Parametric fault identification and dynamic compensation techniques for cellular neural network hardware

M.R. Grimaila
J. Pineda de Gyvez

Abstract: Testing strategies to quantify parametric faults in a fully programmable, two-dimensional cellular neural network (CNN) are presented. The approach is intended to quantify system offsets, time constant mismatches, nonlinearities in the multipliers and state nodes, and the magnitude of the dynamic range of operation which can lead to misconvergence in the CNN array. For some cases, the authors present dynamic solutions by compensating the templates, the input data, and/or the initial condition values to minimise or cancel the undesired effects. The proposed dynamic compensation techniques can be applied to any CNN independent of the array size or topology. To demonstrate the feasibility of the proposed techniques, the authors examine their application to an actual complex VLSI CNN implementation.

1 Introduction

The emergence of cellular neural networks (CNNs) has given origin to many applications [1]. Initially, many of the applications were based on software systems that simulate the CNN behaviour based upon ideal computer models [2–5]. Subsequently, VLSI hardware implementations of CNN arrays have evolved allowing for near ‘real-time’ image processing [6–14]. Unfortunately results obtained from the CNN hardware may be nonideal due to the tolerances and typical lack of accuracy of analog designs [15]. A key issue is how to determine the precision of templates for VLSI use as they may need additional tuning to perform in the same manner as the ideal ones [16]. The problems are partly due to the inherent random nature of VLSI semiconductor fabrication [17, 18]. For certain combinations of template and input values to the CNN, these deviations can simply be detrimental. Various testing strategies have been explored to identify problems and in certain cases it is possible to compensate these problems [19–22].

Several parametric problems may arise during the operation of a CNN. In this paper we consider only four such problems: (i) undesired convergence effects due to offsets, (ii) race conditions among cells due to time constant mismatches, (iii) nonlinear multiplier or state node characteristics, and (iv) the limited dynamic range of operation. The proposed testing strategy is capable of identifying excessive variations in these four areas. The parametric tests extract their information when the state of the cell is in the linear range of the activation function. To compensate the faults, we propose noninvasive techniques to quantify and then compensate spatial cell tolerances independent of the particular CNN architecture. Namely, we propose the use of online compensations, such as slightly altering template values, cell inputs, and/or cell initial conditions while the CNN is in operation.

2 Behavioural operation of a cellular neural network

Consider a two-dimensional CNN array containing M × N locally connected cells. Any cell on the ith row and jth column, C(i, j), is connected only to its cells in its immediate neighbourhood. Fig. 1 shows the schematic of the generic CNN cell [23]. Each cell has a state x, a constant external input u and output y. The first order nonlinear differential equation defining the dynamics of a cellular CNN cell can be written as follows [1]

\[ C \frac{dx_{ij}(t)}{dt} = \frac{x_{ij}(t)}{R} + \sum_{(k,l) \in N(i,j)} A(i,j;k,l)y(t) \]

\[ + \sum_{(k,l) \in N(i,j)} B(i,j;k,l)u_{kl} + I \quad (1a) \]
\[ y_{ij}(t) = \frac{1}{2} (|x_{ij}(t) + 1| - |x_{ij}(t) - 1|) \]  

(1b)

where \( x_{ij} \) is the state of cell \( C(i,j) \), \( x_{ij}(0) \) is the initial condition of the cell, \( C \) is a linear capacitor, \( R \) is a linear resistor, \( I \) is an independent current source, \( A(i,j;k,l) \) is the feedback multiplier template values, \( B(i,j;k,l) \) is the control multiplier template values, and \( y_{ij} \) represents the nonlinear activation function.

