Massively Parallel Mixed-Signal VLSI Kernel Machines

Roman Genov

A dissertation submitted to the Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

May, 2003

Abstract

Recently it has been shown that a simple learning paradigm, the support vector machine (SVM), outperforms some of the most elaborately tuned expert systems and neural networks in object recognition tasks. In run-time, the SVM operates by computing a kernel-based distance between the object's vector at the input and a set of support vectors selected from the training set, and weighting the results to produce the oracle at the output. Real-time SVM recognition of complex objects in streaming video incurs an excessive amount of computation, well beyond even the most powerful digital signal processors available today. This calls for a radically different computational paradigm to efficiently compute kernels in very large dimensions.

I present a massively parallel, fine-grain distributed architecture for real-time kernel "machines" and its efficient implementation in mixed-signal VLSI technology. At the core of the externally digital architecture is a high-density, low-power analog array performing binary-binary matrix-vector multiplication, as the elementary operation in computing inner-product based kernels between presented input and stored support vectors. The three-transistor unit cell in the analog array combines a charge injection device (CID) binary multiplier and analog accumulator with embedded dynamic random-access memory (DRAM). I present various schemes to obtain precise digital results from the internal analog computation in a distributed, parallel fashion, using analog-to-digital quantization of partial binary-binary products computed over the array. High output resolution is achieved with low complexity quantizers by oversampling in the input binary representation combined with delta-sigma modulated quantization at the output. In addition, stochastic encoding of the digital inputs relaxes the precision requirements of the quantizers by the square root of the vector dimension owing to the Central Limit in the accumulation of binary terms in the inner-product.

My dissertation research has resulted in the Kerneltron, the first support vector "machine" in silicon. A 3mm by 3mm 0.5 micron CMOS chip features 256 inputs and 128 support vectors, delivering over 1 trillion (10^{12}) multiply-accumulates-per-second for every Watt of power. An integrated bank of 128 delta-sigma modulated algorithmic analog-to-digital converters produce for each output 8 bits of resolution in 32 cycles. Applications of the Kerneltron include artificial vision, automated surveillance, and human-computer interfaces.

Acknowledgements

With the utmost gratitude I thank my advisor Gert Cauwenberghs for his loyal support and enduring guidance over the years. He has been the most intellectually stimulating, high-principled and optimistic mentor.

My deepest thanks to professors Andreas Andreou and Ralph Etienne-Cummings for their motivating engineering lessons. Their insightful teaching has greatly expanded my interest in and enriched my knowledge of mixed-signal VLSI design. I express my appreciation to professors Tomaso Poggio and Dario Floreano. Their support has enabled me to gain valuable experience while at MIT and EPFL.

I am very thankful to my colleagues and friends Marc Cohen, Tim Edwards and Philippe Pouliquen for their advice and leadership; Kai He, Shantanu Chakrabartty, Milutin Stanacevic for being dependable and supportive; Srinadh Madhavapeddi, Grant Mulliken, Farhan Adil for their input and enthusiasm. Portions of this dissertation have been made possible with their assistance and backing.

Finally, I am immensely grateful to my mother for her prudent unconditional support and my late father who inspired my interest in science and taught me engineering.

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Pattern Recognition with Kernel Machines

1.1 Introduction

Support vector machines (SVM) [1] offer a principled approach to machine learning combining many of the advantages of artificial intelligence and neural network approaches. Underlying the success of SVMs are mathematical foundations of statistical learning theory [2]. Rather than minimizing training error (*empirical risk*), SVMs minimize *structural risk* which expresses an upper bound on the generalization error, *i.e.,* the probability of erroneous classification on yet-to-be-seen examples. This makes SVMs especially suited for adaptive object detection and identification with sparse training data.

Real-time detection and identification of visual objects in video from examples is generally considered a hard problem for two reasons. One is the large degree of variability in the object class, *i.e.,* orientation and illumination of the object or occlusions and background clutter in the surrounding, which usually necessitates a large number of training examples to generalize properly. The other is the excessive amount of computation incurred during training, and even in run-time.

Support vector machines have been applied to visual object detection, with demonstrated success in face and pedestrian detection tasks [3, 4, 5, 6]. Unlike approaches to object detection that rely heavily on hand-crafted models and motion information, SVM-based systems learn the model of the object of interest from examples and work reliably in absence of motion cues. To reduce the computational burden of real-time implementation to a level that can be accommodated with available hardware, a reduced set of features are selected from the data which also result in a reduced number of support vectors [5]. The reduction in implementation necessarily comes at a loss in classification performance, a loss which is more severe for tasks of greater complexity.

The run-time computational load is dominated by evaluation of a kernel between the incoming vector and each of the support vectors. For a large class of permissible kernels, which include polynomial splines and radial kernels, this computation entails matrix-vector multiplication in large dimensions. For the pedestrian detection task in unconstrained environments [5], highest detection at lowest false alarm is achieved for very large numbers (thousands) of input dimensions and support vectors, incurring millions of matrix multiply-accumulates (MAC) for each classification. The computation recurs at different positions and scales across each video frame.

The *Kerneltron*, a massively parallel mixed-signal VLSI kernel machine introduced in this chapter, offers a factor 100-10,000 improvement in computational efficiency (throughput per unit power) over the most advanced digital signal processors available today. It affords this level of efficiency at the expense of specificity: the VLSI architecture is dedicated to massively parallel kernel computation. Speed can be traded for power dissipation. Lower power is attractive in portable applications of kernel-based pattern recognition, such as visual aids for the blind [7].

The rest of the chapter is organized as follows. Section 1.2 provides background information about feature extraction and SVM classification for object detection in streaming video. Section 1.3 argues for applicability of the approach taken. Section 1.4 describes the architecture and circuit implementation of the *Kerneltron*. The organization of the remainder of the dissertation is outlined in Section 1.5.

1.2 Background: Object Detection with Support Vector Machines

A support vector machine is trained with a data set of labeled examples. For pattern classification in images, relevant features are typically extracted from the training set examples using redundant spatial filtering techniques, such as an overcomplete wavelet decomposition [4]. The classifier is trained on these feature vectors. In run time, images representing frames of streaming video are scanned by moving windows of different dimensions. For every unit shift of a moving window, a wavelet feature vector is computed and presented to the SVM classifier to produce a decision. The general block diagram of such a system is outlined in Figure 1.1. A brief functional description of the major components follows next.

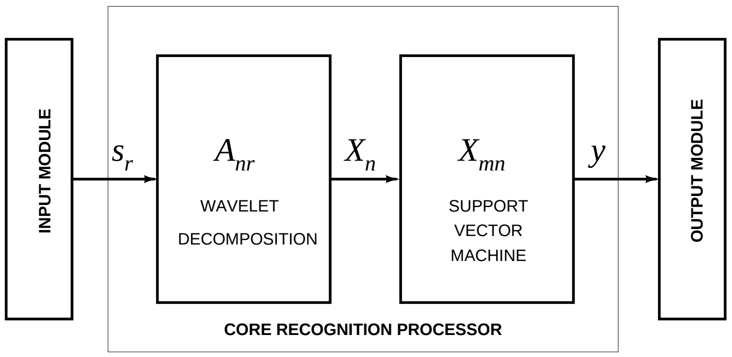


Figure 1.1: Functional block diagram of the SVM classifier. The core of the system is a support vector machine processor for general object detection and classification. An overcomplete wavelet decomposition of the incoming sensory data at the input generates redundant input features to the SVM, providing for robust and relatively invariant classification performance.

1.2.1 Overcomplete Wavelet Decomposition

An overcomplete wavelet basis enables the system to handle complex shapes and achieve a precise description of the object class at adequate spatial resolution for detection [4]. The transformation of the sensory data $\mathbf{s}$ into the feature vector $\mathbf{X}$ is of the linear form

$$X_n = \sum_{r=1}^{R} A_{nr}\, s_r, \quad n = 1, \cdots, N, \tag{1.1}$$

where the wavelet coefficients A_{nr} form an overcomplete basis, *i.e.*, $N > R$.

In visual object detection overcomplete Haar wavelets have been successfully used on pedestrian and face detection tasks [4, 5]. Haar wavelets are attractive because they are robust and particularly simple to compute, with coefficients A_{nr} that are either -1 or 1.

1.2.2 Support Vector Classification

Classification of the wavelet transformed features is performed by a support vector machine (SVM) [1]. From a machine learning theoretical perspective [2], the appealing characteristics of SVMs are:

1. The learning technique generalizes well even with relatively few data points in the training set, and bounds on the generalization error can be directly estimated from the training data.

2. The only parameter that needs tuning is a penalty term for misclassification which acts as a regularizer [8] and determines a trade-off between resolution and generalization performance [9].

3. The algorithm finds, under general conditions, a unique separating decision surface that maximizes the margin of the classified training data for best out-of-sample performance.

SVMs express the classification or regression output in terms of a linear combination of examples in the training data, in which only a fraction of the data points, called "support vectors," have non-zero coefficients. The support vectors thus capture all the relevant data contained in the training set. In its basic form, a SVM classifies a pattern vector $\mathbf{X}$ into class $y \in \{-1, +1\}$ based on the support vectors $\mathbf{X}_m$ and corresponding classes y_m as

$$y = \text{sign}(\sum_{m=1}^{M} \alpha_m \, y_m \, K(\mathbf{X}_m, \mathbf{X}) - b), \tag{1.2}$$

where $K(\cdot, \cdot)$ is a symmetric positive-definite kernel function which can be freely chosen subject to fairly mild constraints [1]. The parameters α_m and b are determined by a linearly constrained quadratic programming (QP) problem [2, 10], which can be efficiently implemented by means of a sequence of smaller scale, subproblem optimizations [3], or an incremental scheme that adjusts the solution one training point at a time [11]. Most of the training data $\mathbf{X}_m$ have zero coefficients α_m; the non-zero coefficients returned by the constrained QP optimization define the support vector set. In what follows we assume that the set of support vectors and coefficients α_m are given, and we concentrate on efficient run-time implementation of the classifier.

1.2.3 Kernel Machines

Support vector machines belong to a broader class of kernel machines. Kernel machines use a nonlinear kernel function to map the data space into a higher-dimensional feature space. In the case of classification the goal is to make the data more linearly separable in the feature space (Figure 1.2). Kernels extend any inner-product-based linear classifier to the non-linear case. A kernel on a data space satisfying Mercer condition [2, 8] implicitly computes inner-products in the feature space:

$$K(\mathbf{X}_m, \mathbf{X}) = \Phi(\mathbf{X}_m) \cdot \Phi(\mathbf{X}), \tag{1.3}$$

where $\Phi(\cdot)$ is a nonlinear map from the data space into the feature space.

Several widely used classifier architectures reduce to special valid forms of kernels $K(\cdot, \cdot)$ in (1.2), like polynomial classifiers, multilayer perceptrons[1], and radial basis functions [12]. The following forms are frequently used:

[1] with logistic sigmoidal activation function, for particular values of the threshold parameter only

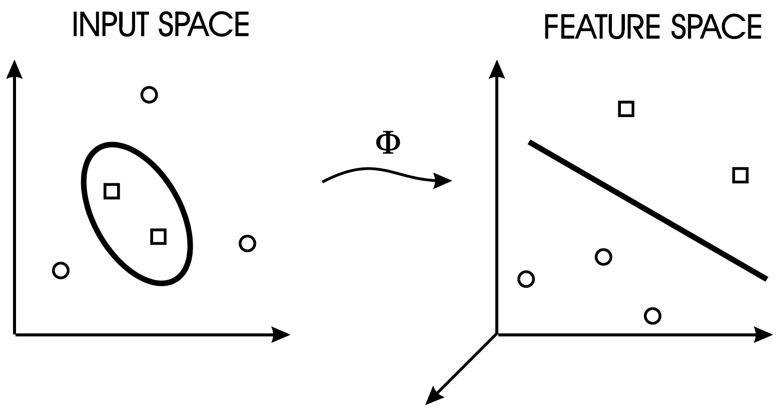


Figure 1.2: Kernel machines use a nonlinear kernel function to map the data space to a higher-dimensional feature space.

1. Inner-product based kernels (*e.g.,* polynomial; sigmoidal connectionist):

$$K(\mathbf{X}_m, \mathbf{X}) = f(\mathbf{X}_m \cdot \mathbf{X}) = f(\sum_{n=1}^{N} X_{mn}\, X_n) \tag{1.4}$$

2. Radial basis functions (L_2 norm distance based):

$$K(\mathbf{X}_m, \mathbf{X}) = f(\|\mathbf{X}_m - \mathbf{X}\|) = f((\sum_{n=1}^{N} |X_{mn} - X_n|^2)^{\frac{1}{2}}), \tag{1.5}$$

where $f(\cdot)$ is a monotonically non-decreasing scalar function subject to the Mercer condition on $K(\cdot, \cdot)$ [2, 8].

With no loss of generality, we concentrate on kernels of the inner-product type (1.4), and devise an efficient scheme of computing a large number of high-dimensional inner-products in parallel. Computationally, the inner-products comprise the most intensive part in evaluating kernels of both types (1.4) and (1.5). Indeed, radial basis functions (1.5) can be expressed in inner-product form:

$$f(\|\mathbf{X}_m - \mathbf{X}\|) = f((-2\mathbf{X}_m \cdot \mathbf{X} + \|\mathbf{X}_m\|^2 + \|\mathbf{X}\|^2)^{\frac{1}{2}}), \tag{1.6}$$

where the last two terms depend only on either the input vector or the support vector. These common terms are of much lower complexity than the inner-products, and can be easily pre-computed or stored in peripheral registers.

1.3 Relevance: Real-Time Object Detection in Video

The computation of inner-products in parallel takes the form of matrix-vector multiplication (MVM), $\sum_{n=1}^{N} X_{mn}\, X_n$; $m = 1, \ldots M$, where M is the number of support vectors. For large-scale problems as the ones of interest here, the dimensions of the matrix $M \times N$ are excessive for real-time implementation of kernel machines even on a high-end processor. As a point of reference, consider the pedestrian and face detection task in [5], for which the feature vector length N is 1,326 wavelets per instance, and the number of support vectors M is in excess of 4,000. To cover the visual field over the entire scanned image at reasonable resolution (500 image window instances through a variable resolution search method) at video rate (30 frames per second), a computational throughput of 75×10^9 multiply-and-accumulate operations per second, is needed. The computational requirement can be relaxed through simplifying and further optimizing the SVM architecture for real-time operation, but at the expense of classification performance [4, 5].

Conventional general-purpose processors and DSPs lack parallelism and memory bandwidth needed for efficient real-time implementation [13, 14, 15]. Multiprocessors and networked parallel computers in principle are capable of high throughput, but are costly, and impractical for embedded real-time applications. Dedicated parallel VLSI architectures have been developed to speed up MVM computation, *e.g.,* [16]. The problem with most parallel systems is that they require centralized memory resources, *i.e.,*, RAM shared on a bus, thereby limiting the available throughput. A fine-grain, fully-parallel architecture, that integrates memory and processing elements, yields high computational throughput and high density of integration [17, 18, 19]. The ideal scenario (in the case of matrix-vector multiplication) is where each processor performs one multiply and locally stores one coefficient, with a throughput scaling linearly with the dimensions of the implemented array. This is the approach taken in the design of the *Kerneltron*, a massively parallel mixed-signal VLSI kernel machine, introduced next.

1.4 Approach: The Kerneltron — Massively Parallel VLSI Kernel Machine

The *Kerneltron* offers the computational power required for the unabridged SVM architecture to run in real time, for optimal out-of-sample classification performance, and energy efficiency needed for implementations on portable platforms. Its fine-grain massively parallel architecture is described next.

1.4.1 Core Recognition VLSI Processor

At the core of the system is a recognition engine, which very efficiently implements kernel-based algorithms, such as support vector machines, for general pattern detection and classification. The implementation focuses on inner-product computation in a parallel architecture.

Both wavelet and SVM computations are most efficiently implemented on the same chip, in a scalable VLSI architecture as illustrated schematically in Figure 1.3. The diagram is the floorplan of the *Kerneltron*, with matrices projected as 2-D arrays of cells, and input and output vector components crossing in perpendicular directions alternating from one stage to the next. This style of scalable architecture also supports the integration of learning functions, through local outer product parameter updates [20], compatible with the recently developed incremental SVM learning rule [11]. The architecture maintains low input/output data rate. Digital inputs are fed into the processor through a properly sized serial/parallel converter shift register. A unit shift of a scanning moving window in an image corresponds to one shift of a new pixel per classification cycle, while a single scalar decision is produced at the output.

The classification decision is obtained in digital domain by thresholding the weighted sum of kernels. The kernels are obtained by mapping the inner-products $\mathbf{X} \cdot \mathbf{X}_m$ through the function $f(\cdot)$ stored in a look-up table.

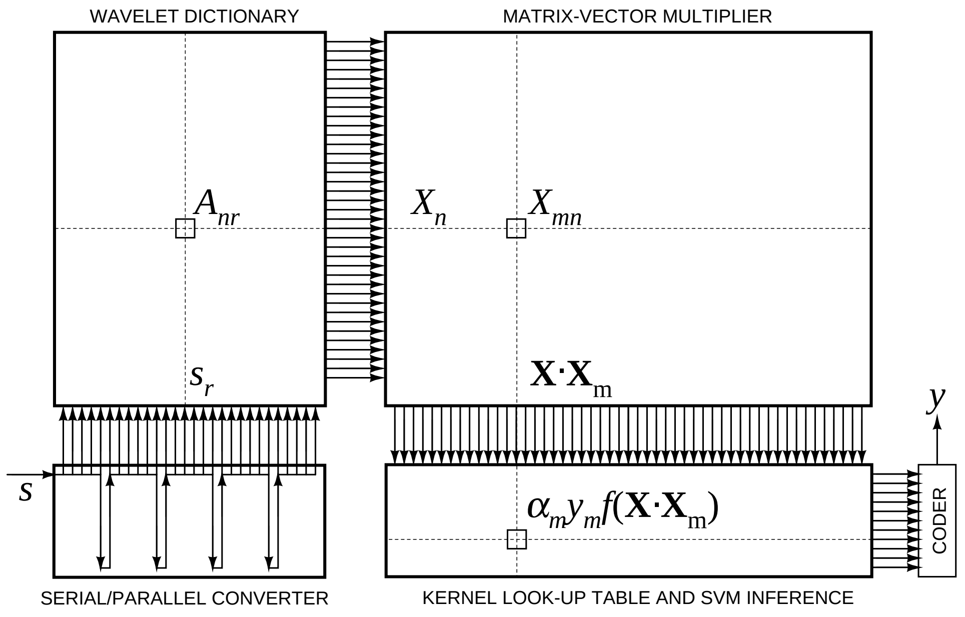

WAVELET DICTIONARY
MATRIX-VECTOR MULTIPLIER
A_{nr}
X_n
X_{mn}
s_r
$\mathbf{X}\cdot\mathbf{X}_\mathrm{m}$
y
s
$\alpha_m y_m f(\mathbf{X}\cdot\mathbf{X}_\mathrm{m})$
CODER
SERIAL/PARALLEL CONVERTER
KERNEL LOOK-UP TABLE AND SVM INFERENCE

Figure 1.3: The architecture of the core recognition processor, combining overcomplete wavelet decomposition with generalized support vector machine classification. Communication with outside modules is through a serial digital input/output interface for maximal flexibility and programmability, while the core internal computations are parallel and analog for optimal efficiency.

By virtue of the inner-product form of the kernel, the computation can be much simplified without affecting the result. Since both the wavelet feature extraction and the inner-product computation represent linear transformations, they can be collapsed into a single linear transformation by multiplying the two matrices:

$$W_{mr} = \sum_{n=1}^{N} X_{mn} A_{nr}. \tag{1.7}$$

Therefore the architecture can be simplified to one that omits the (explicit) wavelet transformation, and instead transforms the support vectors.[2] For simplicity of the argument, we proceed with the inner-product architecture excluding the overcomplete wavelet feature extraction stage, bearing in mind that the approach extends to include wavelet extraction by merging the two matrices.

[2]Referred to the input prior to wavelet transformation, support vectors $\mathbf{s}_m$ need to be transformed *twice*: $W_{mr} = \sum_{n=1}^{N} \sum_{p=1}^{S} A_{np} A_{nr} s_{mp}$.

Computing inner-products between a high-dimensional input vector $\mathbf{X}$ and a large number of template vectors $\mathbf{W}_m$ in parallel is equivalent to the operation of matrix-vector multiplication (MVM) in large dimensions:

$$Y_m = \sum_{n=0}^{N-1} W_{mn} X_n, \tag{1.8}$$

with N-dimensional input vector X_n, M-dimensional output vector Y_m, and $M \times N$ matrix of coefficients W_{mn}. The matrix elements W_{mn} denote the support vectors X_{mn}, or the wavelet transformed support vectors (1.7) for convenience of notation.[3]

1.4.2 Choice of Technology

The recurring problem with digital implementation of kernel machines is the latency in accumulating the result over a large number of cells. Also, the extensive silicon area and power dissipation of a digital multiply-and-accumulate implementation make this approach prohibitive for very large (1,000-10,000) matrix dimensions.

Analog VLSI provides a natural medium to implement fully parallel computational arrays with high integration density and energy efficiency [21]. By summing charge or current on a single wire across cells in the array, low latency is intrinsic. Analog multiply-and-accumulate circuits are so small that one can be provided for each matrix element, making it feasible to implement massively parallel implementations with large matrix dimensions. Fully parallel implementation of (1.8) requires an $M \times N$ array of cells, illustrated in Figure 1.4. Each cell (m, n) computes the product of input component X_n and matrix element W_{mn}, and dumps the resulting current or charge on a horizontal output summing line. The device storing W_{mn} is usually incorporated into the computational cell to avoid performance limitations due to low external memory access bandwidth. Various physical representations of inputs and matrix elements have been explored, using synchronous charge-mode [22, 23, 24, 25], asynchronous transconductance-mode [26, 27, 28], or asynchronous current-mode [29] multiply-and-accumulate circuits.

[3] In the wavelet transformed case, $\mathbf{s}$ should be substituted for $\mathbf{X}$ in what follows.

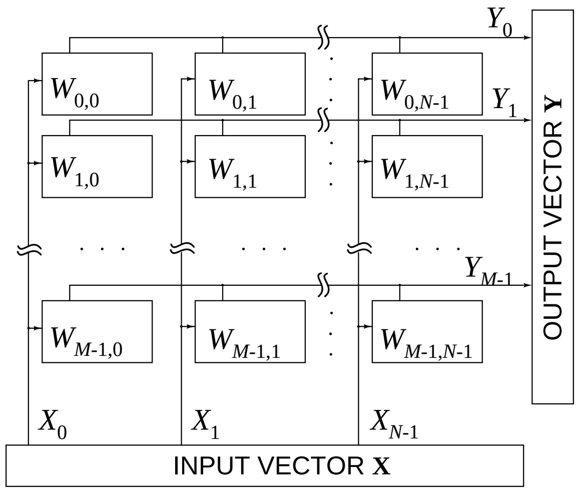


Figure 1.4: General architecture for fully parallel matrix-vector multiplication (MVM).

The main problem with purely analog implementation is the effect of noise and component mismatch on precision. To this end, we propose the use of hybrid analog-digital technology to simultaneously add a large number of digital values in parallel, with careful consideration of sources of imprecision in the implementation and their overall effect on the system performance. Furthermore, we introduce algorithms achieving full digital resolution of mixed-signal VLSI computation. Our approach combines the computational efficiency of analog array processing with the precision of digital processing and the convenience of a programmable and reconfigurable digital interface.

1.5 Organization

The remainder of the dissertation describes our contributions in detail, and is organized by Part as follows:

Part II Kerneltron I: Massively Parallel VLSI Kernel Machine describes a massively parallel processor for matrix-vector multiplication combining the efficiency of analog computation with the precision of digital processing. **Chapter 2** introduces a massively parallel charge-mode analog array architecture performing digital matrix-vector multiplication (MVM) (1.8) in high dimensions. **Chapter 3** covers circuits and architecture of flash quantizers used to convert analog outputs of the computational array back into the digital domain.

Part III Kerneltron II: Oversampling Kernel Machine presents a new generation of *Kerneltron* processors where delta-sigma analog-to-digital conversion of the analog array outputs combined with oversampled unary coding of the digital inputs relaxes precision requirements in the quantization. **Chapter 4** describes the oversampling MVM architecture. **Chapter 5** elaborates on circuits and architecture of delta-sigma algorithmic analog-to-digital quantizers.

Part IV Algorithmic Enhancements covers algorithms enhancing the performance of *Kerneltron* processors. **Chapter 6** utilizes redundant data encoding to enhance computation resolution in kernel machines with Nyquist quantizers (*e.g., Kerneltron I*). **Chapter 7** presents a stochastic encoding scheme relaxing precision requirements in the analog implementation by one bit for each four-fold increase in vector dimension, N in (1.8), while retaining full digital system-level resolution.

Part V Conclusions briefly summarizes our contributions and discusses their impact.

Part II

Kerneltron I: Massively Parallel VLSI Kernel Machine

Chapter 2

Charge-Mode Parallel Architecture for Matrix-Vector Multiplication

In Chapter 1 we gave an overview of kernel machines for pattern recognition. We identified the limitation of existing kernel machine implementations – insufficient computational efficiency in real-time object detection in video. The architecture of the *Kerneltron*, a massively parallel kernel machine, has been introduced, that achieves significantly higher throughput while maintaining low power dissipation. This chapter describes the first mixed-signal VLSI implementation of this architecture, *Kerneltron I*.

2.1 Introduction

As shown in the previous chapter, Support Vector Machine (SVM) data classification and function approximation incur a significant amount of computation, in excess of even the most powerful processors available today. The most computationally intensive operation in SVM decision rule is computing large number of inner-products in parallel in high-dimensional spaces. This amounts to the operation of matrix-vector multiplication (MVM) in large dimensions:

$$Y_m = \sum_{n=0}^{N-1} W_{mn} X_n \tag{2.1}$$

with N-dimensional input vector X_n, M-dimensional output vector Y_m, and $M \times N$ matrix elements W_{mn}. The matrix elements W_{mn} denote the support vectors X_{mn}, or the wavelet transformed support vectors (1.7) for convenience of notation (see the footnote on page 11).

As argued in Chapter 1, analog computational arrays [21, 25, 27, 30] for neural information processing offer very large integration density and throughput as needed for real-time high-dimensional MVM computation. Despite the success of adaptive algorithms and architectures in reducing the effect of analog component mismatch and noise on system performance [20, 31], the precision and repeatability of analog VLSI computation under process and environmental variations is inadequate for some applications. Digital implementation [16] offers absolute precision limited only by word-length, but at the cost of significantly larger silicon area and power dissipation compared with dedicated, fine-grain parallel analog implementation, *e.g.,* [25, 30]. Mixed-signal solutions combine the advantages of analog efficiency and digital precision in the implementation.

In this chapter we present a mixed-signal VLSI implementation of the *Kerneltron* architecture as follows. A mixed-signal MVM array architecture with binary decomposed matrix and vector elements is described in Section 2.2. VLSI implementation with experimental results from fabricated silicon are presented in Section 2.3. Section 2.4 quantifies the improvements in system precision obtained by postprocessing the quantized outputs of the array in the digital domain. An expanded architecture using multiple processors and compensating for analog computation offset errors is discussed in Section 2.5. Conclusions are presented in Section 2.6.

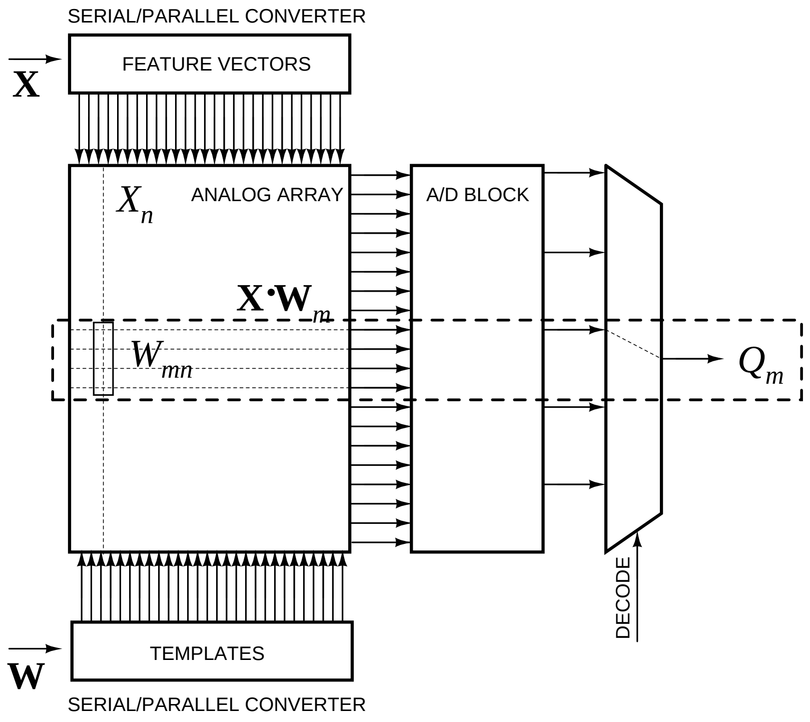


Figure 2.1: Top level architecture of *Kerneltron I* processor.

2.2 Mixed-Signal Architecture

2.2.1 Internally Analog, Externally Digital Computation

Kerneltron I is internally implemented in analog VLSI technology, but interfaces externally with the digital world. This paradigm combines the best of both worlds: it uses the efficiency of massively parallel analog computing (in particular: adding numbers in parallel on a single wire), but allows for a modular, configurable interface with other digital preprocessing and postprocessing systems. This is necessary to make the processor a general-purpose device that can tailor the matrix-vector multiplication task to the particular application where it is being used.

The digital representation is embedded, in both bit-serial and bit-parallel fashion, in the analog array architecture as shown in the top level diagram in Figure 2.1. Inputs are presented in bit-serial fashion, and matrix elements are stored locally in bit-parallel form.

Digital-to-analog (D/A) conversion at the input interface is inherent in the bit-serial implementation, and row-parallel analog-to-digital (A/D) converters are used at the output interface.

