A Reconfigurable Architecture Mapping Multilayer CNN Paradigms

Luigi Raffo[a], Silvio P. Sabatini[b] and Giacomo M. Bisio[b]

[a]Istituto di Elettrotecnica
University of Cagliari
Piazza d'Armi, 09123 Cagliari, ITALY
tel.: +39-70-281342 fax: +39-70-291619
e_mail: luigi@elettro1.unica.it

[b]Department of Biophysical and Electronic Engineering
University of Genova
Via all'Opera Pia 11A, Genova, ITALY
tel.: +39-10-3532163 fax: +39-10-3532795
e_mail: silvio@dibe.unige.it

Abstract - *A digital VLSI implementation of linear template CNNs is presented. The reconfigurable architecture is organized as 12 layers of 64×64 cells. The CNNs are reformulated introducing sets of* generalized *cloning templates to enucleate more sharply the structure of both intra- and inter-layer cooperative computations. In this way it is possible to develop CNN algorithms for complex vision machine tasks. Various applications are considered in edge and connected component detection and in texture segregation.*

1. Introduction

In recent years, the processing capabilities of cellular neural networks (CNNs) have been intensively studied and many applications have been described, expecially for single-layer CNN structures. The growing interest in this field can be ascribed to the fact that CNNs exhibit the powerful processing capabilities of neural networks allowing, at the same time, an efficient VLSI implementation due to their local connectivity property. Still, to exploit all the processing capabilities of CNNs and to face real-world applications, the problem of hardware implementation should be considered in depth.

Various authors have pointed out that CNNs are especially tailored for analog processing and that they can be therefore directly mapped on a grid of simple analog processors. A number ofpapers have been devoted to the study of specific analog building blocks for CNN implementation or to programmable solutions based on operational amplifiers. However, several practical constraits/limitations on the efficacy of analog CNN implementation are posed by occupation area, input-output interfacing, status memorization and flexibility. Moreover these solutions have a limited programmability of the system parameters thus preventing a global reconfiguration of the structure of the elaboration (i.e. network structure for CNNs). In practical situations, programmability is a definitive need since, often, one has to face situations in which several completely different CNNs have to be used consecutively, and computation on large amount of data have to be partitioned on different simple CNNs working in parallel to achieve real-time performance. Hence, architectural solutions able to map, on the same computational substrate, several CNN paradigms, are strongly desirable to fully exploit the potentialities of CNN in real applications.

In this paper, starting from a generalized reformulation of cells' dynamics, we present a reconfigurable digital VLSI architecture able to fulfil both these demands on programmability and the requirements of higher efficiency respect to commercial DSP or hardware accelerators.

2. Generalized CNN

In its discrete time form the dynamics of a CNN cell is described by the equation

$$\begin{aligned} x_{ij}(n+1) &= \sum_{kl \in N_r(ij)} A_{ij;kl}(y_{ij}(n), y_{kl}(n)) + \sum_{kl \in N_r(ij)} B_{ij;kl}(u_{ij}(n), u_{kl}(n)) \qquad (1) \\ &+ \sum_{kl \in N_r(ij)} A^{\tau}_{ij;kl} y_{kl}(n-\tau) + \sum_{kl \in N_r(ij)} B^{\tau}_{ij;kl} u_{kl}(n-\tau) + I_{ij} \\ y(n) &= f(x(n)) \\ x(0) &= 0 \end{aligned}$$

where x, y, u, I denote respectively cell state, output, input, and bias, N_r is the neighborhood function, A, A^{τ}, B, B^{τ} are the cloning templates, τ is the memory duration time, and f is the nonlinear output function. N-dimensional generalizations (multilayer-CNNs) have been straightforwardly introduced in the original paper of Chua and Yang [1]. In multiple-layer CNNs each cell is characterized by several variables instead of only one state variable as in the single-layer case. Whatever be its level of complexity, a CNN can be computationally described by a multitude of *nodes* locally interacting to reach the desired computation. To each cell is associated a set of L nodes, that can be viewed as the components of a column vector $\mathbf{z}_{ij}$, representing the whole set of outputs, delayed outputs, and inputs related to the corresponding location (i, j) on the cell-layer. Each set of nodes interacts only with neighboring sets through a generalized 3D template D, that spans the L layers:

$$\begin{aligned} z^a_{ij}(n+1) &= g^a\Big(\sum_{1 \le b \le L} \sum_{kl \in N_r(ij)} D^{ab}_{ij,kl}(z^a_{ij}(n), z^b_{kl}(n))\Big) \qquad (2) \\ z^a(0) &= z^a_0 \end{aligned}$$

where a and b index the components of the column vector (i.e., the layer) and (i,j) the elements of a layer.

