Charge-Mode Parallel Architecture for Vector–Matrix Multiplication

Roman Genov, *Member, IEEE*, and Gert Cauwenberghs, *Member, IEEE*

***Abstract*—An internally analog, externally digital architecture for parallel vector–matrix multiplication is presented. A three-transistor unit cell combines a single-bit dynamic random-access memory and a charge injection device 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. A prototype 512 $\times$ 128 vector–matrix multiplier on a single 3 mm $\times$ 3 mm chip fabricated in standard 0.5-μm CMOS technology achieves 8-bit effective resolution and dissipates 0.5 pJ per multiply-accumulate.**

***Index Terms*—Analog array processors, analog-to-digital conversion (ADC), charge-injection device (CID), dynamic random-access memory (DRAM), support vector machines (SVM), vector–matrix multiplication (VMM), vector quantization (VQ).**

I. Introduction

REAL-TIME artificial vision systems for interactive human–machine interfaces [1] incur a significant amount of computation, in excess of even the most powerful processors available today. One of the most common, but computationally most expensive operations in machine vision and pattern recognition is that of vector–matrix multiplication (VMM) in large dimensions

$$Y^{(m)} = \sum_{n=0}^{N-1} W^{(m,n)} X^{(n)} \tag{1}$$

with N-dimensional input vector $X^{(n)}$, M-dimensional output vector $Y^{(m)}$, and $N \times M$ matrix elements $W^{(n,m)}$. In artificial neural networks, for instance, the matrix elements $W^{(m,n)}$ correspond to weights, or synapses, between neurons. The elements may also represent templates $X_m^{(n)} = W^{(m,n)}$ in a vector quantizer [2], or support vectors in a support vector machine [3].

Dedicated parallel VLSI architectures have been developed to speed up VMM computation, e.g., [4]. 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 [5]. The ideal scenario (in the case of vector–matrix multiplication) is where each processor performs one multiply and locally stores one coefficient. The advantage of this is a throughput that scales linearly with the dimensions of the implemented array. The recurring problem with digital implementation 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 (100–10 000) matrix dimensions.

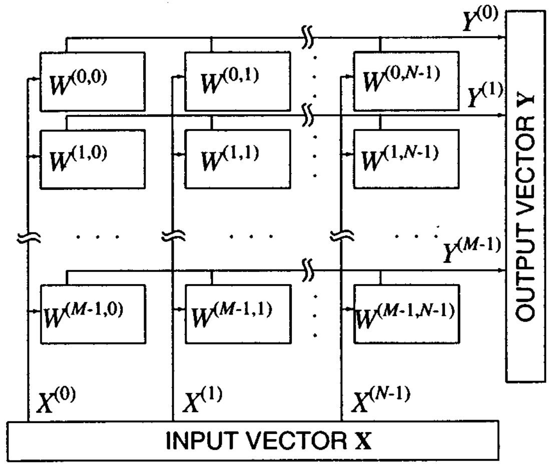

Fig. 1. General architecture for fully parallel vector–matrix multiplication (VMM).

Analog VLSI provides a natural medium to implement fully parallel computational arrays with high integration density and energy efficiency [6]. 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) requires an $M \times N$ array of cells, illustrated in Fig. 1. Each cell (m, n) computes the product of input component $X^{(n)}$ and matrix element $W^{(m,n)}$, and dumps the resulting current or charge on a horizontal output summing line. The device storing $W^{(m,n)}$ 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 [7]–[10], asynchronous transconductance-mode [11]–[13], or asynchronous current-mode [14] multiply-and-accumulate circuits.

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

This paper was recommended by Corresponding Editor A. Payne. Manuscript received March 12, 2001; revised October 1, 2001. This work was supported under Grants NSF MIP-9702346 and ONR N00014-99-1-0612.

The authors are with the Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: roman@bach.ece.jhu.edu; gert@bach.ece.jhu.edu).

Publisher Item Identifier S 1057-7130(01)11046-3.