For simplicity, an unsigned binary encoding of inputs and matrix elements is assumed here, for one-quadrant multiplication. This assumption is not essential: it has no binding effect on the architecture and can be easily extended to a standard one's complement for four-quadrant multiplication, in which the significant bits (MSB) of both arguments have a negative rather than positive weight. Assume further I-bit encoding of matrix elements, and J-bit encoding of inputs:

$$W_{mn} = \sum_{i=0}^{I-1} 2^{-i-1} w_{mn}^{(i)} \tag{2.2}$$

$$X_n = \sum_{j=0}^{J-1} 2^{-j-1} x_n^{(j)} \tag{2.3}$$

decomposing (2.1) into:

$$Y_m = \sum_{n=0}^{N-1} W_{mn} X_n = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} 2^{-i-j-2} Y_m^{(i,j)} \tag{2.4}$$

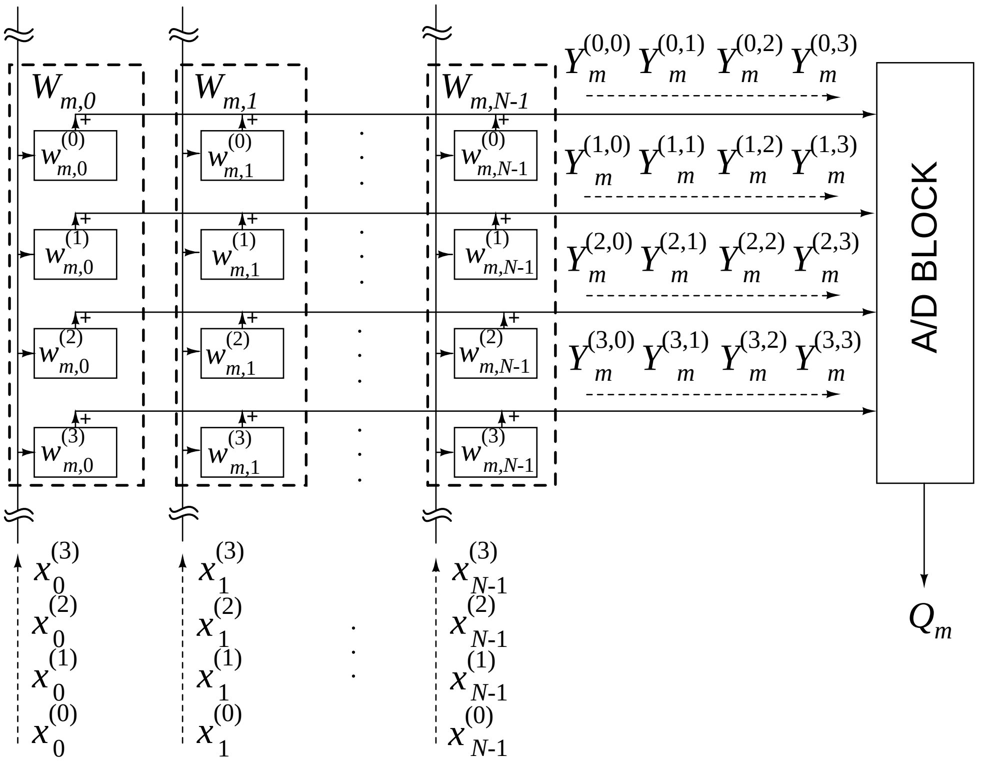
with binary-binary MVM partials:

$$Y_m^{(i,j)} = \sum_{n=0}^{N-1} w_{mn}^{(i)} x_n^{(j)} \ . \tag{2.5}$$

The proposed mixed-signal approach is to compute and accumulate the binary-binary partial products (2.5) using an analog MVM array, and to combine the quantized results in the digital domain according to (2.4).

2.2.2 Array Architecture and Data Flow

To conveniently implement the partial products (2.5), the binary encoded matrix elements ${w_{mn}}^{(i)}$ are stored in bit-parallel form, and the binary encoded inputs ${x_n}^{(j)}$ are presented in bit-serial fashion as shown in Figure 2.2. This figure represents a detailed block diagram of one slice of the top level architecture marked with a dashed line in Figure 2.1.



$W_{m,0}$
$W_{m,1}$
$W_{m,N-1}$
$w_{m,0}^{(0)}$
$w_{m,1}^{(0)}$
$w_{m,N-1}^{(0)}$
$Y_m^{(0,0)}$ $Y_m^{(0,1)}$ $Y_m^{(0,2)}$ $Y_m^{(0,3)}$
$Y_m^{(1,0)}$ $Y_m^{(1,1)}$ $Y_m^{(1,2)}$ $Y_m^{(1,3)}$
$Y_m^{(2,0)}$ $Y_m^{(2,1)}$ $Y_m^{(2,2)}$ $Y_m^{(2,3)}$
$Y_m^{(3,0)}$ $Y_m^{(3,1)}$ $Y_m^{(3,2)}$ $Y_m^{(3,3)}$
A/D BLOCK
Q_m
$x_0^{(3)}$ $x_0^{(2)}$ $x_0^{(1)}$ $x_0^{(0)}$
$x_1^{(3)}$ $x_1^{(2)}$ $x_1^{(1)}$ $x_1^{(0)}$
$x_{N-1}^{(3)}$ $x_{N-1}^{(2)}$ $x_{N-1}^{(1)}$ $x_{N-1}^{(0)}$

Figure 2.2: Block diagram of one row in the matrix with binary encoded elements $w_{mn}{}^{(i)}$, for a single m and with $I = 4$ bits. Data flow of bit-serial inputs $x_n{}^{(j)}$ and corresponding partial outputs $Y_m{}^{(i,j)}$, with $J = 4$ bits.

The bit-serial format was first proposed and demonstrated in [23], with binary-analog partial products using analog matrix elements for higher density of integration. The use of binary encoded matrix elements relaxes precision requirements and simplifies storage [24].

One row of I-bit encoded matrix elements uses I rows of binary cells. Therefore, to store an $M \times N$ *digital* matrix W_{mn}, an array of $MI \times N$ *binary* cells $w_{mn}{}^{(i)}$ is needed. One bit of an input vector is presented each clock cycle, taking J clock cycles of partial products (2.5) to complete a full computational cycle (2.1). The input binary components $x_n{}^{(j)}$ are presented least significant bit (LSB) first, to facilitate the digital postprocessing to obtain (2.4) from (2.5) (as elaborated in Section 2.4).

Figure 2.2 depicts one row of matrix elements W_{mn} in the binary encoded architecture, comprising I rows of binary cells $w_{mn}{}^{(i)}$, where $I = 4$ in the example shown. The data

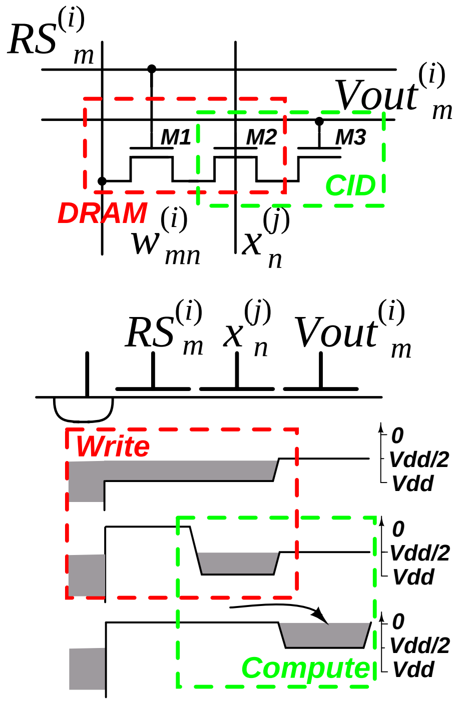


Figure 2.3: CID computational cell with integrated DRAM storage (*top*). Charge transfer diagram for active write and compute operations (*bottom*). Transistor sizes: M1 - 6/2, M2 - 6/6, M3 - 6/6.

flow is illustrated for a digital input series ${x_n}^{(j)}$ of $J = 4$ bits, LSB first (*i.e.*, descending index j). The corresponding analog series of outputs ${Y_m}^{(i,j)}$ in (2.5) obtained at the horizontal summing nodes of the analog array is quantized by a bank of analog-to-digital converters (ADC), and digital postprocessing (2.4) of the quantized series of output vectors yields the final digital result (2.1).

The quantization scheme used is critical to system performance. As shown in Section 2.4, appropriate postprocessing in the digital domain to obtain (2.4) from the quantized partial products ${Y_m}^{(i,j)}$ can lead to a significant enhancement in system resolution, well beyond that of intrinsic ADC resolution. This relaxes precision requirements on the analog implementation of the partial products (2.5). A dense and efficient charge-mode VLSI implementation is described next.

2.3 Charge-Mode VLSI Implementation

2.3.1 CID/DRAM Cell and Array

The elementary cell combines a CID computational unit [23, 24], computing one argument of the sum in (2.5), with a DRAM storage element [32]. The cell stores one bit of a matrix element ${w_{mn}}^{(i)}$, performs a one-quadrant binary-binary multiplication of ${w_{mn}}^{(i)}$ and ${x_n}^{(j)}$, and accumulates the result across cells with common m and i indices. The circuit diagram and operation of the cell are given in Figure 2.3. An array of cells thus performs (unsigned) binary multiplication (2.5) of matrix ${w_{mn}}^{(i)}$ and vector ${x_n}^{(j)}$ yielding ${Y_m}^{(i,j)}$, for values of i in parallel across the array, and values of j in sequence over time.

The cell contains three MOS transistors connected in series as depicted in Figure 2.3. Transistors M1 and M2 comprise a dynamic random-access memory (DRAM) cell, with switch M1 controlled by *Row Select* signal ${RS_m}^{(i)}$. When activated, the binary quantity ${w_{mn}}^{(i)}$ is written in the form of charge stored under the gate of M2. Transistors M2 and M3 in turn comprise a charge injection device (CID), which by virtue of charge conservation moves electric charge between two potential wells in a non-destructive manner [23], [24], [33], [34], [35], [36]. The cell operates in two phases: *Write* and *Compute*. When a matrix element value is being stored, ${x_n}^{(j)}$ is held at Vdd and $Vout$ at a voltage $Vdd/2$. To perform a write operation, either an amount of electric charge is stored under the gate of M2, if ${w_{mn}}^{(i)}$ is low, or charge is removed, if ${w_{mn}}^{(i)}$ is high. The amount of charge stored, $\triangle Q$ or 0, corresponds to the binary value ${w_{mn}}^{(i)}$.

Once the charge has been stored, the switch M1 is deactivated, and the cell is ready to compute. The charge left under the gate of M2 can only be redistributed between the two CID transistors, M2 and M3. An active charge transfer from M2 to M3 can only occur if there is non-zero charge stored, and if the potential on the gate of M2 drops below that of M3 [23]. This condition implies a logical AND, *i.e.,* unsigned binary multiplication, of ${w_{mn}}^{(i)}$ and ${x_n}^{(j)}$. The multiply-and-accumulate operation is then completed by capacitively sensing the amount of charge transferred onto the electrode of M3, the output summing

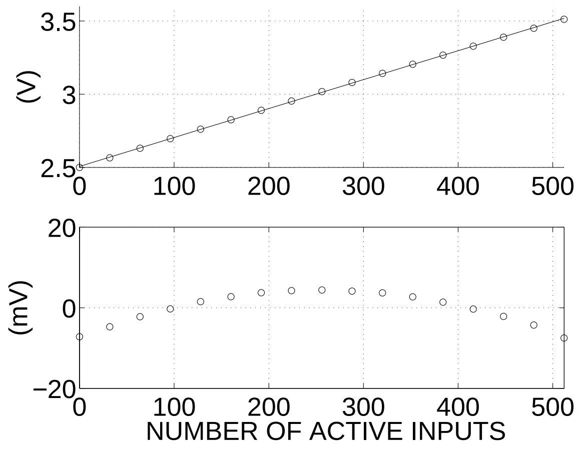


Figure 2.4: Voltage transfer characteristic (*top*) and integral nonlinearity (*bottom*) for a row of 512 CID/DRAM cells, simulated using *SpectreS* with MOS model parameters extracted from a 0.5 μm process.

node. To this end, the voltage on the output line, left floating after being pre-charged to $Vdd/2$, is observed. When the charge transfer is active, the cell contributes a change in voltage

$$\triangle V_{out} = \triangle Q / C_{M3} \tag{2.6}$$

where C_{M3} is the total capacitance on the output line across cells. The total response is thus proportional to the number of actively transferring cells. After deactivating the input ${x_n}^{(j)}$, the transferred charge returns to the storage node M2. The CID computation is non-destructive and intrinsically reversible [23], and DRAM refresh is only required to counteract junction and subthreshold leakage.

The bottom diagram in Figure 2.3 depicts the charge transfer timing diagram for write and compute operations in the case when both ${w_{mn}}^{(i)}$ and ${x_n}^{(j)}$ are of logic level 1. A logic level 0 for ${w_{mn}}^{(i)}$ is represented as Vdd, and a logic level 1 is represented as $Vdd/2$, where Vdd is the supply voltage. For ${x_n}^{(j)}$, logic level 0 is represented as Vdd, and logic level 1 as GND.

Transistor-level simulation of a 512-element row indicates a dynamic range of 43 dB, as illustrated in Figure 2.4, and a computational cycle of 10 μs with power consumption of 50 nW per cell. Experimental results from a fabricated prototype are presented next.

2.3.2 Experimental Results

A VLSI prototype of *Kerneltron I* matrix-vector multiplier, integrated on a $3 \times 3\ \mathrm{mm}^2$ die in 0.5 μm CMOS technology was fabricated and tested. The chip contains an array of 512×128 CID/DRAM cells, and a row-parallel bank of 128 gray code flash ADCs. Figure 2.5 depicts the micrograph and system floorplan of the chip. The layout size of the CID/DRAM cell is $8\lambda \times 45\lambda$ with $\lambda = 0.3\mu m$.

The mixed-signal MVM processor interfaces externally in digital format. Two separate shift registers load the matrix elements along odd and even columns of the DRAM array. Integrated refresh circuitry periodically updates the charge stored in the array to compensate for leakage. Vertical bit lines extend across the array, with two rows of sense amplifiers at the top and bottom of the array. The refresh alternates between even and odd columns, with separate select lines. Stored charge corresponding to matrix element values can also be read and shifted out from the chip for test purposes. All of the supporting digital clocks and control signals are generated on-chip.

Figure 2.6 shows the measured linearity of the computational array. The cases shown are when all binary weight storage elements are actively charged and discharged, and an all-ones sequence of bits is shifted through the input register, initialized to all-zeros bit values. For every 1-bit shift, a computation is performed and the result is observed on the output sense line. The experimentally observed linearity agrees with the simulation results in Figure 2.4. The feed-through input dependent offsets are compensated for as described in Section 2.5.

The chip contains 128 row-parallel flash ADCs, *i.e.,* one dedicated ADC for each m and i. In the present implementation, Y_m is obtained off-chip by combining the ADC quantized outputs ${Y_m}^{(i,j)}$ over i (rows) and j (time) according to (2.4). Issues of precision and complexity in the implementation of (2.4) are studied below.

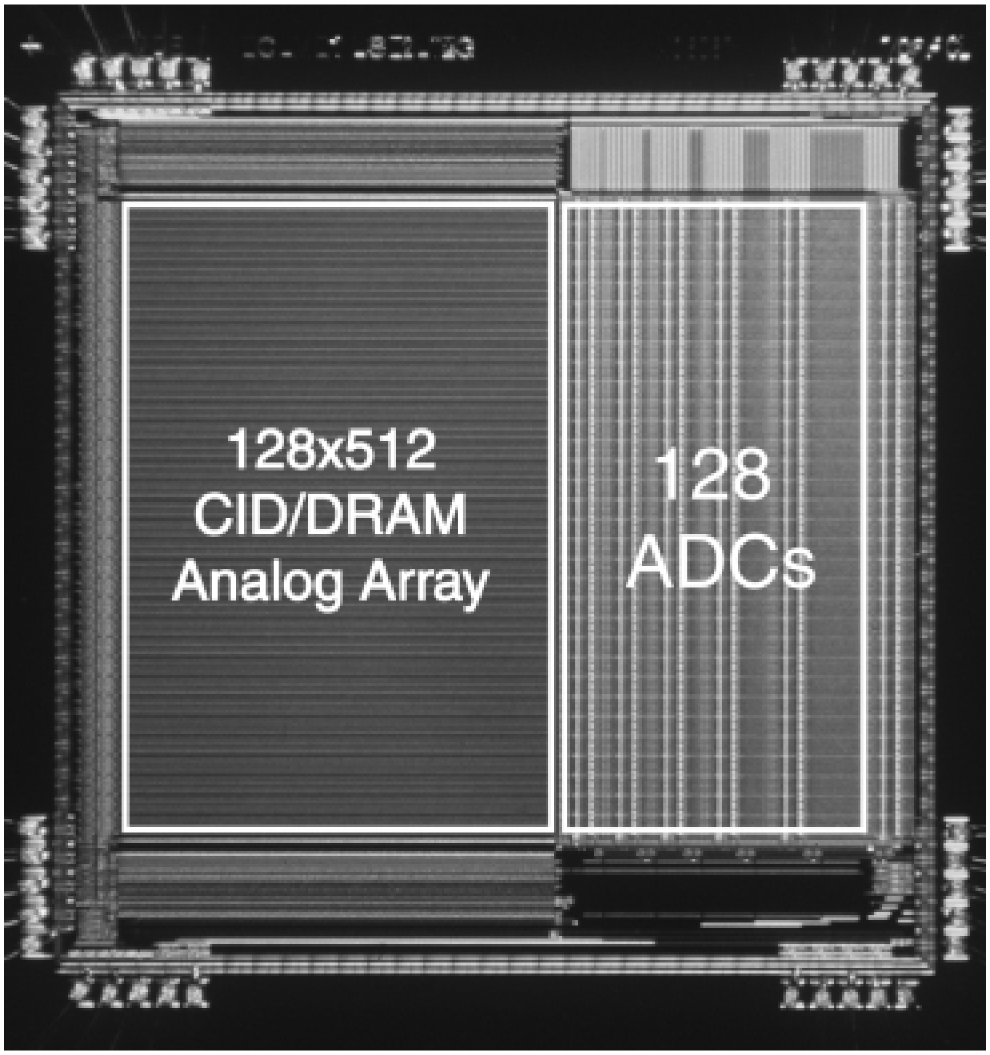


Figure 2.5: Micrograph of a *Kerneltron I* prototype, containing an array of 512×128 CID/DRAM cells, and a row-parallel bank of 128 flash ADCs. Die size is $3 \text{ mm} \times 3 \text{ mm}$ in 0.5 μm CMOS technology.

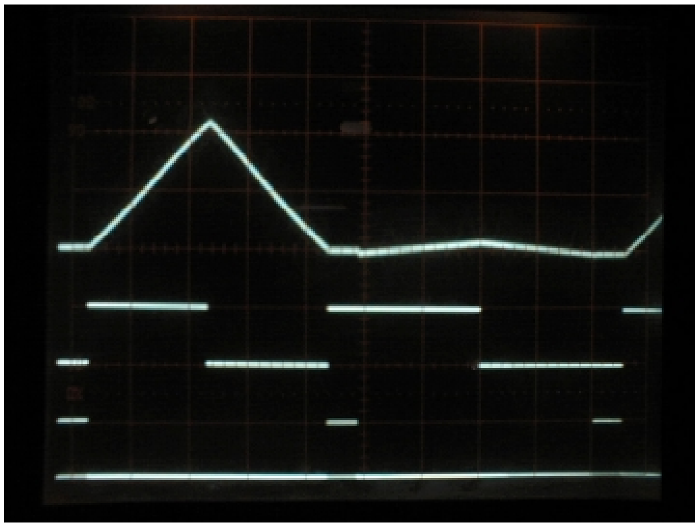


Figure 2.6: Measured linearity of the computational array. Two cases are shown: all binary weight storage elements are actively charged (*left*) and discharged (*right*). All logic "1" sequence of bits is shifted through the input register, initialized to all-"0" bit values. For every 1-bit shift, a computation is performed. Waveforms shown, *top to bottom*: the analog voltage output on the sense line; input data - on an input pin in common for both input and weight shift register; clock for weight shift register.

2.4 Quantization and Digital Resolution Enhancement

Significant improvements in precision can be obtained by exploiting the binary representation of matrix elements and vector inputs, and performing the computation (2.4) in the digital domain, from quantized estimates of the partial outputs (2.5). The effect of averaging the quantization error over a large number of quantized values of ${Y_m}^{(i,j)}$ boosts the precision of the digital estimate of Y_m, beyond the intrinsic resolution of the analog array and the analog-to-digital converters used.

2.4.1 Accumulation and Quantization

The outputs ${Y_m}^{(i,j)}$ for a single m obtained from the analog array over J clock cycles can be conceived as an $I \times J$ matrix, shown in Figure 2.2. Elements of this matrix located along diagonals (*i.e.,* elements with a common value of $i + j$) have identical binary weight in (2.4). Therefore, the summation in (2.4) could be rearranged as:

$$Y_m = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} 2^{-(i+j+2)} Y_m^{(i,j)} = \sum_{k=0}^{K-1} 2^{-(k+2)} Y_m'^{(k)} \qquad (2.7)$$

where $k = i + j$, $K = I + J - 1$ and

$$Y_m'^{(k)} = \sum_{i=\kappa(k,J)}^{k-\kappa(k,I)} Y_m^{(i,k-i)} \qquad (2.8)$$

with $\kappa(k, I) \equiv \max(0, k - I + 1)$ and $\kappa(k, J) \equiv \max(0, k - J + 1)$.

Several choices can be made in the representation of the signals being accumulated and quantized. One choice is whether to quantize each array output, ${Y_m}^{(i,j)}$, and accumulate the terms in (2.8) in the digital domain, or accumulate the terms in the analog domain and quantize the resulting ${Y'_m}^{(k)}$. Clearly, the former leads to higher precision, while the latter has lower complexity of implementation, as discussed in more detail in Chapter 6. We opted for the former, and implemented a parallel array of low-resolution flash ADCs, one for each row output i.

2.4.2 Row-parallel Flash A/D Conversion

Consider the case of row-parallel flash (*i.e.,* bit-parallel) A/D conversion, where all $I \times J$ values of ${Y_m}^{(i,j)}$ are fully quantized. Figure 2.7 presents the corresponding architecture, shown for a single output vector component m. Each of the I horizontal summing nodes, one for each bit-plane i of component m, interfaces with a dedicated flash A/D converter producing a digital output ${Q_m}^{(i,j)}$ of L-bit resolution. The summations (2.8) and (2.7) are then performed in the digital domain:

$$Q_m = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} 2^{-(i+j+2)} Q_m^{(i,j)} = \sum_{k=0}^{K-1} 2^{-(k+2)} Q_m'^{(k)}\ , \qquad (2.9)$$

and

$$Q_m'^{(k)} = \sum_{i=\kappa(k,J)}^{k-\kappa(k,I)} Q_m^{(i,k-i)}\ . \qquad (2.10)$$

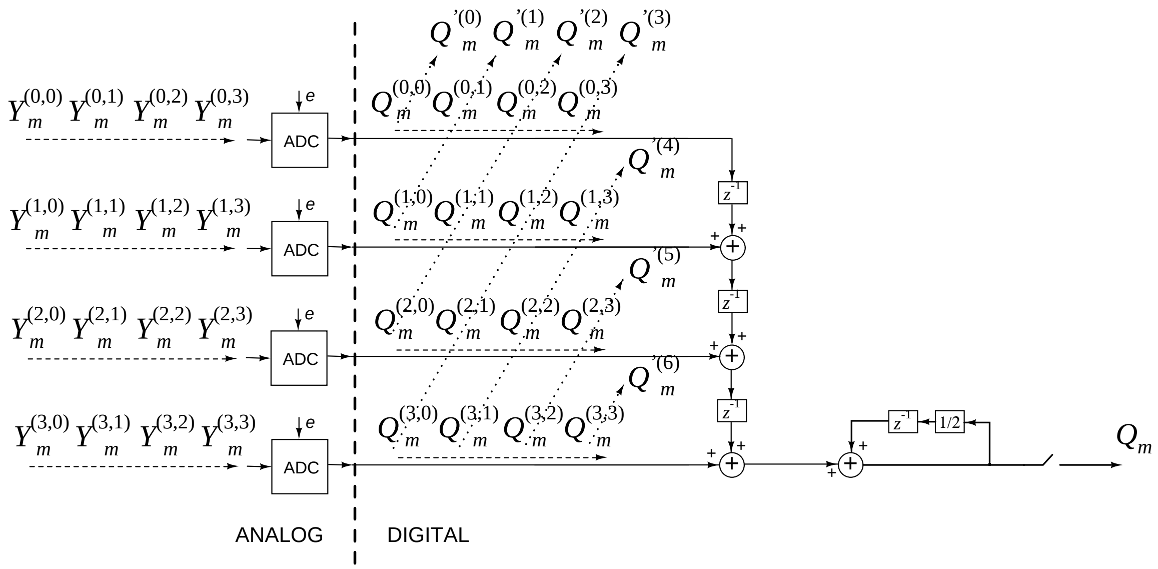


Figure 2.7: Diagram for the A/D quantization and digital postprocessing block in Figure 2.2, using row-parallel flash A/D converters. The example shown is for a single m, LSB-first bit-serial inputs, and $I = J = 4$.

A block diagram for a digital implementation is shown on the right of Figure 2.7, assuming LSB-first bit-serial inputs (descending index j). With radix 2, a shift-and-accumulate operation avoids the need for digital multiplication. The LSB-first bit-serial format minimizes latency and reduces the length of the register accumulating Q_m.

If the ADC is capable of resolving each individual binary term in the analog sum (2.5), then the sum is retrieved from the ADC with zero error, as if computed in the digital domain. For *zero-error* digital reconstruction, the ADC requires (at least) $N + 1$ quantization levels, that coincide with the levels of the charge transfer characteristic for any number (0 to N) of active cells along the output row of the analog array. Provided nonlinearity and noise in the analog array and the ADC are within one LSB (at the $\log_2(N + 1)$-bit level), the *quantization error* then reduces to zero, and the output Q_m is obtained at the maximum digital MVM resolution of $I + J + \log_2(N + 1)$ bits. For large arrays, this is usually more than needed, and places too stringent requirements on analog precision, $L \geq \log_2(N + 1)$.

In the remainder of this section we study the error of the digitally constructed output Q_m in the practical case where the resolution of the ADC is below that of the dimensions of the array, $L < \log_2(N+1)$. In particular, we study the properties of Q_m assuming uncorrelated statistics of quantization error. The analysis yields an estimate of the gain in resolution that can be obtained relative to that of the ADC quantizers, independent of the matrix and input representation N, I, and J. The quantization is modeled as:

$$Q_m^{(i,j)} = Y_m^{(i,j)} + e_m^{(i,j)}, \tag{2.11}$$

where $e_m^{(i,j)}$ represents the quantization error, modeled as uniform random *i.i.d.* within one LSB. Conceptually, the error term $e_m^{(i,j)}$ in (2.11) could also include effects of noise and nonlinear distortion in the analog summation (2.5), although in practice the precision of the array exceeds the ADC resolution, as shown in the experimental data of Section 2.3.2. From (2.9) and (2.11), the error in the digitally constructed output

$$Q_m = Y_m + E_m, \tag{2.12}$$

can then be expanded as

$$E_m = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} 2^{-(i+j+2)} e_m^{(i,j)} . \tag{2.13}$$

Define s the full-scale range of the ADC acquiring ${Q_m}^{(i,j)}$, and S the corresponding range of the constructed digital output Q_m. Then according to (2.9),

$$S = s \sum_{i=0}^{I-1} 2^{-(i+1)} \sum_{j=0}^{J-1} 2^{-(j+1)} = s\,(1-2^{-I})(1-2^{-J}) \tag{2.14}$$

which approaches s for $I, J \to \infty$. Therefore, the full signal range is approximately equal to the output signal range of each of the ADCs for large I and J.

Let the variance of the uniform quantization noise e in (2.11) be σ_e^2, identical $\forall i, j$. In the *Central Limit*, the cumulative quantization error E can be roughly approximated as a normal process, with variance equal to the sum of the variances of all terms in the summation (2.13). Each signal component, ${Q_m}^{(i,j)}$, with quantization noise e but scaled with binary weight $2^{-(i+j+2)}$, contributes a variance $2^{-2(i+j+2)}\sigma_e^2$ in the sum (2.13), and the total variance σ_E^2 of the output error E is expressed as:

$$\sigma_E^2 = \sigma_e^2 \sum_{i=0}^{I-1} 2^{-2(i+1)} \sum_{j=0}^{J-1} 2^{-2(j+1)} = \sigma_e^2 \frac{1-2^{-2I}}{3} \frac{1-2^{-2J}}{3} \tag{2.15}$$

which approaches $(\sigma_e/3)^2$ for $I, J \to \infty$. Therefore, the signal-to-quantization-noise ratio (SQNR) approaches

$$\frac{S}{\sigma_E} \approx 3\frac{s}{\sigma_e} \tag{2.16}$$

for large I and J. In other words, by quantizing each array output ${Y_m}^{(i,j)}$ instead of the combined total Y_m, we obtain an improvement in signal-to-quantization-noise ratio of a factor 3.

To characterize the improved precision in terms of *effective resolution* (in bits), it is necessary to relate the second order statistics of the quantization error e or E to a measure of the error indicative of resolution. There is a certain degree of arbitrariness in doing so, but in what follows we define resolution as the *median* of the absolute error, *i.e.,* the (symmetric) extent of the 50 % confidence interval of the error. The choice of convention matters, because the distributions for e and E are different— e is approximately uniform, and E in the *Central Limit* is normal.

Let e be uniformly distributed in the interval $[-\Delta, \Delta]$. The median absolute value is then $\mathcal{M}_e = \frac{1}{2}\Delta$, and the variance $\sigma_e^2 = \frac{1}{3}\Delta^2$, yielding the relation

$$\mathcal{M}_e = \frac{\sqrt{3}}{2}\sigma_e \tag{2.17}$$

for the uniform distribution. The median absolute value for a normal distribution, in terms of the standard deviation, is approximately

$$\mathcal{M}_E = 0.675\sigma_E \tag{2.18}$$

This allows to express the SQNR gain in (2.16) as a gain in *median resolution*:

$$\frac{S}{\mathcal{M}_E} \approx 3\frac{\sqrt{3}/2}{0.675}\frac{s}{\mathcal{M}_e} \approx 3.85\frac{s}{\mathcal{M}_e} \tag{2.19}$$

or, in other words, a gain of approximately 2 bits over the resolution of each ADC.

