Acknowledgment
The authors are grateful to the anonymous reviewers for their helpful comments and suggestions.

References

Fully Parallel Stochastic Computation Architecture
C. L. Janer, J. M. Quero, J. G. Ortega, and L. G. Franquelo

Abstract—A space-efficient fully parallel stochastic computation architecture is described in this correspondence. It circumvents the main drawbacks of stochastic computation architectures that have been used up to now: the absence of a space-efficient technique of adding weighted input signals in parallel.

I. INTRODUCTION
Stochastic logic systems make pseudonanalog operations using stochastically coded-pulse sequences [1], [2]. In stochastic systems, the terms that are to be processed are synchronous pulse sequences [3]-[4]. These pulses randomly take, at each clock cycle, either a “high” or “low” level. Information is codified as the probability of taking the “high” level. The average of a stochastic signal can be regarded as an analog value in the range [0,1] that is equal to this probability. In order to interface stochastic systems with analog signals, special circuits must be designed [5].

Stochastic computation exhibits some features that make it very appealing. Pseudonanalog computations are easily performed by very space-efficient digital logic and, at the same time, stochastic signals are much less sensitive to noise than analog signals.

In stochastic computing, products are calculated in a very easy way, whereas additions are difficult to evaluate. In order to overcome this problem, we propose a new fully parallel addition stochastic architecture based on the idea that additions can be transformed into products.

II. STOCHASTIC COMPUTATION
Fig. 1 shows how stochastic signals can be generated. During each clock cycle, a binary digital value stored in a register $R$ is compared with a uniformly distributed random number $X$. The output of the comparator pulses if $X < R$, creating a stochastic signal whose firing probability is proportional to $R$.

The firing probability of a stochastic signal can be estimated by accumulating the pulses. The accumulated number has a mean value equal to the firing probability and a variance, which can be regarded as noise, that decreases with time [2]. A natural way to estimate precision versus accumulating time is to consider the fraction $\sigma / \mu$, where $\mu$ stands for the mean value of the accumulated number, and $\sigma$ stands for its standard deviation. It can be easily shown [8] that this fraction takes the value $\sigma / \mu = \sqrt{1 - x/Nx}$, where $x$ represents the mean value of the pulse stream and $N$ the number of clock cycles.

Multiplications can be carried out in stochastic logic in a very simple way. Consider now a set of $n$ pulse streams whose probabilities, at a given clock cycle, of being at a “high” level are $p_1, p_2, \cdots, p_n$. These probabilities are mutually independent. If these sequences are the inputs of an $n$-input AND gate, the probability of the gate output, at a given clock cycle, of being at a “high” level is equal to $p_1 \cdots p_n$.

Stochastic summation is a much more difficult operation to perform. This stems from the fact that the addition of $n$ numbers that take values ranging from 0 to 1 may take values bigger than 1 and, therefore, cannot be represented by a probability.

The up-down counter technique is widely used in some applications like neural network implementation [6]. Pulses coming from other neurons are multiplexed in time (i.e., serialized) and either increase or decrease an up-down counter, depending on its sign. The addition can be represented stochastically because the up-down counter cannot overflow. If the addition is bigger than 1 or smaller than -1, the counter is clamped to either its maximum or minimum value, providing saturation. This nonlinear behavior is shown in Fig. 2.

If two pulse sequences are inputs of an OR gate, the output firing probability can be easily evaluated:

$$Pr(\xi_1 = 1) \cup (\xi_2 = 1) = Pr(\xi_1 = 1) + Pr(\xi_2 = 1) - Pr(\xi_1 = 1) \cap (\xi_2 = 1) = p_1 + p_2 - p_1 p_2.$$ (1)

The output firing probability is not the addition of the input probabilities due to pulse overlap. In [7], the effects of pulse overlap are analyzed. From this analysis, it follows that input pulse densities should be small if an accurate addition is required, especially if many
terms are to be added. The maximum input pulse densities strongly decreases as the number of terms increases.

III. STOCHASTIC ADDITION ARCHITECTURE

A. Exponential Transformation

It has been shown that in order to represent additions by stochastic streams, some sort of compression has to be performed. The up-down counter technique naturally exhibits a nonlinear behavior that circumvents this problem.

