Embedded Dynamic Memory and Charge-Mode Logic for Parallel Array Processing

Roman Genov and Gert Cauwenberghs
Department of Electrical and Computer Engineering
Johns Hopkins University, Baltimore, MD 21218, U.S.A.
E-mail: {roman,gert}@bach.ece.jhu.edu

Abstract—**We present a mixed-signal distributed VLSI architecture for massively parallel array processing, with fine-grain embedded memory. The three-transistor processing element in the array combines a charge injection device (CID) binary multiplier and analog accumulator with embedded dynamic random-access memory (DRAM). A prototype 512 $\times$ 128 vector-matrix multiplier on a single 3 mm $\times$ 3 mm chip fabricated in standard CMOS 0.5 μm technology achieves 8-bit effective resolution, dissipates 0.5 pJ per multiply-accumulate and offers 2 $\times$ 10^{12} binary MACS (multiply accumulates per second) per Watt of power.**

I. Introduction

One of the greatest challenges in the performance of computer systems today is limited memory bandwidth. Conventional solutions to the speed mismatch between microprocessors and memory devote a large fraction of the transistors and area of the chips to static memory caches, leading to sub-optimal computational efficiency and silicon area. Embedded designs with memory and logic integrated together are clearly more desirable for memory intensive tasks.

We propose a massively parallel fine-grain array processor architecture with each cell containing a computing device and a storage element. We employ a multiply-accumulate processing element as a computing device to perform very computationally intensive operation, vector-matrix multiplication (VMM) in large dimensions. VMM in large dimensions is one of the most common, but computationally most expensive operation in algorithms for machine vision, image classification and pattern recognition:

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

with N-dimensional input vector $X^{(n)}$, M-dimensional output vector $Y^{(m)}$, and $N \times M$ matrix elements $W^{(n,m)}$.

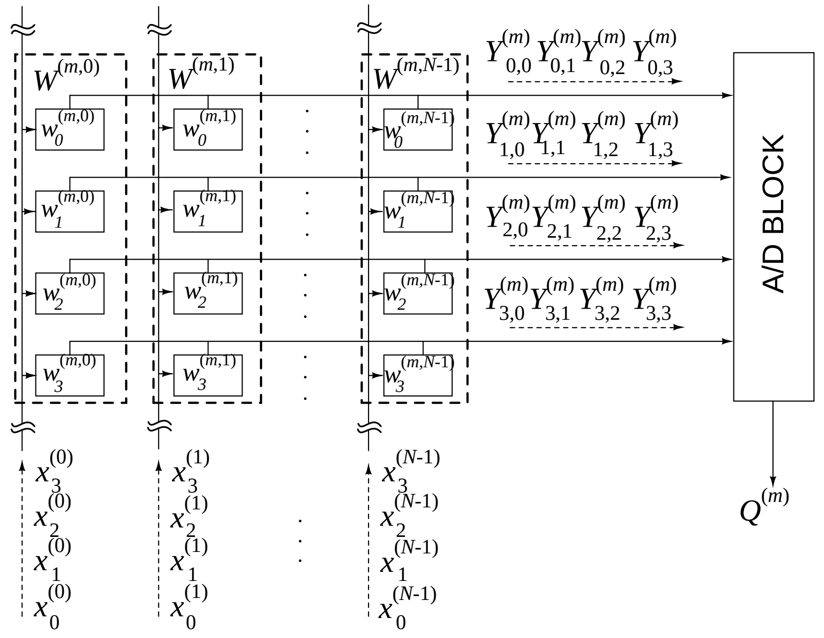
Architectures with distributed processors and embedded memories have been central to recent efforts in implementing parallel high-performance processors. Numerous approaches exist to digital distributed array processing: MIT pixel-parallel image processor [1], NEC integrated memory array processor for vision applications [2], computational RAM [3], FPGA-based bit-level matrix multiplier [4]. Array-based analog computation has been developed by SGS-Thomson Microelectronics [5] for numerous applications in signal processing and Pouliquen, Andreou et. al. in pattern recognition [6].

