Experimental Demonstration of Real-Time Image-Processing using a VLSI Analog Programmable Array Processor

Gustavo Liñán[a,b], Rafael Domínguez-Castro[a,b], Servando Espejo[a,b], Elisenda Roca[a,b], Peter Foldesy[a,c] and Angel Rodríguez-Vázquez[a,b]

[a]Instituto de Microelectrónica de Sevilla, Centro Nacional de Microelectrónica, Avda. Reina Mercedes s/n, Campus Universidad de Sevilla, E-41012 Sevilla (Spain).

[b]Area de Electrónica, Escuela Superior de Ingenieros, Universidad de Sevilla, Camino de los Descubrimientos s/n, Isla de la Cartuja, E-41092 Sevilla (Spain).

Hungarian Academy of Sciences, SzTAKI, Analog and Neural Computing Laboratory, Kende utca 13-17, H-1111, Budapest (Hungary)

ABSTRACT[i]

This paper describes a full-custom mixed-signal chip which embeds distributed optical signal acquisition, digitally-programmable analog parallel processing, and distributed image memory – cache – on a common silicon substrate. This chip, designed in a 0.5μm CMOS standard technology contains around 1,000,000 transistors, 80% of which operate in analog mode; it is hence one the most complex mixed-signal chip reported to now. Chip functional features are in accordance to the CNN Universal Machine [1] paradigm: cellular, spatial-invariant array architecture; programmable local interactions among cells; randomly-selectable memory of instructions (elementary instructions are defined by specific values of the cell local interactions); random storage/retrieval of intermediate images; capability to complete algorithmic image processing tasks controlled by the user-selected stored instructions and interacting with the cache memory, etc. Thus, as illustrated in this paper, the chip is capable to complete complex spatio-temporal image processing tasks within short computation time (~ 200ns for linear convolutions) and using a low power budget (<1.2W for the complete chip). The internal circuitry of the chip has been designed to operate in robust manner with >7-bit equivalent accuracy in the internal analog operations, which has been confirmed by experimental measurements. Hence, to all practical purposes, processing tasks completed by the chip have the same accuracy than those completed by digital processors preceded by 7-bit digital-to-analog converters for image digitalization. Such 7-bit accuracy is enough for most image processing applications.

The paper briefly describes the chip architecture and focus mostly on presenting experimental evidences of the chip functionality. Multiscale low-pass and high-pass filtering of gray-scale images, analog edges extraction, image segmentation, thresholded gradient detection, mathematical morphology operations, shortest path detection in a labyrinth, skeletonizing, image reconstruction, several non-linear type image processing taks like absolute value calculation or gray-scale gradient detection and real-time motion detection in QCIF video sequences are some of the very interesting applications that have been demonstrated as available when using the prototype.

1. INTRODUCTION

In most modern electronic systems the role of analog is basically limited to that of an *interface* between real-world analog signals and the numbers handled by the core digital processors. In the case of systems intended for processing uni-dimensional signals, interface functions comprise basically impedance matching, amplification, filtering and analog-to-digital conversion. Processing is realized in digital domain by subsequent stages of the processing chain. However, in the case of systems intended for processing large, multi-dimensional, interrelated signals – such as image flows – it may be interesting to incorporate parallel processing at the analog interface in order to reduce the amount of data been transferred to the digital section, and hence to improve efficiency of the whole processing chain.

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

In *Applications of Artificial Neural Networks in Image Processing V*, Nasser M. Nasrabadi, Aggelos K. Katsaggelos, Editors, Proceedings of SPIE Vol. 3962 (2000) • 0277-786X/00/$15.00