The CNN array is controlled by choosing the initial conditions, setting the external inputs, and selecting the desired mode of operation. In this global assumption, three behavioural modes of operation can be identified: namely, (i) cell initialisation, (ii) cell evaluation and (iii) result extraction. The behavioural modes are selected by two mutually exclusive signals, the set initial condition signal, \( S \), and the evaluate signal, \( E \), that activate two switches, \( SW_1 \) and \( SW_2 \). In the initialisation phase, \( S \) is activated while \( E \) is negated to allow the state of the cell to be charged to the desired initial condition value \( x(0) \). The CNN enters the evaluation mode when \( E \) is activated and \( S \) is negated, connecting the summer to the integrator and allowing the dynamic processing to commence. After the cell converges, or after a finite period of time, both \( S \) and \( E \) signals are negated which disconnects the summer from the integrator, holding the cell state for the result extraction phase. We can thus write the solution of eqn. 1 in a more descriptive form that includes the effect of the control signals as follows:

\[
\{x_{ij}\}^{S,E} = \{x_{ij}(0)e^{-i/RC}\}^{S,E} + \left\{ \frac{1}{C} \int_0^t e^{-(t')/RC} \left[ \sum_{(k,l)\in N(i,j)} A(i,j;k,l)y_{kl}(t') + \sum_{(k,l)\in N(i,j)} B(i,j;k,l)u_{kl} + 1 \right] \right\}^{S,E} \]

(2)

where the superscripts \( ^{S,E} \) are used to indicate the controlling conditions under which the distinct sections of the equation are executed in the actual hardware. The equation clearly shows that the state of the cell depends upon the mode of operation.

3 VLSI cellular neural network implementation

In this section we briefly describe an actual VLSI CNN implementation fabricated in MOSIS 2μm CMOS [24]. This implementation will be used as the actual test vehicle for our investigation. Each cell in the VLSI CNN array contains 18 multipliers, a lossy integrator, a hard limiter, a biasing circuit, and a sample/hold circuit as shown in Fig. 2. All of the multipliers are located at the output of the cell, as opposed to the generic CNN, where the multipliers are located at the input. A problem with this approach is that cells on the perimeter will have different offsets than interior cells due to the difference in the number of multipliers driving the cell. The power supplies are ±3V and the state voltage is bounded by these values. The hard limiter implements the activation function which saturates at ±0.5V. The array output is the state voltage and requires an external activation function to limit in the range \([-0.5, 0.5]\). The integration capacitance is 1pF and the OTA resistor is adjustable in the range from 141KΩ to 2MΩ. The transconductance factor for the multipliers, \( g_{m\text{MULT}} \), is 20μA/V and the transconductance factor for the bias circuit, \( g_{m\text{BLAV}} \), is 8.5μA/V.

3.1 Offset cancellation

The main effect of offset is that the equilibrium point of the cell will drift to an undesired value leading to possible system instabilities or accumulation of propagated offsets depending upon the templates being used. For example, let us consider the black and white dithered image shown in Fig. 3a. The image is processed with a contrast-sharpening template using a time-multiplexing approach with a 10 × 10 CNN array [25].
\[
C \frac{dE_{i,j}(t)}{dt} = -f(x_{i,j}(t)) + g_{i,j}(t) 
\]  
\[
g_{i,j}(t) = \sum_{k=1}^{K} \sum_{l=1}^{L} a_{i+k,j+l} y_{i+k,j+l}(t) - a_{i,j} y_{i,j}(t) 
\]  
\[
+ \sum_{k=-1}^{K} \sum_{l=-1}^{L} b_{i+k,j+l} u_{i+k,j+l} + I 
\]  
\[
f(x_{i,j}(t)) = -\frac{a_{i,j}}{2} (|x_{i,j}(t) + 1| - |x_{i,j}(t) - 1|) + \frac{x_{i,j}(t)}{R} 
\]  
\[
\text{Now, without loss of generality let us further assume that the offset, } g_{i,j} \text{ originates only from cell } C(i,j) \text{ and that } g_{i,j}(t) = 0. \text{ Then eqn. 3c can be modified to take this offset into account as follows:}
\]
\[
f(x_{i,j}(t)) = -\frac{a_{i,j}}{2} (|x_{i,j}(t) + 1| - |x_{i,j}(t) - 1|) + \frac{x_{i,j}(t)}{R} + \varphi
\]  
\[
\text{Let us now consider the case } a_{i,j} > 1/R \text{ for which the magnitude of the stable equilibrium point is always greater than 1 (i.e. } |x_{i,j}| > 1). \text{ The equilibrium points of eqn. 3a can be obtained for } f(x_{i,j}(t)) = 0 \text{ to yield}
\]
\[
x_{i,j}^\pm(t) = \left\{ \begin{array}{ll}
\varphi & x_{i,j}(t) < 0 \\
-1 < x_{i,j}(t) < 1 & x_{i,j}(t) > 1
\end{array} \right.
\]  
\[
\text{To measure the offset in the array it is necessary to fix the } A \text{ template, the } B \text{ template, the } I \text{ value, the initial condition value } x_0 \text{ and the external input values } u_{i,j} \text{ to zero. The control signal } S \text{ is negated while } E \text{ is asserted, allowing the multipliers and bias circuit offsets to drive the state node. Ideally, the cell state would converge to zero, but due to circuit nonidealities which induce offsets the cell state is typically nonzero. After all inputs are zeroed and the array is in evaluate mode, the offsets for each cell are measured and recorded after five time constants to ensure that a steady state has been reached.}
\]

