Characterization of Analog Local Cluster Neural Network Hardware for Control

Joaquin Sitte, Liang Zhang, and Ulrich Rueckert

***Abstract*—The local cluster neural network (LCNN) was designed for analog realization especially suited to applications in control systems. It uses clusters of sigmoidal neurons to generate basis functions that are localized in multidimensional input space. Sigmoidal neurons are well suited to analog electronic realization. In this paper, we report the results of extensive measurements that characterize the computational capabilities of the first analog very large scale integration (VLSI) realization of the LCNN. Despite manufacturing fluctuations and the inherent low precision of analog electronics, the test results suggest that it may be suitable for use in feedback control systems.**

***Index Terms*—Analog computation, analog very large scale integration (VLSI), function approximation, neural networks (NNs), radial basis function (RBF) networks.**

I. Introduction

THE massively parallel computations inherent in neural networks (NNs) can only be realized by massively parallel hardware. Although different NN hardware realizations have been proposed in the past 20 years none has found broad application. The vast majority of the many successful NN applications simulate their NNs on conventional sequential computers. Applications of NNs to control stand to benefit most from the fast parallel computations of NN hardware, provided that such hardware can be made at low cost and compact. Among the various types of NNs, feedforward networks are the best understood. Feedforward NNs find wide application as classifiers and multivariate function approximators. Function approximation consists of finding a multivariate function that best interpolates a function that is only known at a set of sample points. With the appropriate setting of the network weight parameters, a feedforward NN can reproduce any well-behaved function. Such adaptive mappings find wide applications in control for mapping sensor readings to actuator signals or for plant identification. With control applications in mind, we choose to realize an NN in analog electronics. The main elements of an NN can be realized with analog electronic circuits consisting only of a few transistors. Large parallelism can be achieved by replicating these small circuits many times in an integrated circuit. An analog NN has the further advantage of interfacing directly to analog sensors and actuators without the need for analog-to-digital (A/D) and digital-to-analog (D/A) conversion, saving space, time, and power. Low precision of analog hardware is often seen as a disadvantage; however, in feedback control precision, it is not critical, because any deviations will be corrected by using feedback as long as the direction is approximately right. This is demonstrated by the many successful *bang–bang* control schemes.

For feedforward NNs, the designer has a choice of two main architectures: multilayer perceptrons (MLP) with sigmoidal activation functions, and the networks that use localized activation functions, such as radial basis functions (RBF) [1]–[4].

In the last decade, several analog NN hardware realizations were reported such as [5]–[14]. These works describe blocks and circuits for hardware realizations of NNs, but the accuracy and robustness of these circuits were not discussed in detail. Therefore, it is difficult to compare these implementations and to assess their applicability.

In this paper, we examine the performance of an analog very large scale integration (VLSI) integrated circuit (IC) realization of the local cluster neural network (LCNN) architecture. The LCNN architecture was proposed by Geva and Sitte [15] and realized in an LCX analog chip by Koerner, Rueckert, and Sitte [16], [17]. Henceforth, we identify this chip as the LCX chip. The LCNN is a special kind of MLP where sigmoidal neurons combine in clusters that have a localized response in input space. LCNNs are more general than RBF NNs and have all the computational advantages of the latter, while retaining the analog VLSI implementation advantages of weighted sums and sigmoids.

In Section II, we give the mathematical definition of the LCNN, and then, follow with the description of its integrated circuit realization in Section III. In Section IV, we report comprehensive measurements of the local cluster output for the full range of its weight parameters. We conclude in Section V by summarizing the main results of this paper.

II. LCNN Definition

Fig. 1 shows the signal flow diagram for a segment of two clusters of an LCNN. Like an MLP, it uses sigmoidal neurons in two hidden layers. Clusters of sigmoids form functions localized in input space, similar to RBFs, but capable of representing a wider range of localized functions. Each neuron in the second hidden layer outputs such a local response function. The LCNN output is a linear combination of localized scalar functions in n-dimensional input space

$$y(\vec{x}) = \sum_{\mu=1}^{m} v_\mu L_\mu(\mathbf{W}_\mu, \vec{r}_\mu, k, \vec{x}) \tag{1}$$

where v_μ is the output weight that determines the contribution to the network output of the μth local cluster $L(\mathbf{W}_\mu, \vec{r}_\mu, k, \vec{x})$. $\mathbf{W}$

Manuscript received December 1, 2005; revised September 20, 2006; accepted February 5, 2007.

J. Sitte and L. Zhang are with the School of Software Engineering and Data Communication, Queensland University of Technology, Brisbane, Qld. 4001, Australia (e-mail: j.sitte@qut.edu.au).

U. Rueckert is with the University of Paderborn, D-33098 Paderborn, Germany.

Digital Object Identifier 10.1109/TNN.2007.899518

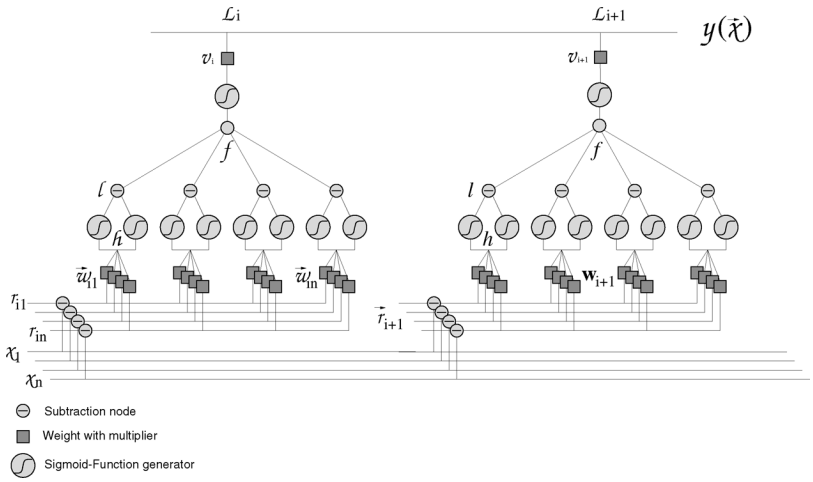

Fig. 1. LCNN with two clusters and four inputs.

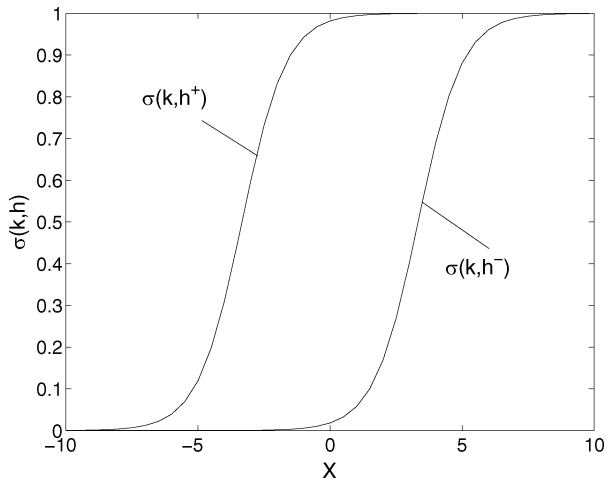

Fig. 2. Two sigmoids: $\sigma(k, h^+)$ and $\sigma(k, h^-)$.

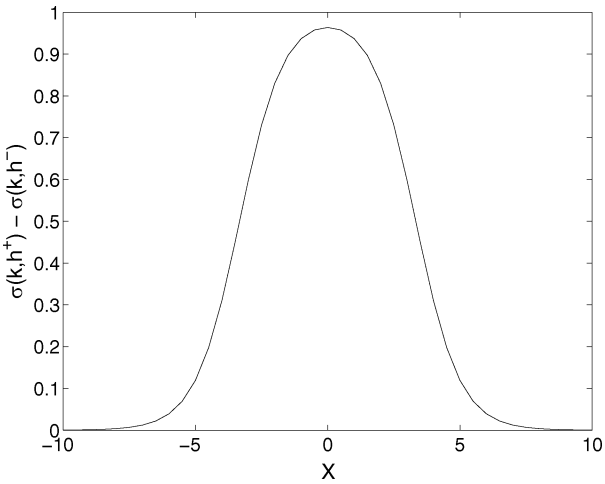

Fig. 3. Ridge $l(\vec{w}, \vec{r}, \vec{x})$: difference of two sigmoids.

is the matrix of weights, $\vec{r}$ is the n-dimensional position vector of the reference point (center) of the cluster, $\vec{x}$ is the n-dimensional input vector, and k is the parameter that controls the steepness of the sigmoid.