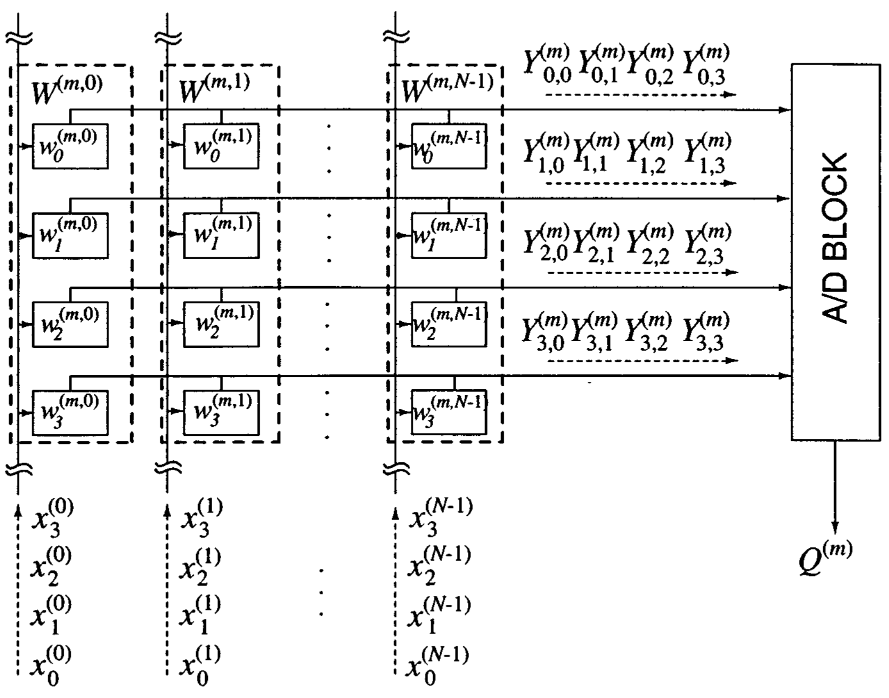
$W^{(m,0)}$
$W^{(m,1)}$
$W^{(m,N\text{-}1)}$
$w_0^{(m,0)}$ $w_1^{(m,0)}$ $w_2^{(m,0)}$ $w_3^{(m,0)}$
$w_0^{(m,1)}$ $w_1^{(m,1)}$ $w_2^{(m,1)}$ $w_3^{(m,1)}$
$w_0^{(m,N\text{-}1)}$ $w_1^{(m,N\text{-}1)}$ $w_2^{(m,N\text{-}1)}$ $w_3^{(m,N\text{-}1)}$
$Y_{0,0}^{(m)}$ $Y_{0,1}^{(m)}$ $Y_{0,2}^{(m)}$ $Y_{0,3}^{(m)}$
$Y_{1,0}^{(m)}$ $Y_{1,1}^{(m)}$ $Y_{1,2}^{(m)}$ $Y_{1,3}^{(m)}$
$Y_{2,0}^{(m)}$ $Y_{2,1}^{(m)}$ $Y_{2,2}^{(m)}$ $Y_{2,3}^{(m)}$
$Y_{3,0}^{(m)}$ $Y_{3,1}^{(m)}$ $Y_{3,2}^{(m)}$ $Y_{3,3}^{(m)}$
A/D BLOCK
$Q^{(m)}$
$x_3^{(0)}$ $x_2^{(0)}$ $x_1^{(0)}$ $x_0^{(0)}$
$x_3^{(1)}$ $x_2^{(1)}$ $x_1^{(1)}$ $x_0^{(1)}$
$x_3^{(N\text{-}1)}$ $x_2^{(N\text{-}1)}$ $x_1^{(N\text{-}1)}$ $x_0^{(N\text{-}1)}$

Fig. 2. Block diagram of one row in the matrix with binary encoded elements $w_i^{(m,n)}$, for a single m and with $I = 4$ bits. Data flow of bit-serial inputs $x_j^{(n)}$ and corresponding partial outputs $Y_{i,j}^{(m)}$, with $J = 4$ bits.

