ARCHITECTURES AND BUILDING BLOCKS FOR CMOS VLSI ANALOG "NEURAL" PROGRAMMABLE OPTIMIZERS

R. Domínguez-Castro, A. Rodríguez-Vázquez, J.L. Huertas and E. Sánchez-Sinencio.

Dept. of Design of Analog Circuits.
Centro Nacional de Microelectrónica. Sevilla, SPAIN

Abstract

A modular reconfigurable serial architecture is presented for the analog/digital implementation of constrained optimization algorithms with digital programmability of the problem weights. Area overhead due to programmability is reduced by using a time multiplexing methodology. It allows all the weights of each multiple inputs processing unit to be digitally-controlled by just using one weighted component array. The proposed architecture is very well suited for MOS VLSI realization using Switched-Capacitor (SC) techniques. SC schematics for the different building blocks are presented and demonstrated via empirical results.

Introduction

It has been recently demonstrated that the problem of minimizing a multivariable *cost function* $\Phi(\bar{x})$ subjected to a set of *constraints* $F_k(\bar{x}) \geq 0$, $(1 \leq k \leq Q)$, can be solved in *real-time* by using analog *neural feedback networks* [1], [2], [3]. These networks show potential for those applications where on-line optimization is required as in robotics, satellite guidance, etc. Previously reported architectures are of the parallel type and do not realistically consider the case where either the problem cost function or the constraints can be electronically controlled. They hence serve just as vehicles to demonstrate the possibility of solving programming problems via integrated analogue circuits but not to support practical applications of this kind of circuits. To this purpose, electrical programmability is a crucial issue to be covered.

Programmability in an analog optimizer can be incorporated by following one of two basic approaches: **a)** Using electrically controlled devices, like for instance the transconductance of a differential pair; **b)** using digitally-controlled component arrays. The first approach may perhaps be more convenient for trainable networks [4]. However in case the weights are to be externally set, as it happens in our application, the use of weighted component arrays leads to much more accurate system implementations.

In this paper a mixed-mode sampled-data architecture is presented which efficiently combines digital and analog techniques for the implementation or programmable real-time optimizers. In order to reduce area overhead due to the use of one weighted component array per input, as would be needed in a parallel architecture, the herein proposed architecture is of the serial type. A time multiplexing methodology is used allowing all the weights of each processing unit to be digitally controlled by just using one weighted component array. The proposed architecture is appropriate for different analog sampled-data techniques, being demonstrated in the paper for the case of switched-capacitor circuits.

Algorithms for Sampled-Data Optimizers.

Let consider the problem of minimizing a quadratic cost function with linear constraints:

$$\textit{minimize}$$

$$\Phi(\mathbf{x}) = \sum_{i=1}^{N} a_i x_i + \frac{1}{2}\left\{\sum_{i=1}^{N}\sum_{j=1}^{N} g_{ij} x_i x_j\right\} \quad , \; g_{ij} \doteq g_{ji} \tag{1}$$

$$\textit{subjected to}$$

$$F_k(\mathbf{x}) = \sum_{i=1}^{N} b_{ki} x_i + b_{k0} \geq 0 \qquad , \qquad 1 \leq k \leq Q$$

The solution of this problem using sampled-data techniques requires to firstly map it onto a discrete-time algorithm. Among different alternatives [5], three approaches will be considered herein: **a)** The use of the modified external penalty technique [3]; **b)** The use of the external quadratic penalty [2], [5] and **c)** The use of Lagrange multipliers [6]. The algorithm for each one of these techniques can be formulated in the form of a discrete-time state equation, respectively:

$$x_i(n+1) = x_i(n) - \frac{1}{\tau_o}\left[U(\mathbf{F})\left\{a_i + \sum_{j=1}^{N} g_{ij} x_j(n)\right\} - \mu \sum_{k=1}^{Q} U(-F_k)\, b_{ki}\right] \tag{2}$$

for the modified external penalty,

$$x_i(n+1) = x_i(n) - \frac{1}{\tau_o}\left[\frac{1}{\mu}\left\{a_i + \sum_{j=1}^{N} g_{ij} x_j(n)\right\} + \sum_{k=1}^{Q} b_{ki} F_k U(-F_k)\right] \tag{3}$$