Each local function $L(\mathbf{W}, \vec{r}, k, \vec{x})$ is the result of applying a sigmoidal *window* to a sum $f(\mathbf{W}, \vec{r}, k, \vec{x})$ of n-dimensional *ridge* functions

$$L(\mathbf{W}, \vec{r}, k, \vec{x}) = \sigma_0(f(\mathbf{W}, \vec{r}, k, \vec{x}) - b). \tag{2}$$

The constant b allows shifting the function f with respect to the window. $f(\mathbf{W}, \vec{r}, k, \vec{x})$ is a sum of n ridge functions $l(\vec{w}_i, \vec{r}, k, \vec{x})$

$$f(\mathbf{W}, \vec{r}, k, \vec{x}) = \sum_{i=1}^{n} l(\vec{w}_i, \vec{r}, k, \vec{x}). \tag{3}$$

The vector $\vec{w}_i$ is the ith column of the weight matrix $\mathbf{W}$. Ridge functions are built from the difference of two opposing n-dimensional sigmoid functions

$$l(\vec{w}, \vec{r}, k, \vec{x}) = \sigma(k, h^+) - \sigma(k, h^-). \tag{4}$$

The arguments h^+ and h^- for the sigmoids in (4) are chosen to displace their inflection hyperplanes by a distance $d/|\vec{w}|$ to the left and to the right of the position $\vec{r}$ along the direction of $\vec{w}$, as shown in Fig. 2

$$h^+ = \vec{w}^\top(\vec{x} - \vec{r}) + d \tag{5}$$

$$h^- = \vec{w}^\top(\vec{x} - \vec{r}) - d. \tag{6}$$

The difference of the two sigmoids determines the shape of the ridge function (4), as shown in Fig. 3.

For $\sigma(k, h)$, we choose the logistic sigmoid function

$$\sigma(k, h) = \frac{1}{1 + e^{-kh}} \tag{7}$$

where h is an affine transform of $\vec{x}$

$$h = \vec{w}^\top(\vec{x} - \vec{r}) + d. \tag{8}$$

We can rewrite the sigmoid function as

$$\sigma(h') = \frac{1}{1 + e^{-h'}} \tag{9}$$

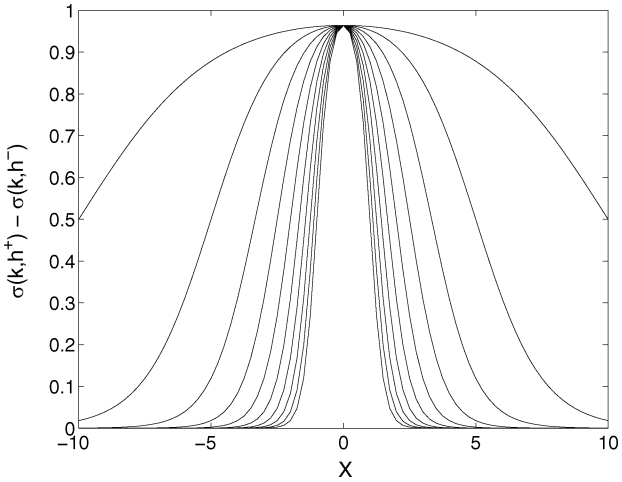

Fig. 4. Dependence of ridge shape on $|\vec{w}|$ for constant $d = 1$. $|\vec{w}| = (0.1 - 1)$ in steps of 0.1.

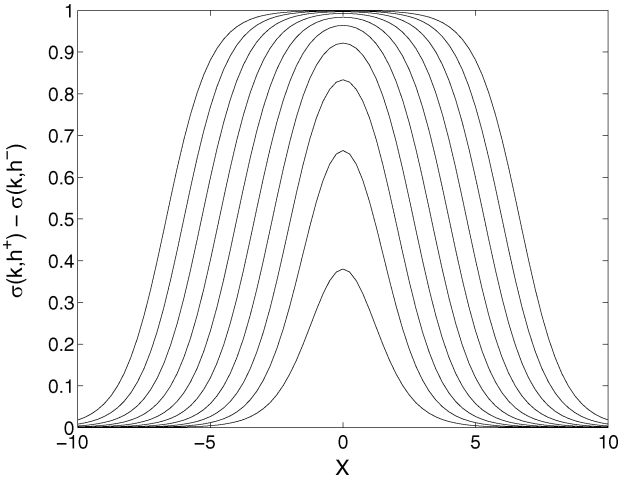

Fig. 5. Dependence of the ridge shape on d with fixed w for $w = 0.3$, $d = (0.2, 2)$ step 0.2.

where

$$h' = k\vec{w}^{\top}(\vec{x} - \vec{r}) + kd \tag{10}$$

$$= \vec{w'}^{\top}(\vec{x} - \vec{r}) + d'. \tag{11}$$

The weights $\vec{w}$ and d are now scaled by k. Because the weights will be determined by a training process, it does not matter whether the training is for $\vec{w}$ or $\vec{w}'$. Equation (11) is of the same form as (8), only the values of the parameters will be different; therefore, we can use the scaled (8) ignoring the primes. The expression for h can be rewritten as

$$h = |\vec{w}| \left(\hat{w}^{\top}(\vec{x} - \vec{r}) - \frac{d}{|\vec{w}|} \right) \tag{12}$$

showing that h represents $|\vec{w}|$ times the distance from $\vec{x}$ to the inflection hyperplane of the sigmoid.[1] $|\vec{w}|$ determines the steepness of the sigmoid and $d/|\vec{w}|$ determines the displacement of the inflection hyperplane from the reference position $\vec{r}$. The magnitude of the weight vector controls both the steepness and the position of the inflection plane. Thus, we have two parameters $|\vec{w}|$ and d that determine two quantities: the steepness and the position of the inflection hyperplane of the sigmoid.

The two parameters $|\vec{w}|$ and d in combination determine the shape of the ridge function. Fig. 4 shows the variation of ridge cross-section shape when d is fixed at 1 and $|\vec{w}|$ is varied. The width of the ridge decreases as the length $|\vec{w}|$ of the weight increases. Fig. 5 in turn shows how the ridge shape varies with d for constant $|\vec{w}|$.

The weight vector $\vec{w}$, and therewith its length $|\vec{w}|$, is determined by training. The parameter d plays the role of a *shape* parameter that we consider fixed by design. The value $d = 1.0$ gives a bell shape close to a Gaussian function. Fig. 5 shows that larger values of d produce trapezoidal or box-like functions useful for classification tasks. Therefore, the LCNN can represent a wider range of functions than an RBF network.

The ridge functions derive their name from their shape in n-dimensional input space. Fig. 6 shows a ridge $l(\vec{w}, \vec{r}, \vec{x})$ in 2-D.

[1]The inflection hyperplane is the locus of $\vec{x}$ such that the argument of the sigmoid function (7) is zero.

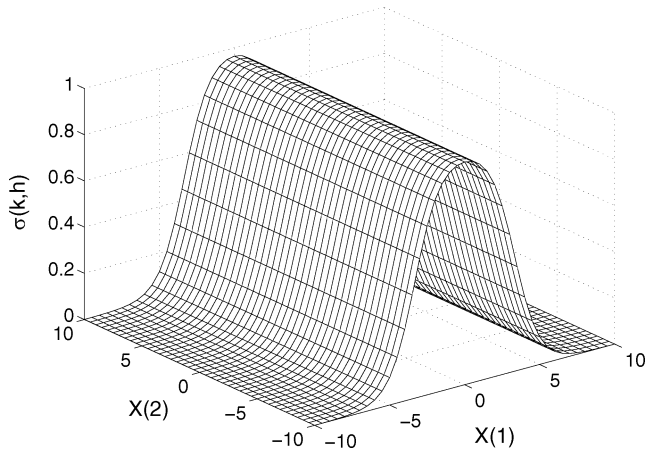

Fig. 6. Ridge function (difference of two displaced sigmoids) in 2-D.

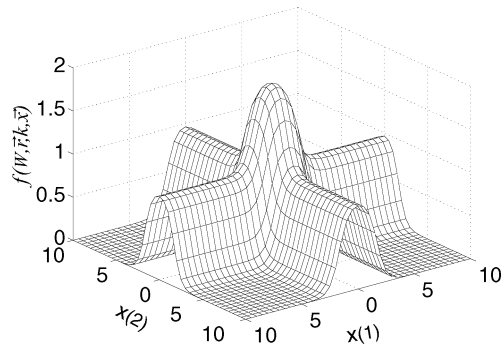

Fig. 7. Addition of two ridges in 2-D. $f(\mathbf{W}, \vec{r}, k, \vec{x})$.