Actually, such analog parallel preprocessing is observed already in the retina – the sensory front-end of natural "image processing" systems. It embeds *photoreceptor* cells to acquire images, and *processing cells* to perform non-linear spatial-temporal processing operations on the incoming flow of images through a sequence of layers. Motivated by the efficiency of natural vision systems, universities and companies have focused their efforts on the development of new generations of devices capable of overcoming the drawbacks of traditional ones through the incorporation of distributed parallel processing, and, by making this processing act concurrently with the acquisition of the signal. One possible strategy to achieve that is through flip-chip bonding of separate sensing and processing devices; another possibility is to incorporate the sensory and the processing circuitry on the same semiconductor substrate. "Silicon retinas", "smart-pixel chips" and "focal-plane array-processors" are members of this latter class of vision chips [2]. Their development is expected to have a significant impact in quite diverse scenarios. However, industrial applications demand chips capable of flexible operation, with programmable features and standard interfacing to conventional equipment. A powerful methodological framework for a systematic development of these types of chips is using the paradigm of analog cellular computing whose most advanced implementation is the so-called *CNN universal machine* (CNN-UM) [1].

Main processing components of this architecture are: a) parallel architecture consisting of an array of locally-connected analog processors; b) a means of storing, locally, pixel-by-pixel, the intermediate computation results, and 3) stored on-chip programmability. When implemented as a mixed-signal VLSI chip with embedded distributed optical sensors, image processing rates of trillions of operations per second can be achieved with very small size and low power consumption. Actually, this has been already demonstrated by the authors through a $0.8\mu m$ mixed-signal chip containing 20×22 cells and capable of only binary image processing [3]. This paper describes, from a high level point of view, another chip belonging to the same family which outperforms previous realizations regarding both size, and functionality; among other things this new chip is capable of gray-scale processing. The paper is organized as follows: Section 2 describes the chip from a system point of view. Section 3 provides the information about the specific electronic implementation focussing into the mapping of the cell state equation and into the description of the enhanced functionalities that have been added to each elementary processing unit to increase the final performances of the prototype. Section 4 shows two application examples while Section 5 deals with the implementation of non-linear image processing tasks.

2. CHIP DESCRIPTION

The chip consists basically of a square array of 64×64 identical cells (plus a surrounding ring of border cells which are obviously different), each of which interacts with its nearest neighbors in accordance to the so called Full Signal Range model (FSR[4]).

$$\tau \frac{dx^{ij}(t)}{dt} = -g[x_{ij}(t)] + \sum_{C(k,l) \in S_r(i,j)} A(i,j;k,l) \cdot x_{kl}(t) + \sum_{C(k,l) \in S_r(i,j)} B(i,j;k,l) \cdot u_{kl} + z \qquad (1)$$

where x_{ij} represents the state of the generic $ij-$th cell; x_{kl} represents the state of the cells located in the interaction neighborhood of the $ij-$th cell; $A(i,j;k,l)$, $B(i,j;k,l)$ and z are programmable parameters; and,

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

In addition to the circuitry needed to implement (1), each cell contains the following: a) an optical sensor; b) four randomly-addressable analog memory points plus other four digital memory points (it means that the chip can store four gray-scale images plus four binary images); c) a programmable local logic unit for realization of logical operations among binary images; d) input/output and control circuitry.

For given input image (represented by the matrix of u_{ij} values), either captured by the optical sensors or retrieved from some of the cache memories, and given value zero-state image (represented by the matrix of $x_{ij}(0)$ values, the chip can realize a large variety of basic processing tasks (instructions) through adjustment of the programmable parameters in (1); namely the *bias* term z and the interaction parameters which can be grouped into the so-called *feedback template* and *control template,*

$$\mathbf{A} = [A(i,j;k,l)] = \begin{bmatrix} A_{tl} & A_{tc} & A_{tr} \\ A_{cl} & A_{cc} & A_{cr} \\ A_{bl} & A_{bc} & A_{br} \end{bmatrix} \qquad \mathbf{B} = [B(i,j;k,l)] = \begin{bmatrix} B_{tl} & B_{tc} & B_{tr} \\ B_{cl} & B_{cc} & B_{cr} \\ B_{bl} & B_{bc} & B_{br} \end{bmatrix} \tag{3}$$

where t means top, c center, b bottom, l left and r right. Thus, for instance, A_{cl} denotes the scaling coefficient of the contribution of the center-left cell state variable into equation (1).