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. 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.

A mixed-signal array architecture with binary decomposed matrix and vector elements is described in Section II. VLSI implementation with experimental results from fabricated silicon are presented in Section III. Section IV quantifies the improvements in system precision obtained by postprocessing the quantized outputs of the array in the digital domain. Conclusions are presented in Section V.

II. Mixed-Signal Architecture

A. Internally Analog, Externally Digital Computation

The system presented 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 vector–matrix 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 (Fig. 2). 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

$$W^{(m,n)} = \sum_{i=0}^{I-1} 2^{-(i+1)} w_i^{(m,n)} \tag{2}$$

$$X^{(n)} = \sum_{j=0}^{J-1} 2^{-(j+1)} x_j^{(n)} \tag{3}$$

decomposing (1) into

$$Y^{(m)} = \sum_{n=0}^{N-1} W^{(m,n)} X^{(n)} = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} 2^{-(i+j+2)} Y_{i,j}^{(m)} \tag{4}$$

with binary–binary VMM partials

$$Y_{i,j}^{(m)} = \sum_{n=0}^{N-1} w_i^{(m,n)} x_j^{(n)}. \tag{5}$$

The proposed mixed-signal approach is to compute and accumulate the binary–binary partial products (5) using an analog VMM array, and to combine the quantized results in the digital domain according to (4).

B. Array Architecture and Data Flow

To conveniently implement the partial products (5), the binary encoded matrix elements $w_i^{(m,n)}$ are stored in bit-parallel form, and the binary encoded inputs $x_j^{(n)}$ are presented in bit-serial fashion. The bit-serial format was first proposed and demonstrated in [8], 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 [9].

One row of I-bit encoded matrix elements uses I rows of binary cells. Therefore, to store an $M \times N$ *digital* matrix $W^{(m,n)}$, an array of $MI \times N$ *binary* cells $w_i^{(m,n)}$ is needed. One bit of an input vector is presented each clock cycle, taking J clock cycles of partial products (5) to complete a full computational cycle (1). The input binary components $x_j^{(n)}$ are presented least significant bit (LSB) first, to facilitate the digital postprocessing to obtain (4) from (5) (as elaborated in Section IV).

Fig. 2 depicts one row of matrix elements $W^{(m,n)}$ in the binary encoded architecture, comprising I rows of binary cells $w_i^{(m,n)}$, where $I = 4$ in the example shown. The data flow is illustrated for a digital input series $x_j^{(n)}$ of $J = 4$ bits, LSB first (i.e., descending index j). The corresponding analog series of outputs $Y_{i,j}^{(m)}$ in (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 (4) of the quantized series of output vectors yields the final digital result (1).

The quantization scheme used is critical to system performance. As shown in Section IV, appropriate postprocessing in the digital domain to obtain (4) from the quantized partial products $Y_{i,j}^{(m)}$ 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 (5). A dense and efficient charge-mode VLSI implementation is described next.

III. Charge-Mode VLSI Implementation

A. CID/DRAM Cell and Array

The elementary cell combines a CID computational unit [8], [9], computing one argument of the sum in (5), with a DRAM storage element. The cell stores one bit of a matrix element $w_i^{(m,n)}$, performs a one-quadrant binary–binary multiplication of $w_i^{(m,n)}$ and $x_j^{(n)}$, and accumulates the result across cells with common m and i indexes. The circuit diagram and operation of the cell are given in Fig. 3. An array of cells thus performs (unsigned) binary multiplication (5) of matrix $w_i^{(m,n)}$ and vector $x_j^{(n)}$ yielding $Y_{i,j}^{(m)}$, 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 Fig. 3. Transistors M1 and M2 comprise a dynamic random-access memory (DRAM) cell, with switch M1 controlled by *Row Select* signal $RS_i^{(m)}$. When activated, the binary quantity $w_i^{(m,n)}$ 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 nondestructive manner [8], [9], [15].