Adding n n-dimensional ridge functions that have the same $\vec{r}$ but different vectors $\vec{w}$ produces the function $f(\mathbf{W}, \vec{r}, k, \vec{x})$ [see (3)] that has a peak around $\vec{r}$ where the ridges intersect. Fig. 7 shows this for 2-D. We call the n ridge functions combined in this way a *local cluster*.

Removing the ridges that radiate outwards from the center $\vec{r}$ leaves the peak, which is the localized function we are after. The removal is done by applying an offset sigmoidal *windowing* function to f. Fig. 8 shows the result $L(\mathbf{W}, \vec{r}, k, \vec{x})$ in 2-D.

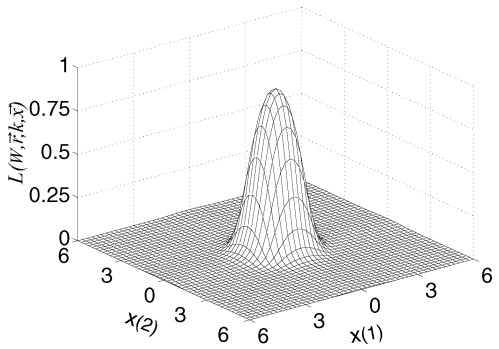

Fig. 8. local function $L(\vec{w}, \vec{r}, k, \vec{x})$ resulting after removing the outward radiating ridges.

III. Hardware Implementation of LCNN

A hardware realization of the LCNN requires modules for carrying out the following operations on the input signals:

- subtraction of the position vector $\vec{r}$ of each cluster from the input vector $\vec{x}$;
- computation of the dot product of each of the n weight vectors in the weight matrix of each cluster with the result from the previous step: $\vec{w}^{\top} \cdot (\vec{x} - \vec{r})$;
- calculation of two displaced sigmoid functions for each dot product: $\sigma(h^{+})$ and $\sigma(h^{-})$;
- subtraction of the two sigmoids to get the ridge function $l(\vec{w}, \vec{r}, \vec{x}) = \sigma(h^{+}) - \sigma(h^{-})$;
- summation of the ridge function outputs $f(\mathbf{W}, \vec{r}, \vec{x}) = \sum_{i=1}^{n} l(\vec{w_i}, \vec{r_i}, \vec{x_i})$;
- passing the summation of the ridge functions through a sigmoidal windowing function to extract the localized peak resulting from the overlap of the ridges and removing of the ridges: $L(\mathbf{W}, \vec{r}, \vec{x}) = \sigma_0(f(\mathbf{W}, \vec{r}, \vec{x}) - b)$;
- multiplication of each output of the windowing module by the corresponding output weight v;
- summation of the resulting values in the last operation to calculate the LCNN output $y(\vec{x}) = \sum_{\mu=1}^{m} v_\mu L_\mu(\mathbf{W}_\mu, \vec{r}_\mu, \vec{x})$.

All of mathematical functions used in the LCNN can be realized in analog current mode complementary metal–oxide–semiconductor (CMOS) VLSI circuits. By current mode, we mean that currents, not voltages, represent the signals. The LCX integrated circuit, described in this paper, is such an analog electronic implementation of the LCNN. The LCX chip was designed by combining simple basic circuits with transistors working in the subthreshold mode. The major blocks of the LCX chip are the multiplier matrix, the ridge generator circuits, and the output sigmoid circuit. For the weights, however, we opted for digital storage because an analog storage technology such as floating gate transistors was beyond our reach. The prototype integrated circuit we describe in this paper consists of eight equal clusters. Each cluster has six inputs, one output and 8-bit digital weight storages. This number of inputs and clusters is already adequate for many control applications. Fig. 9 shows the block diagram of the LCX chip and Fig. 10 shows the structure of a cluster.

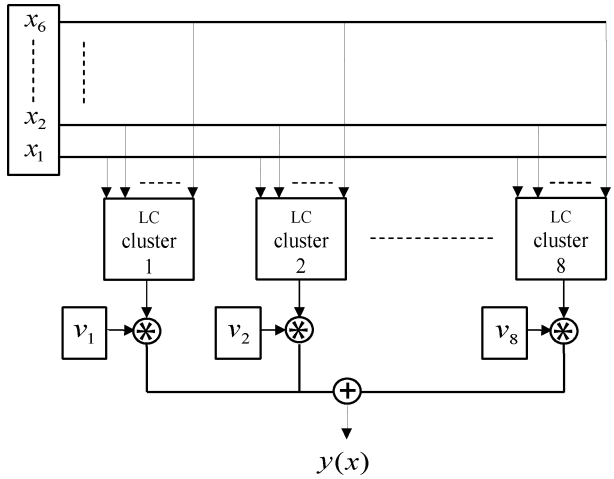

Fig. 9. LCNN with eight clusters on the LCX chip.

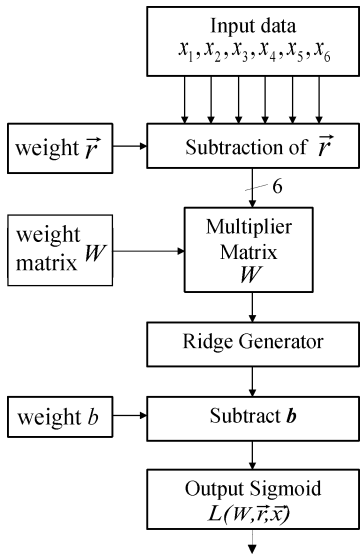

Fig. 10. Structure of a local cluster.

Each cluster has 45 digital weights grouped as follows:

$$\text{cluster center vector: } \begin{bmatrix} r_1 & r_2 & r_3 & r_4 & r_5 & r_6 \end{bmatrix}$$

$$\text{weight matrix: } \begin{bmatrix} w_1 & w_2 & w_3 & w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 & w_{10} & w_{11} & w_{12} \\ w_{13} & w_{14} & w_{15} & w_{16} & w_{17} & w_{18} \\ w_{19} & w_{20} & w_{21} & w_{22} & w_{23} & w_{24} \\ w_{25} & w_{26} & w_{27} & w_{28} & w_{29} & w_{30} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} & w_{36} \end{bmatrix}$$

$$\text{bias: } \begin{bmatrix} b_{hi} & b_{lo} \end{bmatrix}$$

$$\text{output weight: } \begin{bmatrix} v \end{bmatrix}.$$

The bias (or offset) b consists of two registers: the 8-bit wide b_{lo} and the 3-bit extension b_{hi}, which gives a total range for b of (-1023, 1023). The range of values for weight r, w, and v is (-127, 127). In total, the LCX chip has 360 weight registers (8 clusters $\times$ 45 weights). The storage locations do not have addresses. The digital weight store is a single big shift register,

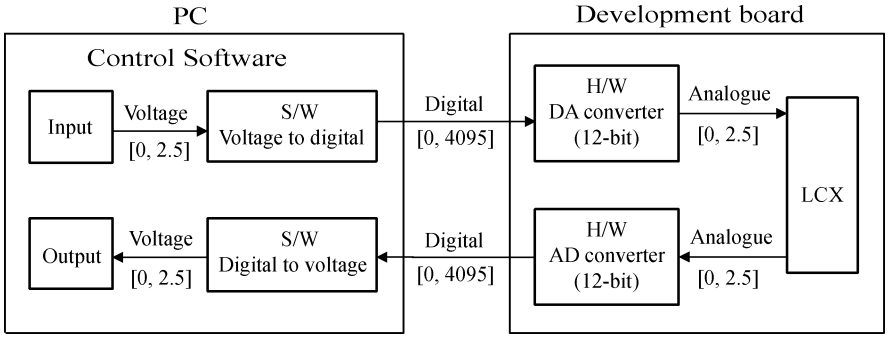

Fig. 11. LCNN chip test setup.

where the values are shifted in and out serially. The indices in the previous matrix indicate the weights' position in the shift register.

The chip's inputs and output is voltage for ease of interfacing. The voltage range for each input is from 1 to 2 V and each cluster receives the same six inputs. All clusters have the same functionality. The chip output is the weighted sum of all cluster outputs.

IV. Test Procedures and Results for LCNN Chip

Analog circuits are much more susceptible to manufacturing inaccuracies and fluctuations than digital circuits. Any deviation from design specification in a circuit component translates directly into a deviation from the designed signal input–output behavior. Because of transistor parameter fluctuations, circuits will not only deviate from their expected design behavior but each one will deviate differently. Signal offsets, deviations, and distortions are the result. The LCX chip contains many identical circuits that will all behave slightly different. Therefore, the first task with the new LCX chips was to thoroughly test them to determine the absolute deviations from the design specification as well as the relative intrachip and interchip fluctuations. The main results of these tests are summarized and discussed in this section.