Basic chip instructions are hence defined by 19 (actually 20) values. In the chip these values are programmable with a resolution of seven bits plus sign. From an external point of view, images may be analog (gray-scale) or binary (black & white). Internally, pixel values are treated as analog in general, with black & white images having extreme analog levels corresponding to the limits of the linear region. Surrounding border cells are used to establish the necessary spatial boundary conditions for processes. Other miscellaneous functions like analog and digital buffering, control, and I/O tasks, are also included within the border cells.

Figure 1 (a) shows the chip architecture. In addition to the core array processor, the chip includes some global control and programming circuitry located in the periphery of the cell array. This includes memory for 32 arbitrary sets of coefficients, which after being programmed can be arbitrarily selected from the outside. Some other analog values related to the processing circuitry, like the limits of the linear region and others, can also be programmed. Digital to analog (DA) converters generate the analog-program signal levels transmitted to the cell array from the selected set of coefficients. Also, 64 arbitrary sets of 35 digital signals that are used as digital instructions to configure properly the cell in order to perform different task ranging from running a process to configure the cell I/O circuitry. These memories can be randomly addressed from the hosting platform once they have been programmed. Figure 1 (b) shows the chip microphotograph.

The external control is completely digital. The interface has been designed to be easily embedded in conventional digital systems centered around a CPU or a DSP unit. Two bidirectional data-buses, one analog and one digital, are employed for image transferences.

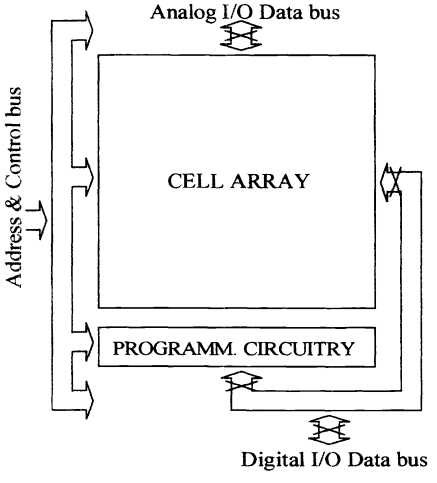


Figure 1. (a) Chip Architecture.

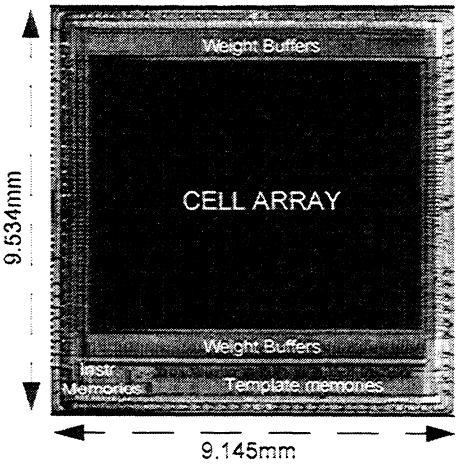


Figure 1. (b) Chip Microphotography

The prototype has been designed and manufactured in a 0.5μm, single-poly, 3-metal layers, CMOS technology. Table 1 shows the most important physical and electrical data of the prototype.

Table 1. Prototype Data.

# of Cells	4096 (64 × 64 Array)
# of Transistors	~1.000.000
# Transistors on the cell	172
Cell Size	120μm x 102.2μm
Cell Density	~82 cells/mm^2
Signal Swing	[0.6, 1.4]V (Programmable.)
Weight Swing	[2.15, 2.95]V (Programmable.)
Time Constant	~1.2μs
Time Constant for Linear Convolutions	~200ns
Spatial Uniformity on the Weight Signals.	7.6-bits
I/O Digital Rate	10MHz
I/O Analog Rate	1MHz
Power Supply	3.3V
Power per cell	250μW
Power Consumption	1.2W (worst case)
# of Templates Memorized	32
# of Instructions	64
Die Size	9145.10μm x 9534μm

