Toward CNN Chip-Specific Robustness

Samuel Xavier-de-Souza, Müştak E. Yalçın, *Student Member, IEEE*, Johan A. K. Suykens, *Member, IEEE*, and Joos Vandewalle, *Fellow, IEEE*

***Abstract*—The promising potential of cellular neural networks (CNN) has resulted in the development of several template design methods. The CNN universal machine (CNN-UM), a programmable CNN, has made it possible to create image-processing algorithms that run on this platform. However, very large-scale integration implementations of CNN-UMs presented parameter deviations that do not occur on ideal CNN structures. Consequently, new design methods were developed aiming at more robust templates. Although these new templates were indeed more robust, erroneous behavior can still be observed. An alternative for chip-independent robustness is chip-specific optimization, where the template is targeted to an individual chip. This paper describes a solution proposal in this sense to automatically tune templates in order to make the chip react as an ideal CNN structure. The approach uses measurements of actual CNN-UM chips as part of the cost function for a global optimization method to find an optimal template given an initial approximation. Further improvements are achieved by generating chip-specific robust templates by doing a search for the best template among the optimal ones. The tuned templates are therefore customized versions that are expected to be much less sensitive to imperfections on the operation of CNN-UM chips. Results are presented for the binary and grayscale cases, including the case of grayscale output. It is expected that as this technique matures, it will give CNN-UM chips enough reliability to compete with digital systems in terms of robustness in addition to advantages of speed.**

***Index Terms*—Chip-specific robustness, template optimization, very large-scale integration (VLSI) cellular neural network (CNN) implementation.**

I. Introduction

CELLULAR neural networks (CNNs) [1], are analog processor arrays which are locally interconnected where the state of each processor cell obeys the dynamics

$$\frac{d}{dt}x_{i,j}(t) = -x_{i,j}(t) + \sum_{k,l\in N(c)} Ay_{k,l} + \sum_{k,l\in N(c)} Bu_{k,l} + z_{i,j} \tag{1}$$

where

$$y(x) = \frac{1}{2}(|x+1| - |x-1|)$$

Manuscript received June 27, 2003; revised January 2, 2004. This work was supported in part by the Belgian Programme on Interuniversity Poles of Attraction, Belgian State Prime Minister's Office for Science, Technology and Culture under Grant IUAP P4-02, Grant IUAP P4-24, and Grant IUAP-V, in part by the Concerted Action Project (MEFISTO) of the Flemish Community, in part by the Fund for Scientific Research (FWO) project, and in part by CE Project IST-1999-19007 (DICTAM). This paper was recommended by Guest Editor P. Arena.

The authors are with the Department of Electrical Engineering, Katholieke Universiteit Leuven, B-3001 Leuven, Belgium (e-mail: Samuel.Xavierdesouza@esat.kuleuven.ac.be; mustak.yalcin@esat.kuleuven.ac.be; johan.suykens@esat.kuleuven.ac.be; joos.vandewalle@esat.kuleuven. ac.be).

Digital Object Identifier 10.1109/TCSI.2004.827618

for $u_{k,l}$ and $y_{k,l}$ being, respectively, the input and output of the cells in the neighborhood $N(c)$ of a given cell. The space-invariant local interconnections A, B, and z compose the small set of free parameters that uniquely determine the total array behavior. These parameters are commonly called *templates*. Because of its common arrangement in a regular two-dimensional grid, CNNs are very suitable for image processing. Few years after its invention, image processing applications have been executed on a CNN platform thanks to another invention: a programmable CNN, the so-called CNN universal machine (CNN-UM) [2], which was the first algorithmically programmable array computer with real-time and supercomputer power in a single chip.

At the time CNN and CNN-UM were invented (1988 and 1992, respectively), several design and learning methods were developed to generate templates for execution of different tasks [3]–[5]. These methods were based on ideal CNN structures and the resulting templates could perform a wide range of operations when executed on simulators implemented on digital computers. The CNN Software Library [6] contains a nice collection of these templates.

Actually, with the analog very large-scale integration (VLSI) technology [7], [8], a considerably larger CNN-UM can be implemented in a single chip [9], [10]. These chips can perform image-processing tasks with extremely high throughput data rates: in the order of tera-operations per second [11]. Such a performance can make CNN chips very suitable for a wide range of image-processing tasks, especially for real-time applications [12]. Nevertheless, the chip parameters are slightly different from those ideal ones used on simulators. The causes of this are mainly noise in electrical components of cells as well as imperfections in the fabrication process resulting sometimes in erroneous behavior of some cells. These differences between ideal structures and real chips prevent the chips to react in the same way as the simulators and cause sometimes serious differences between simulator results and chip results. Tests of the early templates (developed for simulators) on VLSI chips proved that many templates worked incorrectly [13]. Consequently, new *chip-independent* robust template-design methods were developed [14]–[17] for the purpose of generating templates that are more tolerant against inherent parameter deviations and noise in VLSI analog implementations without taking into account specific characteristics of an individual chip.

However, the degree of robustness for different operations are obviously not the same. While templates with a high degree of robustness allow a correct chip response for the given operation, other templates with lower robustness still cause erroneous operation in chip implementations. The robustness value, or degree of robustness, of a template gives a measure of how

1057-7122/04$20.00 © 2004 IEEE

tolerant the template values are to parameter deviations. When the deviation is larger than the respective tolerance range of the given chip parameter, the template does not react properly and produces unexpected and undesirable results. Therefore, even the most chip-independent robust templates will not guarantee a fully correct behavior for a given CNN chip unless its robustness is sufficiently high to overlook the parameter deviations of the chip [13]. Nevertheless, for some applications, one can manually and empirically attempt to tune the templates of a given chip and attempt to make it respond correctly for a given task. Yet, even if the goal is achieved, there is no guarantee that a final template will work for other similar chips. In addition, manually tuning each template used in an application might be a long and very tedious task.

In contrast with the other types of template generation methods, like *design* and *learning* that ignore specific chip characteristics, the work described in [18] proposes a method for template *optimization* and *decomposition* that uses measurements of a specific chip and, therefore, takes into account its inherent characteristics. The approach intends to eliminate, or at least minimize, the errors of actual CNN chips on given uncoupled operations by combining gradient descent optimization with decomposition of ideal CNN uncoupled templates and modifying template values using measurements of real CNN-UM chips.

This paper describes a chip-specific solution proposal to automatically tune CNN templates for correcting erroneous behavior of CNN-UM chips and make them respond in the same way as a simulator. This approach, like in [18], also uses measurements of actual CNN-UM chips, in particular the ACE4K and ACE16k chips [9], [10], but is not restricted to uncoupled templates. The chip measurements are used as part of the cost function for the adaptive simulated annealing (ASA) algorithm [19], [20] to find an optimal template given an initial approximation, e.g., a chip-independent robust template. Additionally, the concept of chip-specific robustness is explained. This concept considers a search for the best template in terms of robustness from a range of optimal templates. The resulting templates are therefore customized versions that are presumed to be insensitive, or at least less sensitive, to imperfections on the manufacturing process and other reasons of erroneous behavior of CNN-UM chips.

This paper is organized as follows. The Section II gives more insights on the erroneous behavior of CNN-UM chips and template robustness. Section III explains in more detail the process of tuning the templates using the ASA algorithm. Section IV explains the concept of chip-specific robustness. Section V presents experimental results.