The following pseudo code summarises this strategy:

\[
A < S, E > \leftarrow 0, B < S, E > \leftarrow 0, I < S, E > \leftarrow 0
\]
\[
a_{i,j} < S, E > \leftarrow 0
\]
\[
[x_{i,j}(0)] < S, E > \leftarrow 0
\]
\[
\forall([y_{i,j}(t > 5\tau)] < S, E > \neq 0) C_{i,j} \Rightarrow \text{ offset}
\]

We propose two different techniques to cancel the offset: (i) a global method and (ii) a local method. The global method consists of first calculating the average offset value of all cells and then applying the necessary corrective quantity to the system via the I bias input. The global offset correction technique has the advantage that it is application independent; that is, it does not depend on the particular templates selected for the CNN application. The amount of I bias input correction required depends only upon the average cell offset, the transconductance of the bias circuit, \(gm\text{bias}\), and the integration resistance. We define the magnitude of the average offset, \(\varphi\), and the amount of corrective signal required to cancel the average offset, \(I^c\), as follows:

\[
\varphi = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} x_{i,j} \quad (6a)
\]
\[
I^c = \frac{-\varphi}{(gm\text{bias}R)} \quad (6b)
\]
The corrective bias input, \( I' \), is then added to the desired bias input, \( I \), dictated by the particular CNN application. A disadvantage of this method is that although it corrects for the average offset, not all the cells have the same offset and thus the offset is not completely eliminated. A more robust scheme would be to correct the offset in each cell independently. The goal of the local offset cancellation is to reduce each cell's offset to within a minimum error value around zero. The error is selected based upon the desired precision of the image processing. The offset reduction is achieved by manipulating both the input image and \( B \) template values such that the desired offset correction is applied to each cell. The drawback of the local compensation technique is that the compensation values are a function of the entries of the \( B \) template, and must be recalculated whenever the \( B \) template changes.

Let us first consider the case when all template values and the bias input are fixed at zero, except for the central template element \( b_{i,j} \) which is fixed to a nonzero value. If we apply an input corrective signal, each cell will have the relationship \( g_{i,j}(t) = b_{i,j}u^{*}_{i,j} \) which adds an external signal to the current value of the state node. Under this situation the equilibrium points of eqn. 3a are given as

\[
x^{*}_{i,j}(t) = \begin{cases} 
\varphi + b_{i,j}u^{*}_{i,j}, & -1 < u^{*}_{i,j} < 1 \\
\alpha_{i,j} + \varphi + b_{i,j}u^{*}_{i,j}, & x^{*}_{i,j}(t) > 1 \\
-\alpha_{i,j} + \varphi + b_{i,j}u^{*}_{i,j}, & x^{*}_{i,j}(t) < -1
\end{cases}
\]  

(7)

From eqn. 7 one can see that the local offset cancellation will take effect for

\[ b_{i,j}u^{*}_{i,j} = -\varphi \]  

