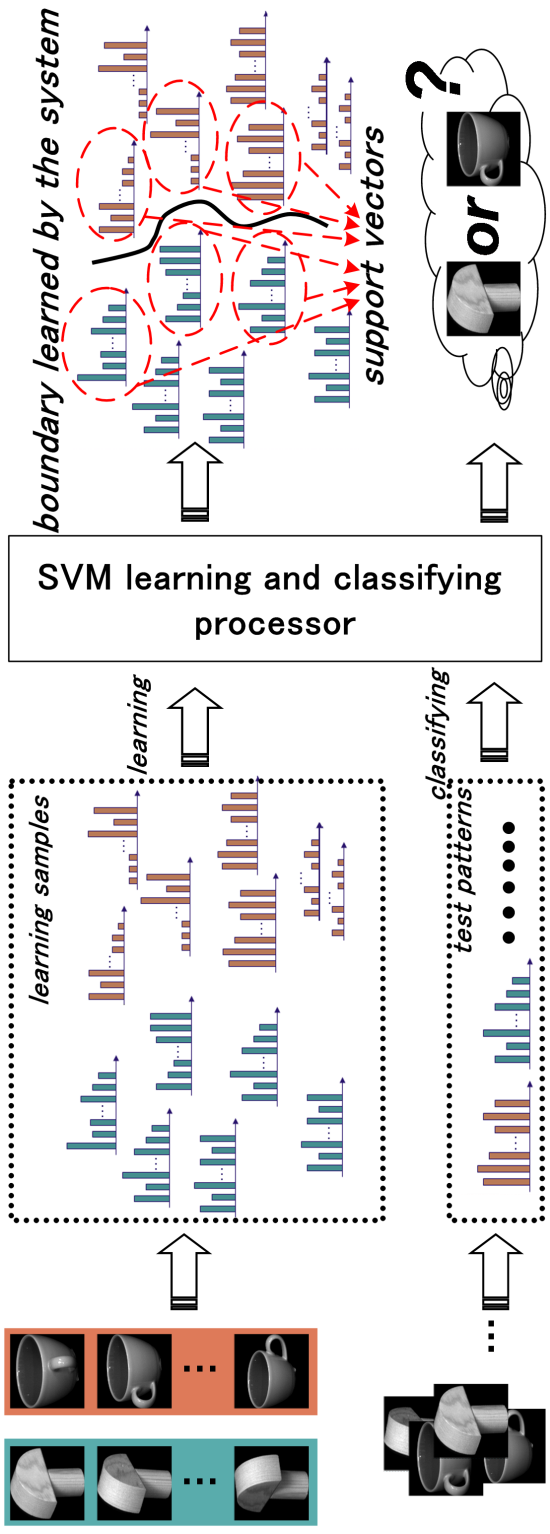


Figure 2.1: Process of image pattern classification applied in this work.

2.2.2 Hardware-friendly SVM algorithm

The SVM algorithm is usually used as a binary classifier to classify the n-dimensional vectors $\mathbb{X}$s with the form of $\mathbb{X} = (x_1, x_2, \ldots, x_n)$. A set of learning samples $(\mathbb{X}_i, y_i)_{1 \leq i \leq N}$ is reqiured, where N is the number of learning samples and $y_i \in \{-1, 1\}$ is the class label of the i-th sample $\mathbb{X}_i$. Upon receiving a vector $\mathbb{X}$, the decision function used to classify this vector can be given as

$$f(\mathbb{X}) = sign[\sum_{i=1}^{N} \alpha_i y_i K(\mathbb{X}, \mathbb{X}_i) + b], \tag{2.1}$$

where $K(\mathbb{X}, \mathbb{X}_i)$ is the kernel function. A learning process used to obtain the suitable coefficients α_i and bias b is realized by solving

$$\max_{b} \min_{0 \leq \alpha_i \leq C} [W(\alpha, b) = \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j K(\mathbb{X}_i, \mathbb{X}_j) - \sum_{i=1}^{N} \alpha_i + b \sum_{i=1}^{N} \alpha_i y_i], \tag{2.2}$$

where C is a regularization parameter. A hardware-friendly type of SVM algorithm (**?**) is applied in this work to obtain α_i by backward propagation. The updating rule can be given by

$$\alpha_i \leftarrow \alpha_i - \eta_i \frac{\partial W(\alpha, b)}{\partial \alpha_i}, \tag{2.3}$$

where the sufficient condition of convergence is $0 \leq \eta_i \leq 2/K(\mathbb{X}_i, \mathbb{X}_i)$. By considering the kernel as a Gaussian function as $K(\mathbb{X}_i, \mathbb{X}_j) = exp(-\gamma(\mathbb{X}_i - \mathbb{X}_j)^2)$, η_i can be set as $\eta_i = 1/K(\mathbb{X}_i, \mathbb{X}_i) = 1$. In this manner, the updating rule is represented as

$$\alpha_i \leftarrow 1 - y_i(\sum_{j(\neq i)} \alpha_j y_j K(\mathbb{X}_i, \mathbb{X}_j) + b). \tag{2.4}$$

Since the bias b has a negligible effect on the performance in this case (36, 41), it is set as 0 in both learning and classifying modes. Thus, the updating rule is transformed into the following form with the consideration of C:

$$\alpha_i \leftarrow \min(C, \max(0, 1 - y_i \sum_{j(\neq i)} \alpha_j y_j K(\mathbb{X}_i, \mathbb{X}_j))). \tag{2.5}$$

In the classification mode, the obtained α values are kept constant and recalled by the decision function.

2.3 Hardware Implementation

2.3.1 Circuitry organization of the proposed SVM processor

By assuming the total number of training samples N, the circuitry organization of the proposed fully parallel learning SVM processor is illustrated in Fig. 2.2. N vectors in the form of digital data are used as inputs. A set of on-chip digital-to-analog converters (DACs) is designed to convert the input vectors into analog forms. N sets of Euclidean distance calculation circuits are constructed in block I to compute the distances between the vector $\mathbb{X}_i$ and all other samples in parallel. The class label y_i is reflected by the switches connected to the row buses.

Each cell in block II contains a capacitor (as an analog memory) and an exponential generation circuit, which generates the final output as a Gaussian function in the current mode. The Euclidean distance values are stored in array II row by row as voltages. As a result, a fully parallel array of Gaussian kernels is implemented in such a small area even for the high dimensionality (64 dimensions of sample vectors of the presented work). The circuits of the α adjuster are current mirror-based adders/subtracters. By collecting all the currents on the row bus, the α adjusters realize the function represented by eq. 2.5.

During the learning process, the α values in block III are fed back to block II and the learning process proceeds autonomously in a fully parallel manner. Therefore, the training process can be accomplished only in a single clock cycle. This is far faster than the clock-based sample-serial iterative approach (41). In the classifying mode, the α values obtained by the learning process are kept constant to realize the function represented by eq. 2.1.

2.3.2 Proposed analog Gaussian generation circuit

To implement a highly dimensional Gaussian function with a low cost in silicon, an analog Gaussian generation circuit was designed, as shown in Fig. 2.3. Vectors with a maximum number of dimensions of 64 are used as inputs in the form of voltage, and the output is given in the form of current.

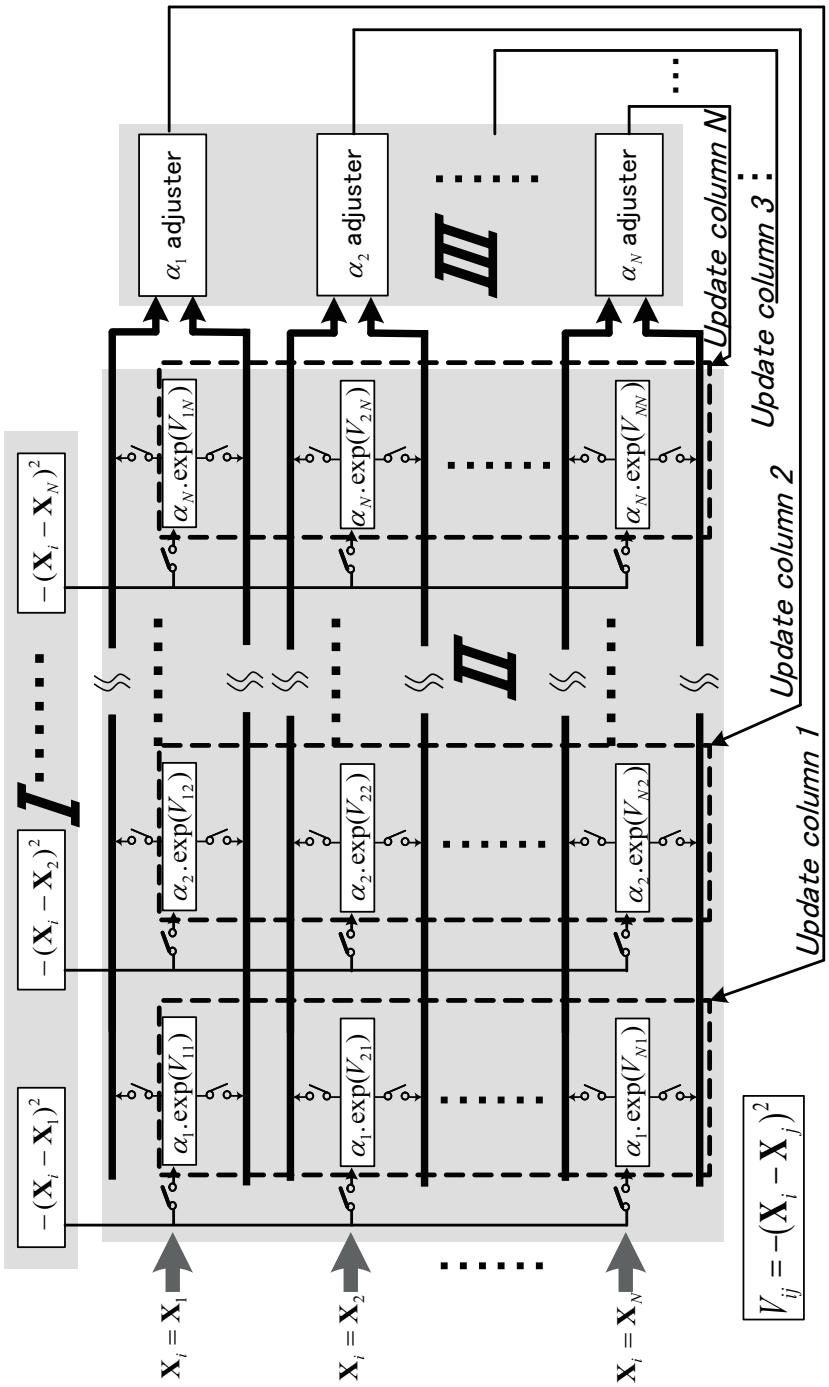


Figure 2.2: Organization of proposed fully parallel learning SVM processor composed of Euclidean distance calculator (I), exponential circuit array (II), and α adjuster (III).

To make it flexible and possible to combine with digital circuits, even a computer program, we also built an on-chip digital-to-analog converter (DAC) for the system. The 64D pattern vectors are converted into analog voltage signals. For instance, the vector $\mathbb{X}_i$ can be represented by voltages as $\{v_{i1}, v_{i2}, \cdots, v_{i64}\}$. The part of the 64D squaring circuit illustrated in Fig. 2.3 is built to calculate $D(\mathbb{X}_i, \mathbb{X}_j)$, where

$$D(\mathbb{X}_i, \mathbb{X}_j) = |\mathbb{X}_i - \mathbb{X}_j|^2 = \sum_{k=1}^{64} (x_{ik} - x_{jk})^2. \tag{2.6}$$

By assuming $v_{i1} > v_{j1}$, the potentials v_1 and v_2 of nodes 1 and 2 are shifted by a sufficiently small current bias I_b as

$$\begin{cases} v_1 \approx v_{i1} + V_{thn} + |V_{thp}| \\ v_2 \approx v_{j1} + V_{thn} + |V_{thp}| \end{cases}, \tag{2.7}$$

where V_{thn} and V_{thp} are the threshold voltages of the n and p-type MOS transistors, respectively. By considering $v_{i1} > v_{j1}$, the transistors in the branch where the current I_1 flows are in the strong inversion region, the transistors in the branch where the current I_2 flows are in the weak inversion region. Thus, $I_1 \gg I_2$ and the current I_1 can be obtained according to

$$I_1 = K_n(v_1 - v_3 - V_{thn})^2 = K_p(v_3 - v_{j1} - |V_{thp}|)^2, \tag{2.8}$$

where K_n and K_p are the amplifier factors of the n and p-type MOS transistors, respectively. By solving this equation with the consideration of eq. 2.7, the potential of node 3, v_3, can be expressed as

$$v_3 = \frac{v_{i1}\sqrt{\frac{K_n}{K_p}} + v_{j1} - |V_{thp}|\sqrt{\frac{K_n}{K_p}} + |V_{thp}|}{1 + \sqrt{\frac{K_n}{K_p}}}. \tag{2.9}$$

Substituting eq. 2.9 into eq. 2.8, the current flowing through this branch can be given by

$$I_1 = \beta(v_{i1} - v_{j1})^2, \tag{2.10}$$

where

$$\beta = \frac{K_n K_p}{(\sqrt{K_p} + \sqrt{K_n})^2}. \tag{2.11}$$

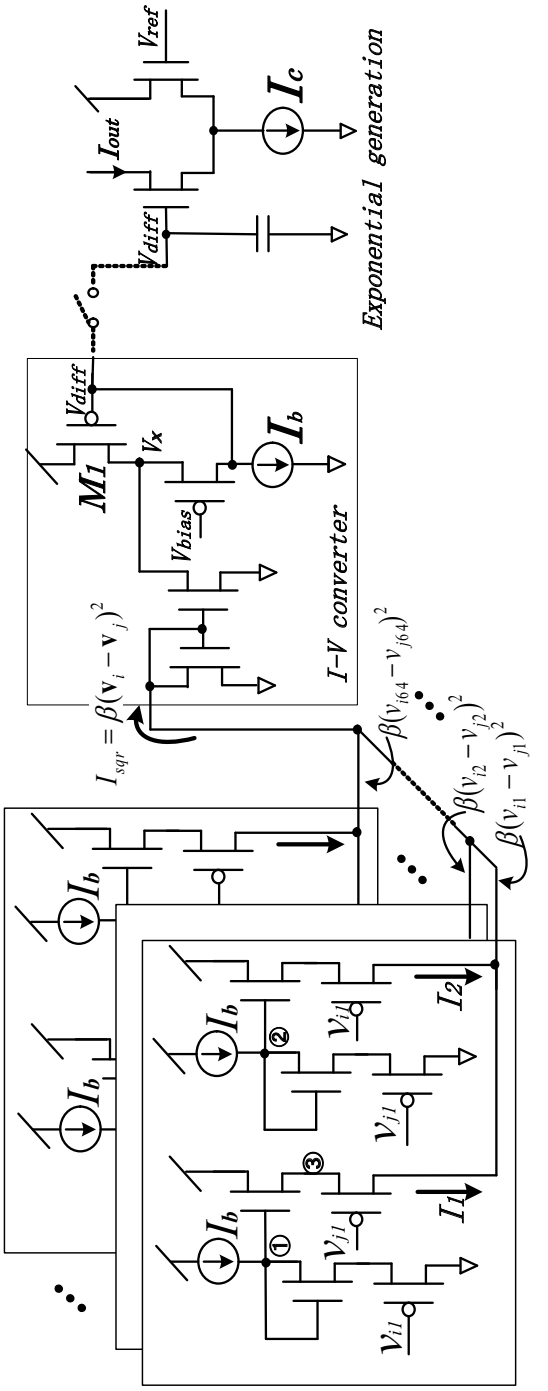


Figure 2.3: Schematic of proposed Gaussian generation circuit.

Since $I_1 \gg I_2$, the total current generated by the 1D squaring circuit is almost equal to I_1 ($I_1 + I_2 \approx I_1$). In the case of $v_{i1} < v_{j1}$, a similar derivation can be applied to obtain

$$I_1 + I_2 \approx I_2 = \beta(v_{j1} - v_{i1})^2. \tag{2.12}$$

Generally, the current generated by the 1-D squaring circuit is $\beta|v_{i1} - vj1|^2$. By collecting the current generated by all dimensions, the output for evaluating $D(\mathbb{X}_i, \mathbb{X}_j)$ can be obtained as

$$I_{sqr} = \beta \sum_{k=1}^{64} |v_{ik} - v_{jk}|^2. \tag{2.13}$$

Compared with traditional analog Gaussian generation circuits (38, 39, 40, 41) (in which the functional transistors operate in the subthreshold region), all the functional transistors in this part operate in the saturation region. Thus, the proposed circuit realizes a large input range and high reliability against process variations.

The current I_{sqr} is copied to the current-to-voltage (I-V) converter, which is biased by a sufficiently small current I_b. With this condition, V_x can be approximated as

$$V_x = V_{bias} + |V_{thp}|. \tag{2.14}$$

Obviously, the gate voltage of M_1 must be lower than its drain voltage. Thus, M_1 operates in the linear region. Assuming its drain-source voltage as V_{ds} (V_{ds}=V_{dd} − V_x), the current flowing across M_1 can be given as

$$I_{in} \approx K_p[(V_{dd} - V_{diff} - |V_{thp}|)V_{ds} - \frac{V_{ds}^2}{2}]. \tag{2.15}$$

By considering eqs. (13) and (14), V_{diff} is calculated as

$$V_{diff} = V_0 - \frac{I_{sqr}}{K_p(V_{dd} - V_{bias} - |V_{thp}|)} = V_0 - \gamma \sum_{k=1}^{64} |v_{ik} - v_{jk}|^2, \tag{2.16}$$

where

$$V_0 = \frac{V_{dd} + V_{bias} - |V_{thp}|}{2}, \tag{2.17}$$

$$\gamma = \frac{K_n}{(V_{dd} - V_{bias} - |V_{thp}|)(\sqrt{K_p} + \sqrt{K_n})^2}. \tag{2.18}$$

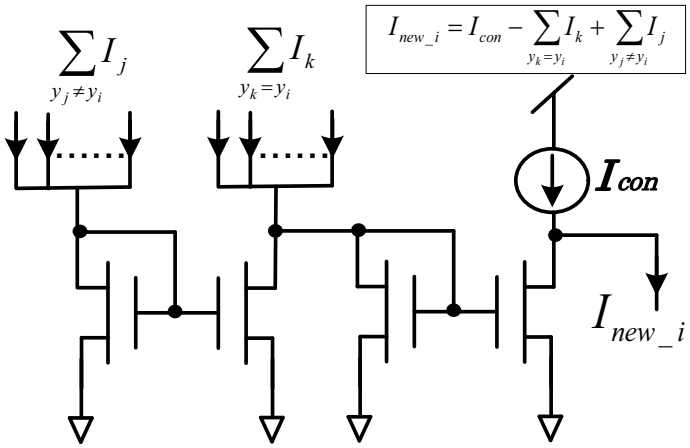


Figure 2.4: Schematic of current mirror-based adder/subtracter.

At the exponential function generating stage, V_{ref} is biased to compensate the effect of V_0 (which can be easily generated by another exactly the same I-V converter without any input current). Thus, the output current I_{out} is given as

$$I_{out} = \frac{I_c}{1 + e^{\Delta V}} \approx \frac{I_c}{2} e^{-\Delta V}, \tag{2.19}$$

where ΔV is the voltage difference between two input terminals of differential pairs. Finally, a wide-input-range Gaussian function is obtained as

$$I_{out} = \frac{I_c}{2} e^{-\gamma |\mathbb{X}_i - \mathbb{X}_j|^2}. \tag{2.20}$$

The peak-height value of the obtained Gaussian function feature can be scaled by I_c, which reflects the α values in the SVM algorithm. The width of the Gaussian function feature is scaled by the voltage bias V_{bias}. The width programmability does not contribute to the SVM learning process theoretically. However, from the viewpoint of hardware, it makes the system practical when various databases (even various numbers of dimensions) are used. To realize a high hardware parallelism, exponential circuits with a number of N^2 should be duplicated for N sample learning. Fortunately, the chip area occupied by this part is much smaller than that of Euclidean distance calculating circuits.

The current mirror-based adder/subtracter is illustrated in Fig. 2.4. By collecting the current from all the kernels (including those of the same class and

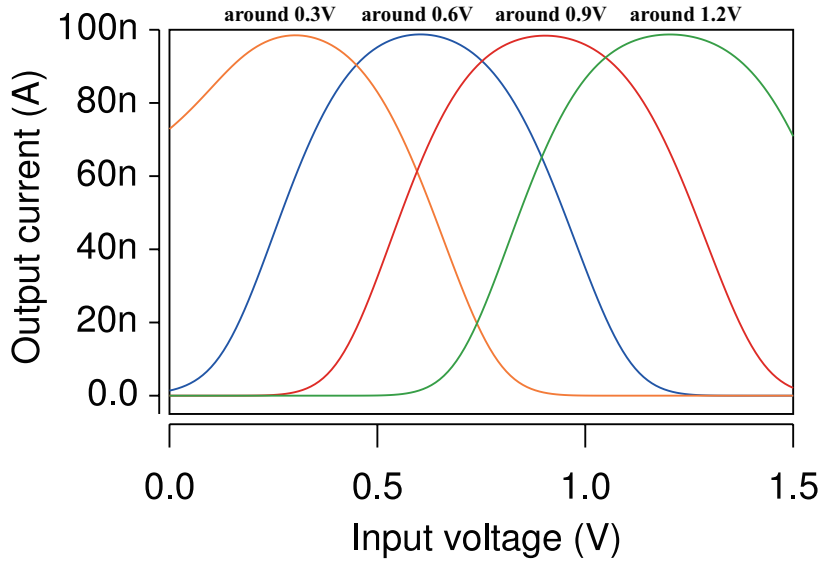


Figure 2.5: Center programmability of Gaussian function feature.

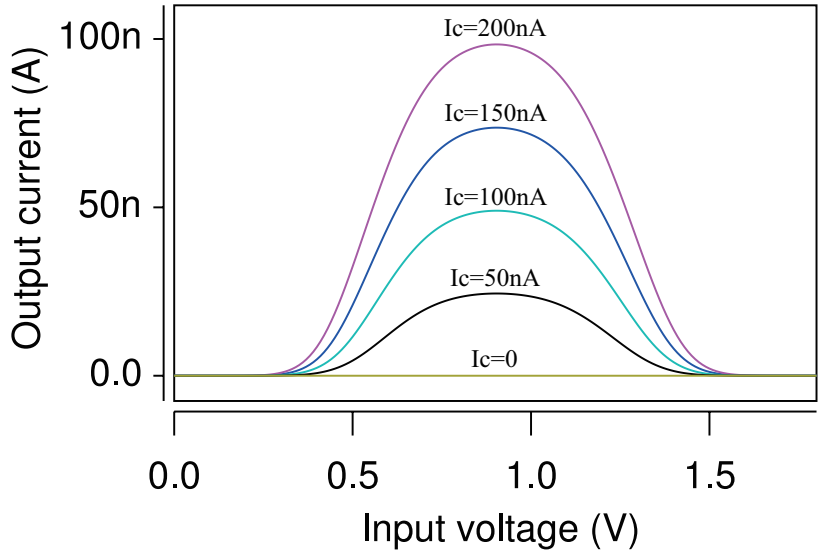


Figure 2.6: Peak-height programmability of Gaussian function feature.

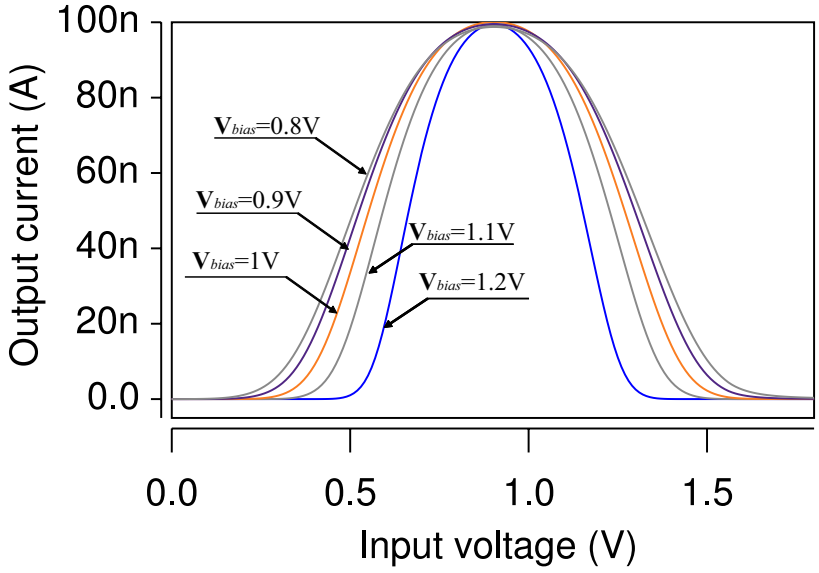


Figure 2.7: Width programmability of Gaussian function feature.

the different class), the current I_{new_i} is obtained as a new alpha value for the $i-th$ kernel:

$$I_{new\_i} = \min(I_{con}, \max(0, I_{con} - y_i \sum_{m(\neq i)} y_m I_m)), \tag{2.21}$$

where I_{con} represents the regularization parameter C in eq. 2.5.