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, or do incorporate memories and digital processing elements together, but tend to use a lot of silicon area to implement those, significantly limiting the dimensions of the matrices operated on. A fine-grain, fully-parallel architecture, that integrates memory and processing elements, yields high computational throughput and high density of integration. The ideal scenario for array processing (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. Analog VLSI provides a natural medium to implement fully parallel computational arrays with high integration density and energy efficiency [5]. 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, each cell containing a product computing device and a storage element. 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], [8], [9], [10], asynchronous transconductance-mode [11], [12], [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 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.

This work was supported by NSF MIP-9702346 and ONR N00014-99-1-0612. Chips were fabricated through the MOSIS foundry service.



$W^{(m,0)}$
$W^{(m,1)}$
$W^{(m,N\text{-}1)}$
$Y^{(m)}_{0,0}\ Y^{(m)}_{0,1}\ Y^{(m)}_{0,2}\ Y^{(m)}_{0,3}$
$Y^{(m)}_{1,0}\ Y^{(m)}_{1,1}\ Y^{(m)}_{1,2}\ Y^{(m)}_{1,3}$
$Y^{(m)}_{2,0}\ Y^{(m)}_{2,1}\ Y^{(m)}_{2,2}\ Y^{(m)}_{2,3}$
$Y^{(m)}_{3,0}\ Y^{(m)}_{3,1}\ Y^{(m)}_{3,2}\ Y^{(m)}_{3,3}$
A/D BLOCK
$Q^{(m)}$
$x^{(0)}_3\ x^{(0)}_2\ x^{(0)}_1\ x^{(0)}_0$
$x^{(1)}_3\ x^{(1)}_2\ x^{(1)}_1\ x^{(1)}_0$
$x^{(N\text{-}1)}_3\ x^{(N\text{-}1)}_2\ x^{(N\text{-}1)}_1\ x^{(N\text{-}1)}_0$

Fig. 1. 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.

A mixed-signal array architecture with binary decomposed matrix and vector elements is described in Section II. VLSI implementation is presented in Section III. Section IV quantifies the improvements obtained 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 V. Conclusions are presented in Section VI.

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 pre-processing and post-processing 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. 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:

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

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

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].

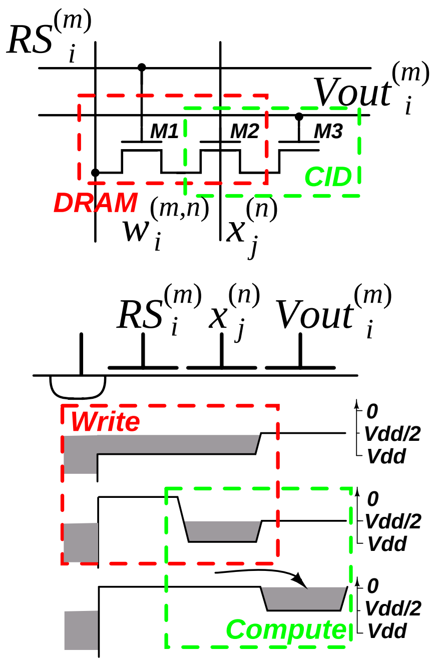


Fig. 2. CID computational cell with integrated DRAM storage (*top*). Charge transfer diagram for active write and compute operations (*bottom*).

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).

Figure 1 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 indices. The circuit diagram and operation of the cell are given in Figure 2. 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 Figure 2. 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 non-destructive 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, 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 non-zero 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 pre-charged to $Vdd/2$, is observed. When the charge transfer is active, the cell contributes a change in voltage

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

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 non-destructive and intrinsically reversible [8], and DRAM refresh is only required to counteract junction and subthreshold leakage.

The bottom diagram in Figure 2 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, and a computational cycle of 10 μs with power consumption of 50 nW per cell. Experimental results from a fabricated prototype are presented next.

