Charge-Mode Parallel Architecture for Matrix-Vector Multiplication

Roman Genov and Gert Cauwenberghs
Department of Electrical and Computer Engineering
The Johns Hopkins University, Baltimore, MD 21218-2686
E-mail: {roman,gert}@bach.ece.jhu.edu

***Abstract*— An internally analog, externally digital architecture for matrix-vector multiplication is presented. Fully parallel processing allows for high data throughput and minimal latency. The analog architecture incorporates an array of charge-mode analog computational cells with dynamic storage and row-parallel flash analog-to-digital converters (ADC). Each of the cells includes a dynamic storage element and a charge injection device computing binary inner product of two arguments. The matrix elements are stored in the array of computational cells in bit-parallel fashion, and the input vector is presented bit-serially. Digital post-processing is then performed on the ADC outputs to construct the resulting product with precision higher than that of each conversion. The analog architecture is tailored for high-density and low power VLSI implementation, and matrix dimensions of 128 × 512 and ADC resolution of 6 bits for an overall resolution in excess of 8 bits are feasible on a 3 mm × 3 mm chip in standard CMOS 0.5 μm technology.**

I. Introduction

Real-time applications in multimedia and sensory signal processing place extreme demands on computing power, in excess of even the most powerful processors available today. At the core of many algorithms for signal processing, and pattern recognition in particular, is the need for matrix-vector multiplication in very large dimensions. Implementations in software or on general-purpose DSPs lack the massive parallelism and memory bandwidth needed for efficient and real-time implementation. Networks of parallel computers could be capable of very high peak throughput rates, but are very costly, and impractical for embedded real-time applications. Currently existing parallel computational systems are limited in the degree of parallelism and still make use of centralized memory resources (*i.e.*, dedicated off-chip DRAM). More desirable would be a fine-grain, fully-parallel architecture in which each processor performs a multiply and locally stores one coefficient. The recurring problem with such an implementation is the latency in accumulating the result over a large number of cells. To this end, we propose the use of hybrid analog-digital technology to efficiently add a large number of digital values simultaneously in parallel, with careful consideration of sources of imprecision in the implementation and their overall effect on the system performance.

Among elementary operations in signal processing and pattern recognition algorithms that call for a parallel implementation, matrix-vector multiplication is one of the most common, but also one of the computationally most expensive:

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

with N-dimensional input vector $X^{(n)}$, M-dimensional output vector $Y^{(m)}$, and matrix elements $W^{(n,m)}$, for $n = 0, 1, ..., N-1$ and $m = 0, 1, ..., M-1$. In neural networks, the matrix elements correspond to weights, or synapses, between neurons. In signal processing, the elements could represent, for instance, coefficients in a filter, or templates $Y_n^{(m)} = W^{(m,n)}$ in a vector quantizer. Specialized architectures have been developed for specific needs.

Analog VLSI solutions offer significant benefits in computational density and energy efficiency [1], [2], [3]. Analog multiplier circuits in this case can be 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 including a product computing device and a storage element. Each cell (m, n) computes a product of an input component $X^{(n)}$ and a matrix element $W^{(m,n)}$ with the horizontal node being the output summing line. The storage device can usually be effectively incorporated into a cell to avoid performance limitations due to low external memory access bandwidth. Tailored to particular applications, systems with various input and matrix elements representations have been explored [1].

Besides other performance considerations, the type of input and output interface, and the ubiquitous availability of low-power and parallel A/D or D/A data conversion arrays, often justify appropriate input and matrix element encoding: *analog*—common for direct interactions with external analog world [4], or *digital*—for chips incorporated in digital systems [5]. In addition, analog implementations can be customized to operate in one of several computing modes, *e.g.*charge [6], [7], conductance [1], or current [8], to fit requirements on the signals they operate on, such as dynamic range, synchronicity (continuous-time, asynchronous, or clocked), and the computation that needs to be performed at the periphery. The architecture presented here is suited for a synchronous digital interface, and emphasizes computational density, energy efficiency and large-scale implementation. In choosing an effective solution, a charge-mode approach with binary encoded bit-serial inputs and bit-parallel matrix elements, was investigated.

This work was supported by ONR YIP (N000149910612) and NSF Career (MIP-9702346).