3. ARRAY CIRCUITRY

In an electronic implementation, equation (1) [4] must be replaced by a similar one in which variables and parameters have physical meaning. The processing circuitry employed in the prototype represents the state variable x and the cell' input u as voltages across capacitors. Contributions from cell to cell are in the form of current, and therefore, template elements have the dimension of conductances. In this manner, incoming contributions, in the form of currents, are easily added at a common node, and easily integrated in a capacitor, while cell' state and input variables, in the form of voltage, are easily distributed to the synapses. The cell state equation can therefore be written, in integral form, as follows [ii]:

$$v_x^c(t) = v_x^c(0) + \frac{1}{C_x}\int_0^t \left[-I_g(v_x^c(\tau)) + \sum_{d=1}^{9} G_A^d v_x^d(\tau) + \sum_{d=1}^{9} G_B^d v_u^d + I_D \right] d\tau \quad (4)$$

$$I_g(v_x^c) = \begin{cases} m_L & , v_x^c < -v_{sat} \\ 0 & , |v_x^c| < v_{sat} \\ m_R & , v_x^c > v_{sat} \end{cases} \qquad \begin{matrix} m_L \to -\infty \\ m_R \to \infty \end{matrix} \quad (5)$$

The nonlinear dissipative term $I_g(v_x^c)$, sketched in Figure 2, performs an effective voltage limitation of the state variable x within the linear region $[-v_{sat}, v_{sat}]$ by "eliminating" whatever current is necessary. The same signal-range is employed for the input signal u, which is sampled and stored in a capacitor C_u identical to that employed for the integration.

If, a we assign the index 0 to an imaginary neighbor generating the bias, or offset, term I_D, and set its input and state variables to the saturation level v_{sat}

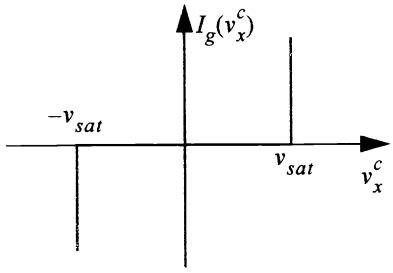


Figure 2. Cell' Non-Linearity.

ii. The index d refers to the eight nearest neighbors of the cell and the cell itself.

$$v_x^0 = v_u^0 = v_{sat} \tag{6}$$

then, (4) can be rewritten as:

$$v_x^c(t) = v_x^c(0) + \frac{1}{C_x}\int_0^t \left[-I_g(v_x^c(\tau)) + \sum_{d=0}^{9} G_A^d v_x^d(\tau) + \sum_{d=0}^{9} G_B^d v_u^d \right] d\tau \tag{7}$$

where the offset term is now expressed as:

$$I_D = (G_A^0 + G_B^0) v_{sat} \tag{8}$$

which is now generated in the same form (i.e., with the same circuit blocks) and with the same reference (saturation) level v_{sat} than any other contribution to the integrand in (4). The use of the saturation level as a reference for the generation of the bias term provides some additional homogeneity, since contributions from saturated cells are proportional to the saturation level. Note also that two additive coefficients are employed to define the offset term thus providing a double range for the offset term.

3.1 Enhanced Functionalities

Some additional functionalities have been incorporated which have specific application in relevant processing functions.

3.1.1 Image Memories

Every cell has the capability of storing four analog (gray-scaled) and four binary (black & white) pixel values. At a system level, this means that the chip allows the storage of a total of eight images. The use of any of these images as input data to some process, or of some specific memory as destination of the result of the processing function, can be realized in a very short time (around 0.1μs). In turn, this results in a significant improvement in the computation time of more complex algorithms requiring iterative template applications, with the possibility of storing intermediate results for bifurcated-flow algorithms. Binary memories employ conventional digital latches, while analog memories relay on "bottom-plate sampling" switched-capacitor stages following the guidelines given in [5].