for the quadratic penalty, and

$$x_i(n+1)=x_i(n)-\frac{1}{\tau_o}\left[a_i+\sum_{j=1}^{N}g_{ij}x_j(n)+\sum_{k=1}^{Q}b_{ki}\lambda_k(n)U[-\lambda_k(n)]+b_{ki}\mu F_k(n)U[-\mu F_k(n)]\right] \tag{4a}$$

$$\lambda_k(n+1)=\lambda_k(n)+\frac{1}{\tau_o}\mu F_k(n)U[-\lambda_k(n)] \tag{4b}$$

for the Lagrange multipliers technique. In previous expressions U($\bar{F}$) and $U(-F_k)$ are threshold operators defined as follows:

$$U(-F_k)=\begin{cases}1 & for \quad F_k<0\\ 0 & otherwise\end{cases} \tag{5}$$

$$U(\bar{F})=\begin{cases}1, & if \quad F_k\geq 0 \quad for\ every\ k\\ 0, & otherwise\end{cases}$$

For accuracy of those solutions located on the border of the feasibility region defined by the constraints, parameter μ in (3) has to be large enough [5], [7]. Quite in the contrary the value of this parameter (called penalty multiplier) is not critical for (2). Actually in this latter case it is possible to choose μ=1 without compromising the accuracy of the solution. However the modified external penalty has the drawback that asymptotic stability is not guaranteed in case the equilibrium point (it is to say the solution point) is on the border of the feasibility region. It imposes the need to use large time constants (parameter τ_0) [5]. The drawbacks of both penalty techniques can be overcome by resorting to (3) where accurate asymptotically stable solutions can be obtained with moderate values of the penalty multiplier. The price to be paid for is a significant increase of the hardware. This can be seen from Fig.1, where conceptual representations are shown for each one of the considered algorithms.

Sampled-Data Serial Programmable Optimizers

The different terms involved in the summations appearing in Fig.1 can be added simultaneously (as it happens in a parallel architecture) or via an accumulative serial process. Using this latter approach allows multiplexing of the components used to implement the weights in the summation blocks. Let us to the purpose of illustration consider the algorithm in (3). During the $(n+1)$*-th* computation cycle (defined as the time required to complete one iteration of (3)) the following summations must be made:

$$\Delta x_i(n+1)=-\frac{1}{\tau_0}\left\{\frac{a_i}{\mu}+\sum_{j=1}^{N}\frac{g_{ij}}{\mu}x_j(n)+\sum_{k=1}^{Q}b_{ki}F_k[n]U(-F_k)\right\} \tag{6a}$$

$$F_k(n+1)=\sum_{j=1}^{N}b_{kj}x_j(n)+b_{k0} \tag{6b}$$

for $i=1,2,...,N$ and $k=1,2,...,Q$. Also a comparison per constraint is required to evaluate the corresponding threshold operator.

Fig.2 shows SC schematics of the processing units required for the implementation of (6) using a serial architecture. Fig.2a is used for the computation of (6a) while Fig.2b corresponds to (6b). Fig.2c serves a double purpose. At the end of each computation cycle it stores the corresponding calculated values for $x_i(n)$ and $F_kU(-F_k)$. Then, during the next computation cycle, these values are one by one sequentially transferred to the output of this unit and hence to the input of the other processing units.