The cell operates in two phases: *Write* and *Compute*. When a matrix element value is being stored, $x_j^{(n)}$ 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,

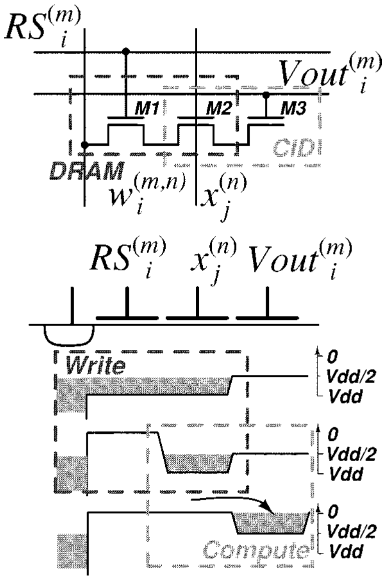

Fig. 3. CID computational cell with integrated DRAM storage (top). Charge transfer diagram for active write and compute operations (bottom).

if $w_i^{(m,n)}$ is low, or charge is removed, if $w_i^{(m,n)}$ is high. The amount of charge stored, $\triangle Q$ or 0, corresponds to the binary value $w_i^{(m,n)}$.

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 nonzero charge stored, and if the potential on the gate of M2 drops below that of M3 [8]. This condition implies a logical AND, i.e., unsigned binary multiplication, of $w_i^{(m,n)}$ and $x_j^{(n)}$. The multiply-and-accumulate operation is then completed by capacitively sensing the amount of charge transferred onto the electrode of M3, the output summing node. To this end, the voltage on the output line, left floating after being precharged to $Vdd/2$, is observed. When the charge transfer is active, the cell contributes a change in voltage

$$\triangle V_{out} = \triangle Q / C_{M3} \tag{6}$$

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 $x_j{}^{(n)}$, the transferred charge returns to the storage node M2. The CID computation is nondestructive and intrinsically reversible [8], and DRAM refresh is only required to counteract junction and subthreshold leakage.

The bottom diagram in Fig. 3 depicts the charge transfer timing diagram for write and compute operations in the case when both $w_i{}^{(m,n)}$ and $x_j{}^{(n)}$ are of logic level 1. A logic level 0 for $w_i{}^{(m,n)}$ is represented as Vdd, and a logic level 1 is represented as $Vdd/2$, where Vdd is the supply voltage. For $x_j{}^{(n)}$, 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 Fig. 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.

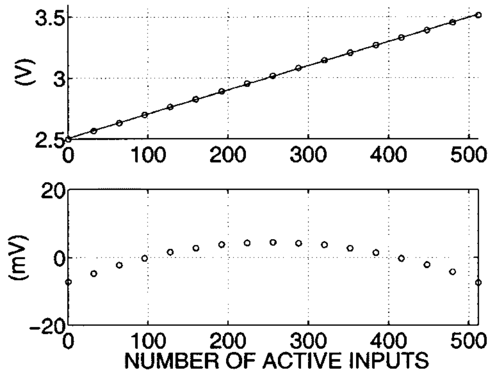

Fig. 4. Voltage transfer characteristic (top) and integral nonlinearity (bottom) for a row of 512 CID/DRAM cells, simulated using *SpectreS* with MOS model parameters extracted from a 0.5-μm process.