2.4 Experimental Verification

2.4.1 Performance characteristics of Gaussian generation circuit

The programmability of our proposed Gaussian circuit is verified by HSPICE simulations employing a 0.18 μm 1.8 V CMOS process. We selected one element from 64 dimensions randomly and kept the inputs of the remaining dimensions constant at the same value. The center programmability of the Gaussian function feature is verified from the simulation results shown in Fig. 2.5. Within an input voltage range of 0 to 1.5 V, the proposed circuit generates Gaussian functions

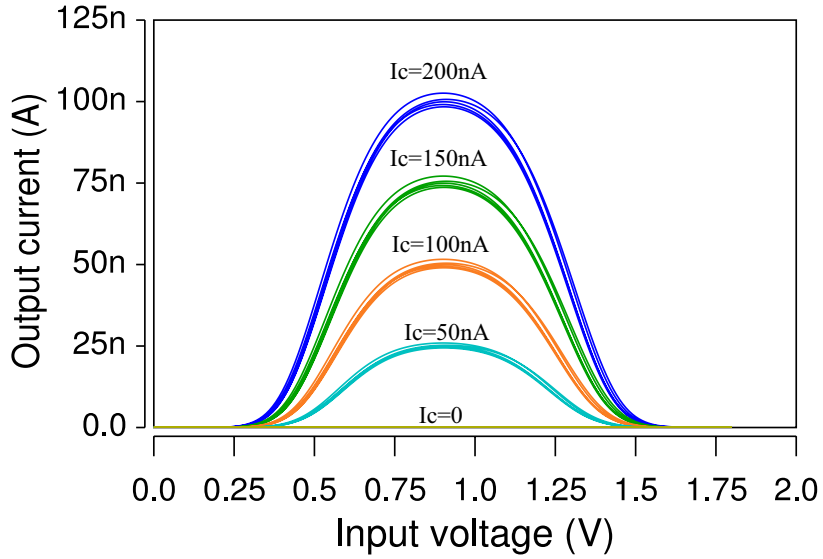


Figure 2.8: Robustness against the process variations of the proposed Gaussian generation circuit.

around various center values directly. Figure 2.6 reflects the peak-height programmability of the obtained Gaussian function feature. The peak-height value can be scaled linearly to the current I_c, which is important in the SVM learning process. As mentioned above, the width programmability, which is shown in Fig. 2.7, does not contribute to the SVM learning process theoretically, but improves the hardware practicability owing to the sensitivity of analog circuits.

The performance robustness against fabrication process variations is shown in Fig. 2.8. The Monte Carlo simulation with five trials is performed to verify the mismatch problem. We randomly select one dimension for DC sweeping and keep the other 63 dimensions constant at the same value. We only focus on the threshold voltage for this simulation since it is the most sensitive factor to process variations. It is assumed that the variations follow a Gaussian distribution, and a sufficiently large variation of 5% is considered (43). According to the simulation results, the maximum performance fluctuation of the proposed 64D Gaussian generation circuit is about 4% when process variations are considered.

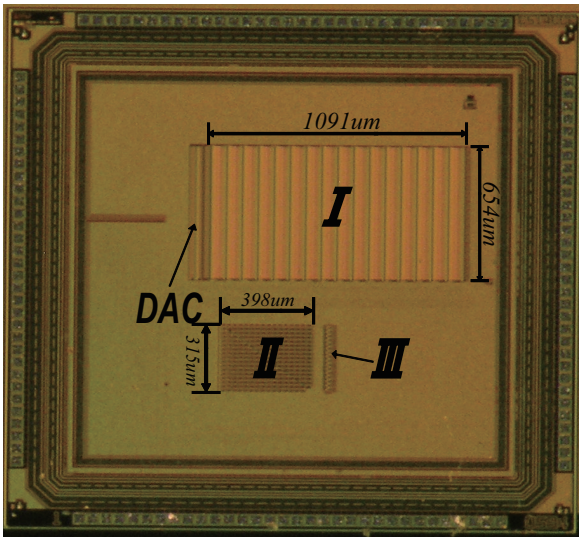


Figure 2.9: Micrograph of proof-of-concept chip for 64D SVM learning/classifying processor.

2.4.2 64D pattern vector classification

The proof-of-concept chip of the proposed fully parallel learning SVM processor was built in a 0.18 μm CMOS process. This VLSI chip has a learning capacity of 16 vectors with 64 dimensions. Its micrograph is shown in Fig. 2.9. Block II occupies a smaller chip area than block I even if it contains many more cells. In other words, the problem regarding the chip area and inner connection explosion was prevented in this experiment.

To verify the learning and classifying performance characteristics of the proposed SVM processor, the object images selected from a real database COIL-20 (Columbia Object Image Library) (44) were introduced as learning samples and test patterns. All the images were preprocessed and extracted into 64D vectors before the verification. Eight images from the same class (named "a" in this work) and an eight other images from another class (named "b") were randomly selected as learning samples. A clock signal with a frequency of 10 MHz was used for the DAC. After inputting the learning samples, the learning process autonomously proceeds and the α values self-converge without any additional clock-based control.

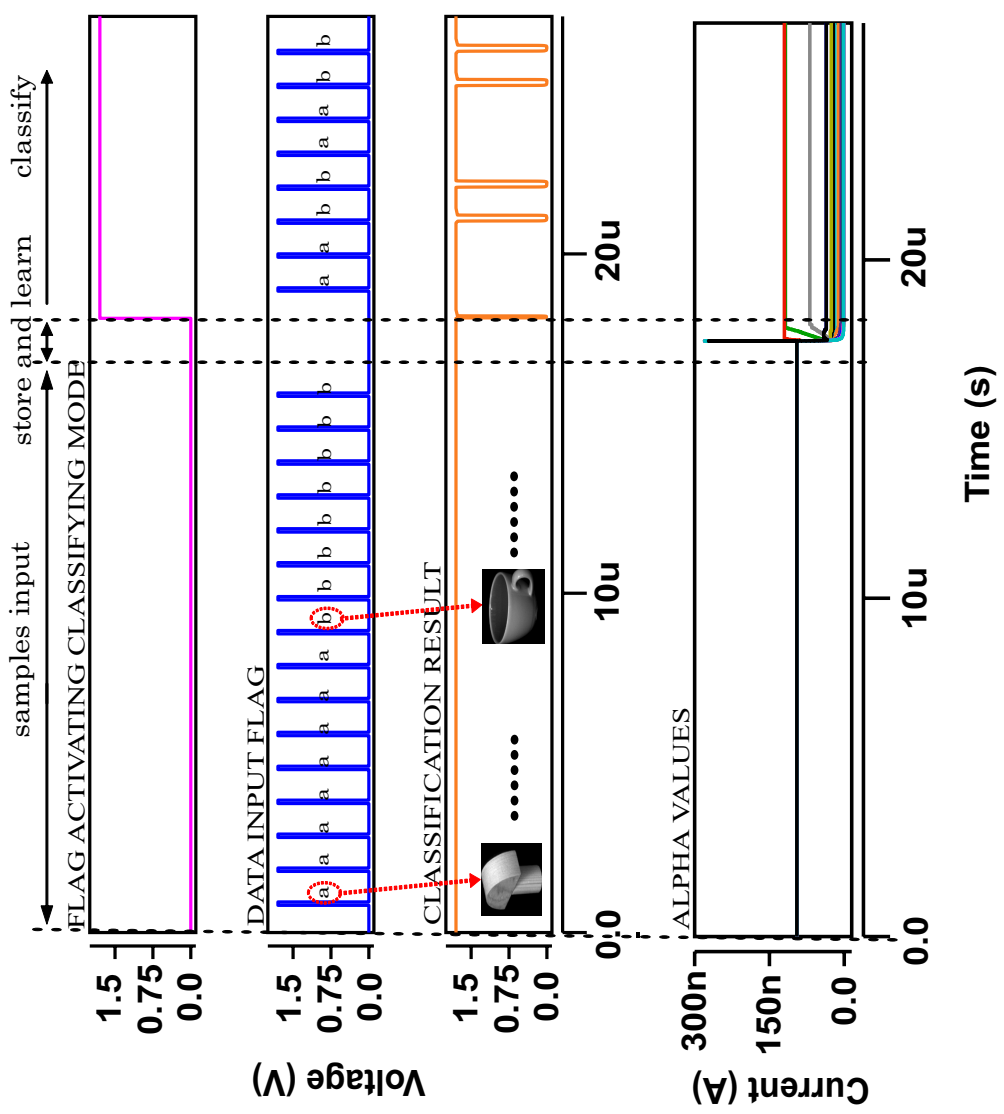


Figure 2.10: Simulation results of learning and classifying processes.

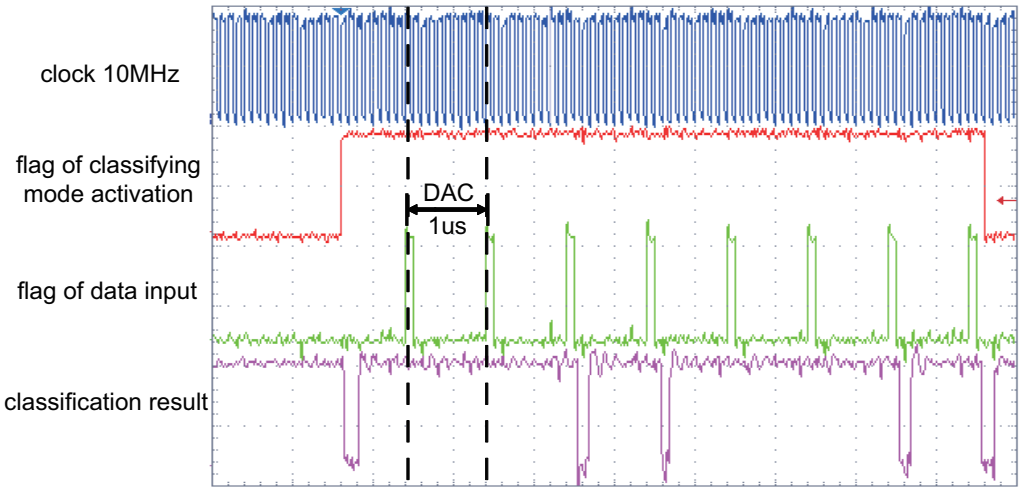


Figure 2.11: Measurement results for 64D pattern vector classification.

The Nanosim simulation results shown in Fig. 2.10 reflect the learning and classifying performance characteristics. Upon receiving test vectors, the SVM processor is activated in the classification mode by a flag signal. According to the simulation results, eight test vectors are all classified into correct classes, which is reflected by a voltage signal (high voltage represents the label of class a, and low voltage represents the label of class b). The performance is confirmed from the real measurement results shown in Fig. 2.11.

2.4.3 2D pattern vector classification

To verify the flexibility of the SVM processor when it deals with various types of databases, a set of 2D pattern vectors is introduced into the same processer. We adjusted the width of the Gaussian kernel function feature for this application. The clock frequency is set as 2.5 MHz and the input data for the remaining 62 dimensions is set as 0. Therefore, the pattern $\mathbb{X}$ is transformed into the form of $\mathbb{X} = \{x_1, x_2\}$. The differences between 2D patterns are obviously much smaller than those of the 64D application. Owing to this result, a lower system clock frequency is applied for 2D experiments. Figure 2.12(a) shows the Nanosim simulation results of this experiment. The pattern $\{x_1, x_2\}$ is formalized into the analog form $\{V_{DAC}x_1/255, V_{DAC}x_2/255\}$ by the on-chip DAC, where V_{DAC} is the

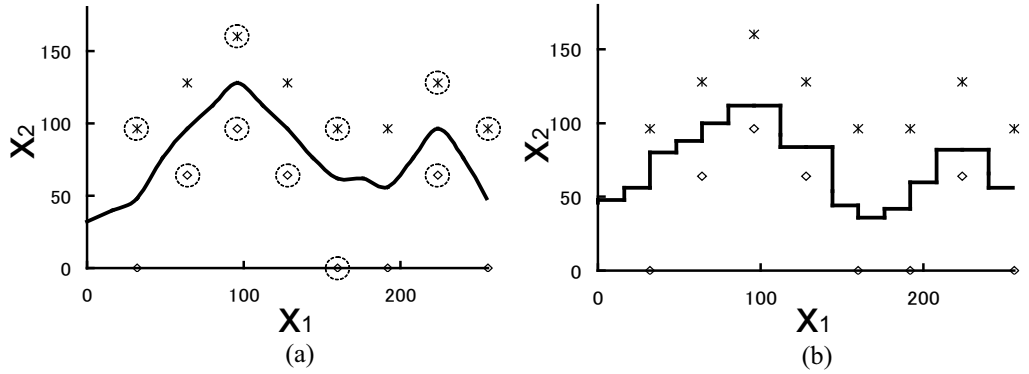


Figure 2.12: Performance characteristics of 2D pattern vector classification: (a) simulation and (b) measurement results.

reference voltage of DAC. The points marked by dotted circles are identified as support vectors. A classification boundary is obtained by a set of test vectors with a resolution of 32×32. The chip measurement results are shown in Fig. 2.12(b). The classification boundary is obtained from a set of test vectors with a resolution of 16×16. When the test vectors are located near the expected boundary, the result output signal becomes poor and unstable. This is the reason why the boundary obtained by measurement is rough. In fact, the classification performance of the 2D application is poorer than that of the 64D application. However, the proposed SVM processor has a potential to extend to various types of pattern, even if the number of pattern vector dimensions is as low as 2.

Table 2.1: Performance comparisons. N represents the number of vectors and S is the chip area occupied by one kernel for a single vector. l is the number of iterations for convergence.

	Ref. (40)	Ref. (41)	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
Chip area	$(N^2+N)\times S$	$N\times S$	$1.17N\times S$
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	$12\times l\times 60$	100
Classifing speed (vectors/s)	N/A	8.7×10^5	10^6

2.4.4 Comparisons

Performance comparisons are shown in Table 2.1. The dimensions of previously developed Gaussian kernel SVM processors (40, 41) are difficult to expand owing to the limitation of the conventional Gaussian generation circuits. Obviously, a conventional fully parallel strategy could achieve a high learning speed but requires a very large number of kernel generation circuits, which is unacceptable for highly dimensional applications. By using the proposed architecture, a high learning speed can be realized within a chip area similar to that of the row-parallel strategy (41), which is much smaller than that of the conventional fully parallel strategy (40). In addition, the number of learning iterations strongly depends on the learning samples, which is usually large and unpredictable. Therefore, a non-clock-based iterative approach is helpful for some real-time learning applications.

2.5 Summary

A fully parallel self-learning SVM processor was developed for the classification of highly dimensional vectors. To implement the kernel function of SVM, an analog highly dimensional Gaussian generation circuit was designed with function feature programmability, which is robust against the process variations. On the basis of this circuit, a fully parallel array of kernels was implemented in a compact chip area by using the proposed architecture. Therefore, the learning process proceeded autonomously without clock-based iterations and converged with a high speed. A proof-of-concept chip was fabricated in a 0.18 μm CMOS technology and measured by employing real images. According to the measurement results, the proposed SVM processor classified all the test images into correct classes.

2. FULLY PARALLEL SUPPORT VECTOR MACHINE PROCESSOR EMPLOYING ANALOG CIRCUITRY

3

Fully Parallel K-Quasi-Centers Clustering Processor Employing Analog Circuitry

As it was introduced in the previous chapters, the proposed fully parallel architecture can be also applied to implement the unsupervised machine learning algorithms such as pattern clustering. A fast self-converging analog learning processor is presented in this chapter for use in image clustering. The image patterns represented by high-dimensional vectors can be clustered into different classes based on our proposed hardware-friendly version of the K-means algorithm, which we called K-Quasi-Center (KQC for short). In order to speed up the KQC learning, we developed an analog circuit to carry out the Euclidean distance between high-dimensional vectors in real-time. Furthermore, the fully-parallel learning and self-converging structure was built employing these analog circuits. The chip-area and interconnect explosion problem has been resolved in this proposed structure. Since the learning autonomously proceeds in a fully-parallel manner via analog free-feedback signals without any clock control, learning can be accomplished within a single clock cycle, which is far faster than the conventional iteration-based approaches. A proof-of-concept system was constructed and verified by the HSPICE and Nanosim simulations. Sixteen actual images of two objects were randomly selected from the database and converted into 64-dimensional vectors. Feeding these vectors into our KQC learning pro-

cessor, the images were all correctly categorized into two classes within one clock cycle for several different initialization conditions.

3.1 Introduction to the Proposed KQC Clustering Processor

The K-means algorithm is one of the most typical unsupervised machine learning algorithms for pattern classification (45), which was originally developed in the computing science area. Unsupervised learning algorithms have been successfully implemented for image processing problems such as the image segmentations (46) and image-sets categorizations (47) using software programs. Like most of other machine learning algorithms, a large amount of numerical computing iterations are always required in K-means clustering problems. Then, this learning process could be very computationally expensive. Consequently, VLSI implementations of the K-means algorithm (48, 49) were considered for the applications of portable electronic equipments or those requiring high speed performances.

Several works employing digital VLSI circuits realized image segmentation based on the K-means algorithm (50, 51). In these approaches, the Manhattan distance was employed as distance measure to avoid the complexity of the Euclidean distance calculation. Considering high-level applications of image processing (47), the image-sets categorization for instance, the number of dimensions of sample vectors becomes much larger since the feature vector describes and represents a lot of important features in the entire image. Due to this reason, the distance calculation becomes much more computationally expensive. Even if the powerful hardware as GPU is applied as in the work of (52), it takes much time to calculate the distances between vectors having a large dimension.

One of possible solution to speed up the learning processes is applying analog circuits to carry out the core functions. The accuracy of analog processing is not as high as that of digital processors, but it is sufficient for carrying out learning iterations. Therefore, there have been several attempts realizing the supervised machine learning algorithms such as the Support Vector Machine classification (41) using the analog circuits. Since the analog circuits take up much smaller

chip area than digital ones, it is possible to integrate multiple analog processing cores on-chip and accomplish all the high-dimensional calculations in parallel for single iteration. However, this approach employing analog processors still needs clock-based iterations during the learning process, which means long time for convergence and complex controlling mechanism are required.

Generally, there is a trade-off between the amount of circuits and the processing speed in the hardware implementations of both digital and analog K-means learning systems. The higher processing parallelism realizes the higher speed, but requires the larger number of circuits on the other hand. The number of learning iterations, which is usually very large and not dependent on the hardware parallelism, has a noticeable influence on the learning speed. Therefore, conventional VLSI implementations employing clock-based iterations consume a lot of time on these iterations indifferently to the degree of hardware parallelism.

The purpose of this work is to build a fast on-chip learning processor for high-dimensional patterns categorization, in which the calculations and learning process are both accomplished in a fully-parallel architecture of analog circuits. A modified scheme of K-means algorithm, named KQC method, is implemented by this learning circuitry, which is essentially similar to the original K-means mechanism but more efficient for the fully parallel implementation. Using the analog free-feedbacks instead of clock-based iterations, the learning process can be accomplished autonomously within only one clock cycle and self-converges without any clock-based controlling, which is far faster than the conventional iteration-based approaches. The proof-of-concept system was built for 64-dimensional vectors categorization. In order to verify the performances of our proposed system, sixteen images of two kinds of objects selected from the COIL-20 database (Columbia-Object-Images-Library) (44) were converted into feature vectors and fed into our KQC learning processor. According to the Nanosim simulation results, all the images were correctly categorized into their respective classes even with several different randomly ill initializations.

The rest parts of this chapter are organized as follows: the modified version of K-means algorithm will be briefly introduced in the section 3.2; in the section 3.3, the system organization and its operational principle will be presented; section 3.4 shows the HSPICE and Nanosim simulation results for the performances of

our proposed analog processors and the whole system respectively; summary will be made in the section 5.

3.2 Algorithm Applied in This Work

The K-means algorithm is widely used in pattern recognition, data mining, and image segmentation as a very powerful learning tool. It is a typical unsupervised clustering method. Using this algorithm, learning samples given in the form of multidimensional vectors can be partitioned into a limited number (K) of clusters according to their feature similarity without supervision. In this work, the original K-means algorithm is adjusted to be a hardware-friendly version for the high-dimensional vector categorization. The purpose is to prevent duplicate of complexly computational processors in order to realize the fully-parallel learning and self-converging mechanism.

3.2.1 Basic idea of the original K-means algorithm

The basic idea of original K-means algorithm is illustrated in Fig. 3.1(a). A large number of sample vectors $\mathbb{X} = \{x_1, x_2, \ldots, x_n\}$ with n dimensions are given. Considering the K value as 2, the target is to cluster the sample vectors into two categories. The unsupervised learning process is described as follows:

1. Initializing the categorization randomly and computing the centroid vector for each cluster;
2. evaluating the distances between a specific vector and all centroid vectors;
3. resetting the categorization label for this specific vector according to the distance comparisons;
4. repeating 2) and 3) till all the given vectors have been re-clustered for this round;
5. computing the new centroid vectors and back to 2);
6. results converging and stopping learning.

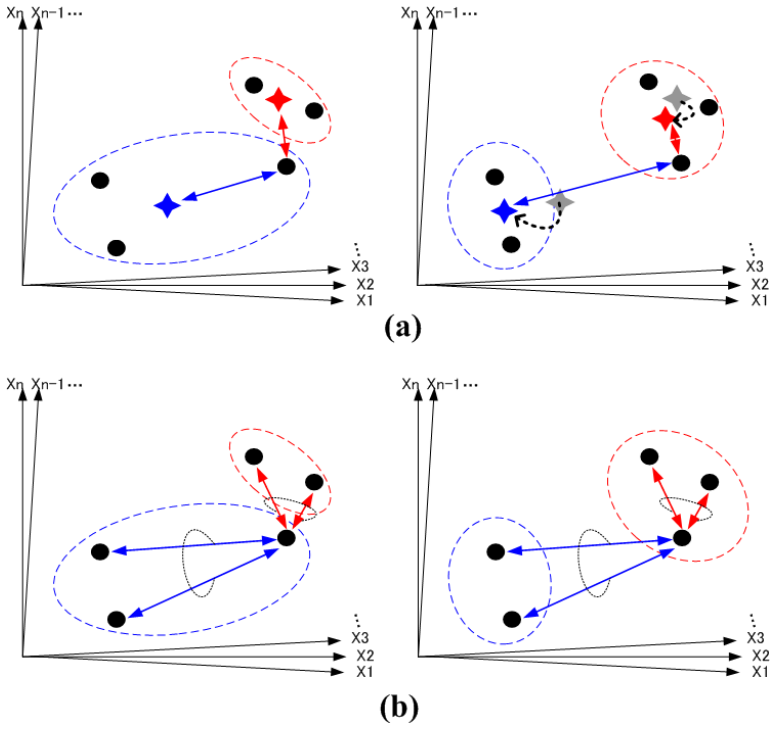


Figure 3.1: Basic idea of the K-means clustering: (a) original K-means clustering and (b) the modified version applied in this work.

Many different types of mathematical representations were introduced to evaluate the distance $D(\mathbb{X}_i, \mathbb{X}_j)$ between vectors $\mathbb{X}_i$ and $\mathbb{X}_j$ such as Euclidean Distance. In some previously proposed VLSI implementations of K-means, the Manhattan Distance was employed since it is computationally simpler than the Euclidean Distance. However, the complex computations for high-dimensional vectors still should be repeated for heaps of times, including average-vector calculations (to search the centroid) and the distance evaluations. These calculations for vectors are usually hungry for the hardware and computing-time especially when the number of vector dimensions is large. Due to this reason, the original K-means algorithm can be hardly implemented by VLSI circuits with fully-parallel learning strategy.

3.2.2 KQC method for highly dimensional pattern clustering

In order to avoid iterating calculations of vectors, a modified version of K-means algorithm was proposed and named K-Quasi-Centers method by this work. The basic idea of our proposed method is illustrated by Fig. 3.1(b). Instead of iterating operations to search centroid vectors and calculate distances between vectors, all the vector calculations are carried out at the beginning and kept for the whole learning process. The Euclidean Distances from every vector to all the others are calculated (in parallel by analog VLSI circuits in this work) and stored in the form of scalars, which can be directly reused during the learning process. In this manner, the only operation needed to be iterated is changing the category label for each vector, which is easily realized by analog VLSI circuits and possible to be proceeded in parallel.