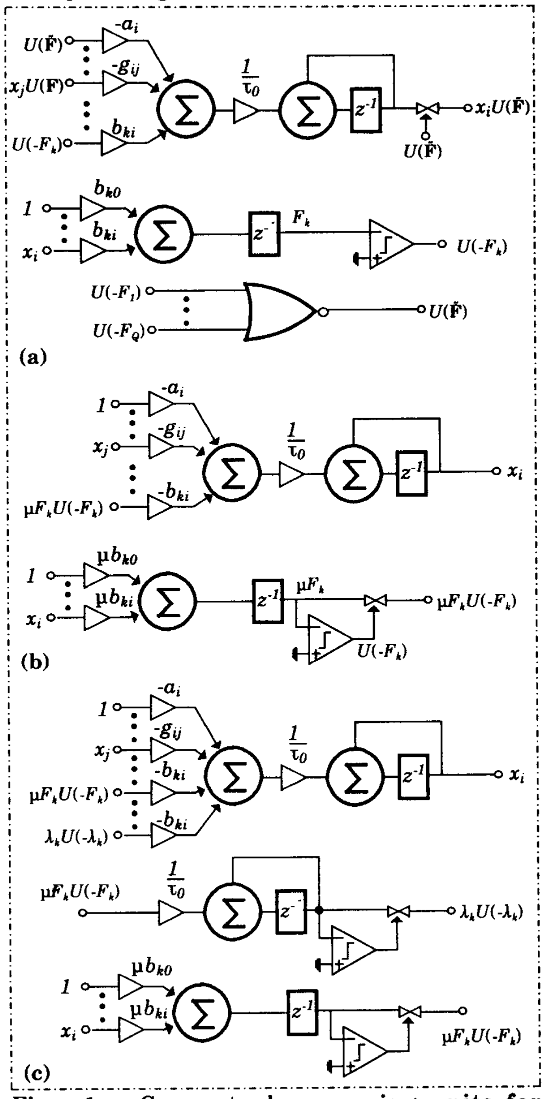

Figure1: Conceptual processing units for different optimization algorithms.

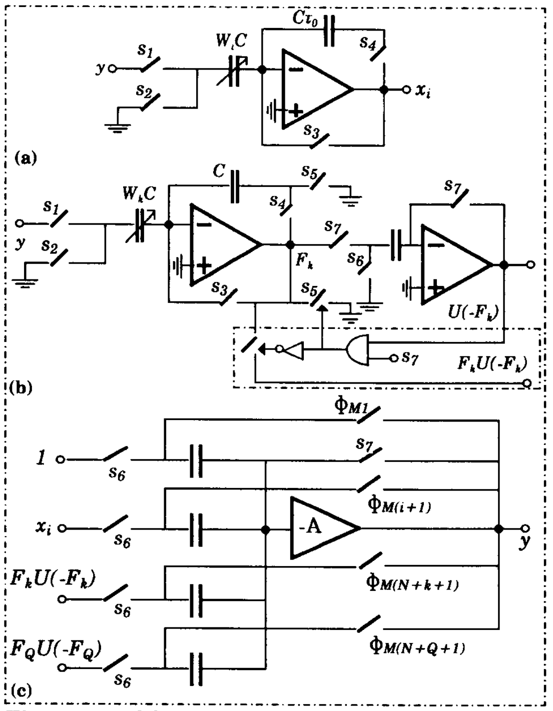

Figure 2: SC buidings blocks for the quadratic penalty optimization algorithm.

Computation flow in the proposed serial architecture is controlled by the clock signals shown in Fig.3. The master clock comprises two nonoverlapping clock phases, ϕ^e and ϕ^o, whose functions are to establish the general timing of the system and to control the transfer of charge among capacitors in the SC computational units. One transfer of charge is made per period of this master clock. As it can be seen from Fig.3, the computation cycle contains $N+Q+1$ periods of the master clock while the initialization cycle contains just two of these periods. Switches labeled S_3 and S_4 are respectively controlled by ϕ^e and ϕ^o all the time. In a similar way S_5 is controlled by ϕ_{I2} while ϕ_{I1} controls the switch S_6 and the complement of this signal is used to control S_7. Remaining switches in Fig.2c are controlled by the clock signals ϕ_{M1} to $\phi_{M(N+Q+1)}$ shown at the bottom of Fig.3. On the other hand, switching schedule of S_1 and S_2 depends on whether the system is in the computation or in the initialization cycle, being:

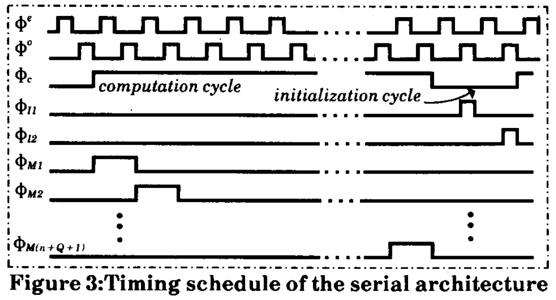

Figure 3:Timing schedule of the serial architecture

Computation cycle:	$s_1=\phi^e$	$s_2=\phi^o$
Initialization cycle:	$s_1=OFF$	$s_2=ON$.