For a flash ADC architecture, two "free" extra bits of resolution above the precision available from the analog array and quantization are significant, since the implementation cost is exponential in the number of bits. Additional gains in resolution of computation on a MVM architecture with low-resolution Nyquist-rate ADCs can be made by redundant data coding and utilizing algorithmic quantization schemes as presented in Chapter 6.

2.5 Multi-Chip Architecture with Offset Compensation

2.5.1 Multi-Chip Architecture

The method of on-chip MVM computation described above allows for an array size of 1000 $\times$ 1000 cells, in a 0.35 μm CMOS technology implemented on a 6 $\times$ 6 mm die. Computations on matrices of higher dimensionality, at maximum possible precision, can be performed by using multiple MVM chips. Processors can be cascaded to expand matrix row or column spaces beyond the limits of a single chip capacity.

In the ideal case, to extend matrix row space, digital outputs of systems with shorter input vector lengths can be combined to perform a computation in a higher dimensional input space. Extension of column space is in principle also unlimited (assuming high read-out speeds). Cascading along rows of the matrix (allowing for higher dimensionality of input vectors) requires addition of digital numbers, while cascading along columns (in order to increase the number of matrix elements for a fixed input vector dimensionality) only necessitates multi-chip output multiplexing in time.

In reality, there are a number of sources of error that contribute offsets to the output of each MVM chip. These offset terms can be compensated for in the multi-chip architecture as described in the next section.

2.5.2 Offset Compensation

In Section 2.3 we already considered some of the sources of computation error. They are imprecision of binary multiplications through charge sharing between potential wells in a CID unit with capacitively coupled output, and of charge-mode analog addition on a single node. Because of linearity errors the result of such a computation is valid up to

Table 2.1: Input-output mapping of CID/DRAM computational cell. The input-output feedthrough error and charge leakage related offset are introduced when the input is logic "1".

$x_n^{(j)}$	$w_{mn}^{(i)}$	$y_{mn}^{(i,j)}$
0	0	0
0	1	0
1	0	$\varepsilon(t)$
1	1	$1+\varepsilon(t)$

approximately 7 bits. We have discussed the effect of these errors and ADC resolution on the overall system precision in Section 2.4.

Other significant sources of error in analog array-based computation are input-output feedthrough and leakage current in DRAM storage cells. The architecture described here is capable of compensating for these analog computation errors in digital domain by using an extra reference MVM chip as shown in Figure 2.8.

The input-output feedthrough error is a result of capacitive coupling of input (vertical) lines onto the output (horizontal) lines through parasitic capacitances (*e.g.,* metal lines overlap capacitance, gate-to-diffusion capacitance). From Section 2.3 we know that the output of the analog cell ideally changes by ΔV_{out} only when both matrix element coefficient and input vector component coefficient are logic "1": ${w_{mn}}^{(i)} = Vdd/2$ (charge is stored) and, in the computation phase, ${x_n}^{(j)} = 0V$ (input is reset). Any other combination of input arguments should produce zero voltage change at the output. The cell is effectively an analog AND gate. In practice, switching the input line ${x_n}^{(j)}$, even when no charge is stored in CID cell, causes a small change in the output voltage, ϵ, as a result of input-output capacitive coupling. This effect is modeled here as shown in Table 2.1. The output of a single cell is denoted as ${y_{mn}}^{(i,j)}$.

From equations (2.5) and (2.11), taking into account the input-output feedthrough error, the MVM quantized output partials of the p-th processor in a P-processor architecture can

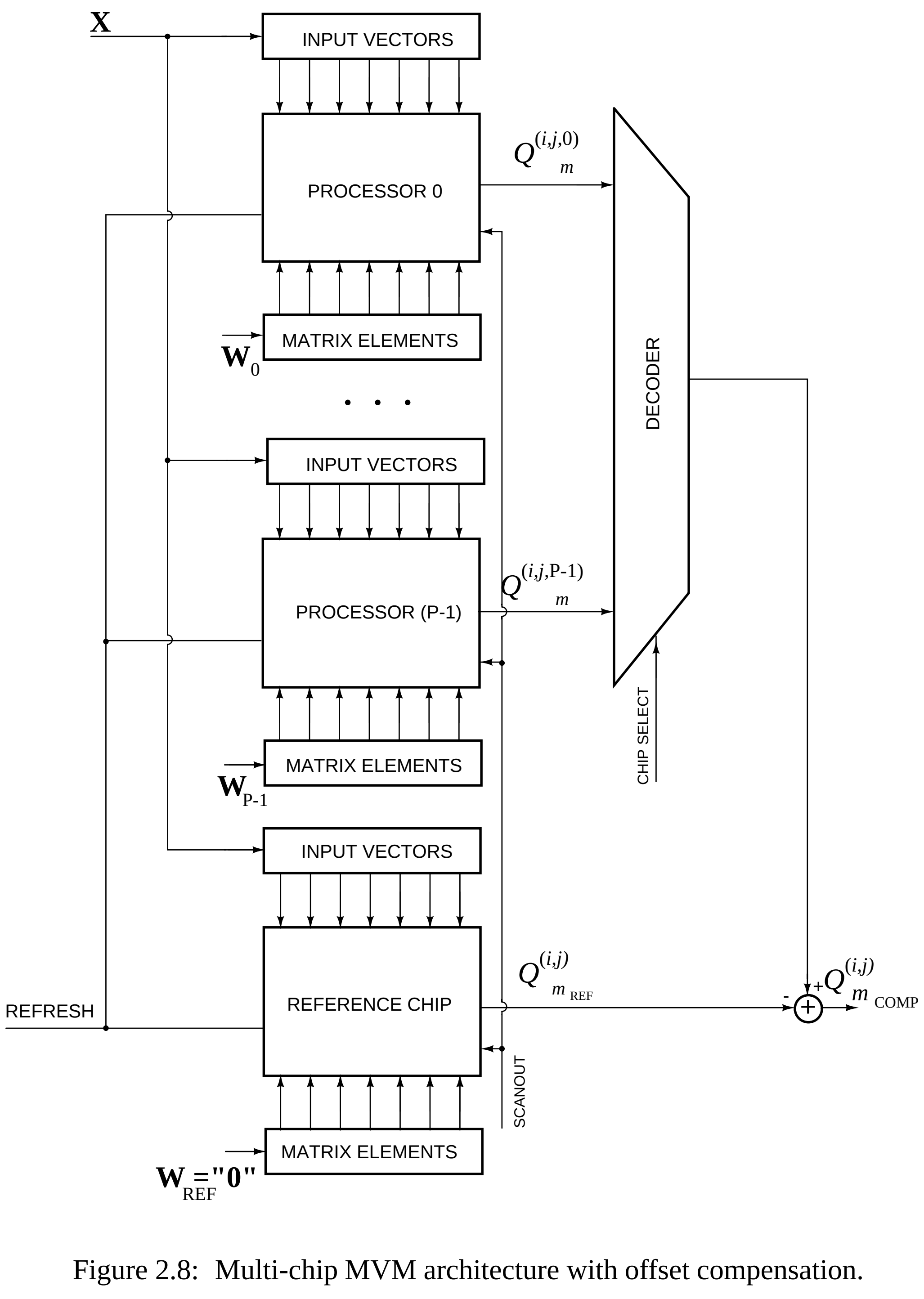


Figure 2.8: Multi-chip MVM architecture with offset compensation.

now be expressed as:

$$\begin{aligned} Q_m^{(i,j,p)} &= \sum_{n=0}^{N-1} y_{mn}^{(i,j,p)} + e_m^{(i,j,p)} \\ &= (1+\epsilon)\sum_{n=0}^{N-1} w_{mn}^{(i,p)} x_n^{(j)} + \epsilon \sum_{n=0}^{N-1} (1 - w_{mn}^{(i,p)}) x_n^{(j)} + e_m^{(i,j,p)} \\ &= \sum_{n=0}^{N-1} w_{mn}^{(i,p)} x_n^{(j)} + \epsilon \sum_{n=0}^{N-1} x_n^{(j)} + e_m^{(i,j,p)} \,. \end{aligned} \tag{2.20}$$

The term $\epsilon \sum_{n=0}^{N-1} x_n^{(j)}$ in equation (2.20) is an offset term proportional to the number of active vector components. To compensate for this term an identical extra, reference, chip is used in the multi-chip architecture as shown in Figure 2.8. The same inputs are presented to all chips in the system while only logic “0” matrix element coefficients are stored in the reference chip. It outputs quantized partials of the form:

$$Q_m^{(i,j)}{}_{REF} = \epsilon \sum_{n=0}^{N-1} x_n^{(j)} + e_m^{(i,j)}{}_{REF} \,. \tag{2.21}$$

In order to compensate for feedthrough offsets, once computation has been performed on chips, the output of the reference chip is subtracted from the output of processors in digital domain:

$$Q_m^{(i,j,p)}{}_{COMP} \;=\; Q_m^{(i,j,p)} - Q_m^{(i,j)}{}_{REF} = \sum_{n=0}^{N-1} w_{mn}^{(i,p)} x_n^{(j)} + e'_m{}^{(i,j,p)} \,, \tag{2.22}$$

where

$$e'_m{}^{(i,j,p)} = e_m^{(i,j,p)} - e_m^{(i,j)}{}_{REF} \,. \tag{2.23}$$

Another important source of errors requiring specific consideration is DRAM leakage current. In a standard two-level DRAM cell, the exact amount of charge stored is not crucial. It is used only for binary operations of readout and refresh, and a slow refresh scheme can be used. In contrast, the CID/DRAM cell produces an analog output which is proportional to the amount of charge stored in the charge injection device (2.6). Over multiple computation cycles, different rows are being refreshed, producing differences in the temporal decay profile of charge stored along rows of cells.

In the multi-chip configuration shown in Figure 2.8, the refresh clock of the same frequency as in standard DRAMs is used. It is fed to all processors, including the reference chip. Using a reference chip with all cells containing logic “0” coefficients and refreshed synchronously with processor chips ensures the same charge decay profile in its cells and voltage increase profile at its outputs. To compensate for charge decays caused by leakage current, the outputs from the reference chip are subtracted from the outputs of the corresponding rows of each of the processors.

Therefore, both input-output feedthrough input-dependent offset and charge decay input and time-dependent offset are compensated for in the multi-chip MVM architecture by using one reference chip supplied with identical inputs, synchronous refresh clock and all logic “0” matrix elements. Subtraction of outputs of equivalent rows in digital domain eliminates both input-dependent and temporal errors.

2.6 Conclusions

A charge-mode VLSI architecture of *Kerneltron I* for parallel matrix-vector multiplication in large dimensions ($N, M =$ 100–10,000) has been presented. An internally analog, externally digital architecture offers the best of both worlds: the density and energetic efficiency of an analog VLSI array, and the noise-robustness and versatility of a digital interface. A three-transistor unit cell combines a single-bit dynamic random-access memory (DRAM) and a charge injection device (CID) binary multiplier and analog accumulator. Digital multiplication of variable resolution is obtained with bit-serial inputs and bit-parallel storage of matrix elements, by combining quantized outputs from multiple rows of cells over time. The combination of analog array processing and digital postprocessing enhances the precision of the digital MVM output, exceeding the resolution of the quantized analog array outputs by 2 bits. Additional enhancements in resolution can be obtained by utilizing redundant data coding and employing other (*e.g.,* algorithmic) Nyquist-rate quantizers as discussed in Chapter 6. Significantly larger gains in precision could be achieved by exploiting the statistics of binary terms in the analog summation (2.5) as demonstrated in Chapter 7. A reference subtraction scheme with one additional processor compensates for clock feedthrough and charge leakage in the array.

Fine-grain massive parallelism and distributed memory, in an array of CID/DRAM cells, provides a computational efficiency (bandwidth to power consumption ratio) exceeding that of digital multiprocessors and DSPs by several orders of magnitude. A 512 $\times$ 128 MVM prototype fabricated in 0.5 μm CMOS offers 2×10^{12} binary MACS (multiply-and-accumulates per second) per Watt of power. This opens up possibilities for low-power real-time implementations of kernel machines for pattern recognition in human-machine interfaces [5], artificial vision [37] and vision prostheses [38].

Requirements on the resolution of analog-to-digital converters can be relaxed by utilizing single-bit quantizers in the delta-sigma loop as described in Chapter 4, while the VLSI implementation of row-parallel flash analog-to-digital converters in *Kerneltron* architecture is presented in the next chapter.

Chapter 3

Flash Analog-to-Digital Converter with Charged-Based MOS Folding Circuit

In the previous chapter we assumed computational array quantizers to be generic flash analog-to-digital converters (ADCs). This chapter provides a thorough treatment of the architecture and circuits of charge-based flash ADCs developed for *Kerneltron I* array processor.

3.1 Introduction

High-performance data conversion can be achieved either by expending power and area to achieve high precision in a single analog architecture, or by distributing the architecture over multiple low-resolution quantization tasks each implemented with relatively imprecise analog circuits, and combined in the digital domain. The latter approach has proven superior in attaining very high precision, by distributing the quantization process over time using delta-sigma modulation [39]. Both high speed and high resolution can be achieved by distributing the quantization process in space. To this end, it is necessary to implement very space efficient, low-resolution quantizers.

We present a charge-based circuit that implements an offset-compensated comparator with one capacitor and four nMOS transistors. The design targets applications in hybrid analog-digital computing using large-scale analog arrays [40], specifically *Kerneltron I*, where parallelism, redundancy in information representation (Chapter 6), and statistical data coding ([41], Chapter 7) can be used to compensate for imperfections in analog computation.

3.2 Charge-Domain Correlated Double Sampling Comparator

3.2.1 Capacitor-nMOS Integrator

To obtain high density and high speed in a comparator and folding circuit, the challenge is to design a single stage producing a current-mode or charge-mode signal that is a saturating high-gain and offset-compensated function of a difference in input voltage. We show that in the charge domain this can be achieved using a circuit incorporating a capacitor and an exponential element, such as a diode [42] or a MOS transistor operating in subthreshold regime [43], [44], where the differential voltage is presented as an initial condition at the input. Offset compensation is achieved in the charge domain, as for the CMOS charge-transfer comparator described in [45].

In the circuit of Figure 3.1*(a)* the nMOS transistor is source coupled to a capacitor. In the subthreshold and saturation region, the drain current is exponential in gate and source voltage, and the large-signal dynamics of the integrator are described by:

$$C\frac{dV_s}{dt} = I_s = \frac{W}{L}I_o e^{(\kappa V_g - V_s)/V_t}, \tag{3.1}$$

where V_t is the thermal voltage. Integrating the differential equation (3.1) yields:

$$CV_t e^{\frac{V_s}{V_t}} = \frac{W}{L}I_o e^{\kappa V_g/V_t} t + c_1, \tag{3.2}$$

where c_1 is an integration constant. Direct substitution yields:

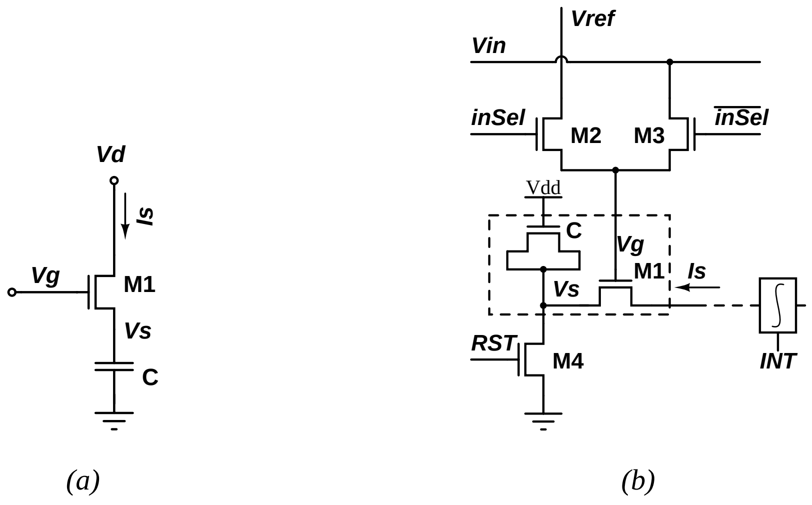


Figure 3.1: *(a)* Capacitor-nMOS integrator and *(b)* charge-based comparator. Transistor sizes: M1 - 20/2, the other are minimum size.

$$I_s(t) = \frac{CV_t}{t + \frac{c_1}{\frac{W}{L} I_o e^{\kappa V_g / V_t}}}. \tag{3.3}$$

At time $t = 0$, the input voltage is switched from $V_g(0^-)$ to $V_g(0^+)$ while the capacitor instantly retains the source voltage $V_s(0)$. The source current therefore switches from $I_s(0^-)$ to $I_s(0^+)$ over the transition at the gate:

$$I_s(0^+) = I_s(0^-) e^{\kappa \triangle V_g / V_t}, \tag{3.4}$$

where $\triangle V_g = V_g(0^+) - V_g(0^-)$. The output current (3.3) can thus be expressed in terms of initial conditions:

$$I_s(t) = \frac{I_s(0+)}{\frac{I_s(0+)}{CV_t} t + 1} = \frac{I_s(0-)}{\frac{I_s(0-)}{CV_t} t + e^{-\kappa \triangle V_g / V_t}}. \tag{3.5}$$

Interestingly, for $t \gg CV_t / I_s(0^+)$, $I_s(t)$ becomes independent of initial conditions:

$$I_s(t) \approx \frac{CV_t}{t}. \tag{3.6}$$

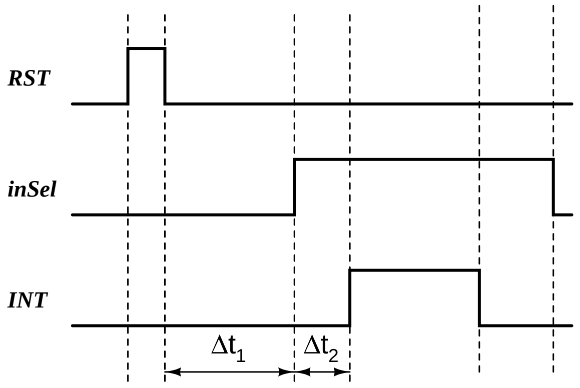


Figure 3.2: Control signal timing diagram for the comparator circuit in Figure 3.1*(b)*.

3.2.2 Comparator

Saturation of the output current of the circuit in Figure 3.1*(a)* as a function of a change in the input voltage V_g is utilized in the design of the charge-based comparator as shown in Figure 3.1*(b)*. The nMOS capacitor is initially charged by pulsing RST as shown in Figure 3.2. Over a time interval $\triangle t_1$, the capacitor discharges to raise V_s until $M1$ reaches well into the subthreshold region. The end of the interval defines the initial condition for the source current $I_s(0^-)$. The differential input voltage is presented as a transient on the gate, $\triangle V_g = V_{\text{ref}} - V_{\text{in}}$, implemented using an analog multiplexer M2-M3 and controlled by $inSel/\overline{inSel}$ timing signals in Figure 3.2. In subthreshold[1], this gate voltage transition produces a change in source current according to (3.5). By combining equations (3.4) and (3.5), the input-output characteristic of the comparator can be expressed as:

$$I_s(t) = I_{sat}\ \sigma(A(\triangle V_g - V_{\text{off}}(t))), \tag{3.7}$$

where

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.8}$$

[1]For large values of $\triangle V_g$, the nMOS may initially enter the strong inversion region. This affects the timing but not the operation of the circuit, since once the capacitor has raised V_s to reach subthreshold, the asymptotic relationship (3.6) holds again.

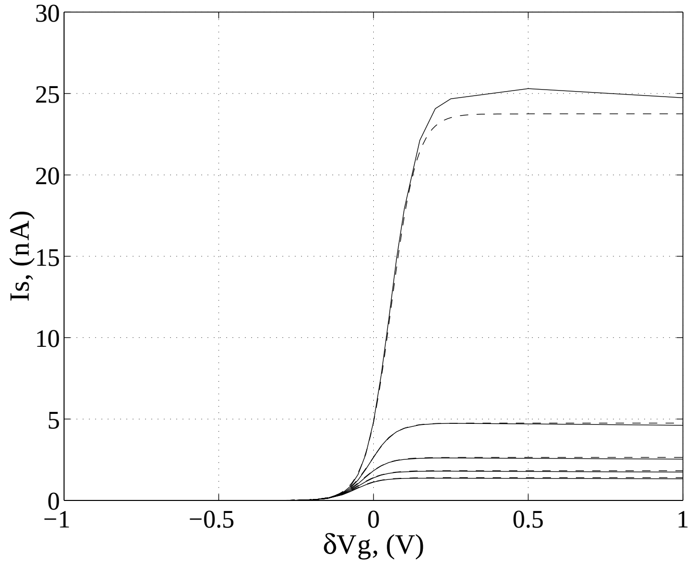


Figure 3.3: The input-output characteristics of the charge-based comparator at different fixed time intervals $\triangle t_2$ after switching the inputs, calculated using equation (3.7) (*dashed line*) and simulated using SpectreS for a 0.5 μm CMOS process (*solid line*). (*Top to bottom*: $\triangle t_2$ = 50, 250, 450, 650, 850 ns.)

is a logistic function, the amplitude saturates to $I_{sat} = CV_t/t$, the voltage is scaled by $A = \kappa/V_t$, and the offset voltage

$$V_{\mathrm{off}}(t) = \frac{V_t}{\kappa} log \frac{CV_t}{I_s(0^-)t} \tag{3.9}$$

is a logarithmic function of time. Note that by virtue of *correlated double sampling* [46] in the differential transient $\triangle V_g$ by switching M2-M3, the offset $V_{\mathrm{off}}(t)$ is independent of M1 threshold variations and, to first order, $1/f$ noise.

Figure 3.3 illustrates the input-output characteristics of the comparator for different time interval $\triangle t_2$ after switching the inputs, calculated using equation (3.7) and simulated using SpectreS for a 0.5 μm CMOS process. The offset of the comparator as a function of time from (3.9) is plotted in Figure 3.4. It scales logarithmically in time. It also depends on $I_s(0^-)$ which is controlled by the time interval, $\triangle t_1$, between the moment V_{in} is applied (after the reset) and the time V_{ref} is presented to the gate of $M1$. Figure 3.5 illustrates theoretical and simulated source current transients for different values of $\triangle V_g$.

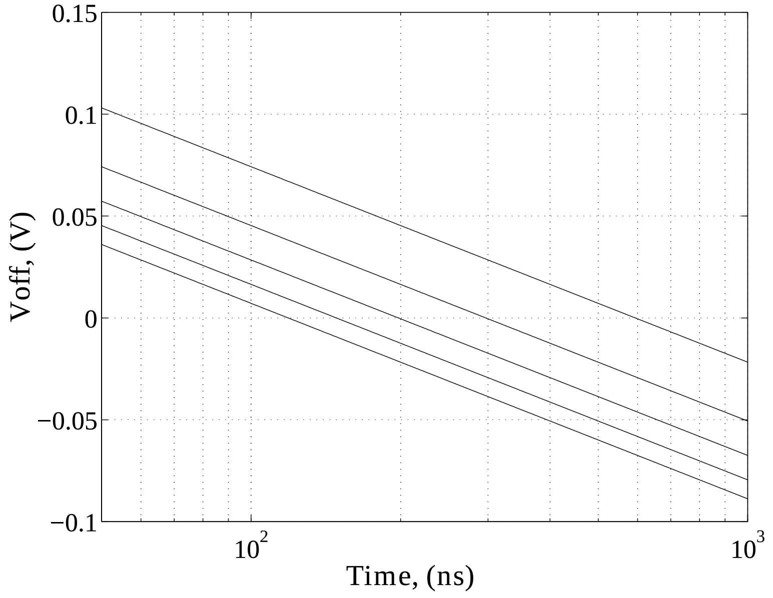


Figure 3.4: The offset voltage of the comparator as a function of time after switching the inputs ($\triangle t_2$ in Figure 3.2). Theoretical results obtained using equation (3.9) for different values of $I_s(0^-)$. (*Top to bottom*: $I_s(0^-)$ = 2, 4, 6, 8, 10nA.)

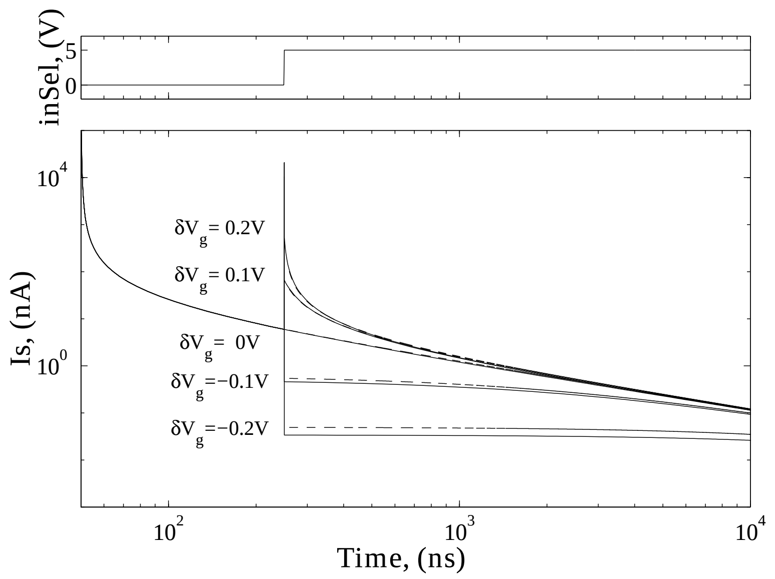


Figure 3.5: Comparator output current transients for different values of $\triangle V_g$. The *solid* line shows the SpectreS simulation results for a 0.5 μm CMOS process; the *dashed* line was obtained using equation (3.7). The initial current $I_s(0^-) = 6nA$. Effective nMOS capacitor value is 45fF.

3.3 Charge-Based Folding Circuit and Gray Code ADC

A number of solutions for gray code A/D conversion exist [47], [48], [49], [50]. Conventional folded differential logic [49] (FDL) can eliminate code errors due to its wired gray code encoding scheme. An improvement of FDL, folding cascoded differential logic (FCDL), was introduced in [50]. It allows for higher number of comparators in an encode block.

3.3.1 Gray Code Flash ADC Architecture

Figure 3.6*(a)* shows the architecture of a 3-bit gray code differential logic ADC. Comparators produce the output current

$$I_s^n = I_b((f(V_{\text{ref}}^n - V_{\text{in}}) + 1)/2), \qquad (3.10)$$

where I_b is a bias current, ${V_{\text{ref}}}^n$ is a respective reference voltage level between ${V_{\text{ref}}}^{\min}$ and ${V_{\text{ref}}}^{\max}$, $n = 1, ..., 7$, and $f(.)$ is a decision function such as $\text{sign}(.)$ or a logistic function. Each output bit, D_i, $i = 0, ..., 2$, is obtained by connecting comparator outputs differentially in a folding circuit, and then comparing the accumulated differential currents. The input-output characteristics of the ADC are illustrated in Figure 3.6*(b)*.

3.3.2 Folding Circuit

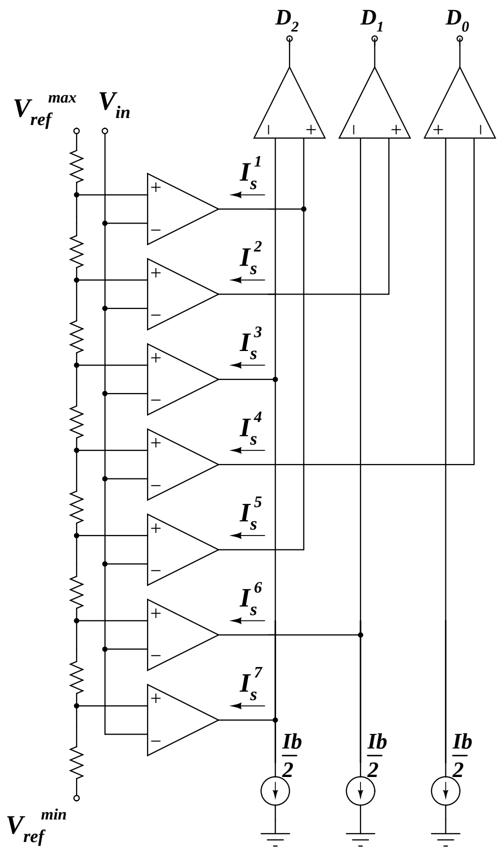
An LSB folding circuit for a 5-bit flash ADC is shown in Figure 3.7. The output currents can be expressed as:

$$I_+ = I_{sat} \sum_{n=0}^{(N-1)/4} \sigma(A(V_{\text{in}} - V_{\text{ref}}^{4n+1} - V_{\text{off}}(t))), \qquad (3.11)$$

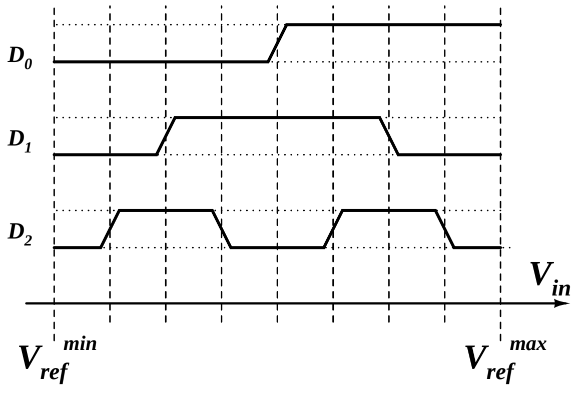
$$I_- = I_{sat} \sum_{n=0}^{(N-1)/4} \sigma(A(V_{\text{in}} - V_{\text{ref}}^{4n+3} - V_{\text{off}}(t))) + I_b/2, \qquad (3.12)$$

where

$$I_b = I_{sat}\sigma(A(V_{\text{ref}}^{\max} - V_{\text{ref}}^{\min} - V_{\text{off}}(t))). \qquad (3.13)$$



(a)



(b)

Figure 3.6: A 3-bit gray code flash A/D converter: *(a)* architecture; *(b)* characteristic.