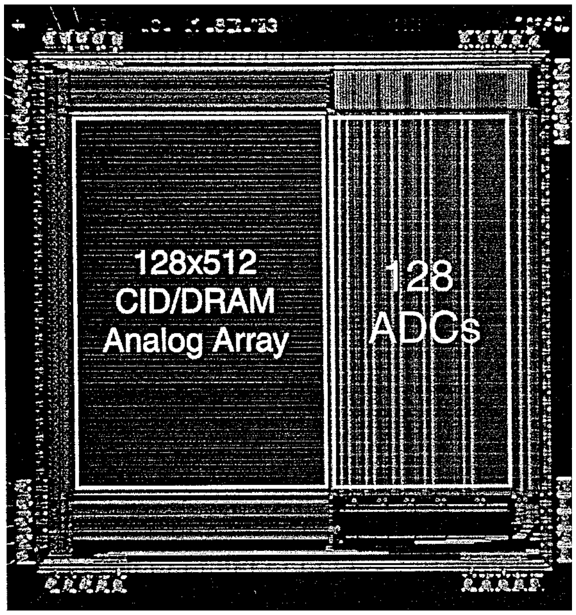

Fig. 5. Micrograph of the mixed-signal VMM prototype, containing an array of 512 $\times$ 128 CID/DRAM cells, and a row-parallel bank of 128 flash ADCs. Die size is 3 mm $\times$ 3 mm in 0.5-μm CMOS technology.

B. Experimental Results

We designed, fabricated and tested a VLSI prototype of the vector–matrix multiplier, integrated on a 3 $\times$ 3 mm^2 die in 0.5-μ m CMOS technology. The chip contains an array of 512 $\times$ 128 CID/DRAM cells, and a row–parallel bank of 128 gray-code flash ADCs. Fig. 5 depicts the micrograph and system floorplan of the chip. The layout size of the CID/DRAM cell is 8λ $\times$ 45λ with $\lambda = 0.3$ μm.

The mixed-signal VMM 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.

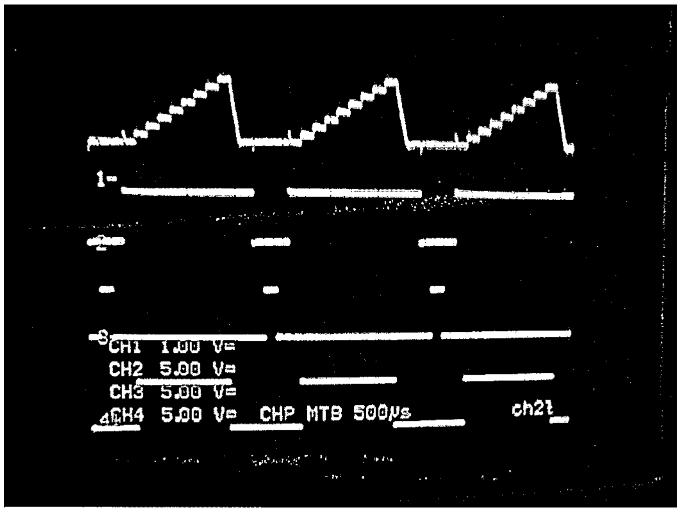

Fig. 6. Measured linearity of the computational array. The number of active charge-transfer cells is swept in increments of 64, with the analog voltage output on the sense line shown on the top scope trace.

Fig. 6 shows the measured linearity of the computational array. The number of active cells on one row transferring charge to the output line is incremented in steps of 64. The case shown is where all binary weight storage elements are actively charged, and an all-ones sequence of bits is shifted through the input register, initialized to all-zeros bit values. For every shift of 64 positions in the input, a computation is performed and the result is observed on the output sense line. The experimentally observed linearity agrees with the simulation in Fig. 4.

The chip contains 128 row–parallel 6-bit 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 $Y_{i,j}^{(m)}$ over i (rows) and j (time) according to (4). Issues of precision and complexity in the implementation of (4) are studied later.

IV. 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 (4) in the digital domain, from quantized estimates of the partial outputs (5). The effect of averaging the quantization error over a large number of quantized values of $Y_{i,j}^{(m)}$ boosts the precision of the digital estimate of $Y^{(m)}$, beyond the intrinsic resolution of the analog array and the A/D quantizers used.

