INTERNATIONAL JOURNAL OF CIRCUIT THEORY AND APPLICATIONS
Int. J. Circ. Theor. Appl. 2005; **33**:307–319
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cta.322

Synthesis of multiport resistors with piecewise-linear characteristics: a mixed-signal architecture

Mauro Parodi[1], Marco Storace[1,*,†] and Pedro Julián[2,3]

[1] *Biophysical and Electronic Engineering Department, University of Genoa, Via Opera Pia 11a, I-16145 Genova, Italy*
[2] *Universidad Nacional del Sur, Av. Alem 1353, Bahía Blanca, Argentina*
[3] *Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), 1033 Cap. Federal, Argentina*

SUMMARY

Non-linear multiport resistors are the main ingredients in the synthesis of non-linear circuits. Recently, a particular PWL representation has been proposed as a generic design platform (*IEEE Trans. Circuits Syst.-I* 2002; **49**:1138–1149). In this paper, we present a mixed-signal circuit architecture, based on standard modules, that allows the electronic integration of non-linear multiport resistors using the mentioned PWL structure. The proposed architecture is fully programmable so that the unit can implement any user-defined non-linearity. Moreover, it is modular: an increment in the number of input variables can be accommodated through the addition of an equal number of input modules. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: piecewise-linear approximations; non-linear circuit synthesis; mixed-signal circuits

1. INTRODUCTION

In recent papers [1–3] the PWL approximation technique proposed by Julián *et al.* [4, 5] has been applied to the approximate synthesis of non-linear circuits. The basic idea of the synthesis methodology is to approximate each constitutive equation $y = f(\boldsymbol{x})$ of a non-linear resistive element of a circuit (where y is a generic dependent descriptive variable and $\boldsymbol{x}$ is the vector of independent descriptive variables) using a canonical PWL representation. It is well known that non-linear multiport resistors play a fundamental role in non-linear network synthesis [6].

*Correspondence to: Marco Storace, Department of Biophysical and Electronic Engineering, University of Genoa, Via Opera Pia 11a, I-16145 Genova, Italy.
†E-mail: marco.storace@unige.it

Contract/grant sponsor: MIUR
Contract/grant sponsor: University of Genoa
Contract/grant sponsor: Fundación Antorchas; contract/grant number: 14169-12

Received 21 July 2004
Revised 12 February 2005

On the other hand, canonical representations describe a function with the minimum number of parameters. This efficiency is fundamental for the approximation of large systems, or systems of moderate size where high accuracy is needed. In particular, the canonical representation proposed in Reference [4] can be developed systematically using samples of the system to be approximated, and—under very mild conditions—permits to obtain arbitrary accuracy, by adjusting the grid that subdivides the domain. In this formulation, the PWL function, namely f_{PWL}, is obtained as a linear combination of a set of N basis functions over an n-dimensional compact domain $S \subset \mathbb{R}^n$, i.e. $f_{\mathrm{PWL}} : S \to \mathbb{R}^n$, where S is a hyperrectangle (rectangle, if $n=2$) in the form of

$$S = \{\boldsymbol{x} \in \mathbb{R}^n\colon\ a_i \leqslant x_i \leqslant b_i,\ i = 1, \dots, n\} \tag{1}$$

Every dimensional component x_i of the domain S is subdivided into m_i subintervals of length $(b_i - a_i)/m_i$, producing a boundary configuration H [4]. The resulting hyperrectangles contain $n!$ non-overlapping hypertriangular (triangular, if $n=2$) regions called simplices. As a result, S is partitioned (*simplicial partition*) into $\prod_{i=1}^{n} m_i$ hyperrectangles and contains $N = \prod_{i=1}^{n}(m_i + 1)$ vertices $\boldsymbol{v}_i$. A characteristic of this subdivision is that the domain and its boundary configuration H can be completely described by the triplets $(a_i,\ b_i,\ m_i)$, $i = 1, \dots, n$.

As shown in References [4, 5], the class of continuous PWL functions f_{PWL} defined over a fixed boundary configuration of the domain constitutes an N-dimensional Hilbert space PWL$[S_H]$, which is defined by the domain S, its simplicial partition H, and a proper inner product (see Reference [2] for details). Each function belonging to PWL$[S_H]$ can be represented as a sum of N basis functions (arbitrarily organized into a vector), weighted by an N-length coefficient vector $\boldsymbol{c}$. The coefficient vector $\boldsymbol{c}$ determines the shape of f_{PWL} uniquely. As shown in Reference [1], such a vector can be systematically found by applying optimization techniques (e.g. a least-squares criterion) to a set of properly distributed samples of f over the domain S.

There are many possible choices for the PWL basis functions, each of which is made up of N (linearly independent) functions belonging to PWL$[S_H]$. Some bases are more convenient for function interpolation and some others for function approximation. In addition, from an implementation viewpoint, some bases are computationally attractive whereas others are attractive because they lead to convenient circuit topologies. In particular, for the circuit synthesis we shall refer to the so-called $\boldsymbol{\beta}$-basis and $\boldsymbol{\alpha}$-basis [1, 2].