A. Test Setup for the LCNN Chip

Fig. 11 shows a block diagram of the test setup for the LCNN chip. For testing the LCX chip, we used a purposedly built development board that can host up to four LCNN ICs and it connects to a personal computer over the parallel port. The LCNN development software on the PC controls the measurement process, loading weights onto the chips, generating input signals for the chips, and capturing the outputs. The development board provides the D/A conversion of the digital input values generated on the PC and A/D conversion of the outputs. In addition to the cluster outputs, diverse strategic test points in the network can be read.

B. Test Procedure for the LCNN Chip

The aim of the tests is first to determine how closely the input–output transfer mapping of the chip matches the ideal mathematical model described in Section II and, second, the amount of variation found in the local clusters within a chip (intrachip) and between chips (interchip). The input space for each cluster is a 6-D cube. The output of each cluster depends on the 45 digital weight parameters described in Section III. It is practically impossible to test the chip exhaustively over the 45-D parameter space and, therefore, a test strategy is required that is capable of characterizing the chip's capabilities appropriately without having to sample exhaustively the parameter space.

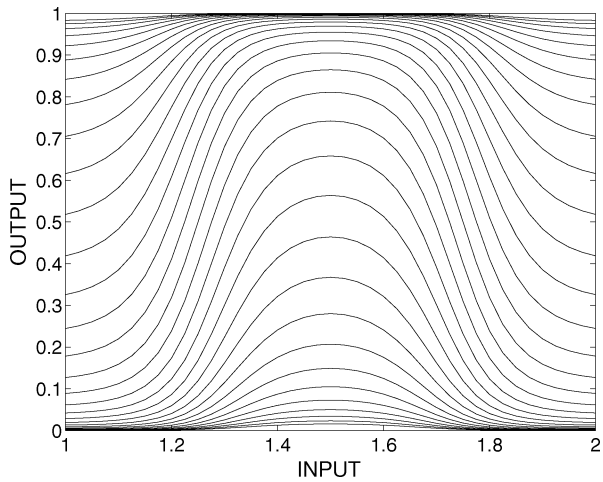

Fig. 12. Effect of the bias b of the windowing sigmoid function on the cluster output in an ideal cluster. $b = (-1.0, 2.0)$ with 0.1 interval.

The testing strategy we follow emphasizes the visualization of the key functional dependencies that determine the network output. The tests consist of measuring the network output as a function of the network inputs at selected points in parameter space. Inputs will be varied along specific trajectories, or over 2-D hyperplanes, in input space. Mostly, these will be straight lines parallel to one of the axes of the input space or planes aligned with the axes.

C. Positioning of the Windowing Sigmoid for the Cluster Output

The most critical parameter in a cluster is the offset b of the *windowing* sigmoid function in (2). If this *window* is misplaced, then there will be a distorted output from the cluster, or no output at all. Fig. 12 shows the effect on the output of varying the bias b of the windowing sigmoid function in the ideal network. As b increases, the window shifts down and the output peak moves into the window from below growing until it is squashed (cut) at the top. Then, the ridges appear and move up until they reach

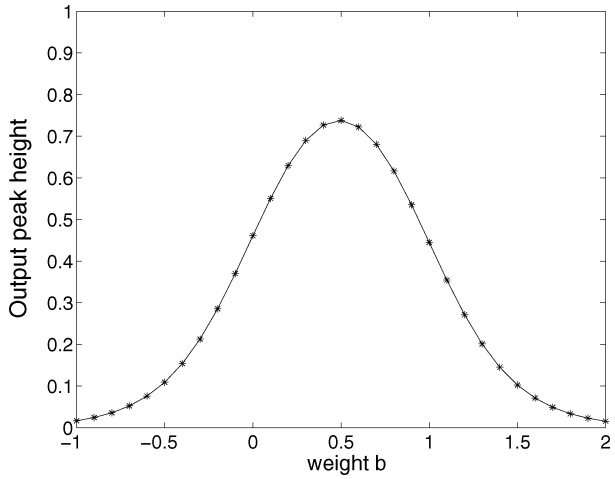

Fig. 13. Dependence of cluster output height on b.

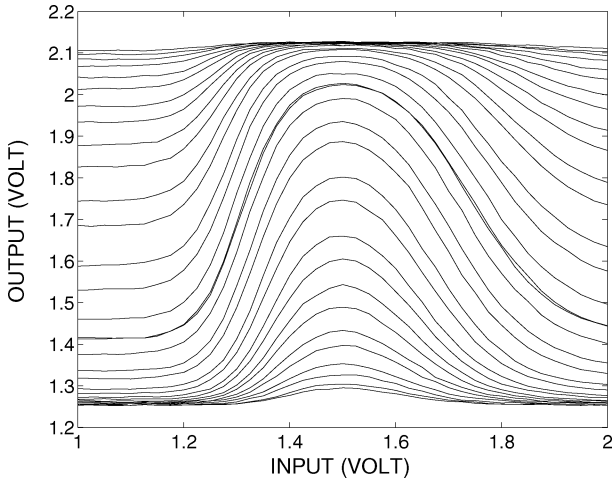

Fig. 14. Effect of the bias b of the windowing sigmoid function on a cluster output. $b = (-960, 960)$ with interval of 64.

the top of the window. The amplitude of the windowing sigmoid should be about twice the amplitude of the ridge sigmoid. This allows positioning of the center of the windowing sigmoid at the maximum of the sum of the ridge functions [function f in (2)] so that the top of the local cluster function falls into the linear part of the windowing sigmoid and is not distorted. Fig. 13 shows that in the ideal case the maximum height of the cluster output peak is attained at the inflection point of the windowing sigmoid. For the ideal network, the value of b is the same for all clusters and can be calculated. It only depends on the number of active inputs, that is, on how many ridges are added. On the chip, the best value for b has to be found experimentally for each cluster. There is no dependency on the number of inputs because all inputs are always active. When there are fewer input signals than input lines on the chip, the unused inputs have to be held at the neutral input value of 1.5 V.

Fig. 14 shows how the output varies with the offset b as measured on the chip. The optimal value of b can be read off from Fig. 15 as the value of b where the output height has its maximum. At that point, the top has not been flatted and the tails are just about to appear.

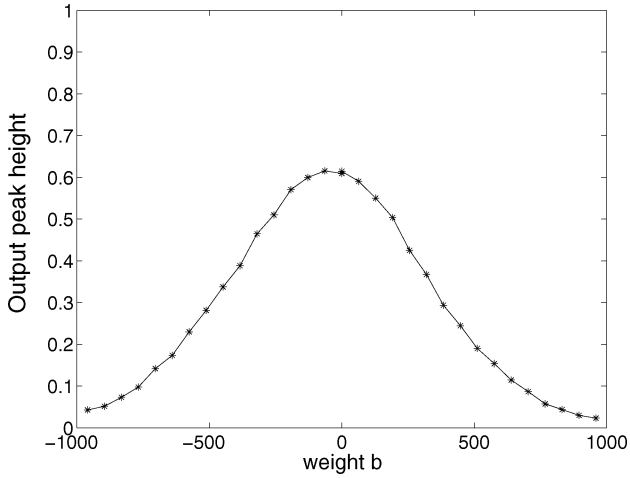

Fig. 15. Dependence of cluster output height on b.

The results show that the output sigmoid can be positioned quite accurately. Variations in the output sigmoid amplitude do not affect the output peak because only the lower half of the input range is used. Variations in slope affect the height of the output peak but this can be compensated with the adaptation of the output weight v of the cluster.

D. Output Dependence on the Weights

The shape of the ridge functions l, built by subtracting two displaced sigmoids as defined in (4), determines the shape of the localized cluster output function. The individual ridge generators (see Fig. 1) can be tested by setting the weight matrix of the cluster to a diagonal matrix. In that case, the ridges are aligned with the axes in input space and each ridge depends on a single parameter. By scanning the range of one of the inputs, while holding all other inputs constant, the network output follows the cross section of the ridge associated with that input. The different ridges of a cluster can be compared by setting all weights on the diagonal to the same value w. Furthermore, if the values of the output weight v for all the clusters are the same, ridges can be compared across clusters.

The shape of the ridge functions tested in this way is affected by the error in the position of the peak of the local cluster output. In order to obtain the maximum amplitude for the ridge function, the cross section has to go through the peak in input space. In the mathematical model, this is the case if $\vec{r} = \vec{0}$. On the chip, there is no guarantee that the cross section passes through the peak as the maxima of the ridges may be displaced from their ideal positions. Instead, the components of $\vec{r}$ have to be set at values such that the maximum of the ridge function is at the center of the ridge's input range.