1. Calculating the Euclidean Distances from every vector to all the others and initializing the categorization randomly;
2. evaluating the similarities from a specific vector to all the categories (by calculating the average distances from the specific vector to all the vectors from the same category);
3. resetting the categorization label for this specific vector according to the evaluation;
4. repeating 2) and 3) till all the given vectors have been re-clustered for this round;
5. results converging and stopping learning.

During the learning operation, the only calculation is to compute the average values among scalars. Therefore, the chip area and processing time for learning processor can be reduced. In the actual VLSI implementation by this work, the similarity evaluations and the label resetting are both proceeded in vector-parallel rather than one by one based on the clock cycle. The temporal results are immediately feedback to the circuity and autonomously converge.

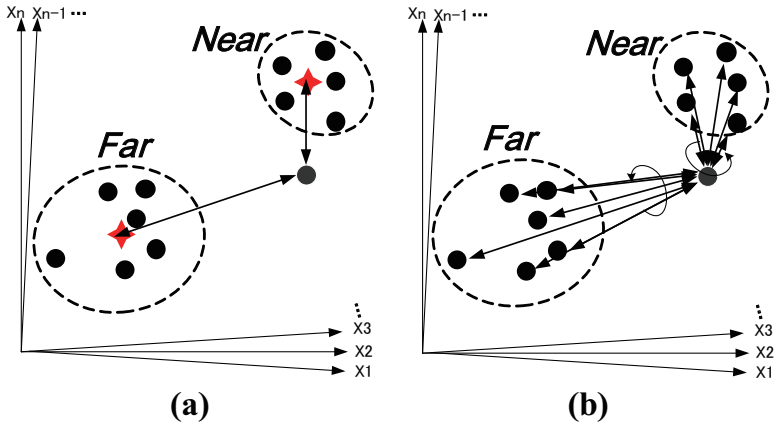


Figure 3.2: Distance evaluations from a specific sample vector to different clusters: (a) using the representatives of centroid vectors and (b) using the average calculations.

3.2.3 Comparison between the original K-means algorithm and proposed KQC method

Regarding the original K-means algorithm, the centroid vector (or called gravity) of each cluster is represented when the clustering scheme is changed by learning. The distances from any specific sample vector to the gravities of all the clusters are calculated. Essentially, the distances from a specific sample to all the clusters are evaluated by this calculation. These distance evaluations are compared, and the smallest one is selected to determine the new cluster label of this sample vector. Namely, the representatives of centroid vectors even the values of distances have no direct effects on the updating, but the comparison among distance evaluations is important. In our proposed KQC method, the distances from a specific sample to different clusters are evaluated by the average calculation of distances from this sample to all the members in a cluster. Comparing Fig. 3.2 (a) with (b), these two types of evaluations give the same comparisons of distances from a specific sample to different clusters in general cases. Therefore, two types of evaluations lead to the same updating operation in those cases. The learning operation will stop and converge till the cluster label for each sample does not change, which is the same as original K-means mechanism. From this point of view, the suggested

scheme of clustering method has similar convergence performance to the original K-means algorithm.

3.3 Hardware Implementation

3.3.1 Architecture of KQC on-chip learning processor

In this implementation, N vectors represented by $\mathbb{X}_i$s with high-dimensions (64-dimensional in this actual work) can be autonomously clustered into two categories after a fast KQC learning process. The $Label_i$ with a binary form (one bit for two-category implementations) is introduced to reflect the category index of the $i-th$ given vector respectively.

The analog KQC learning circuitry is mainly composed of three blocks using analog VLSI circuits as shown in Fig. 3.3. Block I contains N sets of distance calculators, which evaluate the distance between two vectors $\mathbb{X}_i$ and $\mathbb{X}_j$ as following equation:

$$D(\mathbb{X}_i, \mathbb{X}_j) = |\mathbb{X}_i - \mathbb{X}_j|^2 = \sum_{k=1}^{64} (x_{ik} - x_{jk})^2, \tag{3.1}$$

where, $\mathbb{X}$ is a 64-dimensional vector as $\mathbb{X} = \{x_1, x_2, \ldots, x_{64}\}$. The evaluation results are given in current mode and stored in the analog current memory array as block II row by row in time sequence. Each memory cell is connected with two switches (in fact, the number of switches depends on the number of categories for a specific application) controlled by its respective category label. Since the evaluation results are read out in current mode, it is easy to sum them up for each category by a bus wire. The analog dividers in block III are built to compute the average distance from a specific vector to all the others belonging to a same category. As it is shown in Fig.3.3, the N_A and N_B represent the numbers of vectors in category A and category B respectively. According to the distances from a given vector to two categories, the categorization result for this vector in the form of a binary label is output by using a simple analog comparator. All the calculations and operations are carried out using analog circuits. Thus, the operations can be accomplished in real-time and in parallel by compact circuits.

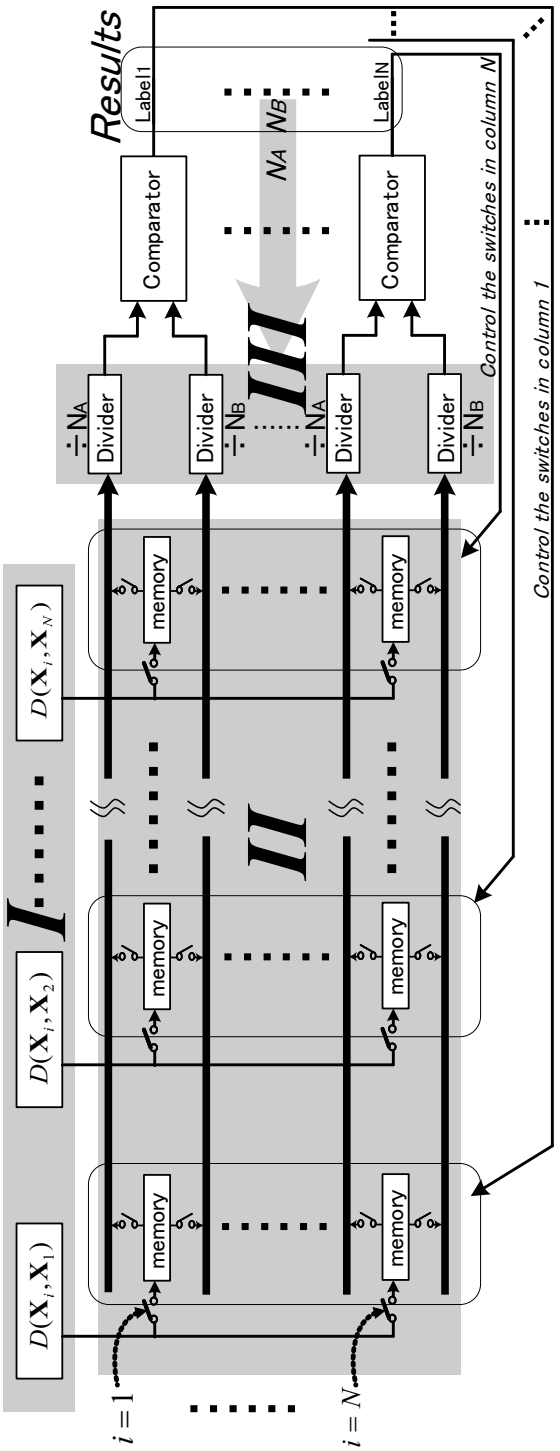


Figure 3.3: Architecture of proposed fully-parallel self-converging KQC learning circuitry with a configuration of 2-category clustering.

Since only the comparisons among distances are concerned rather than the accurate values, the accuracy of analog circuits is acceptable in this implementation.

During the learning process, the categorization results in the form of binary labels are immediately feedback into the circuitry to control the switches in block II and count the number of N_A and N_B in block III. These switching operations are also easy to be realized in real-time by analog VLSI circuits. The dynamics of analog free feedback realizes the mechanism of step iterations, which was mentioned in the algorithm description. Without any additional controlling mechanism, the learning process is accomplished autonomously and self-converges with very high speed (within single clock cycle). The block II contains much more cells compared with other blocks. However, the values stored and processed by block II are all in the form of scalars, which means the chip area of single cell in block II is much smaller than that of block I. Furthermore, it is easy to collect the information by row bus-wires. Therefore, the chip area and interconnect explosion problem can be prevented. Theoretically, a triangular array in block II is sufficient to store necessary values since $D(\mathbb{X}_i, \mathbb{X}_j) = D(\mathbb{X}_j, \mathbb{X}_i)$. However, this distance value contributes to the label updating of two different patterns. The switches and wire connection associated to these two contributions are separate. The elimination of the redundant capacitors does not greatly reduce the chip area of this block, but increase the complexity of inner connection. Therefore, a full array is designed in block II. In the case of category expansion, more bus-wires for categories could be added.

3.3.2 Analog circuitries in the KQC learning processor

Our proposed analog circuit to calculate the distance between two vectors $\mathbb{X}_i$ and $\mathbb{X}_j$ with 64 dimensions is shown by Fig. 3.4. In order to make it flexible and able to combine with digital circuits even computer programs, we also design a Digital-to-Analog Converter (DAC) for the processor. The 64-D pattern vectors are converted into analog voltage signals. For instance, the vector $\mathbb{X}_i$ can be represented by voltages as $\{v_{i1}, v_{i2}, \ldots, v_{i64}\}$. In fact, this circuit is a part of the the Gaussian-generation circuit introduced in the previous chapter. Thus, it is

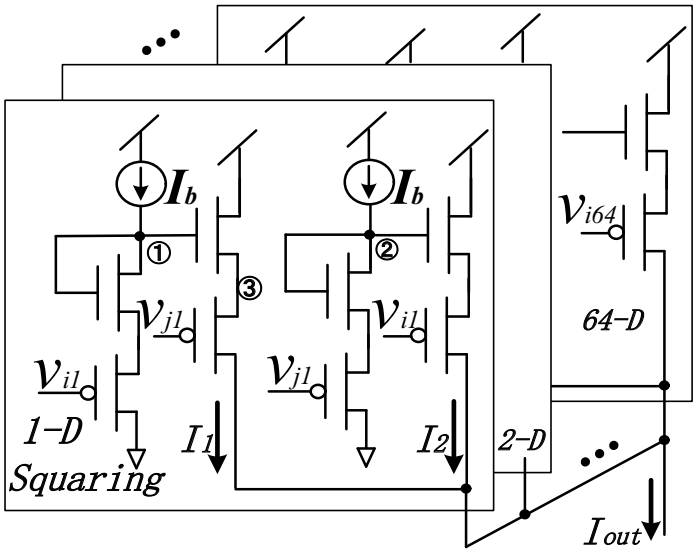


Figure 3.4: Circuit schematic of the distance calculator for 64-D vectors.

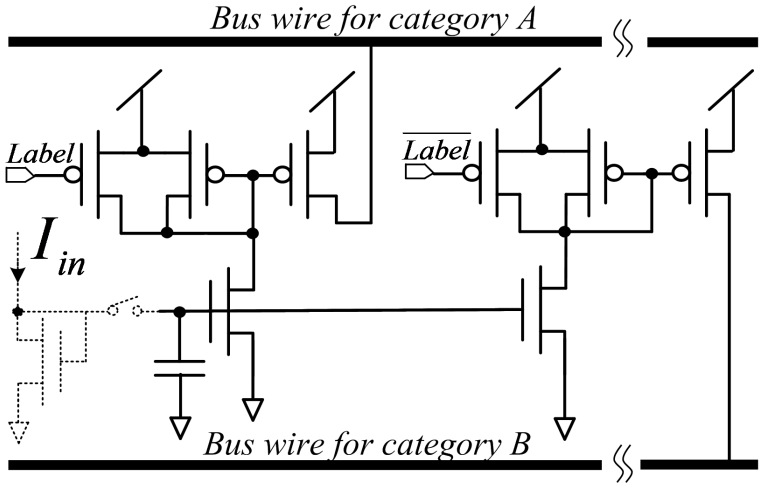


Figure 3.5: Analog current memory cell with switching function.

easily to obtain:

$$I_{out} = \alpha \sum_{k=1}^{64} |v_{ik} - v_{jk}|^2. \tag{3.2}$$

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

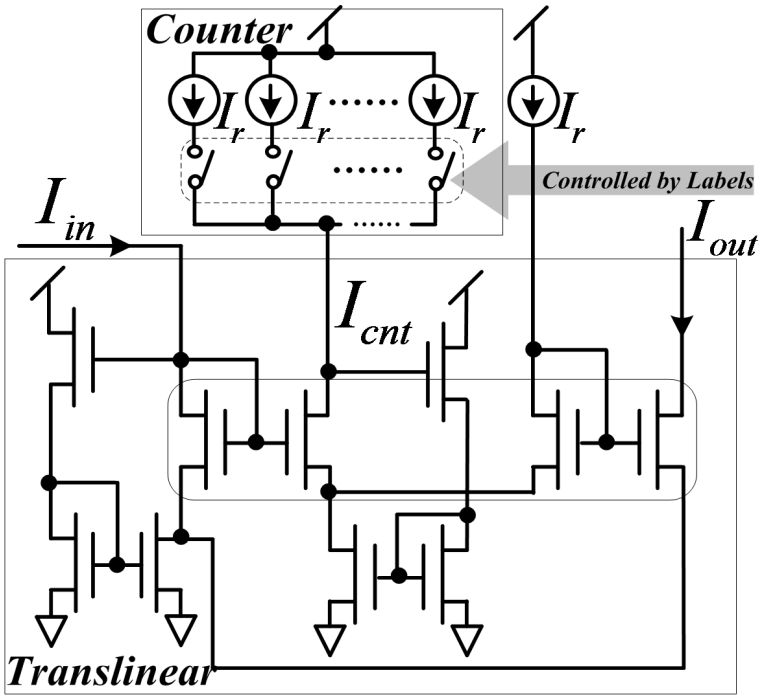


Figure 3.6: An advanced Translinear circuit (54) as an analog divider with the vectors counter.

This current which evaluates the distance $D(\mathbb{X}_i, \mathbb{X}_j)$ is stored in an analog current memory as illustrated in Fig. 3.5. Each memory cell is combined with two analog switches controlled by the categorization label (the number of switches depends on the number of categories). The label of a specific vector determines the current value stored in its respective cell should contribute to category "A" or "B". The bus-wires collect all results in current mode and feed them into the analog dividers to calculate the average distances from this vector to all the others of each category.

An advanced topology of Translinear circuit (54) is applied in this work as an analog divider as shown in Fig. 3.6. According to the Translinear principle, the output current of this circuit is obviously obtained by: $I_{out} = I_{in}\frac{I_r}{I_{cnt}}$, where I_r is a constance current reference and I_{cnt} can be generated by our proposed vector counter. The switches in this counter are controlled by the labels for all the vectors. For instance, the I_{cnt} generated by the counter for category A can

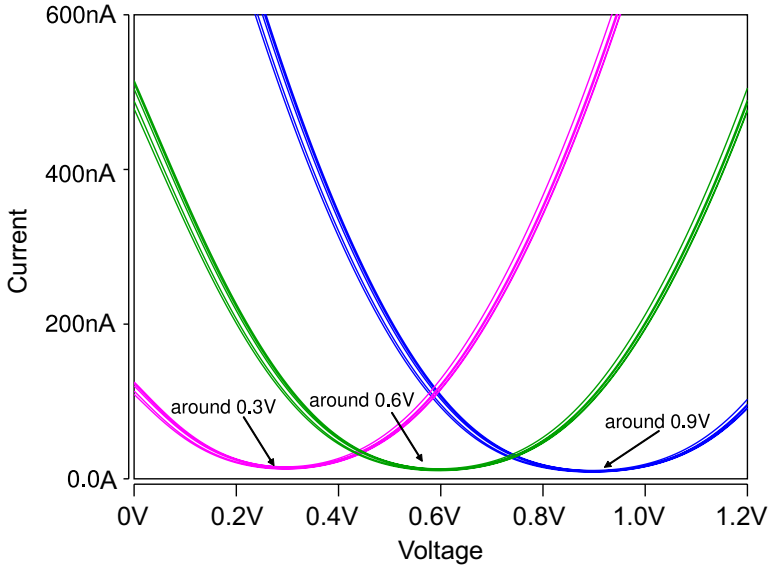


Figure 3.7: Performances of distance calculator (the output current against input voltage with different center values).

be obtained as $I_{cnt} = N_A \cdot I_r$, where N_A is the number of vectors from category A. Finally, the output current is given by: $I_{out} = \frac{I_{in}}{N_A}$ (for the dividers on bus-wire of category A) or $I_{out} = \frac{I_{in}}{N_B}$ (for the dividers on bus-wire of category B). These average values are used to generate new labels by the simple comparators. The new labels are immediately feedback to the operational blocks till results converge.

3.4 Experiments

A $0.18\mu m$ $1.8V$ CMOS technology is employed to construct the proof-of-concept processor for 64-D vectors categorization. The HSPICE and Nanosim simulations were introduced to verify the performances of our proposed analog circuits and the KQC learning processor.

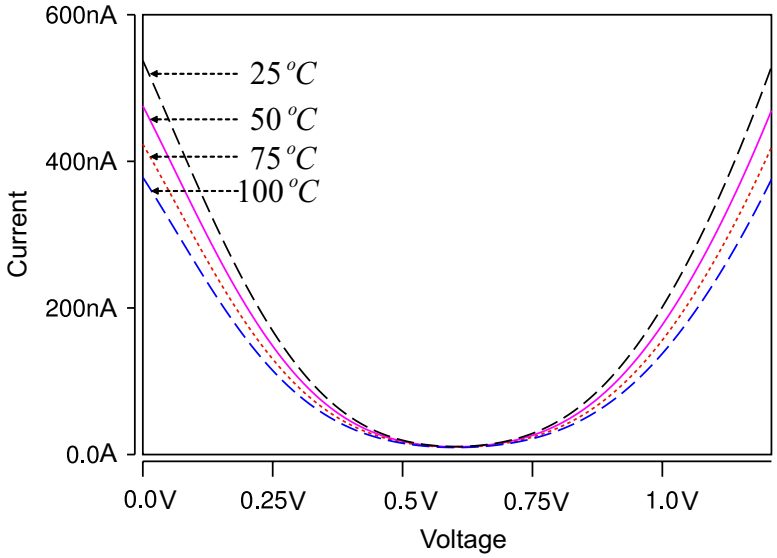


Figure 3.8: Temperature effects on the Euclidean distance calculation.

3.4.1 Performances of employed analog circuitries

The performances of analog circuits applied in our proposed processor are verified by the HSPICE simulations. Considering the distance between vectors $\mathbb{X}_i$ and $\mathbb{X}_j$, the $k-th$ dimension was selected randomly. The voltage v_{jk} was set as a "center" value. For other dimensions, $v_{il} = v_{jl}|_{l \neq k} = 0$. By sweeping the voltage v_{ik}, the output current can be obtained as shown in Fig. 3.7, which reflects the distance generated by this dimension.

The performance robustness against fabrication process variations is also shown in this simulation. The Monte Carlo simulation with five trials is performed to verify the mismatch problem. We focus on the threshold voltage V_{th} for this simulation since it is the most sensitive factor to process variations. Considering the model proposed by the previous works (56), the variation of V_{th} can be described by the following equation:

$$\sigma^2_{\Delta V_{th}} = \sigma^2_{\Delta V_{th}}\,|_{V_{BS}=0}\,(1-\frac{V_{BS}}{\phi_B}) + \frac{2\sqrt{q^3\epsilon_{si}\phi_B N_A}}{3WLC^2_{ox}}(\sqrt{1-\frac{V_{BS}}{\phi_B}} - (1-\frac{V_{BS}}{\phi_B})). \quad (3.3)$$

The variation of V_{th} is related to the mismatch problem during the fabrication process and the dynamic bias V_{BS}. From the results of some related works (56),

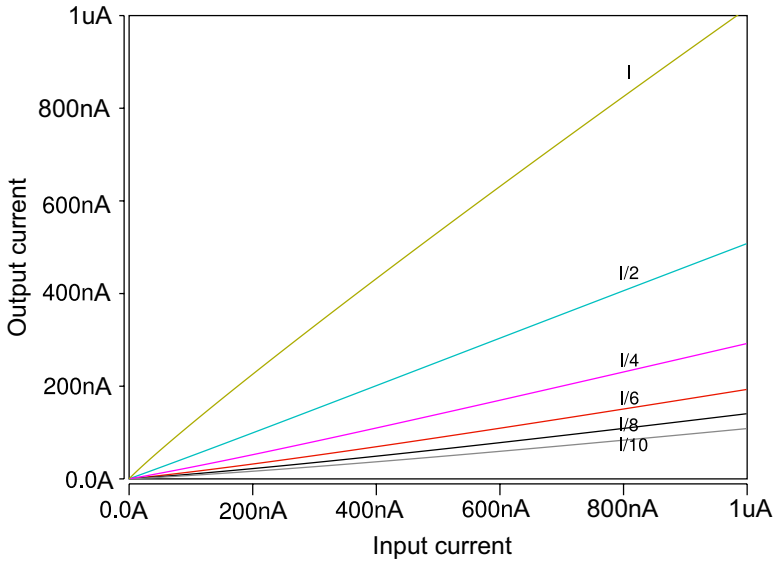


Figure 3.9: Performances of analog divider (the output current against input current).

the worst case of V_{th} variation was evaluated as $\sigma(\Delta V_{th}) \approx 3\% V_{th}$ in a $0.18\mu m$ technology, where V_{th} is the typical value of the threshold voltage. In this simulation, it is assumed that the variations follow a Gaussian distribution. An excessive value of $\sigma(\Delta V_{th})$ is set as 5%. The fluctuation of Euclidean distance calculation is illustrated in Fig. 3.7.

Figure. 3.8 shows the temperature effects on the Euclidean distance calculation. The operational temperatures are set as 25 °C, 50 °C, 75 °C and 100 °C for this simulation. From the simulation results, the temperature effects on the Euclidean distance calculation are approximately symmetrical to the center value. Namely, the temperature has equal effect on the calculation of each dimension for every learning sample, which is negligible to the learning result theoretically.

The simulation results shown in Fig. 3.9 present the performances of the analog divider, which carries out the average current $I_{out} = \frac{I_{in}}{N}$, where N is the amount of vectors. By setting different amounts of $N = 1, 2, 4, 6, 8$ and 10 for instances, the output current with the ratios of $I_{out} = I_{in}, \frac{I_{in}}{2}, \frac{I_{in}}{4}, \frac{I_{in}}{6}, \frac{I_{in}}{8}$ and $\frac{I_{in}}{10}$ can be obtained as presented in Fig. 3.9.

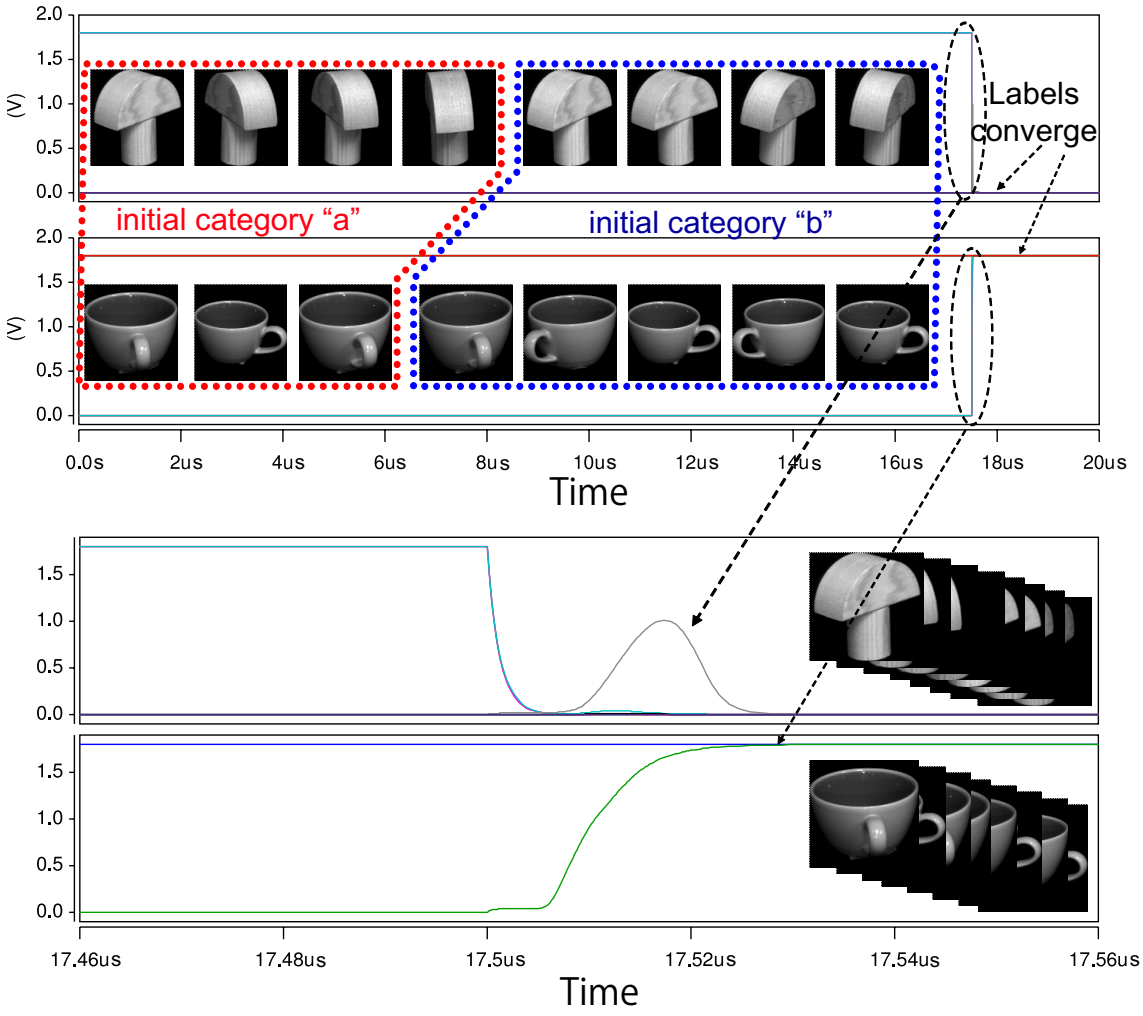


Figure 3.10: Learning performances with an illy random initialization of category labels.