II. Erroneous Chip Behavior and Template Robustness

The erroneous behavior observed in VLSI implementations of CNN-UM may be caused by a combination of reasons that includes manufacturing process variations and environmental effects (such as temperature and noise). Although adaptive techniques have been employed in CNN-UM chip implementations to ensure accurate external control and system robustness against parameter variations, analog VLSI implementations

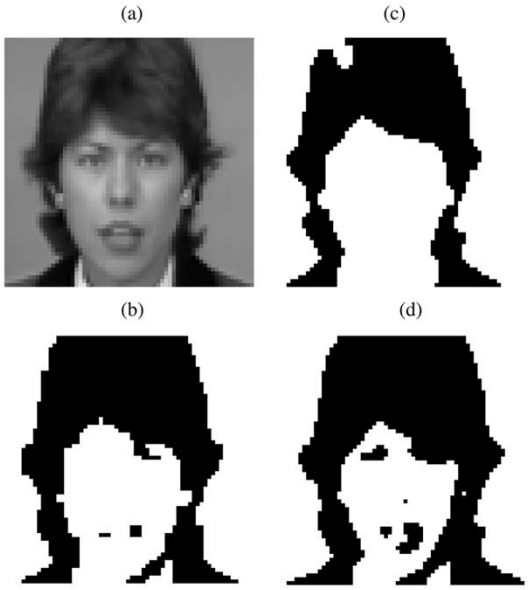

Fig. 1. For the same input image (a), initial state, and template values, the same chip (an ACE4k) can produce different outputs (b), (c), and (d). These results were obtained for subsequent runs, what make temperature and noise interferences mostly stable and therefore the different outputs are assumed to be due to imperfect loading from off-chip to on-chip memories.

can only guarantee a rough accuracy (5%–10%) in relation to ideal parameter values and, also, template parameters have a discrete range of implementable values (about 7 bits for the actual chips) [8]. In addition to errors caused by manufacturing process variations and environmental effects, some templates developed for use in ideal CNN structures, like the AVERTRSH (average with binary output) [6], can even produce different erroneous results for runs with the same input and initial conditions for a given chip (see Fig. 1). These so far unavoidable and undesirable features make CNN-UM chips loose reliability for certain template operations. Fig. 1 clearly shows that the source of errors is not only due to manufacturing imperfections but also susceptibility to post-manufacturing interference. The reasons then could be noise in the components or temperature variation. However, another reason for the inconsistent behavior in Fig. 1 may also be imperfect or noisy loading of the input and initial state from off-chip to on-chip memory prior to an operation, which may also contribute to the overall undesirable chip behavior. According to these assumptions, the main reasons for the erroneous behavior observed in CNN-UM chips may be summarized as follows:

- parameter variation introduced during fabrication process;
- noise in the electrical components of the cells;
- imperfect or noisy loading of the input and initial state from off-chip to on-chip memory prior to operation;
- temperature variation.

Methods that consider the design of robust templates have been extensively developed in order to avoid or minimize the effects of these obstacles for correct operation of analog VLSI implementations of CNN-UMs [21].

There are several definitions of robustness [14], [21] that, in general, define a measure for susceptibility of CNN templates to modifications in their values while still producing the correct

output. Basically, robust templates are expected to have their values in the middle of the correct operation interval so that tiny variations in this values are still within the interval. Methods for designing or learning of robust templates generate templates operations with different degrees of robustness. The most robust operations can often be employed in chip implementations with minimal disturbance of the correct output and therefore qualifying these methods as sufficiently good for these template operations.

Despite these advantages, methods for generation of robust templates (or chip-independent methods) can not avoid erroneous behavior on CNN-UM analog VLSI implementations for other less robust template operations. Nonetheless, making use of information (measurements) of a specific chip, one can achieve better results for this chip when these templates are tuned (or calibrated) [22].

III. Template Tuning

The methodology described in this paper is composed of two optimization steps. In the first step, a search for the optimal tuning of a given template operation is performed. The resulting template is then used as an initial point for the second step, which has the objective of finding a *robust* optimum for the template values.

Both searches for better templates in the two phases are done using the ASA algorithm [19], [20]. The choice for a global optimization method that, in contrast with common local optimization methods, does not need information on the gradient of the cost function has some advantages: optimization of coupled templates is no longer a restriction as it is in [18]; global optimization can handle nondifferentiable problems and/or with many local minima. Besides, ASA is known as a robust and fast method to search for a global optimum in nonlinear complex problems with multiple local optima, which may reflect well the characteristics of CNN cost functions.

In the first phase of the tuning, optimization of templates is performed by a search around the parameter values of the initial template using the ASA algorithm. The goal is therefore to find a modified version of the initial template that allows a correct template operation. A similar and also chip-specific approach that uses a local optimization method based on gradient descent was proposed in [18] to optimize uncoupled templates. The convergence of the gradient method was reported to be fast but an error-free template was not certain to be found, when then template decomposition was recursively applied for completion of the method. The advantage of a gradient descent method is its speed of convergence. However, it can lead the solution to a poor local minimum and force unnecessary decomposition into different templates. Besides, one of the most interesting features of CNNs, global interaction, is ignored since the propagating templates can not be optimized due to the lack of proper error derivative including the feedback connections. On the other hand, optimizing with a global method eliminates the restrictions to the template operation but the convergence is not as fast as in a local optimization method.

For a fully correct optimization, a very important step is also the choice of the training set, which is composed by a set of triplets θ containing the input u, the initial state x, and the desired output y^d. Each individual element of a triplet entry has, in this approach, its values ranging from 0 to 1. In [18], the importance of this step is discussed and a good method to compose training sets for uncoupled operations is proposed. However, for more general and coupled templates, sample images and random images suitable for the given operations have to be considered.

The cost function chosen for the first phase of the tuning is a normalized version of the same cost function used in [3] for learning purposes

$$g(p,\theta)=\frac{1}{\sqrt{k}}\sqrt{\sum_{i=1}^{k}\left(y_i^d-y_i(\infty)\right)^2} \tag{2}$$

where p denotes the parameter vector, i.e., the probing template, θ is the current training triplet, k is the number of cells, y_i^d is the value of the ith pixel of the desired output, and $y_i(\infty)$ is the corresponding value of the steady-state output, whose values are acquired from direct chip measurements. Hence, the cost function $g(p,\theta)$ of the probe template p for the input and initial state contained in the triplet θ gives the rms value of the distance between the desired output vector y^d and the steady-state output $y(\infty)$. The objective of the ASA algorithm is, therefore, to minimize $g(p,\theta)$ given an initial template p_{init}.

The ASA algorithm is a very robust, yet flexible, optimization method that allows different parameters to have distinct finite ranges. Each parameter also have distinct sensitivities. These are measured by the immediate gradient at a local minimum and are dependent on the annealing time. The probing parameters p_i, with $i=(1,\ldots,D)$, are randomly generated from the cumulative probability distribution

$$P_i^T(\gamma_i)=\frac{1}{2}+\frac{\operatorname{sgn}(\gamma_i)}{2}\frac{\ln\left(1+\frac{|\gamma_i|}{T_i}\right)}{1+\frac{1}{T_i}}$$

where

$$\gamma_i=\operatorname{sgn}\left(u_i-\frac{1}{2}\right)T_i\left[\left(1+\frac{1}{T_i}\right)^{|2u_i-1|}-1\right]$$

is generated from the uniform distribution $u_i\in U[0,1]$, with $p_i^{j+1}=p_i^j+\gamma_i(p_{\max,i}-p_{\min,i})$. The annealing temperatures are scheduled according to

$$T^j=T^0\exp\left(-cj^{\frac{1}{D}}\right) \tag{3}$$

where T^0 represents the initial temperatures, and c is an ASA adjust parameter. The acceptance temperature is analogously scheduled at each accepted point. The temperatures are re-annealed after a given number of accepted points, e.g., 100, according to the formula

$$T_{\text{new}}^j=T^j\left(\frac{s_{\max}}{s}\right) \tag{4}$$

where the sensitivities $s=(\partial g/\partial p_i)$ are calculated at the most current minimum value of the cost function (2). The indexes j are updated isolating them from (3) and substituting (4).