Fig. 16 shows the ridge shapes from four different clusters (clusters 4 to 7) for all input channels (channels 0 to 5) when weight w is 96, v is 96, and b is -640. The values for r were chosen for each ridge as to position the maximum of the ridge function at the center of the input range. As can be seen, the widths and heights of the ridge functions vary considerably on the same chip.

The statistical analysis of the variations of the ridge function could be done by extracting a set of characteristic parameters

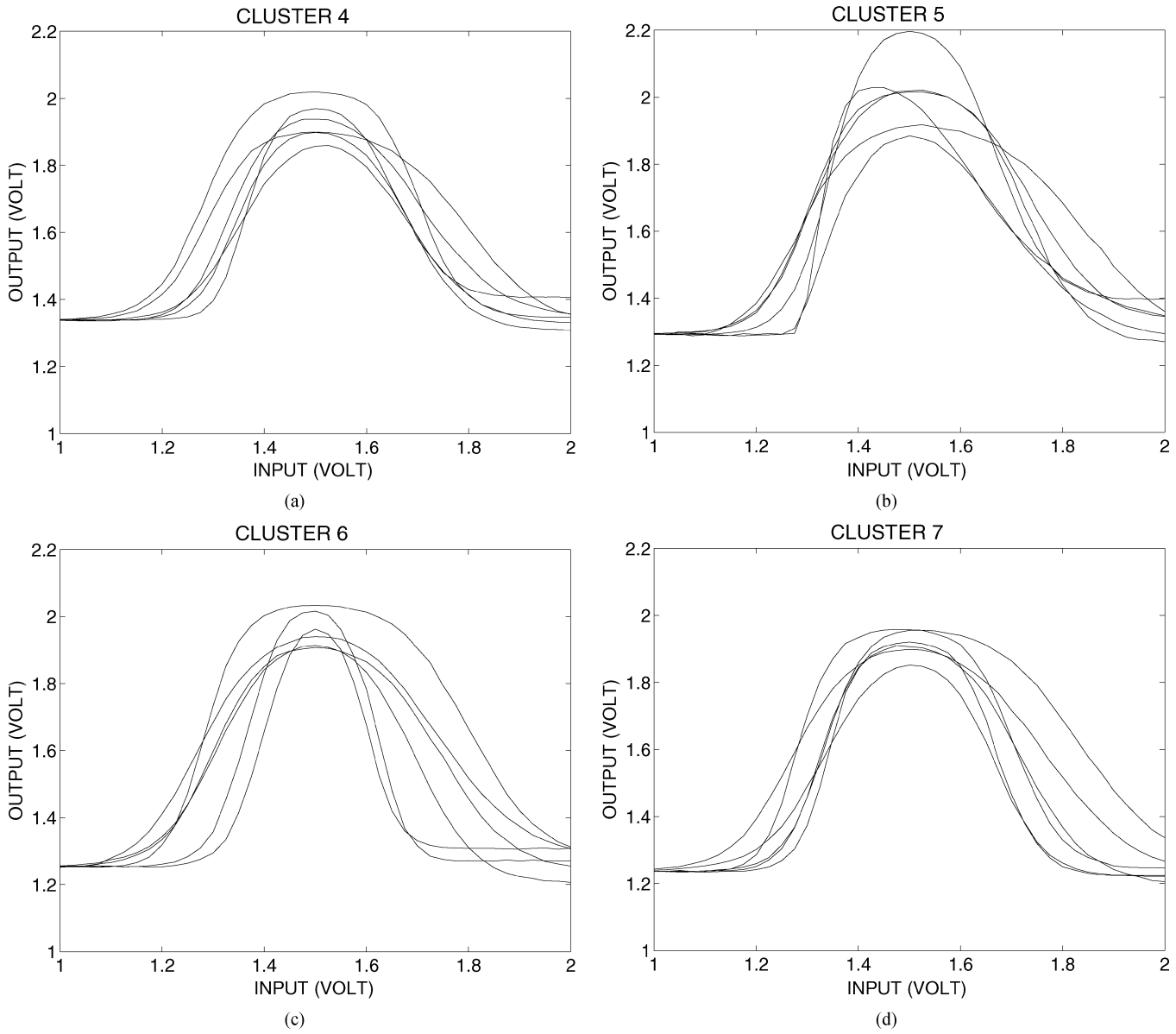

Fig. 16. Fluctuations of the ridge cross sections in different clusters and input channels on the same LCX chip. The diagonal elements are all $w = 96$, the output weights were also 96, and $b = -640$.

from the measured functions. These could be, for example, the parameters that will best fit a ridge function [see(4)] to the measured functions. Instead, we choose a simple characterization in terms of the height and width of the measured ridge function. Figs. 17 and 18 show the results for the ridges in Fig. 16.

The magnitude of the weight vector affects the width of the ridge function. Fig. 4 shows how in the mathematical model the width of the bell-shaped ridge functions decreases as the magnitude of the weight vector increases. This dependence on $|\vec{w}|$ can be observed on the chip by scanning, as before, the inputs for different values w of elements in the diagonal weight matrix. Figs. 19 and 20 show a typical result for one ridge on the chip.

The height of the output peak of the local function can be adjusted with the weight v. In this way, differences in the height of the ridges can be compensated with v. Fig. 21 shows how a cluster output function changes with v. Fig. 22 shows that the amplitude of the cluster output is closely linear as it should be.

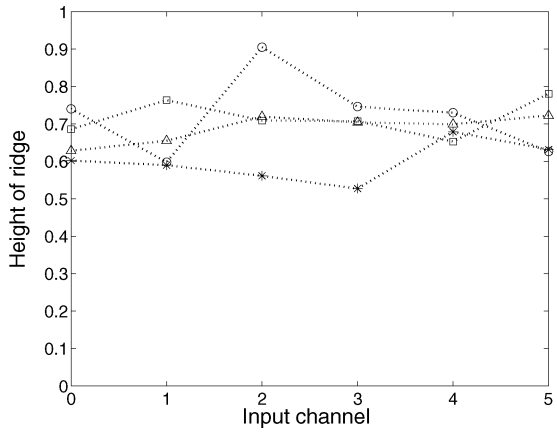

Fig. 17. Output fluctuations in the height of ridges in four clusters on the same chip. $*$ cluster 4, $\circ$ cluster 5, $\square$ cluster 6, and Δ cluster 7.

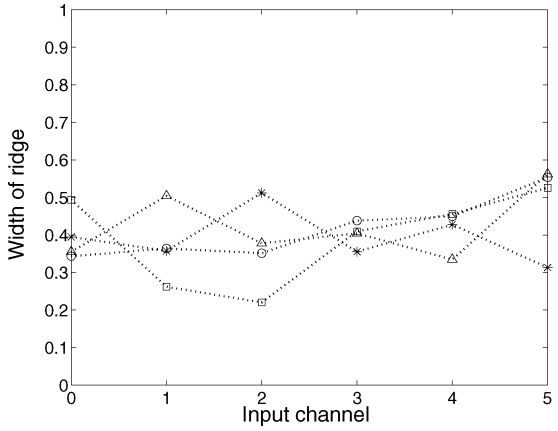

Fig. 18. Output fluctuations in the width of the ridges in four clusters on the same chip. $*$ cluster 4, $\circ$ cluster 5, $\square$ cluster 6, and Δ cluster 7.

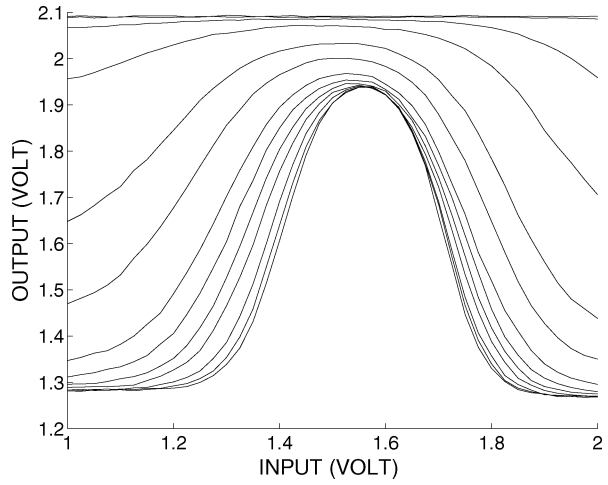

Fig. 19. Dependence of cluster output on weight w in cluster 7. $w = (32, 127)$ with intervals of 8.