A. Accumulation and Quantization

The outputs $Y_{i,j}^{(m)}$ for a single m obtained from the analog array over J clock cycles can be conceived as an $I \times J$ matrix, shown in Fig. 2. Elements of this matrix located along diagonals (i.e., elements with a common value of $i + j$) have identical binary weight in (4). Therefore, the summation in (4) could be rearranged as

$$Y^{(m)} = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} 2^{-(i+j+2)} Y_{i,j}^{(m)} = \sum_{k=0}^{K-1} 2^{-(k+2)} Y'^{(m)}_{k} \quad (7)$$

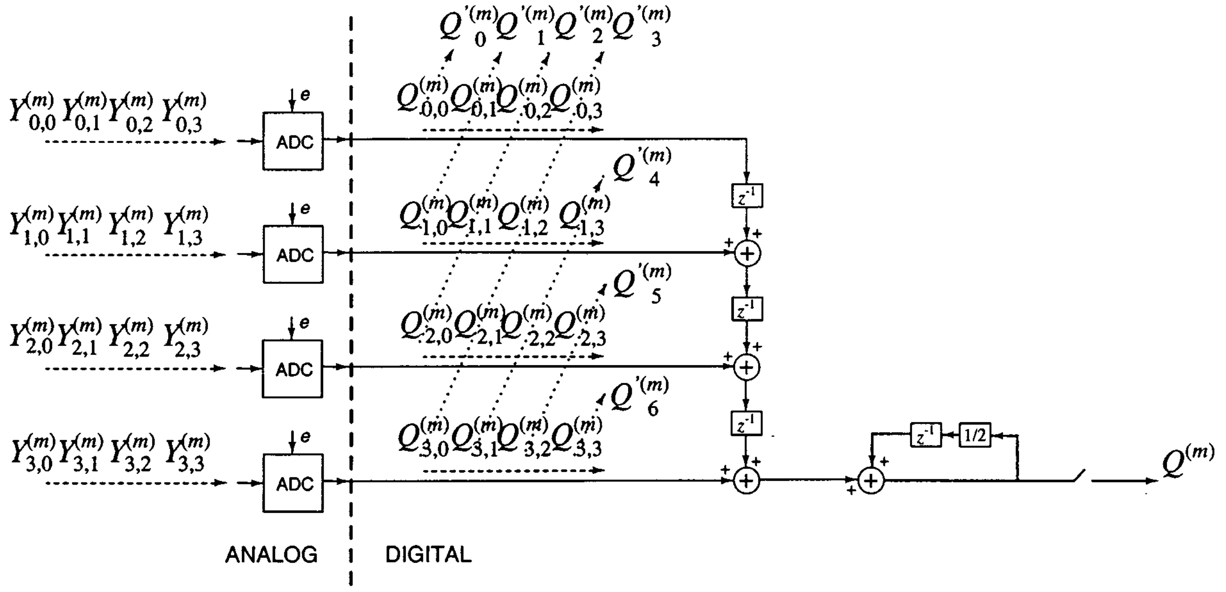

Fig. 7. Diagram for the A/D quantization and digital postprocessing block in Fig. 2, using row–parallel flash A/D converters. The example shown is for a single m, LSB-first bit-serial inputs, and $I = J = 4$.

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

$$Y'^{(m)}_k = \sum_{i=\kappa(k, J)}^{k-\kappa(k, I)} Y^{(m)}_{i, k-i} \tag{8}$$

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 (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. We opted for the former, and implemented a parallel array of low-resolution (6-bit) flash ADCs, one for each row output i.

B. Row–Parallel Flash A/D Conversion

Consider first 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. Fig. 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 (8) and (7) are then performed in the digital domain:

$$Q^{(m)} = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} 2^{-(i+j+2)} Q^{(m)}_{i,j} = \sum_{k=0}^{K-1} 2^{-(k+2)} Q'^{(m)}_k \tag{9}$$

and

$$Q'^{(m)}_k = \sum_{i=\kappa(k, J)}^{k-\kappa(k, I)} Q^{(m)}_{i, k-i}. \tag{10}$$

A block diagram for a digital implementation is shown on the right of Fig. 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 (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 VMM 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 what follows 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

$$Q^{(m)}_{i,j} = Y^{(m)}_{i,j} + e^{(m)}_{i,j} \tag{11}$$

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 (11) could also include effects of noise and nonlinear distortion in the analog summation (5), although in practice the precision of the array exceeds the ADC resolution, as shown in the experimental data of Section III-B. From (9) and (11), the error in the digitally constructed output

$$Q^{(m)} = Y^{(m)} + E^{(m)} \tag{12}$$

can then be expanded as

$$E^{(m)} = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} 2^{-(i+j+2)} e_{i,j}^{(m)}. \tag{13}$$