We propose a different compression function (exponential) that permits evaluation in parallel of the addition using very simple digital circuits. This exponentiating method transforms the interval \([0, \infty)\) into the interval \((0,1]\), as is shown in Fig. 3. The reason why we choose this transformation is that it allows us to convert additions into multiplications that are trivial in the stochastic domain. Taking into account the properties of the exponential compression function, it follows that

\[
\prod_{i=1}^{n} e^{-x_i} = \exp \left( \sum_{i=1}^{n} -x_i \right)
\]

where all terms \(x_i\) are supposed to be positive. This means that if the exponential transformation can be evaluated stochastically, the exponentially transformed streams of pulses can be multiplied by means of an AND gate, producing at the output of this gate another stream that represents the exponentially compressed addition of all inputs; see Fig. 4.

The exponentiation can be stochastically computed in a very simple fashion if only the first few terms of its Taylor’s expansion are considered:

\[
\begin{align*}
\exp(x) &\approx 1 + x - \frac{x^2}{2} \\
\exp(-x) &\approx 1 - x + \frac{x^2}{2} - \frac{x^3}{6}
\end{align*}
\]

The exponential transformations are not evaluated exactly. This leads to addition errors that are bounded in Appendix A. In this appendix, it is proved that to a first approximation, the accuracy of the addition does not depend on the number of terms that are being added. Digital circuits that evaluate the first-, second-, and third-order transformation are shown in Fig. 5. Flip-flops are used to delay the sequences in order to permit the stochastic multiplication of a pulse stream by itself.

Making use of De Morgan’s theorem, the first-order transformation leads to the OR gate addition structure. However we do not interpret the output of the OR gate as the addition of the input pulses as in [7]. It represents their exponentially compressed addition. The input numbers must be kept below a certain quantity simply to ensure that the linear approximation of the exponential function is accurate enough. We have used the linear approximation in artificial neural network implementation [8]–[9].

Positive and negative numbers can be exponentially transformed separately and then multiplied. This produces two stochastic streams, representing the compressed addition of positive and negative terms. These two terms must be combined in order to obtain a stochastic stream that represents the addition of all input numbers.

B. Adding Positive and Negative Terms

In the previous section, we have seen that a stochastic stream representing the addition of \(n\) equally signed terms can be carried out

Fig. 1. Synchronous stochastic signal generation.

Fig. 2. Piece-wise linear compression function.

Fig. 3. Exponential compression function.

Fig. 4. Exponentially compressed multiplication.
by simple digital circuits. The circuit that combines the two stochastic signals representing, respectively, the compressed addition of positive and negative inputs is represented in Fig. 6.

One of the states of the flip-flop drives the "up" signal of the up-down counter. The other state drives the "down" signal. In Appendix B, we prove that the average net number of times that the up-down counter is increased is given by

$$1 - \exp\left( - \left( \sum_{i} x_i^+ - \sum_{i} x_i^- \right) \right)$$

If the addition does not take values close to either 1 or -1, this function can be approximated by $(\Sigma_i x_i^+ - \Sigma_i x_i^-)/2$.

This expression is no more than the linear approximation of (3). The amount of nonlinearity can be bounded by limiting the maximum input pulse densities. The whole addition structure is represented in Fig. 7. Input stochastic streams are exponentially compressed.

The transformed streams are driven to either the positive or the negative AND gate by a demultiplexer, depending on their sign. The demultiplexer's output that is not driving the stochastic stream stays at "high" level in order not to inhibit the other AND gate. The outputs of the two AND gates are the inputs of a logic block. This circuit checks if a positive pulse is present and a negative one is not. Whenever this happens, the output flip-flop is reset. It also checks if a negative pulse is present and a positive one is not. Whenever this happens, the output flip-flop is set. If a digital representation of the output stream is needed, the flip-flop's state can drive the up-down signal of a counter.

C. Comparison with Other Techniques

The up-down counter structure is very simple and has been widely used in neural network implementation. It uses an intermediate digital representation of the addition that is easily converted into a stochastic stream of pulses. The stochastic addition saturating response is naturally achieved in this technique because the counter is not allowed to overflow. The up-down counter technique, due to its time-multiplexed nature, does not permit fully parallel computation.

The output stochastic signal of any stochastic computing process must be accumulated in order to calculate its mean value. The combination of accumulation delays and serialization may lead to slow computation times.

In our approach, the exponential compression permits us to convert additions into products resulting in a fully parallel summation structure that is very efficient in terms of consumed area, and no intermediate representation of the summation is needed.