Proc. 43rd IEEE Midwest Symp. on Circuits and Systems, Lansing MI, Aug 8-11, 2000
0-7803-6475-9/00/$10.00 ©IEEE 2000

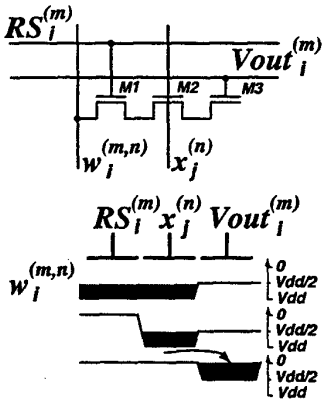

Fig. 1. *Computational cell with dynamic storage (top) and charge transfer diagram for the all-one product case (bottom).*

II. Mixed-Signal Architecture

A. Internally Analog, Externally Digital Computation

The system presented is a bit unconventional in that it 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 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 architecture. Inputs are presented in bit-serial fashion, and matrix elements are stored locally in bit-parallel form. D/A conversion at the input interface is inherent in the bit-serial implementation, and row-parallel flash 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:

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

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

The equation (1) can be rewritten as:

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

where

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

B. Analog Implementation

The multiplier cell constituting the basic element of the processor array computes one argument of the sum in the equation (5). Its diagram is presented in Figure 1. The cell performs multiplication of two 1-bit (unsigned) binary numbers: $w_i^{(m,n)}$ and $x_j^{(n)}$, for one particular i and j. A logic level 0 is represented as Vdd, and a logic level 1 is represented as $Vdd/2$ and 0 Volts for $w_i^{(m,n)}$ and $x_j^{(n)}$ respectively, where Vdd is the supply voltage. The cell contains three serially connected MOS transistors as shown at the top of the figure. Transistor M1 is a switch controlled by the *Row Select* signal, $RS^{(m)}$. When activated, it allows to write a binary number corresponding to $w_i^{(m,n)}$ in the form of charge stored under the gate of M2. Transistors M2 and M3 comprise a charge injection device (CID), which by virtue of charge conservation moves electric charge between two potential wells in a non-destructive manner [7], [6], [13].

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 $Vdd/2$. When a "write" is performed, depending on the value of $w_i^{(m,n)}$, a fixed amount of electric charge is either stored under the gate of M2, when $w_i^{(m,n)}$ is low, or removed from under the M2 gate, when $w_i^{(m,n)}$ is high. The amount of charge being moved corresponds the value of $w_i^{(m,n)}$ and ideally is ΔQ or 0 respectively. The bottom diagram in Figure 1 depicts the charge transfer diagram for the case when both $w_i^{(m,n)}$ and $x_j^{(n)}$ are of logic level 1.

Once the charge has been stored, the switch M1 turns off. Then the cell is ready to compute. The charge under the gate of M2 can only be redistributed between the two CID transistors. If the gate of M3 is now left floating at the pre-charged level of $Vdd/2$, the voltage $Vout$ will ideally change only when both charge ΔQ was stored under the gate of M2 and, in the computation phase, $x_j^{(n)}$ of 0 volts was applied. This produces a binary output which is a boolean AND function of the input vector component and the matrix element component. When both arguments are 1, the voltage $Vout$ will change by:

$$\Delta V_{out} = \Delta Q / C_{M3} \quad (6)$$

The computational cells are arranged in an array with each row containing a summing output node to produce $Y_{i,j}^{(m)}$ in the equation (5). Transistor-level simulation of 512-element row shows nonlinearity of 1 LSB level with a dynamic range of 43dB as shown in Figure 2. Computation speed is of the order of 10μSec for the whole array. With potentially millions of cells operating in parallel, this gives a computational throughput exceeding by far the fastest processors available to date. At this rate, real time implementation of demanding computational tasks in multimedia and pattern recognition is practical.

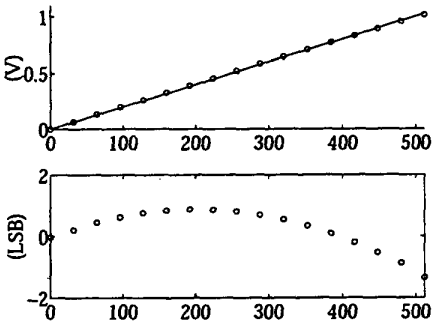

Fig. 2. *Simulated linearly-fitted transfer characteristic (top) and differential nonlinearity (bottom) for 8-bit resolution for a 512-cell row of the computational array, using* SpectreS *and transistor models extracted from a 0.5 μm CMOS process.*