Fixing a component of a vector $\mathbf{z}_{ij}$, one can recognize in the corresponding section of the 3D template the conventional control operators B and B^{τ}, and the feedback operators A and A^{τ}. By example, the CNN described in Eq. 1 can be implemented assuming: $z^0 \equiv I$, $z^1 \equiv u$, $z^2 \equiv y$; $D^{00} \equiv D^{11} \equiv D^{20} = 1$ for $(i, j) = (k, l)$ and null elsewhere, $D^{21} \equiv B$,

$D^{22} \equiv A$, D^{01}, D^{02}, D^{10}, D^{12} null; $z_0^0 \equiv I$, $z_0^1 \equiv u$, $z_0^2 \equiv y$; $g^0(x) = x$, $g^1(x) = x$, $g^2(.) \equiv f(.)$.

It is worth noting that in this way D defines not only the strength of connections among the cells of the CNN but also the structure of the CNN itself, thus allowing the higher degree of programmability required.

3. Architectural specification

3.1. Resources and organization

From an architectural point of view two main blocks must be implemented: the storage block in which the $\mathbf{z}$ vectors are stored, and the processing block which updates each vector according to the cloning templates D. The cloning templates are mapped into the processing block through a set of instructions.

Considering linear, space invariant templates, Eq. 2 can be mapped on digital hardware with high efficiency. Evaluation of the current output of a cell, indeed, can be thought as the result of convolution operations with a set of masks and implemented through pipeline schemata. In this respect, each template can be viewed as a 3D array. Since many elements of a cloning template are null, in order to make the implementation more efficient, the 3D template can be *projected* onto a reduced number of 2D masks stored in small RAMs. The need of an efficient, real-time implementation, and the impossibility of realizing a fully-parallel architecture (one processing unit for each cell) suggests the partitioning of computation in domains with a limited number of processing units (PUs). Each PU performs a local weighted sum operation that is implemented with pipeline schemata followed by a non-linear block.

3.2. Implementation

Limits on VLSI technology, power consumption and speed of computation pose some constraints on the number of layers and cells, and on the dimension of the instruction masks. Taking into account the application domain, we consider 12 layers of 64×64 cells and interactions among first neighbors only (i.e. $3 \times 3 \times 12$ cloning template). The non linear function for each layer is implemented by a clipping function with a programmable slope. The architectural configuration of our system is depicted in Fig. 1.

A single port RAM is used to implement the *storage block*. The current/previous output of cellular neural network elements are stored in a RAM with 64 x 64 locations of 96 bits, functionally subdivided in 12 groups of 8 bits to implement 12 layers (L01,..., L12). The random I/O port allows to transfer data from/to the RAM cell array.

The *processing block* is composed of 12 processing units (one for each layer), 12 pairs of 16-bits row-buffer and 12 istruction mask memory. The behavior of each processing unit is controlled by a set of masks whose elements determine the weight sign and magnitude and the number that identifies the layer in which to read the output. Each processing unit performs the weighted sum of three elements addressed in a row of a mask by 16 bits adders. Each processing unit extracts a portion of the datum coming from RAM according to the corresponding element of the mask. The result is shifted, and complemented if requested, and added to the partial sum for the other rows stored in the first 16-bits word row-buffer. All operations occur in pipeline. When a row is completely processed the

