2000 6TH IEEE International Workshop on Cellular Neural Networks and Their Applications Proceedings

Realization of Non-Linear Templates using the CNNUC3 Prototype

G. Liñán, P. Foldesy, A. Rodríguez-Vázquez, S. Espejo and R. Domínguez-Castro
Instituto de Microelectrónica de Sevilla – CNM-CSIC
Edificio CICA-CNM, C/Tarfia s/n, 41012- Sevilla, SPAIN
Phone: +34 95 4239923, Fax: +34 95 4231832, E-mail: linan@imse.cnm.es

ABSTRACT

This paper demonstrates the processing capabilities of a recently designed Analog Programmable Array Processor chip [1] *– CNNUC3 – which follows the Cellular Neural Network Universal Machine computing paradigm* [2][3][4]. *Due to its very advanced features and algorithmic capabilities, this chip has been demonstrated to be able to perform not only linear templates executions, but also to be very adequate for the implementation of non-linear templates by using a decomposition method. This paper focus on the application examples of the execution of non-linear templates with the CNNUC3 prototype. A brief description of the theoretical background is also presented in the paper.*

1. INTRODUCTION[†]

Cellular Neural Networks (CNNs) [2] exhibits outstanding image processing capabilities. They are capable to realize linear as well as nonlinear operations by using either linear or non-linear templates. Linear templates correspond to the case where interaction strengths (weights) are independent of signal values. Non-linear templates correspond to the case where interaction strengths depend on signal values.

Non-linear templates are needed for the realization of important image processing tasks. Techniques to implement non-linearities using integrated circuit primitives are available elsewhere [5][6]. However, incorporating these techniques into CNN chips seriously compromises cell density. Actually, CNN chips reported to now are only capable of establishing linear interactions. Still, piecewise-linear interactions can be emulated by taking advantage of the internal memory and the reconfigurability capabilities of last generation CNN-UM chips. Particularly, the so-called CNNUC3 chip [1] incorporates the following advanced CNN-UM features:

- capability to run algorithms with dozens of operations without external code or data movement;
- capability of storing four gray-scale and four binary images, on a pixel-by-pixel basis;
- capability to sum or subtract gray-scale images, also pixel-by-pixel;
- capability of selecting which cells are going to be processed (so called freezing map);
- capability to combine, pixel-by-pixel, two binary images through any user-selectable logic operation (such as logic "AND", "OR").

This paper experimentally demonstrates that these capabilities can be exploited for the realization of piecewise-linear templates based on the algorithmic methods presented in [7]. The paper is organized as follows; Section 2 establishes a theoretical background about the technique of non-linear templates decomposition. Section 3 describes some applications examples. Some additional comments are provided in Section 3.4. Section 4 deals with the implementation of arbitrary non-linear functions. Finally, the conclusions are presented in Section 5.

2. DECOMPOSITION OF NON-LINEAR TEMPLATES

Implementing a piecewise-linear template by decomposing it into several linear ones is not a new idea. It has been previously studied in [7] at the algorithmic level. However, until now such implementation has never been demonstrated using actual CNN chips. As in [7], his paper deals with the case where only the **B** template is piecewise-linear, and assumes that the template is a 3×3 one.

CNNUC3 employs the so-called FSR CNN model [8] where cell dynamic evolution is given by:

†. This work has been partially funded by ONR-NICOP N68171-98-C-9004, DICTAM IST-1999-19007 and TIC 990826.

$$\tau\frac{dx_{ij}(t)}{dt} = -g[x_{ij}(t)] + \sum_{C(k,l)\in S_r(i,j)} A_{i,j;k,l}\cdot x_{kl}(t) + \sum_{C(k,l)\in S_r(i,j)} B_{i,j;k,l}\cdot u_{kl} + z \tag{1}$$

$$g[x_{ij}(t)] = \begin{cases} m_L & , x_{ij}(t) < -1 \\ 0 & , |x_{ij}(t)| < 1 \\ m_R & , x_{ij}(t) > 1 \end{cases} \qquad \begin{matrix} m_L \to -\infty \\ m_R \to \infty \end{matrix} \tag{2}$$

where entries of the **B** template are linear.