Imposing an initial approximation p_{init} seems to be of less importance for a global optimization method like ASA. However, for this approach the approximation is used to set

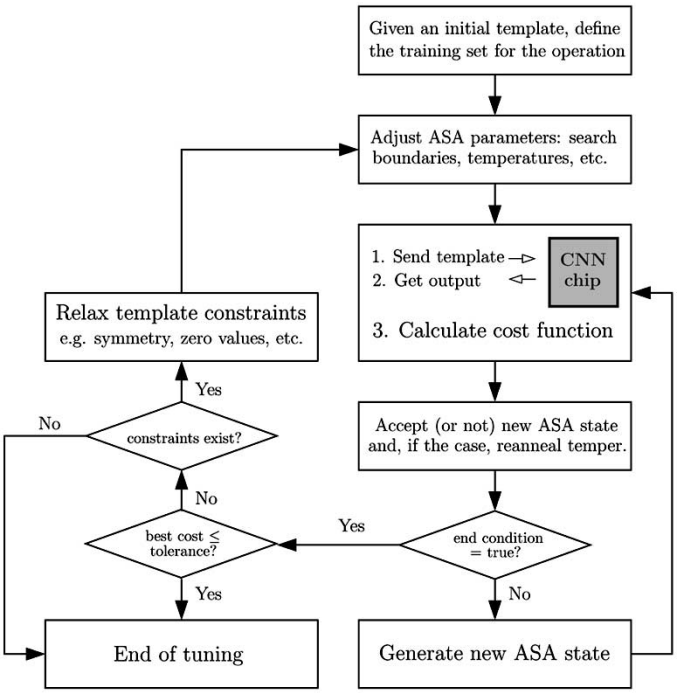

Fig. 2. Outline of the tuning with ASA.

the boundaries of the search since the objective here is tuning and not learning, where the whole parameter range would be used instead. Specifically, the boundaries for the search are $p_{\min,i} = p_{\text{init},i} - b$ for the lower bound and $p_{\max,i} = p_{\text{init},i} + b$ for the upper bound, where i is the index of each template parameter and b is a small value. Observe that here two assumptions are made: the initial template is assumed to be a fully correct working template on a simulator; and the parameter deviations that disturb its values on the chip are assumed to be smaller than b. Narrow boundaries for the ASA search decreases duration of the optimization and allows the use of smaller annealing temperatures, resulting in a much more efficient optimization. Besides, these boundaries are not rigorously strict since a mechanism of self-adjustment may be easily introduced with no significant loss for the algorithm, e.g., if any of the boundaries is very close to and/or is constantly bounding its respective component it may be shortly extended.

Another advantage of using finite boundaries, besides modeling physical limitations, is the role played by recursive optimization runs with relaxation of constraints. Applying constraints (such as symmetry, imposing zero values, or dependence between values) to the template under optimization shrinks the search space and allows a faster search. The result of this search is then further reused in another search with less constraints (more parameters) and narrower boundaries (smaller parameter search space). This recursion is applied until no more constraints are left. The first searches, with more constrained templates and broader boundaries, serve to roughly localize the global optimum in the search space for further refinements using less constrained templates. The outline of the method can be visualized in Fig. 2.

Once the training set and the search boundaries are defined, the optimization can be performed. The procedure is finished and considered successful when the cost function becomes smaller than a certain end condition value or until when the annealing temperatures decrease through a given limit of influence on the algorithm. If the cost for the best template is zero or smaller than a tolerance value then the tuning is finished. Otherwise, if the solution is restricted by any constraint, such as symmetry or imposed zero values, they are relaxed to the next level and another ASA optimization is initiated. The result of the last optimization is considered to be an optimal template. However, this solution might not be unique, i.e., the optimum can be located inside a region of multiple optima. Section IV describes a solution proposal to search among a set of optimal templates in order to find the *best* template in terms of *robustness* for whenever there exists the possibility of multiple optima.

IV. Chip-Specific Robustness

In contrast with design and learning, template optimization (or template tuning) does not intend to create new CNN templates but to improve existent ones. The aim of the optimization involves normally either robustness improvement [23] or error minimization for a specific CNN chip implementation [18]. In this approach, both goals are pursued in combination. In order to increase the error minimization, robustness improvement is employed to the error minimized template within the scope of the given chip. This procedure produces templates that are expected to be optimal, in the sense of error minimization, and robust, with relation to variations on the optimal template values due to post-manufacturing interference. The concepts of robustness and optimality here are restricted to the specific chip, neither the term optimal nor robust can be employed with the tuned template for use in a different chip without proper repetition of the whole procedure.

As discussed in Section II, robust template-generation methods place the template values in the middle of a correct operation interval but these templates still present errors in VLSI implementations. It is obvious then that the nominal values of these robust templates are shifted in the chip to the outside of the correct operation interval. If the nominal values are shifted and a compensation can be found for a given chip, so might the interval, i.e., the same interval that exists in simulations should exist also in the chip but it is shifted together with the nominal values. Therefore, it is surely worthwhile to develop methodologies toward chip-specific robustness. *chip-specific robustness* is, therefore, the robustness associated to the degree of susceptibility of a given CNN template, that had been tuned for a specific chip, to variations in its values caused by any post-manufacturing interference in the chip. This methodology can not only correct errors but also decrease the degree of sensitivity of template operations to variations in their values due to post-manufacturing disturbances.

Chip-independent and chip-specific robustness can be directly associated with manufacturing and post-manufacturing error sensitivities, respectively. Additionally, chip-specific robust templates can also attempt to correct manufacturing errors alike chip-independent ones.

Considering the class of binary CNN operations, namely operations with binary input and output, an optimal template is not necessarily unique. The error surface in the region of the optimum is often flat or very shallow instead of a deep isolated point (see Fig. 5, shown later, and [18]). Therefore, from this observation one can conclude that these operations are asymptotically stable regarding template values as initial conditions, i.e., as for a stable operation with a given input and initial state, for small variations on the initial conditions (template values), the output remains in the same fixed point. Due to the discrete nature of the results and to the continuous nature of the template values, asymptotic stability seems obvious for binary template operations. Nevertheless, it is less evident when considering real-valued (grayscale) inputs operations.

Considering statistical circuit design [24], where techniques such as design centering attempt to find a center for an acceptability region, an analog formulation for the problem of finding chip-specific robust templates can be established, where a center for an interval of correct operation needs to be estimated.