3.4.2 Performances of entire processor

The images selected from an actual database COIL-20 are employed to verify our proposed fully-parallel learning architecture. Before the learning process, all the images are converted into 64-dimensional vectors applying the PPED method. A set of classic serial DACs has also been designed to translate this digital information into analog voltage signals. The Nanosim simulation results are presented by Fig. 3.10 with the illy random initialization for category labels.

The binary signals "1" ($1.8V$ in this work) and "0" ($0V$) are used to reflect the category label for each vector. According to the upper windows in Fig. 3.10, all the images describing the same objects are clustered into the same categories, even if they are ill-initialized. The learning process is autonomously proceeded and self-converges within one clock cycle (0.1 μs as a system clock frequency of 10 MHz). The label signals during the learning process are zoomed in and shown in the lower windows of Fig. 3.10. Without any additional controlling mechanism, they are self-reset for several rounds and completely converges. In the traditional applications based on lock-driven iterations, the learning speed are seriously related to the amount of iterations (55). Assuming the amount of iterations and learning samples are l and N, respectively, the learning process requires a number of clock cycles as $l \times N$ for clock-based works. In other words, the learning process of our proposal is $l \times N$ times faster than the traditional clock-based processors at least.

3.4.3 Discussions

The numbers of learning iterations are usually unpredictable and might be quite different depending on both the database and initialization. This problem could result in the inconvenience especially for the real-time applications. From this point of view, limiting the learning process within one clock cycle is not only helpful to improve the speed performance and reduce the controlling complexity, but also helpful to formalize the learning time. Furthermore, since the learning speed is very high by using our proposed architecture, repeating the whole learning process is acceptable. Therefore, it is possible to apply the on-line learning strategy based on this architecture for the applications with a large amount of learning vectors.

3.5 Summary

An analog VLSI KQC learning processor was proposed by this work employing the fully-parallel self-converging circuitry. An analog circuit has also been

proposed to carry out the distance between two high-dimensional vectors in real-time for this purpose, which is robust against the fabrication process variations. Based on the this modified version of K-means algorithm, the learning process can autonomously proceed and self-converge within one clock cycle without any additional controlling mechanism. Furthermore, the chip-area and interconnect explosion problem can be prevented by using our proposed architecture. The images selected from an actual database were employed to verify the learning performances of the proof-of-concept system. According to the simulation results, all the images were correctly clustered into two categories, even if they were initialized as messed categories randomly. Since the learning speed of the proposed processor is very high, the on-line learning strategy will be considered based on this processor in order to tackle larger amount of learning vectors.

4

On-line Learning Strategy Based on the Fully Parallel Architecture

In the previous chapters, the analog implementations of SVM and KQC algorithms were introduced on the basis of proposed fully parallel architecture. The proof-of-concept processors were designed with sixteen learning samples for both SVM and KQC implementations. Obviously, in the real-world applications, the number of learning samples is usually much larger than sixteen. On the other hand, the processing capacity of the proposed processors is limited by the fabrication technology, and fixed as soon as the chip is built. Thus, the gap between application demands and implicit limitation of hardware should be filled.

Here, an on-line-learning strategy is proposed to fill this gap. The basic idea is to efficiently use the hardware. In order to expend the capacity of processing (the number of learning samples), the learning operation can be repeated for many times by the same VLSI processor. The learning results are harvested in each round of learning. At the same time, the efficiency or importance of each learning sample is evaluated in fully parallel. Some parts of VLSI processors occupied by inefficient samples can be released for accepting new comers on-line. In this manner, the capacity of on-chip learning processors is expended with the consideration of limited hardware resource.

In fact, this on-line-learning strategy is obvious and well-known. However, it is not widely applied in the previously proposed works on the related implementations. There are two reasons that the on-line-learning strategy is particularly

suitable for our work. Firstly, the evaluation of each learning sample's efficiency is expensive especially when a high parallelism is desired. Secondly, the learning operation is time-consuming in the traditional implementations of machine learning. Repeating the learning operations for many times is unacceptable somehow in these works. But in our work, the learning speed is sufficiently high to support the on-line-learning strategy.

To verify the on-line-learning strategy based on the proposed architecture, both SVM and KQC algorithms are implemented in the on-line scheme. The system design is introduced in the following sections.

4.1 On-Line-Learning SVM

Regarding the hardware implementations of SVM, the previously presented digital and analog processors have very limited capacities on the amounts of learning samples due to the limited hardware resource. From the essential point of view, most learning samples (so-called non-support-vectors) are useless for the classification after the learning. Namely, they occupy a large part of hardware but do not contribute to the classification. Since a remarkable part of hardware is inactive, the hardware-efficiency is not high for the traditional VLSI implementations of SVMs.

One reasonable solution to enhance the capacity is applying on-line learning strategy in SVMs, which was originally developed by software programs (57, 58). Several works have implemented the on-line learning SVMs into the real-time applications such as the object tracking problems (59, 60) using software. However, since on-line learning results in a large number of SVM learning operations, the real-time applications usually require the high learning speed, which is difficult to realize using software or traditional VLSI processors. In addition, these on-line learning strategies always increase the number of learning samples. As a result, they are hardly implemented by VLSI circuits with the consideration of limited hardware resource.

The purpose of this work is to propose a hardware-efficient on-line leaning SVM system for the classification of high-dimensional pattern vectors. The SVM on-chip learning processor is used as the core part of this system. Based on this

SVM processor, a hardware-efficient on-line learning strategy has been proposed with limited hardware resource. The performances of the proposed on-line learning system were verified by circuit simulation results. The image patterns from an actual database were used as initial learning samples and test patterns. All the test patterns were classified into correct classes, and the ineffective samples were updated by the test patterns along with on-line learning.

4.1.1 Hardware-efficient on-line learning SVM methodology

In the real-time applications of on-line learning SVMs, the number of learning samples is usually very large and unpredictable. As a result, the traditional on-line incrementally learning strategies (57, 58) can be hardly implemented by VLSI circuits since the hardware resource is strictly limited. Fortunately, in SVM theory some of the learning samples (non-support vectors) are ineffective, which can be removed from sample space. A hardware-efficient on-line learning strategy with constant number of learning samples is proposed in this work. In order to reduce the loss of accuracy, the effectiveness of each sample is evaluated and only the most ineffective sample is replaced by an on-line pattern. In this manner, the learning sample space can be expanded within compact chip area.

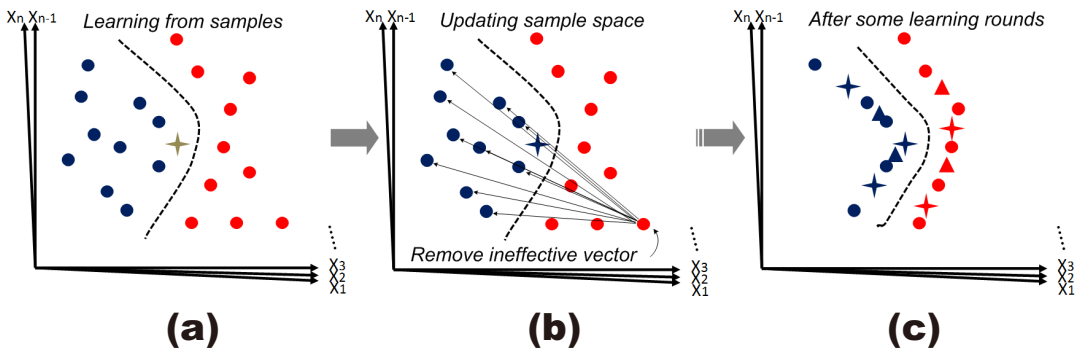


Figure 4.1: Proposed on-line learning strategy: (a) initial learning according to a small set of samples; (b) on-line pattern is classified and the most ineffective pattern is identified; (c) only effective patterns (support vectors) remain after sufficient on-line learning operations.

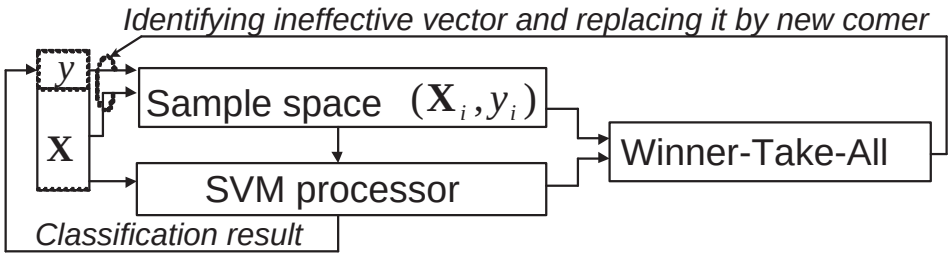


Figure 4.2: Architecture of proposed on-line learning SVM system.

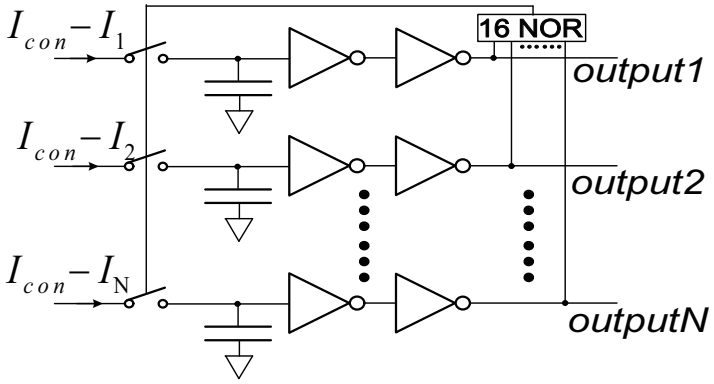


Figure 4.3: Schematic of WTA circuit.

The strategy of proposed on-line learning SVM with a constant number of samples is illustrated in Fig. 4.1. A set of initial learning samples $(\mathbb{X}_i, y_i)_{1\leq i\leq N}$ is needed, where N is the number of learning samples and $y_i \in \{-1, 1\}$ is the class label of the i-th sample $\mathbb{X}_i$. The SVM learning proceeds as soon as one on-line pattern is received. The function used to identify the most ineffective pattern is given by

$$\min_i \sum_{j(y_j \neq y_i)} \alpha_j G(\mathbb{X}_i, \mathbb{X}_j). \tag{4.1}$$

The process of this on-line learning strategy is shown as follows:

1. Initial SVM learning according to a small set of samples as it is shown in Fig. 4.1(a);

2. Classifying the new-received on-line pattern;

3. Evaluating the effectiveness of previous samples and replacing the most inefficient one by new-received pattern as it is shown in Fig. 4.1(b);

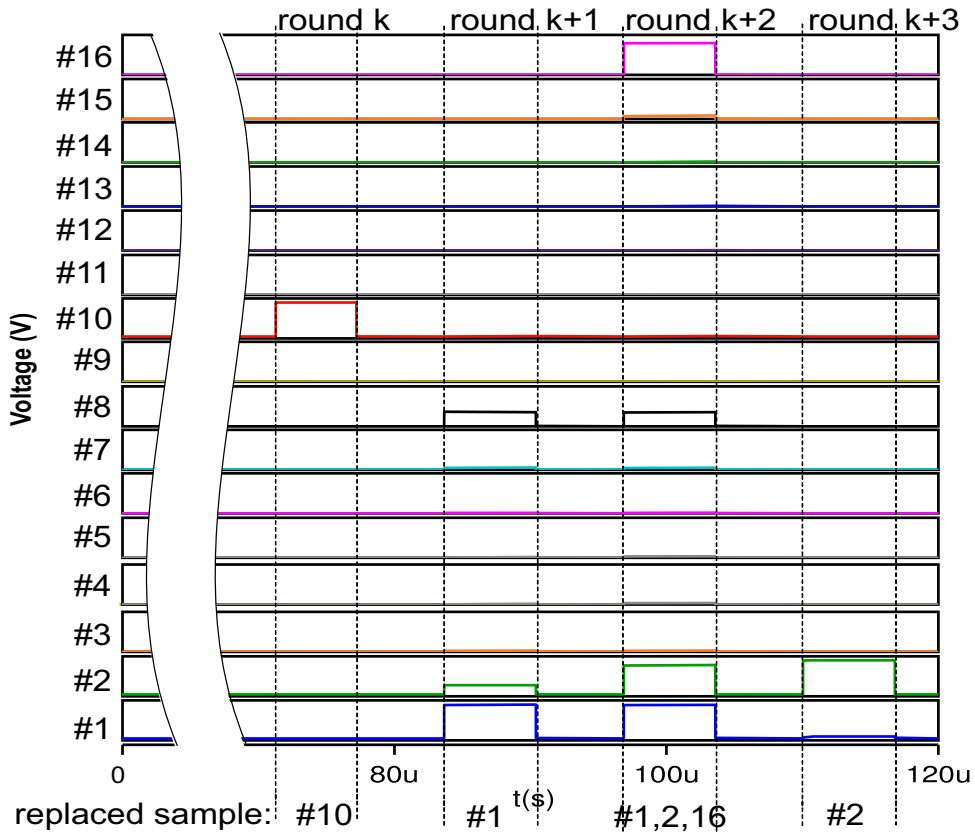


Figure 4.4: Circuit simulation results of the proposed on-line learning SVM system by Nanosim: identification labels to search the most ineffective sample.

4. SVM learning according to the updated samples;

5. Receiving new on-line pattern and repeating 2, 3, and 4.

After sufficient on-line learning operations, all the inefficient samples are replaced by significant on-line patterns as shown in Fig. 4.1(c).

4.1.2 Hardware implementation

The architecture of proposed on-line learning SVM system is illustrated in Fig. 4.2. This system is composed of an analog on-chip learnable SVM processor and a Winner-Take-All (WTA) circuit with N candidates. According to the learning samples $(\mathbb{X}_i, y_i)_{1 \leq i \leq N}$, on-line pattern $\mathbb{X}$ is classified by the SVM processor. As

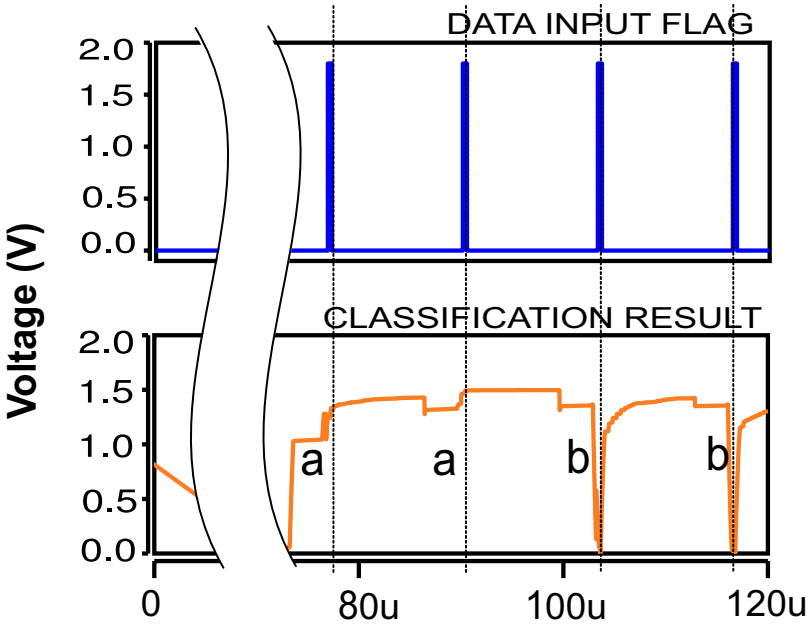


Figure 4.5: Circuit simulation results of the proposed on-line learning SVM system by Nanosim: on-line classification results.

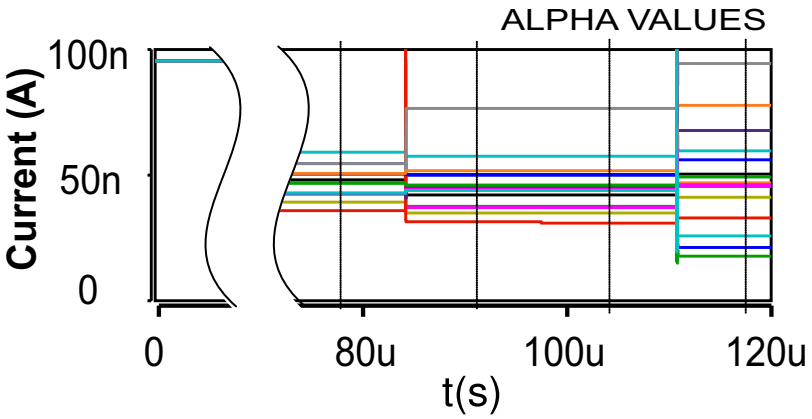


Figure 4.6: Circuit simulation results of the proposed on-line learning SVM system by Nanosim: α values during the on-line learning operations.

soon as the most ineffective sample is identified by WTA circuit, $\mathbb{X}$ replaces this sample along with classification result y. The circuit design of WTA block is given in Fig. 4.3.

4.1.3 Experiments

Images in two classes from an actual database COIL-20 are used as learning samples, and several other images are used as test patterns. The class labels are represented by a high voltage signal (class "a") and a low voltage signal (class "b"). All the images are converted into 64-dimensional vectors employing the PPED method. Receiving the on-line learning patterns, the inefficient ones of previous learning samples are indexed by high voltage signals as shown in Fig. 4.4. According to the database, all the on-line classification results in Fig. 4.5 are correct. The α values are self-adjusted while every on-line learning pattern is input as shown in Fig. 4.6.

4.1.4 An example of real-world application: object tracking problem

In order to investigate the feasibility to apply the proposed on-line SVM in the real-world task, the object tracking problem is demonstrated. A combination of FPGA and our fabricated SVM processor is used to implement the on-line-learning SVM for human face tracking. The FPGA board, which was programmed by my colleague Mr. Pushe Zhao, is employed to do the preliminary visual processes including edge detection and feature vector extraction. While a frame of video is input to the FPGA board, plenty of sub-images (windows) are converted to 64-D vectors and fed to the SVM learning chip. From the stored samples, the SVM chip recognizes the human face and returns the results to the FPGA board. Namely, the FPGA board is used as framework, and the analog SVM chip is used as a recognition core.

Traditionally, a large number of samples considering all the possibility of object's appearance should be given before the tracking. But this is not practical in some applications. The benefit to employ SVM chip is that, only a small number of initial samples are needed to recognize objects for a long sequence of video. In fact, in our experiment, only one sample of human faces and eight samples of background are initially given. Along with the on-line learning, the samples are updated when the appearances of face and background are changing. Figure 4.7 represent the contribution of the analog SVM processor to the tracking

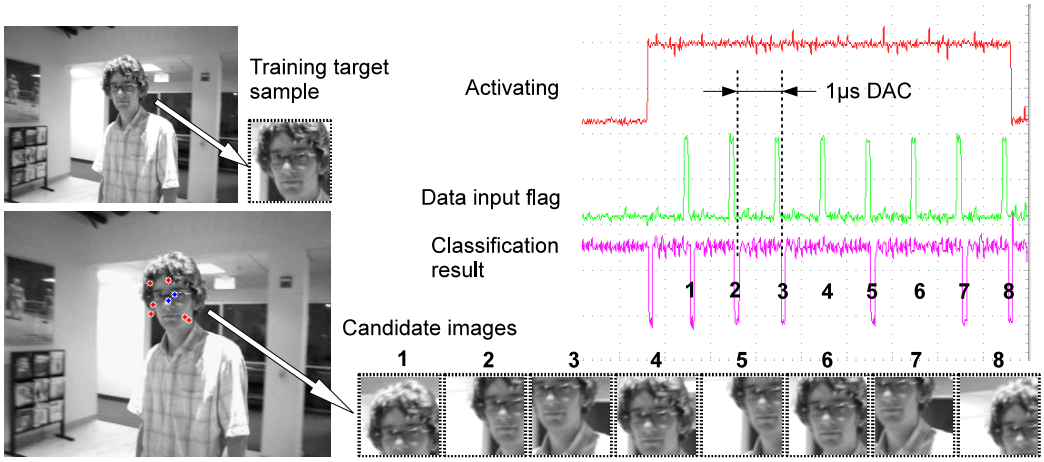


Figure 4.7: In one frame of video, the SVM chip successfully recognizes the human face from eight candidates.

task. In one frame of video, the SVM chip successfully recognizes the human face from several candidates, and returns these results to FPGA board. The updating operations for on-line strategy is managed by the FPGA.

4.2 On-Line-Learning KQC clustering

The on-line-learning strategy can be also applied in the KQC clustering problems to expend the number of samples. By previously proposed work, the on-line learning strategy (63) has been developed for the K-means algorithm. Its learning process kept receiving on-line samples when the sample space was expended. It has been proved that, this on-line learning strategy is helpful to deal with the incremental sample space and the ill initialization. However, this method was realized by a software program, and the number of dimensions of sample vectors was only two. The purpose of this work is to implement the on-line learning KQC algorithm by VLSI circuits for the categorization of high dimensional sample vectors. The previously introduced KQC processor is used as learning core of this system. In order to verify the proof-of-concept processor, the images of two objects selected from the COIL-20 database (44) are converted into 64-D feature vectors as learning samples. Upon receiving an on-line sample vector, the learning process autonomously proceeds and self-converges within one clock. According

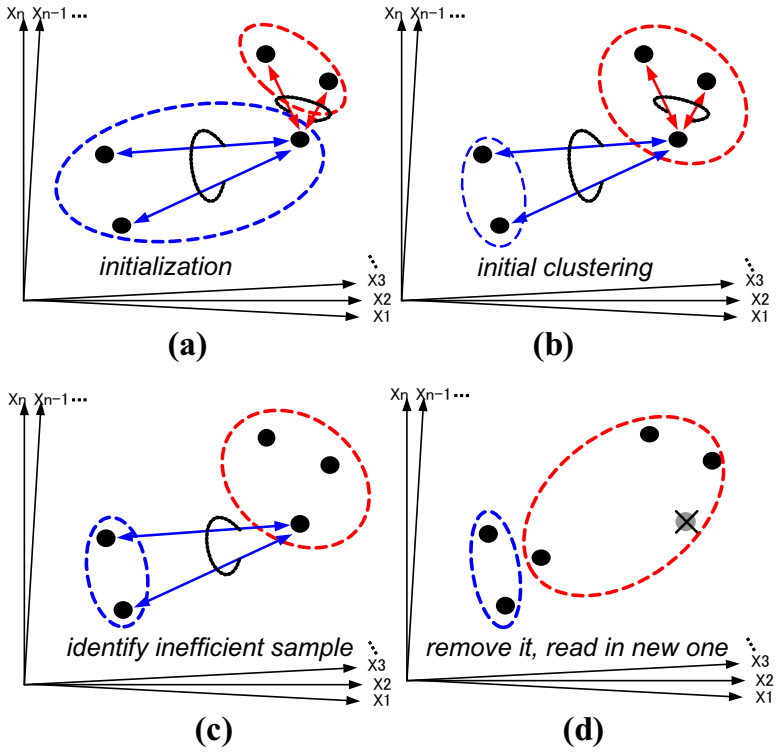


Figure 4.8: On-line-learning scheme of KQC clustering method.

to the Nanosim simulation results, all the images were categorized into correct classes with a randomly ill initialization.