The $\boldsymbol{\beta}$-basis seems to be more suitable for completely analog implementations, via architectures commonly used in CMOS analog design, in particular current-mode and sub-threshold techniques (see References [7–10]). Indeed, if a current mode configuration is used, then the outputs of the different circuit blocks implementing the basis functions can be weighted and added over a single node providing a straightforward architecture. The drawback, however, is that in the β formulation, different functions, implementing different nestings of absolute values, are needed. In addition, functions with the same number of nestings require different bias voltages or currents. This lack of uniformity is not convenient if large arrays of PWL basis functions are to be integrated.

The $\boldsymbol{\alpha}$-basis, on the contrary, is particularly well suited for a synthesis based on a mixed analog/digital architecture. The $\boldsymbol{\alpha}$-basis is composed of N PWL functions $\alpha_1(\boldsymbol{x}), \dots, \alpha_N(\boldsymbol{x})$

satisfying (for $k, j = 1, \ldots, N$)

$$\alpha_k(\boldsymbol{v}_j) = \begin{cases} 1 & \text{for } k = j \\ 0 & \text{for } k \neq j \end{cases} \tag{2}$$

When regarded as a multidimensional surface, each function $\alpha_k(\boldsymbol{x})$ can be seen as a hyperpyramid with value 1 at the vertex $\boldsymbol{v}_k$ and value 0 at all other vertices. In other words, each α-function is of a local nature, since it is different from 0 only over a reduced number of simplices of the domain. As a consequence, the value of the approximate function $f_{\mathrm{PWL}}(\boldsymbol{x})$ can be obtained, for any $\boldsymbol{x}$, by combining a limited subset of the basis functions weighted by the corresponding coefficients. This architecture is much more convenient to be integrated in an electronic fashion. The reason is that all basis functions perform basically the same operation. The difference between two basis functions is that they operate over two different regions of the domain. Therefore, the evaluation can be done using only one function circuit block and an algorithm to shift the inputs. For every evaluation point, all non-zero basis functions need to be evaluated, weighted and added, but they are at most $n+1$ (that is, linear with respect to the domain dimension), so that the implementation is still remarkably efficient. This principle has been exploited in References [11, 12] to derive an electronic implementation for an image processing CNN based on PWL coupled elements. In this paper, we generalize the solution proposed in Reference [11] in the framework of PWL multiport resistor implementations. The proposed solution exploits the technique proposed in Reference [13] to automatically find the simplex containing a given input $\boldsymbol{x}$. The circuit has been validated by implementing a four-dimensional domain prototype. The main advantage of the proposed solution is its *modularity* and *flexibility*: incrementing the input dimension requires the addition of an element to the main board, whereas changing the PWL function requires a change in the contents of a memory.

The paper is organized as follows. In Section 2 we shall introduce some basic definitions. Section 3 describes a three-step method yielding a principle implementation scheme for a given PWL approximation. Section 4 proposes a simpler analog/digital circuit approximation scheme. In Section 5 we shall provide a prototype and some experimental results. Some concluding remarks are given in Section 6.

2. BASIC DEFINITIONS

The input arguments to the function are assumed to belong to the hyperrectangular domain S. The length of the ith side of this hyperrectangle is $(b_i - a_i)$, and is divided into m_i equal segments. The (invertible) transformation

$$\boldsymbol{z} = \boldsymbol{T}(\boldsymbol{x}) = \begin{bmatrix} \frac{m_1}{b_1 - a_1} & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \frac{m_n}{b_n - a_n} \end{bmatrix} (\boldsymbol{x} - \boldsymbol{a}) \tag{3}$$

maps the domain S into the scaled domain $\{\boldsymbol{z} \in \mathbb{R}^n : \boldsymbol{0} \leqslant \boldsymbol{z} \leqslant \boldsymbol{m}\}$.[‡] By assuming that all axes have the same number of segments $m = 2^p - 1$ (p is a positive integer), each component of $\boldsymbol{m}$ can be encoded into a binary number of p bits and the scaled domain is an (hypercube) and will be henceforth denoted as C_p.

The hardware realization of the block implementing transformation $\boldsymbol{T}(\boldsymbol{x})$ is rather direct (e.g. through op-amps). Moreover, it can be viewed as a 'boundary' operation with respect to the PWL processing of the normalized scaled vector $\boldsymbol{z}$. For this reason, our hardware implementation only deals with the PWL block, which is the 'core' of the realization.

The domain C_p is partitioned into $\prod_{i=1}^n m_i = m^n$ hypercubes. The set of $N = \prod_{i=1}^n (m_i + 1) = (m+1)^n = 2^{np}$ hypercube vertices contained in C_p is defined as $V_p = \{\boldsymbol{I} \in \mathbb{N}^n : \boldsymbol{0} \leqslant \boldsymbol{I} \leqslant \boldsymbol{m}\}$, where every component of $\boldsymbol{I}$ is 0 or a positive integer $\leqslant m$. Each hypercube is in turn partitioned into $n!$ simplices, and every simplex is identified by its $n+1$ vertices. The subdivision of hypercubes of C_p into non-overlapping simplices is done by properly arranging the vertices of V_p in a fixed order [13]. Then, a generic point $\boldsymbol{z} \in C_p$ can be uniquely expressed as