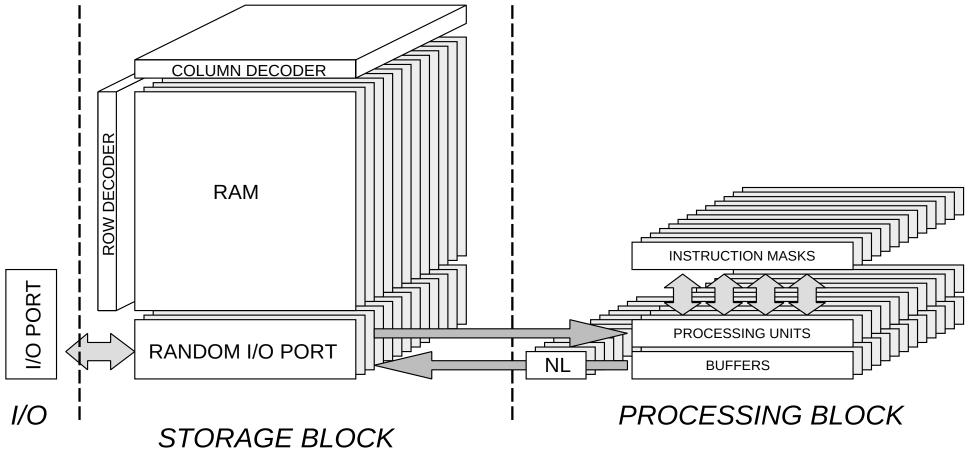
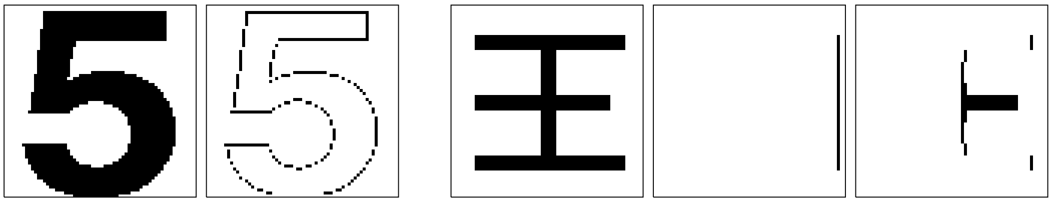


Figure 1: *Overall structure of the architecture, control signals are omitted.*

content of the second row-buffer is moved to the RAM through the programmable non linear blocks that implement a hard limited linear function with variable slope.

4. Simulation results

Edge detection

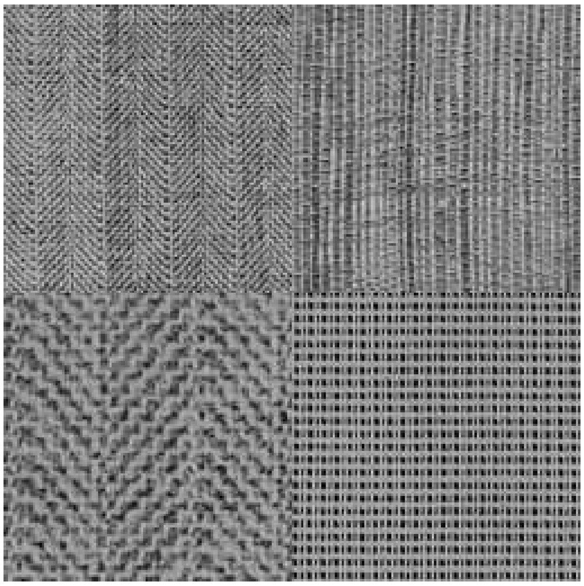


(a) (b) (c) (d) (e)

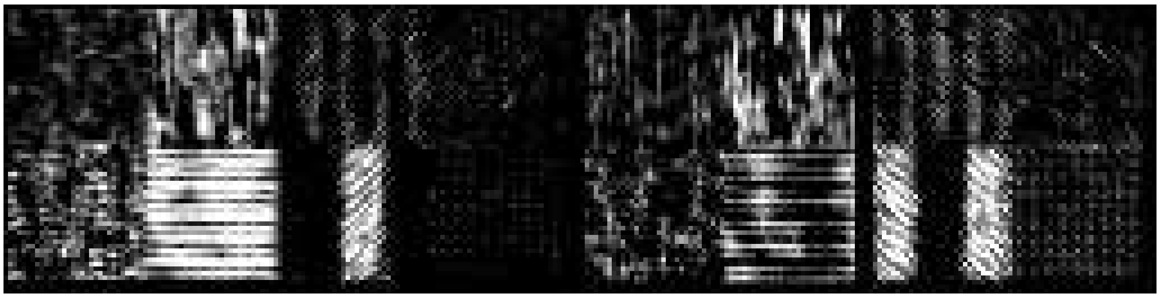
Figure 2: *(a) Test image. (b) Output of the edge detection CNN. In (d) is the result of the elaboration of (c) with a connected component detection CNN described only by an A template. In (e) is the result of the elaboration of (d) with the same A operator together with a delay-type operator A^τ with $\tau = 3$. A^τ is mapped on the architecture by considering three additional layers in which the outputs of the previous ones are copied at each step, realizing a memory of the last four output values.*