Let us consider now that these entries are non-linear, namely,

$$B_{i,j;k,l}(u_{ij}, u_{kl}) = \Psi(\alpha\cdot u_{ij} + \beta\cdot u_{kl}) \tag{3}$$

where α and β are real numbers. Let us define,

$$\xi = \alpha\cdot u_{ij} + \beta\cdot u_{kl} \tag{4}$$

and assume that $\Psi(\xi)$ is a piecewise-linear function with m breaking points $\{\xi_1, \xi_2, \ldots, \xi_m\}$. Then, as demonstrated in [7], the non-linear template can be decomposed into a sequence of linear ones by using the algorithm described below [7].

- The process starts by selecting the first linear region of the non-linear function. Let us call R_1 this region that is defined by the breaking points ξ_1, ξ_2.
- The next step is to select which are the cells belonging to that region. This calculation is realized by two templates executions and a logic operation (all of them are done on-chip). With the first template, the so called threshold template, we drive to black all those cells having $\xi > \xi_1$, while with the second one, the so called inverse threshold, we drive to black all those cells having $\xi < \xi_2$. Finally a logic AND operation of both results will select those pixels where $\xi_1 < \xi < \xi_2$[‡].

Equations (5), (6), show the threshold and the inverse threshold template[††].

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \beta & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad z = -\xi_1 \tag{5}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} -\beta & 0 & 0 \\ 0 & -\alpha & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad z = \xi_2 \tag{6}$$

- The non-selected cells are "frozen", by using the freezing mask provided by the chip, while in the selected ones the corresponding contribution to the state equation is evaluated and stored as a "bias map" that will be updated (or not) in the next iteration by adding the new result to the one that was previously stored. The updating law for the state variables of the cells that are selected must be given by the equation of a straight line (due to the fact that $\Psi(\xi)$ is linear between each two breaking points) crossing the points ξ_1 and ξ_2. All the points belonging to this line satisfy:

$$\frac{\xi - \xi_1}{\xi_2 - \xi_1} = \frac{\Psi(\xi) - \Psi(\xi_1)}{\Psi(\xi_2) - \Psi(\xi_1)} \tag{7}$$

And from the CNN theory, it can be demonstrated that this relationship is obtained if the following template is executed[‡‡]:

‡. Keep in mind that $\xi = \alpha\cdot u_{ij} + \beta\cdot u_{kl}$ and the subindex kl denotes the cell' neighbours.

††. These are the FSR version of the templates. In order to get the original Chua-Yang template increase by one the self-feedback term.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} k\cdot\beta & 0 & 0 \\ 0 & k\cdot\alpha & 0 \\ 0 & 0 & 0 \end{bmatrix}$$
$$z = \Psi(\xi_1) - k\cdot\xi_1 \tag{8}$$

where,

$$k = \frac{\Psi(\xi_2) - \Psi(\xi_1)}{\xi_2 - \xi_1} \tag{9}$$

- The process continues for the next linear region.
- Finally, a template execution is needed. In this template the feedback term is the same as in the original one defined in (1), the feedforward term is set to zero (modified **B** template), since it has been already calculated, and the offset term is the addition of the original one z, and the ***"bias map"*** that is stored in some memory at the cell.

3. APPLICATION EXAMPLES

3.1 Absolute Value Calculation

Fig.1 shows the corresponding templates and non-linearity. Because the transformation is pixel-wise – **B** template has 1×1 size – the decomposition method can be simplified, precluding accumulation of partial results. Besides, sub-interval selection reduces to estimating sign of the input and can be accomplished by using threshold templates,

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad z = -\xi_1 \tag{10}$$

particularly where $\alpha = 1$, $\beta = 0$ and $\xi_1 = 0$. Since there are two sub-intervals, the number of cell maps is also two. The first one contains black pixels at the cell positions where the corresponding input image pixels are negative, and the second is the opposite. As a special case, the first transformation is equal to inversion and the second one can be avoided in practice since it lets the cells unchanged at their original values. Fig.2. shows the result of the execution of the absolute value calculation on CNNUC3.

$$\mathbf{A} = \begin{bmatrix} 1 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \Psi(\xi) & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad z = 0$$

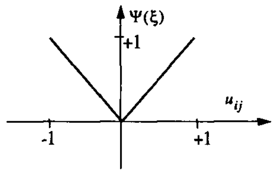


Fig. 1: The absolute value calculation template.

3.2 Gradient Calculation

The gradient template is defined in Fig.3. This template contains eight non-zero entries each of which has two possible intervals. Thus, the decomposition method results into a total of 32 linear template executions and threshold functions.