(8)

Notice that this cancellation scheme is in fact local because we have only considered a single cell. To use it in practice, \( u^{*}_{i,j} \) is added to the actual external input signal \( u_{i,j} \). Naturally, cell \( C(i,j) \) is not isolated for all cases and one must take into consideration the effect of surrounding cell inputs for template entries \( b_{i,j} \neq 0 \). Therefore eqn. 8 can be extended to the following:

\[
u^{*}_{i,j} = \frac{1}{b_{i,j}} \left( \sum_{k=1}^{M} \sum_{l=1}^{N} b_{i+k,j+l}u^{*}_{i+k,j+l} + \varphi_i \right)
\]  

(9)

where \( u^{*}_{i,j} \) is the offset correction factor of each cell \( C(i,j) \). The equation above results in a set of \( M \) times \( N \) simultaneous equations which must be solved to find the corrective \( u^{*}_{i,j} \) values.

We present now an example at the VLSI level (see Section 3) of the detrimental effect of offset and how the two previously explained correction schemes are used. Consider a 1 \( \times \) 6 CNN array processing an input image using a connected component detection (CCD) template [3]. Assume that the integration resistance for this example is fixed at 141k\( \Omega \). The \( A \) template values are 0.1 \(-0.45\) 0.1 while the \( B \) template and the \( I \) value are fixed at zero. The input and expected output images are shown in Fig. 5. To simulate offset, an HSPICE Monte Carlo simulation was performed in which the width and length of each transistor was varied by 15%. Fig. 6 shows the simulation results for each of the six cells shown in panels 1 to 6, respectively. Note that four of the ten Monte Carlo runs result in at least one cell converging incorrectly. One of the Monte Carlo variations that resulted in incorrect array convergence was selected to test the techniques of offset correction. The offset for each of the six cells in the faulted array was measured and recorded. Each cell contains a different offset as shown in the first row of Table 1.

![Image 5](image5.png)

**Fig 5** CNN 1 \( \times \) 6 CCD

**Fig 6** CNN 1 \( \times \) 6 CCD simulation 15% Monte Carlo variation of transistor geometry

- 1+0.5V
- 1-0.5V
- 1+0.2V
- 1-0.2V

The average offset value, \( \bar{\varphi} \), is \(-140.2\) mV. Using the global correction method, we were able to subtract the average offset value in each cell by applying the corrective bias value. Since the transconductance of the bias circuit is 8.5\( \mu \)A/V, the calculated bias correction term is 116.9\( \mu \)V. The cell offset values after application of the global correction technique are shown in the second row of Table 1.

**Table 1**: Cell offsets for a 1 \( \times \) 6 CNN array resulting from a 15% variation in transistor width and length

<table>
<thead>
<tr>
<th>Cell</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>mV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-85</td>
<td>-136</td>
<td>-146</td>
<td>-249</td>
<td>-91</td>
<td>-134</td>
</tr>
<tr>
<td>2</td>
<td>31.9</td>
<td>-19.1</td>
<td>-29.1</td>
<td>-132.1</td>
<td>25.9</td>
<td>-17.1</td>
</tr>
<tr>
<td>3</td>
<td>60.28</td>
<td>94.45</td>
<td>103.5</td>
<td>176.8</td>
<td>64.54</td>
<td>95.04</td>
</tr>
</tbody>
</table>

After applying the global bias correction input, the CCD simulation was repeated to see if the array would yield the unfaulted result. Unfortunately, for this particular application applying the global offset correction did not yield the unfaulted expected result. Next, the local offset correction technique was applied to the faulted array. The CCD template shown above has the \( B \) template fixed at zero. For local correction, we must set the centre element of the \( B \) template to \( b_{i,j} = 0.5V \) to allow each cell's input to influence the state of the cell. The necessary values to correct the offset were calculated based upon the \( b_{i,j} \) value, the transconductance of the multiplier circuit, and the integration resistance.
The corrective input values required to cancel each cell's offset are shown in the third row of Table 1.