In statistical circuit design, *designable parameters* are used during circuit design as decision variables and will represent here the 19-dimensional vector $p = (p_1, \ldots, p_D)$ of template parameters. *Random variables*, or *noise parameters*, will represent the parameter deviations, such as manufacturing parameter variation, temperature, and etc., denoted by the vector $e = (e_1, \ldots, e_D)$ of random variables with zero mean. *Circuit variables*, which in statistical circuit design represent the variables used in circuit, process, or system simulation, will represent, in this approach, a noisy template parameter vector[1] denoted by $p' = p + e$. And finally, the vector of *circuit performances* will be represented by a simple scalar denoting the value of a cost function $G(p, e)$.

The acceptability region, which in statistical circuit design is defined as a region for which all inequality and equality constraints imposed on the vector of circuit performances are fulfilled, will be defined here in the p'-space as such a set of p' vectors in the 19-dimensional space for which the inequality $G(p') \leq S$ is fulfilled, where S denotes a limit imposed on the cost function.

The goal of this approach is therefore the same as in design centering, where the center of the acceptability function is to be found. There are several methods in the literature that solve this problem with the use of derivatives of circuit performances or their estimates. However, due to the difficulty in finding a good estimate for the derivative of a cost function that uses chip measurements, ASA is, again, used to find the optimum, now, for a noise corrupted cost function.

The addition of the noise in the cost function eliminates the flat regions of optima in the error surface and allows further improvements to an initial optimum. The cost function now contains several different embedded measurements instead of only one and the probe templates assimilate a small perturbation

$$G(p, e, \theta) = \frac{1}{r} \sum_{j=1}^{r} g(p + e_j, \theta) \tag{5}$$

[1] $p' = p + e$ models **absolute** parameter spreads and results in $\mathrm{var}\{p'_i\} = \mathrm{var}\{e_i\}$. Alternatively, one can model $p' = p + (1 + e)$, which models **relative** parameter spreads and results in $\mathrm{var}\{p'_i\} = p_i^2 \mathrm{var}\{e_i\}$.

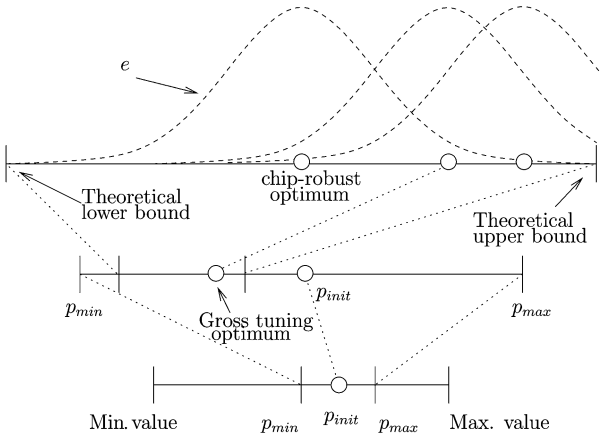

Fig. 3. Chip-specific robustness: illustration for one component of the parameter vector.

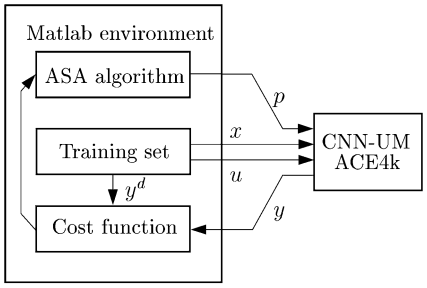

Fig. 4. Overview of the structure used in the optimization.

where r denotes the number of runs executed for the triplet θ, and e is a vector where each element corresponds to Gaussian noise with zero mean and small variance.

The addition of different samples of e to the probe template p in (5) generates a *smoothed* cost function that will statistically make this function minimal when p has its elements in the middle of each corresponding dimensional range of optima, i.e., in the middle of the acceptability region. Fig. 3 depicts the effect of this cost function in the final result for one component of the parameter vector.

The region of the error surface where the initial optimal template was located is no longer flat. With the addition of the perturbation e to the template values, it became noisy. The set of template values located in the middle of this noisy region have now stochastically more chances to generate the correct output than those closer to the borders. As a result, the final template will be very close to the most robust template for specific use in a given chip. Chip-specific robustness is thus the concept of robustness within a given chip. A similar principle was used in [23] to do chip-independent robustness optimization.

V. Experiments

The experiments were performed using the Aladdin system [25] in connection with the Matlab environment. The main features of the ASA algorithm were written for Matlab and were triggered by a the Analogic Macro Code (AMC) program running on the given CNN-UM chip. Two CNN-UM chips were used in the experiments: an ACE4k, a programmable CNN with 4096 cells disposed in a 64×64 regular grid; and an ACE16k

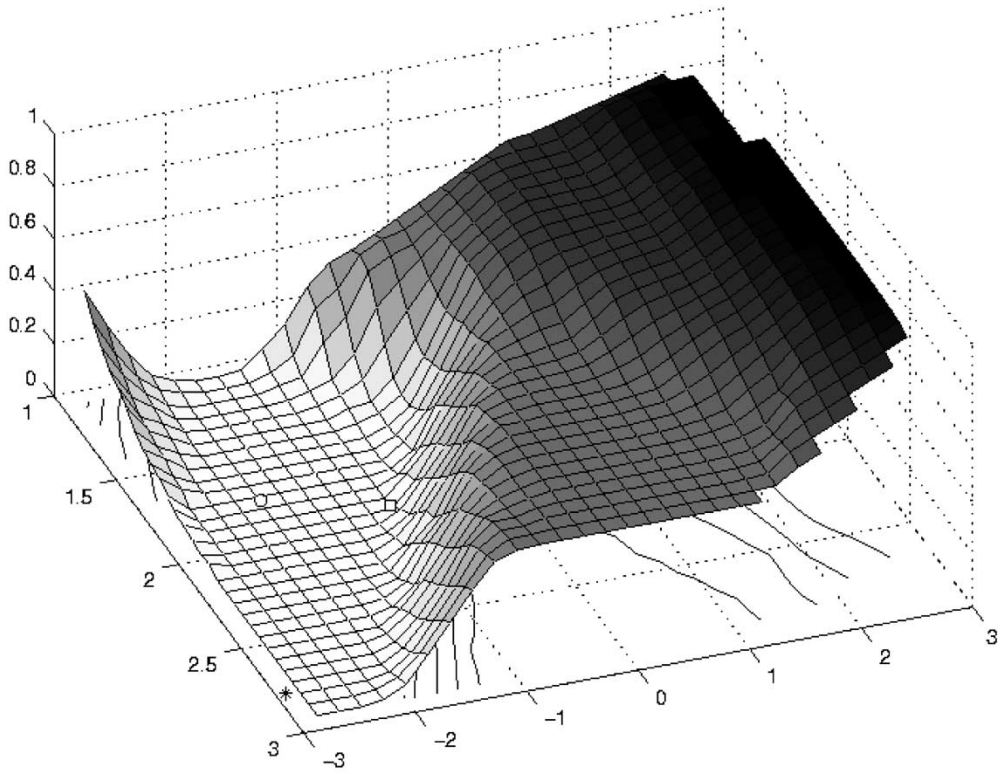

Fig. 5. Logic difference template error surface for the ACE4k chip. ◇-initial template, ∘-chip-specific optimal template, ∗-chip-specific robust template.

with 16 384 cells disposed in a 128×128 grid. All measurements were made *on-the-fly*. Fig. 4 depicts an overview of the structure used in the optimizations.