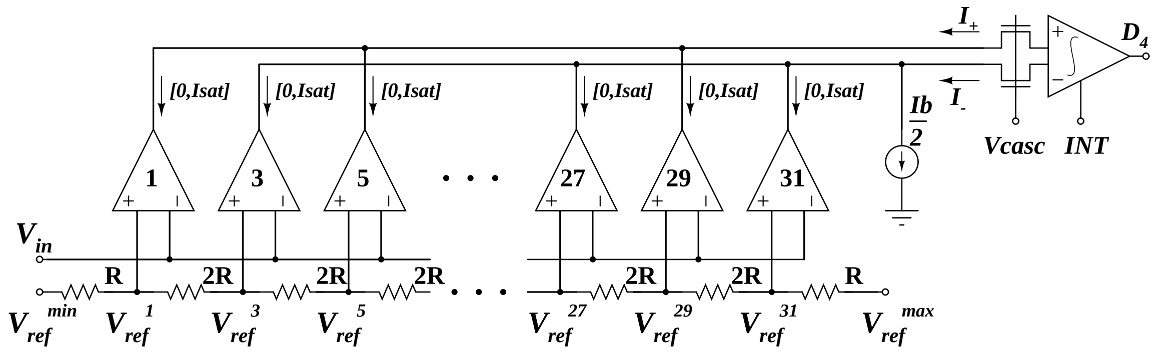


Figure 3.7: Folding circuit for a charge-based gray code flash ADC.

Theoretical and simulated output currents as a function of the input voltage for a folding circuit of a flash ADC are demonstrated in Figure 3.8.

3.3.3 Integrating Sense Amplifier

Bit decisions are made on integrated, differentially folded comparator currents using a correlated double sampling sense amplifier. The time-dependent comparator voltage offset (3.9) is inconsequential to the integration as it is in common to all comparator cells.

The integrating regenerative sense amplifier is shown in Figure 3.10. A cascode stage, $M9 - M10$, controlled by the bias voltage V_{casc}, provides low impedance input to the sense amplifier to improve the conversion speed and reduce the effect of the output conductance of the comparator cells. Current-domain correlated double sampling is achieved by swapping differential current inputs to the sense amplifier at start of integration using multiplexers $M5 - M6$ and $M7 - M8$, from precharge to evaluate mode. In precharge mode (time interval $\triangle t_2$ in Figure 3.2) capacitors C are precharged through transistors $M3$ and $M4$ to the difference in gate voltages of transistors $M1$ and $M2$ set by currents I_- and I_+ (including any offsets in the threshold voltages). In evaluate mode (time interval $\triangle t_3$ in Figure 3.2), the multiplexers $M5-M6$ and $M7-M8$ switch the input currents. The corresponding change in gate voltages causes a change in the output voltages V_{o-} and V_{o+}. The correlated double sampling scheme compensates for sense amplifier input-referred offset

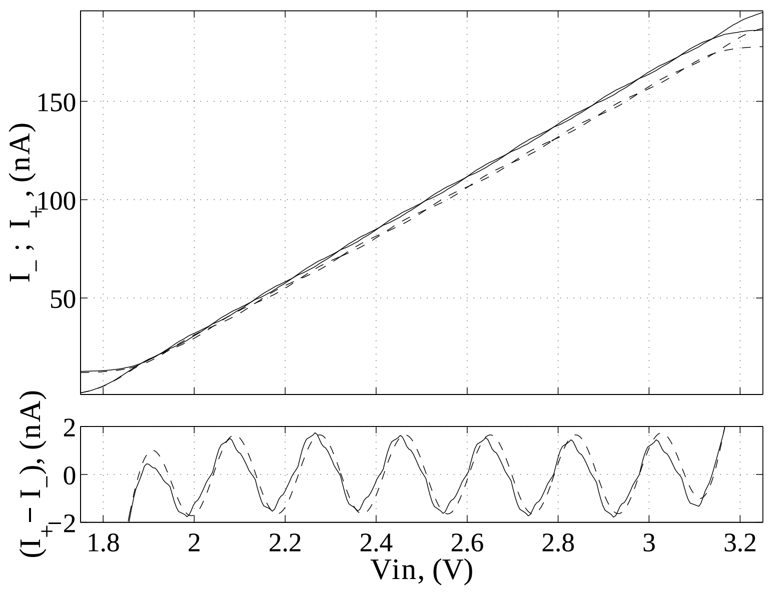


Figure 3.8: Output currents of the folding circuit of Figure 3.7. The solid line represents the SpectreS simulation results for a 0.5 μm CMOS process; the dashed line corresponds to equations (3.11) and (3.12). The time interval $\triangle t_2$=50 ns.

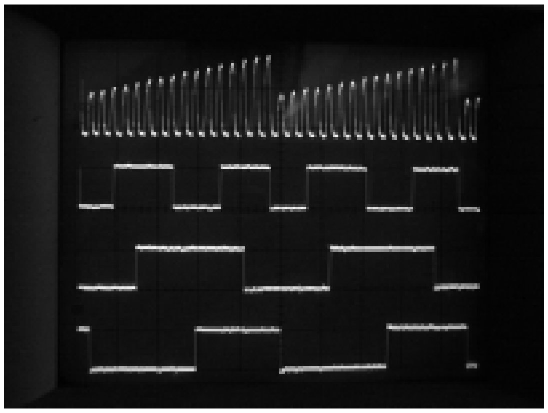


Figure 3.9: Recorded gray code flash ADC output waveforms as a function of input voltage.

and doubles its input dynamic range. An additional comparator stage, not shown, amplifies the difference $V_{o+} - V_{o-}$ and latches the result at the output.

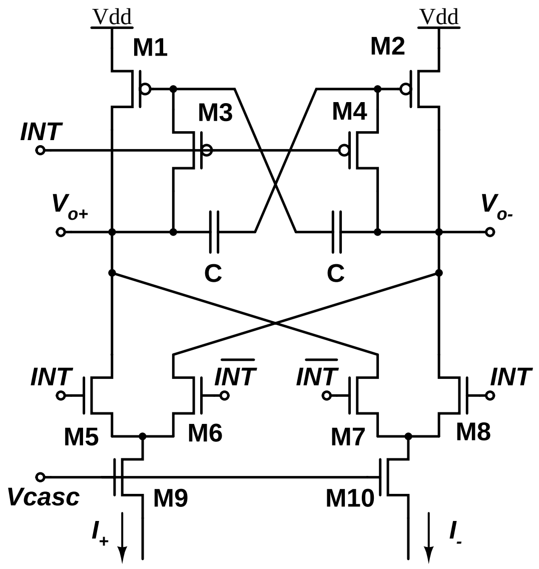


Figure 3.10: Integrating Sense Amplifier.

3.4 Experimental results

A prototype 128-channel charge-based gray code ADC was fabricated in a 0.5 μm CMOS process. The die micrograph is shown in Figure 2.5 on page 24. The chip contains, besides the parallel bank of flash ADCs, a massively parallel mixed-signal computational array (Chapter 2). The ADC bank measures 0.75 mm $\times$ 2 mm, and dissipates 76 mW of power at 128 MS/sec sampling rate. Output waveforms for a ramp input signal are presented in Figure 3.9.

3.5 Conclusions

A flash analog-to-digital converter utilizing a novel charge-based comparator and a folding circuit have been reported. Correlated double sampling comparison is performed using a log-domain integrator, implemented by a subthreshold nMOS transistor with the source coupled to a capacitor. The circuit produces a current that is a logistic function of the change in voltage on the gate, with an input-referred offset voltage that is a logarithmic function of time. Folding operation for analog-to-digital conversion is obtained by differentially combining currents from a bank of these comparators. The circuit operates in weak inversion and yields both high speed and low power. A prototype 128-channel parallel gray code analog-to-digital converter has been implemented in a 0.5 μm CMOS process, delivering 128 MS/sec at 76 mW power dissipation. *Kerneltron I* architecture presented in Chapter 2 utilizes the described flash quantizers with the overall precision of the mixed-signal computation of 6 bits. The design is suited for parallel data conversion on mixed-signal computational arrays, in particular massively parallel VLSI kernel machines with Nyquist-rate quantizers.

Chapters 4 and 5 describe another *Kerneltron* VLSI architecture where an oversampling analog-to-digital converter, replacing flash quantizers, accumulates computational array outputs in analog domain, with array inputs encoded in unary format.

Part III

Kerneltron II: Oversampling Kernel Machine

Chapter 4

Oversampling Array Processor with Delta-Sigma Quantizers

In Chapters 2 and 3 we described *Kerneltron I* processor comprised of an array of charge-mode unsigned multiply-and-accumulate cells performing matrix-vector multiplication. Array outputs were quantized by row-parallel flash analog-to-digital converters. In this chapter we present a new generation of *Kerneltron* processors, with oversampled inputs, and delta-sigma analog-to-digital converters accumulating outputs of an array of signed multiply-and-accumulate cells in analog domain.

4.1 Introduction

As in Chapter 2 we focus on the computational core of template matching operations in image processing and pattern recognition – the operation of matrix-vector multiplication (MVM) in high dimensions:

$$Y_m = \sum_{n=0}^{N-1} W_{mn} X_n \tag{4.1}$$

with N-dimensional input vector X_n, M-dimensional output vector Y_m, and $M \times N$ matrix elements W_{mn}. The matrix elements W_{mn} denote the support vectors X_{mn}, or the wavelet transformed support vectors (1.7) for convenience of notation.

The VLSI architecture for matrix-vector multiplication described in Chapters 2 and 3 employs flash analog-to-digital converters to quantize partial inner-products (2.5) which are subsequently combined in digital domain according to (2.4) to yield a digital output resolution exceeding the analog precision of the array and the quantizers. In this chapter, an oversampling analog-to-digital converter (ADC) accumulates the inner sum in (2.4) in the analog domain, with inputs encoded in unary format. This avoids the need for high-resolution flash ADCs, which are replaced with single-bit quantizers in the delta-sigma loop. High throughput is maintained by oversampling ADCs averaging across all input vector unary bit planes in a single quantization cycle. The mixed-signal oversampling processor presented in this chapter contains a fine-grain parallel array of signed multiply-and-accumulate charge-mode cells, eliminating input-output feedthrough and enhancing analog accumulation linearity.

4.2 Mixed-Signal Computation

4.2.1 Internally Analog, Externally Digital Computation

The approach combines the computational efficiency of analog array processing with the precision of digital processing and the convenience of a programmable and reconfigurable digital interface.

The digital representation is embedded in the analog array architecture, with matrix elements stored locally in bit-parallel form

$$W_{mn} = \sum_{i=0}^{I-1} 2^{-i-1} w_{mn}^{(i)} \tag{4.2}$$

and inputs presented in bit-serial fashion

$$X_n = \sum_{j=0}^{J-1} \gamma_j x_n^{(j)} \tag{4.3}$$

where the coefficients γ_j are assumed in radix two, depending on the form of input encoding used. The MVM task (4.1) then decomposes into

$$Y_m = \sum_{n=0}^{N-1} W_{mn} X_n = \sum_{i=0}^{I-1} 2^{-i-1} Y_m^{(i)} \tag{4.4}$$

with MVM partials

$$Y_m{}^{(i)} = \sum_{j=0}^{J-1} \gamma_j Y_m^{(i,j)}, \tag{4.5}$$

and

$$Y_m{}^{(i,j)} = \sum_{n=0}^{N-1} w_{mn}^{(i)} x_n^{(j)} \ . \tag{4.6}$$

The binary-binary partial products (4.6) are conveniently computed and accumulated, with zero latency, using an analog MVM array [21], [22], [23], [24]. For this purpose we developed a 1-bit signed multiply-and-accumulate CID/DRAM cell.

4.2.2 CID/DRAM Cell and Array

The unit cell in the analog array combines a CID (charge injection device [33]) computational element [23, 24] with a DRAM storage element. The cell stores one bit of a matrix element $w_{mn}{}^{(i)}$, performs a one-quadrant binary-unary (or binary-binary) multiplication of $w_{mn}{}^{(i)}$ and $x_n{}^{(j)}$ in (4.6), and accumulates the result across cells with common m and i indices. The circuit diagram and operation of the cell are given in Figure 2.3 on page 20. It performs a non-destructive computation since the transferred charge is sensed capacitively at the output. An array of these cells thus performs (unsigned) binary-unary multiplication (4.6) of matrix $w_{mn}{}^{(i)}$ and vector $x_n{}^{(j)}$ yielding $Y_m{}^{(i,j)}$, for values of i in parallel across the array, and values of j in sequence over time as described in detail in Chapter 2.

To improve linearity and to reduce sensitivity to clock feedthrough, we use differential encoding of input and stored bits in the CID/DRAM architecture using twice the number of columns and unit cells as shown in Figure 4.1. This amounts to exclusive-OR (XOR), rather than AND, multiplication on the analog array, using signed, rather than unsigned, binary values for inputs and weights, $x_n{}^{(j)} = \pm 1$ and $w_{mn}{}^{(i)} = \pm 1$.

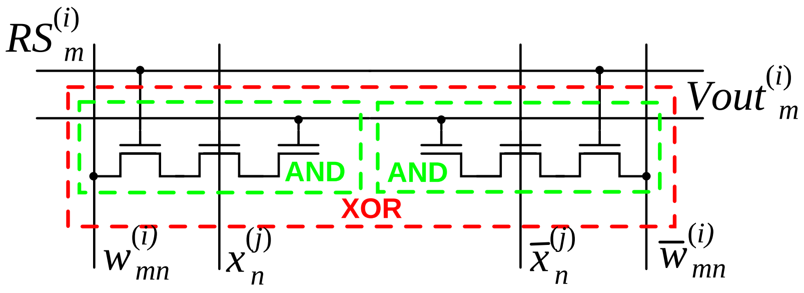


Figure 4.1: Two charge-mode AND cells configured as an exclusive-OR (XOR) multiply-and-accumulate gate.

In principle, the MVM partials (4.6) can be quantized by a bank of flash ADCs, and the results accumulated in the digital domain according to (4.5) and (4.4) to yield a digital output resolution exceeding the analog precision of the array and the quantizers as described in Chapter 2. In the architecture presented here, an oversampling ADC accumulates the sum (4.5) in the analog domain, with inputs encoded in unary format ($\gamma_i = 1$). This avoids the need for high-resolution flash ADCs, which are replaced with single-bit quantizers in the delta-sigma loop.

4.3 Oversampling Mixed-Signal Array Processing

The precision of computation is limited by the resolution of the analog-to-digital converters (ADC) digitizing the analog array outputs. The conventional delta-sigma ($\Delta\Sigma$) ADC design paradigm allows to reduce requirements on precision of analog circuits to attain high resolution of conversion, at the expense of bandwidth. In the presented architecture a high conversion rate is maintained by combining delta-sigma analog-to-digital conversion with oversampled encoding of the digital inputs, where the delta-sigma modulator integrates the partial multiply-and-accumulate outputs (4.6) from the analog array according to (4.5).

4.3.1 Array Architecture

Figure 4.2 depicts one row of matrix elements W_{mn} in the $\Delta\Sigma$ oversampling architecture, encoded in $I = 4$ bit-parallel rows of CID/DRAM cells. One bit of a unary-coded input vector is presented each clock cycle, taking J clock cycles to complete a full computational cycle (4.1). The data flow is illustrated for a digital input series ${x_j}^{(n)}$ of $J = 16$ unary bits.

Over J clock cycles, the oversampling ADC integrates the partial products (4.6), producing a decimated output

$$Q_m^{(i)} \approx \sum_{j=0}^{J-1} \gamma_j Y_m^{(i,j)} \ , \tag{4.7}$$

where $\gamma_j = 1$ for unary coding of inputs. Decimation for a first-order delta-sigma modulator is achieved using a binary counter.

4.3.2 Row-parallel $\Delta\Sigma$ Algorithmic ADC

Higher precision can be obtained in the same number of cycles J by using a higher-order delta-sigma modulator topology. However this drastically increases the implementation complexity. Instead, we use a modified topology shown in Figure 4.3 that resamples the residue of the integrator after initial conversion. A sample-and-hold resamples the residue voltage of the integrator and presents it to the modulator input for continued conversion at a finer scale. The principle is analogous to extended counting [51] but avoids additional hardware by reusing the same $\Delta\Sigma$ modulator to quantize the residue.

Similar to residue resampling in an algorithmic (or cyclic) ADC, for each resampling the scale of conversion subranges to the LSB level of the previous conversion. For a first-order incremental $\Delta\Sigma$ ADC [52], resampling of the residue scales the range by a factor L, where L is the number of modulation cycles. If L is of radix two, *i.e.*, $L = 2^\ell$, then the subranging is conveniently accomplished in the architecture of Figure 4.3 by shifting the bits in the decimating counter by ℓ positions for every resampling of the residue.

Every resampling improves the output resolution by a factor L, or ℓ bits, limited by

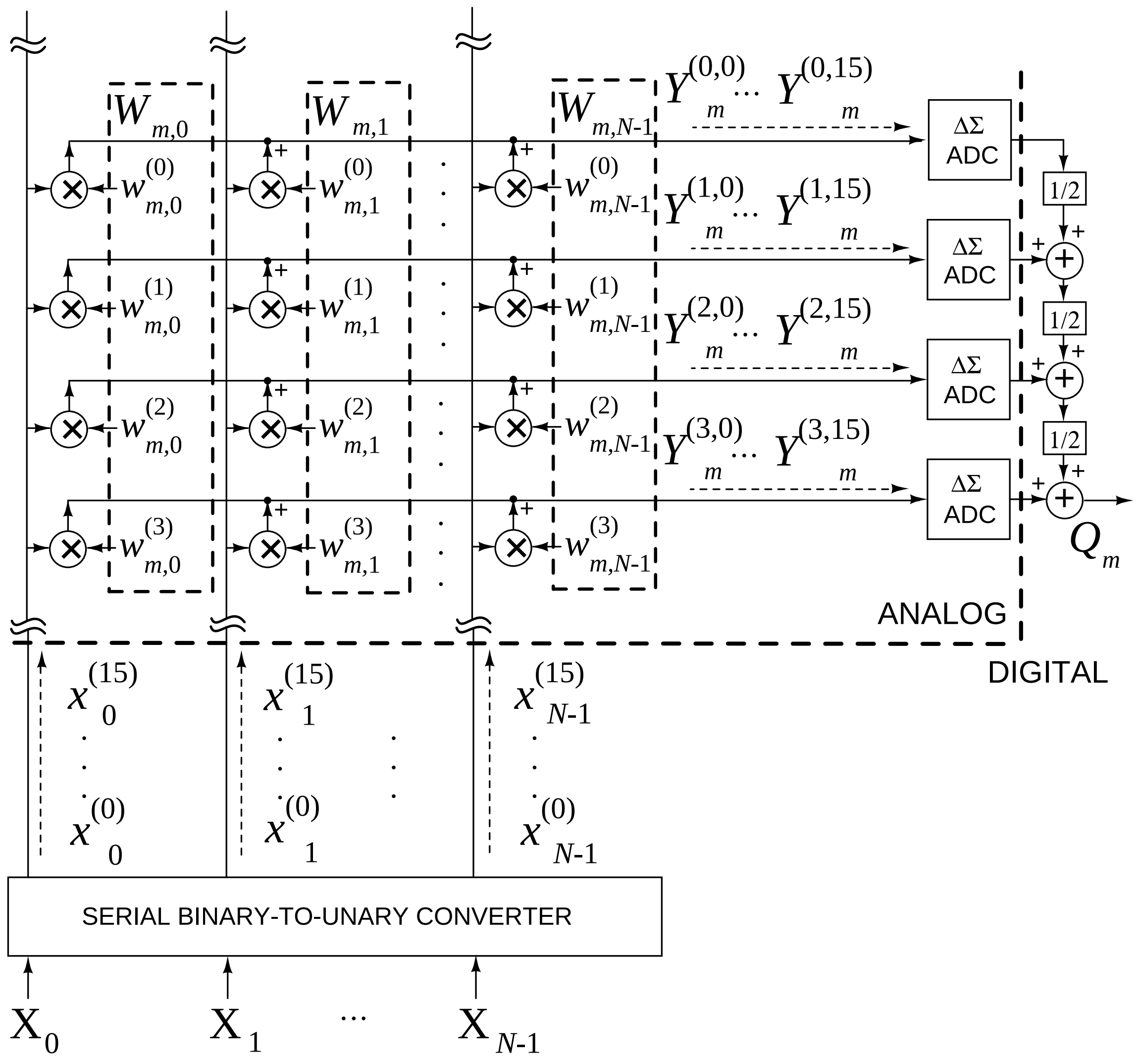


Figure 4.2: Block diagram of one row in the matrix with binary encoded elements $w^{(i)}{}_{mn}$, for a single m and unary encoded inputs. Data flow of bit-serial inputs $x^{(j)}{}_n$ and corresponding partial product outputs $Y^{(i,j)}{}_m$, with $J = 16$ bits. The full product for a single row $Y^{(i)}{}_m$ is accumulated and quantized by a delta-sigma ADC. The final product is constructed in the digital domain according to (4.4).

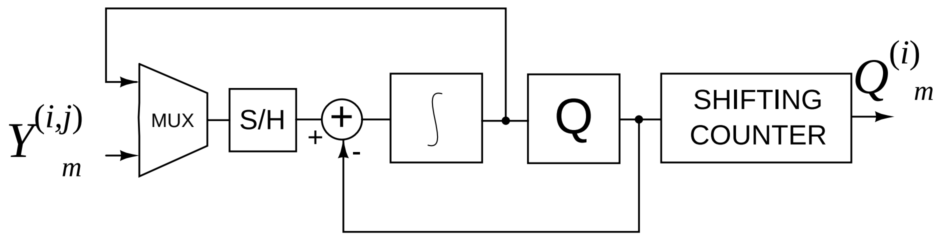


Figure 4.3: Block diagram of $\Delta\Sigma$ algorithmic ADC with residue resampling, including decimator. One ADC is provided for each row in the array architecture.

noise and mismatch in the implementation. The effect of capacitance mismatch is minimized by using a ratio-insensitive scheme for resampling the residue, shown in Figure 5.2 on page 65. The presented scheme is equivalent to algorithmic A/D conversion, but avoids interstage gain errors without the need for precisely ratioed analog components.

The resampling of the residue in the oversampled ADC can be combined with correspondingly rescaling the coefficients γ_j in the input encoding. In principle, higher resolution digital inputs can be presented by unary encoding bits in groups of ℓ, each covering L modulation cycles of the subranging oversampled ADC. In the example of Figure 4.2, only the first 4 bits are unary encoded and presented in the first algorithmic cycle, with $L = 16$. With a single resampling of the residue, the $\Delta\Sigma$ modulator obtains $2\ell = 8$ bit effective resolution in $2L = 32$ cycles.

Chapter 5 contains a more rigorous analysis of circuits, architecture and measured performance of delta-sigma algorithmic analog-to-digital converters.

4.4 Experimental Results

A *Kerneltron II* prototype was integrated on a $3 \times 3\ \mathrm{mm}^2$ die and fabricated in 0.5 μm CMOS technology. The chip contains an array of 256×128 CID/DRAM cells, and a row-parallel bank of 128 algorithmic $\Delta\Sigma$ ADCs. Figure 4.4 depicts the micrograph and system floorplan of the chip.

The processor interfaces externally in digital format. Two separate shift registers load

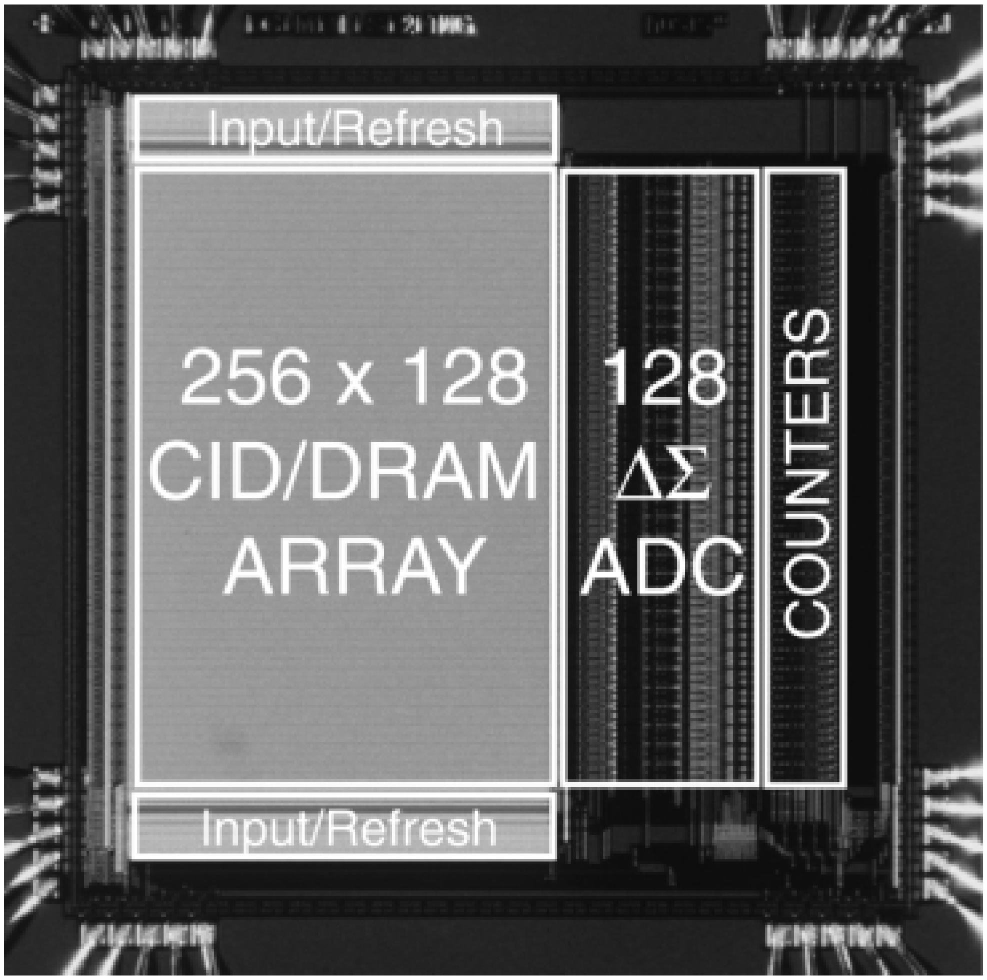


Figure 4.4: Micrograph of a *Kerneltron II* prototype, containing an array of 256×128 CID/DRAM cells, and a row-parallel bank of 128 $\Delta\Sigma$ algorithmic ADCs. Die size is $3\ \mathrm{mm} \times 3\ \mathrm{mm}$ in 0.5 μm CMOS technology.

Table 4.1: Measured performance of *Kerneltron II*

Technology	0.5 μm CMOS
Area	3mm $\times$ 3mm
Power	5.9 mW
Supply Voltage	5 V
Dimensions	256 inputs $\times$ 128 templates
Throughput	6.5 GMACS
Output Resolution	8-bit

the templates (support vectors) along columns of the DRAM array and download input data. Integrated refresh circuitry periodically updates the charge stored in the array to compensate for leakage. Vertical bit lines extend across the array, with two rows of sense amplifiers at the top and bottom of the array. The refresh alternates between even and odd columns, with separate select lines. Stored charge corresponding to matrix element values can also be read and shifted out from the chip for test purposes. All of the supporting digital clocks and control signals are generated on-chip.

Figure 4.5 shows the observed linearity of the computational array, configured differentially for signed (XOR) multiplication. The case shown is where all complementary weight storage elements are actively set, and an alternating sequence of bits in blocks N is shifted through the input register.[1] For every shift in the input register, a computation is performed and the result is observed on the output sense line. The array dissipates 3.3 mW for a 10 μs cycle time. The bank of $\Delta\Sigma$ ADCs dissipates 2.6 mW yielding a combined conversion rate of 12.8 Msamples/s. Table 4.1 summarizes the measured performance.

4.5 System-Level Performance

Figure 4.6 compares template matching performed by a floating point processor and by the *Kerneltron*, illustrating the effect of quantization and limited precision in the analog array architecture. An 'eye' template was selected as a 16×16 fragment from the

[1] ${w_{mn}}^{(i)} = 1$; ${x_n}^{(j)} = 1$ for $n = 1, \ldots N$; and ${x_n}^{(j)} = -1$ for $n = N+1, \ldots 2N$.

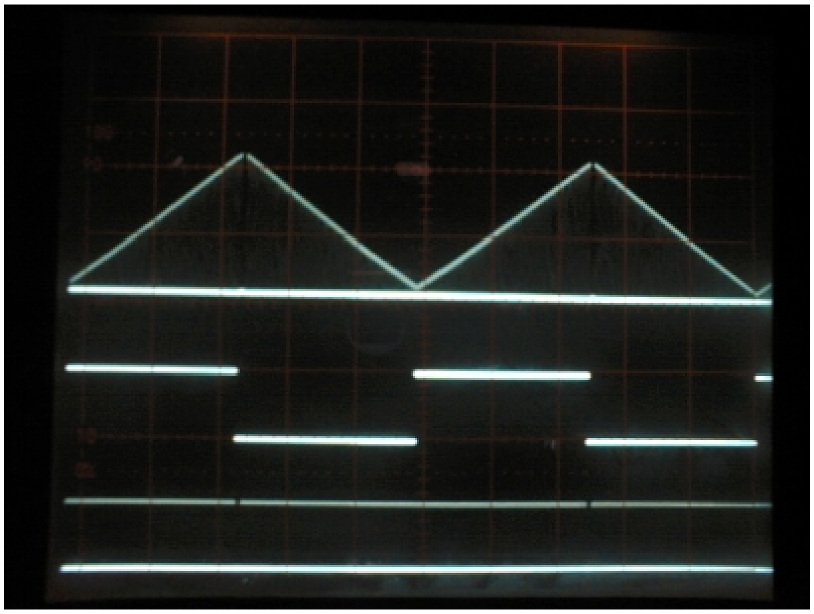
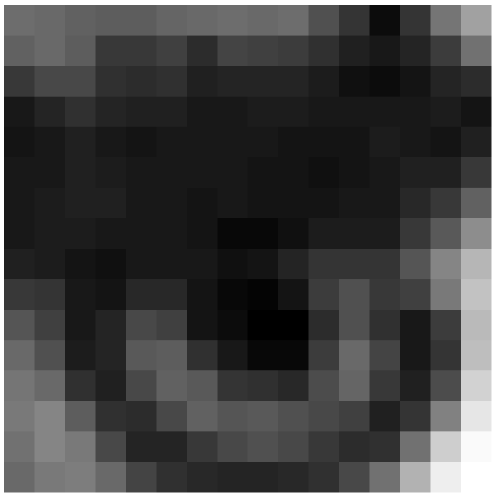


Figure 4.5: Measured linearity of the computational array configured for signed multiplication on each cell (XOR configuration). Two cases are shown: binary weight storage elements are all actively charged, and all discharged. Waveforms shown are, *top to bottom*: the analog voltage output on the sense line; input data (in common for both input and weight shift register); and input shift register clock.