4.2.1 On-line-learning KQC algorithm

With the similar mechanism to on-line-learning SVM, the essence of on-line KQC clustering is to update the learning samples along with each round of learning. At the beginning, the first round of clustering is processed by using the initial learning samples. Upon receiving a new sample vector on-line, the efficiency of each current sample vector is evaluated according to: $\min_i \sum_{j(y_j \neq y_i)} \alpha_j D(\mathbb{X}_i, \mathbb{X}_j)$, where y_i is the category label of the i-th sample. The most inefficient sample is updated by the coming sample. The learning process flow is given as follows:

1. Setting up an initial sample space with a small size;

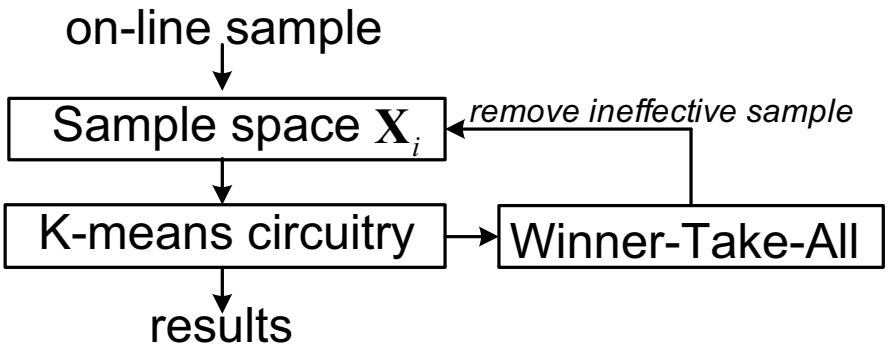


Figure 4.9: Organization of proposed on-line learning KQC system.

2. Calculating the Euclidean Distances from every vector to all the others and initializing the categorization randomly as shown in Fig. 4.8(a);

3. evaluating the similarities from a specific vector to all the categories (by calculating the average distances from the specific vector to all the vectors from the same category);

4. resetting the categorization label for this specific vector according to the evaluation as shown in Fig. 4.8(b);

5. repeating steps 3 and 4 till all the given vectors have been re-clustered;

6. results converging and updating the sample space as shown in Fig. 4.8(c) and (d);

In this manner, the number of learning samples can be expended to infinite theoretically.

4.2.2 Hardware implementation

The proof-of-concept processor is designed to realize the on-line learning KQC clustering function. Figure 4.9 illustrates the processor organization, which consists of an analog fully-parallel self-converging K-means circuitry and a winner-take-all (WTA) circuit. Sixteen sample vectors with 64 dimensions are used as

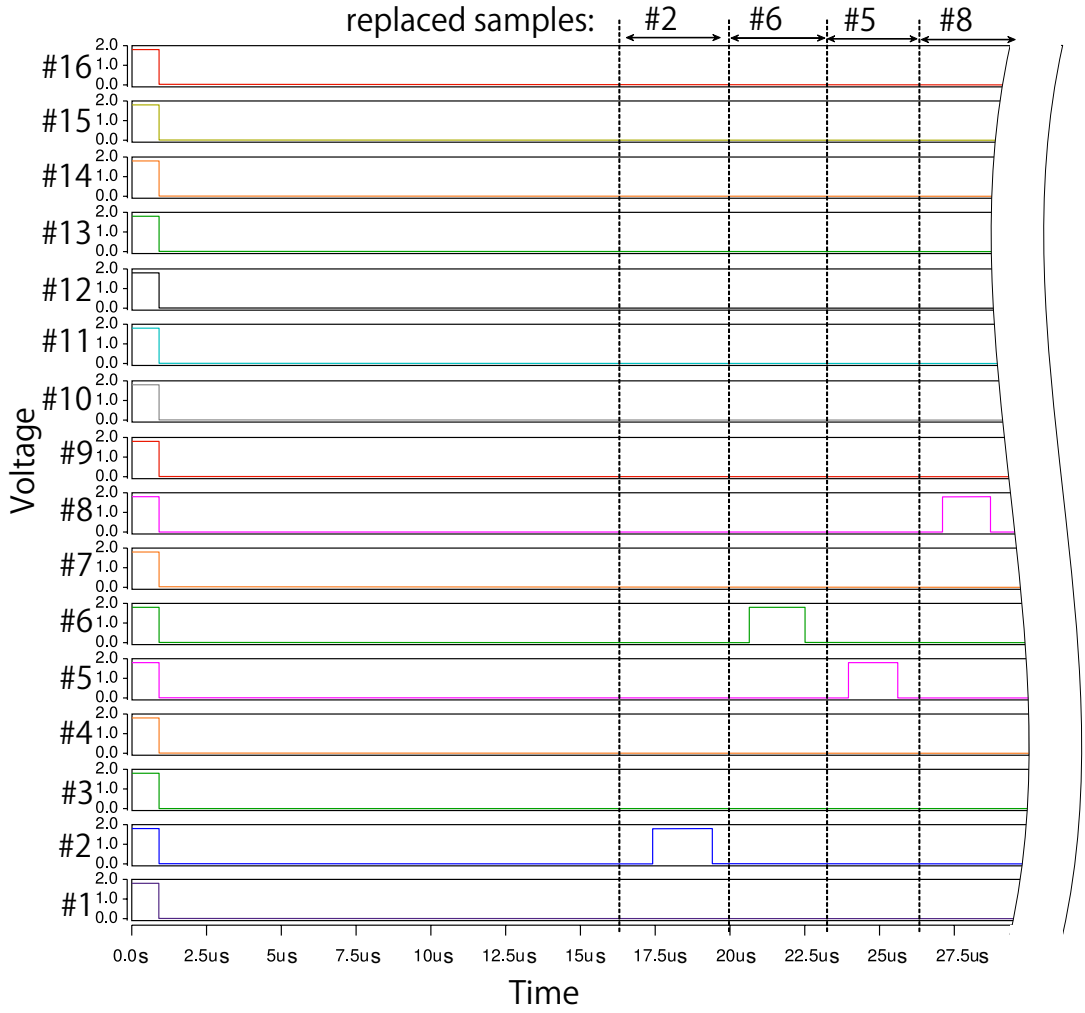


Figure 4.10: Indexes to select the most inefficient sample.

a fixed sample space. The analog K-means circuitry clusters these sample vectors into two categories in real-time. The WTA circuit is introduced to evaluate the current sample space, and identify the most inefficient sample, which will be replaced by an new coming sample on-line.

4.2.3 Simulation results of on-line learning process

The KQC on-line learning processor has a fixed hardware set for sixteen sample vectors. A small number of sample vectors are initially read in the learning cir-

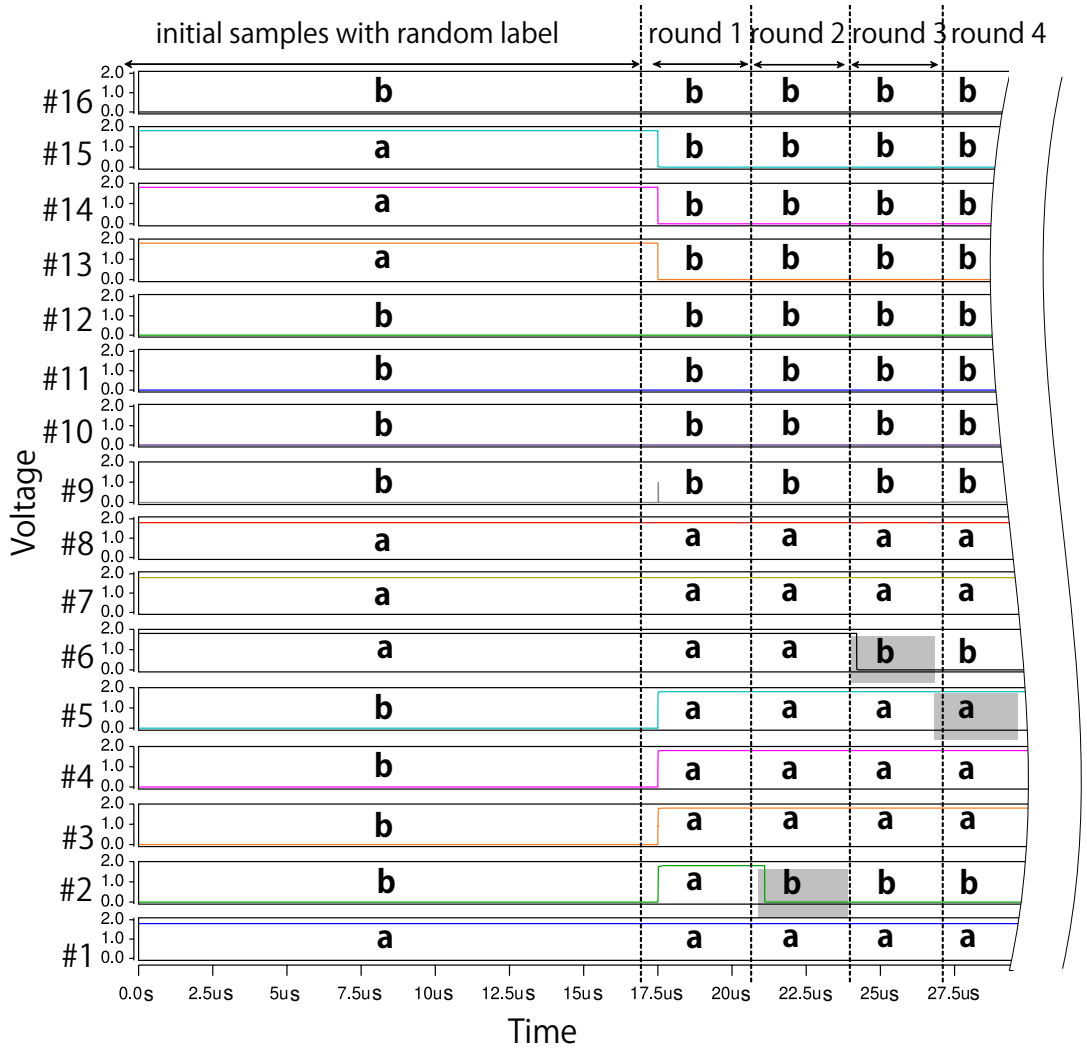


Figure 4.11: On-line learning process in different rounds. The shadow marks a sample which has been replaced by a new sample during the previous round.

cuitry, and the learning operation proceeds for the first round. During each round of learning process, the most ineffective sample is indexed by a selection-signal as shown in Fig. 4.10. Upon receiving a new sample on-line, this ineffective sample will be replaced by the new sample according to the selection-index. When the sample space is updated, the learning operation proceeds as the next round. The cluster-label for each sample is presented in Fig. 4.11 in several on-line learning rounds. Considering plenty of different learning samples (random initialization

Table 4.1: Performance comparisons.

	(61)	(62)	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
Number of samples	2905	76.8×10^3	on-line

$*$ Some of the calculations were pre-processed by the software program.

and on-line-updating samples), the learning operations successfully converge in every round of the entire on-line sequence. The shadow marks a sample which has been replaced by a new sample during the previous round. From the Nanosim simulation results, all the on-line test samples are clustered into correct categories.

4.2.4 Comparisons

The performance comparisons among this work and some other works are shown in Tab. 4.1. The learning complexity is greatly related to the number of dimensions of the sample vectors. Thus, previously reported works (61, 62) were difficult to be implemented in the highly dimensional applications. By using the proposed architecture, a larger number of dimensions and samples, even a higher learning speed were achieved with less complexity of hardware. Furthermore, the self-convergence is helpful to improve the learning performances compared with the traditional approach, which stops learning by setting a fixed number of iterations.

4.2.5 Reliability

Generally, the accuracy of calculational analog circuits is poorer than that of software programs and digital circuits. Plenty of factors influence the precision

of the storage and calculations for analog signals, which might lead to defective results of K-means learning.

The error of calculations is mainly resulted by the mismatch and deviation problems on all the devices (especially the threshold voltages of MOS transistors). On the other hand, the storage of sampling voltages also results in errors. Since the capacitors are used as analog memories, the switch channel charges and the leakage current of capacitors cause the voltage drops ΔV_{sw} and ΔV_{leak}, respectively. In the K-means learning circuitry, voltages representing the element values of patterns are broadcasted to block I controlled by sixteen sets of switches. Considering the originally stored voltage as V_{orig}, the effect of switch channel charge can be described as $\Delta V_{sw}/V_{orig} = \frac{16C_{sw}}{C_{sampling}+16C_{sw}}$, where $C_{sampling}$ and C_{sw} are the voltage sampling capacitance and the equivalent capacitance due to switch channel charge, respectively. In this sense, large sampling capacitance and small size of switching transistors are suggested to reduce the voltage drops. The voltage drops due to the leakage current of sampling capacitors are related to the entire time of usage. Therefore, this effect appears when the proposed processor operates in the on-line learning mode and the number N of on-line patterns is very large. Since the digital data with a bit-length of eight is used as input, the expected accuracy of analog voltages is $V_{DAC}/256$ (about 4 mV in this work) after D/A conversion. The information of patterns will be correctly kept in the analog memories if the voltage drop is smaller than 4 mV. Assuming the leakage current density and capacitance density are J and C_{den}, respectively, the reliable condition becomes $\frac{10NJ}{C_{den}f} < \frac{V_{DAC}}{256}$, where f is the operational frequency.

The effects of all these errors to the clustering results are also related to the learning patterns. Fortunately, in the K-means theory, the similarity comparisons among patterns are concerned rather than the absolutely accurate calculations. In some related works (64), it is reported that the poor calculational accuracy has acceptable effect on K-means learning result.

4.3 Summary

An on-line-learning strategy for implementing learning algorithms was proposed in this chapter. The essence of this strategy is to expend the learning capacity

with a consideration of limited hardware resource. This methodology is particularly suitable for our proposed hardware architecture due to high learning speed and parallelism. To verify the proposed methodology, the SVM and KQC algorithms have been implemented in a on-line-learning form and verified by circuit simulation. Even, a real-world application of object tracking problem was demonstrated as an example. It should pointed out that, the practical capacity of on-line-learning processor is strongly limited by some reliability factors.

4. ON-LINE LEARNING STRATEGY BASED ON THE FULLY PARALLEL ARCHITECTURE

5

Support Vector Domain Description

An analog VLSI hardware implementation of support vector domain description (SVDD) has been developed in this work. SVDD algorithm is found more flexible and practical than the standard SVM algorithm in the multi-class recognition problems. On the other hand, it is more complicated to solve mathematically and implement in silicon. A special solution scheme for SVDD problems is proposed on the basis of fully parallel process. The on-chip learning operation of SVDD algorithm was implemented by an analog on-chip learning processor based on the fully parallel architecture. A proof-of-concept chip was built for sixteen learning sample vectors. From the circuit simulation results, the entire learning operation is accomplished within 0.6 μs, and the domain of sample space is described by a reduced number of sample vectors. In addition, the various forms of domain description can be realized by tuning the kernel function feature dynamically.

5.1 Introduction to the Analog SVDD Processor

Traditionally, the SVM algorithm was developed for the binary classification problems. Two classes of learning samples are needed for a standard SVM classification. A complicated mechanism was introduced to solve multi-class classifica-

tion problems (65). In the real-world applications, various numbers of classes might be required, even only a single class of learning samples are available in some applications. To solve these problems, a data domain description theory (also called one-class classification) was developed as an extension of SVM theory (66, 67, 71), which is named support vector domain description (SVDD). The SVDD algorithm has been applied in some classification problems (68), even unsupervised clustering problems (70) by software programs.

However, the learning operation of SVDD algorithm requires iterative matrix computations. Therefore, software implementations of SVDD learning are very time-consuming, even if some advanced mathematical methods are employed (69). Furthermore, the conventional hardware implementation strategies can be hardly applied for SVDD problems due to the matrix computations. Regarding the highly dimensional pattern recognition, the problem becomes even more complicated.

The purpose of this work is to report a feasibility study of VLSI hardware implementation for SVDD problems. The proposed fully parallel architecture employing the Gaussian generation circuit is applied for this purpose. By using a proposed solution of SVDD, a proof-of-concept chip is built for the multi-class recognition of image patterns. From the experiment results, the entire learning operation is accomplished within 0.6 μs, and the domain of sample space is described by a reduced number of sample vectors. Furthermore, various forms of domain descriptions can be realized by tuning some learning parameters according to the original SVDD theory. All the test image patterns are classified into correct classes from the measurement results.

5.2 The Application of SVDD in This Work

The target application of this work is the image recognition with multi-class samples. In the standard SVM algorithm, learning is processed by using two classes of learning samples. Thus, SVM is one of the so-called binary classification algorithms. However, multi-class recognition is always needed in the real-world applications. Figure 5.1 represents the multi-class classification task for images employing the SVDD mechanism.

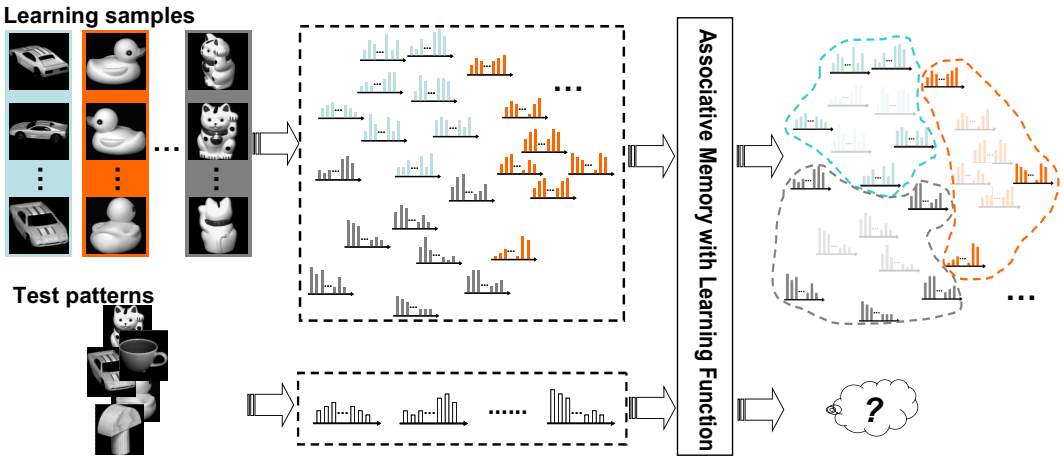


Figure 5.1: Multi-class classification task for images employing the SVDD mechanism.

In fact, each class of samples and its learning operation are independent to others. Therefore, the number of classes can be freely increased or decreased according to the application demands. One VLSI learning chip is applied for the on-chip learning of one class. During the recognition session, the test pattern is broadcasted to all the chips as it is shown in Fig. 5.2. The result is output by this system in real-time. In this manner, multi-class recognition is achieved with a high performances.

There are several benefits using the SVDD algorithm in the image recognition problems. Sharing some properties with the standard SVM, SVDD algorithm can obtain an accurate classification boundary by remarkably reduced number of samples. On the other hand, the number of classes can be freely expended. Furthermore, the volume of data domain can be reasonably given by the learning operation.

5.3 Algorithm

To classify the n-dimensional vectors $\mathbb{X}$s with the form of $\mathbb{X} = (x_1, x_2, \cdots, x_n)$. A set of learning samples $\{\mathbb{X}_i, 1 \leq i \leq N\}$ is given, where N is the number of samples.

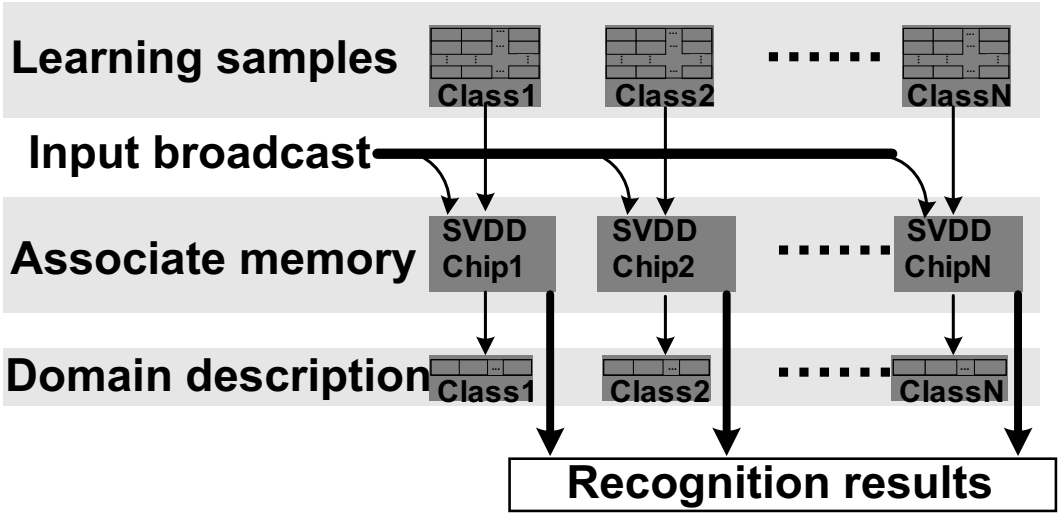


Figure 5.2: Organization of multi-class recognition system employing SVDD algorithm.

We try to find a sphere with minimum volume R, containing all (or most of) the data objects, which is defined by:

$$F(R, \mathbf{a}, \xi_i) = R^2 + C \sum_i \xi_i, \tag{5.1}$$

where, $\mathbf{a}$ is the center of this sphere; ξ_i is a slack variable for error tolerance; and parameter C gives the trade-off between simplicity (or volume of the sphere) and the number of errors (number of target objects rejected). This function has to be minimized under the constraints:

$$(\mathbb{X}_i - \mathbf{a})^{\mathbf{T}}(\mathbb{X}_i - \mathbf{a}) \leq R^2 + \xi_i, \tag{5.2}$$

Applying the theory developed for original SVDD algorithm, and considering the Gaussian Kernel function $K(\mathbb{X}_i, \mathbb{X}_j) = e^{-\frac{\|\mathbb{X}_i - \mathbb{X}_j\|^2}{\sigma}}$, the task becomes the following Quadratic Programming (QP) problem:

$$\min L = 1 - \sum_i \alpha_i^2 - \sum_{i \neq j} \alpha_i \alpha_j K(\mathbb{X}_i, \mathbb{X}_j), \tag{5.3}$$

under the constraints:

$$\sum_i \alpha_i = 1, \tag{5.4}$$

and $0 \leq \alpha_i \leq C$. In order to solve this QP problem, another Lagrangian function is constructed with multiplier λ:

$$\begin{array}{c} \mathfrak{L} = L + \lambda(\sum_i \alpha_i - 1) \\ = 1 - \sum_i \alpha_i^2 - \sum_{i \neq j} \alpha_i \alpha_j K(\mathbb{X}_i, \mathbb{X}_j) + \lambda(\sum_i \alpha_i - 1) \end{array} . \tag{5.5}$$

Then, the following equations should be solved to minimize L:

$$\begin{cases} \frac{\partial \mathfrak{L}}{\partial \alpha_i} = 0 \\ \sum_i \alpha_i - 1 = 0 \end{cases} \tag{5.6}$$

The expansion of these equations is in the linear form as:

$$\begin{cases} -2\alpha_1 - \sum_{j \neq 1} \alpha_j K(\mathbb{X}_1, \mathbb{X}_j) + \lambda = 0 \\ -2\alpha_2 - \sum_{j \neq 2} \alpha_j K(\mathbb{X}_2, \mathbb{X}_j) + \lambda = 0 \\ \vdots \\ -2\alpha_N - \sum_{j \neq N} \alpha_j K(\mathbb{X}_N, \mathbb{X}_j) + \lambda = 0 \\ \alpha_1 + \alpha_2 + \ldots + \alpha_N = 1 \end{cases} . \tag{5.7}$$

With the consideration of Gaussian function kernel ($\alpha_i K(\mathbb{X}_i, \mathbb{X}_i) = \alpha_i$), these linear equations can be equivalently converted into:

$$\begin{cases} -2\alpha_1 - \sum_{j \neq 1} \alpha_j K(\mathbb{X}_1, \mathbb{X}_j) + \lambda = 0 \\ -2\alpha_2 - \sum_{j \neq 2} \alpha_j K(\mathbb{X}_2, \mathbb{X}_j) + \lambda = 0 \\ \vdots \\ -2\alpha_N - \sum_{j \neq N} \alpha_j K(\mathbb{X}_N, \mathbb{X}_j) + \lambda = 0 \\ -\sum_i \sum_j \alpha_j K(\mathbb{X}_i, \mathbb{X}_j) + N\lambda = 1 \end{cases} . \tag{5.8}$$

The Jacobian Iterative method is applied to solve these linear equations. Thus, the iterative updating rule is obtained as:

$$\begin{cases} \alpha_i \leftarrow \frac{1}{2}(\lambda - \sum_{j \neq i} \alpha_j K_{ij}) \\ \lambda \leftarrow \frac{1}{N}(1 + \sum_i \sum_j \alpha_j K(\mathbb{X}_i, \mathbb{X}_j)) \end{cases} , \tag{5.9}$$

By considering the upper and lower boundary of α and λ, the final updating rule becomes:

$$\begin{cases} \alpha_i \leftarrow \max(0, \min(\frac{1}{2}(\lambda - \sum_{j \neq i} \alpha_j K_{ij}), C)) \\ \lambda \leftarrow \max(0, \frac{1}{N}(1 + \sum_i \sum_j \alpha_j K(\mathbb{X}_i, \mathbb{X}_j))) \end{cases} . \tag{5.10}$$

The trade-off parameter C should be set within an interval of $1/N \leq C \leq 1$ to guarantee the solution of QP problem can be obtained (66).