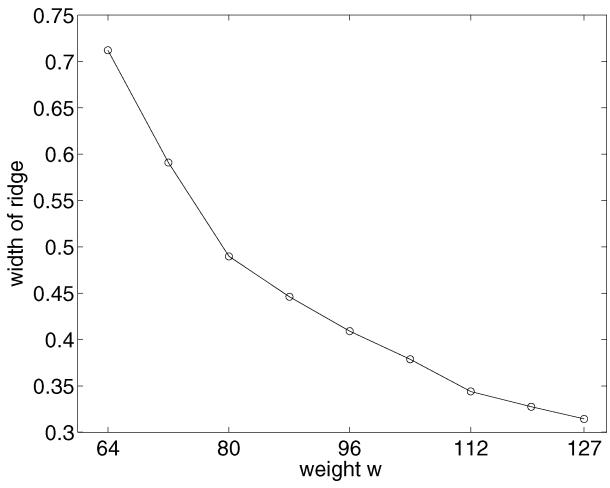

Fig. 20. Width of the ridge as a function of $|\vec{w}|$.

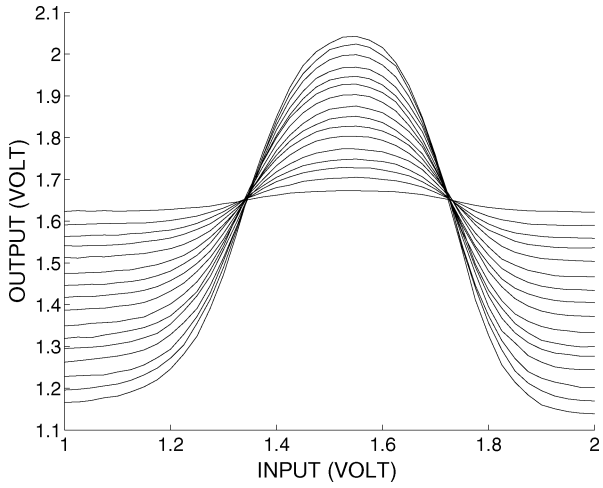

Fig. 21. Dependence of cluster output on weight v in cluster 7. $v = (8, 127)$ with interval of 8.

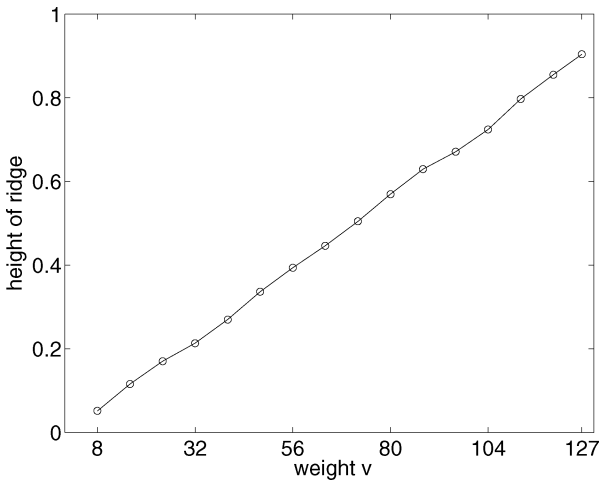

Fig. 22. Amplitude of output peak as function of v.

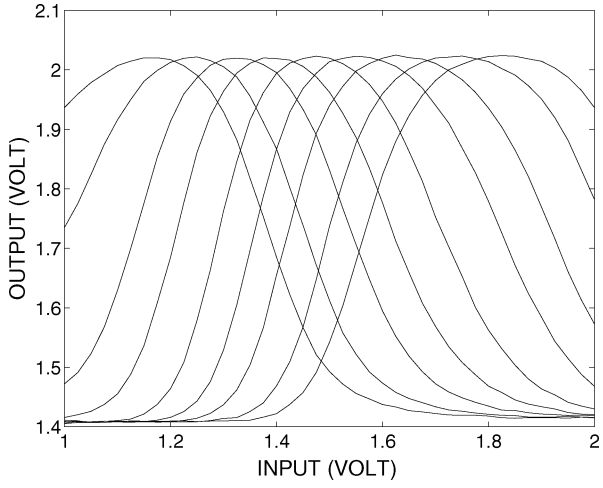

Fig. 23. Dependence of output with the variation of weight r in cluster 4. $r = (-127, 127)$ with interval of 32.

As mentioned before, the position of the peak is at the intersection of all ridges, and therefore, the position depends on the position of the maxima of the ridges. According to the mathematical model for $\vec{r} = \vec{0}$, the peak of the output of a local cluster is at the origin of input space. In the LCX chip, the maxima of the ridge functions are shifted from this reference position by various amounts. Two test are indicated here. First is linearity of the displacement dependence on the corresponding component

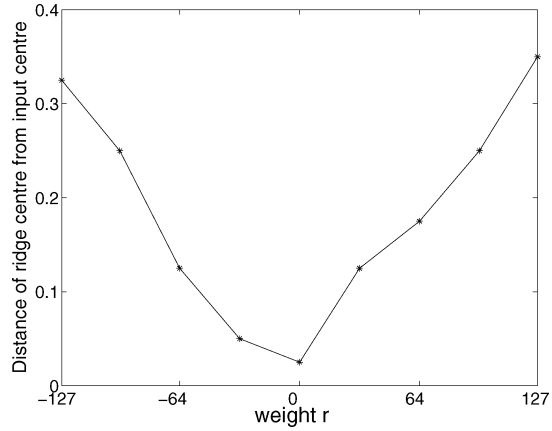

Fig. 24. Distance of ridge maximum from input range center over the full range of r.

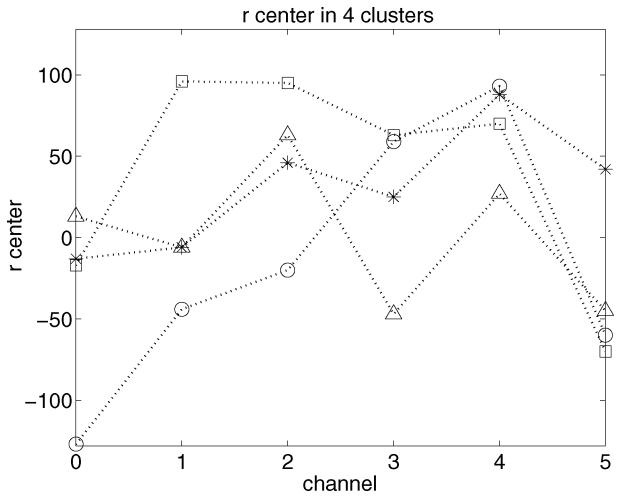

Fig. 25. Center of weight r in different clusters. $*$ cluster 4, $\circ$ cluster 5, $\square$ cluster 6, and $\triangle$ cluster 7.

of r, and second the offset of the position from the reference position. This can be measured in the same way as before for each ride function.

Fig. 23 shows the shape and position of a ridge function over the whole range of r. Fig. 24 shows that the displacement of the peak of the ridge function is a fairly linear function of the value of the ridge's component of $\vec{r}$. Fig. 25 shows the offset required for the ridge's component of $\vec{r}$ such that the peak of the ridge function is in the center of the input range. There are considerable displacements; however, they will be compensated by training the components of $\vec{r}$.

E. Visualization in 2-D

The output of the network can be visualized by varying two inputs, and holding the remaining inputs constant. Fig. 26(a) and (b) shows the 2-D LCNN output in channel 4 and channel 5. The bump increases in height when the output weight v increases. Fig. 27(a) to (d) shows the effect of b in 2-D. Finally, Fig. 28(a) and (b) shows, in 2-D, the LCX chip's capability of representing general functions. These functions were obtained by training the LCX chip with two different training sets.

F. Dynamic Response

All the data presented so far characterize the static behavior of the LCX chip. For control applications, it is of interest to know how fast the output can follow changes in the input and how big the signal propagation delay is. The measurement of the dynamic response of the circuit is complicated by the nonlinearity in the sigmoids. Simulations showed that the various subcircuits have quite different frequency response. While the multiplier circuit output only dropped between 1% and 5% at 1 Mhz compared to the static output, this occurred at 250 kHz for the ridge generators and at 125 kHz for the output sigmoid.

The frequency response of the LCX chip was tested with a sinusoidal input signal of 1 V over the range of 10–150 kHz. The 3-dB attenuation for the multipliers is around 150 kHz. For the cluster output, it was measured at 50 kHz. There is no attenuation of the output at up to 15 kHz. The input-to-output propagation delay was measured to be about 1 μs. The limit of 15 kHz for the undistorted operation is due to the low output currents available to buffer external capacitances.