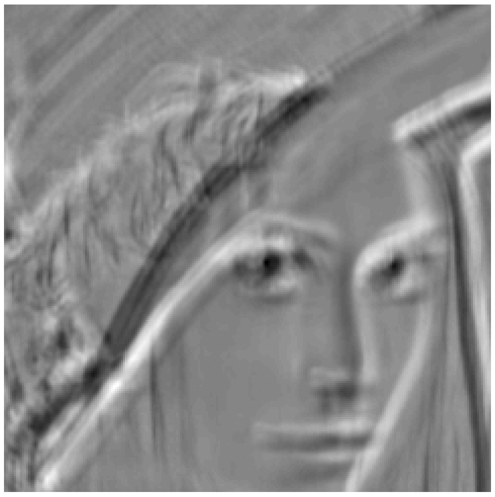
Lena image, yielding a 256-dimensional vector. Figure 4.6 (c) depicts the two-dimensional cross-correlation (inner-products over a sliding window) of the 8-bit image with the 8-bit template computed with full precision. The same computation performed by the *Kerneltron,* with 4-bit quantization of the image and template and 8-bit quantization of the output, is given in Figure 4.6 (d). Differences are relatively small, and both methods return peak inner-product values (top matches) at both eye locations in the image.[2] The template matching operation is representative of a support vector machine that combines nonlinearly transformed inner-products to identify patterns of interest.

[2]The template acts as a spatial filter on the image, leaking through spectral components of the image at the output. The *Lena* image was mean-subtracted.

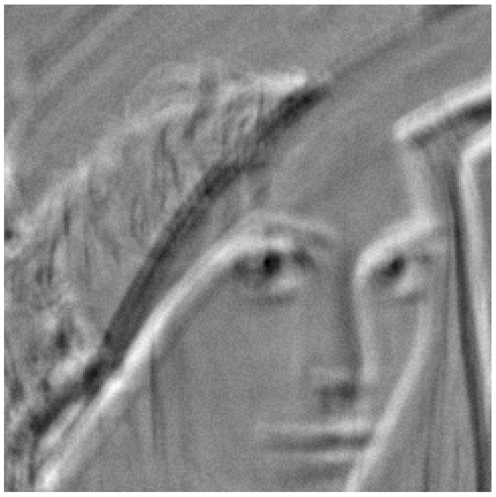
(a)



(b)



(c)



(d)

Figure 4.6: Cross-correlation of fragments of *Lena (a)* and the eye template *(b)* computed by a 32-bit floating point processor with 8-bit encoded inputs *(c)* and by *Kerneltron* with 8-bit quantization and 4-bit encoded inputs *(d)*.

4.6 Conclusions

The oversampling *Kerneltron II* architecture for parallel matrix-vector multiplication has been presented. An internally analog, externally digital architecture offers the best of both worlds: the density and energetic efficiency of an analog VLSI array, and the convenience and versatility of a digital interface. The three-transistor CID/DRAM unit cell combines single-bit dynamic storage, binary multiplication, and zero-latency analog accumulation. Differential configuration of two such cells implements a signed multiply-and-accumulate unit avoiding input-output feedthrough and improving linearity of analog summation. Delta-sigma analog-to-digital conversion of the analog array outputs performed in synchrony with oversampled unary coding of the digital inputs relaxes precision requirements in the quantization.

Additional gains in precision could be obtained by exploiting binomial statistics of binary terms in the analog summation (4.6) as described in Chapter 7. In the present scheme, this would entail stochastic encoding of the digital inputs prior to unary oversampled encoding.

A 256 $\times$ 128 cell prototype was fabricated in 0.5 μm CMOS. The combination of analog array processing, oversampled input encoding, and $\Delta\Sigma$ algorithmic analog-to-digital conversion delivers a computational throughput of over 1 GMACS per mW of power, while maintaining 8-bit effective digital resolution.

Chapter 5

Delta-Sigma Algorithmic Analog-to-Digital Conversion

In Section 4.3 of the previous chapter we gave a brief overview of row-parallel quantizers in the oversampling CID/DRAM computational array. This chapter contains a more rigorous analysis of the architecture and circuits of delta-sigma algorithmic analog-to-digital converters.

5.1 Introduction

Delta-sigma ($\Delta\Sigma$) modulation has emerged as the architecture of choice for high-resolution analog-to-digital (A/D) conversion using low-precision analog components [39]. The increased resolution comes at the expense of reduced conversion bandwidth or increased clock speed due to oversampling, and increased digital complexity to decimate the modulator output stream. For very low bandwidth applications, lowest digital complexity is achieved with a first-order $\Delta\Sigma$ incremental converter [53], where a counter implements a rectangular decimation filter. Higher-order $\Delta\Sigma$ incremental converters [52] are capable of attaining higher conversion bandwidth, using additional analog and digital circuitry. Alternatively, higher bandwidth can be obtained from a first-order incremental converter by further refining the modulation residue on the integrator at the end of conversion using a Nyquist A/D converter [54, 51, 55]. The principle is similar to dual-quantization oversampled converters [56, 57], except the second quantizer operates at the conversion rate and requires no decimation.

The presented architecture uses the same first-order modulator, with virtually no overhead in analog and digital hardware, to incrementally convert the modulation residue of preceding incremental conversion. By using the same signal path in ratio-insensitive manner, precise matching between multiple quantization results is obtained. Matching is a concern in the precision of multiple-quantization oversampled data converters [58], usually requiring compensation in the digital domain [59]. The presented scheme is similar to algorithmic A/D conversion, but avoids interstage gain errors when precisely ratioed analog components are not available. Very high integration density can be achieved by virtue of the simple modulator and decimator architecture.

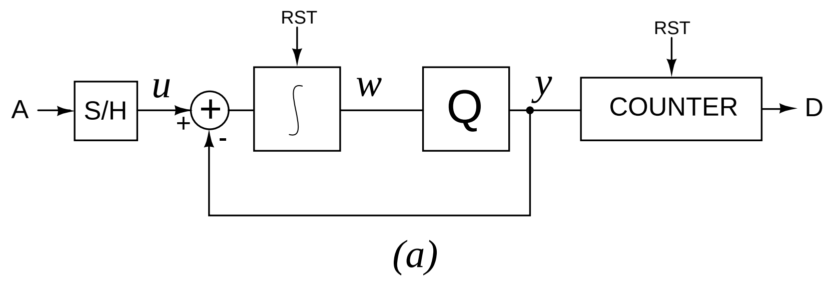
5.2 $\Delta\Sigma$ Incremental A/D Conversion

For clarity of exposition we start the formulation with that of the first-order incremental A/D converter [53], depicted in Figure 5.1*(a)*. A first-order $\Delta\Sigma$ modulator converts an analog sequence $u[i]$ into a bitstream $y[i]$, using a 'resetable' (RST) analog accumulator

$$
\begin{aligned}
w[0] &= 0 && (5.1)\\
w[i+1] &= w[i] + \alpha\,(u[i] - y[i])\,, \\
&\qquad\qquad i = 0, \dots N-1 && (5.2)\\
w[N+1] &= w[N] - \alpha\, y[N] && (5.3)
\end{aligned}
$$

and a single-bit quantizer

$$
\begin{aligned}
y[0] &= -1 && (5.4)\\
y[i] &= \mathrm{sign}(w[i])\,, \quad i = 1, \dots N\,. && (5.5)
\end{aligned}
$$



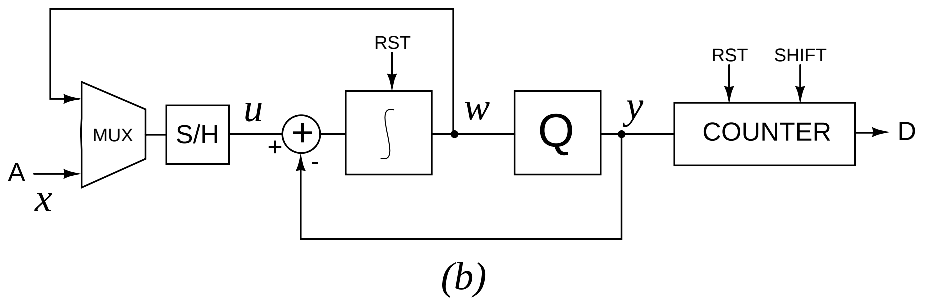


Figure 5.1: *(a)* First-order $\Delta\Sigma$ incremental A/D converter. *(b)* $\Delta\Sigma$ algorithmic A/D converter, with residue resampling across the accumulator, and shifting counter for the decimator.

A binary counter accumulates the bits[1] $y[i]$ to produce a decimated output. The rectangular decimation window, and initial reset of the accumulator, avoid tones in the quantization noise spectrum at DC input that are characteristic of a conventional first-order $\Delta\Sigma$ modulator with lowpass decimation filter [53]. The quantization error (conversion residue) is directly given by the final accumulator value $w[N+1]$, as verified by summing (5.2) and (5.3) over i:

$$\sum_{i=0}^{N} y[i] = \sum_{i=0}^{N-1} u[i] - \frac{1}{\alpha} w[N+1] \; . \tag{5.6}$$

For an input $u[i]$ within the conversion range $[-1, 1]$ (in dimensionless units), $w[N+1]$ in (5.3) is bounded by the range $[-\alpha, \alpha]$, and a worst-case resolution of $\log_2(N)$ bits is warranted[2].

Higher resolution at lower oversampling N can be obtained using higher-order incre-

[1]$y = -1$ corresponds to logic 0 for notational convenience.

[2]Note that the last modulation cycle (5.3) contributes one full bit of resolution, since $w[i]$ for $i \leq N$ in (5.2) is bounded by 2α in amplitude. The extra zero-input modulation cycle (5.3) can be avoided by quantizing $w[i] + \alpha u[i]$ instead of $w[i]$ in (5.5).

mental conversion [52]. A lower-complexity alternative is presented next.

5.3 $\Delta\Sigma$ Algorithmic A/D Conversion

The conversion residue $\frac{1}{\alpha}w[N+1]$ in (5.6) is converted further into digital form to obtain higher resolution. As in other multiple-quantization oversampled converters [52], [54], [51], [55], [56], [57], gain mismatch between quantization signal paths is a limiting factor in the precision available [58]. Ratio-insensitive matching is achieved by resampling the residue through the same signal path as used for accumulation (5.2), and employing the same modulator to convert the residue.

Assume a constant input (or its average) x presented to the incremental converter for N initial cycles, $u[i] = x$, $i = 0, \ldots N-1$,

$$\sum_{i=0}^{N} y[i] = N\,x - \frac{1}{\alpha}w[N+1]\,. \tag{5.7}$$

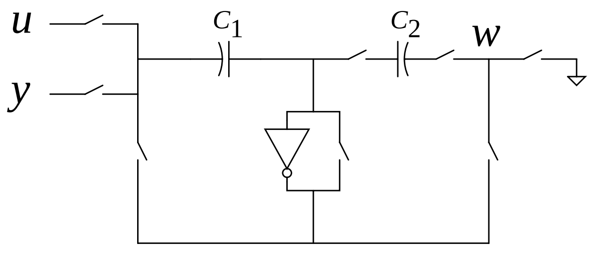
At the end of conversion, the residue $w[N+1]$ is fed back to the input for subsequent incremental conversion over N' additional cycles, $u'[i] = \beta w[N+1]$, $i = 0, \ldots N'-1$, where β represents the residue resampling gain. Thus

$$\sum_{i=0}^{N'} y'[i] = N'\beta\,w[N+1] - \frac{1}{\alpha}w'[N'+1] \tag{5.8}$$

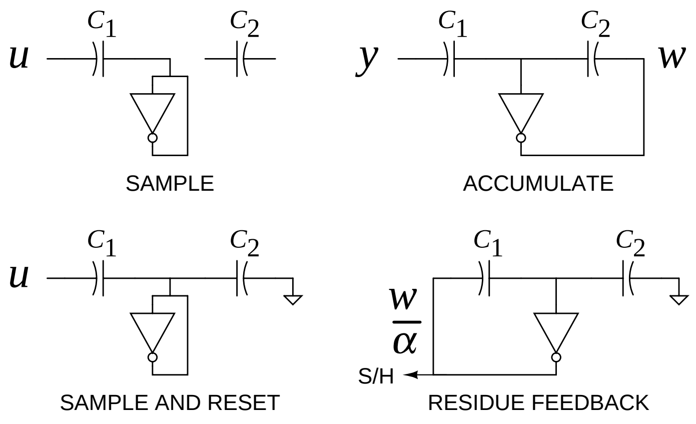
which under the matching condition $\alpha\beta \equiv 1$ combines with (5.7) to cancel the first residue $w[N+1]$:

$$N'\sum_{i=0}^{N} y[i] + \sum_{i=0}^{N'} y'[i] = NN'\,x - \frac{1}{\alpha}w'[N'+1]. \tag{5.9}$$

The left-hand side of (5.9) is readily available in digital form, and the right-hand terms conform to (5.6) with effective NN'-level resolution. Therefore, the residue conversion (5.8) enhances the resolution of (5.7) by an extra $\log_2(N')$ bits. The algorithmic recursion can be continued to produce $\log_2(N)$ bits for every conversion of the preceding residue over N incremental cycles. The recursion corresponds to a radix-N algorithmic A/D converter, but without the need for N-ratioed analog components.



(a)



(b)

Figure 5.2: Invertible, resetable analog accumulator. *(a)* Circuit diagram. *(b)* Operational modes.

The matching condition $\alpha\beta \equiv 1$ for ratio-insensitive operation is met using an invertible circuit topology for the analog accumulator, described next.

5.4 Implementation

The hardware complexity of the $\Delta\Sigma$ algorithmic A/D converter, depicted in Figure 5.1*(b)*, is essentially identical to that of the $\Delta\Sigma$ incremental converter in Figure 5.1*(a)*. The accumulator is extended to sample the residue in ratio-insensitive manner, and the counter is extended to shift bits in between residue conversions for proper scaling in the decimation.

5.4.1 Invertible, Resetable Analog Accumulator

Critical to attaining ratio-insensitive sampling of the residue is the design of the accumulator. A simple circuit achieving multiple objectives is depicted in Figure 5.2 *(a)*, with different modes of operation illustrated in Figure 5.2 *(b)*. The circuit is shown with minimum number of components using an inverting amplifier, but can be directly extended to differential designs for higher resolution. Correlated double sampling (CDS) in the accumulation of the difference $u[i] - y[i]$ according to (5.2) offers the advantage of $1/f$ noise and offset cancellation. The switched-capacitor accumulator has ratio-dependent gain $\alpha = C_1/C_2$ that is prone to mismatch; however this mismatch is immaterial in the operation of the first-order modulator with single-bit quantizer. For ratio-insensitive sampling of the residue, the signal and feedback path of the accumulator is inverted by interchanging the C_1 and C_2 components in the circuit topology, shown in Figure 5.2 *(b)*. As a result, the matching condition $\alpha\beta \equiv 1$ between (5.7) and (5.8) is satisfied independent of C_1 and C_2, to a precision limited by the finite gain of the amplifier, and noise and charge injection in the sampling.

5.4.2 Decimating Shifting Counter

For every algorithmic iteration, the preceding decimated output needs to be scaled by a factor N, the number of incremental cycles in the residue conversion, prior to continued counting. It is particularly convenient to assume N to be radix-2, so that the scaling is simply performed by a binary shift in the decimation count, as suggested in Figure 5.1*(b)*. A shift register is readily incorporated in a binary counter, with little overhead in the circuit complexity of the implemented decimator.

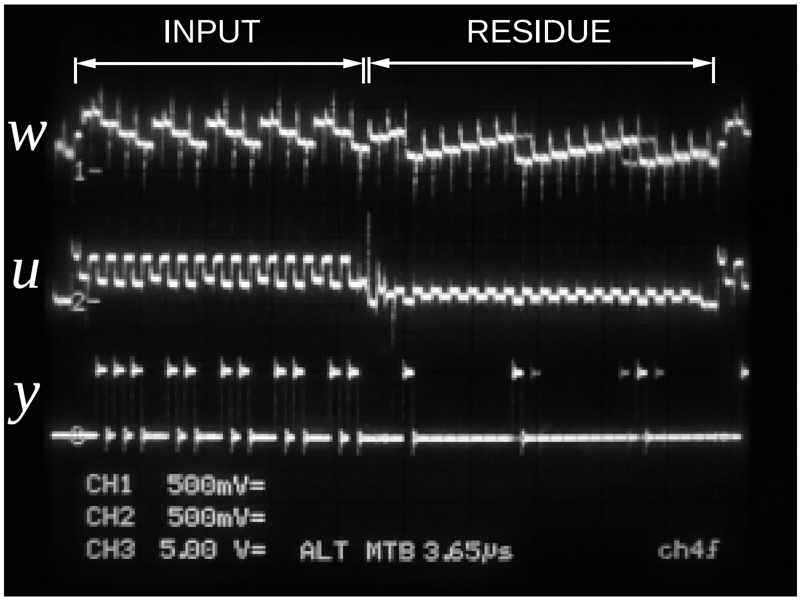


Figure 5.3: Observed waveforms for two-step $\Delta\Sigma$ algorithmic A/D conversion. *Top:* Integrator voltage $w[i]$. *Center:* Sample-and-hold voltage $u[i]$. *Bottom:* Output bits $y[i]$ prior to decimation.

5.5 Experimental Results from Integrated A/D Array

A bank of 128 $\Delta\Sigma$ algorithmic converters has been implemented as part of *Kerneltron II* mixed-signal processor described in Chapter 4. A 256×128 array performs externally digital matrix-vector multiplication using internally analog elements for very large energetic efficiency (10^{12} multiply-accumulates per Watt of power). The array core performs analog accumulation of binary-binary partial products, requiring row-parallel A/D conversion for each of the 128 output vector components. The micrograph of the integrated system is shown in Figure 4.4. Each of the 128 A/D channels measure 14 μm by 850 μm.

To maximize integration density, the analog path of the $\Delta\Sigma$ algorithmic architecture of Figure 5.1*(b)* uses single-stage cascoded *n*MOS inverting amplifiers throughout, with nominal gain of -300. Capacitances C_1 and C_2 are nominally 0.25 pF and 0.5 pF, respectively, with $\alpha = 0.5$. Sample-and-hold and comparator blocks are implemented in standard switched-capacitor circuitry, with CDS $1/f$ noise and offset compensation. Dynamic logic

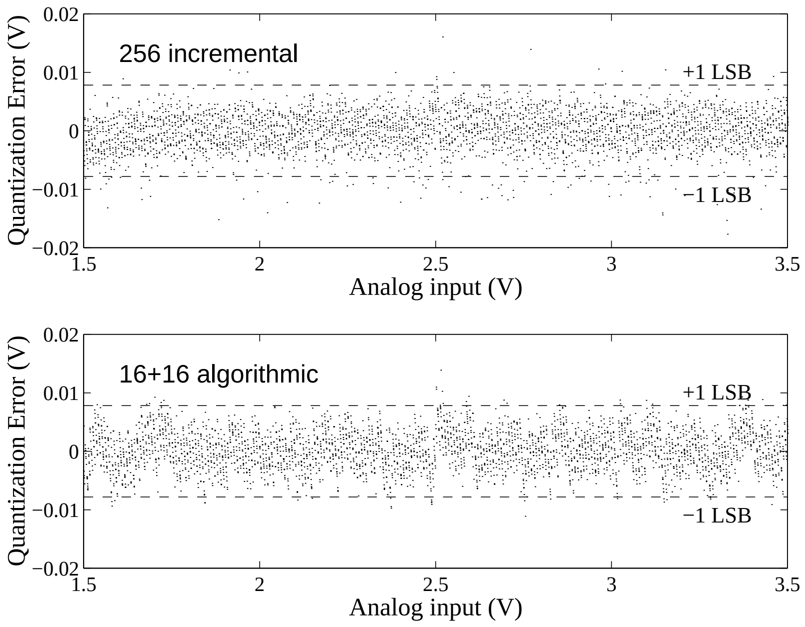


Figure 5.4: Integral quantization residue, recorded from a single channel of the VLSI A/D array in Figure 4.4, configured for 8-bit conversion. *Top:* $\Delta\Sigma$ incremental conversion ($N = 256$). *Bottom:* $\Delta\Sigma$ algorithmic A/D conversion ($N = 16$, 2-step).

implements binary counter and registers.

Example conversion waveforms from the fabricated array are illustrated in Figure 5.3. Least-squares fits of integral quantization error observed over one channel of the array, configured both for 8-bit incremental and algorithmic $\Delta\Sigma$ conversion, are within the LSB level as shown in Figure 5.4. However, incremental conversion requires $2^8 + 1 = 257$ cycles, while 2-step algorithmic conversion takes $2 \times (2^4 + 1) = 34$ cycles. With a 300 ns clock in 2-step 8-bit algorithmic mode, the 128-channel A/D bank delivers 12.8 Msamples/s at 2.6 mW power dissipation, including decimation.

Resolutions larger than 10-bit require higher-gain amplifiers at the cost of decreased integration density and increased power dissipation. In large-scale kernel machine implementations (as well as in digital imaging and other integrated sensing modalities), maintaining high spatial resolution is usually a more stringent requirement than increasing amplitude resolution.

Note that matching *is* at stake even at low (8-bit) resolution. Precise matching between capacitors with up to 12-bit uncalibrated precision can be achieved through careful layout in centroid geometry, but not within dimensions pitch-matched to a 45 λ (14 μm) CID/DRAM cell (Sections 2.3.1 and 4.2.2).

5.6 Conclusion

The delta-sigma algorithmic analog-to-digital converter used for quantization of analog outputs of *Kerneltron II* CID/DRAM computational array (Chapter 4) has been presented. Delta-sigma modulation for analog-to-digital conversion resolves a number of bits logarithmic in the number of modulation cycles, and linear in modulation order. As an alternative to higher-order noise shaping, we presented an algorithmic scheme that iteratively resamples the modulation residue, by feeding the integrator output back to the input. This yields a bit resolution linear in the number of cycles, similar to an algorithmic analog-to-digital converter. The scheme simplifies the design of the digital decimator to a single shifting counter, and avoids interstage gain errors in conventional algorithmic analog-to-digital converters.

The proposed technique combines advantages of $\Delta\Sigma$ modulation and algorithmic (cyclic) A/D conversion in a single, simple architecture. Both are included as special limiting cases: a single algorithmic iteration reduces to $\Delta\Sigma$ incremental conversion, and $N = 2$ cycles per iteration yield a ratio-insensitive form of algorithmic A/D conversion.

The reduced hardware complexity and improved component tolerance of the architecture offer a major advantage in integrated applications calling for large-scale parallel quantization at low to medium resolution (8 to 12 bits) but very high integration densities, with a potential for extremely large combined quantization throughputs (Gsample/s range at mW power levels). Bandwidth can be traded for resolution in reconfigurable manner through external control of clock waveforms. Ratio-insensitive matching makes the architecture especially suited for applications of integrated digital acquisition in spatial sensor arrays.

Experimental results from *Kerneltron II* processor containing a bank of 128 delta-sigma algorithmic analog-to-digital converters show the utility of the design for large-scale parallel quantization in hybrid analog-digital kernel machines implementation.

Part IV

Algorithmic Enhancements

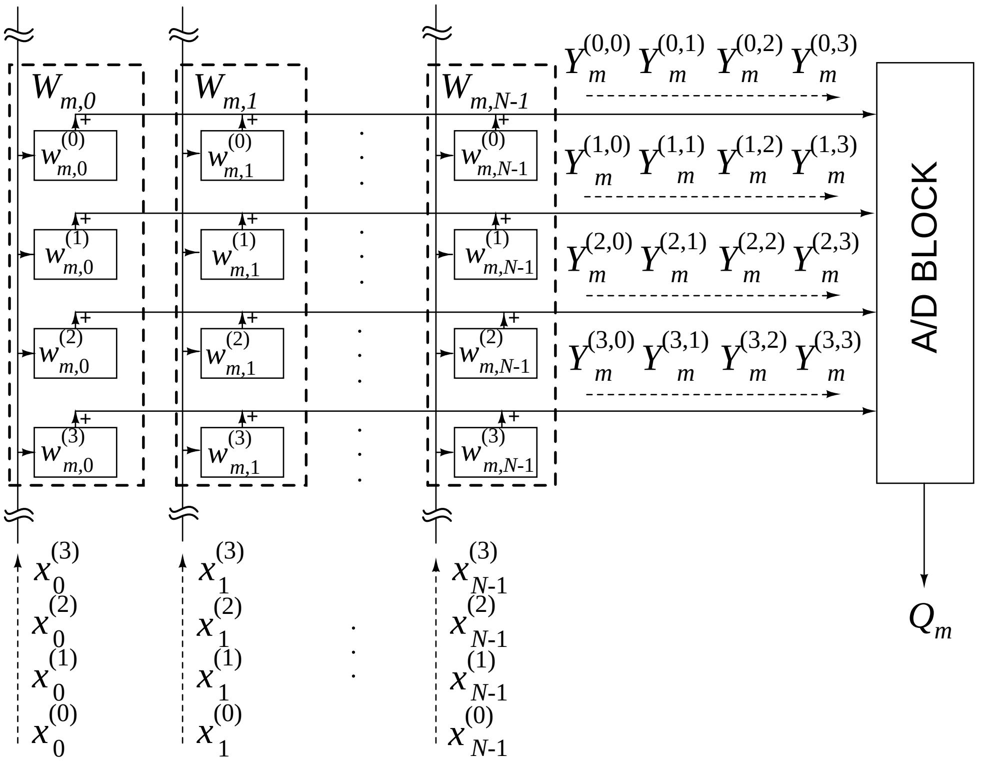
Chapter 6

Resolution Enhancement in Nyquist-rate Kernel Machines

In Chapter 2 we presented an internally analog, externally digital architecture for dedicated VLSI kernel-based array processing that outperforms purely digital approaches by several orders of magnitude in throughput, density and energy efficiency. Section 2.4 introduced the subject of computation resolution enhancement in a *Kerneltron* architecture employing low-resolution row-parallel flash analog-to-digital converters (ADC). In this chapter we extend this analysis by utilizing a redundant data representation scheme to further enhance digital resolution of the computation, and introduce other Nyquist-rate quantization schemes offering advantages of lower power and area resources utilization.

6.1 Introduction

Redundant data representation techniques trade an overhead in inexpensive resources for gains in a costly critical parameter in a system. Radix-less-than-2 data encoding has been commonly used in mixed-signal VLSI systems to relax requirements on errors introduced by system components, such as linearity or gain errors [60], [61]. Non-radix-2 designs typically require only a moderate increase in a number of homogeneous stages in



$W_{m,0}$
$W_{m,1}$
$W_{m,N-1}$
$w_{m,0}^{(0)}$ $w_{m,0}^{(1)}$ $w_{m,0}^{(2)}$ $w_{m,0}^{(3)}$
$w_{m,1}^{(0)}$ $w_{m,1}^{(1)}$ $w_{m,1}^{(2)}$ $w_{m,1}^{(3)}$
$w_{m,N-1}^{(0)}$ $w_{m,N-1}^{(1)}$ $w_{m,N-1}^{(2)}$ $w_{m,N-1}^{(3)}$
$Y_m^{(0,0)}$ $Y_m^{(0,1)}$ $Y_m^{(0,2)}$ $Y_m^{(0,3)}$
$Y_m^{(1,0)}$ $Y_m^{(1,1)}$ $Y_m^{(1,2)}$ $Y_m^{(1,3)}$
$Y_m^{(2,0)}$ $Y_m^{(2,1)}$ $Y_m^{(2,2)}$ $Y_m^{(2,3)}$
$Y_m^{(3,0)}$ $Y_m^{(3,1)}$ $Y_m^{(3,2)}$ $Y_m^{(3,3)}$
A/D BLOCK
Q_m
$x_0^{(3)}$ $x_0^{(2)}$ $x_0^{(1)}$ $x_0^{(0)}$
$x_1^{(3)}$ $x_1^{(2)}$ $x_1^{(1)}$ $x_1^{(0)}$
$x_{N-1}^{(3)}$ $x_{N-1}^{(2)}$ $x_{N-1}^{(1)}$ $x_{N-1}^{(0)}$

Figure 6.1: Block diagram of one row in the matrix with binary encoded elements ${w_{mn}}^{(i)}$, for a single m and with $I = 4$ bits ($I' = 4$ in radix-less-than-2 case). Data flow of bit-serial inputs ${x_n}^{(j)}$ and corresponding partial outputs ${Y_m}^{(i,j)}$, with $J = 4$ bits ($J' = 4$ in radix-less-than-2 case).

a multi-stage architecture or in a number of time steps in a cyclic implementation. The benefit is a boosted tolerance to imperfections of analog VLSI components resulting in significant increases in costly precision.

In Chapter 2 we described an internally analog, externally digital architecture for parallel matrix-vector multiplication (MVM). A three-transistor unit cell combines a single-bit dynamic random-access memory (DRAM) and a charge injection device (CID) binary multiplier and analog accumulator. Digital multiplication of variable resolution is obtained with bit-serial inputs and bit-parallel storage of matrix elements, by combining quantized outputs from multiple rows of cells over time as we illustrate again in Figure 6.1.

In this chapter we use non-radix-2 data encoding to enhance the precision of MVM computation on mixed-signal array processors which employ flash as well as other Nyquist-rate ADCs. In Section 6.2 we describe redundant binary representation of the input vector and matrix coefficients in the context of Section 2.4 to obtain additional gains in precision of MVM computation on the *Kerneltron I* architecture. In Section 6.3 we introduce another Nyquist-rate quantization scheme, algorithmic partial analog-to-digital conversion, where several algorithmic partial analog-to-digital conversion cycles are interleaved with computation on the matrix-vector multiplying array. This allows to perform only one algorithmic quantization for multiple computation cycles, making algorithmic partial ADCs more area and power efficient than flash ADCs of equivalent precision. A row-cumulative version of the algorithmic partial ADC, further reducing area and power requirements but allowing for smaller gains in resolution, is presented in Section 6.4. Section 6.5 compares row-parallel flash ADCs, and row-parallel and row-cumulative algorithmic partial ADCs, with and without radix-less-than-2 data encoding, in terms of resolution gain, and discusses implementation complexity issues.