3.1.2 Local Logic Unit

The local logic unit (LLU) is a programmable boolean gate whose truth table is defined as part of the digital instructions stored in the programming circuitry. It allows a completely parallel realization of arbitrary bit-to-bit logic operations between images stored at two specific binary memories. The resulting image can be down-loaded or stored in any of the four binary memories. Conventional digital circuitry is employed for this purpose.

3.1.3 Freezing Mask

Having a "freezing" mask means that the content of one specific binary image memory can (optionally) be used as a flag which disables the evolution of the marked pixels during processing transients, keeping their state variables time-invariant. The realization of this function requires just a few analog switches.

3.1.4 Global Gates

In many cases it is interesting to find out if some specific image (for instance the outcome of some process) is completely white or completely black, without wasting the time required to download the whole image. The prototype incorporates two global gates, one NAND and one NOR, to perform these logic operations over the pixel values of one specific binary memory. With this functionality, the time required to check if some image is completely black or white is about 3μs.

3.1.5 Optical Input

In many real-life high-speed applications, the information to be processed by the network is an image that is available in optical form while the output contains only a few details extracted from the input. In these situations, the readout process is extremely simplified and hence speeded up. However, the input image is always a complete frame and therefore, the time needed to transfer the image to the array can constitute a bottleneck. In those cases, the capability of combining the sensory and the processing planes, provides a dramatic increase of the system performances, since it produces systems that do not only exploit the advantages of the fully parallel processing but also those of the fully parallel image acquisition that are provided by a matrix of photosensors merged with that of processors. Accordingly to this, the chip incorporates a photosensing device within each cell that allows the acquisition of images that are directly projected over the silicon surface. The sensing scheme is based on the integration, in the capacitor of any of the analog image memory, of the current that is generated by diffusion-substrate photodiode.

4. APPLICATION EXAMPLES

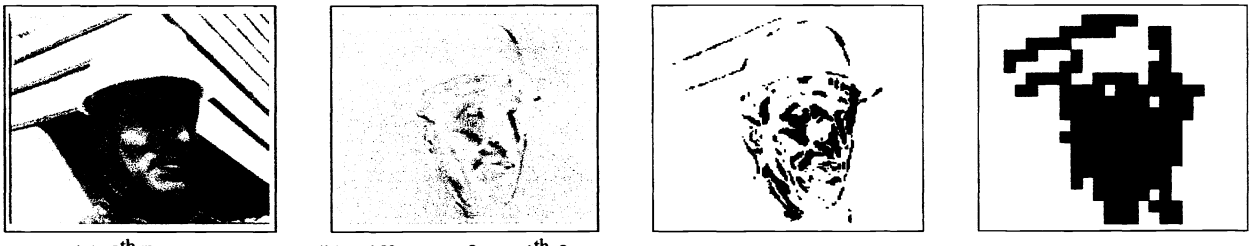
4.1 Movement Detection

One of the key-ideas when coding a video sequence is the interframe motion compensation [6]. This strategy codifies the first image on the sequence as an independent item while the remaining frames are codified as a predicted frame and the prediction error. If the motion of the objects in the scene is small enough and mostly translational, this scheme comprises the video sequence in a very efficient manner.

Out from all tasks needed for MPEG-1,2 [7] video compression, block motion estimation is crucial in terms of time consumption. Block motion estimation is based on systematic block-matching search. This search is based on finding the candidate block, considering a Mean Absolute Difference criterion (MAD) which is the most similar to the predicted one. In this subsection, we will show, only with example purpose, how the first low-level, but very intensive tasks, are performed by the new prototype. Namely, the extraction of blocks that are changed.

A first step in the block motion estimation, is the detection of the changes among two consecutive frames. During this task, image substraction, low-pass filtering, and double-thresholding are performed on chip (see Figure 3). The time needed to accomplish this first step is $110\mu s$, that is, the processing time is 25ns per pixel, while the total number of operations per pixel is 1 substraction (eight bits of resolution) + 1 heat diffusion ($N \times$ (10 multiplications+ 10 additions) where N is the number of iterations needed to achieve the desired equilibrium point) + 1 logic operation.