Computation is made as follows. During the $(n+1)$-*th* initialization cycle the previously calculated values are stored in the capacitors of Fig.2c (for ϕ_{I1} high). Also, when ϕ_{I2} becomes high the feedback capacitors of the units used to evaluate the constraints are zeroed. Then, during the $(n+1)$-*th* computation cycle the output of Fig.2c sequentially takes the values $1, x_1(n)$, $x_2(n)$, ..., $x_N(n)$, $F_1(n)U(\text{-}F_1)$, $F_2(n)U(\text{-}F_2)$, ..., $F_Q(n)U(\text{-}F_Q)$, one per period of the master clock. This sequence is applied to the input of Fig.2a and Fig.2b, resulting in the following sequence at the corresponding outputs, respectively:

$$x_i\left(n+\frac{m}{M}\right)=x_i\left(n+\frac{m-1}{M}\right)+y\left(n+\frac{m}{M}\right)\frac{w_i}{\tau_0} \tag{7a}$$

$$F_k\left(n+\frac{m}{M}\right)=F_k\left(n+\frac{m-1}{M}\right)+y\left(n+\frac{m}{M}\right)w_k \tag{7b}$$

for $m=1,2,...,\ N+Q+1$ and where $M=N+Q+1$and, remind, $F(n)=0$.

We can hence see that the accumulation process formulated in (7) yields (6), after $N+Q+1$ cycles of the master clock, provided w_i and w_k changes during the computation cycle according to what is required by (6). In practice this can be achieved by implementing the associated variable capacitors via binary weighted capacitors arrays controlled by a cyclic memory containing the weights codes, as it is illustrated in Fig.4.

The proposed architecture can be modified in a very simple way for the implementation of the algorithm in (2). In this case a scheme similar to that enclosed by broken lines in Fig.2(b) has to be used to multiply the output of the processing unit of Fig.2(a) by the threshold operator $U(\mathbf{F})$. This operator can be evaluated as the result of the logical NOR operation among the Q individual operators $U(\text{-}F_k)$. Adaptation of the architecture for the implementation of the Lagrange multiplier technique given by (3) is also possible. In this case, as it can be seen from Fig.1, Q more processing units are required as compared to the quadratic penalty algorithm. The SC schematics for these new units is similar to the one in Fig.2c.

Programmability Reconfigurability Issues

Since weights in the proposed serial architecture are stored in digital form, programmability issues can be very easily incorporated. It just requires the setting of a digital circuitry allowing digital weights to be externally modified. An important programmability

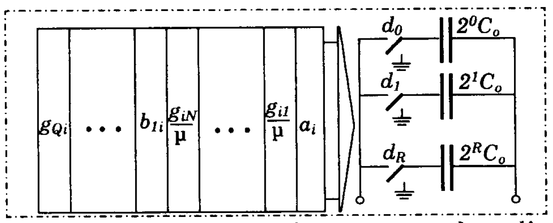

Figure 4: Binary capacitor array and cyclic control memory.

issue is related to the control of the integrator time constant, which may significantly influence the stability versus operation speed tradeoff. Since accurate control of the time constant value is nor crucial for proper operation, we can resort to this purpose to the use of a C-2C ladder, as the one shown in Fig.5

Reconfigurability can be also easily incorporated to the proposed serial architecture. Observe the functionality provided by Fig.2a is the same as for the first stage of Fig.2b, the difference being in that the feedback capacitor in this latter figure is zeroed during each initialization cycle. Thus a modular reconfigurable serial architecture can be built containing only processing units as the one inn Fig.2b plus some extra reconfiguration logic and corresponding control buses.

The combined programmability and reconfigurability features allows easier incorporation of testability issues. A testing mode of operation can be selected by the control logic where each processing unit is isolated and its operation is compared to that of a reference. The reconfigurability properties of the architecture can also be exploited to bypass faulty units and hence provide fault tolerance.