6.2 Non-radix-2 Row-parallel Flash A/D Conversion

As shown in Section 2.4, significant improvements in precision can be obtained by exploiting the binary representation of matrix elements and vector inputs, and performing the computation (2.4) in the digital domain, from quantized estimates of the partial outputs (2.5) converted to digital domain by row-parallel flash quantizers. The effect of averaging the quantization error over a large number of quantized values of ${Y_m}^{(i,j)}$ boosts the precision of the digital estimate of Y_m, beyond the intrinsic resolution of the analog array and the A/D quantizers used.

Higher resolution, beyond the aforementioned enhancements, can be traded for a relatively modest increase in implementation complexity (in terms of circuit area and computation time), by utilizing a non-radix-2 binary representation of the input vector and matrix elements in Figures 6.1 and 6.2. Assume, in the case $L < \log_2(N+1)$, a *radix*-γ encoding

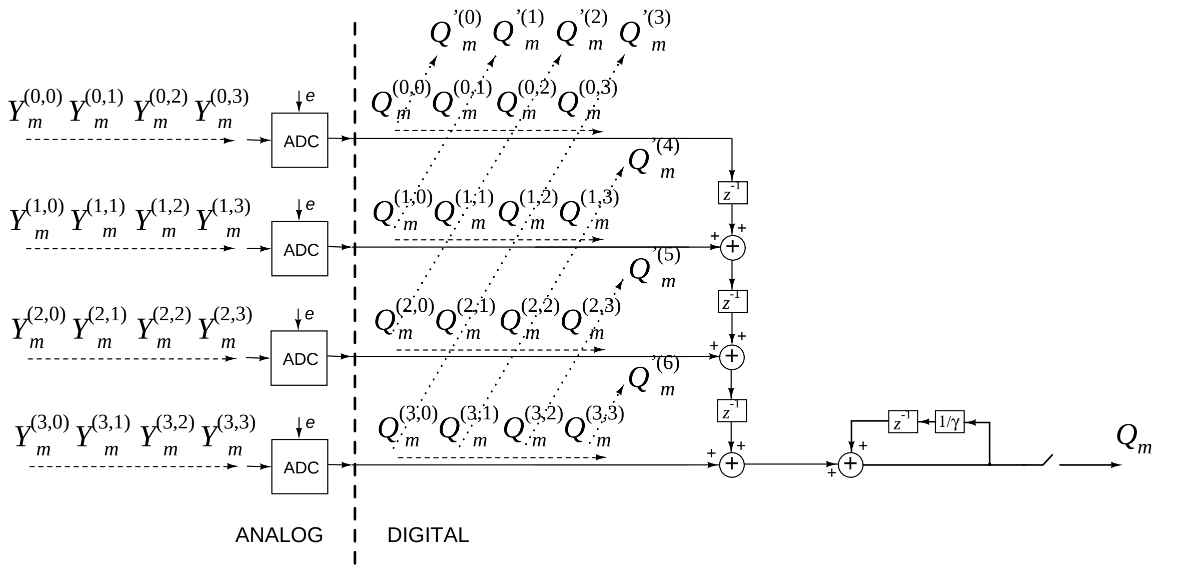

$Y_m^{(0,0)}$ $Y_m^{(0,1)}$ $Y_m^{(0,2)}$ $Y_m^{(0,3)}$
$Y_m^{(1,0)}$ $Y_m^{(1,1)}$ $Y_m^{(1,2)}$ $Y_m^{(1,3)}$
$Y_m^{(2,0)}$ $Y_m^{(2,1)}$ $Y_m^{(2,2)}$ $Y_m^{(2,3)}$
$Y_m^{(3,0)}$ $Y_m^{(3,1)}$ $Y_m^{(3,2)}$ $Y_m^{(3,3)}$
e
ADC
$Q_m^{(0,0)}$ $Q_m^{(0,1)}$ $Q_m^{(0,2)}$ $Q_m^{(0,3)}$
$Q_m^{(1,0)}$ $Q_m^{(1,1)}$ $Q_m^{(1,2)}$ $Q_m^{(1,3)}$
$Q_m^{(2,0)}$ $Q_m^{(2,1)}$ $Q_m^{(2,2)}$ $Q_m^{(2,3)}$
$Q_m^{(3,0)}$ $Q_m^{(3,1)}$ $Q_m^{(3,2)}$ $Q_m^{(3,3)}$
$Q'^{(0)}_m$ $Q'^{(1)}_m$ $Q'^{(2)}_m$ $Q'^{(3)}_m$
$Q'^{(4)}_m$
$Q'^{(5)}_m$
$Q'^{(6)}_m$
z^{-1}
$1/\gamma$
Q_m
ANALOG
DIGITAL

Figure 6.2: Diagram for the radix-γ A/D quantization and digital postprocessing block in Figure 6.1, using row-parallel flash A/D converters. The example shown is for a single m, LSB-first bit-serial inputs, and $I' = J' = 4$.

of matrix elements:

$$W_{mn} = \sum_{i=0}^{I'-1} \gamma^{-(i+1)} w_{mn}^{(i)}, \tag{6.1}$$

and input vectors:

$$X_n = \sum_{j=0}^{J'-1} \gamma^{-(j+1)} x_n^{(j)}, \tag{6.2}$$

with $1 < \gamma \leq 2$ in general, and $\gamma < 2$ in particular. This representation is *redundant* since $\gamma < 2$ requires more binary coefficients in the encoding to obtain a given resolution, than the corresponding number of bits for $\gamma = 2$. For example, consider the number of radix-2 bits I and number of radix-γ coefficients I' in the encoding of weight coefficients. It is shown that the quantization error in a radix-γ (algorithmic) representation (6.1) with $0 < \gamma < 2$ is bounded by $\gamma^{-(I+1)}$ [61]. Equating resolution in terms of (bounds on) the worst-case quantization error in both radix-2 and radix-γ cases [61],

$$\gamma^{-I'} = 2^{-I} \tag{6.3}$$

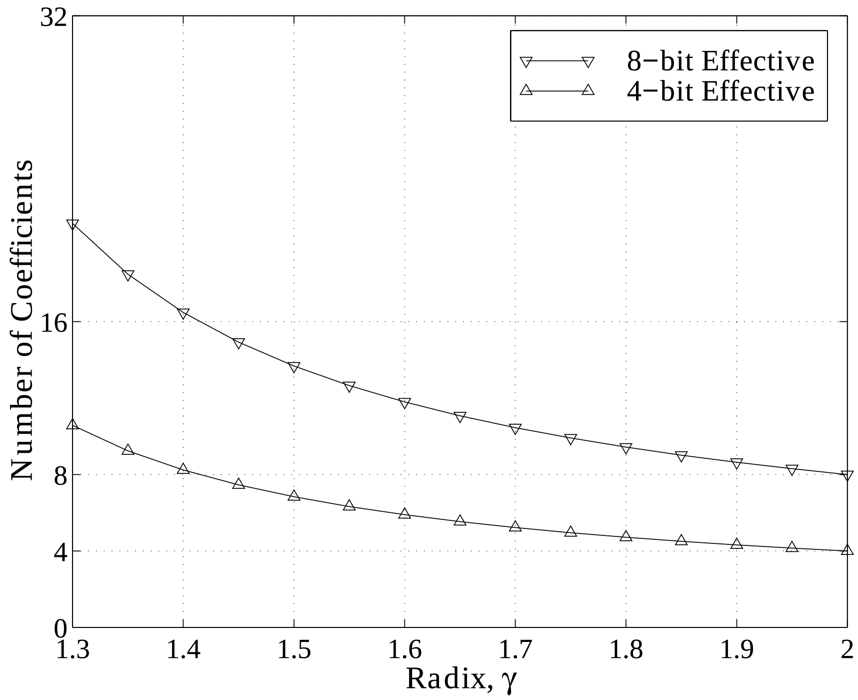


Figure 6.3: The number of binary coefficients required for input vector and matrix element radix $\gamma < 2$ encoding (for the signal range equivalent to radix-2 4-bit and 8-bit encoding).

produces an estimate for the (maximum) required number of radix-γ coefficients

$$I' = I\frac{\log 2}{\log \gamma} = \frac{I}{\log_2 \gamma}\ , \tag{6.4}$$

where thus $I' > I$ for $\gamma < 2$. This dependence is illustrated for a range of γ values in Figure 6.3. Two cases, $I = 4$ and $I = 8$, are shown. It can be observed that for $\gamma = \sqrt{2} \approx 1.41$ the number of binary coefficients doubles to maintain resolution. The same arguments hold for J and J' in the radix-γ encoding of the inputs:

$$J' = J\frac{\log 2}{\log \gamma} = \frac{J}{\log_2 \gamma}\ . \tag{6.5}$$

Following the derivations in the radix-2 case (Section 2.4), the signal range in the radix-

γ case:

$$\begin{aligned} S &= s \sum_{i=0}^{I'-1} \gamma^{-(i+1)} \sum_{j=0}^{J'-1} \gamma^{-(j+1)} \\ &= s \, \frac{1-\gamma^{-I}}{\gamma-1} \frac{1-\gamma^{-J}}{\gamma-1} \approx s \, \frac{1}{(\gamma-1)^2} \end{aligned} \tag{6.6}$$

together with the noise variance:

$$\begin{aligned} \sigma_E^2 &= \sigma_e^2 \sum_{i=0}^{I'-1} \gamma^{-2(i+1)} \sum_{j=0}^{J'-1} \gamma^{-2(j+1)} \\ &= \sigma_e^2 \, \frac{1-\gamma^{-2I}}{\gamma^2-1} \frac{1-\gamma^{-2J}}{\gamma^2-1} \approx \sigma_e^2 \, \frac{1}{(\gamma^2-1)^2} \end{aligned} \tag{6.7}$$

yields, for large I and J, an expression for signal-to-quantization-noise (SQNR) ratio in terms of the radix γ:

$$\frac{S}{\sigma_E} \approx \frac{\gamma^2-1}{(\gamma-1)^2} \frac{s}{\sigma_e} = \frac{\gamma+1}{\gamma-1} \frac{s}{\sigma_e} \tag{6.8}$$

It is clear that a redundant representation, $\gamma < 2$, leads to improved precision at the output, although at a cost. By doubling the number of binary coefficients in the representation of weights *and* inputs ($\gamma = \sqrt{2}$), the SQNR improves by a factor $3 + 2\sqrt{2} = 5.83$, almost twice the improvement (by a factor 3) in the radix-2 case. Thus, coefficient doubling yields an improvement of one bit in *median* resolution, which as in (2.19) is given by (6.8), (2.17) and (2.18):

$$\frac{S}{\mathcal{M}_E} \approx \frac{\sqrt{3}/2}{0.675} \frac{\gamma+1}{\gamma-1} \frac{s}{\mathcal{M}_e} \approx 1.28 \frac{\gamma+1}{\gamma-1} \frac{s}{\mathcal{M}_e} \tag{6.9}$$

The dependence on γ of the improvement in resolution at the output relative to that of the ADC is shown in Figure 6.4.

6.3 Row-parallel Algorithmic Partial A/D Conversion

Another choice of implementation of the A/D block is to accumulate computational array outputs in analog domain and quantize the resulting sums for every row. These quantized values can then be accumulated across multiple rows corresponding to a single matrix

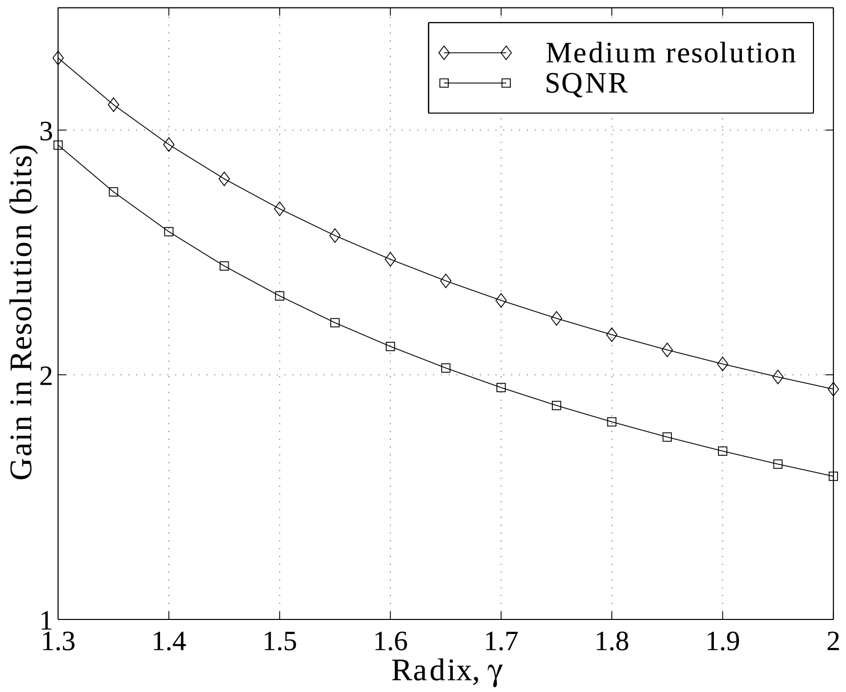


Figure 6.4: Overall resolution enhancement in a MVM architecture with row-parallel flash ADCs with non-radix-2 binary weights. Improvement in bits in median resolution and SQNR of overall system as compared to that of each of the row-parallel flash A/D converters with binary weights $\gamma < 2$.

element in digital domain. We propose a row-parallel algorithmic partial A/D converter, combining properties of both pipelined and iterative algorithmic ADCs by interleaving bit-serial input signal with the previous cycle's residue.

To design a row-parallel algorithmic partial A/D converter we use residue modulators [52] comprising a quantizer and an adder. Its diagram and transfer function are shown in Figure 6.5 *(a)*. The digital code is generated as:

$$D_{out1} := (V_{in1} > V_{ref}), \tag{6.10}$$

with the signal range decreasing by a factor of two:

$$V_{out1} = V_{in1} - V_{ref} D_{out1}, \tag{6.11}$$

as shown in the figure.

A radix-γ iterative algorithmic A/D converter consisting of a residue modulator described above, multiplier, and a delay element is shown in Figure 6.5 *(b)*. The output code is generated serially:

$$D_{out} := (\gamma V'_{in} > V_{ref}) \tag{6.12}$$

The input signal is sampled once:

$$V'^{(0)}_{in} = V_{in}, \tag{6.13}$$

producing MSB value in that clock cycle. The rest of the bits are obtained by performing the cyclic A/D conversion on respective amplified residues:

$$V'^{(i+1)}_{in} = V^{(i)}_{out} \tag{6.14}$$

The output voltage for a general radix-γ case is defined as:

$$V_{out} = \gamma V'_{in} - D_{out} V_{ref}, \tag{6.15}$$

where V_{ref} is equal to the (effective) input range, s. The range of the residue signal is the same as the input range as shown on the transfer characteristic plot in the same figure.

The inherent property of the analog array architecture is bit-serial representation of the row outputs, ${Y_m}^{(i,j)}$. If the input vectors binary coefficients, ${x_n}^{(j)}$, are presented MSB-first over time, the outputs are generated MSB-first as well. We use this property in a design of a residue-input additive algorithmic partial A/D converter presented in Figure 6.5 *(c)*.

In this scheme, the input to the rightmost quantizer is not only the residue of the previous cycle computation, but a function of its sum with the row output produced in the given cycle. This way, after a more significant bit is computed, the analog residue of this computation is combined with the next (in a bit-serial stream) incoming binary coefficient of the same weight. This addition doubles the signal range making it $2s$. This calls for an additional block which would perform extra 1-bit quantization and halve the range. For this purpose we use a residue modulator shown in Figure 6.5 *(a)* with generated digital code described in (6.10).

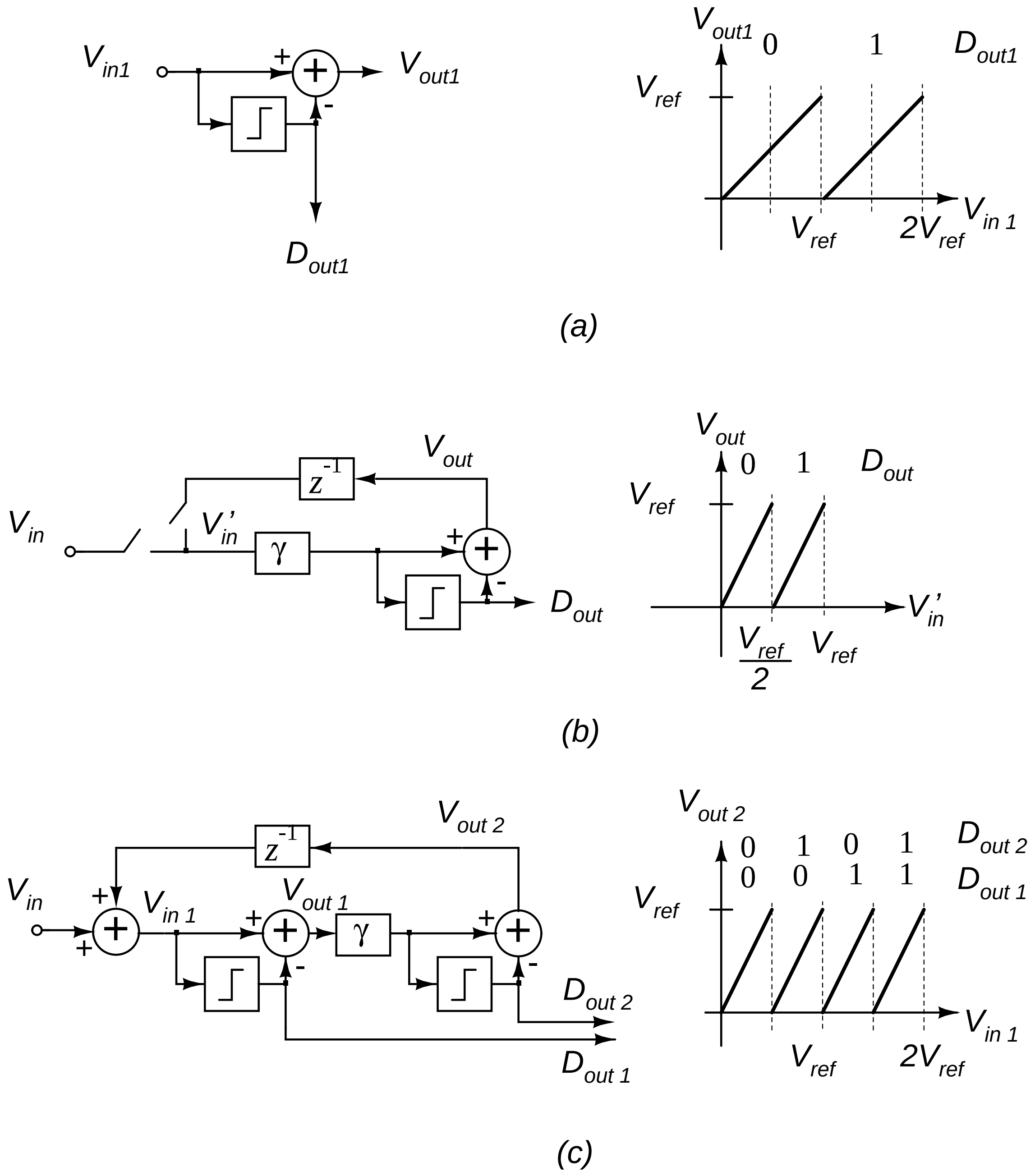


Figure 6.5: Block diagram of a residue modulator, *(a)*; a standard algorithmic A/D converter, *(b)*; a algorithmic partial A/D converter (PADC) with cumulative input, *(c)*, and their respective transfer characteristics for radix $\gamma = 2$.

The code for the second output bit of the algorithmic partial A/D converter is:

$$D_{out2} := (\gamma V_{out1} > V_{ref}) \tag{6.16}$$

Substituting equation (6.10) into (6.11) and then into (6.16), we get:

$$\begin{aligned} D_{out2} &:= [\gamma(V_{in1} - D_{out1}V_{ref}) > V_{ref}] \\ &:= [\gamma(V_{in1} - (V_{in1} > V_{ref})V_{ref}) > V_{ref}] \end{aligned} \tag{6.17}$$

The expression for the output voltage

$$V_{out2} = \gamma V_{out1} - D_{out2}V_{ref} \tag{6.18}$$

can be expanded, using equation (6.11), as

$$V_{out2} = \gamma(V_{in1} - D_{out1}V_{ref}) - D_{out2}V_{ref}, \tag{6.19}$$

describing a four-segment, double-range, gain-γ transfer characteristic shown in Figure 6.5 *(c)*.

The 2-bit-per-iteration algorithmic partial A/D converter described is used in the row-parallel architecture shown in Figure 6.6, implementing the A/D block of Figure 6.1.

After all of matrix elements are fed into the algorithmic partial A/D converter, the conversion can be continued by operating on the signal residue (as in standard iterative algorithmic ADCs). In this case the input signal is equal to zero and the residue is of the range s, so the output of the first quantizer is always zero. The resolution in this case is limited only by circuit implementation inaccuracies (*i.e.* gain error, second quantizer comparator offset). However, in order to perform precision analysis, equivalent to previously considered row-parallel conversion schemes, we limit the number of conversion cycles to L (for radix-2 case, or more for the same signal range for radix-γ case). Clearly, L should be greater or equal to J'. Otherwise, some LSB coefficients of X would be omitted from computation. We consider the non-ideal case when the overall system precision is limited by that of the ADC (as before), or $L < log_2(N+1)$.

Assuming radix-γ matrix elements and input vector encoding as in (6.1) and (6.2), in this scheme the A/D outputs $Q''^{(i)}_m$ are full digital quantized outputs of i-th row computed

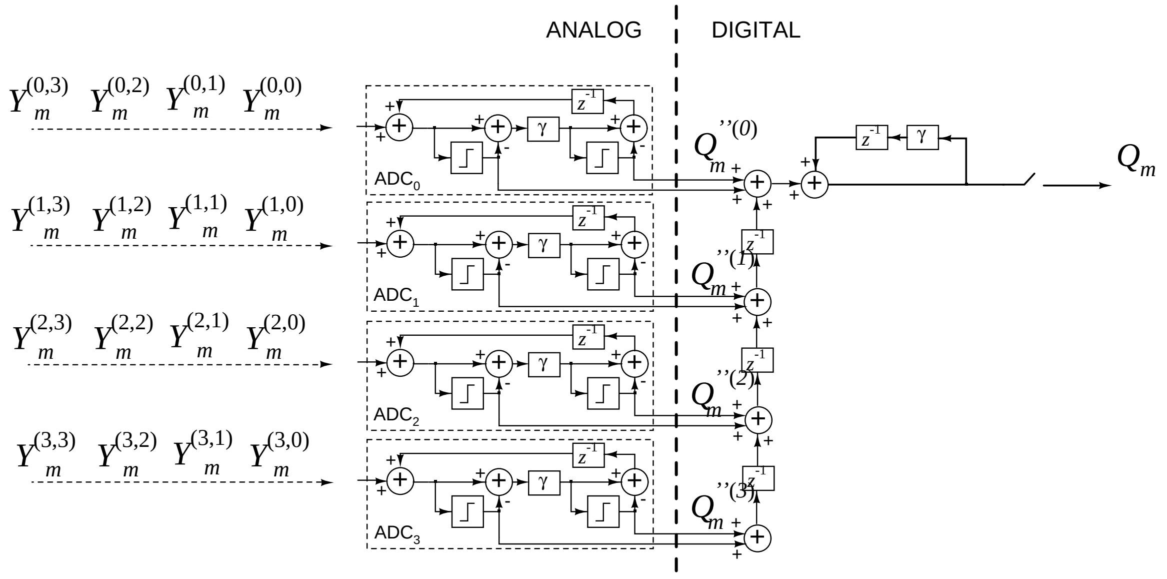

ANALOG
DIGITAL
$Y_m^{(0,3)}$ $Y_m^{(0,2)}$ $Y_m^{(0,1)}$ $Y_m^{(0,0)}$
$Y_m^{(1,3)}$ $Y_m^{(1,2)}$ $Y_m^{(1,1)}$ $Y_m^{(1,0)}$
$Y_m^{(2,3)}$ $Y_m^{(2,2)}$ $Y_m^{(2,1)}$ $Y_m^{(2,0)}$
$Y_m^{(3,3)}$ $Y_m^{(3,2)}$ $Y_m^{(3,1)}$ $Y_m^{(3,0)}$
ADC$_0$
ADC$_1$
ADC$_2$
ADC$_3$
z^{-1}
γ
$Q_m^{''(0)}$
$Q_m^{''(1)}$
$Q_m^{''(2)}$
$Q_m^{''(3)}$
Q_m

Figure 6.6: Block diagram of the ADC block implemented with row-parallel algorithmic partial A/D converters, in common for m-th output vector component with I-bit matrix elements and J-bit input vector. The case of $I' = J' = 4$ ($I' = I = J' = J$, when $\gamma = 2$) with MSB-first input vectors bit-serial representation.

over J' clock cycles, as opposed to J' binary coefficients, ${Q_m}^{(i,j)}$, for a given i, in the row-parallel flash ADC architecture described in Section 6.2. Ideally, they relate as:

$$Q_m''^{(i)} = \sum_{j=0}^{J'-1} \gamma^{-(j+1)} Q_m^{(i,j)} \tag{6.20}$$

In the row-parallel algorithmic partial A/D conversion case coefficients ${Q_m}^{(i,j)}$ are not explicitly computed. Instead analog residue from the previous cycle is added to the input of the current one. The final inner-product computation result can be constructed off chip, in digital domain as:

$$Q_m = \sum_{i=0}^{I'-1} \gamma^{-(i+1)} Q_m''^{(i)} \tag{6.21}$$

To asses the overall system resolution assume that after L conversion cycles the quantization error, e, is random and uniform. Let S be the signal range of the constructed digital output Q_m as before. According to equation (6.21), the latter can be expressed as:

$$S = s \sum_{k=0}^{I'-1} \frac{1}{\gamma^{i+1}} \approx s \frac{1/\gamma}{1 - 1/\gamma} = s \frac{1}{\gamma - 1}, \tag{6.22}$$

for large I'. Similarly to the previous sections, the noise variances relate as:

$$\sigma_E^2 = \sigma_e^2 \sum_{k=0}^{I'-1} \frac{1}{\gamma^{2(i+1)}} \approx \sigma_e^2 \frac{1}{\gamma^2 - 1} \tag{6.23}$$

In terms of signal-to-quantization-noise ratio (obtained by dividing (6.22) by the square root of (6.23)), this corresponds to:

$$\frac{S}{\sigma_E} \approx \frac{1/(\gamma-1)}{\sqrt{1/(\gamma^2-1)}} \frac{s}{\sigma_e} = \sqrt{\frac{\gamma+1}{\gamma-1}} \frac{s}{\sigma_e} \tag{6.24}$$

Thus, using row-parallel radix-γ algorithmic partial A/D conversion scheme, we achieve an improvement in signal-to-quantization-noise ratio by the factor derived above. This result is plotted in Figure 6.7.

To characterize the system performance in terms of median resolution, we substitute the values of noise variances in both cases from equations (2.17) and (2.18) into equation (6.24):

$$\frac{S}{\mathcal{M}_E} \approx \frac{\sqrt{3}/2}{0.675} \sqrt{\frac{\gamma+1}{\gamma-1}} \frac{s}{\mathcal{M}_e} \approx 1.28 \sqrt{\frac{\gamma+1}{\gamma-1}} \frac{s}{\mathcal{M}_e} \tag{6.25}$$

Figure 6.7 illustrates the resulting improvement in median resolution of the overall system as compared with that of each A/D converter, as a function of radix. In radix-2 case, the gain in median resolution is $log_2(2.22) \approx 1.1$ bits. For smaller radixes, the median resolution increases further. For instance, when the number of binary coefficients doubles ($\gamma \approx 1.4$), the median resolution improves by 1.6 bits.

6.4 Row-cumulative A/D Conversion

Analog-to-digital conversion can also be performed by a single algorithmic partial A/D converter described above for multiple analog array rows corresponding to a single matrix row. Multiple analog outputs are added together to construct the inner-product computation overall analog output value. This operation is performed in both analog and digital domains by means of the cumulative architecture shown in Figure 6.8 (for the number of rows equal to four).

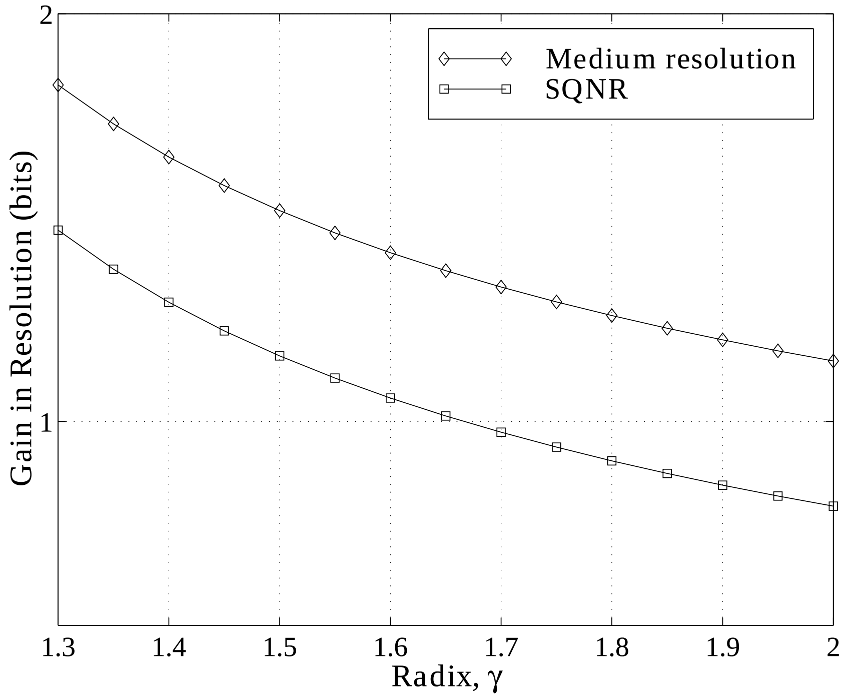


Figure 6.7: Overall resolution enhancement in a MVM architecture with row-parallel partial algorithmic ADCs for non-radix-2 binary inputs and weights. Improvement in bits in median resolution and SQNR of overall system as compared to that of each of the row-parallel A/D converters with binary weights $\gamma < 2$;

Row analog output values are first combined in analog domain. Assuming that the input vector, X, is presented MSB-first in a bit-serial fashion, elements, ${Y_m}^{(i,j)}$, of the analog array output matrix have the same weight for $i + j = const$ (this corresponds to elements on diagonals parallel to the main diagonal of the ${Y_m}^{(i,j)}$ matrix) with more significant bits available first. A delay line of sample-and-hold cells ensures that the signals added are of the same binary weight in the constructed final product. Assume again that a row output signal range is s. Every addition of two equally weighted outputs produces a signal of range $2s$. This could cause an overflow in the analog accumulative delay line. A mixed-signal accumulator cell consisting of a resetable $\Delta - \Sigma$ converter and a digital delay element are used to handle overflows for every addition in the analog delay line. Whenever the sum is out of range, a carry-bit is generated and propagated by a digital accumulator through an equivalent delay line, now in digital domain. A similar mixed-signal mash $\Delta - \Sigma$ architecture was reported in [52]. Effectively two goals are reached. The signal range is kept constant, and the analog value is partially quantized.