The corrective inputs were applied and the offset in each cell was reduced to less than ±1 mV. The CCD simulation was repeated to see if the local offset correction restored the expected result. Fortunately, the local compensation technique corrected the cell misconvergence. Fig. 7 shows the uncompensated, the global compensated, and the local compensated results for each one of the six cells. The results show that the global compensation technique is not effective in correcting array misconvergence. The local compensation technique proved to be effective in correcting array convergence.

![Fig. 7](image_url)

**Fig. 7** CNN 1 x 6 CCD simulation faulted, global correction and local correction results

- 4.5V
- 1.05V
- uncompensated ( )
- local ( )
- global ( )

3.2 Time constant mismatches

Ideally, all cells in the array have the same time constants. However, when the actual hardware is fabricated there are variations that result in time constant mismatches. If the mismatch is severe enough, the array will not converge to the desired values depending upon the templates being used. For example, let us consider the black and white dithered image as previously shown in Fig. 6. If there are variations, then misalignments may arise. As with the previous offset example, the local compensation technique was used.

To see analytically the effect of time constant mismatches, let us consider a cell C(i, j) and its interaction with cell C(i, j+1). Let us further assume that C(i, j) and C(i, j+1) have time constants τ1 and τ2, respectively, and that I bias and both A and B templates are zero except for entry aji,j+1. Consider now the case when C(i, j+1) is initially charged as follows:

\[ x_{i,j+1} = x_{i,j+1}(0) e^{-t/\tau_2} \]  

(10)

The state of C(i, j) is obtained by solving eqn. 1. This leads to the following solution:

\[ x_{i,j}(t) = x_{i,j}(0)e^{-t/\tau_1} + \frac{1}{C} \int_0^t e^{-(t-u)/\tau_1} \left[ a_{i,j+1} x_{i,j+1}(0) e^{-u/\tau_1} \right] du \]  

(11a)

\[ = x_{i,j}(0)e^{-t/\tau_1} + \frac{1}{C} \left[ -\tau_1 e^{-t/\tau_1} \right] \]  

(11b)

\[ + \tau_1 a_{i,j+1} \left( x_{i,j+1}(0) e^{-t/\tau_1} \right) \left( 1 - e^{-t/\tau_1} \right) \]  

(11c)

Of particular interest is the expression between brackets \{ \}. This expression contains the value of C(i, j+1) to be transmitted to C(i, j). Notice however that it keeps its time constant τ2. If this cell is too fast, that is, \( \tau_2 < \tau_1 \), a race condition may arise as C(i, j) will see only a zero value because of the fastest exponential decay of C(i, j+1). The net effect is simply that C(i, j+1) will have a meaningless contribution to state node x_{i,j}. This effect can be detrimental in situations like edge detection or connected component detection in which the current pixel values change depending upon the neighbour values.

Let us address now the testing procedure to characterise the time constants of each cell. First, the array is locally compensated to cancel the offset in each cell. The A template is fixed at zero and a pulse is injected into the bias input, I. The magnitude of the input pulse is chosen so that the resulting state voltage remains in the activation function's linear region. The width of the pulse is chosen to be at least ten times the cell time constant to allow mismatches of up to 100% to be measured. Each cell is monitored and the output voltage transition from zero to the maximum output voltage is recorded. If we examine the solution of the equation representing the capacitor voltage charging from a zero initial value to a final value, \( V_f \), we obtain the following equation:

\[ v(t) = V_f (1 - e^{-t/\tau}) \]  

(12)

As shown in eqn. 12, at time t = 3\( \tau \) the capacitor voltage will reach 95% of its final value. By measuring the time required to charge to this value, and dividing by three, we obtain the cell time constant. The pseudocode for the procedure is given below:

- \( A_{<5,E>} \) ← 0, \( B_{<5,E>} \) ← 0
- \( u_{<5,E>} \) ← 0
- \( [x_{i,j}(0)]_{<5,E>} \) ← 0
- \( [I]_{<5,E>} \) ← Pulse input
- \( \text{Record } \forall \left( y_{i,j}(3\tau) \right)_{<5,E>} \)