$$\boldsymbol{z} = \sum_{j=0}^{q} \mu_j \boldsymbol{I}_j \tag{4}$$

where $\mu_j > 0$, $\boldsymbol{I}_j \in V_p$, $q \leqslant n$, $\sum_{j=0}^{q} \mu_j = 1$ and $\boldsymbol{I}_0 \leqslant \boldsymbol{I}_1 \leqslant \ldots \leqslant \boldsymbol{I}_q \leqslant \boldsymbol{I}_0 + 1$. If $q = n$, then no vector $\boldsymbol{I}_j$ has a weighting coefficient μ_j equal to zero. In this case, $\boldsymbol{z}$ lies within the simplex defined by the vertices $\{\boldsymbol{I}_0, \boldsymbol{I}_1, \ldots, \boldsymbol{I}_n\}$. Otherwise, $\boldsymbol{z}$ lies on the boundary given by the convex combination of those q vertices with associated coefficients $\mu_j > 0$. If $q = 0$, then the point $\boldsymbol{z}$ coincides with the vertex $\boldsymbol{I}_0$ (and $\mu_0 = 1$). Then, index j defines a *local* (i.e. related to the simplex or to the boundary where $\boldsymbol{z}$ lies) labelling. For the sake of compactness, we shall define the set $\Im_z = \{0, 1, \ldots, q\}$.

3. CIRCUIT ARCHITECTURE

Our aim is to obtain a circuit synthesis for multiport resistors described by piecewise-linear (PWL) constitutive equations. Every individual function is in the form of $f(\boldsymbol{z}) = \sum_{k=1}^{N} c_k \alpha_k(\boldsymbol{z})$, where $\alpha_k(\boldsymbol{z})$ denotes the kth component of a basis of PWL functions such that $\alpha_k(\boldsymbol{I}_j) = \delta_{kj}$ (i.e. each basis function α_k is a hyperpyramid with compact support and the c_k's are proper weighting coefficients [1]).

With this in mind, the generic value $f(\boldsymbol{z})$ can be expressed as follows:

$$f(\boldsymbol{z}) = \sum_{k=1}^{N} c_k \alpha_k \left(\sum_{j \in \Im_z} \mu_j \boldsymbol{I}_j \right) = \sum_{k=1}^{N} c_k \sum_{j \in \Im_z} \mu_j \alpha_k(\boldsymbol{I}_j) = \sum_{j \in \Im_z} c_j \mu_j \tag{5}$$

[‡]Henceforth, according to Reference [13], we shall use the following notations (with $\boldsymbol{x}$, $\boldsymbol{y}$ n-length vectors and α scalar):
$\boldsymbol{x} \leqslant \boldsymbol{y}$ means $x_i \leqslant y_i$ $(i = 1, \ldots, n)$
$\boldsymbol{x} + \alpha$ means the vector whose components are $x_i + \alpha$ $(i = 1, \ldots, n)$.

As a consequence, $f(\mathbf{z})$ can be calculated once the local set $\Im_z$ and the corresponding coefficients μ_j and weights c_j have been found. As a matter of fact, a circuit realization of a PWL approximation requires three steps:

(S1) a method to find, for any given $\mathbf{z}$, the set $\Im_z$ and the corresponding coefficients μ_j;
(S2) a memory where the c_k coefficients are stored;
(S3) a circuit block performing operation (5).

Step (S1): The procedure used to find the μ_j coefficients is formulated in such a way that it always produces $n+1$ values. When $q<n$, $(n-q)$ coefficients μ_j are equal to zero. This helps in doing a fully parallel implementation.

Following the ideas found in Reference [13], we define

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} \tag{6}$$

$$\mathbf{I}_0 = \lfloor \mathbf{z} \rfloor = \begin{bmatrix} \lfloor z_1 \rfloor \\ \lfloor z_2 \rfloor \\ \vdots \\ \lfloor z_n \rfloor \end{bmatrix} \tag{7}$$

The definition of $\mathbf{I}_0$ is the first operation to be performed by the circuit. Then, we define

$$\delta\mathbf{z} = \mathbf{z} - \mathbf{I}_0 = \begin{bmatrix} \delta z_1 \\ \delta z_2 \\ \vdots \\ \delta z_n \end{bmatrix} \tag{8}$$

$$\delta\hat{\mathbf{z}} = \begin{bmatrix} \delta z_{\mathrm{MAX}} \\ \vdots \\ \delta z_{\mathrm{MIN}} \end{bmatrix} \tag{9}$$

where $\delta z_{\mathrm{MAX}} = \max_i \{\delta z_i\}$, $\delta z_{\mathrm{MIN}} = \min_i\{\delta z_i\}$, and the remaining components of $\delta\hat{\mathbf{z}}$ are organized in decreasing order of magnitude. It must be stressed that, owing to the definition of the C_p domain, $\delta z_i \geqslant 0$ for any i.