Many cloning templates for edge detection have been presented [2]. In Figs. 2(a-b) is shown the result of the implementation of a 3×3 cloning template A with circular symmetry (2 in the middle, -0.25 the neighbors). This operator is mapped on the architecture according to the example in Introduction.

Connected component detection To show the capability of implementing delay-type cloning templates, we present in Figs. 2(c-e) the results of the simulation of the CNN proposed in [3] for connected component detection (see caption). The results completely match the expected results.



(a)



(b)

Figure 3: *(a)* 256×256 *pixel test image, (b) Outputs of the four hypercomplex layers for the four different orientations (* $0°$, $45°$, $90°$ *and* $135°$*).*

Preattentive vision In [4] has been proposed a computational model of the functionalities of the early stages of visual cortex. This model can be formulated in terms of local interactions among different layers of cells. Three classes of cells (simple, complex, hypercomplex) are considered depending on their computation specificity, and each class consists of four layers of cells with different orientation specificity, for a total number of 12 layers. Four layers store the output of simple cells, provided by an external block acting on the input image. Other four layers store the output of complex cells. The last four layers store the output of hypercomplex cells. The values in complex and hypercomplex layers are recurrently updated according to the programmed rules (cloning templates) and the previous values.

We have tested the performances of the architecture on natural textured images (see

Fig. 3). The output row in Fig. 3(b) evidences the same texture-segregation capabilities presented in [4].

5. Conclusion

In this paper a new VLSI architecture has been presented which provides a set of computational primitives organized in 12 layers of 64×64 cells, on which to implement one or more CNNs. This is determined by specifying for each type of implementation a set of *generalized* cloning templates. The same hardware is used in solving all problems that can be tackled by 3×3 linear template CNNs, including those that use sequences of templates and complex interactions among different layers. Simulations show the high efficiency of the implementation. In particular, in preattentive vision applications, taking into account the number of elements per layer, the number of masks per layer, the number of iterations and assuming a clock frequency of 50 MHz, a complete texture segregation of 256 x 256 pixel images is estimated be obtained in about 3msec, allowing to process images at a commercial camera frame rate (25 images/second). The VLSI implementation of this architecture is estimated to require 15 mm x 15 mm of silicon in a 0.5 μm technology.

Acknowledgments This work was supported by *CEC ESPRIT-BRA Project CORMORANT* by *"Programma Nazionale di Ricerca sulle Tecnologie per la Bioelettronica"* of MURST (Italian Ministry of Scientific and Technological Research) and SGS-Thomson Microelectronics; *Progetto Strategico Reti Neurali* of the National Research Council (CNR) of Italy; *Progetti 40% Sistemi Neurali* and *Architetture VLSI* of MURST.

References

[1] L.O. Chua and L. Yang. Cellular neural networks: Theory. *IEEE Trans. on Circuit and Systems*, 35:1257–1272, 1988.

[2] L.O. Chua and C.W. Wu. On the universe of stable cellular neural networks. *International Journal of Circuit Theory and Applications*, 20:497–518, 1992.

[3] T. Roska and L.O. Chua. Cellular neural networks with non-linear and delay-type template elements and non-uniform grids. *International Journal of Circuit Theory and Applications*, 20:469–482, 1992.

[4] G. Indiveri, L. Raffo, S.P. Sabatini, and G.M. Bisio. A neuromorphic architecture for cortical multi-layer integration of early visual tasks. *Machine Vision and Applications*, 8, 1995.