‡‡. The position of the β coefficient must be rotated in order to perform this operation for each of the neighbours of the cell appearing as a non-linear connection on the original B template. Therefore, each linear region could require up to 16 templates and 8 logic operations to be selected, 8 templates to update the state variable, and 8 templates to perform the addition of the results, that is 32 templates and 8 logic operations.

(a) Input (b) Absolute value

Fig. 2: The absolute value calculation.

$$A = \begin{bmatrix}1\end{bmatrix} \quad B = \begin{bmatrix}\Psi(\xi) & \Psi(\xi) & \Psi(\xi)\\ \Psi(\xi) & 0 & \Psi(\xi)\\ \Psi(\xi) & \Psi(\xi) & \Psi(\xi)\end{bmatrix} \quad z = 0$$

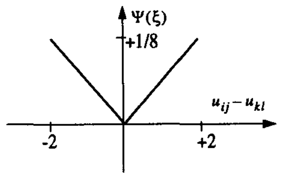


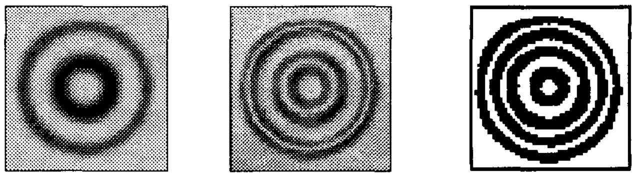
Fig. 3: The Gradient calculation template.

The thresholded gradient operation is similar to the previous. The only difference is a shifting in the offset term,

$$z = -z_{threshold} \tag{11}$$

Thus, the black pixels in the output image correspond to the input locations where the absolute value of the gradient is larger than $z_{threshold}$.

Fig.4 show measurements taken from CNNUC3 which demonstrate the implementation of both gradient and thresholded gradient templates on silicon.



(a) Input (b) Gradient (c) Thresholded gradient

Fig. 4: The Gradient and Thresholded Gradient Calculation Templates.

3.3 Contour Detection on Gray-Scale Images

Fig.5 shows the associated templates and non-linearity. The result of this operation is an image having black pixel at those locations where the corresponding input is larger than some of the neighbours by a certain amount – 0.1 in the case of Fig.5.

Both the number of used mask generating templates and transformation templates are 16 †††. Fig.6(b) shows the result of executing this non-linear template on CNNUC3 with the input image of Fig.6(a).

$$A = \begin{bmatrix}2\end{bmatrix} \quad B = \begin{bmatrix}\Psi(\xi) & \Psi(\xi) & \Psi(\xi)\\ \Psi(\xi) & 0 & \Psi(\xi)\\ \Psi(\xi) & \Psi(\xi) & \Psi(\xi)\end{bmatrix} \quad z = -0.5$$

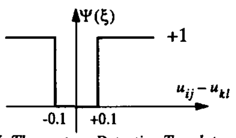
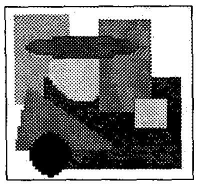


Fig. 5: The contour Detection Template.


(a) Input

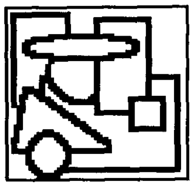

(b) Detected contour

Fig. 6: Contour Detection on Gray-Scale Images.

3.4 Additional Comments

This section contains some additional ideas about decomposition, which are intended to reduce the number of required operations. This reduction is made possible due to some special functions incorporated to the CNNUC3 chip.

- If the number of intervals is only two, the "freezing" masks are opposite. Hence, the calculation of the second mask through template execution can be replaced by a logic operation.
- There are special cases where the general method can be modified in order to get a more efficient decomposition. Specially, when the partial results contains only B&W pixels.In those cases, the generated interval maps contain all the information about the partial results. Moreover, when the final result is the logic sum (operation OR) or logic product (operation AND) of the partial outputs, the final result can be accumulated by the Local Logic Unit (LLU) in a Local Logic Memory (LLM) instead of by using the gray-scale accumulation process in an analog memory.

4. IMPLEMENTATION OF A GENERIC NON-LINEAR FUNCTION

The above described decomposition method can be used for piecewise-linear approximation of general non-linear entries $\Psi(\alpha \cdot u_{ij} + \beta \cdot u_{kl})$ taking advantage of the fact that CNNUC3 is capable to distinguish 20 breaking points within a characteristic curve. Assume that the non-linear coefficient can be approximated by:

$$F(\xi) \cong \sum_{i=0}^{19} \left(\left[F(\xi_i) + \frac{F(\xi_{i+1}) - F(\xi_i)}{\xi_{i+1} - \xi_i} \cdot \xi \right] \cdot [u(\xi_{i+1} - \xi) - u(\xi_i - \xi)] \right) \tag{12}$$

where,

$$u(\xi) = \begin{cases} 1 & \xi \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Then, the decomposition method can be applied to implement the approximated the function.