Each of operations (8) and (9) needs a proper circuit block. As explained in Reference [13], we can calculate the μ_j coefficients as follows:

$$\begin{aligned} \mu_0 &= 1 - \delta z_{\text{MAX}} = 1 - \delta \hat{z}_1 \\ \mu_1 &= \delta \hat{z}_1 - \delta \hat{z}_2 \\ &\cdots \\ \mu_{n-1} &= \delta \hat{z}_{n-1} - \delta \hat{z}_n = \delta \hat{z}_{n-1} - \delta z_{\text{MIN}} \\ \mu_n &= \delta \hat{z}_n = \delta z_{\text{MIN}} \end{aligned} \tag{10}$$

When $\boldsymbol{z}$ lies on the boundary of a simplex, the number q is smaller than n. Correspondingly, $(n-q)$ elements calculated by Equations (10) are null. By definition, the indices j for which $\mu_j > 0$ are those belonging to $\Im_z$.

Operations (10) also need a proper circuit block.

Step (S2): Now, we need to retrieve from a memory the coefficients corresponding to the μ_j's (i.e. to the set $\Im_z$). To do that, we have to associate a binary address (made up of $n \times p$ bits, i.e. p bits for each dimensional component of the domain) to each vertex belonging to V_p. These addresses are obtained as follows. The binary vector $\mathscr{B}(\boldsymbol{I}_0)$ obtained by applying to the $\boldsymbol{I}_0$'s co-ordinates the binarizing operator $\mathscr{B}(\cdot)$ gives the address of $\boldsymbol{I}_0$. The other addresses are obtained by adding $\mathscr{B}(\boldsymbol{I}_0)$ to the binary 'differential' vectors $\boldsymbol{d}_j$ $(j = 1, \ldots, n)$, which derive from the following procedure.

First of all, we define the unitary step function:

$$u(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geqslant 0 \end{cases} \tag{11}$$

The vectors $\boldsymbol{d}_j$ $(j = 1, \ldots, n)$ are obtained as

$$\boldsymbol{d}_j = \begin{bmatrix} u(\delta z_1 - \delta \hat{z}_j) \\ u(\delta z_2 - \delta \hat{z}_j) \\ \vdots \\ u(\delta z_n - \delta \hat{z}_j) \end{bmatrix} \tag{12}$$

Then, the parallel calculation of these vectors requires n circuit blocks. Each of them takes the vector $\delta \boldsymbol{z}$ and one of the $\delta \hat{\boldsymbol{z}}$ components (from δz_{MAX} to δz_{MIN}) and applies the binary function u to each of the n difference terms. The complete addresses a_j for the needed c_j coefficients are obtained by first calculating the binary vectors $\boldsymbol{a}_j = \mathscr{B}(\boldsymbol{I}_0) + \boldsymbol{d}_j$ and then by applying to each $\boldsymbol{a}_j$ the '*string*' operator, which juxtaposes the components of the vector $\boldsymbol{a}_j$:

$$a_j = \text{string}(\boldsymbol{a}_j) = [\boldsymbol{a}_j(n) | \boldsymbol{a}_j(n-1) | \cdots | \boldsymbol{a}_j(1)] \tag{13}$$

This amounts to say that $\mathscr{B}(\boldsymbol{I}_0)$ and the n sums $\boldsymbol{a}_j = \mathscr{B}(\boldsymbol{I}_0) + \boldsymbol{d}_j$ need $n+1$ further circuit blocks (the first yields $\mathscr{B}(\boldsymbol{I}_0)$, and the others provide the vector sums). The binary string generated by expression (13) corresponds to a number which labels a vertex according to the arrangement chosen in Reference [13].

Step (S3): This is the final step. The addressed c_j coefficients and the μ_j terms must be multiplied and then added, according to $f(\boldsymbol{z}) = \sum_{j \in \Im_z} c_j \mu_j$.

Figure 1 shows a scheme for steps (S1) and (S2), whereas Figure 2 shows a scheme for step (S3).

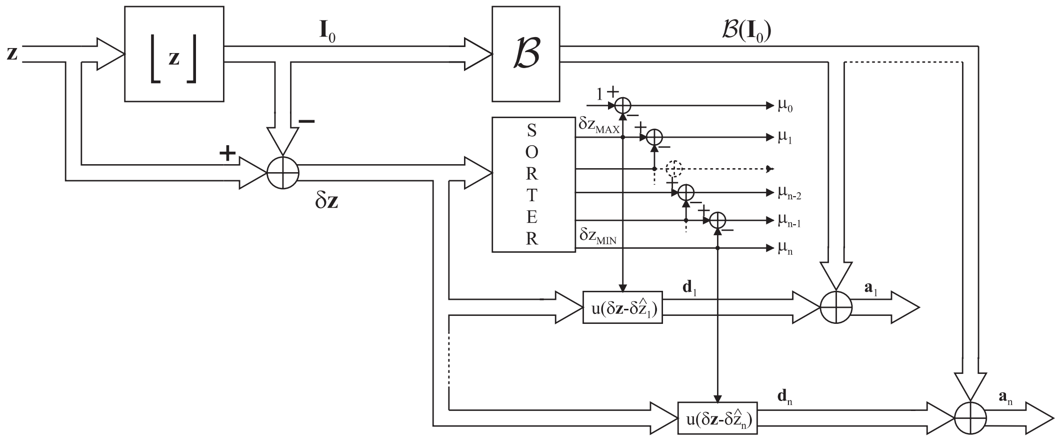

Figure 1. Principle scheme for steps (S1) and (S2).