By the time an analog value is accumulated over I' rows, $I' - 1$ bits of its digital representation are already available. Accumulated values are further processed by an algorithmic partial A/D converter described in detail in the previous section. Here, they are combined with the residue of a previous cycle's conversion to generate the remaining bits. In digital domain all of the bits are accumulated to produce the K-bit representation of the constructed inner-product computation result, Q_m, for m-th matrix row.

Once again, after all of matrix elements are fed into the partial algorithmic A/D converter, the conversion can be continued by operating on the signal residue. The precision would only be limited by circuit implementation inaccuracies. However, in order to perform comparative precision analysis, equivalent to the previous conversion schemes, we limit the number of conversion cycles to K.

The cumulative algorithmic partial A/D conversion scheme uses one ADC per one output vector component, Y_m. This means that precision of the inner-product computation is the same as that of an A/D conversions - L bits. Addition in analog domain prior to conversion requires fewer ADCs per array. It also significantly simplifies digital postprocessing, eliminating most of the need for addition and multiplication. The system however exhibits lower precision and is not as versatile or scalable.

6.5 Comparative Study

The signal-to-quantization-noise ratio and the median resolution of kernel machines with three types of Nyquist-rate quantizers, for $\gamma = \sqrt{2}$, are plotted in Figures 6.9 and 6.10 respectively. When $L < log_2(N + 1)$ the SNR is enhanced by approximately 2.5 bits and the median resolution improved by 2.9 bits in a flash ADC architecture, and by 1.3 bits and 1.6 bits, respectively, in a row-parallel algorithmic partial ADC architecture. The gains are

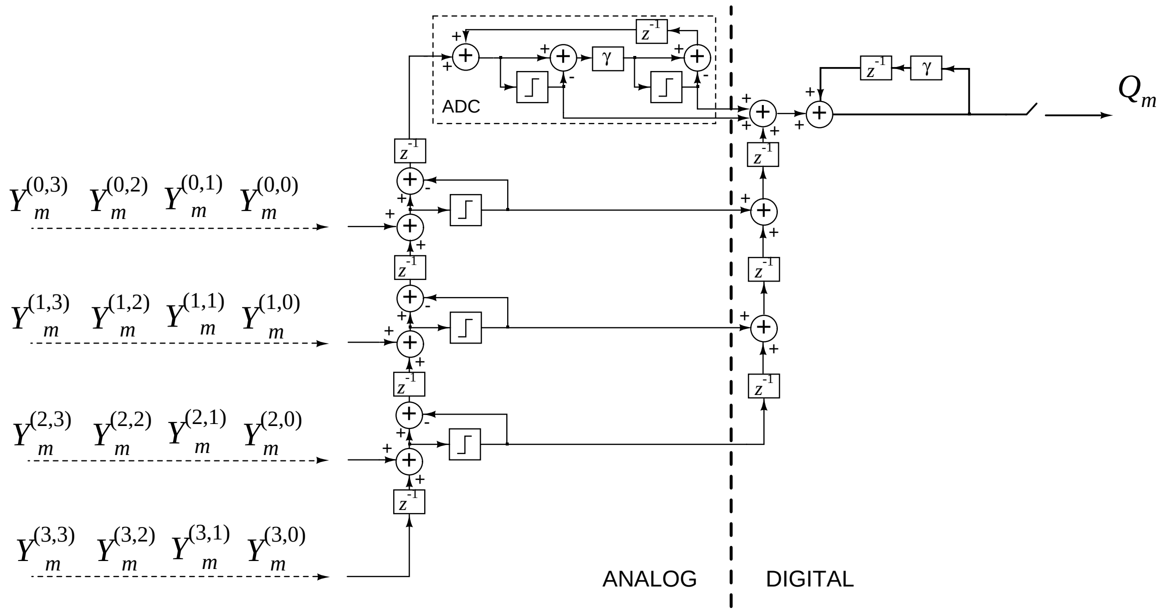

ADC
z^{-1}
γ
Q_m
$Y_m^{(0,3)}$ $Y_m^{(0,2)}$ $Y_m^{(0,1)}$ $Y_m^{(0,0)}$
$Y_m^{(1,3)}$ $Y_m^{(1,2)}$ $Y_m^{(1,1)}$ $Y_m^{(1,0)}$
$Y_m^{(2,3)}$ $Y_m^{(2,2)}$ $Y_m^{(2,1)}$ $Y_m^{(2,0)}$
$Y_m^{(3,3)}$ $Y_m^{(3,2)}$ $Y_m^{(3,1)}$ $Y_m^{(3,0)}$
ANALOG
DIGITAL

Figure 6.8: Block diagram of the ADC block implemented with a single algorithmic A/D converter in common for m-th output vector component with I-bit matrix elements and J-bit input vector. The case of $I' = J' = 4$ ($I' = I = J' = J$, when $\gamma = 2$) with MSB-first input vectors bit-serial representation.

due to averaging in digital accumulation of partial products, $Q_m^{(i,j)}$ and $Q''^{(i)}_m$, of redundantly encoded input vectors and matrix row elements. When $I' = J' = 4$, $N = 511$ and the ADC resolution L is 9 bits, the resolution of ADC exactly matches the number of columns in the array. In this case, assuming sufficient precision of analog array computation, an ideal precision of 17 bits is obtained in all cases making the system truly transparent for the outside digital world.

Algorithmic partial ADCs perform partial or full accumulation in analog domain and therefore produce smaller gains in resolution caused by averaging quantized partial products in digital domain. They however have an advantage of lower area and power requirements and simplified digital postprocessing, allowing for higher resolution implementations using equivalent amount of resources and limited only by circuit implementation inaccuracies.

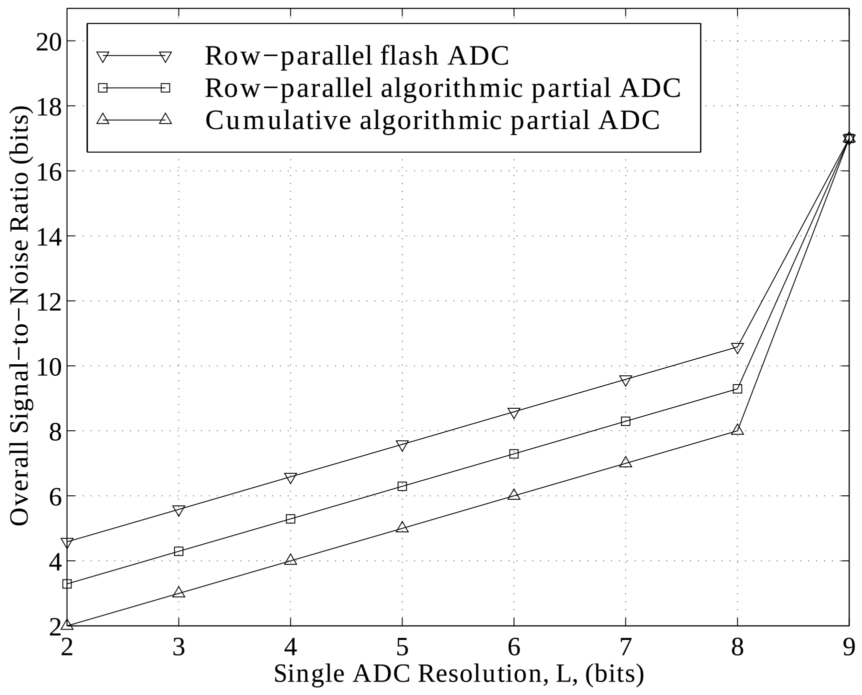


Figure 6.9: SQNR of three kernel machines with different types of Nyquist-rate quantizers versus a single ADC resolution, L. Shown is the case for $I = J = 4$, $N = 511$, $\gamma = \sqrt{2}$.

6.6 Conclusions

Parallelism in computation, bit-wise data encoding and redundancy in the data representation offer ways to enhance the precision of the MVM computation an a *Kerneltron* architecture with Nyquist-rate ADCs (*e.g.,Kerneltron I*). Flash and algorithmic quantization schemes trade off between higher resolution and lower implementation costs.

The resolution of computation on *Kerneltron I* architecture is enhanced by approximately 2 bits above each row-parallel flash quantizer resolution as was initially shown in Section 2.4. Further enhancements are achieved by utilizing non-radix-2 redundant data representation. Doubling the number of coefficients in the representation of inputs and matrix elements yields an additional improvement of 1 bit in resolution, for a total of approximately 3-bit enhancement.

In the general framework of radix-γ data representation, row-parallel and row-cumulative algorithmic partial ADCs have been introduced. Algorithmic partial ADCs produce smaller

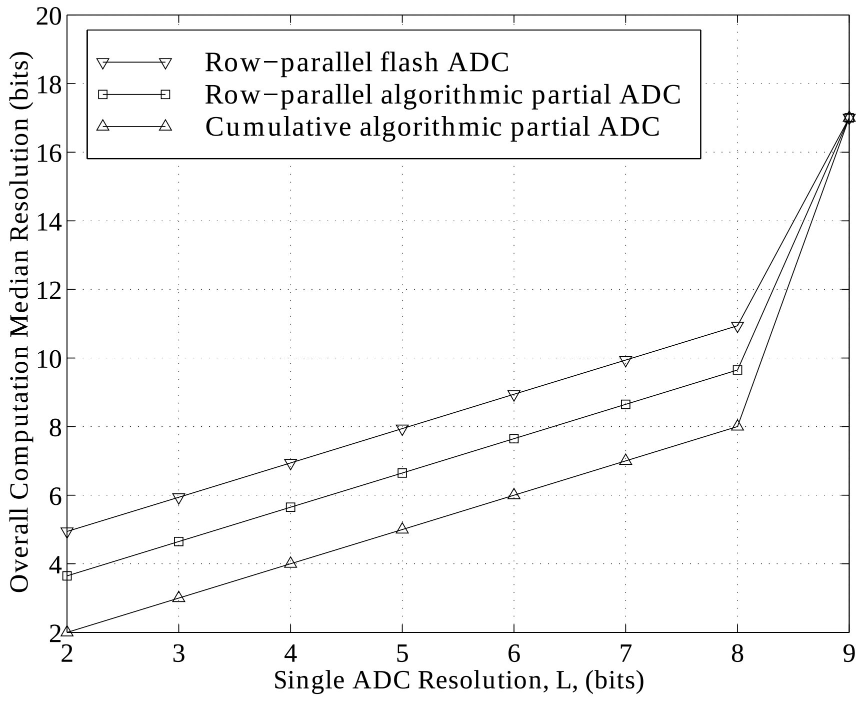


Figure 6.10: Median resolution of three kernel machines with different types of Nyquist-rate quantizers versus a single ADC resolution, L. Shown is the case for $I = J = 4$, $N = 511$, $\gamma = \sqrt{2}$.

gains in resolution, but have an advantage of lower area and power requirements and simplified digital postprocessing, allowing for higher resolution implementations with equivalent resources usage.

In principle, virtually unlimited precision can be obtained by carefully configuring the *Kerneltron* so that the resolution of the analog computation and the ADC matches the number of columns for each block. This makes the internal analog implementation of the architecture truly transparent to the user at the digital interface. For high-dimensional input spaces (*e.g.*, $N = 1024$), the nonlinearity in row-wise analog accumulation, as well as area and power requirements in the ADC implementation make this approach impractical. In Chapter 7 we present a stochastic technique allowing to significantly relax requirements on quantizers resolution and to avoid limitations of analog accumulation nonlinearity in high-dimensional kernel machines.

Chapter 7

Stochastic High-Dimensional Kernel Machines

In Chapters 2 - 5 we presented internally analog, externally digital architectures for dedicated VLSI kernel-based array processing that outperform purely digital approaches with a factor 100-10,000 in throughput, density and energy efficiency. Chapter 6 was devoted to methods of enhancing computation resolution in kernel machines with low-resolution Nyquist-rate quantizers. In this chapter we unveil a stochastic technique allowing to implement very high-dimensional kernel machines operating at full digital resolution. Requirements on the resolution of quantizers and precision of the analog implementation are significantly relaxed at the expense of a small overhead in data coding.

7.1 Introduction

As demonstrated in Section 1.4.1, the computational core of inner-product based kernel operations in image processing and pattern recognition is that of matrix-vector multiplication (MVM) in high dimensions:

$$Y_m = \sum_{n=0}^{N-1} W_{mn} X_n \tag{7.1}$$

with N-dimensional input vector X_n, M-dimensional output vector Y_m, and $M \times N$ matrix elements W_{mn}. The matrix elements W_{mn} correspond to support vectors in a support vector machine [2]. *Kerneltron* architectures are tailored to perform MVM operation with very high computational efficiency and low area utilization.

Significant digital resolution enhancement in *Kerneltron* architectures implemented with with low-resolution analog components can be obtained by exploiting Bernoulli random statistics of binary vectors. Largest gains in system precision are obtained for high input dimensions. The framework allows to operate at full digital resolution with relatively imprecise analog hardware, and with minimal cost in implementation complexity to randomize the input data.

In what follows we present a full-digital-resolution stochastic *Kerneltron* architecture. Section 7.2 identifies sources of imprecision of MVM computation on mixed-signal arrays. Section 7.3 demonstrates relaxed requirements on analog implementation due to reduction of active range of analog array outputs when inputs are Bernoulli distributed, and describes a method of pseudo-random encoding of real image data. Conclusions are given in Section 7.4.

7.2 Kerneltron System-Level Computation Resolution

The *Kerneltron* architecture combines the computational efficiency of analog array processing with the precision of digital processing and the convenience of a programmable and reconfigurable digital interface.

As shown in Chapters 2 and 4, the digital representation is embedded in the analog array architecture, with inputs presented in bit-serial fashion (2.3) or (4.3), and matrix elements stored locally in bit-parallel form (2.2) and (4.2). The key is to compute and accumulate the binary-binary partial products (2.5) and (4.6) using an analog MVM array, and to combine the quantized results in the digital domain according to (2.4) or accumulate the sum (4.5) in the analog domain using oversampling ADCs. The addends in (2.5) and (4.6) are computed using a single charge-mode AND cell in Figure 2.3, for unsigned multiply-and-accumulate operation, or a complementary exclusive-OR cell in Figure 4.1, for signed multiplication and accumulation.

The overall system resolution of *Kerneltron I* is limited by the precision in the (flash) quantization of the outputs from the array. Through digital postprocessing, two bits are gained over the resolution of row-parallel ADCs as shown in Chapter 2 and [40]. *Kerneltron II* architecture employs higher resolution oversampling quantizers. The overall system resolution in this case is limited mainly by the linearity of analog accumulation (4.6). In both of these kernel machines larger resolutions in high-dimensional implementations can be obtained by accounting for the statistics of binary terms in the addition (2.5) and (4.6). This allows to relax requirements on both quantizer resolution and analog accumulation accuracy (*e.g.,* linearity, noise) , as elaborated in the next section. While the approach is suitable for both *Kerneltron I* and *Kerneltron II* processors, in the rest of this chapter we concentrate on the former.

7.3 Resolution Enhancement Through Stochastic Encoding

Since the analog inner-product (2.5) is discrete, zero error can be achieved (as if computed digitally) by matching the quantization levels of the ADC with each of the $N + 1$ discrete levels in the inner-product. Perfect reconstruction of ${Y_m}^{(i,j)}$ from the quantized output, for an overall resolution of $I + J + \log_2(N + 1)$ bits, assumes the combined effect of noise and nonlinearity in the analog array and the ADC is within one LSB (least significant bit). For large arrays, this places stringent requirements on analog precision and ADC resolution, $L \geq \log_2(N + 1)$.

The implicit assumption is that all quantization levels are (equally) needed. A straightforward study of the statistics of the inner-product, below, reveals that this is poor use of

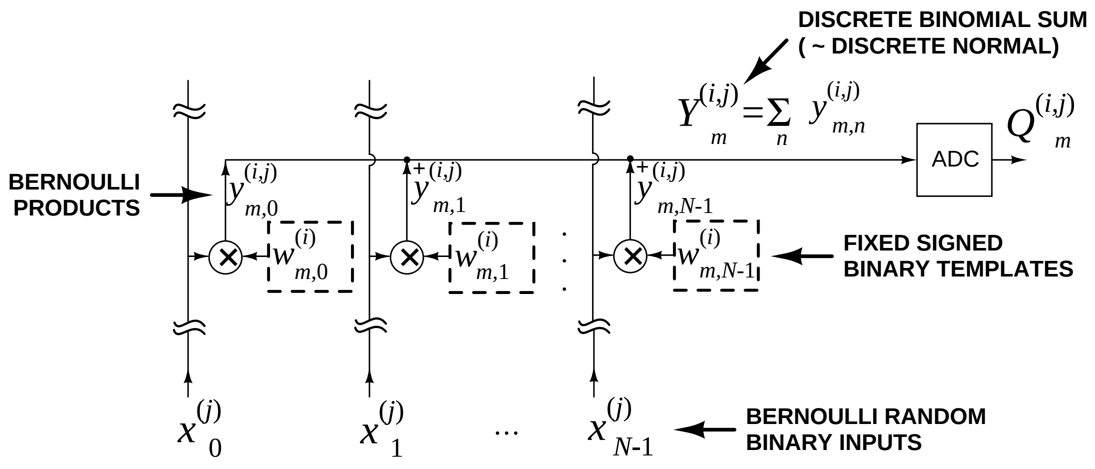


Figure 7.1: A single row of the analog array in the stochastic architecture with Bernoulli modulated signed binary inputs and fixed signed weights. The multiply-and-accumulate cells compute one-bit signed binary-binary products $y_{mn}^{(i,j)}$ which are also Bernoulli distributed. The random variables are accumulated into a binomially distributed sum $Y_m^{(i,j)}$ which approximates a normal distribution for large N.

available resources.

7.3.1 Bernoulli Statistics

For the purposes of this chapter we assume signed, rather than unsigned, binary values for inputs and weights, ${x_n}^{(j)} = \pm 1$ and ${w_{mn}}^{(i)} = \pm 1$. This translates to exclusive-OR (XOR), rather than AND, multiplication on the analog array, an operation that can be easily accomplished with the CID/DRAM architecture by differentially coding input and stored bits using twice the number of columns and unit cells, as shown in Section 4.2.2. A single row of such a differential architecture is depicted in Figure 7.1.

To show that equal spacing of quantization levels over the full range of the array output ${Y_m}^{(i,j)}$ leads to poor use of resources, let us investigate statistics of the output when input bits ${x_n}^{(j)}$ are Bernoulli distributed (*i.e.*, fair coin flips). For input bits ${x_n}^{(j)}$ that are Bernoulli distributed, the (XOR) product terms ${w_{mn}}^{(i)}{x_n}^{(j)}$ in (2.5) are Bernoulli distributed, regardless of ${w_{mn}}^{(i)}$. Their sum ${Y_m}^{(i,j)}$ thus follows a binomial distribution

$$\Pr(Y_m^{(i,j)} = 2k - N) = \binom{N}{k} p^k (1-p)^{N-k} \tag{7.2}$$

with $p = 0.5$, $k = 0, ..., N$, which in the Central Limit $N \to \infty$ approaches a normal distribution with zero mean and variance N. In other words, for random inputs in high dimensions N the active range (or standard deviation) of the inner-product is $N^{1/2}$, a factor $N^{1/2}$ smaller than the full range N.

Figure 7.2 illustrates the effect of Bernoulli distribution of the inputs on the statistics of an array row output. Reduction of the active range of the inner-product to $N^{1/2}$ allows to relax the effective resolution of the ADC by a factor proportional to $N^{1/2}$, as the number of quantization levels is proportional to $N^{1/2}$, not N. This gain is especially beneficial for parallel (flash) quantizers, as their area requirements grow exponentially with the number of bits. Additionally, Bernoulli modulation of inputs allows to significantly relax requirements on the linearity of the analog addition (2.5) by making non-linearity outside the reduced active range irrelevant.

In principle, any reduction in conversion range will result in a small but non-zero probability of overflow. In practice, the risk of overflow can be reduced to negligible levels with a few additional bits in the ADC conversion range. An alternative strategy is to use a variable resolution ADC which expands the conversion range on rare occurrences of overflow.[1]

7.3.2 Experimental Results

While the reduced range of the analog inner-product supports lower ADC resolution in terms of number of quantization levels, it requires low levels of mismatch and noise so that the discrete levels can be individually resolved, near the center of the distribution. To verify this, we conducted the following experiment.

Figure 7.3 shows the measured outputs on one row of 128 CID/DRAM cells, configured differentially to compute signed binary (exclusive-OR) inner-products of stored and

[1] Or, with stochastic input encoding, overflow detection could initiate a different random draw.

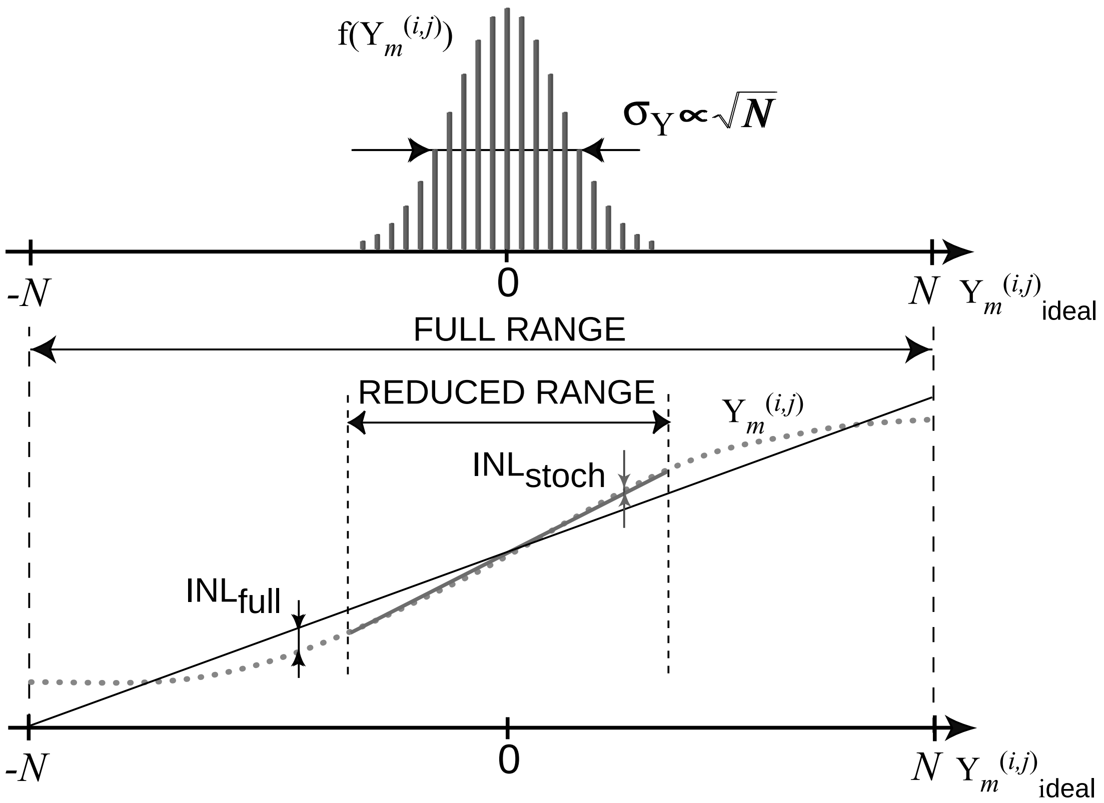
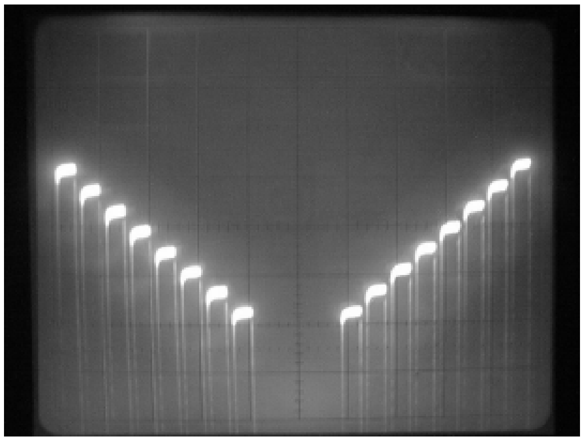
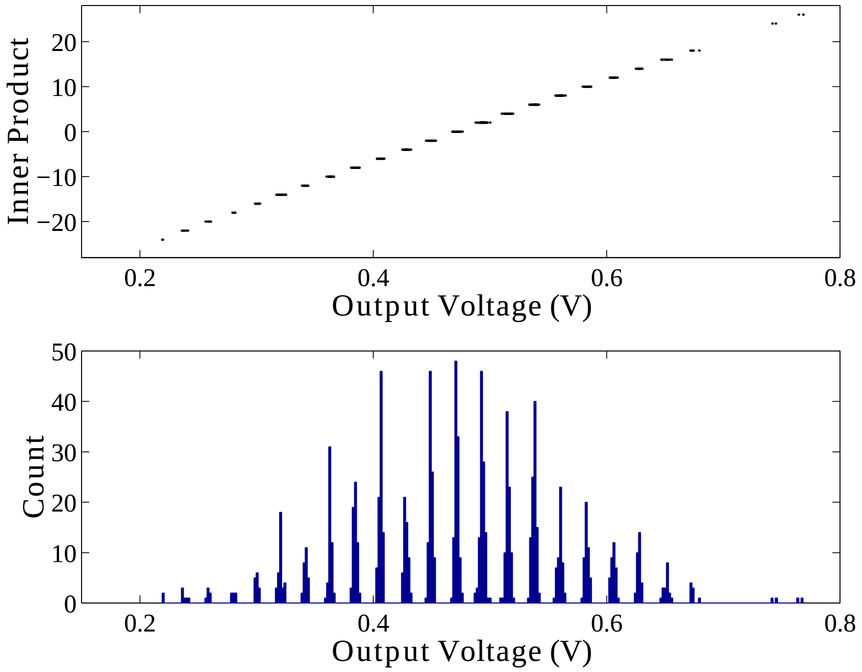


Figure 7.2: Output of a single row of the analog array, $Y_m^{(i,j)}$, in the stochastic architecture with Bernoulli encoded inputs. *Top:* $Y_m^{(i,j)}$ is a discrete random variable with probability density approaching normal distribution for large N. In Central limit the standard deviation is proportional to the square root of the full range, $\sqrt{N}$. The requirement on ADC resolution is relaxed by a factor $\sqrt{N}$, as the number of quantization levels is proportional to $\sqrt{N}$, not N. *Bottom:* The requirement on linearity of analog accumulation (2.5) is relaxed as the range of $Y_m^{(i,j)}$ is reduced by a factor $\sqrt{N}$.



(a)



(b)

Figure 7.3: Experimental results from CID/DRAM analog array. *(a)* Output voltage on the sense line computing exclusive-or inner-product of 64-dimensional stored and presented binary vectors. A variable number of active bits is summed at different locations in the array by shifting the presented bits. *(b) Top:* Measured output and actual inner-product for 1,024 samples of Bernoulli distributed pairs of stored and presented vectors. *Bottom:* Histogram of measured array outputs.

presented binary vectors in 64 dimensions. The scope trace in Figure 7.3 *(a)* is obtained by storing all $+1$ bits, and shifting a sequence of input bits that differ with the stored bits by 32 ± 4 bits. The left and right segment of the scope trace correspond to different selections of active bit locations along the array that are maximally disjoint, to indicate a worst-case mismatch scenario. The measured and actual inner-products in Figure 7.3 *(b)* are obtained by storing and presenting 1,024 pairs of random binary vectors. The histogram shows a clearly resolved, discrete binomial distribution for the observed analog voltage.

For very large arrays, mismatch and noise may pose a problem in the present implementation with floating sense line. A sense amplifier with virtual ground on the sense line and feedback capacitor optimized to the $N^{1/2}$ range would provide a simple solution.

7.3.3 Real Image Data

Although most randomly selected patterns do not correlate with any chosen template, patterns from the real world tend to correlate, and certainly those that are of interest to kernel computation [2]. The key is stochastic encoding of the inputs, as to randomize the bits presented to the analog array.

Randomizing an informative input while retaining the information is a futile goal, and we are content with a solution that approaches the ideal performance within observable bounds, and with reasonable cost in implementation. Given that "ideal" randomized inputs relax the ADC resolution by $log_2 N/2$ bits, they necessarily reduce the word-length of the output by the same. To account for the lost bits in the range of the output, it is necessary to increase the range of the "ideal" randomized input by the same number of bits.

One possible stochastic encoding scheme that restores the range is $N^{1/2}$-fold oversampling of the input through (digital) delta-sigma modulation. This is a workable solution; however we propose one that is simpler and less costly to implement. For each I-bit input component X_n, pick a random integer $U_n = \sum_{j=0}^{J-1} 2^{-j-1} u_n^{(j)}$ in the range $\pm(N^{1/2} - 1)$, and

[2]This observation, and the binomial distribution for sums of random bits (7.2), forms the basis for the associative recall in a *Kanerva* memory.