The second step is to mark the changed blocks. The goal of this step is to mark the blocks where at least one pixel has changed. This is done by using the fixed state map and black wave propagation. The wave fills up the block-sized areas where a black pixel exists, while the fixed state map stops the wave evolution at the border of each block. The required time is $26\mu s$, thus providing a processing time per pixel of about 6ns .The result of this step can be seen also in Figure 3.



(a) 5th Frame (b) Difference from 4th frame (c) Heat Diffusion (Low Pass Filtering) + Double thresholding+ OR (d) Marked blocks

Figure 3. Detection of the changes in a video sequence.

4.2 Impulse Noise Removal

The problem of removing noisy pixels from an input image can be formulated in a general way as follows. Given a set of pixel intensities $I(i,j)$ where $1 \le i \le N_{Rows}$, $1 \le j \le N_{Cols}$ and a maximum rate of change among pixels R, all those pixels $p_{i,j}$ where the local rate of change r is larger than R should be substituted by the local averaging M of the intensity of its neighbors, where:

$$M = \frac{\sum_{m=-1}^{m=1} \sum_{k=-1}^{k=1} I(i+k, j+m)}{8} \qquad [m,k] \neq [0,0] \tag{9}$$

The flow chart of the algorithm that we are using in order to implement this function is shown in Figure 4 and it works as follows:

- First of all, the user has to define what is noise, that is, the user has to determine which is the maximum value for the difference among two pixels that makes a pixel to be considered as correct instead of as noisy[iii].
- Second, the local maxima and minima pixels are detected. During this calculation, only those pixels where the difference between its intensity and that of any of its neighbors is larger (or lower, it depends on whether we look for maxima or minima locations) than the previously defined threshold will turn to black.
- The result for the maxima and minima detections are aggregated by using a logic *NOR* via the Local Logic Unit. After this last operation, the resulting image contains white dots only where a local maxima or minima was detected.
- This last image is sent to the fixed-state image memory, and a heat diffusion (averaging) process is performed over the input image. Due to fixed state function, that is included within each cell, only those pixels that are not marked (black) will be processed. Since those pixels are considered as noise contributions to the image, the are replaced by the local averaging of their neighbors.

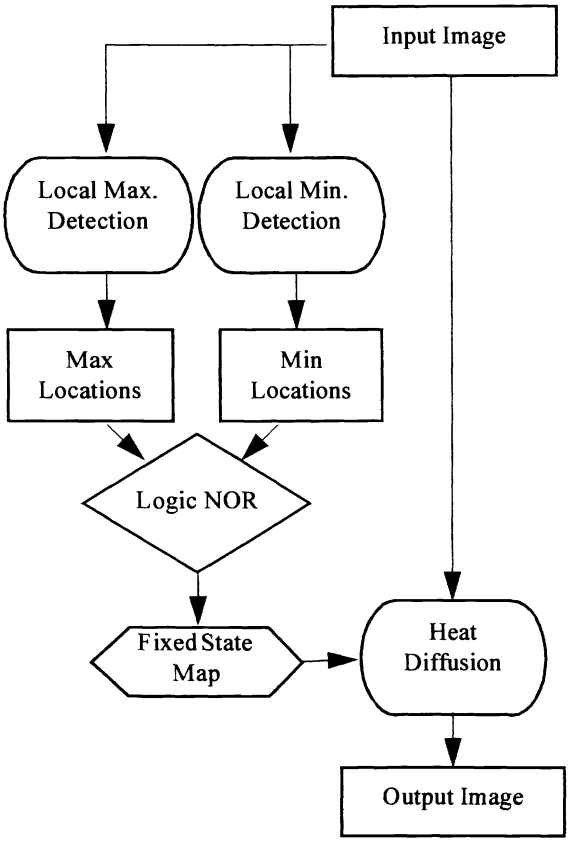


Figure 4. Algorithm for Impulse Noise Removal.