In order to ease the ASA search, each template optimization might start with constraints or structure in the template values such as symmetry or imposed zero values. The constraints are then softened for subsequent runs until the point where all 19 template values become free for optimization, according to the main algorithm shown in Fig. 2.

The input and initial state for each optimized template operation were in general random binary or grayscale images, with some exceptions[2] . The respective desired output was obtained from simulators of ideal CNN-UM using robust templates available in the literature [6]. To avoid tiling, the size of the images were chosen exactly the same as the chip sizes, i.e., 64×64 pixels and 128×128 pixels for ACE4k and ACE16k, respectively.

Owing to the speed of the chips, each annealing iteration consumed more time for the generation and acceptance of new probe templates in Matlab than for the evaluation of the cost function itself, which is done by sending the input and initial state images to the chip and acquiring its output. The total duration of a simple annealing iteration was about 50 ms and the number of iterations for each optimization was in average in the order of tens of thousand. The difference in size did not effect the duration of the measurements for the two different chips used here since CNN computations are totally parallel.

[2]Template operation like binary edge detection do not optimize well with random images.

In what follows, a demonstrative example of chip-specific robustness and more precise explanations for each individual experiments are presented. Among the experiments performed in the lab, two optimizations on the ACE4k chip are described here: binary edge detection, and average with binary output; and three optimizations on the ACE16k chip: average with binary output, thresholding to binary, and sobel edge detection.

A. Example of a Chip-Specific Robust Template

To demonstrate the concept of chip-specific robustness, Fig. 5 shows the error surface measured with the chip for the logic difference template operation. For illustrative purposes, the template was optimized with only two free parameters. The structure of this template with the free parameters a and b is

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad z = -1. \quad (6)$$

Observe that, although the initial template values are located in an optimal point in the error surface, this point is near other nonoptimal points. This proximity may result in errors as soon as any interference shifts slightly the template values. After the chip-specific optimization, the resulting template is occasionally located in a better error neighborhood. However, it is after the optimization targeting chip-specific robustness that the template assumes the most robust positioning for the given error surface.

B. Binary Edge Detection on the ACE4k Chip

Fig. 6 shows the results for this template operation. The costs of the original and the final template are presented in Table I

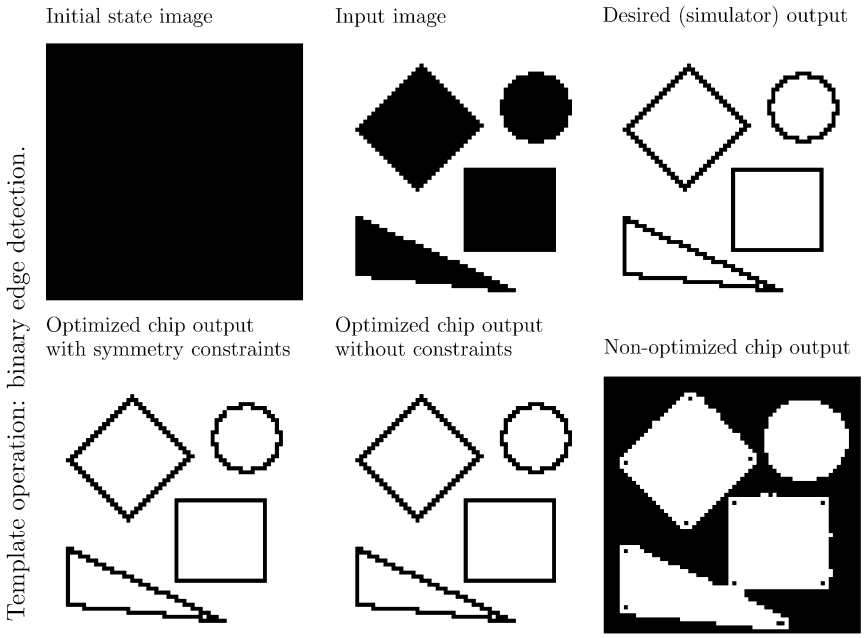

Fig. 6. Results for binary edge detection template operation on the ACE4k chip.

TABLE I
TEMPLATE VALUES AND RESPECTIVE ERROR COSTS FOR A SPECIFIC ACE4k CHIP

Operation		Template values			Cost
binary edge detection	original	$A=\begin{bmatrix}0.0000 & 0.0000 & 0.0000\\ 0.0000 & 1.0000 & 0.0000\\ 0.0000 & 0.0000 & 0.0000\end{bmatrix}$ $B=\begin{bmatrix}-1.0000 & -.00001 & -1.0000\\ -1.0000 & 8.0000 & -1.0000\\ -1.0000 & -1.0000 & -1.0000\end{bmatrix}$	$z=-1.0000$	0.7719
	final	$A=\begin{bmatrix}-0.2364 & -0.0743 & -0.1902\\ -0.1892 & 2.0833 & -0.1614\\ -0.2667 & -0.1895 & -0.1470\end{bmatrix}$ $B=\begin{bmatrix}-0.5402 & -0.0664 & -0.5350\\ 0.0304 & 2.9279 & -0.0770\\ -0.4165 & -0.0311 & -0.4229\end{bmatrix}$	$z=-1.9382$	zero
Average with binary output	original	$A=\begin{bmatrix}0.0000 & 1.0000 & 0.0000\\ 1.0000 & 2.0000 & 1.0000\\ 0.0000 & 1.0000 & 0.0000\end{bmatrix}$ $B=\begin{bmatrix}0.0000 & 0.0000 & 0.0000\\ 0.0000 & 0.0000 & 0.0000\\ 0.0000 & 0.0000 & 0.0000\end{bmatrix}$	$z=0.0000$	0.6370
	final	$A=\begin{bmatrix}-0.3513 & 1.6183 & -0.2364\\ 1.5837 & 2.9538 & 1.5837\\ -0.2364 & 1.6183 & -0.3513\end{bmatrix}$ $B=\begin{bmatrix}0.2021 & 0.7155 & 0.0181\\ 0.6530 & 1.0219 & 0.6530\\ 0.0181 & 0.7155 & 0.2021\end{bmatrix}$	$z=-2.3909$	0.1251

together with the respective template values. In Fig. 6, one can also find the output of the intermediate symmetric template

$$A=\begin{bmatrix}-0.2367 & -0.2416 & -0.2207\\ -0.1710 & 1.8889 & -0.1710\\ -0.2207 & -0.2416 & -0.2367\end{bmatrix}$$
$$B=\begin{bmatrix}-0.6522 & 0.0306 & -0.8290\\ -0.0722 & 2.8511 & -0.0722\\ -0.8290 & 0.0306 & -0.6522\end{bmatrix},\qquad z=-1.3631$$

which presented an average cost of 0.0050.

C. Average With Binary Output on the ACE4k Chip

With this template operation, the chip reacted better for symmetric templates. The results can be seen in Fig. 7 and the respective template values are in Table I. The template values for the nonsymmetric optimal template are

$$A=\begin{bmatrix}-0.5005 & 1.8088 & -0.1728\\ 1.7479 & 2.9586 & 1.6937\\ -0.2007 & 1.8192 & -0.4994\end{bmatrix}$$
$$B=\begin{bmatrix}0.3354 & 0.7805 & -0.1653\\ 0.6249 & 0.8771 & 0.7702\\ -0.2512 & 0.5741 & 0.4052\end{bmatrix},\qquad z=-2.5131$$

and its average cost was 0.1451. Its results can also be seen on Fig. 7.