The frequency of 15 kHz seems small compared to gigahertz clock speeds for digital circuits; however, it has to be remembered that in one cycle the whole input range is transversed twice. Because of the nonlinearities, the output contains frequencies up to ten times the input frequency. According to the Nyquist sampling theorem, a digital circuit would have to sample the output at 300 kHz. This leaves a digital processor around 3 μs to do all the computations for all eight clusters on the chip. Estimating the number of operations conservatively at around 1000, a piplined processor with a clock speed of 300 MHz would be needed to the same job as the LCX chip, and at a much higher power consumption.

V. Conclusion

The software version of LCNN has proven its versatility in function approximation and classification tasks. The mathematical operations of the LCNN can be implemented in analog NN hardware allowing the parallel computation inherent in the NN model to be realized at low cost. However, analog circuits are susceptible to manufacturing fluctuations and noise. We investigated the accuracy and precision of an analog hardware realization of the LCNN. The results give an indication of the magnitude and nature of the deviations of the function performed by the integrated circuit from the desired ideal mathematical behavior. We found that although the computations of the analog NN match the mathematical model quite accurately, the parameters that characterize the clusters have large variations across the clusters on the same chip. Therefore, it is not possible to predict the output of a cluster for a given set of weights. However, values for the weights can be found so that cluster output function closely matches the mathematical model. The implication of this is that sets of weighs are not transferable between the different LCX chips, and therefore, each chip has to be trained separately to approximate a given function. With his proviso, the analog LCNN provides a fast, low-power, and low-cost hardware solution for function approximation. The training algorithms and results for the LCX chip are the subject of a separate forthcoming paper.

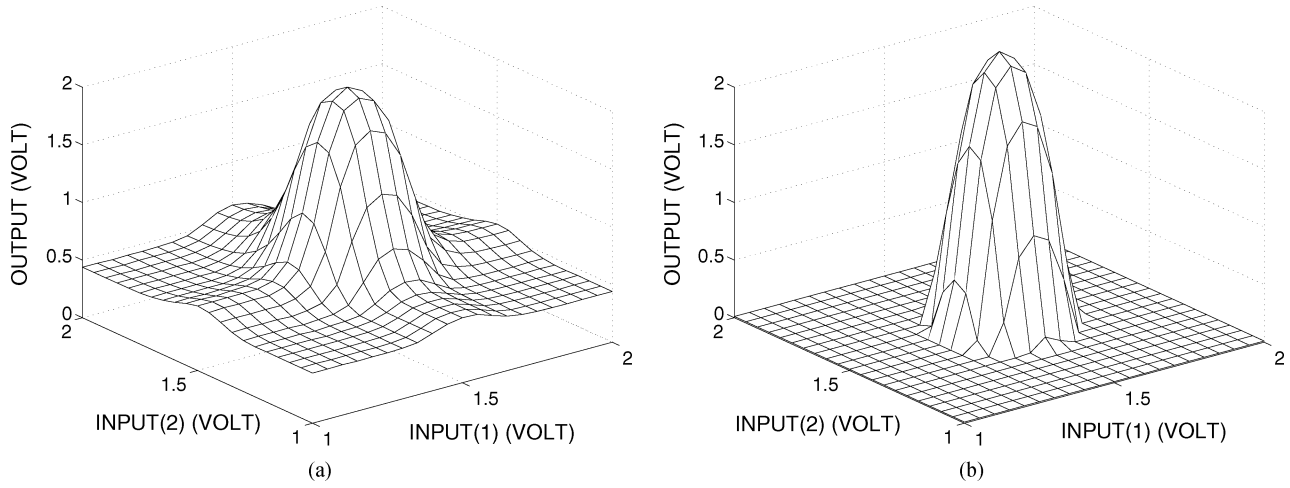

Fig. 26. Change in network output with increasing value of v. (a) The 2-D LCNN in $v = 64$, $w = 127$, and $b = -896$. (b) The 2-D LCNN in $v = 127$, $w = 127$, and $b = -896$.

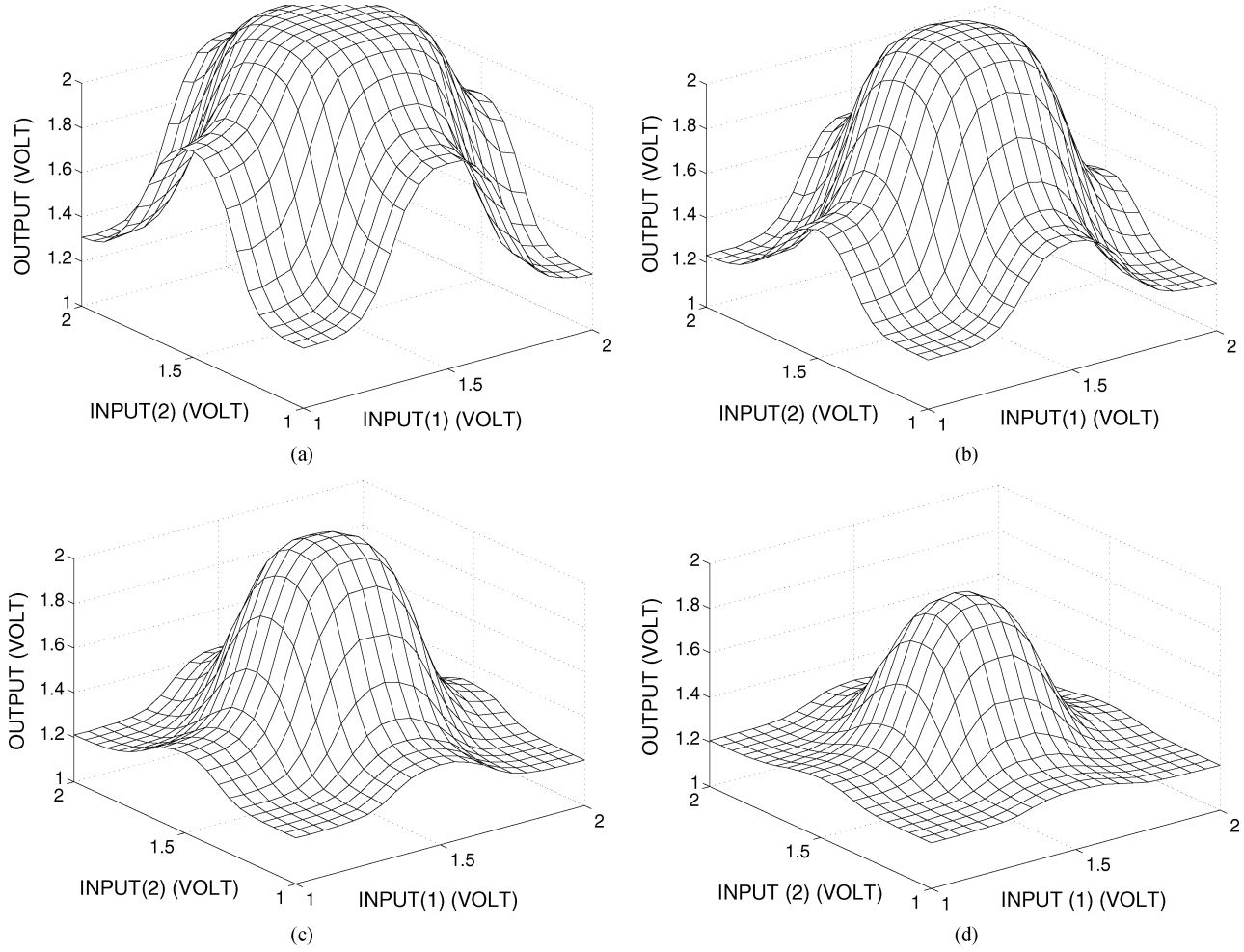

Fig. 27. Cluster output in 2-D for different positions of the output sigmoid in cluster 5 and channel 3 and 4. (a) The 2-D LCNN in $v = 96$, $w = 96$, and $b = -128$. (b) The 2-D LCNN in $v = 96$, $w = 96$, and $b = -384$. (c) The 2-D LCNN in $v = 96$, $w = 96$, and $b = -640$. (d) The 2-D LCNN in $v = 96$, $w = 96$, and $b = -896$.