Figure 4 shows the flow chart of this algorithm while Figure 5 shows an example of application to a noisy image. It can be observed how noisy pixels at the input are replaced by the averaging of the intensities of their nearest neighbors thus increasing significantly the quality of the image. The input image is an standard QCIF (176×144) size image (the image is cut into nine overlapped pieces that are processed one after the other) and the total time consumption (including the time for 9 loading and 9 downloading process) is slightly below 5.2ms thus yielding a pixel rate of 200ns. Among this time, the processing takes 23ns. During this 23ns each pixel performs 18 comparisons, 1 logic operation and a heat diffusion ($N \times$ (10 multiplications+ 10 additions)) where N is the number of iterations needed to achieve the desired equilibrium point.

iii. We consider that noisy pixels are always separated by correct pixels (thus distinguishing noise from sharp edges)

(a) Noisy Input Image

(b) Output

Figure 5. The Impulse Noise Removal Application

5. IMPLEMENTATION OF NON-LINEAR INTERACTIONS

5.1 Theoretical Background

Non-linear cell interactions can be realized by decomposing them into a sequence of several linear ones [8]. We will deal with the decomposition of templates where only the **B** term is a non-linear function. Furthermore, we will assume that the non-linearities appearing on the feedforward term are piecewise-linear functions and that the input image is time invariant.

Let us suppose that the non-linear piecewise function can be expressed as:

$$\Psi(\xi) = \Psi(\alpha \cdot u_{ij} + \beta \cdot u_{kl}) \tag{10}$$

where α and β are real numbers, and that the linear regions are defined by a set of m breaking points $\xi_1, \xi_2, \ldots, \xi_m$. In that case, that is also the most common in practice, the non-linear template can be decomposed into a sequence of linear template executions. The algorithm[8] runs as follows:

•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 $\xi_1 < \xi < \xi_2$ [iv].

•Equations (11), (12), show the threshold and the inverse threshold template[v].

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \beta & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad z = -\xi_1 \tag{11}$$

---

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

v. These are the FSR version of the templates [4].

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} -\beta & 0 & 0 \\ 0 & -\alpha & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad z = \xi_2 \tag{12}$$

•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:

$$\frac{\xi - \xi_1}{\xi_2 - \xi_1} = \frac{\Psi(\xi) - \Psi(\xi_1)}{\Psi(\xi_2) - \Psi(\xi_1)} \tag{13}$$

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

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} k \cdot \beta & 0 & 0 \\ 0 & k \cdot \alpha & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{14}$$

$$z = \Psi(\xi_1) - k \cdot \xi_1$$

where,

$$k = \frac{\Psi(\xi_2) - \Psi(\xi_1)}{\xi_2 - \xi_1} \tag{15}$$

•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, 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 on the cell.

5.1.1 Absolute Value Calculation

In this subsection we consider only pixel-wise transformations, or with other words, **B** templates with the size of $1x1$.

As a consequence of missing neighbor connections the decomposition method can be simplified, avoiding the accumulation of the partial results. Moreover, the selection of cells belonging a given interval is done by the two threshold templates, which also contain only central elements (where $\alpha = 1$, $\beta = 0$ and $\xi_1 = 0$):

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad z = -\xi_1 \tag{16}$$

---

vi. The position of the β coefficient must be rotated in order to perform this operation for each of the neighbors 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.