The OR-based add function is based on the assertion that performing a logical OR of $N$ input independent stochastic signals produces another stochastic signal whose firing probability is equal to the
addition of the $N$ input firing probabilities. However, the addition is
distorted by pulse overlap. In order to add many signals accurately,
pulse densities must be kept very low in this technique, limiting the
speed of calculations.

In our approach, inputs with high pulse densities may be added
accurately, and the number of input signals does not limit their
maximum input densities.

IV. A CASE STUDY: FINITE IMPULSE RESPONSE FILTER IMPLEMENTATION

The stochastic architecture that we propose is suitable for the
implementation of finite impulse response (FIR) filters. In Fig. 8,
coefficients are loaded in registers $c_1, c_2, \ldots, c_n$ in order to program
the filter. An input signal is sampled and stored in register $i_1$.

 Registers $i_1, i_2, \ldots, i_n$ are shifted in order to update the $n-1$ delayed
inputs. The digital representations of the inputs and their coefficients
are converted to stochastic pulse sequences. These two sets of streams
are multiplied by means of AND gates, producing another set of input
streams that are to be added. The pulse sequences are exponentially
compressed and separated, depending on their sign. These two groups
are multiplied separately. The output of these two gates are the set
and reset signals of a flip-flop that combines the two streams in order
to evaluate the addition. A digital representation of the addition can
be obtained using an up-down counter.

A programmable logic device—a Texas Instruments TPC 1280
FPGA—has been designed using Verilog-XL, which is a behavioral
language within the CADENCE DESIGN FRAMEWORK II. This
circuit permits us to implement FIR filters up to a number of
delayed inputs equal to 40. If a high resolution of coefficients and inputs is needed, a higher order of the Taylor’s expansion should be used. However, the higher the order of the approximation, the more complex the circuits become. We have found a good compromise, taking a second-order approximation. The registers’ size is 9 b. One bit is used to represent the sign, and the other 8 b are used to represent the value.

We have programmed the 40 coefficient registers so that a 12th-order Yule–Walker bandpass filter is approximated by recursive substitution. The normalized low and high frequencies are 0.1 and 0.25. The switching frequency is 10 kHz. The operating frequency of the stochastic system is 10 MHz. The coefficients of the filter before truncating them into the 8-b internal representation are shown in Table I.

The operation of the filter is quite simple: The input is sampled and shifted into the input registers, then the filter evolves during 1000 clock cycles, the digital codification of the output is written in an external register, and then, the process starts all over again.

Figs. 9 and 10 show that the output of the circuit and the simulation of the output of a floating-point multiplier-based digital filter are very similar.

V. CONCLUSIONS AND FURTHER WORK

We have presented a fully parallel stochastic computing architecture that is useful to solve problems where a large number of products and additions are to be computed because these operations are all carried out simultaneously by very simple digital circuits (a
few AND, OR, NOT gates and flip-flops). The simplicity of the circuits that evaluate these operations leads to a high efficiency in terms of consumed area and small delays.

**APPENDIX A**

**ERROR BOUNDING IN THE ADDITION OF EQUALLY SIGNED INPUTS**

Due to the fact that the exponential transformation is not evaluated exactly, we conclude that the addition of the $n$ equally signed input terms is distorted. In order to evaluate this distortion, let us consider the following expressions:

$$x_1 = e^{r_1 + \xi_1}$$
$$x_2 = e^{r_2 + \xi_2}$$
$$\rightarrow \quad e^{x_1 + \xi_1} = \prod_{i=1}^{n}(e^{x_i} + \xi_i)$$
$$x_n = e^{x_n + \xi_n}.$$

The first column represents the probabilities of the input terms of being at a high level. The first arrow takes after the exponential transformations. Due to the fact that these transformations are not evaluated exactly, we have taken into account the errors, which are represented by the $\xi_i$ terms. The second arrow represents the product of the transformed terms. If we develop the last expression, we find

$$\prod_{i=1}^{n}(e^{x_i} + \xi_i) \approx \prod_{i=1}^{n}e^{x_i} + \sum_{i=1}^{n} \xi_i \prod_{j \neq i}^{n} e^{x_j}$$

$$= \prod_{i=1}^{n}e^{x_i} \left(1 + \sum_{j=1}^{n} \xi_j\right)$$

(4)

where $\hat{\xi}_i := \xi_i/x_i$. If we evaluate the logarithm of this expression and define $\bar{\xi}_i := \hat{\xi}_i/x_i$, it follows that

$$\sum_{i=1}^{n} x_i + \log \left(1 + \sum_{j=1}^{n} x_j \bar{\xi}_j\right)$$

(5)

that is

$$\text{err} \leq \log \left(1 + \sum_{j=1}^{n} x_j \bar{\xi}_m\right) \leq \sum_{j=1}^{n} x_j \bar{\xi}_m$$

where $\bar{\xi}_m$ is the maximum $\bar{\xi}_i$. We deduce that the rate error verifies

$$\text{err} \leq \frac{\text{err}}{\sum_{i=1}^{n} x_i} \leq \bar{\xi}_m.$$

(6)