The test results indicate that the LCX chip is suitable for feedback control applications. As an example, consider the well-known cartpole control problem. The purpose of the controller is to balance an inverted pendulum mounted on a mobile platform (cart) such that balancing is achieved at prescribed target position for the cart. This control task has four-state variable, pole angle, pole angular velocity, cart position, and cart speed. The control output is the acceleration of the cart. A simple state-

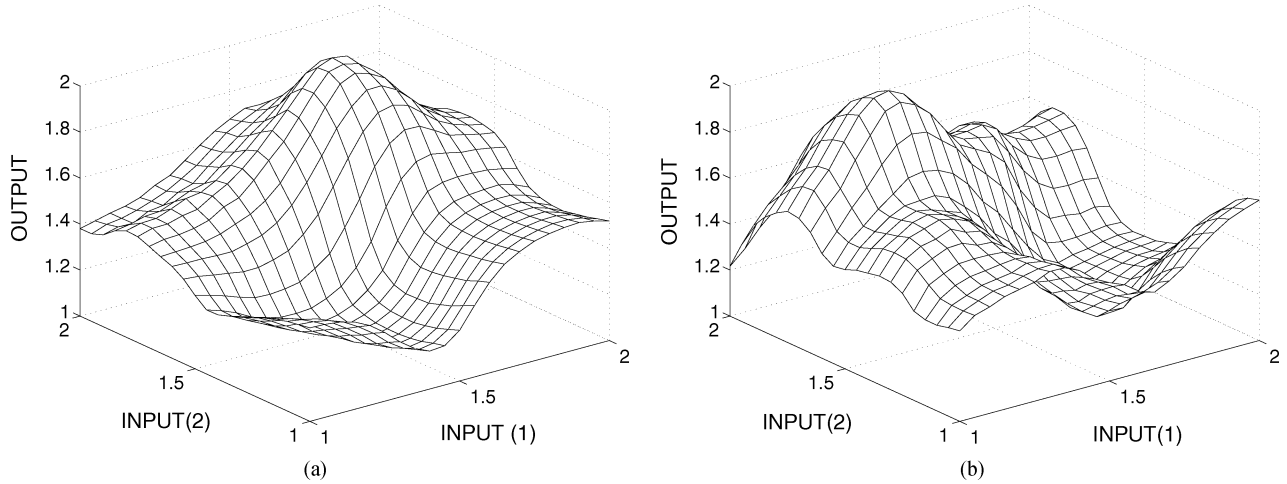

Fig. 28. General functions formed by linear combination of several local clusters. (a) Linear combination of two local cluster functions in 2-D. (b) Linear combination of six local cluster functions in 2-D.

space feedback controller of the form $a = f(\theta, \omega, x, v)$ [18] has been shown. The function is amenable to be generated by the LCX chip. These variables can be derived from analog sensors and input directly to the LCX chip and the output can be used directly to a power amplifier for a direct current (dc) motor. No digital processor is needed.

REFERENCES

[1] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 78, no. 9, pp. 1481–1497, Sep. 1990.

[2] J. Park and I. Sandberg, "Universal approximation using radial-basis-functions networks," *Neural Comput.*, pp. 246–257, 1991.

[3] R. Sanner and J. Slotine, "Gaussian networks for direct adaptive control," *IEEE Trans. Neural Netw.*, vol. 3, no. 6, pp. 837–863, Nov. 1992.

[4] S. Lee and M. K. Rhee, "A Gaussian potential function network with hierarchical self-organizing learning," *Neural Netw.*, vol. 4, pp. 207–224, 1991.

[5] M. Jabri and B. Flower, "Weight perturbation: An optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks," *IEEE Trans. Neural Netw.*, vol. 3, no. 1, pp. 154–157, Jan. 1992.

[6] F. Stupmann, S. Rode, W. Schmidt, and N. Fredrich, "Analog backpropagation chip with on-chip-learing," in *Proc. 5th Int. Heinz Nixdorf Symp.: Autonom. Minirobots Res. Edutainment (AMiRE), HNI-Verlagsschriftenreihe*, 2001, pp. 263–268.

[7] G. Cauwenberghs, "A learning analog neural network chip with continuous-time recurrent dynamics," in *Advances in Neural Information Processing Systems*. New York: Morgan Kaufmann, 1994, vol. 6, pp. 77–86.

[8] G. Cairns and L. Tarassenko, "Learning with analogue VLSI MLPs," in *Proc. 4th Int. Conf. Microelectron. Neural Netw. Fuzzy Syst. (Microneuro)*, Los Alamitos, CA, 1994, pp. 67–76.

[9] T. Morie and Y. Amemiya, "An all-analog expandable neural network LSI with on-chip backpropagation learning," *IEEE J. Solid-State Circuits*, vol. 29, no. 9, pp. 1096–1093, Sep. 1994.

[10] M. R. Belli, M. Conti, and C. Turchetti, "Analog Brownian weight movement for learning of artificial neural networks," in *Proc. Eur. Symp. Artif. Neural Netw.*, 1995, pp. 75–80.

[11] M. Mestari, "An analog neural network implementation in fixed time of adjustable-order statistic filters and applications," *IEEE Trans. Neural Netw.*, vol. 15, no. 3, pp. 766–785, May 2004.

[12] J. Liu, M. Brooke, and K. Hirotsu, "A CMOS feedforward neural-network chip with on-chip parallel learning for oscillation cancellation," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1178–1186, Sep. 2002.

[13] V. Beiu, J. M. Quintana, and M. J. Avedillo, "VLSI implementations of threshold logic – A comprehensive survey," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1217–1243, Sep. 2003.

[14] P. Fleury, H. Chen, and A. F. Murray, "On-chip contrastive divergence learning in analogue VLSI," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2004, pp. 1723–1728.

[15] S. Geva, K. Malmstrom, and J. Sitte, "Local cluster neural net: Architecture, training and applications," *Neurocomput.*, vol. 20, pp. 35–56, 1998.

[16] T. Körner, "Analog VLSI Implementation of a local cluster neural net," Ph.D. dissertation, Heinz Nixdorf Inst., Univ. Paderborn, Paderborn, Germany, 2000.

[17] J. Sitte, T. Körner, and U. Rückert, "Local cluster neural net analog VLSI design," *Neurocomput.*, no. 19, pp. 185–197, Aug. 1997.

[18] S. Geva and J. Sitte, "A cartpole experiment benchmark for trainable controllers," *IEEE Control Syst. Mag.*, vol. 13, no. 5, pp. 40–51, Oct. 1993.

Joaquin Sitte received the Licenciado degree in physics from the Universidad Central de Venezuela, Caracas, Venezuela, in 1968 and the Ph.D. degree in quantum chemistry from Uppsala University, Uppsala, Sweden, in 1974.

Until 1985, he was an Associate Professor at the Universidad de Los Andes, Merida, Venezuela, where he also headed the Surface Physics Research Group. Since 1986, he has been on the faculty of Queensland University of Technology, Brisbane, Australia, where currently, he is an Associate Professor in the Faculty of Information Technology and he teaches in the areas of computer systems architecture and computational intelligence. He has published extensively in computational intelligence and autonomous systems, with special emphasis on autonomous robots for education and entertainment. He has a special interest in the use of neural networks for sensing, thinking, learning, and actuation in autonomous robots. Recent work with his students include the PLSOM algorithm, morphogenetic techniques for evolvable hardware and resource efficient real-time selective attention algorithms.

Dr. Sitte has chaired several conferences and serves on editorial boards and conference technical committees. He is the Editor-in-Chief of the open-access journal *Advance in Artificial Neural Systems*.

Liang Zhang received the B.S. degree in mechanical engineering from the Tsinghua University, Beijing, China, in 1982 and the M.S. degree in information technology from the Queensland University of Technology, Brisbane, Australia, in 1998. Currently, she is working towards the Ph.D. degree at the Queensland University of Technology.

Her research interests include neural network training algorithms, neural network analog hardware on-chip training, and control problems.

Ulrich Rueckert received the diploma (M.Sc.) degree in computer science and the Dr.-Ing. (Ph.D.) degree in electrical engineering, both from the University of Dortmund, Dortmund, Germany, in 1984 and 1989, respectively.

He joined the Department of Electronic Components, University of Dortmund, in 1985, where he developed the first VLSI implementations of artificial neural networks in Europe. In 1993, he became an Associate Professor of Microelectronics and computer-aided design (CAD) at the Research Centre for Information and Communication Technology, Technical University of Hamburg, Harburg, Germany. Since 1995, he has been a Full Professor of Electrical Engineering at the University of Paderborn, Paderborn, Germany, where he heads the System and Circuit Technology research group at the Heinz Nixdorf Institute. The group works on innovative circuit design and development of microelectronic systems for massive-parallel and resource-efficient information processing. The main research interests are distributed intelligent systems, neural information processing, and reconfigurable computing architectures. He is member of the managing board of the Heinz Nixdorf Institute, and Adjunct Professor at the Faculty of Information Technology, Queensland University of Technology, Brisbane, Australia.