As an example we show how the absolute value can be calculated. The used operation and template is shown in Figure 6.Since there are two intervals, the positive and negative valued cells, there are two cell maps. The first one contains black

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

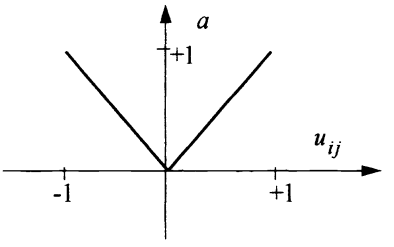
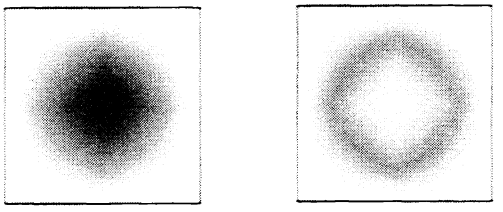


Figure 6. The absolute value calculation template.

pixels at the cell positions where the input image contained negative values and the second is the opposite of it. As a special case, the first transformation is equal to inversion and the second one practically can be avoided (since it lets the cells unchanged at their original values). Figure 7. shows the result of the execution of the absolute value calculation.



(a) Input (b) Output

Figure 7. The absolute value calculation.

5.1.2 Gradient Calculation

The second example is the calculation of the gradient and the thresholded gradient.

The gradient template is defined in Figure 8.

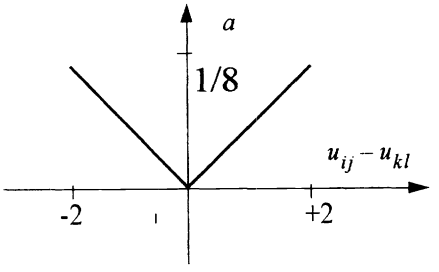
The template contains eight neighboring connections that can belong to two intervals. After the usage of the decomposition method the total number of linear template executions and threshold functions is 32 .

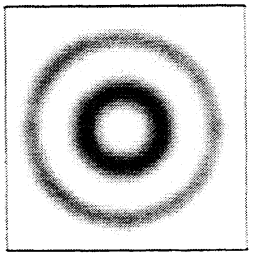
The thresholded gradient operation differs from the gradient calculations in the values of the modified **B** template.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad z = z_{threshold} \tag{17}$$

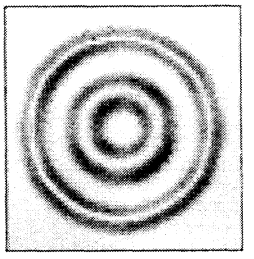
Execution examples can be seen in Figure 8 (b) and (c).

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

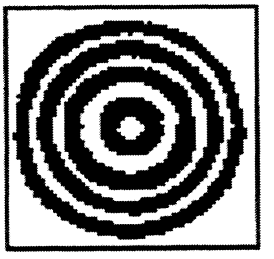




(a) Input



(b) Gradient



(c) Thresholded gradient

Figure 8. The gradient template and application examples.

6. REFERENCES.

1. 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.
2. C. Koch, H. Li (Eds.), *Vision Chips, Implementing Vision Algorithms with Analog VLSI Circuits*, IEEE Press, 1995. ISBN: 0-8186-6492-4
3. R. Domínguez-Castro, S. Espejo, A. Rodríguez-Vázquez, R. A. Carmona, P. Foldesy, A. Zarandy, P. Szolgay, T. Sziranyi and T. Roska, "A 0.8μm CMOS Two-Dimensional Programmable Mixed-Signal Focal-Plane Array Processor with On-Chip Binary Imaging and Instructions Storage". *IEEE Journal of Solid-State Circuits*, Vol 32, pp 1013-1026, July 1997.
4. 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.
5. R. Carmona, S. Espejo, R. Dominguez-Castro, A. Rodriguez-Vazquez, T. Roska, T. Kozek, L.O. Chua, "A 0.5 μm CMOS CNN Analog Random Access Memory Chip for Massive Image Processing". *Proc. of 5th IEEE Int. Workshops on Cellular Neural Networks and their Applications*, pp. 271-276, London, April 1998.
6. A. Puri, R. Aranvid and B. G. Haskell, "Adaptive Frame/Field Motion Compensated Video Coding". *Signal Processing Image Commun.* Vol. 1-5, pp 39-58, February 1993.
7. ITU-T Recommendation H.261, *Video codec for audiovisual services at* p*64 kbits/sec.
8. 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.