Obviously, a large scale of parallel matrix computation is needed to update λ by this method. It is the reason why the fully parallel architecture is considered in this work based on the analog Gaussian-cell array.

During the test session, to determine whether a test object $\mathbb{Z}$ is within the sphere, the distance to the center of the sphere has to be calculated. When $(\mathbb{Z}-\mathbf{a})^{\mathbf{T}}(\mathbb{Z}-\mathbf{a}) \leq R^2$ is satisfied, the object $\mathbb{Z}$ is accepted. Expressing the sphere center in forms of the support vectors, the object $\mathbb{Z}$ is accepted when

$$1 - 2\sum_{i} \alpha_i K(\mathbb{Z}, \mathbb{X}_i) + \sum_{i,j} \alpha_i K(\mathbb{X}_i, \mathbb{X}_j) \leq R^2. \tag{5.11}$$

Theoretically, the distance from any support vector $\mathbb{X}_s$ to the center should be R. Thus, the constant items in Eq. 5.11 could be observed by using any support vector in the following form:

$$1 + \sum_{i,j} \alpha_i K(\mathbb{X}_i, \mathbb{X}_j) - R^2 = 2\sum_{i} \alpha_i K(\mathbb{X}_s, \mathbb{X}_i). \tag{5.12}$$

In this manner, the object $\mathbb{Z}$ is accepted when the following condition is satisfied:

$$\sum_{i} \alpha_i K(\mathbb{Z}, \mathbb{X}_i) \geq \sum_{i} \alpha_i K(\mathbb{X}_s, \mathbb{X}_i). \tag{5.13}$$

In this work, the support vector with the largest alpha value is selected for observation.

5.4 Hardware Implementation

A proof-of-concept processor is built to implement the learning operation of SVDD algorithm in a $0.18\mu m$ CMOS technology. The organization of proposed processor is illustrated in Fig. 5.3 along with its chip micrograph in Fig. 5.4. The entire processor contains an analog Gaussian-cell array, a set of alpha adjusters, and a lambda adjuster. All the Gaussian kernel functions are carried out by the Gaussian-cell array in real-time and fully parallel. During the learning session, these function values are fed into the alpha and lambda adjusters; the updated alpha and lambda values are freely fed back to the Gaussian-cell array in the forms

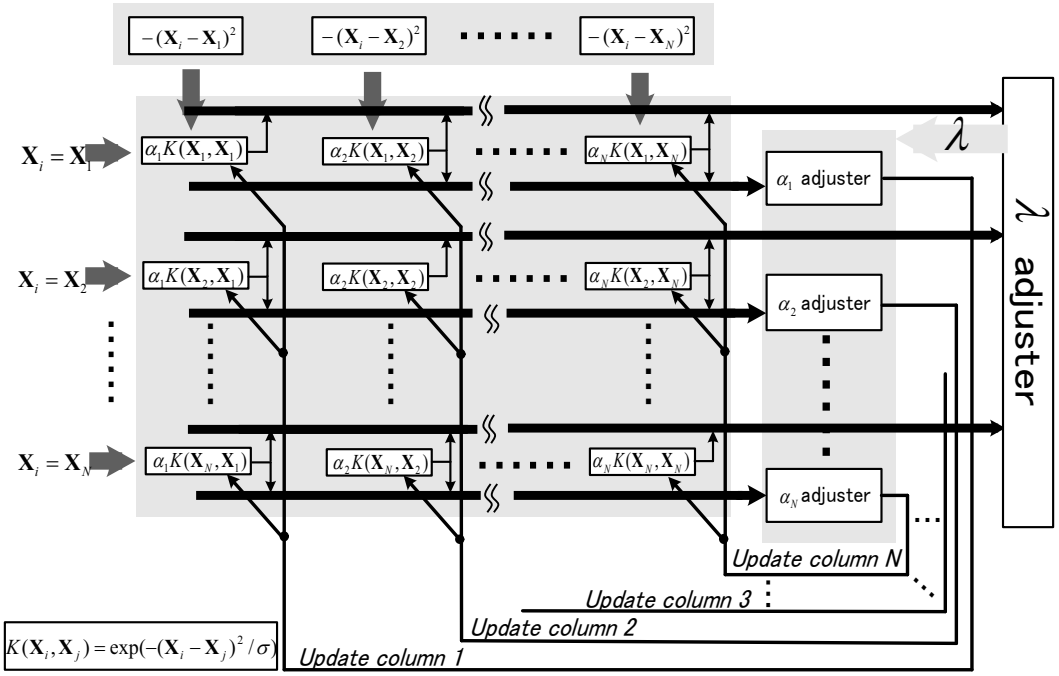


Figure 5.3: Fully parallel on-chip learning SVDD processor.

of analog signals. In this manner, the learning operation proceeds autonomously and self-converges without any clock-based control. During the test session, only the Gaussian cells in the first row are used, and all the calculations for an object test are also carried out in parallel and real-time according to Eq. 5.13.

The analog Gaussian generation circuit has been introduced in the previous chapters. Receiving two vectors in the form of voltages $\mathbb{V}_i = (v_{i1}, v_{i2})$ and $\mathbb{V}_j = (v_{j1}, v_{j2})$, the Gaussian function related to these two vectors can be computed as

$$I_{out} = \frac{I_\alpha}{2} e^{-\gamma||\mathbb{V}_i - \mathbb{V}_j||^2}, \tag{5.14}$$

where

$$\gamma = \frac{K_n}{(V_{dd} - V_{bias} - |V_{thp}|)(\sqrt{K_p} + \sqrt{K_n})^2}, \tag{5.15}$$

V_{thn} and V_{thp} are the threshold voltages of the n-type and p-type MOS transistors, respectively. The current I_α reflects the alpha value in Eq. 5.10, which is dynamically programmed by the alpha adjusters. The spread-width of Gaussian function feature is programmed by tuning the voltage signal V_{bias}.

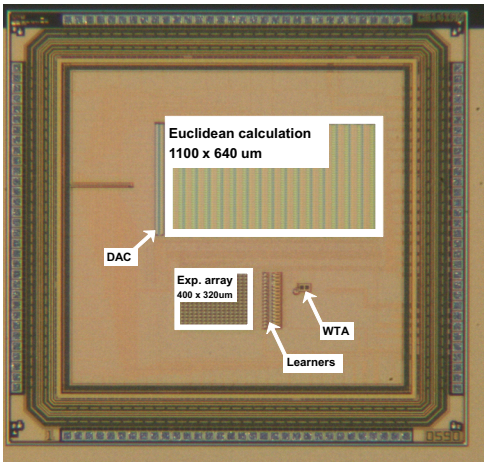


Figure 5.4: Micrograph of fabricated SVDD learning chip.

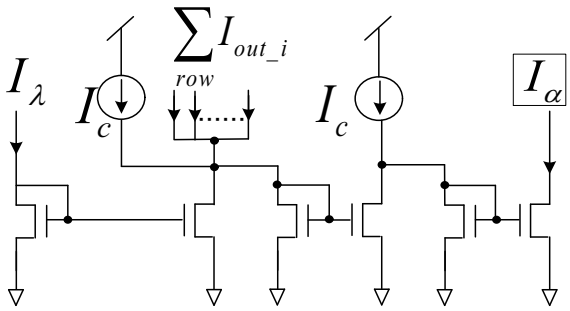


Figure 5.5: Schematic of alpha adjuster circuit.

A current mirror based circuit is designed as the alpha adjuster as shown in Fig. 5.5. By collecting the output current of Gaussian cells in a specific row, the updated current I_α is output to the respective column of Gaussian-cell array according to update rule in Eq. 5.10. The current source I_c is introduced to reflect the trade-off parameter C. The circuit schematic of lambda adjuster is shown in Fig. **??**. The lambda adjuster collects the output current from all the Gaussian cells, and the updated lambda value is fed into all the alpha adjusters. The current source I_{unit} reflects the constant factor "1" in Eq. 10. Therefore, the parameter C is represented by $C = I_c/I_{unit}$.

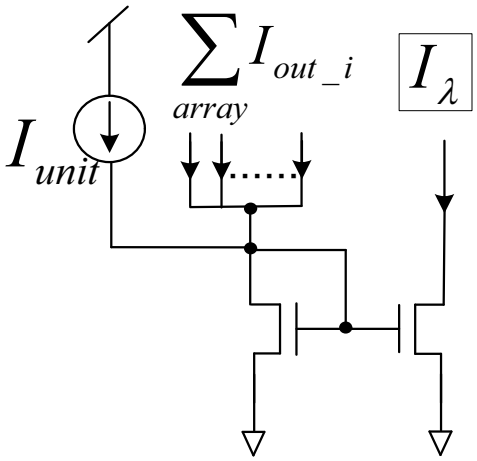
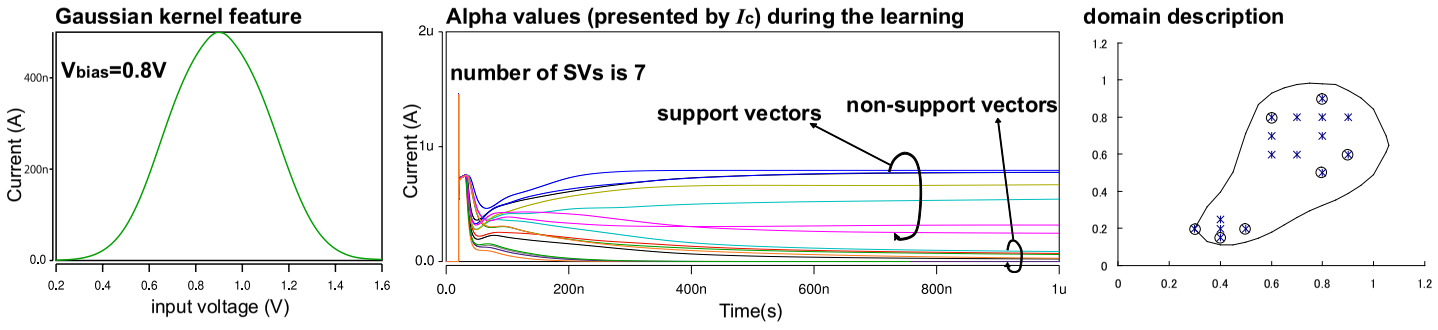


Figure 5.6: Schematic of alpha adjuster lambda.

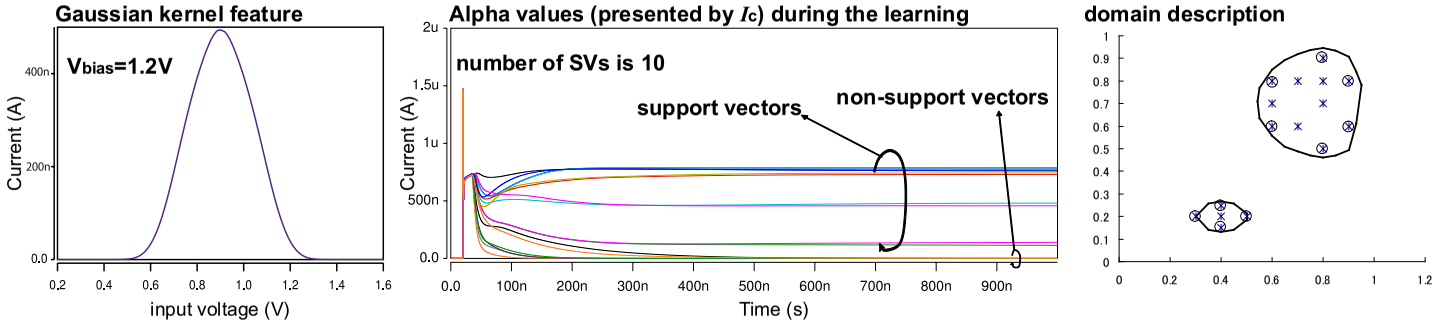
5.5 Experiments

5.5.1 2-D pattern recognition employing SVDD

A toy-example with sixteen learning samples is set up to verify the performances of the proposed proof-of-concept processor. Each learning sample is represented by a set of voltage signals. For instance, the learning sample $\mathbb{X} = \{0.6, 0.6\}$ is represented by $\mathbb{V} = \{0.6V, 0.6V\}$. In this application, a Gaussian-cell array with 16×16 elements is constructed. The HSPICE simulation results and description boundaries are shown in Fig. 9 by applying different Gaussian function features. For this experiment, the parameter C is set as 0.08 by setting the current I_{unit} and I_c as $10\mu A$ and $0.8\mu A$, respectively. From the simulation results, there is no any outlier with these configurations, and the learning operation is accomplished within about $0.6\mu s$. When a wide spread of Gaussian function feature is applied, a rough domain is described by 7 support vectors as it is shown in Fig . 5.7(a). An improved domain description is obtained by applying a narrow spread of Gaussian function feature, which is shown in Fig. 5.7(b). However, the number of support vectors is increased in this case.



(a)



(b)

Figure 5.7: Support vector domain description of a toy-example with sixteen learning samples: (a) when a wide spread of Gaussian function feature is applied, a rough boundary is described by 7 support vectors; (b) when a narrow spread of Gaussian function feature is applied, an improved boundary is described by 10 support vectors.

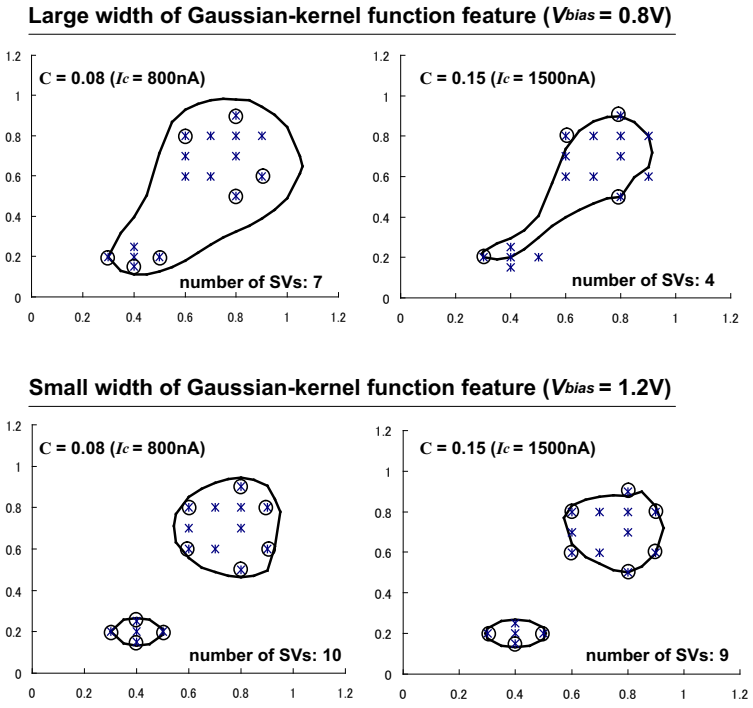


Figure 5.8: Variations of support vector domain description when different parameter C is set.

The effect of parameter C on the variations of domain description is also investigated. Using the same learning samples, the parameter C is set as 0.15 by setting the current I_{unit} and I_c as $10\mu A$ and $1.5\mu A$. The number of support vectors is obviously reduced. On the other hand, some errors (outliers) are introduced by these domain descriptions as shown in Fig. 5.8.

5.5.2 64-D pattern recognition employing SVDD

Regarding the real-world applications, an associative memory system for image recognition is constructed by the SVDD learning chips. In the proof-of-concept system, three fabricated chips are used as SVDD learner. Namely, three classes of images from the COIL-20 database are employed as learning samples. The recognition performances of single SVDD learning chip is given in Fig. 5.9 by the circuit simulation results.

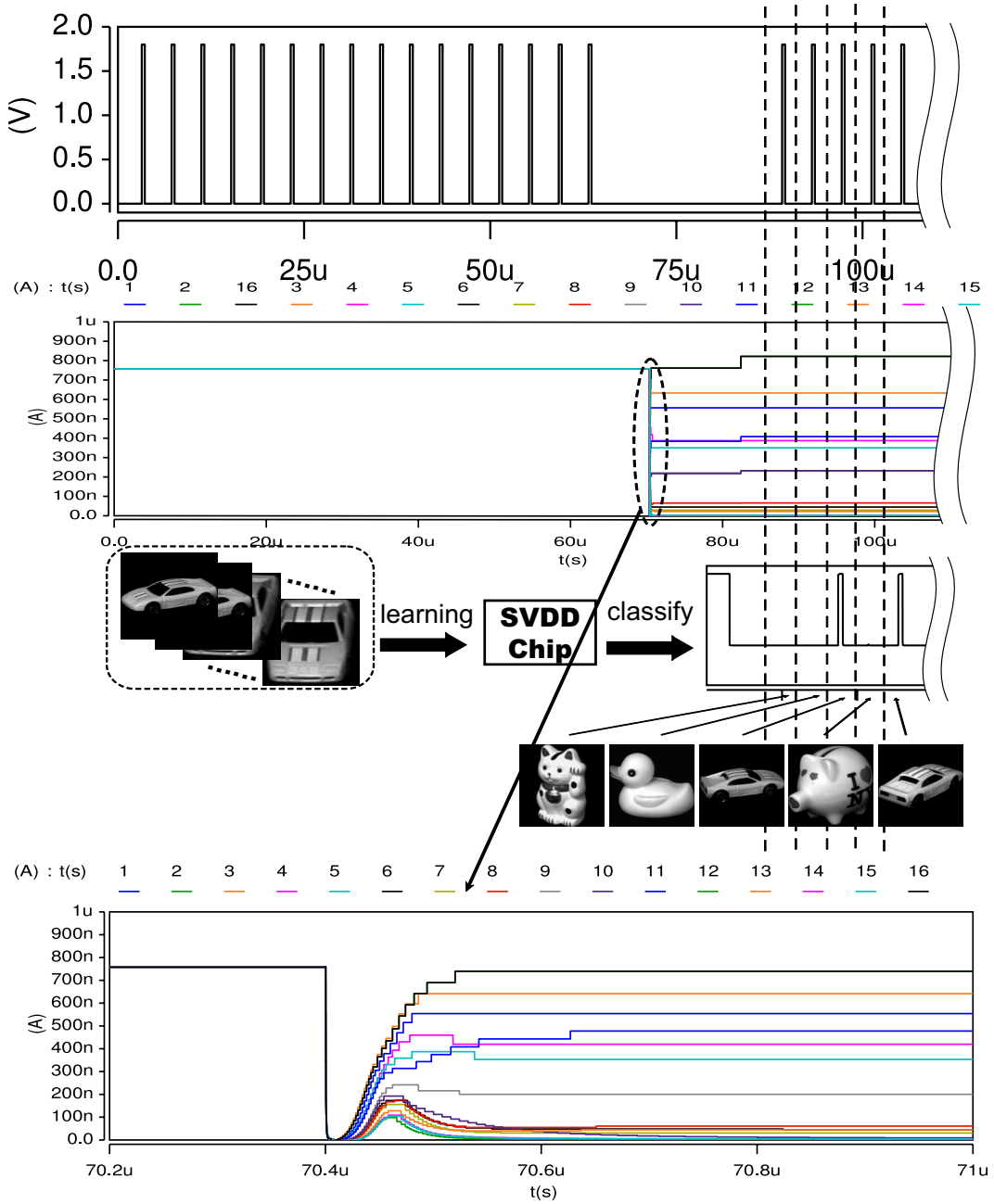


Figure 5.9: Circuit simulation results of 64-D learning/classifying performances of single SVDD learning chip.

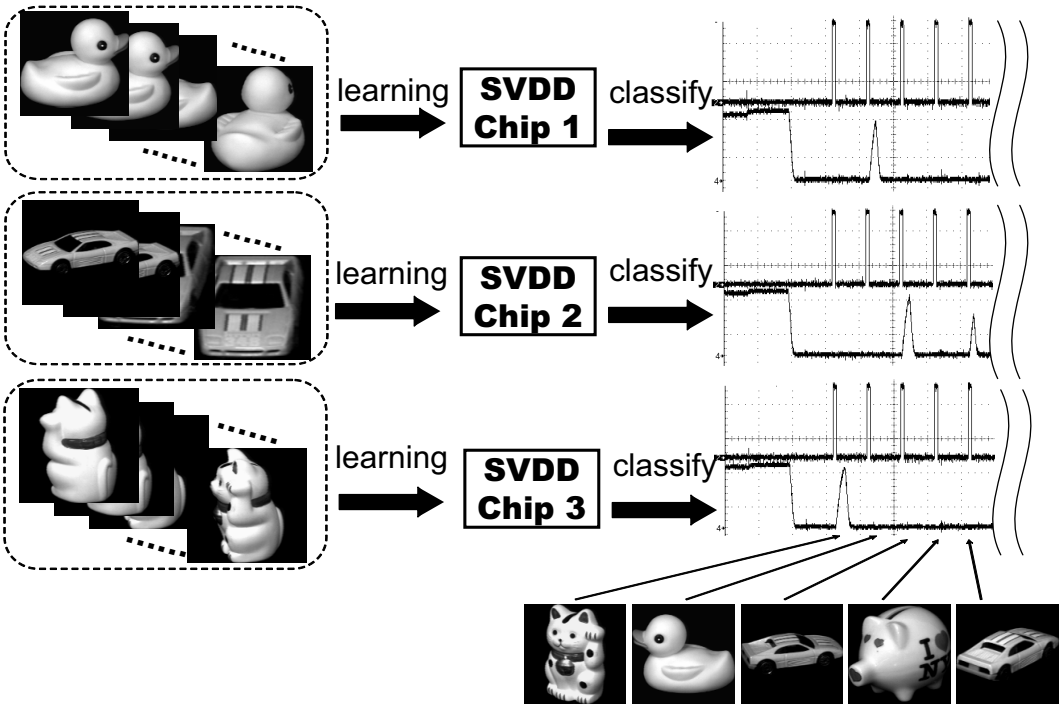


Figure 5.10: Chip measurement results of an associative memory system employing SVDD learning processor: three chips are used as SVDD processors independently. The test patterns are broadcasted to all the chips.

Chip measurement results for the proposed associative memory system is shown in Fig. 5.10. Three chips are used as SVDD processors independently. The test patterns are broadcasted to all the chips. In this manner, three classes of learning samples are used for data domain description. After the learning session, test test patterns are broadcasted to all the chips. If the test pattern belongs to a specific class, the respective chip outputs a signal of high voltage. From this figure, some test patterns do not belong to any sample class. Thus, none of the output signals is high voltage.

5.5.3 Comparisons

Table 5.1: Performance comparisons.

	(41)	(68)	This work
Implementation	Analog VLSI	Software (Matlab)	Analog VLSI
Algorithm	Standard SVM	SVDD	SVDD
Learning parallelism	Row parallel	N/A	Fully parallel
Kernel function	Gaussian	Gaussian	Gaussian
No. of samples	12	unlimited	16
No. of dimensions	2	$2 \sim 18$	$2 \sim 64$
Learning speed	$16.7/l^* \times 10^6$ vectors/sec	2.78×10^3 vectors/sec (2-D vectors)	26.7×10^6 vectors/sec

* l is the number of iterations for convergence.

The performance of proposed SVDD on-chip learnable processor is presented in Tab. 5.1. An analog implementation of standard SVM and a software implementation of SVDD algorithm are introduced as comparisons. Since an analog fully parallel array is constructed in this work, the learning speed is much higher compared with the software implementation, even the analog implementation of standard SVM.

5.6 Summary

The feasibility of analog VLSI implementation of on-chip learnable SVDD was studied in this work. For this purpose, a fully parallel analog on-chip learning SVDD processor was built in a $0.18\mu m$ CMOS technology. The learning operation of this SVDD processor autonomously proceeds without any clock-based control, and self-converges with a high speed. From the HSPICE simulation results of a toy-example, the learning can be accomplished within $0.6\mu s$, and the domain boundary is described by a reduced number of learning samples. Furthermore, various forms of domain description were also demonstrated by circuit simulation results. For the multi-class recognition problems, an associative memory system was built employing three SVDD learning chips. Three classes of real images were used as learning samples. From the measurement results, all the test patterns were classified into correct classes respectively.