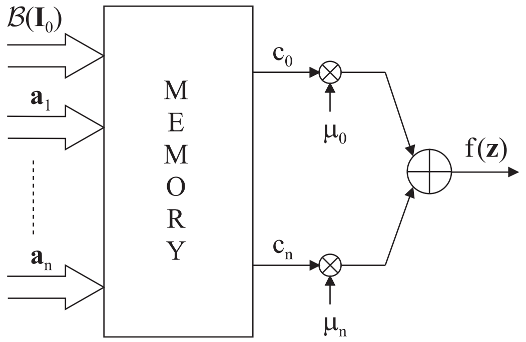

Figure 2. Principle scheme for step (S3).

Example
We shall consider a two-dimensional domain C_p with $m = 2^4 - 1 = 15$ (then $N = 256$). If we assume $\boldsymbol{z} = \begin{bmatrix} 5.2 \\ 9.5 \end{bmatrix}$, then

$$\boldsymbol{I}_0 = \lfloor \boldsymbol{z} \rfloor = \begin{bmatrix} 5 \\ 9 \end{bmatrix} \Longrightarrow \mathscr{B}(\boldsymbol{I}_0) = \begin{bmatrix} 0101 \\ 1001 \end{bmatrix}$$

$$\delta\boldsymbol{z} = \begin{bmatrix} 0.2 \\ 0.5 \end{bmatrix}, \quad \delta\hat{\boldsymbol{z}} = \begin{bmatrix} 0.5 \\ 0.2 \end{bmatrix} = \begin{bmatrix} \delta z_{\text{MAX}} \\ \delta z_{\text{MIN}} \end{bmatrix}$$

$$\mu_0 = 1 - \delta z_{\text{MAX}} = 0.5$$

$$\mu_1 = \delta z_{\text{MAX}} - \delta\hat{z}_2 = 0.5 - 0.2 = 0.3$$

$$\mu_2 = \delta z_{\text{MIN}} = \delta\hat{z}_2 = 0.2$$

Furthermore, we have

$$\delta\boldsymbol{z} - \delta\hat{z}_1 = \begin{bmatrix} 0.2 \\ 0.5 \end{bmatrix} - 0.5 = \begin{bmatrix} -0.3 \\ 0 \end{bmatrix} \Longrightarrow \boldsymbol{d}_1 = u(\delta\boldsymbol{z} - \delta\hat{z}_1) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\Longrightarrow \boldsymbol{a}_1 = \mathscr{B}(\boldsymbol{I}_0) + \boldsymbol{d}_1 = \begin{bmatrix} 0101 \\ 1001 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0101 \\ 1010 \end{bmatrix}$$

$$\delta\boldsymbol{z} - \delta\hat{z}_2 = \begin{bmatrix} 0.2 \\ 0.5 \end{bmatrix} - 0.2 = \begin{bmatrix} 0 \\ 0.3 \end{bmatrix} \Longrightarrow \boldsymbol{d}_2 = u(\delta\boldsymbol{z} - \delta\hat{z}_2) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\Longrightarrow \boldsymbol{a}_2 = \mathscr{B}(\boldsymbol{I}_0) + \boldsymbol{d}_2 = \begin{bmatrix} 0101 \\ 1001 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0110 \\ 1010 \end{bmatrix}$$

Finally,

$$a_1 = \text{string}(\boldsymbol{a}_1) = 10100101$$

$$a_2 = \text{string}(\boldsymbol{a}_2) = 10100110$$

4. A PRACTICAL CALCULATION SCHEME

The simpler scheme shown in Figure 3 can be obtained by generalizing the ideas proposed in Reference [12]. The central idea is that the sorting of the $\delta\boldsymbol{z}$ components is obtained *sequentially* through a voltage ramp $r(t)$ associated with a set of n voltage comparators (the