D. Average With Binary Output on the ACE16k Chip

For this chip, in contrast with the ACE4k, average with binary output operation presented better results with the template

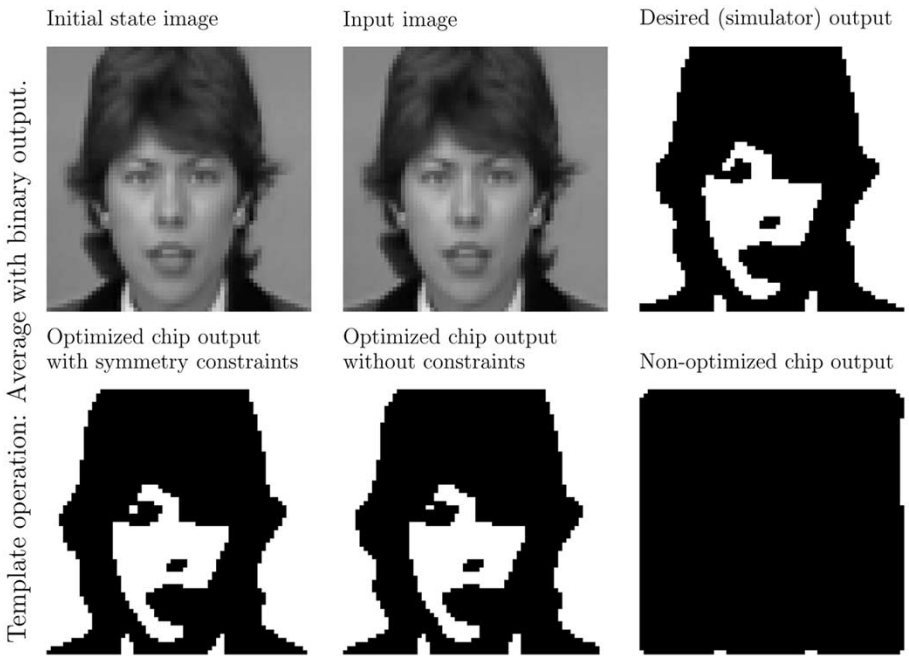

Fig. 7. Results for average with binary output template operation on the ACE16k chip.

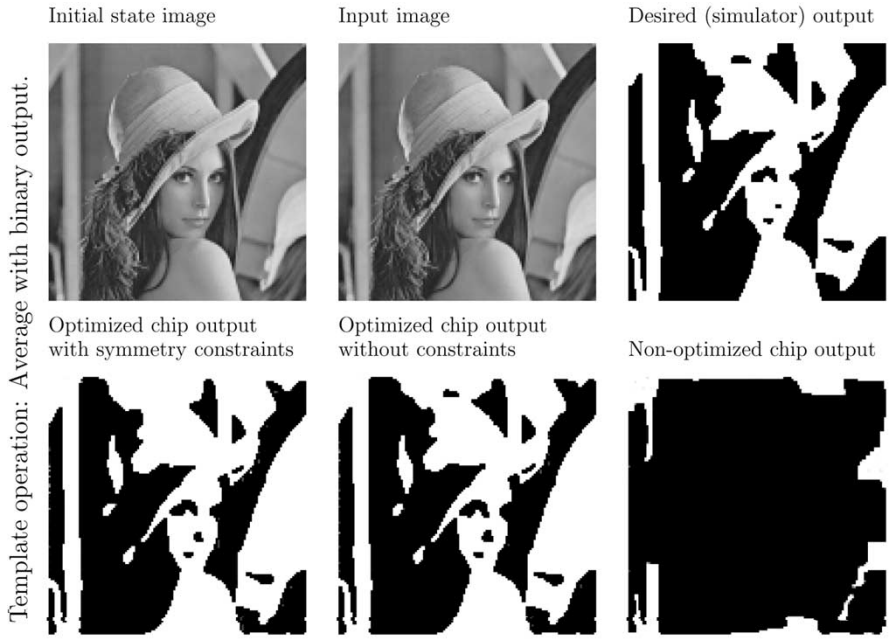

Fig. 8. Results for average with binary output template operation on the ACE16k chip.

without symmetric constraints. The best template is in Table II. The intermediate symmetric template

$$A = \begin{bmatrix} 0.3754 & 2.6839 & 0.3195 \\ 2.9885 & 5.3824 & 2.9885 \\ 0.3195 & 2.6839 & 0.3754 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 4.1252 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 \end{bmatrix}, \qquad z = -3.1911$$

which presented and average cost of 0.1734, has its results depicted in Fig. 8 together with the other results.

E. Thresholding to Binary on the ACE16k Chip

For this operation, symmetry constraints did not effect the average cost much. The symmetric template

$$A = \begin{bmatrix} -0.1854 & -0.2078 & -0.1681 \\ -0.1445 & 5.9253 & -0.1445 \\ -0.1681 & -0.2078 & -0.1854 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 5.9974 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 \end{bmatrix} \qquad z = -2.4572$$

TABLE II
TEMPLATE VALUES AND RESPECTIVE ERROR COSTS FOR A SPECIFIC ACE16k CHIP

Operation		Template values	Cost
Average with binary output	original	$A = \begin{bmatrix} 0.0000 & 1.0000 & 0.0000 \\ 1.0000 & 2.0000 & 1.0000 \\ 0.0000 & 1.0000 & 0.0000 \end{bmatrix}$ $B = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 \end{bmatrix}$ $z = 0.0000$	0.6467
	final	$A = \begin{bmatrix} 0.4370 & 2.8775 & 0.4542 \\ 2.9294 & 5.6667 & 2.8169 \\ 0.6100 & 2.7833 & 0.8236 \end{bmatrix}$ $B = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 4.9116 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 \end{bmatrix}$ $z = -3.5225$	0.1583
Threshold	original	$A = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 2.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 \end{bmatrix}$ $B = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 \end{bmatrix}$ $z = 0.0000$	0.5032
	final	$A = \begin{bmatrix} -0.1449 & -0.2467 & -0.2201 \\ -0.1023 & 5.8520 & -0.2287 \\ -0.1542 & -0.1225 & -0.2217 \end{bmatrix}$ $B = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 5.8885 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 \end{bmatrix}$ $z = -2.5662$	0.2200
Sobel	original	$A = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & -4.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 \end{bmatrix}$ $B = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 \\ -3.0000 & 0.0000 & 3.0000 \\ 0.0000 & 0.0000 & 0.0000 \end{bmatrix}$ $z = -1.7000$	0.3099
	final	$A = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & -3.4036 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 \end{bmatrix}$ $B = \begin{bmatrix} 0.2404 & 0.0145 & -0.0678 \\ -2.5880 & 0.3143 & 2.8074 \\ -0.1014 & -0.0414 & 0.1741 \end{bmatrix}$ $z = -2.4656$	0.0672