6

Conclusion

6.1 Summary of This Thesis

A fully parallel analog VLSI architecture was proposed in this thesis for implementing learning algorithms. Several analog circuitries were designed to carry out the complex functions such as Gaussian function and Euclidean distance. These computations in the learning algorithms can be done in real time within the compact chip area. Employing these analog circuitries, a general applied analog architecture was developed preventing the chip area explosion problem. Since the chaos of analog signals is used for learning instead of clock-based numerical iterations, the learning operation can be accomplished autonomously and self-converges with a high speed.

In order to solve some pattern recognition problems, several machine learning algorithms were implemented by the proposed fully parallel architecture. A fully parallel SVM on-chip learning processor was built for pattern classification problem. As the kernel element of SVM processor, an analog Gaussian generation circuit was developed for highly dimensional pattern vectors. The center, height, and width of the generated Gaussian function feature can all be programmed easily. Furthermore, the chip-area-hungry part for highly dimensional Euclidean distance computations and the much smaller part for exponential computation are built separately. Only the exponential computing circuits should be duplicated for a high degree of parallelism. Therefore, the fully parallel architecture

was fabricated within a compact chip area. To verify the performance of proposed SVM processor, sixteen object images from a database were converted into 64-dimensional vectors and fed into the processor as learning samples. After self-learning, several other vectors were used as test patterns. From measurement results, the SVM processor classified all the testing patterns into correct classes with a high speed.

The proposed fully parallel architecture can be also used to implement the unsupervised machine learning algorithms. On the basis of K-means mechanism, which is an important pattern clustering algorithm, a hardware efficient version was developed and named as KQCs method. By implementing this modified clustering algorithm, the proposed analog fully parallel architecture was applied to solve the unsupervised pattern clustering problem. The proof-of-concept processor was designed for 64-dimensional vectors categorization. From the circuit simulation results, this processor correctly carried out the learning function for image pattern clustering in a high speed.

Furthermore, an on-line-learning strategy was proposed to expend the number of learning sample considering the limited hardware resource. The learning operation can be repeated for many times by the same VLSI processor. Learning results are harvested in each round of learning. At the same time, the efficiency or importance of each learning sample is evaluated in fully parallel. Some parts of VLSI processors occupied by inefficient samples can be released for accepting new comers on-line. In this manner, the capacity of on-chip learning processors is expended with the consideration of limited hardware resource. Both SVM and KQC algorithm were implemented in the scheme of on-line learning, and verified by the circuit simulation results.

At last, the SVDD algorithm was implemented by the proposed fully parallel architecture, which was found more advanced than the standard SVM in the multi-class classification problems. An analog SVDD learning processor was built for the 64 dimensional vectors. Employing three chips of this processor, an associative memory system was constructed for the recognition of images from three classes. From chip measurement results, all the test patterns were correctly recognized with a high speed.

6.2 Perspectives

As it is well known, the analog computational circuits have poorer accuracy than that of software program or digital circuits. Moreover, the storage and processing of analog signals are not always reliable. This is one of important reasons that machine learning algorithms were mainly implemented by software programs and digital VLSI circuits by the previously developed works. In this thesis, a fully parallel architecture of analog VLSI circuits was introduced for implementing some learning algorithms. This architecture performs several benefits over learning speed, chip area and power consumption. It has even been fabricated in silicon and applied for solving some real-world problems. However, the reliability and flexibility are still poorer than the digital approaches. Thus, deeper investigation on the reliability issue is always needed for the analog implementations.

One of reasonable solution to reliably apply analog learning processor is the combination of analog and digital circuits (even software) as it was pointed out by some early works (14). We have ever tried the combination of analog SVM learning chip and FPGA board, even applied it for the real-world task. This attempt encourages us to make further exploration of analog-digital mixed processors.

Ambitiously, new fabricated technologies even physics devices are desired in the future exploration. For instance, if the 3-D fabricated technology is available in our work, the number of vector dimensions would not be limited by the pins of a VLSI chip. In the beyond CMOS era, the use of analog signals could be promising and exciting.

Appendix A

CMOS Supporting Circuitries for Nano-Oscillator-Based Associative Memories

the pattern recognition problem can be solved by using not only machine learning algorithms but also some other soft-computing technologies based on the associative memory function. "Let physics do computing" is a promising approach to new-paradigm computing in the beyond CMOS era. Building associative memories based on the physics of nano oscillators presents a lot of potential for pattern recognition. Using CMOS ring oscillators to emulate the nano oscillator behavior, how to produce the associative memory function and to use it for image recognition is investigated in this thesis.

A.1 Introduction

Facing the fundamental limitations in the scaling of conventional devices, utilizing the physics of nano-scale devices directly for computation is quite appealing. It not only enhances the functionality per device but also allows us to integrate a profusion of high-functionality devices in a small chip area. It was proposed to use the highly non-linear resonance-type I-V characteristics in quantum-effect devices for building associative memories (72, 73). The concept was verified by experiments using simple CMOS circuits emulating the resonance characteristics

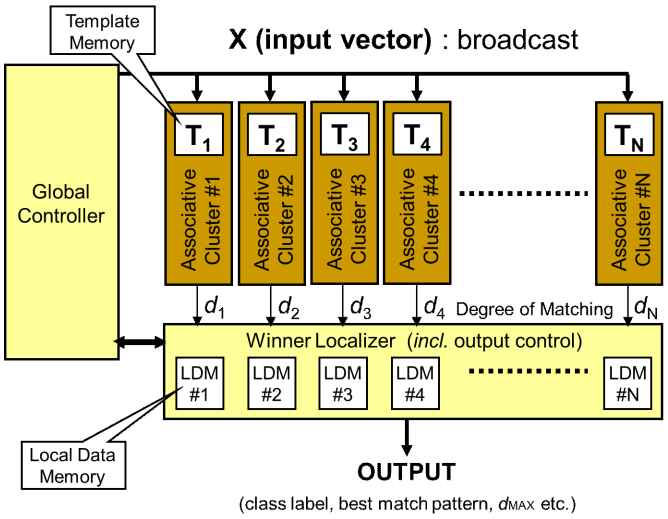


Figure A.1: Associative memory configuration.

(**?**), and also using single electron transistors operating at a room temperature (74). On the other hand, it is well known that the non-linear dynamics of coupled oscillators presents rich computational powers (75). In the non-Boolean architecture project supported by the Intel Labs University Research office, methodologies have been explored to build associative memories based on the dynamics of coupled nano oscillators, such as STOs (76) and RBTs (77).

In this appendix, the supporting circuits that interface between the fabric of nano device oscillators and digital computing is presented. In order to clarify the role of CMOS supporting circuitries, nano oscillators are emulated by frequency-tunable CMOS ring oscillators and the recognition operation has been demonstrated by HSPICE simulation.

A.2 Associative Memory Architecture

Fig. A.1 illustrates the basic associative memory configuration composed of multiple associative clusters. Each associative cluster computes the degree of matching (similarity) between the input vector $\mathbb{X}$ and the template vector $\mathbb{T}_i$ stored in the cluster. The maximum degree-of-match location is identified by the winner localizer and a flag "1" is set at the location. (Such a circuit is called the

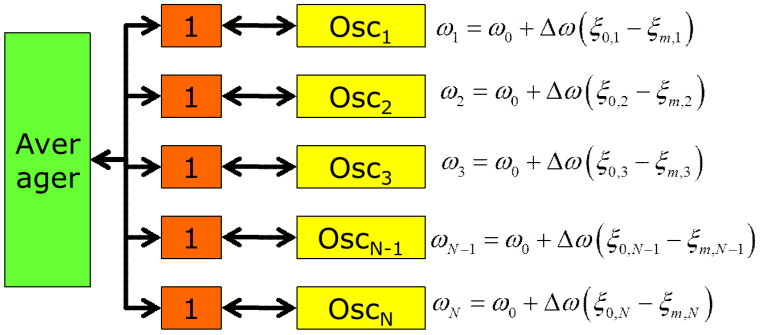


Figure A.2: Star frequency keying model.

winner-take-all (WTA) and a variety of architectures have been developed either in analog or digital CMOS technologies.) This flag controls the data output from the local data memory (LDM) at the winner location. In most cases, output is the class label, thus the classification or recognition of the input data being accomplished. In either digital (78) or analog (19) CMOS implementation of associative memories, associative clusters occupy the largest area on the chip. Therefore, in the non-Boolean architecture project it is planned to build the associative clusters in the nano device physics domain using coupled nano oscillators and other parts by conventional CMOS circuits.

An important issue here is how to implement the template memories that store the pattern data to be matched with input patterns preferably with non-volatility. In CMOS implementations (19, 78), SRAMs were used. In the case of analog matching cell, on-chip D/A converters were employed and analog template data were temporarily stored as charges on the capacitors in the analog matching cell (19). In nano device implementation, spin torque transfer memory could be a good choice if spin-transfer nano-oscillators (STNOs) are employed in building associative memories.

Two schemes have been explored for coupled-oscillator-based associative memories: "star phase keying" and "star frequency keying" (79). The former was applied to building Hopfield network architecture and the results by Hoppensteadt and Izhikevichin (80) were reproduced. Fig. A.2 shows the latter scheme in which each nano oscillator frequency is shifted by the difference between the template and input vector elements, and their outputs are summed by an averager and the

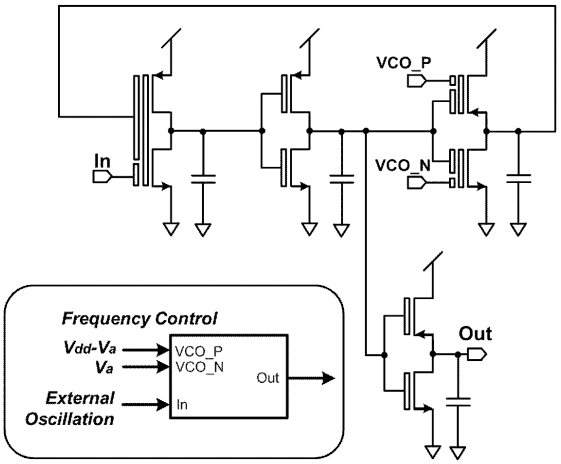
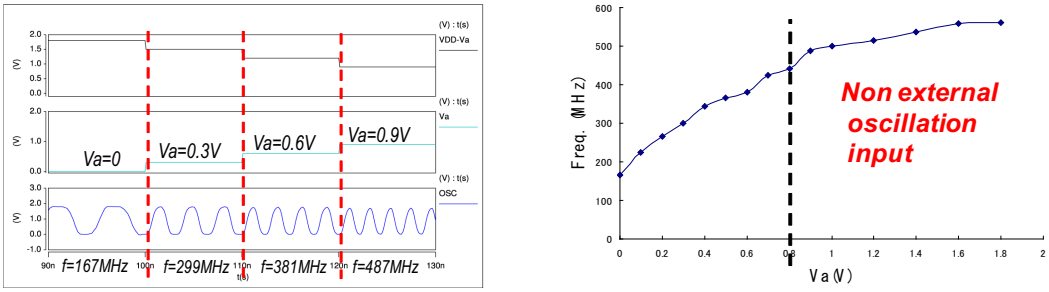


Figure A.3: Frequency-tunable CMOS ring oscillator emulating a nano oscillator. The insert at the bottom left shows the symbol representing the oscillator.



(a) (b)

Figure A.4: Natural frequency control vs. V_a.

sum is broadcasted back to each oscillator. The synchronization among oscillators maximizes when the input vector is close to the template vector because oscillator frequencies come closer to each other. In STNOs, the oscillation frequency is easily tuned by the DC bias current (81).

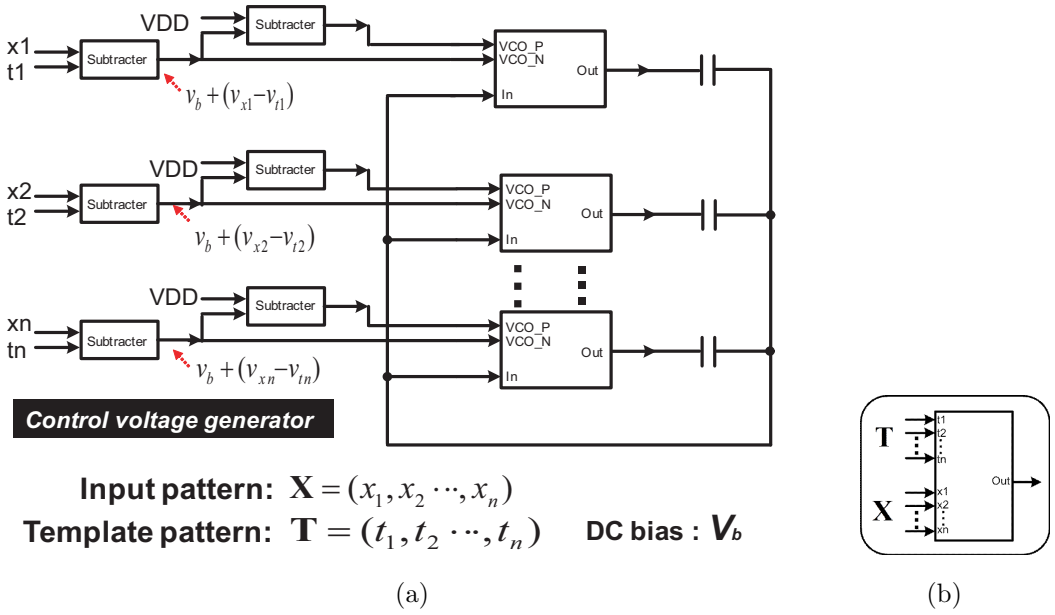


Figure A.5: (a) Associative cluster composed of frequency-tunable CMOS ring oscillators. (b) A symbol representing associative cluster.

A.3 Associative Memory Circuits

A.3.1 Emulating STNO by neuron MOS ring oscillator

A three-stage CMOS ring oscillator shown in Fig. A.3 was employed to emulate a frequency tunable oscillator. The variable threshold concept of neuron MOS (multiple-input floating-gate MOS (82)) is used in the last stage CMOS inverter, thus making the stage delay variable. When a positive DC voltage V_a is given to the input terminal marked VCO_N, it boosts up the floating gate voltage of the NMOS via capacitive coupling and reduces the apparent threshold voltage of the NMOS as seen from the other input terminal. $V_{dd}-V_a$ is applied to the control terminal VCO_P, which lowers the apparent PMOS threshold. As a result, the signal propagate delay in this inverter stage is reduced. The capacitance coupling is also employed in the first stage, thus enabling the mixing of external oscillation with internal oscillation.

Dynamical control of the natural frequency by control voltage V_a is demon-

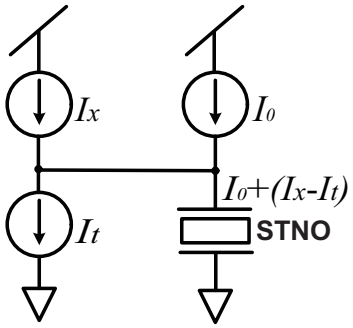


Figure A.6: DC current biasing scheme of STNOs for star frequency keying model.

strated in Fig. A.4(a), where no external oscillation was applied to "In" terminal. The results were obtained by HSPICE simulation where the device parameters of a typical $1.8\mu m$ CMOS technology was employed for the simulation. Fig. A.4(b) shows the natural frequency as a function of control voltage V_a.

A.3.2 Associative cluster circuit

Fig. A.5(a) depicts the associative cluster configuration in which the output from each oscillator is averaged via capacitance coupling and the averaged value is fed back to the input ("In" terminal) of each oscillator. The control voltage generator produces a voltage proportional to the difference between the input and template vector elements, determining the frequency shift in each oscillator. In the present simulation control voltage generators were configured using OP amps. In real STNOs in which DC current bias is used to determine the frequency, such control signals can be rather easily produced via Kirchhoff summation as illustrated in Fig. A.6. In the rest of the paper, the associative cluster of Fig. Here an important comment is give concerning the simulation of circuits including neuron MOS' (neuMOS'). Thermal equilibrium condition is assumed for the floating gate charge of a neuMOS, i.e., the net charge on the floating gate is zero when all other terminals are grounded. Such condition is usually achieved after device fabrication, or if hot electron injection occurs during circuit operation, injected charges can be removed by UV irradiation and thermal equilibrium is restored (82, 83). In the HSPICE simulation, the floating gate charge initialization is

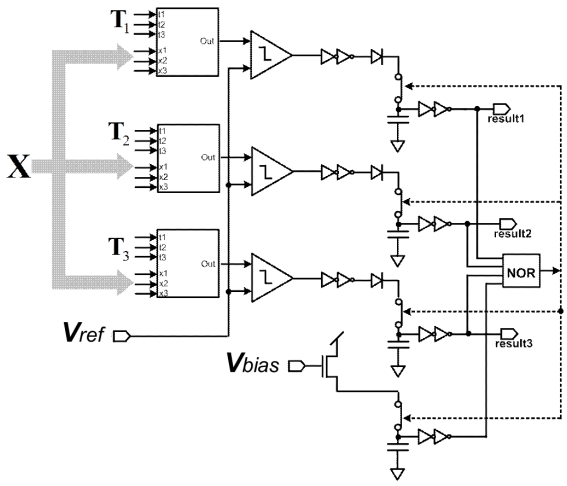


Figure A.7: Associative memory composed of three associative clusters and WTA circuitry.

carried out by circuit operation as follows. Floating nodes and input terminals are all equipped with switching transistors and they are all grounded firstly. Then the floating nodes are disconnected from ground with all other inputs being grounded. After this, the circuits can be operated in normal ways. A.5(a) is represented by a simple symbol of Fig. A.5(b). If such a switched neuron MOS configuration (84) is employed in real CMOS circuit implementation, the subtraction operation yielding the difference between the input and template vector element voltages ($V_x - V_t$ in Fig. A.5(a)) can be directly carried out at the very neuMOS transistor level. The operation is simple. In the initialization step, VCO_N terminal is biased to V_t instead of being grounded and given with V_x in the operation mode. This results in an effective input of $V_x - V_t$, and thus the bulky OP amp circuits can be eliminated. (We need an additional input gate to yield the constant bias V_b.) This means that oscillator-based implementation of associative memories using only CMOS circuits is also feasible as analog VLSI chips.

A.3.3 Associative memory circuit

An associative memory composed of three associative clusters is shown in Fig. A.7. Each output, after going through a comparator and two inverters, drives a peak-detector circuit composed of a diode and a capacitor. Since the time

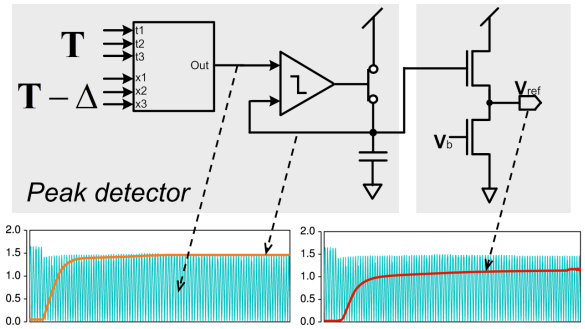


Figure A.8: V_{ref} generator circuit. Simulation results are shown for $\Delta = 0$.

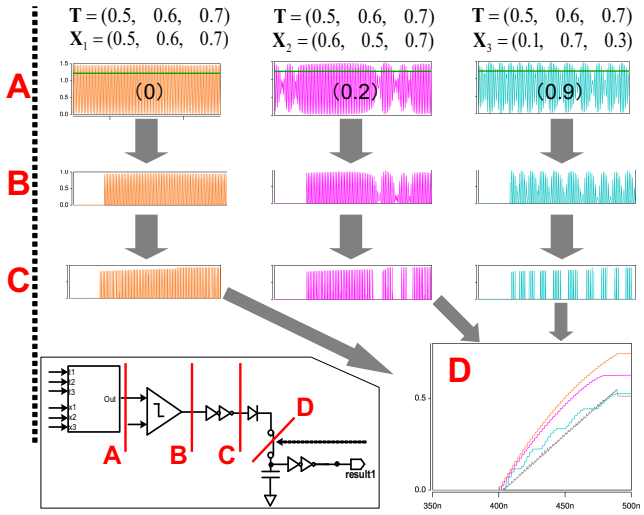


Figure A.9: Oscillation wave forms for three vector matching results.

constant of the peak detector is made much larger than the period of oscillation, it works as a slow integrator. As a result, the voltage across the capacitor increases depending on the oscillation mode. Regular oscillation yields faster voltage rise, while irregular oscillation yields slower voltage rise. When the capacitor voltage reaches the inverting threshold of the following inverter, it turns on and upsets the NOR circuit, which stops the charging of all integrators. In this manner, the winner is identified as the first upsetting inverter, and a flag "1" is set at the location of the winner. The circuit generating V_{ref} for comparators is shown in Fig. A.8. The dummy capacitor charging circuit at the bottom using an NMOS with V_{bias} specifies the lower bounds to the degree of matching. If the charging of oscillators are all slower than this circuit, the dummy is identified as the winner

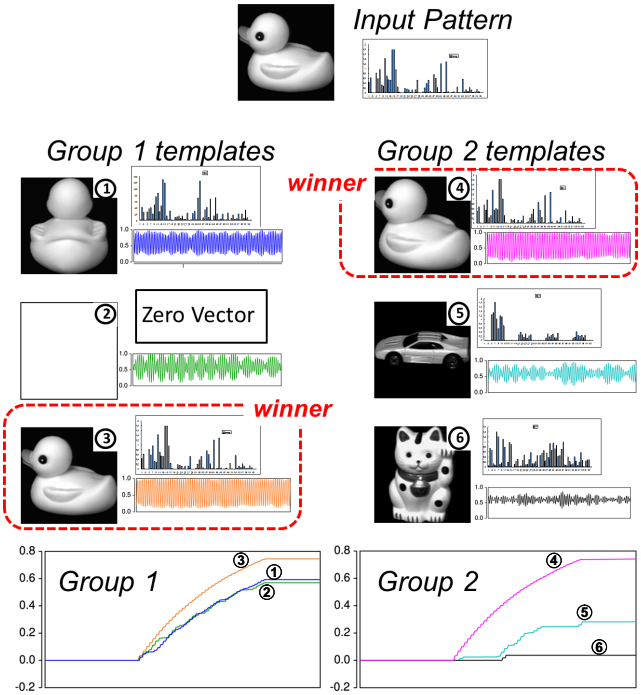


Figure A.10: Matching experiments using COIL-20 database (44) by HSPICE simulation. Matching was carried out for Group 1 and Group 2 templates, separately. APED vectors of images and oscillating signals from associative clusters are also shown.

and "no match" would be the result. Such lower bounds can be varied by V_{bias}.

A.4 SPICE Simulation Experiment

HSPICE simulation was carried out using the device parameters of a typical $1.8\mu m$ CMOS technology. The oscillation wave forms at respective locations in each associative cluster for three different vector matching results are demonstrated in Fig. A.9. Depending on the similarity between the input vector $\mathbb{X}_i$ and the template vector $\mathbb{T}$, the oscillation mode changes and the capacitor voltage increases accordingly. The slowest charging curve in the bottom right figure represents the one from the dummy charging circuit.

The comparator placed in front of each associative cluster plays an important role of cutting out only the upper part of an oscillating wave form, thus enhancing

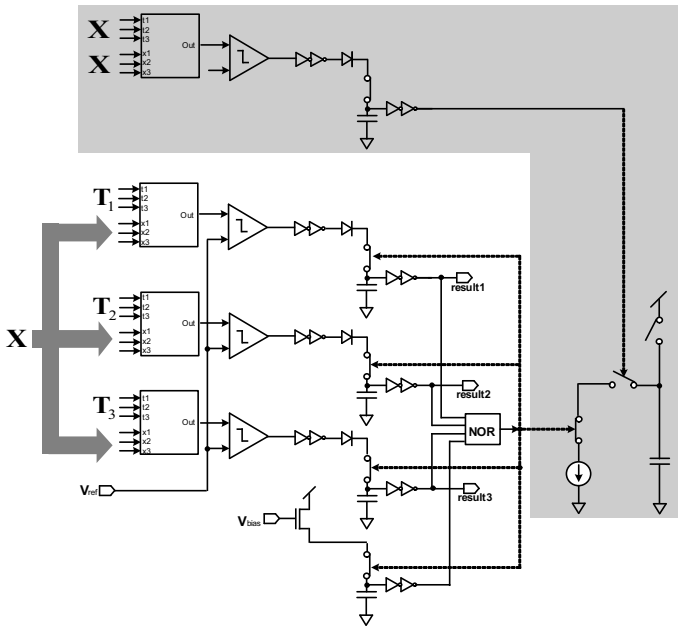


Figure A.11: Timer circuit yielding the degree of matching.