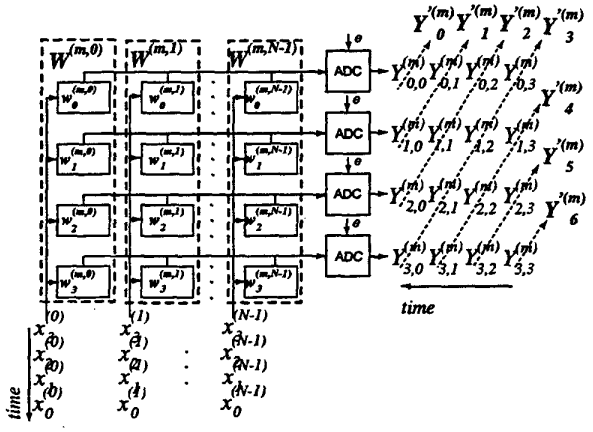

Fig. 3. *Block diagram for one output component m in the parallel architecture, with* $I = J = 4$.

C. Data Flow Organization and Digital Postprocessing

Although analog inside, externally the system is fully digital. For convenience of implementation, matrix elements are stored in bit-parallel, and inputs are presented in bit-serial fashion, with least-significant bit (LSB) first. One row of I-bit encoded matrix elements uses I rows of the computational array. Therefore, to store an $M \times N$ matrix, a computational array of $MI \times N$ capacity is needed. One bit of an input vector is fed in for each clock cycle yielding a full computational cycle length of J clock cycles.

Figure 3 presents the architecture of a set of rows in common for one output component m, with I-bit matrix elements and a J-bit input vector. The analog outputs at each of the horizontal summing nodes are converted to a digital vector with row-parallel ADCs. Over J clock cycles a $I \times J$ matrix is produced. Its elements are $Y_{i,j}^{(m)}$ as defined by the equation (5). Diagonals of this matrix marked in the figure with dashed lines contain elements of the same binary weight. Digital summation of these elements along the diagonals produces $K = I + J - 1$ bits of the resulting inner product value, as can be seen by rearranging equation (4):

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

where $k = i + j$ and

$$Y_k'^{(m)} = \sum_{i=\frac{sign(k-I)+1}{2}(k-I+1)}^{k+\frac{sign(k-I)+1}{2}(I-k-1)} Y_{i,k-i}^{(m)}. \quad (8)$$

D. Performance Analysis

To asses the overall system performance (*i.e.*, precision in effective number of bits), let us assume that each of the row parallel A/D converters quantizes the value on the horizontal summing node in L bits with noise, e, of less than 1 LSB level. We first consider the case when $L < log_2(N)$, and assume that e is random and uniform.

For a full signal range s of $Y_{i,j}^{(m)}, \forall i, j$, and large I, the signal range of $Y^{(m)}$ can be approximated as:

$$S \approx \frac{1}{4} s \sum_{k=0}^{K-2} \frac{k+1}{2^k} \approx s \quad (9)$$

Let the variance of the uniform quantization noise e introduced to $Y_{i,j}^{(m)}, \forall i, j$ by an ADC be q^2. This noise component is relative to the full ADC scale, and scales linearly in amplitude with the signal when the output is binary weighted, and thus quadratically in variance. The cumulative quantization error in the summation in (8) can be roughly approximated by Central Limit Theorem for additive variances. For large I and J, each sum over k can be considered a normal process with cumulative variance of each of the arguments. Each signal component with binary weight $2^{-(i+j)}$ thus contributes a variance $2^{-2(i+j)}$ to the noise, and the total variance for large I and J can be expressed as:

$$Q^2 \approx \frac{1}{4^2} q^2 \sum_{k=0}^{K-2} \frac{k+1}{2^k} \approx \frac{1}{4^2} q^2 (\frac{4}{3})^2 = (\frac{1}{3})^2 q^2 \quad (10)$$

Therefore, the standard deviation of the total noise in $Y^{(m)}$ is $1/3q$. The effective overall $Y^{(m)}$ noise level is:

$$\frac{Q}{S} \approx \frac{1}{3}\frac{q}{s} \quad (11)$$

For a normal process, to be contained within 1 LSB on average, the noise level can be estimated as:

$$Q_{ave} = 0.69Q = 0.23q \quad (12)$$

and the average effective signal-to-noise ratio as:

$$\frac{S}{Q_{ave}} = \frac{s}{0.23q} \approx 4.3\frac{s}{q} \quad (13)$$

Therefore, for the case when $L < log_2(N)$, the overall system precision is below the ideal $(I + J + log_2(N))$ and is limited