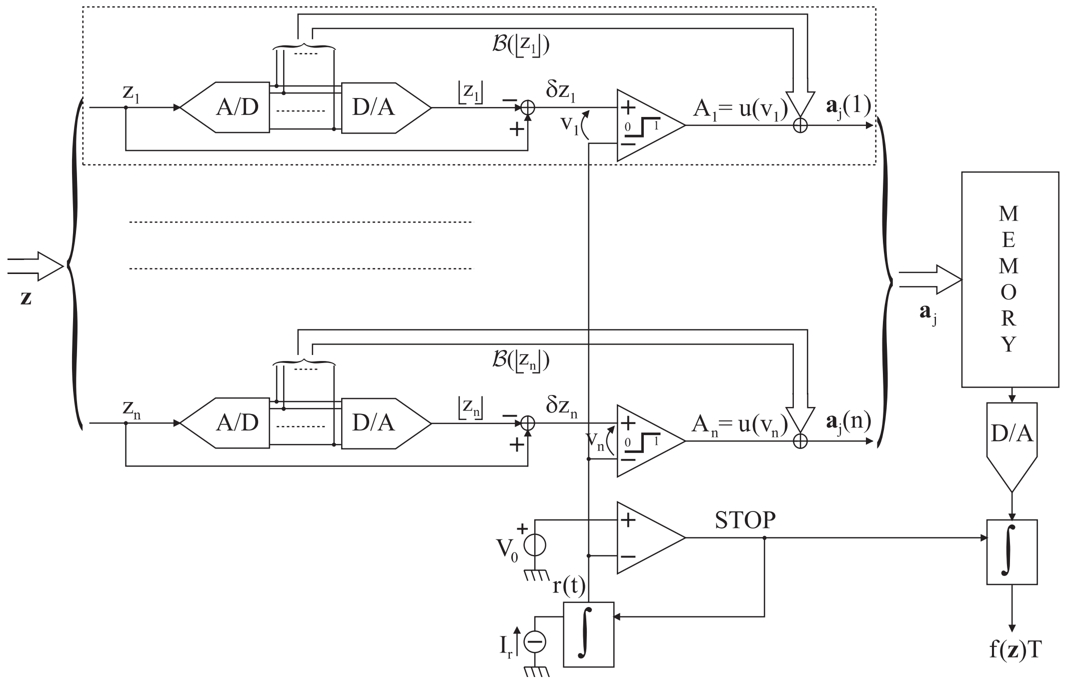

Figure 3. A simpler calculation scheme.

ramp $r(t)$ is defined for $t \in [0, T]$, where T is such that $r(T) = V_0 = 1V$). Each output bit A_i (initially set to 1) switches from 1 to 0 as the voltage ramp at the input side of its voltage comparator reaches the value $\delta \boldsymbol{z}_i$. The time-varying binary sequence $A_1, \ldots, A_n$ and the binary vector $\mathscr{B}(\boldsymbol{I}_0)$ represent the complete information necessary to address the $n+1$ memory positions corresponding to the simplex containing the point $\boldsymbol{z}$. In particular, each component $\boldsymbol{a}_j(i)$ $(i = 1, \ldots, n)$ of the vector $\boldsymbol{a}_j$ giving the coefficient c_j in the memory is a binary string obtained by simply summing the string $\mathscr{B}(\lfloor z_i \rfloor)$ and the actual value of the bit $A_i = u(\delta z_i - r(t))$ in the less significant position. Thus, the sequence of the n binary strings $\boldsymbol{a}_j(1), \ldots, \boldsymbol{a}_j(n)$ define the memory address a_j. The output of the D/A converter in cascade with the memory is the coefficient c_j addressed by a_j. Each address a_j holds its value on until the ramp $r(t)$ sets to 0 the output of a further voltage comparator. It is rather easy to verify that by integrating the analog output of the D/A converter over the time interval $[0, T]$ the output voltage is proportional to $c_n \mu_n + c_{n-1} \mu_{n-1} + \cdots + c_0 \mu_0 = f(\boldsymbol{z})$, as each of the c_j coefficients at the input of the integrator holds on for a time interval proportional to μ_j.

We point out the *modularity* of the proposed scheme, as, for each dimensional component of the input vector $\boldsymbol{z}$, we only have to add to the circuit scheme a basic block of the kind shown in the dashed box in Figure 3.

Example

We consider the same example as before. Following the scheme shown in Figure 3, we have:

$$\mathscr{B}(\lfloor z_1 \rfloor) = 0101, \quad \mathscr{B}(\lfloor z_2 \rfloor) = 1001$$

$$\delta z_1 = 0.2V, \quad \delta z_2 = 0.5V$$

$$r(t) = V_0 \frac{t}{T}, \quad V_0 = 1V$$

$$v_1(t) = \delta z_1 - r(t) \Longrightarrow A_1(t) = u(v_1(t)) = \begin{cases} 1 & \text{for } 0 \leqslant t \leqslant t_1 \\ 0 & \text{for } t > t_1 \end{cases}$$

with

$$t_1 = T \frac{\delta z_1}{V_0} = 0.2T = \mu_2 T$$

$$v_2(t) = \delta z_2 - r(t) \Longrightarrow A_2(t) = u(v_2(t)) = \begin{cases} 1 & \text{for } 0 \leqslant t \leqslant t_2 \\ 0 & \text{for } t > t_2 \end{cases}$$

with

$$t_2 = T \frac{\delta z_2}{V_0} = 0.5T = (\mu_1 + \mu_2)T$$

In the interval $[0, t_1]$, $A_1(t) = A_2(t) = 1$ and

$$\mathscr{B}(\lfloor z_1 \rfloor) + A_1 = 0110 = \boldsymbol{a}_2(1), \quad \mathscr{B}(\lfloor z_2 \rfloor) + A_2 = 1010 = \boldsymbol{a}_2(2) \Longrightarrow a_2 = 10100110$$

The addressed coefficient c_2 is integrated in the same time interval. At $t = t_1$, the resulting value is $c_2 \mu_2 T$.