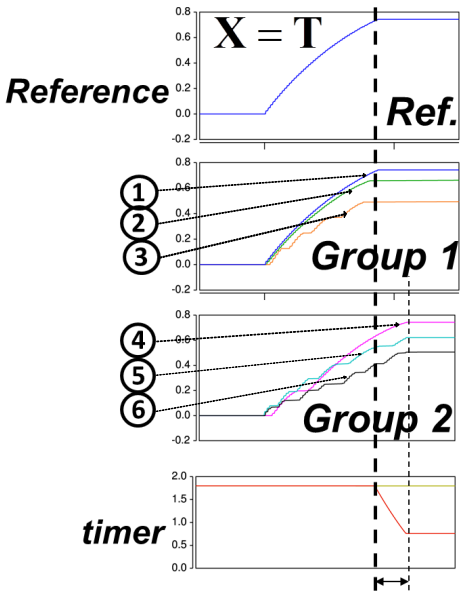


Figure A.12: HSPICE simulation results showing the operation of timer circuit for matching experiments. Patterns used in this simulation are different from those in Fig. A.10.

the difference in the oscillation modes. The cut level is specified by V_{ref} produced by the V_{ref} generator shown in Fig. A.8. The oscillating signal from the dummy associative cluster is rectified by the peak detector and the source follower yields the output V_{ref} with a voltage drop determined by V_b. The V_{ref} level is tunable either by Δ or V_b. The simulation results in the figure was obtained with $\Delta = 0$.

The matching results for images from Columbia Object Image Library (COIL-20) (44) represented by APED (averaged principal edge distribution) vectors (73) are demonstrated in Fig. A.10. Each associative cluster is composed of 64 oscillators to handle 64-dimension vectors. The oscillation mode becomes most stable for the most similar patterns and the winner-take-all circuitry correctly identifies the best match pattern.

The degree of matching value is obtained by activating a timer circuit that measures the time for the winner to be identified as shown in Fig. A.11. The timer is started when the reference associative cluster accepting exact matching patterns upsets the inverter.This yields the maximum degree of matching. Therefore, the capacitor charged up to V_{DD} is started to discharge at this timing, and when the winner is found, the discharging is stopped.

Simulation results are shown in Fig. A.12. For Group 1 templates, the capacitor voltage of the timer is highest because pattern 1 is the same as the input. In Group 2, the capacitor voltage becomes smaller because pattern 4 is similar to the input but not the same. In this manner the degree of matching is measured by the time to win and represented by the capacitor voltage. The scale of the value is easily adjusted by the current source that discharges the capacitor voltage.

A.5 Summary

Using frequency-tunable CMOS ring oscillators to emulate the behavior of nano oscillators, the basic CMOS supporting circuitries to produce associative memory function has been explored and demonstrated by HSPICE simulation. Since the array of nano oscillators and CMOS parts are well segregated in the configuration, full advantage of nano device integration could be exploited.

A. CMOS SUPPORTING CIRCUITRIES FOR NANO-OSCILLATOR-BASED ASSOCIATIVE MEMORIES

References

[1] J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman. **How to Grow a Mind: Statistics, Structure, and Abstraction**. In *J. Science*, **331**: pages 1279–1285, 2011. 1

[2] P. Bloom. **How Children Learn the Meanings of Words**. *MIT Press, Cambridge, MA*, 2000. 1

[3] F. Xu and J. B. Tenenbaum. **Word learning as Bayesian inference**. In *J. Psychol. Rev.* , **114**(2): pages 245–272, 2007. 1

[4] C. M. Bishop. **Pattern Recognition and Machine Learning**. *Springer*, 2006. 1

[5] T. Mitchell. **Machine Learning**. *McGraw-Hill, New York*, 1997. 1

[6] V. N. Vapnik,. **An Overview of Statistical Learning Theory**. In *Neural Network, IEEE Transactions on*, **10** (5): pages 988–999 , 1999. 2

[7] E. Mjolsness and D. DeCoste. **Machine Learning for Science: State of the Art and Future Prospects**. In *J. Science*, **293**: pages 2051–2055, 2001. 1

[8] G. -B. Huang, H. Zhou, X. Ding, and R. Zhang. **Extreme Learning Machine for Regression and Multiclass Classification**. In *System, Man, and Cybernetics Part B: Cybernetics, IEEE Transactions on*, **42** (2): pages 513–529, 2012. 2

[9] C. Rudin, D. Waltz, R. N. Anderson, A. Boulanger, A. Salleb-Aouissi, M. Chow, H. Dutta, P. N. Gross, B. Huang, S. Ierome, D.F. Isaac, A. Kressner, R. J. Passonneau, A. Radeva and L. Wu. **Machine Learning for the New York City Power Grid**. In *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **34** (2): pages 328–345, 2012. 2

[10] W. -Y. Lin, Y. -H. Hu, and C. -F. Tsai. **Machine Learning in Financial Crisis Prediction: A Survey**. In *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, **42** (4): pages 421–436, 2012. 2

[11] M. Narwaria and W. Lin. **SVD-Based Quality Metric for Image and Video Using Machine Learning**. In *System, Man, and Cybernetics Part B: Cybernetics, IEEE Transactions on*, **42** (2): pages 347–364 , 2012. 2

[12] G. M. Edelman. **Learning in and from Brain-Based Devices**. In *J. Science*, **318**: pages 1103–1105, 2012. 2

[13] S. Mitra, S. Fusi, and G. Indiveri. **Real-Time Classification of Complex Patterns Using Spike-Based Learning in Neuromorphic VLS**. In *Biomedical Circuits and Systems, IEEE Transactions on*, **3** (1): pages 32–42 , 2009. 2

[14] J. Misra and I. Saha. **Artificial neural networks in hardware: A survey of two decades of progress**. In *J. Neurocomputing (Elsevier)*, **74**: pages 239–255 , 2010. 2, 87

[15] L. Zhang, Y. Han, H. Li and X. Li. **Fault tolerance mechanism in chip many-core processors**. In *Tsinghua Science and Technology*, **12**: pages 169–174 , 2007. 2

[16] A. Chaudhary, S. Kolhe and R. Kamal. **Machine Learning Techniques for Mobile Intelligent Systems: A Study**. In *Wireless and Optical Communications Networks (WOCN), IEEE Int. Conf. on* , pages 1–5, 2012. 2, 4

[17] J. Madrenas, M. Verleysen, P. Thissen and J. L. Voz. **A CMOS Analog Circuit for Gaussian Functions**. In *Circuits and Systems–II: Analog and Digital Signal Processing, IEEE Transactions on*, **43** (1): pages 70–74, 1996. 7

[18] S. Moshfe, A. Khoei, K. Hadidi and B. Mashoufi. **A fully programmable nano-watt analogue CMOS circuit for Gaussian functions**. In *IEEE Intl Conf. on ICEDSA*, pages 82–87, 2010. 7

[19] T. T. Bui and T. Shibata. **Compact Bell-Shaped Analog Matching-Cell Module for Digital-Memory-Based Associative Processors**. In *Japanese Journal of Applied Physics*, **47** (4): pages 2788–2796, 2008. 7, 91

[20] N. Abramson, D. Braverman, and G. Sebestyen. **Pattern recognition and machine learning** . In *IEEE J. Solid-State Circuits*, **9** (4): pages 257–261, 1963. 2, 14

[21] V. N. Vapnik. **The Nature of Statistical Learning Theory**. *Springer, New York*, 1995. 3, 14

[22] A. M. Mesleh and G. Kanaan. **Support vector machine text classification system: Using Ant Colony Optimization based feature subset selection**. In *Computer Engineering and Systems, IEEE Int. Conf.*, pages 143–148, 2008. 14

[23] B. Goertzel and J. Venuto. **Accurate SVM Text Classification for Highly Skewed Data Using Threshold Tuning and Query-Expansion-Based Feature Selection**. In *Neural Networks, IEEE Int. Joint Conf.*, pages 1220–1225, 2008. 14

[24] Y. Li, S. Xia, and Y. Zhou. **A Supervised Local Linear Embedding Based SVM Text Classification Algorithm**. In *Web Information Systems and Applications, IEEE Conf.*, pages 21–26, 2009. 14

[25] X.-Y. Wang, P.-P. Niu, and H.-Y. Yang. **A Robust, Digital-Audio Watermarking Method**. In *IEEE J. Multimedia*, **16** (3): pages 60–68, 2009. 14

[26] T. Habib, J. Inglada, G. Mercier, and J. Chanussot. **Speeding up Support Vector Machine (SVM) image classification by a kernel series expansion**. In *Image Processing, IEEE Int. Conf.*, pages 865–868, 2008. 14

[27] X. Wang, X. Li, and X. Liu. **Nude image detection based on SVM**. In *Computational Intelligence and Natural Computing, IEEE Int. Conf.*, pages 178–181, 2009. 14

[28] Y. Huang, J. Zhang, Y. Zhao, and D. Ma. **Medical Image Retrieval with Query-Dependent Feature Fusion Based on One-Class SVM**. In *Computational Intelligence and Natural Computing, IEEE Int. Conf.*, pages 176–183, 2010. 14

[29] X. Wang and Y. Zhong. **Statistical Learning Theory and State of the Art in SVM**. In *IEEE Int. Conf. Cognitive Informatics*, pages 55–59, 2003. 3

[30] R. Genov and G. Cauwenberghs. **Kerneltron: support vector "machine" in silicon**. In *Neural Networks, IEEE Transactions on*, **14** (5): pages 1426–1434, 2003. 7, 14

[31] S. Chakrabartty and G. Cauwenberghs. **Sub-Microwatt Analog VLSI Trainable Pattern Classifier**. In *IEEE J. Solid-State Circuits*, **42** (5): pages 1169–1179, 2007. 7, 14

[32] D. Anguita, S. Ridella, and S. Rovetta. **Circuital implementation of support vector machines**. In *IEEE Electron. Lett.*, **34** (16): pages 1596–1597, 1998. 14

[33] Y. Xia and J. Wang. **A One-Layer Recurrent Neural Network for Support Vector Machine Learning**. In *System, Man, and Cybernetics*

Part B: Cybernetics, IEEE Transactions on, **34** (2): pages 1261–1269, 2004. 14

[34] R. Perfetti and E. Ricci. **Analog Neural Network for Support Vector Machine Learning**. In *Neural Networks, IEEE Transactions on*, **17** (4): pages 1085–1091, 2006. 14

[35] B. Schoelkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. **Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers**. In *Signal Process, IEEE Transactions on*, **45** (11): pages 2758–2765, 1997. 14

[36] D. Anguita, A. Boni, and S. Ridella. **A Digital Architecture for Support Vector Machines: Theory, Algorithm, and FPGA Implementation**. In *Neural Networks, IEEE Transactions on*, **14** (5): pages 993–1009, 2003. 8, 14, 18

[37] M. Shi and A. Bermak. **An Efficient Digital VLSI Implementation of Gaussian Mixture Models-Based Classifier**. In *VLSI Systems, IEEE Transactions on*, **14** (9): pages 962–974, 2006. 6, 14

[38] T. Delbruck. **'Bump' circuits for computing similarity and dissimilarity of analog voltages**. In *Neural Networks, IEEE Int. Joint Conf.*, pages 475–479, 1991. 7, 14, 23

[39] M. Ogawa and T. Shibata. **NMOS-based gaussian-element-matching analog associative memory**. In *IEEE Proc. European Conf. Solid-State Circuits*, pages 257–260, 2001. 7, 14, 23

[40] S.-Y. Peng, B. A. Minch, and P. E. Hasler. **Analog VLSI implementation of support vector machine learning and classification**. In *IEEE Proc. Int. Symp. Circuits Syst.*, pages 860–863, 2008. 8, 14, 23, 32, 33

[41] K. Kang and T. Shibata. **An On-Chip-Trainable Gaussian-Kernel Analog Support Vector Machine**. In *Circuits and Systems, IEEE*

Transactions on, **57** (7): pages 1513–1524, 2010. 7, 8, 14, 18, 19, 23, 32, 33, 36, 82

[42] M. Yagi and T. Shibata. **An image representation algorithm compatible with neural-associative-processor-based hardware recognition systems**. In *Neural Networks, IEEE Transactions on*, **14** (5): pages 1144–1161, 2003. 5, 16

[43] J. A. Croonat, M. Rosmeulen, S. Decoutere, W. Sansen, and H. E. Maes. **An easy-to-use mismatch model for the MOS transistor**. In *IEEE J. Solid-State Circuits*, **37** (8): pages 1056–1064, 2002. 27

[44] S. A. Nene, S. K. Nayar and H. Murase. **Columbia Object Image Library (COIL-20)**. Technical Report CUCS-005-96. x, 28, 37, 60, 97, 99

[45] J. MacQueen. **Some methods for classification and analysis of multivariate observations**. In *Proc. 5th Berkeley Symp. Math. Stat. Probab.*, pages 281–297, 1967. 36

[46] S. Ray and R. H. Turi. **Determination of number of clusters in K-means clustering and application in colour image segmentation**. In *Proc. 4th Int. Conf. Adv. Pattern Recog. Digit. Techn.*, pages 137–143, 1991. 2, 36

[47] S. Gordon, H. Greenspan, and J. Goldberger. **Applying the information bottleneck principle to unsupervised clustering of discrete and continuous image representations**. In *Proc. 9th Int. Conf. Computer Vision*, pages 370–377, 2003. 36

[48] B. Maliatski and O. Yadid-Pecht. **Hardware-driven adaptive K-means clustering for real-time video imaging**. In *Circuits Syst. Video Technol., IEEE Transactions on*, **15** (1): pages 164–166, 2005. 36

[49] T.-W. Chen and S.-Y. Chien. **Bandwidth adaptive hardware architecture of K-means clustering for video analysis**. In *Very Large*

Scale Integr. (VLSI) Syst., IEEE Transactions on, **18** (6): pages 957–966, 2010. 36

[50] T.-W. Chen, C.-H. Sun, J.-Y. Bai, H.-R. Chen, and S.-Y. Chien. **Architectural analyses of K-means silicon intellectual property for image segmentation**. In *Proc. IEEE Int. Symp. Circuits Syst.*, pages 2578–2581, 2008. 36

[51] T. Maruyama. **Real-time K-means clustering for color images on reconfigurable hardware**. In *Proc. Int. Conf. Pattern Recog.*, pages 816–819, 2006. 36

[52] I. Chiosa, A. Kolb. **GPU-Based multilevel clustering**. In *Visualization and Computer Graphics, IEEE Transactions on*, **17**: pages 132–145, 2011. 36

[53] Y. Ma and T. Shibata. **A binary-tree hierarchical multiple-Chip architecture for real-time large-scale learning processor systems**. In *Japanese Journal of Applied Physics*, **49**: pages 04DE08, 2010.

[54] B. A. Minch. **MOS translinear principle for all inversion levels**. In *Circuits and Systems, IEEE Transactions on*, **55**: pages 121–125, 2008. viii, 46

[55] X. Wang and M. Leeser. **K-means Clustering for Multispectral Images Using Floating-Point Divide**. In *IEEE 15th Int. Symp. Field-Programmable Custom Computing Machines*, pages 151–162, 2007. 51

[56] J. A. Croon, M. Rosmeulen, S. Decoutere, W. Sansen and H.E. Maes. **A Simple Characterization Method for MOS Transistor Matching in Deep Submicron Technologies**. In *IEEE Proc. Int. Conf. Microelectronic Test Structures*, **55**: pages 213–218, 2001. 48

[57] G. Cauwenberghs and T. Poggio. **Incremental and Decremental Support Vector Machine Learning**. In *Advances in Neural Information Processing Systems*, 2001. 54, 55

[58] L. Matthews, T. Ishikawa and S. Baker. **The Template Update Problem**. In *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **26**: pages 810–815 2004. 54, 55

[59] M. Tian, W. Zhang and F. Liu. **On-Line Ensemble SVM for Robust Object Tracking**. In *Yagi, Y., Kang, S., Kweon, I., Zha, H. (eds.) ACCV. LNCS*, **4843**: pages 355–364, Springer, Heidelberg, 2007. 54

[60] F. Tang, S. Brennan, Q. Zhao and H. Tao. **Co-Tracking Using Semi-Supervised Support Vector Machines**. In *IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007. 54

[61] H. M. Hussain, K. Benkrid, H. Seker and A. T. Erdogan. **FPGA Implementation of K-means Algorithm for Bioinformatics Application: An Accelerated Approach to Clustering Microarray Data**. In *NASA/ESA Conf. Adaptive Hardware and Systems*, pages 246–255, 2011. 65

[62] T.-W. Chen and S.-Y. Chien. **Flexible Hardware Architecture of Hierarchical K-Means Clustering for Large Cluster Number**. In *Very Large Scale Integr. (VLSI) Syst., IEEE Transactions on*, **11**(8): pages 1336–1345, 2011. 65

[63] C. Wang., J. Lai, J. Zhu. **A Conscience On-line Learning Approach for Kernel-Based Clustering**. In *IEEE Conf. Data Mining*, pages 531–540, 2010. 60

[64] T.-W. Chen, C.-H Sun, H.-H. Su, S.-Y. Chien, D. Deguchi, I. Ide, and H. Murase. **Power-Efficient Hardware Architecture of K-Means Clustering With Bayesian-Information-Criterion Processor for Multimedia Processing Applications**. In *IEEE J. Emerging and Selected Topics in Circuits and Systems*, **1**(3): pages 357–368, 2011. 6, 66

[65] Y. Wang. **A Tree-based Multi-class SVM Classifier for Digital Library Document**. In *Proc. IEEE Int. Conf. MultiMedia and Information Technology*, pages 15–18, 2008. 70

[66] D. M. J. Tax, and R. P. W. Duin. **Data Domain Description Using Support Vectors**. In *Proc. European Symposium on Artificial Neural Networks*, pages 251–256, 1999. 70, 73

[67] D. M. J. Tax, and R. P. W. Duin. **Support Vector Domain Description**. In *Pattern Recognition Letters*, **20**: pages 1191–1199, 1999. 2, 4, 70

[68] H. Wang, G. Zhao, and H. Gu. **A fast training method for OC-SVM based on the random sampling lemma**. In *Proc. IEEE Int. Conf. Natural Computation*, pages 824–827, 2010. 70, 82

[69] J. Xu, J. Yao, and L. Ni. **Fault Detection Based on SVDD and Cluster Algorithm**. In *Proc. IEEE Int. Conf. Electronics, Communications and Control*, pages 2050–2052, 2011. 70

[70] X. Huang, and X. Chen. **A Novel Clustering Algorithm Based on One-Class SVM**. In *Proc. IEEE Global Congress on Intelligent Systems*, pages 486–490, 2009. 70

[71] H. Galmeanu and R. Andonie. **Incremental / decremental SVM for function approximation**. In *Proc. IEEE Int. Conf. Optimization of Electrical and Electronic Equipment*, pages 155–160, 2008. 70

[72] T. Yamasaki and T. Shibata. **An Analog Similarity Evaluation Circuit Featuring Variable Functional Forms**. In *Proc. IEEE Int. Symp. Circuits and Systems*, pages III-561–564, 2001. 89

[73] T. Shibata. **Computing based on the physics of nano devices – A beyond-CMOS approach to human-like intelligent systems**. In *Solid-State Electronics*, **53**: pages 1227–1241, 2009. 89, 99

[74] M. Saitoh, H. Harata, and T. Hiramoto. **Room-Temperature Demonstration of Integrated Silicon Single-Electron Transistor Circuits for Current Switching and Analog Pattern Matching**. In *International Electron Devices Meeting (IEDM)*, pages 187–190, 2004. 90

[75] T. Roska and Á. Rodriguez-Vázquez. **Towards Visual Microprocessors**. In *Proceedings of the IEEE*, **90**: pages 1244–1257, 2002. 90

[76] M. R. Pufall, W. H. Rippard, S. E. Russek, S. Kaka, and J. A. Katine. **Electrical Measurement of Spin-Wave Interactions of Proximate Spin Transfer Nanooscillators**. In *Phys. Rev. Letters*, **97**: pages 087206, 2006. 90

[77] Dana Weinstein, and Sunil A. Bhave. **The Resonant Body Transistor**. In *Nano Lett.*, pages 1234–1237, 2010. 90

[78] A. Nakada, T. Shibata, M. Konda, T. Morimoto, and T. Ohmi. **A Fully-Parallel Vector Quantization Processor for Real-Time Motion Picture Compression**. In *IEEE Journal of Solid-State Circuits*, **34**(6): pages 822–830, 1999. 91

[79] Dmitri E. Nikonov, Gyorgy Csaba, Wolfgang Porod, Tadashi Shibata, Dan Hammerstrom, and George Bourianoff. **Coupled-oscillator associative memory array operation**. *to be submitted to IEEE Nano Transactions special issue.* 91

[80] Frank C. Hoppensteadt and Eugene M. Izhikevich. **Oscillatory Neurocomputers with Dynamic Connectivity**. In *Physical Review Letters*, **82**(14): pages 2983–2986, 1999. 91

[81] György Csaba, Matt Pufall, Dmitri Nikonov, George Bourianoff, Andras Horvath, Tamas Roska, and Wolfgang Porod. **Spin Torque Oscillator (STO) Models for Applications in Associative Memories**. In *IEEE Int. Conf. Cellular Nano-scale Networks*, 2012. 92

[82] Tadashi Shibata and Tadahiro Ohmi. **A functional MOS transistor featuring gate-level weighted sum and threshold operations**. In *Electron Devices IEEE Transactions on*, **39**(6): pages 1444–1455, 1992. 93, 94

[83] A. Luck, S. Jung, R. Brederlow, R. Thewes, K. Goser, and W. Weber. **On the design robustness of threshold logic gates using multi-input floating gate MOS transistors**. In *Electron Devices IEEE Transactions on*, **47**(6): pages 1231–1240, 2000. 94

[84] K. Kotani, T. Shibata, M. Imai, T. Ohmi. **Clocked-neuron-MOS logic circuits employing auto-threshold-adjustment**. In *Digest of Technical papers, IEEE International Solid-State Circuits conference*, pages 320–321, 1995. 95

List of Publications

Journal Papers:

[1] Renyuan Zhang and Tadashi Shibata, “Fully Parallel Self-Learning Analog Support Vector Machine Employing Compact Gaussian-Generation Circuits”, Jpn. J. Appl. Phys., vol. 51, no. 4, pp. 04DE10-1 - 04DE10-7. (2012)

[2] Renyuan Zhang and Tadashi Shibata, “An Analog On-Line-Learning K-means Processor Employing Fully Parallel Self-Converging Circuitry”, J. Analog Integrated Circuits and Signal Processing (Springer), vol. 75, no. 2, pp.267-277.(2013)

Refereed papers at International Conferences:

[1] Renyuan Zhang and Tadashi Shibata, “A Fully-Parallel Self-Learning Analog Support Vector Machine Employing Compact Gaussian-Generation Circuits”, Int. Conf. Solid-State Device and Materials, 2011, pp. 174-175.

[2] Renyuan Zhang and Tadashi Shibata, “An Analog K-means Learning Processor Employing Fully-Parallel Self-Converging Circuitry”, Int. Analog VLSI Workshop, 2011, pp. 91-96.

[3] Pushe Zhao, Renyuan Zhang, and Tadashi Shibata, “Real-time Visual Tracking Algorithm Employing On-Line Support Vector Machine and Multiple Candidate Regeneration,” L. Rutkowski et al. (Eds.): Lecture Notes in Computing Science (Proc. ICAISC2012), vol. 7267, Part I, pp. 617-625. Springer, Heidelberg ISBN: 978-3- 642-29346-7.

[4] Renyuan Zhang and Tadashi Shibata, “Real-Time On-Line-Learning Support Vector Machine Based on A Fully-Parallel Analog VLSI processor,” L. Rutkowski et al. (Eds.): Lecture Notes in Computing Science (Proc. ICAISC2012), vol.7268, Part II, pp.223-230. Springer, Heidelberg ISBN: 978-3-642-29349-8.

[5] Tadashi Shibata, Hongbo Zhu, Ruihan Bao, Pushe Zhao, and Renyuan Zhang, “A VLSI System for Motion Perception and Action Recognition,” in the Proceedings of the 2nd Solid-State Systems Symposium (4S 2012), Ho Chi Minh City, Vietnam, August 22-24, 2012.

[6] Tadashi Shibata, Renyuan Zhang, Steven P. Levitan, Dmitri Nikonov, and George Bourianoff, “CMOS Supporting Circuitries for Nano-Oscillator-Based Associative Memories”, invited by IEEE Int. Workshop on Cellular Nanoscale Networks and their Applications, Turin, Italy, Aug. 29-31, 2012.

[7] Renyuan Zhang and Tadashi 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, Turin, Italy, Aug. 29-31, 2012.