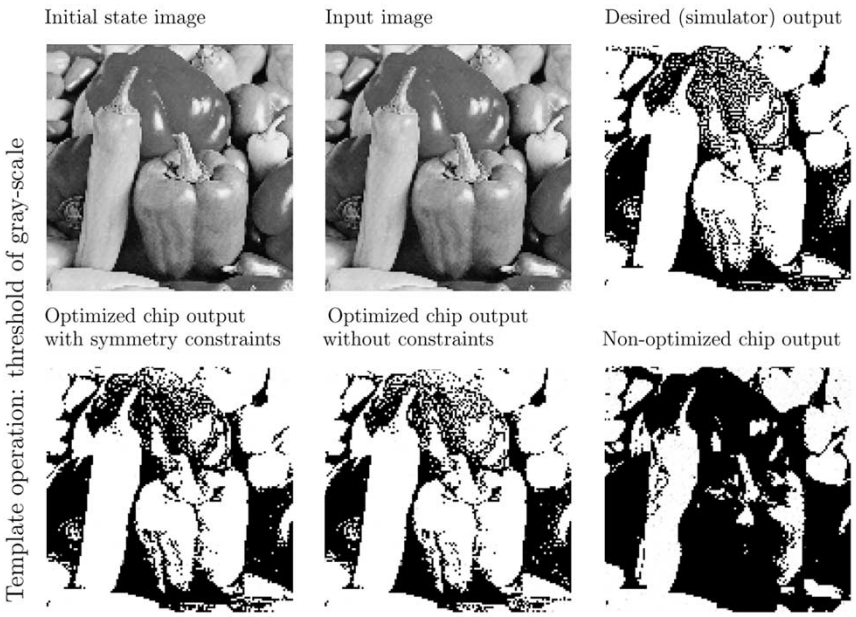

Fig. 9. Results for thresholding to binary template operation on the ACE16k chip.

with average cost equal to 0.2188 was slightly worst the its nonsymmetric version, which values and cost can be seen in Table II. The results can be visualized in Fig. 9.

F. Sobel Edge Detection on the ACE16k Chip

Although template operations with grayscale output are difficult to optimize in CNN chips, it was possible with sobel edge detection due to the stability of this operation. However for the general grayscale case, a more complex approach which takes into account desirable trajectories has to be considered. Fig. 10 presents the results. The template values and respective costs for this operation can also be found in Table II.

VI. CONCLUSION

Despite the extraordinary speed performance of CNN-UM chips for image-processing tasks, digital systems are still predominant in the field owing to their superior reliability. The development of a method toward chip-specific robustness contributes to diminish this superiority. The method described here works well for all tested stable binary output template operations and a grayscale template operation. Using an optimization method that does not rely on information about the gradient of the cost function allowed this approach to efficiently tune not only uncoupled templates but also coupled

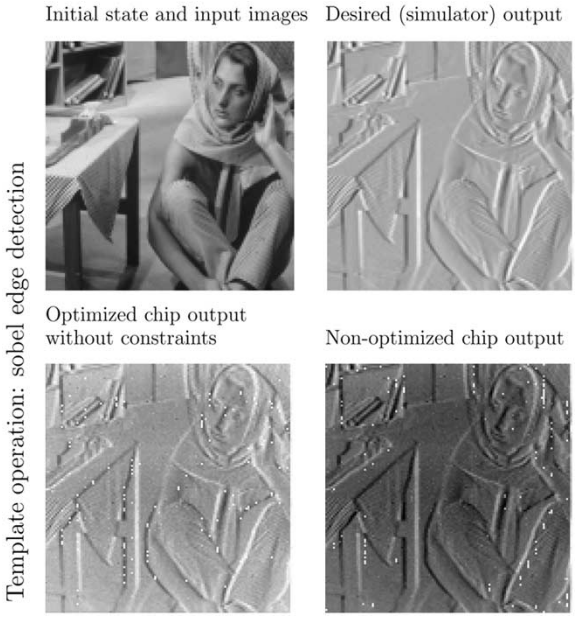

Fig. 10. Results for sobel edge detection template operation on the ACE16k chip.

ones. Additional improvements for the method can include the use of hardware parameters (e.g., zero level of templates) in the optimization variables. For the case of grayscale outputs, a more elaborated approach that takes into account the transient time has to be developed and will be also considered for future research. Chip-specific robust tuning of templates provides a method to place parameters values in the middle of a correct operating range. This minimizes the erroneous behavior of CNN chips for optimized templates due to parameter variations caused by post-manufacturing disturbance, e.g., temperature and noise, which may cause the parameter values to fall outside the correct working range. Chip-specific robustness exposes a trend of analog and mixed-signal self-test and self-tuning chips that can be perhaps explored by embeddable implementation of this concept.

REFERENCES

[1] L. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1272–1290, Oct. 1988.

[2] T. Roska and L. O. Chua, "The CNN universal machine," *IEEE Trans. Circuits Syst. II*, vol. 40, pp. 163–173, Mar. 1993.

[3] T. Kozek, T. Roska, and L. O. Chua, "Genetic algorithm for CNN template learning," *IEEE Trans. Circuits Syst. I*, vol. 40, pp. 392–402, Mar. 1993.

[4] B. Chandler, C. Rekeczky, Y. Nishio, and A. Ushida, "Adaptive simulated annealing in CNN template learning," *IEICE Trans. Fund.*, vol. E82, no. 2, pp. 398–402, 1999.

[5] C. Güzelis, S. Karamahmut, and I. Genç, "A recurrent perception learning algorithm for cellular neural networks," *Interdiscip. J. Phys. Eng. Sci.*, vol. 51, no. 4, pp. 296–309, 1999.

[6] T. Roska, L. Kek, L. Nemes, A. Zarándy, M. Brendel, and P. Szolgay, "CNN software library," in *CADETWin*, Budapest, Hungary: Computer and Automation Institute of the Hungarian Academy of Sciences, 1998.

[7] T. Roska and A. Rodrguez-Vázquez, "Toward visual microprocessors," *Proc. IEEE*, vol. 90, pp. 1244–1257, July 2002.

[8] S. Espejo, R. D. Castro, R. Carmona, and A. Rodríguez-Vázquez, "A CNN universal chip in CMOS technology," *Int. J. Circuit Theory Applicat.*, vol. 24, pp. 93–109, 1996.

[9] G. Linan, S. Espejo, R. Domínguez-Castro, and A. Rodríguez-Vázquez, "ACE4k: An analog I/O 64 × 64 visual microprocessor chip with 7-bit analog accuracy," *Int. J. Circuit Theory Applicat.*, vol. 30, no. 2–3, pp. 89–116, 2002.

[10] ——, "ACE16k: A 128 × 128 focal plane analog processor with digital I/O," in *Proc. IEEE Int. Workshop Cellular Neural Networks and Their Applications, (CNNA'02)*, Frankfurt, Germany, July 2002, pp. 132–138.

[11] L. O. Chua, T. Roska, T. Kozek, and A. Zarndy, "CNN universal chips crank up computing power," *IEEE Circuits Devices Mag.*, vol. 12, pp. 18–28, July 1996.

[12] K. R. Crounse and L. O. Chua, "Methods for image processing and pattern formation in cellular neural networks: A tutorial," *IEEE Trans. Circuits Syst. I*, vol. 42, pp. 583–601, Oct. 1995.

[13] T. Roska, L. Kék, L. Nemes, Á. Zarándy, M. Brendel, and P. Szolgay, *CADETWin*, Budapest, Hungary: Computer and Automation Institute of the Hungarian Academy of Sciences, 1998.

[14] J. A. Nossek, "Design and learning with cellular neural networks," *Int. J. Circuit Theory Applicat.*, vol. 24, pp. 15–24, 1996.