In the interval $[t_1, t_2]$, $A_1(t) = 0$ and $A_2(t) = 1$. Then,

$$\mathscr{B}(\lfloor z_1 \rfloor) + A_1 = 0101 = \boldsymbol{a}_1(1), \quad \mathscr{B}(\lfloor z_2 \rfloor) + A_2 = 1010 = \boldsymbol{a}_1(2) \Longrightarrow a_1 = 10100101$$

The addressed coefficient c_1 is integrated in the same time interval, thus giving $c_1(t_2 - t_1) = c_1(T/V_0)(\delta z_2 - \delta z_1) = c_1(0.3T) = c_1 \mu_1 T$. At $t = t_2$, the resulting value is $(c_2\mu_2 + c_1\mu_1)T$.

In the interval $[t_2, T]$, $A_1(t) = 0$ and $A_2(t) = 0$. Then,

$$\mathscr{B}(\lfloor z_1 \rfloor) + A_1 = 0101 = \boldsymbol{a}_0(1), \quad \mathscr{B}(\lfloor z_2 \rfloor) + A_2 = 1001 = \boldsymbol{a}_0(2) \Longrightarrow a_0 = 10010101 = \text{string}(\mathscr{B}(\boldsymbol{I}_0))$$

The addressed coefficient c_0 is integrated in the same time interval, thus giving $c_0(T - t_2) = c_0 T(1 - (\delta z_2/V_0)) = c_0(0.5T) = c_0\mu_0 T$. Finally, at $t = T$, the resulting value is $(c_2\mu_2 + c_1\mu_1 + c_0\mu_0)T = f(\boldsymbol{z})T$.

5. ILLUSTRATIVE EXAMPLE: A VALIDATION PROTOTYPE

The architecture proposed in Section 4 must be regarded as a reference scheme, where the analog parts should ensure a good performance in terms of processing speed and accuracy.

The purpose of this section is to illustrate the presented approach. With this idea in mind, an experimental prototype was built. Given that the objective was the validation and illustration of the algorithm, rather than speed or accuracy, the architecture was simplified by working with digital signals only. The advantage of realizing a prototype on a completely digital basis is an easier validation of the algorithm due to the robustness offered by digital processing circuits. Following this line of reasoning, in our prototype the analog part has been greatly reduced. In particular:

- the D/A converter in the input blocks is eliminated;
- the ramp generator is replaced by a digital ramp;
- digital comparators substitute the analog comparators;
- the output block (D/A converter plus integrator) is replaced by a properly controlled logic circuit which sums the outputs of the EPROM and performs the D/A conversion of the result.

Figure 4 shows the results obtained by using a clock frequency of 2.5 MHz for the digital counter implementing the ramp generator and an input sampling rate of 10 kHz. Then, the maximum allowed frequency component in the spectrum of the input signal is less than 5 kHz. Actually, this upper limit is merely theoretical, since, due to non-linearity, both $y(t) = f(\boldsymbol{x}(t))$ and its PWL approximation $f_{\mathrm{PWL}}(\boldsymbol{x}(t))$ can have a frequency spectrum widely larger than that of the input signal $\boldsymbol{x}(t)$.

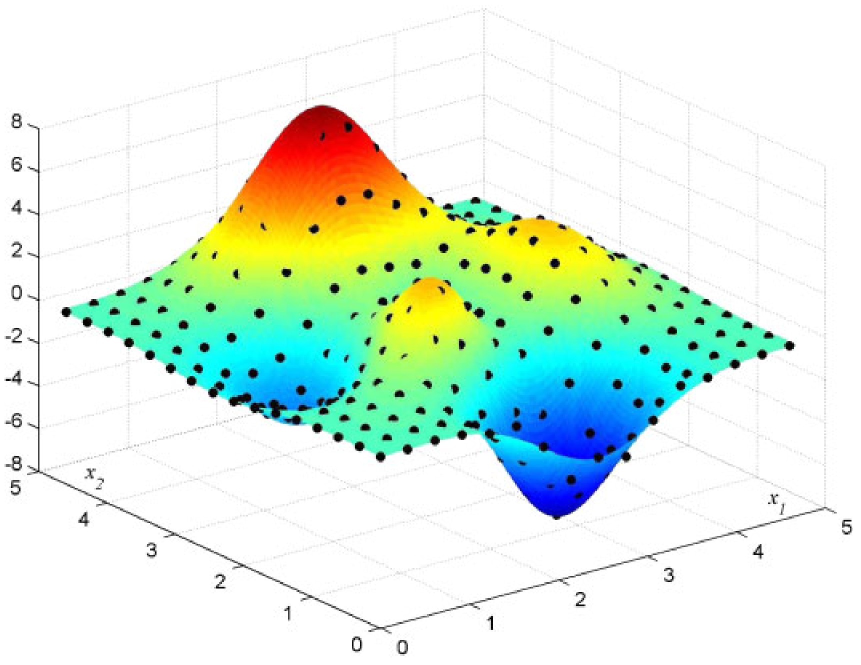

Figure 4. Matlab ‘peaks’ function (smooth surface) and circuit samples of its PWL approximation (black dots).