B. Experimental Results

We designed, fabricated and tested a VLSI prototype of the inner-product array processor, integrated on a 3×3 mm^2 die in 0.5 μm CMOS technology. The chip contains an array of

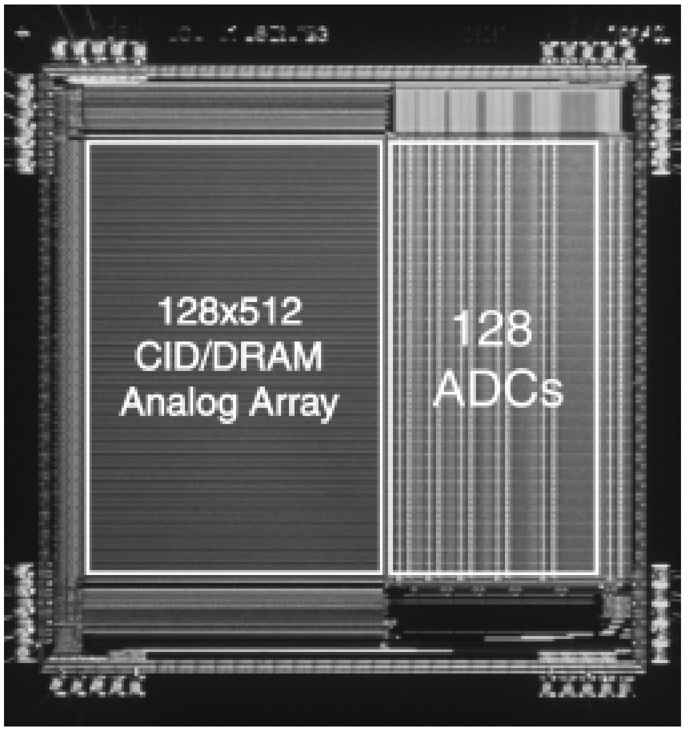


Fig. 3. Micrograph of the mixed-signal VMM prototype, containing an array of 512×128 CID/DRAM cells, and a row-parallel bank of 128 flash ADCs. Die size is $3\ \mathrm{mm} \times 3\ \mathrm{mm}$ in 0.5 μm CMOS technology.

512×128 CID/DRAM cells, and a row-parallel bank of 128 gray-code flash ADCs. Figure 3 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 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.

Figure 4 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 [16]. The feed-through input dependent offsets are compensated for as described in Section V.

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 below.

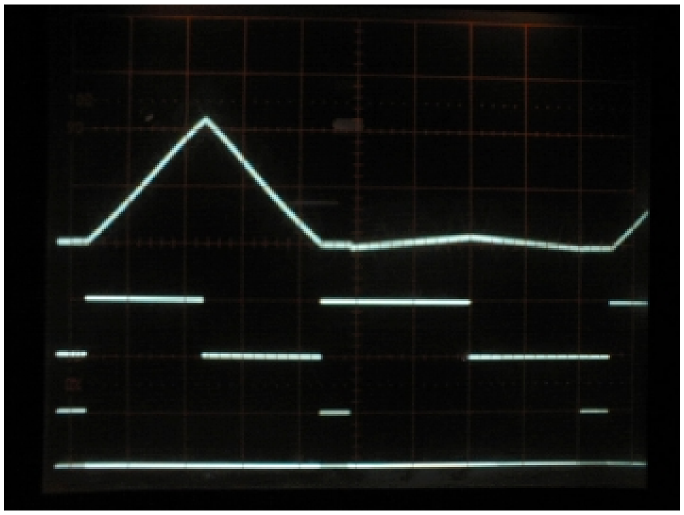


Fig. 4. Measured linearity of the computational array. Two cases are shown: all binary weight storage elements are actively charged (*left*) and discharged (*right*). All logic "1" sequence of bits is shifted through the input register, initialized to all-"0" bit values. For every 1-bit shift, a computation 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.

IV. Quantization and Digital Resolution Enhancement

A. Accumulation and Quantization

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).