The maximum rate error boundary does not depend on the number of terms that are being added. In fact, this statement is not exactly true. If second-order terms become important in the evaluation of the product of the transformed numbers, then the approximation that has been done in (4) is no longer true. The number of first-order terms is $n$, and the number of second-order terms is $C_n^2$. Therefore, second-order terms cannot be neglected if $C_n^2 \xi^2/n \xi \approx n \xi^2/2 \leq 1$.

The quantities $\xi_m$ and $\bar{\xi}_m$ are plotted for the first-, second-, and third-order approximation of the exponential transformation in Figs. 11 and 12.

**APPENDIX B**

**STATISTICAL ANALYSIS OF THE ADDITION CIRCUIT**

Consider Fig. 6. The two stochastic streams of pulses can be easily used to evaluate the addition of the input terms. The output stream is a two-state random variable. The two states will be denoted "positive"
The transition matrix of the process is

\[ \begin{bmatrix} \begin{array}{cc} p_1q_2 & p_2q_1 \\ p_1q_2 & p_2q_1 \end{array} \end{bmatrix} \]

The eigenvalues of this matrix are 1 and \(1 - p_1q_2 - p_2q_1 < 1\). Consequently, the Markovian process is ergodic, that is, it makes sense to talk of an average number times that the process is in each state. The steady-state vector is

\[ \hat{\mathbf{x}} = \begin{bmatrix} \frac{p_1q_2}{p_1q_2 + p_2q_1} \\ \frac{p_2q_1}{p_1q_2 + p_2q_1} \end{bmatrix}. \]

This means that in the long run, the probability of the counter being increased is equal to \(p_1q_2/(p_1q_2 + p_2q_1)\), and the probability of the counter being decreased is equal to \(p_2q_1/(p_1q_2 + p_2q_1)\). If we substract the second term from the first one, we obtain

\[ 1 - \exp \left( -\sum_{i=1}^{\infty} x_i \right) = \frac{1}{2}. \]

In order to deduce this expression, we have assumed that \(2p_1p_2 \ll p_1 + p_2\). This restriction can be easily fulfilled, provided that we add

\[ \begin{array}{cc} \text{inc} & \text{dec} \\ \text{inc} & 1 - p_2q_1 & p_2q_1 \\ \text{dec} & p_1q_2 & 1 - p_1q_2 \end{array} \]
two equally valued inputs of opposite signs so that $p_1$ and $p_2$ become sufficiently small.

ACKNOWLEDGMENT

The authors wish to thank J. Armario for helping them with the filter implementation. They would also like to thank the reviewers for their constructive comments and suggestions, which have greatly improved the quality of this paper.

REFERENCES


A Systolic Array for Recursive Least Squares Computations: Mapping Directionally Weighted RLS on an SVD Updating Array

Marc Moonen

Abstract—A systolic algorithm/array is described for recursive least squares (RLS) estimation, which achieves an $O(n^3)$ throughput rate with $O(n^2)$ parallelism. The array is also useful for several other applications, such as, e.g., SVD updating and Kalman filtering. An additional advantage is that unlike with other RLS-arrays, it is now possible to incorporate alternative data weighting strategies, such as directional weighting, without sacrificing speed.

Manuscript received March 23, 1993; revised December 21, 1995. This work was supported, in part, by ESPRIT Basic Research Action 6632. The associate editor coordinating the review of this paper and approving it for publication was Dr. W. Wojtek Przytula.

The author is with ESAT—Katholieke Universiteit Leuven, K. Mercierlaan 94 3001, Heverlee, Belgium (e-mail: marc.moonen@esat.kuleuven.ac.be). He is a research associate with the Belgian National Fund for Scientific Research (N.F.W.O.).

Publisher Item Identifier S 1053-587X(96)03276-2.