Another technique consists of approximating the non-linear function into a stair-steps type non-linearity. In this case, the non-linear function is sampled at N points (up to 20 different in CNNUC3), and approximated by:

$$\mathbf{A} = \begin{bmatrix} 1 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad z = 0$$

Fig. 7: The Cubic Template in (15)

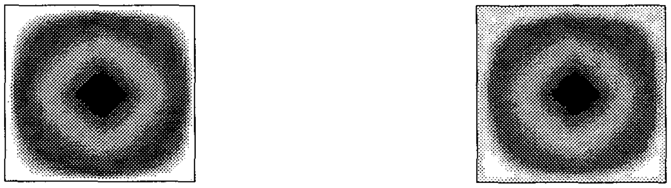
†††. See that the number of required templates is not 64 as it should correspond to the case of having 8 non-linear connections. This is explained by the fact that the linear regions have an infinite or zero slope, and so, the linear transformation defined by (8) is not needed.

$$F(\xi) \cong \sum_{i=0}^{19} F(\xi_i) \cdot [u(\xi_{i+1} - \xi) - u(\xi_i - \xi)] \quad (14)$$

To illustrate this technique we have implemented the following non-linear cubic-type function, considering an approximation containing 12 sampling points.

$$a(u_{ij}) \doteq u_{ij} \cdot (u_{ij} - 0.75) \cdot (u_{ij} + 0.75) \quad (15)$$

Fig.8(a) shows the result of the simulation of this template while Fig.8(b) shows the result provided by the CNNUC3 prototype. The input image has been already displayed in Fig.2 (a). Due to the sampling process, the output image needs post-processing – low-pass filtering also implemented by the chip – for proper signal reconstruction.



(a) Simulated (b) By CNNUC3

Fig. 8: Cubic Template Execution

5. CONCLUSIONS

The executions of non-linear templates define an important application area in the field of image processing. However, previous VLSI CNNs implementations did not provide to the template engineers sufficiently accurate and versatile features to map the non-linear-to-linear existing algorithms. This paper presents experimental evidences about how a wide set of non-linear templates can be executed with by using CNNUC3. We have also briefly outlined the general decomposition method for implementing non-linear-to-linear template transformations in [7].

6. REFERENCES

[1] G. Liñán, S.Espejo, R. Domínguez-Castro and A. Rodríguez-Vázquez., "The CNNUC3: An Analog I/O 64 x 64 CNN Universal Machine Chip Prototype with 7-bit Analog Accuracy". *Proc. of the CNN2000*, submitted.

[2] L.O. Chua and L. Yang, "Cellular Neural Networks: Theory". *IEEE Trans. Circuits and Systems*, Vol. 35, pp. 1257-1272, Oct. 1988.

[3] T. Roska and L.O. Chua, "The CNN Universal Machine: An Analogic Array Computer". *IEEE Trans. Circuits and Systems II*, Vol. 40, pp 163-173, March 1993.

[4] L.O. Chua and T. Roska, "The CNN Paradigm". *IEEE Trans. Circuits and Systems I*, Vol.40, pp.147-156, March 1996.

[5] A. Rodríguez-Vázquez, M. Delgado-Restituto, J.L. Huertas and F. Vidal, "Synthesis and Design of Nonlinear Circuits". Chapter 32 in *The Circuits and Filters Handbook* (W.K. Chen — editor), CRC Press, New-York 1995.

[6] M. Delgado-Restituto, A. Rodríguez-Vázquez and F. Vidal: "Nonlinear Synthesis using ICs". in *Encyclopedia of Electrical and Electronics Engineering* (J.G. Webster — editor), John Wiley & Sons 1999.

[7] L. Kek and A. Zarandy, "Implementation of Large Neighborhood Non-Linear Templates on the CNN Universal Machine". *International Journal of Circuit Theory and Applications*, Vol.26, pp. 551-566, 1998.

[8] S. Espejo, R. Carmona, R. Domínguez-Castro and A. Rodríguez-Vázquez: "A VLSI-Oriented Continuous-Time CNN Model". *International Journal of Circuit Theory and Applications*. Vol 24, pp. 341-356, May-June 1996.