We quantize all $I \times J$ values of ${Y_{i,j}}^{(m)}$ using row parallel flash A/D converters. Figure 5 presents the corresponding architecture, shown for a single output vector component m. The partials summation is then performed in the digital domain:

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

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

$$Q_k'^{(m)} = \sum_{i=\kappa(k,J)}^{k-\kappa(k,I)} Q_{i,k-i}^{(m)} \ , \quad (8)$$

with $\kappa(k, I) \equiv \max(0, k - I + 1)$ and $\kappa(k, J) \equiv \max(0, k - J + 1)$. A block diagram for a digital implementation is shown on the right of Figure 5.

As shown in [16], 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. We obtain an improvement in signal-to-quantization-noise ratio of a factor 3 and a median resolution gain of approximately 2 bits over the resolution of each ADC.

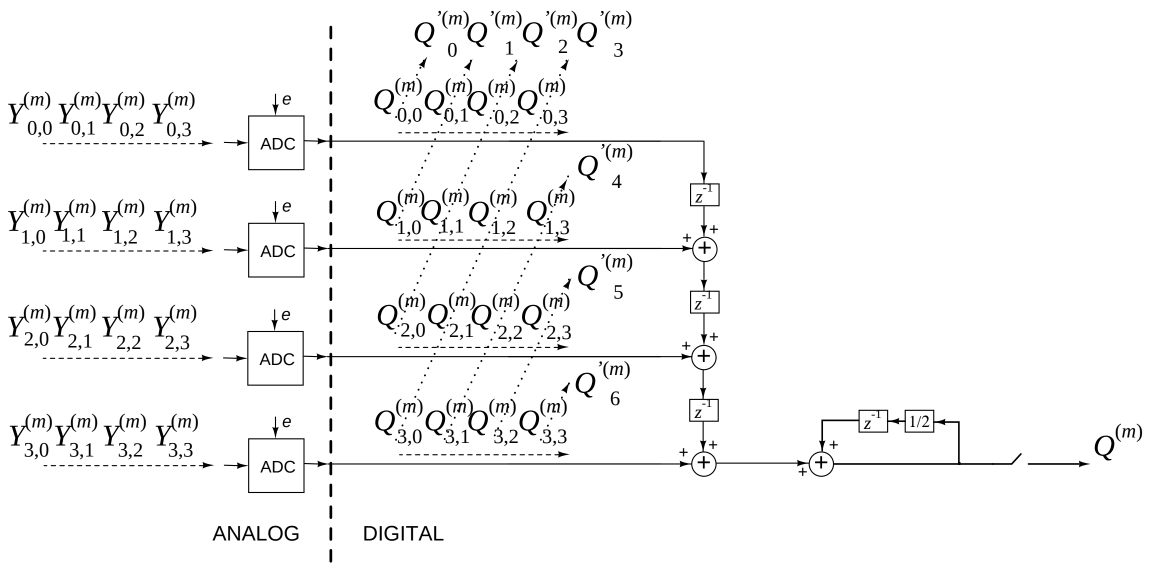

$Q'^{(m)}_0$ $Q'^{(m)}_1$ $Q'^{(m)}_2$ $Q'^{(m)}_3$
$Y^{(m)}_{0,0}$ $Y^{(m)}_{0,1}$ $Y^{(m)}_{0,2}$ $Y^{(m)}_{0,3}$
$Y^{(m)}_{1,0}$ $Y^{(m)}_{1,1}$ $Y^{(m)}_{1,2}$ $Y^{(m)}_{1,3}$
$Y^{(m)}_{2,0}$ $Y^{(m)}_{2,1}$ $Y^{(m)}_{2,2}$ $Y^{(m)}_{2,3}$
$Y^{(m)}_{3,0}$ $Y^{(m)}_{3,1}$ $Y^{(m)}_{3,2}$ $Y^{(m)}_{3,3}$
e
ADC
$Q^{(m)}_{0,0}$ $Q^{(m)}_{0,1}$ $Q^{(m)}_{0,2}$ $Q^{(m)}_{0,3}$
$Q^{(m)}_{1,0}$ $Q^{(m)}_{1,1}$ $Q^{(m)}_{1,2}$ $Q^{(m)}_{1,3}$
$Q^{(m)}_{2,0}$ $Q^{(m)}_{2,1}$ $Q^{(m)}_{2,2}$ $Q^{(m)}_{2,3}$
$Q^{(m)}_{3,0}$ $Q^{(m)}_{3,1}$ $Q^{(m)}_{3,2}$ $Q^{(m)}_{3,3}$
$Q'^{(m)}_4$
$Q'^{(m)}_5$
$Q'^{(m)}_6$
z^{-1}
1/2
$Q^{(m)}$
ANALOG
DIGITAL