[15] B. Mirzai, D. Lim, and G. S. Moschytz, "Robust CNN templates: Theory and simulations," in *Proc. IEEE Int. Workshop Cellular Neural Networks and Their Applications, (CNNA'96)*, Seville, Spain, 1996, pp. 393–398.

[16] P. Kinget and M. Steyaert, "Evaluation of CNN template robustness toward VLSI implementation," *Int. J. Circuit Theory Applicat.*, vol. 24, no. 1, pp. 93–110, 1996.

[17] A. Zarándy, "The art of CNN template design," *Int. J. Circuit Theory Applicat.*, vol. 27, no. 1, pp. 5–23, 1999.

[18] P. Földesy, L. Kék, A. Zarándy, and G. B. T. Roska, "Fault-tolerant design of analogic CNN templates and algorithms—Part I: The binary output case," *IEEE Trans. Circuits Syst.*, vol. 46, pp. 312–322, Feb. 1999.

[19] L. Ingber, "Very fast simulated Re-annealing," *J. Math. Comput. Model.*, vol. 12, pp. 967–973, 1989.

[20] ——, (2002) *Adaptive Simulated Annealing (ASA) Version 24.1* [Online]. Available: http://www.ingber.com

[21] M. Hanggi and G. S. Moschytz, "An exact and direct analytical method for the design of optimally robust CNN templates," *IEEE Trans. Circuits Syst.*, vol. 46, pp. 304–311, Feb. 1999.

[22] S. Xavier-de-Souza, M. E. Yalcin, J. A. K. Suykens, and J. Vandewalle, "Automatic chip-specific CNN template, optimization using adaptive simulated annealing," in *Proc. Eur. Conf. Circuit Theory and Design (ECCTD'03)*, Krakow, Poland, Sept. 2003.

[23] M. Hanggi and G. S. Moschytz, "Stochastic and hybrid approaches toward robust templates," in *Proc. IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'98)*, London, U.K., Apr. 1998, pp. 366–371.

[24] M. A. Styblinski, "Statistical circuit design," in *Computer-Aided Design and Optimization*. Boca Raton, FL: CRC Press, 1995, ch. 55, pp. 1453–1486.

[25] Analogic Computers Aladdin Visual Computer (2003). [Online]. Available: http://www.analogic-computers.com



Samuel Xavier-de-Souza was born in Natal, Brazil, in 1976. He received the computer engineering degree from the Federal University, Rio Grande de Norte, Brazil, in 2000. He is currently working toward the Ph.D. degree in applied sciences from the Katholieke Universiteit, Leuven, Belgium.

He was with the Interuniversity Microelectronics Center (IMEC), Leuven, Belgium, working on advanced design technologies. His research interests are in neural networks and their applications, in particular, optimization of cellular neural/nonlinear networks and their very large-scale integration implementations.



Müştak E. Yalçın (S'03) was born in Unye, Turkey, in 1971. He recieved the B.Sc. and M.Sc. degrees in electronics and telecommunications engineering from the Istanbul Technical University, Istanbul, Turkey, in 1993 and 1997, respectively. He is currently pursuing the Ph.D. degree in applied sciences from the Katholieke Universiteit, Leuven, Belgium.

His research interests include nonlinear circuits and systems, neural networks and their applications, in particular, cellular nonlinear networks, multi-scroll chaotic attractors, spatio-temporal waves and synchronization.



Johan A. K. Suykens (M'03) was born in Willebroek Belgium, in 1966. He received the degree in electromechanical engineering and the Ph.D. degree in applied sciences from the Katholieke Universiteit, Leuven, Belgium, in 1989 and 1995, respectively.

In 1996, he was a Visiting Postdoctoral Researcher at the University of California, Berkeley. He was a Postdoctoral Researcher with the Fund for Scientific Research (FWO) Flanders, Belgium, and is currently an Associate Professor with Katholieke Universiteit. His research interests are mainly in the areas of the theory and application of neural networks and nonlinear systems. He is author of the books *Artificial Neural Networks for Modeling and Control of Nonlinear Systems* (Norwell, MA: Kluwer, 1995) and *Least Squares Support Vector Machines* (Singapore: World Scientific, 2002) and editor of the books *Nonlinear Modeling: Advanced Black-Box Techniques* (Norwell, MA: Kluwer, 1998) and *Advances in Learning Theory: Methods, Models and Applications* (Amsterdam, The Netherlands: IOS Press, 2003).

Dr. Suykens received the IEEE Signal Processing Society 1999 Best Paper (Senior) Award and several Best Paper Awards at International Conferences. He is a recipient of the International Neural Networks Society 2000 Young Investigator Award for significant contributions in the field of neural networks. In 1998, he organized an International Workshop on Nonlinear Modeling with Time-series Prediction Competition. He has served as Director and Organizer of a NATO Advanced Study Institute on Learning Theory and Practice, Leuven, Belgium, July 2002. He has served as an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: FUNDAMENTAL THEORY AND APPLICATIONS (1997–1999), and since 1998, he is an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS.



Joos Vandewalle (F'92) was born in Kortrijk, Belgium, in 1948. He received the electrical engineering degree and the doctoral degree in applied sciences from the Katholieke Universiteit, Leuven, Belgium, in 1971 and 1976, respectively.

From 1976 to 1978, he was a Research Associate and from July 1978 to July 1979, a Visiting Assistant Professor at the University of California, Berkeley. Since July 1979, he has been with the Department of Electrical Engineering (ESAT), Katholieke Universiteit, where he is Full Professor since 1986 and the Head of the SCD division at ESAT, that has more than 120 researchers. From August 1996 to August 1999, he was Chairman of the Department of Electrical Engineering and from August 1999 until July 2002, he was the Vice-Dean, Faculty of Engineering, Katholieke Universiteit. since 1984, he is also an Academic Consultant at the Interuniversity Microelectronics Center, Leuven, Belgium. In the second semester of 2002–2003 he was on sabbatical leave at the I3S laboratory of French National Center for Scientific Research (CNRS) Sophia Antipolis, France. He teaches courses in linear algebra, linear and nonlinear system and circuit theory, signal processing and neural networks. His research interests are mainly in mathematical system theory and its applications in circuit theory, control, signal processing, cryptography and neural networks. His recent research interests are in nonlinear methods (support vector machines, multilinear algebra) for data processing. He has authored or coauthored more than 200 international journal papers in these areas. He is the co-author of four books and co-editor of five books. He is a member of the editorial board of the *International Journal of Circuit Theory and its Applications*, *Neurocomputing*, *Neural Networks*, and the *Journal of Circuits Systems and Computers*. Since 2001, he is a member of the Advisory Board of the *International Journal on Information Security* (IJIS). Since January 2001, he is Co-editor-in-Chief of *Journal A, the Benelux journal on Automation*.

Dr. Vandewalle received several Best Paper Awards and Research Awards. In 1991–1992, he held the Francqui Chair on Artificial Neural Networks at the University of Liége (Belgium), and in 2001–2002, he held the Chair on Advanced Data Processing techniques at the Free University of Brussels (Belgium). From 1989 to 1991, he was an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: FUNDAMENTAL THEORY AND APPLICATIONS and its Deputy Editor-in-Chief from January 2002 to December 2003. He is a member of the Academia Europaea and of the Belgian Academy of sciences and of two Committees of the Fonds voor Wetenschappelijk Onderzoek Vlaanderen (Belgium). He is also Fellow of the Institute of Electrical Engineers, U.K.