The circuit allows an input dimension $n = 4$. As an example, we propose the PWL approximation of a function defined over a two-dimensional domain. The normalized domain C_p is defined with $m = 15$ (i.e. $p = 4$). As a consequence, each address a_j is a string of $n \times p = 16$ bits. The $N = 256$ coefficients c_k are stored in a small fraction of the 1 Mb EPROM necessary in the more general four-dimensional case.

6. CONCLUDING REMARKS

In this work, we have described in detail a synthesis procedure that, starting from the simplicial subdivision (1) and the consequent PWL approximation f_{PWL}, yields a complete A/D circuit architecture.

A remarkable aspect of this work is the modularity and flexibility of the architecture: an increase of a unit in the domain dimension simply corresponds to the addition of an input module, and the function to be implemented can be changed by re-programming the memory. Other advantages with respect to fully analog implementations (see Reference [7]) concern the higher accuracy and controllability provided by mixed-signal and fully digital implementations. Generally speaking, the higher accuracy is payed in terms of speed. In the fully digital implementation proposed in Section 5, this depends *in primis* on the relationship between the clock frequency f_c, the output sampling period T (see Section 4) and the number n_b of bits used to represent the $\delta\boldsymbol{z}$ components: $T = 2^{n_b}/f_c$. In a mixed-signal implementation, some parts of the proposed architecture can be implemented in analogic technology, thus increasing the processing speed at the cost of a lower accuracy.

The overall accuracy of the PWL approximation depends also on both the number N of basis functions (and coefficients) and the bit size for each coefficient stored in the memory, i.e. on the available memory size. On the other side, the maximum allowed working speed of the overall circuit depends also on the frequency spectrum of the signal $y(t) = f(\boldsymbol{x}(t))$.

A first prototype of the proposed architecture has been implemented on a fully digital fashion. This was made for the sake of simplicity, as we aimed to validate the proposed synthesis procedure. However, since the proposed circuit is based on standard electronic blocks, a CMOS implementation, whether it is mixed-signal or fully digital, follows immediately from the principles described. An implementation of this kind could be tailored depending on the application to achieve different functional specifications, for instance high speed or high density for cases of high precision or high dimensionality of the input space.

ACKNOWLEDGEMENTS

The authors wish to thank Mr Giorgio Carlini and the students Giovanni Bozza and Mauro Gaggero for the hardware implementation of the prototype, and Dr Lorenzo Repetto for helpful suggestions and comments. M. Parodi and M. Storace have been partially supported by the MIUR, within the PRIN and FIRB frameworks, and by the University of Genoa. P. Julián was partially supported by project No. 14169-12 'Estructuras de Procesamiento en Paralelo Basadas en Redes Celulares No-Lineales' of Fundación Antorchas.

REFERENCES

1. Storace M, Julián P, Parodi M. Synthesis of nonlinear multiport resistors: a PWL approach. *IEEE Transactions on Circuits and Systems—I* 2002; **49**:1138–1149.
2. Storace M, Repetto L, Parodi M. A method for the approximate synthesis of cellular nonlinear networks—Part 1: Circuit definition. *International Journal of Circuit Theory and Applications* 2003; **31**:277–297.
3. Repetto L, Storace M, Parodi M. A method for the approximate synthesis of cellular nonlinear networks—Part 2: Circuit reduction. *International Journal of Circuit Theory and Applications* 2003; **31**:299–313.
4. Julián P, Desages A, Agamennoni O. High-level canonical piecewise linear representation using a simplicial partition. *IEEE Transactions on Circuits and Systems—I* 1999; **46**:463–480.
5. Julián P, Desages A, D'Amico B. Orthonormal high level canonical PWL functions with applications to model reduction. *IEEE Transactions on Circuits and Systems—I* 2000; **47**:702–712.
6. Chua LO. Device modeling via basic nonlinear circuit elements. *IEEE Transactions on Circuits and Systems* 1980; **27**:1014–1044.
7. Storace M, Parodi M. Towards analog implementations of PWL two-dimensional non-linear functions. *International Journal of Circuit Theory and Applications* 2005; **33**:147–160.
8. Serrano-Gotarredona T, Linares-Barranco B. Current-mode fully-programmable piece-wise-linear block for neuro-fuzzy applications. *Electronics Letters* 2002; **38**:1165–1166.
9. Balsi M, Giuliani G. Current-mode programmable piecewise-linear neural synapses. *International Journal of Circuit Theory and Applications* 2003; **31**:265–275.
10. Andreou AG, Boahen KA. Translinear circuits in subthreshold CMOS. *Journal of Analog Integrated Circuits and Signal Processing* 1996; **9**:141–166.
11. Julián P, Dogaru R, Chua LO. A piecewise-linear simplicial coupling cell for CNN gray-level image processing. *IEEE Transactions on Circuits and Systems—I* 2002; **49**:904–913.
12. Dogaru R, Julián P, Chua LO, Glesner M. The simplicial neural cell and its mixed-signal circuit implementation: An efficient neural-network architecture for intelligent signal processing in portable multimedia applications. *IEEE Transactions on Neural Networks* 2002; **13**:995–1008.
13. Chien M, Kuh E. Solving nonlinear resistive networks using piecewise-linear analysis and simplicial subdivision. *IEEE Transactions on Circuits and Systems* 1977; **24**:305–317.