Define s the full-scale range of the ADC acquiring $Q_{i,j}^{(m)}$, and S the corresponding range of the constructed digital output $Q^{(m)}$. Then according to (9),

$$S = s \sum_{i=0}^{I-1} 2^{-(i+1)} \sum_{j=0}^{J-1} 2^{-(j+1)} = s\left(1 - 2^{-I}\right)\left(1 - 2^{-J}\right) \tag{14}$$

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.

Let the variance of the uniform quantization noise e in (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 (13). Each signal component, $Q_{i,j}^{(m)}$, 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 (13), and the total variance σ_E^2 of the output error E is expressed as

$$\begin{aligned}\sigma_E^2 &= \sigma_e^2 \sum_{i=0}^{I-1} 2^{-2(i+1)} \sum_{j=0}^{J-1} 2^{-2(j+1)} \\ &= \sigma_e^2 \frac{1 - 2^{-2I}}{3} \frac{1 - 2^{-2J}}{3}\end{aligned} \tag{15}$$

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

$$\frac{S}{\sigma_E} \approx 3 \frac{s}{\sigma_e} \tag{16}$$

for large I and J. In other words, by quantizing each array output $Y_{i,j}^{(m)}$ 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 $\mathcal{M}_e = (1/2)\Delta$, and the variance $\sigma_e^2 = (1/3)\Delta^2$, yielding the relation

$$\mathcal{M}_e = \frac{\sqrt{3}}{2} \sigma_e \tag{17}$$

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

$$\mathcal{M}_E = 0.675\sigma_E. \tag{18}$$

This allows to express the SQNR gain in (16) as a gain in *median resolution*

$$\frac{S}{\mathcal{M}_E} \approx 3 \frac{\sqrt{3}/2}{0.675} \frac{s}{\mathcal{M}_e} \approx 3.85 \frac{s}{\mathcal{M}_e} \tag{19}$$

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 are significant, since the implementation cost is exponential in the number of bits. For the VMM processor described in Section III-B, a 6-bit flash ADC architecture gives 8-bit (median) output resolution. The choice of 6-bit ADC resolution was dictated by two considerations. First, a larger resolution would have incurred a disproportionate cost in implementation, since the 128 parallel ADCs already comprise a significant portion of the total silicon area as shown in Fig. 5. Second, a lower resolution would compromise the 7 “bits” of precision available from the analog array (Figs. 4 and 6).

V. Conclusion

A charge-mode VLSI architecture for parallel vector–matrix 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. The combination of analog array processing and digital post-processing also enhances the precision of the digital VMM output, exceeding the resolution of the quantized analog array outputs by 2 bits. Significantly larger gains in precision could be obtained by exploiting the statistics of binary terms in the analog summation (5) [18].

Fine-grain massive parallelism and distributed memory, in an array of 3-transistor 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 VMM prototype fabricated in 0.5-μm CMOS offers 2 $\times$ 10^{12} binary MACS (multiply accumulates per second) per Watt of power. This opens up possibilities for low-power real-time pattern recognition in human–machine interfaces [1], artificial vision [16], and vision prostheses [17].

Acknowledgment

The authors would like to thank the MOSIS Foundry Service for fabricating the Chips.

References

[1] C. P. Papageorgiou, M. Oren, and T. Poggio, “A general framework for object detection,” in *Proc. Int. Conf. Computer Vision*, 1998.
[2] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Norwell, MA: Kluwer, 1992.