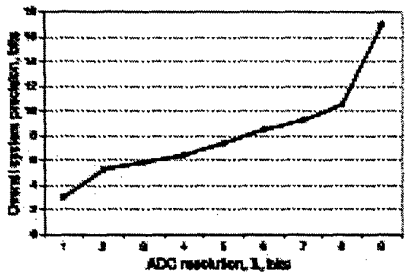

Fig. 4. *Simulated average overall system precision for different A/D converter resolutions L, and for* $I = J = 4$, $N = 512$.

mainly by the precision of the A/D converter. However, by digitally combining outputs from bit slices as described, the overall precision is better than that of each ADC result in the sum. With the lower bound as defined by (11), on average this digital architecture boosts the system precision by 2 bits above L bits. This is not surprising since improved precision at the macro level is generally an attribute of large scale analog computation [9]–[11].

In the case when $L = log_2(N)$, no information is lost in the ADC quantization step, and the effective system resolution is $I + J + log_2(N)$ bits, which can be arbitrarily larger than L. The key here is that the analog array performs in essence a digital (or, more precisely, *quantized*) computation: adding binary numbers in parallel, one per column. As long as the precision of the ADC resolves each individual column in the sum, and the combined effect of noise in the analog summation and the ADC is within one LSB, then the binary addition is retrieved from the ADC with zero error. Fortunately, the precision requirements on the analog MOS array and ADC increase only with the *logarithm* of the number of columns in the parallel analog sum, and a large number of columns (several hundreds) is practical. Computations with higher dimensionality, and still at maximum possible precision, can easily be achieved by cascading multiple chips. To extend matrix column space, systems with shorter word lengths can be effectively combined to perform a computation in a wider space by using, for example, modulo arithmetics [12]. Extension of row space is in principle also unlimited. The digital outputs from A/D converters on multiple chips can be multiplexed on a same bus, read out and processed by a digital postprocessor.

Results of Monte Carlo simulations validating the precision of the architecture are presented in Figure 4. The bit components of inputs $x_j^{(n)}$ and matrix elements $w_i^{(m,n)}$ were generated as $i.i.d.$ Bernoulli random variables. Inner products of the input vector and each of the rows of the binary array of matrix elements were overlayed with uniform quantization noise over the interval $(-1/2^{L+1}, 1/2^{L+1})$. The values then were quantized to L bits and combined according to the above algorithm. The resulting average computational precision was recorded for different values of ADC resolution. As predicted, when $L < log_2(N)$ the system precision was on average over 2 bits better than individual A/D converter resolution. For $I = J = 4$, $N = 512$ and an ADC resolution of $L = 9$ bits, an ideal precision of 17 bits is obtained.

III. Conclusions

An internally analog, externally digital architecture for matrix-vector multiplication has been presented, tailored for a high-density and low-power VLSI implementations. Fine-grain and massively parallel with distributed memory, the architecture easily provides real-time operation making it suitable for a wide range of computation-intensive tasks in multimedia signal processing and pattern recognition. The architecture embeds storage and multiplication in distributed fashion, down at a 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.

The combination of internally analog and externally digital processing also provides for an overall accuracy that exceeds the intrinsic precision (of the analog array and parallel ADC) by at least 2 bits. Virtually unlimited precision can be obtained by carefully configuring the array 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.

References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] G. Cauwenberghs, C. Neugebauer, A. Yariv, "An adaptive CMOS matrix-vector multiplier for large scale analog hardware neural network applications," Proc. IEEE Int. Joint Conference on Neural Networks (IJCNN'91), Seattle, WA, vol. 1, pp 507-511, 1991.

[9] C. Mead, *Analog VLSI and Neural Systems*, Addison-Wesley, 1989.

[10] E. Vittoz, "Micropower Techniques," in *Design of Analog-Digital VLSI Circuits for Telecommunications and Signal Processing*, Franca and Tsividis, Eds., Prentice Hall, 2nd. ed., pp 53-96, 1994.

[11] M. Ismail and T. Fiez, Eds., *Analog VLSI for Signal and Information Processing*, McGraw-Hill, 1995.

[12] B. Arambepola, *Common VLSI Architecture for a Practically Useful Residue Number System*, Proc. of IEEE Int. Symp. on Circuits and Systems (ISCAS'91), vol. 5, pp 2951 -2954, 1991.

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

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