(Time required to charge to 95% of final value)

In some cases, a global faster convergence can be achieved by (i) scaling eqn. 1 [26] or by (ii) changing...
the slope of the activation function using an annealing like technique [20]. Let us investigate in more detail of these two methods and determine the conditions under which the CNN will work. For both cases consider the steady state condition of eqn. 1. It is important to maintain the same equilibrium state as the transformation may lead to undesired equilibrium points if eqn. 1 is not properly scaled. Let us first address case (i). The steady state scaled solution, by a factor $m$, is shown in eqn. 13:

$$m \frac{x(t)}{R} = m \left[ \sum_{C(k,j) \in N(i,j)} A(i,j;k,l)y(t) + \sum_{C(k,j) \notin N(i,j)} B(i,j;k,l)u_{k,l} + I \right]$$  \hspace{1cm} (13)

Notice that the left hand side of eqn. 13 is equivalent to having $R = R/m$. The scaling gives origin to a new time constant, $\tau < \tau$, which will force the array to converge faster for $m$ greater than 1. Therefore, by simply making all template values including $I$ directly proportional to the scaling factor $m$ and in turn making the state resistor inversely proportional to $m$, a faster convergence is achieved.

For case (ii) let us take a closer look at eqn. 1b. Let us change the slope of this function as follows:

$$y_{i,j}(t) = \frac{m}{2} \left( x_{i,j}^r(t) + \frac{1}{m} - x_{i,j}^r(t) - \frac{1}{m} \right)$$  \hspace{1cm} (14)

Notice in this equation that even though the saturation limits are the same, the state is not, that is, $x_{i,j}^r(t) = mx_{i,j}^r$. This gives the appearance that the state resistor has been reduced by a factor $m$. Unfortunately, with this scheme, the steady state solution is not the same as the one using an overall scaling factor. The effect is present only in the $A$ template and hence unbalances the equation. However, in cases where the equilibrium point is not of interest, only the saturation levels, this scheme is deemed to be good.

### 3.3 Dynamic range

Dynamic range testing is important to determine the maximum and minimum template values that can be used with the current hardware architecture. Ideally, the maximum template entry value is bounded by the linear range of operation of the CNN circuit. In a similar manner, the minimum value is determined by its proximity to the noise floor base. In this section we formalise these two issues. Let us address first the maximum entry value. For this reason let us rewrite eqn. 1 as a voltage based equation assuming multipliers with transconductance $g_m$ and a bias circuit with transconductance $g_v$:

$$C \frac{dv}{dt} = - \frac{v}{R} + (R_{g_m} A) v_g + (R_{g_m} B) v_u + (R_{g_m}) v_l$$  \hspace{1cm} (15)

Let us further assume that the state's linear range is limited by the power supply and denote it as $V_{sat}$. That the linear range limit of the multiplier's product is $V_{sat}$, that the actual saturation limits of the activation function are $V_{sat}$ and $|v| < V_{sat}$ $|v_i(0)| < V_{sat}$. With this last constraint eqn. 15 can be rewritten as follows:

$$C \frac{dv}{dt} = - \frac{v}{R V_{sat}} + (R_{g_m}) A \cdot v_g + (R_{g_m}) B \cdot v_u + \left( \frac{R_{g_m}}{V_{sat}} \right) v_l$$  \hspace{1cm} (16a)

Now assume $K$ nonzero entry values in both the $A$ and $B$ templates. Consider the worst case situation is dominated by the $A$ template because the product is limited by $V_{sat}$ of the activation function. Let us consider now the following inequalities for a maximum voltage based template value $v_c$. Notice first that $V_{sat} > V_{sat} > V_{sat}$ to allow the state node to grow beyond saturation. This condition ensures that once the state node enters the saturation region it remains there. The maximum output voltage, $v_o$, is determined by the voltage of the activation function $V_{sat}$, this is qualified by eqns. 17a and $b$. Hence, the maximum weighted template voltage that one can present to the CNN system, with respect to the other template entries, is obtained by eqn. 17c.