[3] V. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed. New York: Springer-Verlag, 1999.
[4] 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, 1996, vol. 8, pp. 619–625.
[5] 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.
[6] A. Kramer, "Array-based analog computation," *IEEE Micro*, vol. 16, no. 5, pp. 40–49, 1996.
[7] A. Chiang, "A programmable CCD signal processor," *IEEE J. Solid-State Circuits*, vol. 25, no. 6, pp. 1510–1517, 1990.
[8] C. Neugebauer and A. Yariv, "A parallel analog CCD/CMOS neural network IC," in *Proc. IEEE Int. Joint Conf. Neural Networks (IJCNN'91)*, vol. 1, Seattle, WA, 1991, pp. 447–451.
[9] V. Pedroni, A. Agranat, C. Neugebauer, and A. Yariv, "Pattern matching and parallel processing with CCD technology," in *Proc. IEEE Int. Joint Conf. Neural Networks (IJCNN'92)*, vol. 3, 1992, pp. 620–623.
[10] G. Han and E. Sanchez-Sinencio, "A general purpose neuro-image processor architecture," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS'96)*, vol. 3, 1996, pp. 495–498.
[11] 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, 1989, pp. 191–196.
[12] F. Kub, K. Moon, I. Mack, and F. Long, "Programmable analog vector–matrix multipliers," *IEEE J. Solid-State Circuits*, vol. 25, pp. 207–214, 1990.
[13] 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, pp. 488–497, May 1992.
[14] 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 Trans. Neural Networks*, vol. 2, pp. 205–213, 1991.
[15] M. Howes and D. Morgan, Eds., *Charge-Coupled Devices and Systems*. New York: Wiley, 1979.
[16] T. Poggio and D. Beymer, "Learning to see," *IEEE Spectrum*, pp. 60–67, May 1996.
[17] G. Dagniele and R. W. Massof, "Toward an artificial eye," *IEEE Spectrum*, pp. 20–29, May 1996.
[18] R. Genov and G. Cauwenberghs, "Stochastic mixed-signal VLSI architecture for high-dimensional kernel machines," in Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press, 2002, vol. 14, to be published.

Roman Genov (M'97) received the B.S. degree in electrical engineering from Rochester Institute of Technology, Rochester, NY, in 1996 and the M.S. degree in electrical and computer engineering from the Johns Hopkins University, Baltimore, MD, in 1998, where he is currently working toward the Ph.D. degree.

He held engineering positions with Atmel Corporation, Columbia, MD, in 1995 and Xerox Corporation, Rochester, in 1996. He was a visiting researcher with the Robot Learning Group of the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, in 1998 and with the Center for Biological and Computational Learning at Massachusetts Institute of Technology, Cambridge, in 1999. His research interests include mixed-signal VLSI systems, machine learning, image and signal processing, and neuromorphic systems.

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.

Gert Cauwenberghs (S'89–M'94) received the engineer's degree in applied physics from the University of Brussels, Belgium, in 1988, and the M.S. and Ph.D. degrees in electrical engineering from the California Institute of Technology, Pasadena, in 1989 and 1994.

In 1994, he joined Johns Hopkins University, Baltimore, MD, where he is an Associate Professor of electrical and computer engineering. During 1998–1999, he was on sabbatical as Visiting Professor of Brain and Cognitive Science with the Center for Computational and Biological Learning, Massachusetts Institute of Technology, Cambridge, and with the Center for Adaptive Systems, Boston University, Boston, MA. He recently coedited the book *Learning on Silicon* (Boston, MA: Kluwer, 1999). His research covers VLSI circuits, systems, and algorithms for parallel signal processing, adaptive neural computation, and low-power coding and instrumentation.

Dr. Cauwenberghs is an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: ANALOG AND DIGITAL SIGNAL PROCESSING and the IEEE *Sensors Journal*. He is Chair of the IEEE Circuits and Systems Society Analog Signal Processing Technical Committee and has organized special sessions at conferences and special issues of journals on learning, adaptation, and memory. He was Francqui Fellow of the Belgian American Educational Foundation in 1988, and received the National Science Foundation Career Award in 1997, the Office of Naval Research Young Investigator Award in 1999, and the Presidential Early Career Award for Scientists and Engineers (Pecase) in 2000.