Stability is another system level issue that must be assessed for practical implementations. Since the herein considered implementations do not exhibit continuous-time feedback loops, potential instability problems are of the numerical type. Stability analysis of the different algorithms, for solutions inside and on the border of the feasibility region, allows the calculation of design conditions ensuring stability. Knowing these conditions allows in his turn to optimize the stability versus speed tradeoff. For instance for the Lagrange multipliers the following can be calculated for a linear problem to be solved at a maximum speed:

$$\tau_0 = 2 \tag{7a}$$

$$\mu = \frac{4}{\sum_{k=1}^{Q} \|\mathbf{B}_k\|^2} \tag{7b}$$

Similar conditions can be calculated for the other algorithms [5].

Discussion of Results

Floorplanning strategies for serial input architectures have been devised for us and the corresponding area occupation for each case has been calculated. Fig.6 shows such area as function of the number of processing units. For the purpose of comparison the estimated area figure for a parallel architecture has been also drawn. The area estimation has been made on the basis of assuming identical

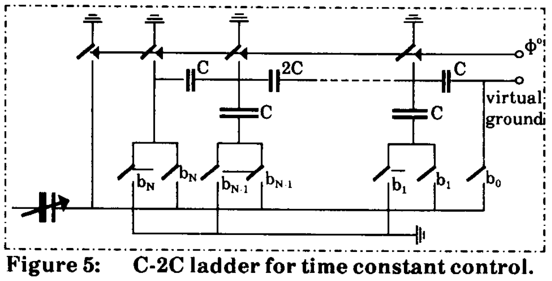

Figure 5: C-2C ladder for time constant control.

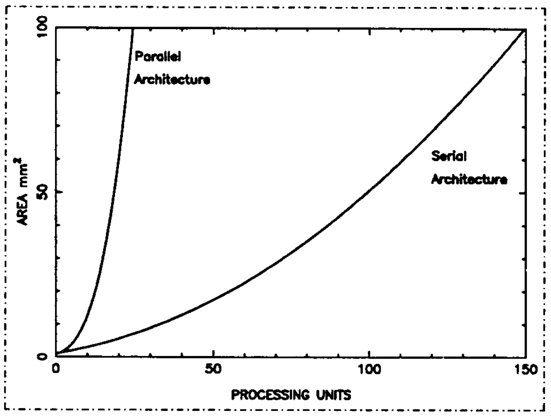

Figure 6: Comparative area figures for the serial and the paralell architecture.

opamps and unit capacitors values for both architectures. As it can be seen, an important area reduction can be achieved by using the serial input architecture. On the other hand, power consumption for the two architectures have been estimated by us to be very similar while operation speed for a given power can be also shown to be similar. (For a given opamp the finest clock signal in the serial architecture can be made to be about $N+Q+1$ times faster than the clock signal for the parallel inputs architecture).

References

[1] D.A. Tank and J.J. Hopfield: "Simple Neural Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit". *IEEE Trans. Circuit and Systems*, Vol.33, pp 533-541, May 1986.

[2] M.P. Kennedy and L.O. Chua: "Neural Networks for Nonlinear Programming". *IEEE Trans. Circuits and Systems*, Vol. 35, pp 554-562, May 1988.

[3] A. Rodríguez-Vázquez, et al: "Nonlinear Switched-Capacitor "Neural" Networks for Optimization Problems". ibid, Vol. 37, pp. 384-397, March 1990.

[4] C.Mead and M. Ismail (ed): "*Analog VLSI Implementation of Neural Systems*". Kluwer Academic 1989.

[5] R. Domínguez-Castro, et al: "Modeling and Design of Analog VLSI "Neural" Optimizers". *Proc. of the 1991 ECCTD*, pp. 489-497, 1991.

[6] R.Fletcher: "Practical Methods of Optimization", John Wiley & Sons, Vol.2 1980

[7] G.V. Vanderplaats: "*Numerical Optimization Techniques for Engineering Design: with Applications*", Mc. Graw-Hill 1984.