$$V_{sat} \geq K v_c (R_{g_m}) v_c$$  \hspace{1cm} (17a)

$$V_{sat} \geq K v_a (R_{g_m}) V_{sat}$$  \hspace{1cm} (17b)

$$v_a \leq V_{sat} \frac{1}{R_{g_m} K}$$  \hspace{1cm} (17c)

A similar analysis can be carried out for the $B$ template; this yields the following:

$$v_b \leq V_{sat} \frac{1}{R_{g_m} K}$$  \hspace{1cm} (18)

Since it is possible that $v_b > V_{sat}$ we deem the value of $v_b$ as the maximum ideal template entry due to hardware constraints. We stress the word ideal because the inherent circuit nonidealities are not yet taken into account. This voltage based template value still needs to be scaled to the logical template values.

An alternate method for determining the ideal template range involves characterising the linearity of a path through the CNN. A total harmonic distortion (THD) analysis is used to obtain a figure of merit of each cell's linearity. To measure the linearity of the CNN array, we apply a sinewave to each cell input and measure the THD at the cell output. The frequency of the sinewave is chosen to be two decades lower than the bandwidth of the OPAMP to insure that the THD is not degraded by the limited frequency response of the integrator. The magnitude of the input sinewave varied from 1/100 of the maximum linear input range to twice the linear input range. For this test we isolate the path from the cell input, through the multiplier, to the cell output by setting all templates to zero, except for centre $B$ template entry which is set to a value of magnitude equal to the maximum linear input range. The data obtained from the THD analysis is graphed with THD against the input sinewave magnitude to determine the ideal input range. A maximum allowable THD is selected and a line is drawn across the graph generated above. The maximum and minimum input values are located at the intersection of the THD graph and the maximum allowable THD. The pseudocode for the procedure is as follows:

$$A < S,E > \leftarrow 0, B < S,E > \leftarrow \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 0.5 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}, C < S,E > \leftarrow 0$$

$$[y_{i,j}(0)] < S,E > \leftarrow 0$$

$$u_{i,j} < S,E > \leftarrow \sin(\omega t)$$

$$\forall ([y_{i,j}(t)] < S,E > \text{record total harmonic distortion})$$

---

IEE Proc.-Circuits Devices Syst., Vol. 143, No. 5, October 1996
Consider a $1 \times 6$ VLSI CNN array zeroed using the local compensation technique for the THD analysis. The $A$ template and the $I$ value are fixed at zero. The $B$ template has its centre element fixed at 0.5V. The bandwidth of the OPAMP used in the integrator was measured to be 1MHz. As a result, a 10kHz sinewave was chosen for this analysis to prevent the limited integrator bandwidth from influencing the results. The amplitude of the sinewave was varied from 5mV to 1V and the resulting THD was measured. If we specify that the maximum allowable THD is 4%, we fix a minimum and maximum input range by drawing a horizontal line across the graph and projecting the intersection downward. Fig. 9 shows the graph of THD against the input sinewave magnitude. By projecting the intersections of the graph and the maximum THD line we obtain a minimum absolute input voltage value of 0.025V and a maximum absolute input voltage value of 0.45V.

4 Conclusions

In this paper we have presented testing strategies to quantify parametric faults in a fully programmable, two-dimensional cellular neural networks. The techniques allowed us to quantify system offsets, time constant mismatches, nonlinearities in the multipliers and state nodes, and the magnitude of the dynamic range of operation. In certain cases we presented dynamic solutions either by compensating the templates, the input data, and/or the initial condition values to minimise or cancel the undesired effects. We have shown that the dynamic compensation techniques can be applied to any CNN independent of the array size or topology. The results show that the compensation techniques can correct certain dynamic faults without the need or expense of additional hardware.

5 Acknowledgments

The authors are indebted to Gunhee Han for providing us with the design of the VLSI CNN. We would also like to thank Prof. Edgar Sanchez-Sinencio for his advice during the development of this work. This research has been supported by the Office of Naval Research under contract grant N00014-91-1-0516.

6 References