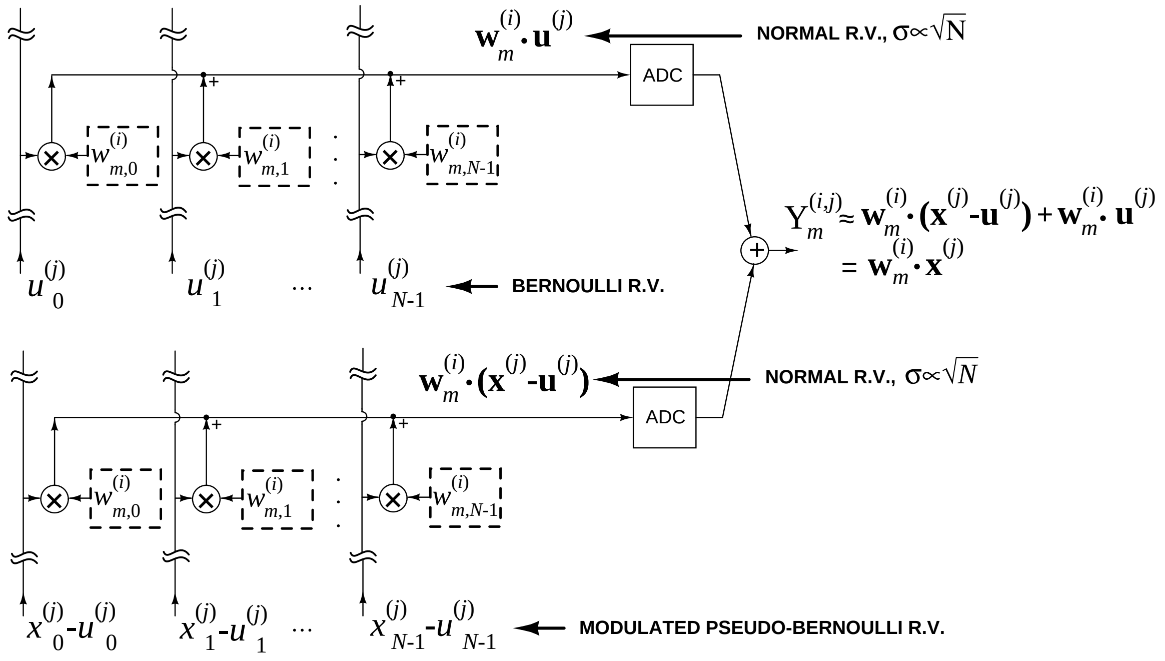

$\mathbf{w}_m^{(i)} \cdot \mathbf{u}^{(j)}$
NORMAL R.V., $\sigma \propto \sqrt{N}$
ADC
$w_{m,0}^{(i)}$
$w_{m,1}^{(i)}$
$w_{m,N-1}^{(i)}$
$Y_m^{(i,j)} \approx \mathbf{w}_m^{(i)} \cdot (\mathbf{x}^{(j)} - \mathbf{u}^{(j)}) + \mathbf{w}_m^{(i)} \cdot \mathbf{u}^{(j)} = \mathbf{w}_m^{(i)} \cdot \mathbf{x}^{(j)}$
$u_0^{(j)}$ $u_1^{(j)}$ ... $u_{N-1}^{(j)}$
BERNOULLI R.V.
$\mathbf{w}_m^{(i)} \cdot (\mathbf{x}^{(j)} - \mathbf{u}^{(j)})$
NORMAL R.V., $\sigma \propto \sqrt{N}$
ADC
$x_0^{(j)} - u_0^{(j)}$ $x_1^{(j)} - u_1^{(j)}$ ... $x_{N-1}^{(j)} - u_{N-1}^{(j)}$
MODULATED PSEUDO-BERNOULLI R.V.

Figure 7.4: Input modulation and output reconstruction scheme in the stochastic MVM architecture. The input vector binary coefficients are modulated by subtracting from them Bernoulli coefficients of a larger uniformly distributed random variable U_n to produce pseudo-Bernoulli coefficients $(x_n^{(j)} - u_n^{(j)})$. The corresponding analog inner-product is quantized by a ADC with resolution requirements relaxed by a factor $\sqrt{N}$. The analog inner-product $\mathbf{w}_m^{(i)} \cdot \mathbf{u}^{(j)}$ is computed once, quantized, and stored in a digital memory. It is subsequently added to the inner-product of each modulated input vector with stored weights to reconstruct the true partial inner-product $Y_m^{(i,j)}$.

subtract it to produce a modulated input $\tilde{X}_n = X_n - U_n$ with $log_2 N/2$ additional bits. It can be shown that for worst-case deterministic inputs X_n the mean of the inner-product for $\tilde{X}_n$ is off at most by $\pm N^{1/2}$ from the origin.

Note that U_n is uniformly distributed across its range, and therefore its binary coefficients $u_n^{(j)}$ are Bernoulli random variables. Figure 7.4 illustrates this encoding method for particular i and j. Two rows of the array are shown. Truly Bernoulli inputs $u_n^{(j)}$ are fed into one row. The inputs of the other row are stochastically modulated binary coefficients of the informative input $\tilde{x}_n = x_n - u_n$. Inner-products of approximately normal distribution are computed on both rows. Their smaller active range allows to relax the requirements

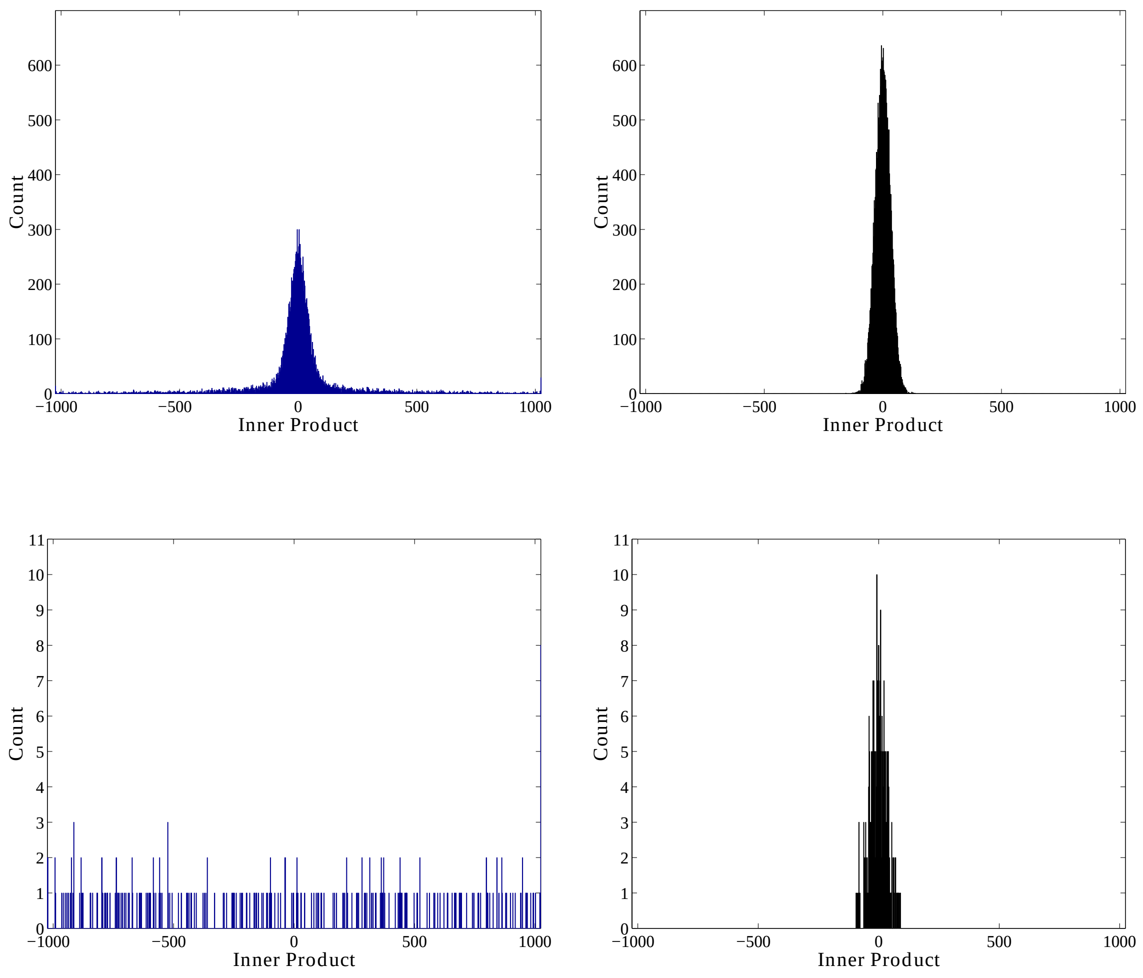


Figure 7.5: Histograms of partial binary inner-products ${Y^{(i,j)}}_m$ for 256 pairs of randomly selected 32 $\times$ 32 pixel segments of *Lena*. *Left*: with unmodulated 8-bit image data for both vectors. *Right:* with 12-bit modulated stochastic encoding of one of the two vectors. *Top*: all bit planes i and j. *Bottom*: most significant bit (MSB) plane, $i = j = 0$.

on the resolution of the quantizer by a factor $N^{1/2}$. The desired inner-products for X_n are retrieved by digitally adding the inner-products obtained for $\tilde{X}_n$ and U_n. The random offset U_n can be chosen once, so its inner-product with the templates can be pre-computed upon initializing or programming the array (in other words, the computation performed by the top row in Figure 7.4 is only performed once). The implementation cost is thus limited to component-wise subtraction of X_n and U_n, achieved using one full adder cell, one bit register, and ROM storage of the ${u_n}^{(i)}$ bits for every column of the array.

Figure 7.5 provides a proof of principle, using image data selected at random from *Lena*. 12-bit stochastic encoding of the 8-bit image, by subtracting a random variable in a range 15 times larger than the image, produces the desired binomial distribution for the partial bit inner-products, even for the most significant bit (MSB) which is most highly correlated.

7.4 Conclusions

We presented a mixed-signal paradigm for high-resolution parallel inner-product computation in very high dimensions, suitable for efficient implementation of kernels in image processing. At the core of the externally digital architecture is a high-density, low-power analog array performing binary-binary partial matrix-vector multiplication described in Chapters 2 - 5. Full digital resolution is maintained even with low-resolution analog-to-digital conversion, owing to random statistics in the analog summation of binary products. A random modulation scheme produces near-Bernoulli statistics even for highly correlated inputs, relaxing precision requirements in the analog implementation by one bit for each four-fold increase in vector dimension. The approach is validated with real image data, and with experimental results from a CID/DRAM analog array prototype in 0.5 μm CMOS.

Part V

Conclusions

Chapter 8

Contributions and Impact

8.1 Summary of Contributions

Detection of complex objects in streaming video poses two fundamental challenges: training from sparse data with proper generalization across variations in the object class and the environment; and the computational power required of the trained classifier running real-time. The *Kerneltron* supports the generalization performance of a Support Vector Machine (SVM) and offers the bandwidth and efficiency of massively parallel architecture. This mixed-signal VLSI processor is dedicated to the most intensive of SVM operations: evaluating a kernel over large numbers of vectors in high dimensions. An internally analog, externally digital architecture offers the best of both worlds: the density and energetic efficiency of an analog VLSI array, and the convenience and versatility of a digital interface.

8.1.1 Kerneltron I

At the core of *Kerneltron I* is an internally analog, fine-grain computational array performing externally digital matrix-vector multiplication of an incoming vector and a set of support vectors. The three-transistor unit cell in the array combines single-bit dynamic storage, binary multiplication, and zero-latency analog accumulation. Analog partial products are quantized using low-resolution row-parallel flash analog-to-digital converters utilizing a compact charge-based comparator and a folding circuit. Digital multiplication of variable resolution is obtained with bit-serial inputs and bit-parallel storage of matrix elements, by combining quantized outputs from multiple rows of cells over time. The combination of analog array processing and digital postprocessing enhances the precision of the digital MVM output, exceeding the resolution of the quantized analog array outputs by 2 bits. A 512 $\times$ 128 *Kerneltron I* prototype fabricated in 0.5 μm CMOS offers 2×10^{12} binary MACS (multiply-and-accumulates per second) per Watt of power.

8.1.2 Kerneltron II

In the *Kerneltron II* architecture an oversampling analog-to-digital converter accumulates partial inner-products in the analog domain, with inputs encoded in unary format. This avoids the need for high-resolution flash ADCs, which are replaced with single-bit quantizers in the delta-sigma loop. High throughput is maintained by delta-sigma algorithmic analog-to-digital converters averaging across all input vector unary bit planes in a single quantization cycle. Iteratively resampling of the modulation residue, by feeding the integrator output back to the input, yields a bit resolution linear in the number of cycles, similar to an algorithmic analog-to-digital converter. A 256 $\times$ 128 cell prototype was fabricated in 0.5 μm CMOS. The combination of analog array processing, oversampled input encoding, and delta-sigma algorithmic analog-to-digital conversion delivers a computational throughput of over 1 GMACS per mW of power, while maintaining 8-bit effective digital resolution.

8.1.3 Algorithmic Enhancements

Redundant data coding allows to further increase computational precision at the expense of relatively modest increase in implementation complexity of Nyquist-rate kernel machines. This enhancement is especially important for high-dimensional kernel machines

Table 8.1: Comparative analysis of *Kerneltron II* and other processors ([13], [14], [15], [18], [19], [17], [62]).

Processor	Year	Technology, μm	Silicon Area, mm^2	Data Encoding, bits	Throughput		Degree of Parallelism	Clock Frequency, MHz	Power, mW	Power Efficiency, mW/MMACS		Supply Voltage, Volts
					GMACS	GOPS				4-bit	1-bit	
Ultra Sparc II	1998	0.25	-	128	-	1(0.003[1])	4	400	57,500	-	-	2.6/3.3
TI DSP TMS320C6701	2000	0.18	169	32	0.3	1(0.02[1])	8	167	1,800	**600**	600	1.9/3.3
Motorola DSP StarCore	2000	0.13	225	16	0.3	-	4	300	15	**0.13**	0.13	0.9
Sony Linear Array	1996	0.4	226	1	-	5.4	4320	50	-	**0.29**	0.05	2/3.3
NEC IMAP	1997	0.55	219	8	1.28	-	256	40	1,600	**0.78**	0.78	3.3
MIT (Sodini)	1999	0.6	79	8	0.28	-	64x64	17	300	**1.07**	0.02	2.5/3.3
ACE16K	2001	0.35	143.96	8 (A)	-	330	128x128	-	4,000	**0.012**	0.012	3.3
Kerneltron II	2001	0.5	9	1	6.6	-	128x256	0.1	5.9	**0.014**	0.0009	5
Kerneltron-III (prelim.)	-	*0.35*	*48*	*1*	*105*	-	*512x1024*	*0.1*	*34*	*0.005*	*0.0003*	*3.3*

(*e.g.,* $N = 1024$). In addition, an alternative Nyquist-rate quantization scheme, algorithmic partial A/D conversion, allows to achieve precision higher than that of a flash ADC using equivalent amount of area and power resources by performing several partial quantizations in a single computation cycle.

Ultimately, high-resolution matrix-vector multiplication in very high dimensions is performed using stochastic data encoding. Full digital resolution is maintained even with low-resolution analog-to-digital conversion, owing to random statistics in the analog summation of binary products. A random modulation scheme produces near-Bernoulli statistics even for highly correlated inputs, relaxing precision requirements in the analog implementation by one bit for each four-fold increase in vector dimension. The approach is validated with real image data, and with experimental results from a *Kerneltron* prototype in 0.5 μm CMOS.

8.2 Comparative Study

A comparative analysis of *Kerneltron II* and other modern processors ([13], [14], [15], [18], [19], [17], [62]) is presented in Table 8.1. The third last column represents power efficiency (power in mW over throughput in millions of multiply-and-accumulates) for 4-bit data encoding [2]. Power efficiency accounts for both throughput and power making it a good characteristic of performance of processors used in portable devices. *Kerneltron II* outperforms all listed digital processors in terms of power efficiency by several orders of magnitude [3]. The power efficiency is comparable to ACE16K CNN-inspired mixed-mode architecture which is implemented in a more advanced technology. In addition, *Kerneltron*

[1] *From the previous page:* Throughput is limited by memory access time, memory-processor communication bandwidth, and code and compiler efficiency.

[2] For object detection tasks 4-bit data encoding is deemed sufficient both for live and artificial classification systems.

[3] Theoretically, Motorola StarCore DSP core power efficiency is a factor of 10 worse than that of *Kerneltron II*. However, the data for StarCore external memory access time and compiler overhead are not available and are estimated to introduce significant additional degrade in attained throughput when performing matrix-vector multiplication.

II uses only a small fraction of other processors silicon area to deliver a significant throughput. It is implemented in a conservative technology and operates at conservative frequency and supply voltage.

Kerneltron processors achieve computational efficiency above 1GOPS/mWatt at full digital resolution, a factor of 10,000 better than a modern multiscalar CPU. The rationale behind such an advantage is based on the following set of principles: *large-scale fine-grain array processing* – for high-dimensional matrix operations; *massive-parallelism and embedded memories* – for real-time performance in computationally-intensive tasks; *compact mixed-signal implementation with parallel data converters* – for autonomous low-power operation with convenient digital interface; *redundant and stochastic data coding schemes* – to achieve fully-digital computation precision with both analog and digital components.

Kerneltron massively parallel mixed-signal VLSI architecture is easily scalable and capable of delivering 105 GMACS at 34 mW of power in a 0.35 μm technology (*Kerneltron III* in Table 8.1) - a level of throughput and power efficiency by far sufficient for real-time support vector detection of complex objects on a portable platform, with applications ranging from artificial vision to automated surveillance to human-computer interfaces.

Bibliography

[1] Boser, B., Guyon, I. and Vapnik, V., “A training algorithm for optimal margin classifier,” in *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pp 144-52, 1992.

[2] Vapnik, V. *The Nature of Statistical Learning Theory*, Springer Verlag, 1995.

[3] Osuna, E., Freund, R., and Girosi, F., “Training support vector machines: An application to face detection,” in *Computer Vision and Pattern Recognition*, pp 130-136, 1997.

[4] Oren, M., Papageorgiou, C., Sinha, P., Osuna, E. and Poggio, T. “Pedestrian detection using wavelet templates,” in *Computer Vision and Pattern Recognition*, pp 193-199, 1997.

[5] Papageorgiou, C.P, Oren, M. and Poggio, T., “A General Framework for Object Detection,” in *Proceedings of International Conference on Computer Vision*, 1998.

[6] H. Sahbi, D. Geman and N. Boujemaa, “Face Detection Using Coarse-to-Fine Support Vector Classifiers,” *IEEE Int. Conf. Image Processing (ICIP’2002),* Rochester NY, 2002.

[7] S. Kang, and S.-W. Lee,“Handheld Computer Vision System for the Visually Impaired,” Proc. of 3rd International Workshop on Human-Friendly Welfare Robotic Systems, Daejeon, Korea, pp. 43-48, 2002.

[8] Girosi, F., Jones, M. and Poggio, T. “Regularization Theory and Neural Networks Architectures,” *Neural Computation*, vol. **7**, pp 219-269, 1995.

[9] Pontil, M. and Verri, A., “Properties of Support Vector Machines,” *Neural Computation*, vol. **10**, pp 977-996, 1998.

[10] Burges, C., “A Tutorial on Support Vector Machines for Pattern Recognition,” in U. Fayyad, editor, *Proceedings of Data Mining and Knowledge Discovery*, pp 1-43, 1998.

[11] Cauwenberghs, G. and Poggio, T., “Incremental and Decremental Support Vector Machine Learning,” in Adv. Neural Information Processing Systems, Proc. of 2000 IEEE NIPS Conf., Cambridge MA: MIT Press, 2001.

[12] Schölkopf, B., Sung, K., Burges, C., Girosi, F., Niyogi, P., Poggio, T. and Vapnik, V. "Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Functions Classifiers," *IEEE Transactions on Signal Processing*, 1997.

[13] "UltraSPARCtm-II CPU Module," *Sun Microsystems Datasheet*, July, 1999.

[14] K. Castille, "TMS320C6000 Power Consumption Summary," *Texas Instruments Application Report*, 1999.

[15] "StarCore. Brighter DSP Technology," *Motorola Literature # FLDR36/D-2*, 2000.

[16] J. Wawrzynek, et al., "SPERT-II: A Vector Microprocessor System and its Application to Large Problems in Backpropagation Training," in *Advances in Neural Information Processing Systems*, Cambridge, MA: MIT Press, vol. 8, pp 619-625, 1996.

[17] J.C. Gealow and C.G. Sodini, "A Pixel-Parallel Image Processor Using Logic Pitch-Matched to Dynamic Memory," *IEEE J. Solid-State Circuits*, vol. **34**, pp 831-839, 1999.

[18] M. Kurokawa, A. Hashiguchi, et al., "5.4GOPS Linear Array Architecture DSP for Video-Format Conversion," *Proc. of IEEE Int. Solid-State Circuits Conference (ISSCC'96)*, pp 254-255, 1996.

[19] N. Yamashita, Y. Fujita, et al., "An Integrated Memory ArrayProcessor with a Synchronous-DRAM Interface for Real-Time Vision Applications," *Proc. of IEEE Int. Conference on Pattern Recognition (ICPR'96)*, pp 575-580, 1996.

[20] Cauwenberghs, G. and Bayoumi, M., *Learning on Silicon, Analog VLSI Adaptive Systems*, Norwell MA: Kluwer Academic, 1999.

[21] A. Kramer, "Array-based analog computation," *IEEE Micro*, vol. **16** (5), pp. 40-49, 1996.

[22] A. Chiang, "A programmable CCD signal processor," *IEEE Journal of Solid-State Circuits*, vol. **25** (6), pp. 1510-1517, 1990.

[23] C. Neugebauer and A. Yariv, "A Parallel Analog CCD/CMOS Neural Network IC," Proc. IEEE Int. Joint Conference on Neural Networks (IJCNN'91), Seattle, WA, vol. **1**, pp 447-451, 1991.

[24] V. Pedroni, A. Agranat, C. Neugebauer, A. Yariv, "Pattern matching and parallel processing with CCD technology," Proc. IEEE Int. Joint Conference on Neural Networks (IJCNN'92), vol. **3**, pp 620-623, 1992.

[25] G. Han, E. Sanchez-Sinencio, "A general purpose neuro-image processor architecture," Proc. of IEEE Int. Symp. on Circuits and Systems (ISCAS'96), vol. **3**, pp 495-498, 1996.

[26] M. Holler, S. Tam, H. Castro and R. Benson, ”An Electrically Trainable Artificial Neural Network (ETANN) with 10,240 Floating Gate Synapses,” in *Proc. Int. Joint Conf. Neural Networks*, Washington DC, pp 191-196, 1989.

[27] F. Kub, K. Moon, I. Mack, F. Long, “ Programmable analog vector-matrix multipliers,” *IEEE Journal of Solid-State Circuits,* vol. **25** (1), pp. 207-214, 1990.

[28] G. Cauwenberghs, C.F. Neugebauer and A. Yariv, “Analysis and Verification of an Analog VLSI Incremental Outer-Product Learning System,” *IEEE Trans. Neural Networks,* vol. **3** (3), pp. 488-497, May 1992.

[29] A.G. Andreou, K.A. Boahen, P.O. Pouliquen, A. Pavasovic, R.E. Jenkins, and K. Strohbehn, “Current-Mode Subthreshold MOS Circuits for Analog VLSI Neural Systems,” *IEEE Transactions on Neural Networks,* vol. **2** (2), pp 205-213, 1991.

[30] G. Cauwenberghs and V. Pedroni, “A Charge-Based CMOS Parallel Analog Vector Quantizer,” *Adv. Neural Information Processing Systems (NIPS*94), Cambridge,* MA: MIT Press, vol. 7, pp. 779-786, 1995.

[31] A. Murray and P.J. Edwards, “Synaptic Noise During MLP Training Enhances Fault-Tolerance, Generalization and Learning Trajectory,” in *Advances in Neural Information Processing Systems,* San Mateo, CA: Morgan Kaufman, vol. **5**, pp 491-498, 1993.

[32] B. Prince, *Semiconductor Memories. A Handbook of Design, Manufacture, and Applications,* 2nd ed., John Wiley & Sons, 1991.

[33] M. Howes, D. Morgan, Eds., *Charge-Coupled Devices and Systems*, John Wiley & Sons, 1979.

[34] R. Melen, D. Buss, Eds., *Charge-Coupled Devices: Technology and Applications*, IEEE Press, 1976.

[35] D.F. Barbe, Ed., *Charge-Coupled Devices,* Springer-Verlag, 1980.

[36] C. Sequin, M Tompsett *Charge-Transfer Devices,* Academic Press, 1975.

[37] Poggio, T. and Beymer, D., “Learning to See,” *IEEE Spectrum*, pp 60-67, May 1996.

[38] Dagniele, G. and Massof, R.W., “Toward an Artificial Eye,” *IEEE Spectrum*, pp 20-29, May 1996.

[39] J.C. Candy and G.C. Temes, “Oversampled Methods for A/D and D/A Conversion,” in *Oversampled Delta-Sigma Data Converters*, IEEE Press, pp 1-29, 1992.

[40] R. Genov, G. Cauwenberghs “Charge-Mode Parallel Architecture for Matrix-Vector Multiplication,” *IEEE T. Circuits and Systems II,* vol. **48** (10), pp. 930-936, 2001.

[41] R. Genov, G. Cauwenberghs, “Stochastic Mixed-Signal VLSI Architecture for High-Dimensional Kernel Machines,” *Advances in Neural Information Processing Systems*, Cambridge, MA: MIT Press, vol. 14, 2002.

[42] R. Sarpeshkar, J. Kramer, et. al., “Analog VLSI Architectures for Motion Processing: From Fundamental Limits to System Applications,” *Proc. of The IEEE,* vol. **84** (7), pp. 969-987, 1996.

[43] K. Boahen, “Retinomorphic Vision Systems,” Proc. of 5th Int. Conf. MicroNeuro’96, 1996.

[44] A. Andreou and K. Boahen, “Translinear circuits in subthreshold mos,” *J. Analog Integrated Circ. Sig. Proc.,* March, 1996.

[45] K. Kotani, T. Shibata, et. al., “CMOS Charge-Transfer Preamplifier for Offset-Fluctuation Cancellation in Low-Power Converters,” *IEEE J. Solid-State Circuits,* vol. **33** (5), pp. 762-769, 1998.

[46] R. Gregorian and G.C. Temes, “Analog MOS Integrated Circuits for Signal Processing,” New York, NY: John Wiley, 1986.

[47] P. Pouliquen, K. Boahen and A.G. Andreou, “A Gray-code MOS Current-Mode Analog-to-digital Converter Design,” *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS’91),* pp 1924-1927, 1991.

[48] S. Signell, B. Johnsson, et al., “New A/D Converter Architectures Based on Gray Coding,” *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS’97),* pp 413-416, 1997.

[49] H. Kimura, A. Matsuzawa, T. Nakamura “A 10b 300MHz Interpolated-Parallel A/D Converter,” *IEEE T. Circuits and Systems II,* vol. **44** (2), pp. 65-85, 1997.

[50] K. Ono, T. Matsuura, et. al., “Error Suppressing Encode Logic of FDCL in 6-bit Flash A/D Converter,” *IEEE J. Solid-State Circuits,* vol. **32** (9), pp. 1460-1464, 1997.

[51] R. Harjani and T.A. Lee, “FRC: A Method for Extending the Resolution of Nyquist Rate Converters Using Oversampling,” *IEEE T. Circuits and Systems II,* vol. **45** (4), pp 482-494, 1998.

[52] O.J.A.P. Nys and E. Dijkstra, “On Configurable Oversampled A/D Converters,” *IEEE J. Solid-State Circuits,* vol. **28** (7), pp 736-742, 1993.

[53] J. Robert, G.C. Temes, V. Valencic, R. Dessoulavy, and P. Deval, “A 16-Bit Low-Voltage CMOS A/D Converter,” *IEEE J. Solid-State Circuits,* vol. **SC-22**, pp 157-163, 1987.

[54] C. Jansson, “A High-Resolution, Compact, and Low-Power ADC Suitable for Array Implementation in Standard CMOS,” *IEEE T. Circuits and Systems I,* vol. **42** (11), pp 904-912, 1995.

[55] P. Rombouts, W. De Wilde, and L. Weyten, “A 13.5-b 1.2-V Micropower Extended Counting A/D Converter,” *IEEE J. Solid-State Circuits,* vol. **36** (2), pp 176-183, 2001.

[56] T. Hayashi, Y. Inabe, K. Uchimura, T. Kimura, “A Multistage Delta-Sigma Modulator without Double Integration Loop,” *ISSCC Tech. Dig. Pap.*, vol. **39**, pp 182-183, 1986.

[57] T.C. Leslie and B. Singh, “An Improved Sigma-Delta Modulator Architecture,” *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS’90)*, pp 372-375, 1990.

[58] D.B. Ribner, “A Comparison of Modulator Networks for High-Order Oversampling Sigma-Delta Analog-to-Digital Conversion,” *IEEE T. Circuits and Systems*, vol. **38**, pp 145-159, 1991.

[59] G. Cauwenberghs and G.C. Temes, “Adaptive Digital Correction of Analog Errors in MASH ADCs — Part I. Off-Line and Blind On-Line Calibration,” *IEEE Trans. Circuits and Systems II*, vol. **47** (7), pp. 621-628, July 2000.

[60] A. Karanicolas and H.-S. Lee, “A 15b 1Ms/s Digitally Self-Calibrated Pipeline ADC,” *Proc. of IEEE Int. Solid-State Circuits Conference (ISSCC’93),* pp 60-61, 1993.

[61] G. Cauwenberghs, “Blind On-Line Digital Calibration of Multi-Stage Nyquist-Rate and Oversampled A/D Converters,” *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS’98),* Monterey CA, vol. 11, pp. 508-511, 1998.

[62] G. Liñán, R. Domínguez-Castro, S. Espejo, A. Rodríguez-Vázquez, “ACE16k: an Advanced Focal-Plane Analog Programmable Array Processor,” *Proc. European Solid-State Circuits Conference (ESS-CIRC’2001),* Frontier Group, pp. 216-219, 2001.

Vita

Roman Genov received the B.S. degree in Electrical Engineering from Rochester Institute of Technology, NY in 1996 and the M.S. degree in Electrical and Computer Engineering from Johns Hopkins University, Baltimore, MD in 1998.

He held engineering positions at Atmel Corporation, Columbia, MD in 1995 and Xerox Corporation, Rochester, NY in 1996. He was a visiting researcher in the Robot Learning Group at Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland in 1998 and in the Center for Biological and Computational Learning at Massachusetts Institute of Technology, Cambridge, MA in 1999. His research interests include analog and digital VLSI circuits, systems and algorithms for parallel signal processing and high-performance computing with application to pattern recognition, focal-plane imaging, autonomous system design, and low-power instrumentation.

Mr. Genov is a member of the IEEE Circuits and Systems Society and Computer Society. He received a Best Presentation Award at IEEE IJCNN'2000 and a Student Paper Contest Award at IEEE MWSCAS'2000.