Fig. 5. Diagram for the A/D quantization and digital postprocessing block in Figure 1, using row-parallel flash A/D converters. The example shown is for a single m, LSB-first bit-serial inputs, and $I = J = 4$.

V. Multi-Chip Architecture with Offset Compensation

A. Multi-Chip Architecture

The proposed method of on-chip VMM computation allows for an array size of 1000 × 1000, in 0.35 μm CMOS technology implemented on a 6 × 6 mm die. Computations on matrices of higher dimensionality, at maximum possible precision, can be performed by using multiple VMM 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 VMM chip. These offset terms can be compensated for in the multi-chip architecture as described in the next section.

B. Offset Compensation

In Section III 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 charge-mode analog addition on a single line. Because of linearity errors the result of such a computation is valid up to approximately 7 bits. We have discussed the effect of these errors and ADC resolution on the overall system precision in Sections IV.

Other significant sources of error in analog array-based computation are input-output feedthrough and leakage current in DRAM storage cells. The architecture presented is capable of compensating these analog computation errors in digital domain by using an extra reference VMM chip as shown in Figure 6.

TABLE I
Input-output mapping of CID/DRAM computational cell. The input-output feedthrough error is introduced when the input is logic "1".

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

The input-output feedthrough error is a result of capacitive coupling of input (vertical) lines onto the output (horizontal) lines through parasitic capacitances (metal lines overlap capacitance, gate-to-diffusion capacitance, etc). From Section III 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": ${w_i}^{(m,n)} = Vdd/2$ (charge stored) and, in the computation phase, ${x_j}^{(n)} = 0V$ (input switched). 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 of the input line ${x_j}^{(n)}$, 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. We model this effect as shown in Table I. The output of a single cell is denoted as ${y_{i,j}}^{(m,n)}$.

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

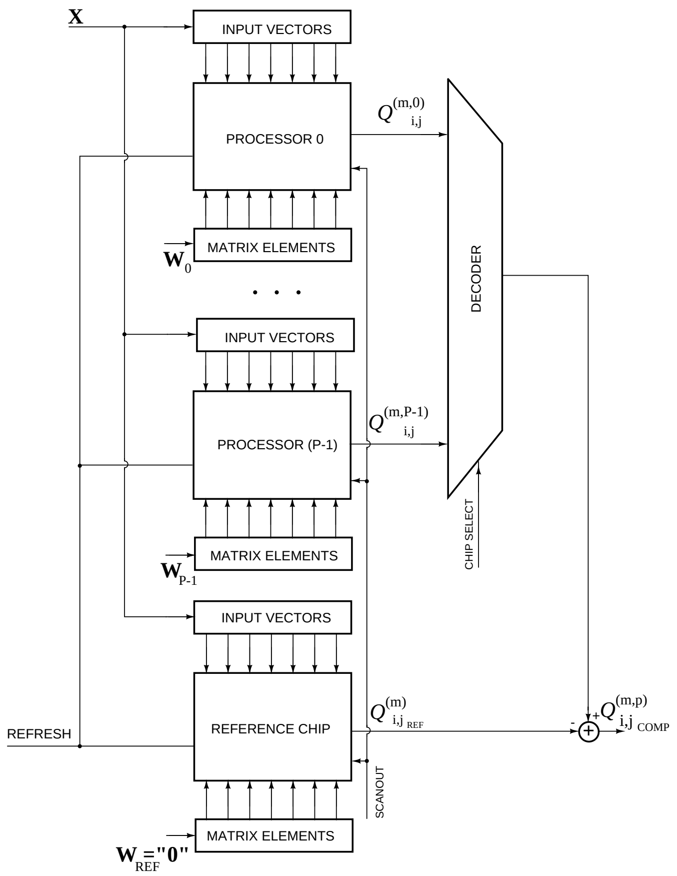


Fig. 6. Multi-Chip Architecture with Offset Compensation.

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 (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 order to compensate for these offsets, once computation has been performed on chips, the output of the reference chip is subtracted from the output of processors in digital domain as described in more detail in [17]. In the multi-chip configuration shown in Figure 6, 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 VMM 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.

VI. CONCLUSIONS

A charge-mode VLSI array processor for matrix operations in large dimensions ($N, M = $ 100–10,000) has been presented. The architecture embeds storage and multiplication in distributed fashion, down to the cellular level. With only three transistors, the cell for multiplication and storage contains little more than either a DRAM or a CID cell. This makes the analog cell very compact and low power, and the regular array of cells provides for a scalable architecture that can easily be extended. Fine-grain massive parallelism and distributed memory provide computational efficiency (bandwidth to power consumption ratio) exceeding that of digital multiprocessors and DSPs by several orders of magnitude. A 9mm^2 512 $\times$ 128 VMM prototype fabricated in 0.5 μm CMOS offers 2×10^{12} binary MACS (multiply accumulates per second) per Watt of power.

REFERENCES

[1] 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.
[2] Y. Fujita, S.A. Kyo, et. al., 10 GIPS SIMD processor for PC-based real time vision applications -architecture, algorithm implementation and language support,” *Proc. of IEEE International Workshop on Computer Architecture for Machine Perception, CAMP 97.*, pp. 22-32, 1997.
[3] D.G. Elliott, M. Stumm, et. al., “Computational RAM: implementing processors in memory,” *IEEE Design & Test of Computers*, vol. **16** (1), pp. 32-41, 1999.
[4] A. Amira, A. Bouridane, et. al., “A high throughput FPGA implementation of a bit-level matrix product,” *IEEE Workshop on Signal Processing Systems, SiPS 2000*, pp. 356-364, 2000.
[5] A. Kramer, “Array-based analog computation,” *IEEE Micro,* vol. **16** (5), pp. 40-49, 1996.
[6] P. Pouliquen, A.G. Andreou, K. Strohbehn “Winner-takes-all associative memory: a hamming distance vector quantizer,” *Journal of Analog Integrated Circuits and Signal Processing,* vol. **13**, pp. 211-222, 1997.
[7] A. Chiang, “A programmable CCD signal processor,” *IEEE Journal of Solid-State Circuits,* vol. **25** (6), pp. 1510-1517, 1990.
[8] 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.
[9] 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.
[10] 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.
[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, pp 191-196, 1989.
[12] 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.
[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** (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 Transactions on Neural Networks,* vol. **2** (2), pp 205-213, 1991.
[15] M. Howes, D. Morgan, Eds., *Charge-Coupled Devices and Systems,* John Wiley & Sons, 1979.
[16] R. Genov and G. Cauwenberghs, “Charge-Mode Parallel Architecture for Matrix-Vector Multiplication,” *Proc. 43rd IEEE Midwest Symp. Circuits and Systems (MWSCAS’2000),* Lansing MI, August 8-11, 2000.
[17] R. Genov, G. Cauwenberghs, “Analog Array Processor with Digital Resolution Enhancement and Offset Compensation,” *Proc. of Conf. on Information Sciences and Systems (CISS’2001),* Baltimore, MD 2001.