Neuromicro94 proceedings, Turin (Italy), 26-28 September 1994, pp. 138-144

# Analog VLSI implementation of kernel-based classifiers

M. Verleysen, Ph. Thissen
Université Catholique de Louvain, Laboratoire de Microélectronique,
3 pl. du Levant, B-1348 Louvain-la-Neuve (Belgium)

J. Madrenas
Universitat Politecnica de Catalunya, Departament d'Enginyeria Electrònica,
Gran Capità s/n, mòdul C4, E-08071 Barcelona (Spain)

## Abstract

*Kernel-based classifiers are neural networks (Radial Basis Functions) where the probability densities of each class of data are first estimated, to be used thereafter to approximate Bayes boundaries between classes. Such algorithm however involves a large number of operations, and its parallelism makes it an ideal candidate for a dedicated VLSI implementation. We present in this paper the architecture for a dedicated processor for kernel-based classifiers, and the implementation of the original cells.*

## 1 Introduction

Many different types of neural networks are used in classification tasks. Classification in this context means first to estimate in a high-dimensional space regions attributed to the different classes, according to given input/output pairs, and then to choose the best candidate (more probable) between the different classes when a new input point is presented to the network.

It is know that the Bayes law can be used directly to choose the most probable class for each point in the space, once the probability densities of each class and the a priori probabilities of the classes are known. Estimation of probability densities can be done through sums of kernels (Gaussian for example). In this paper, we present the implementation of a dedicated processor for a Bayes classifier based on estimation of probability densities. After a short description of the algorithm, we first present the global architecture of the system and of the processor, and then the implementation of the specific cells (kernels, analog memory points and distance computations).

## 2 Algorithms for classification tasks

Many neural-like algorithms may be used for classification tasks. As stated above, we will concentrate on kernel-based estimators of probability densities, used to build Bayes classifiers. The proposed implementation however includes the circuitry for LVQ-like algorithms, as it will become clear with the description of the processor.

*Kernel-based classifiers* (KBC) work in two phases. First, the probability density of the data distribution inside each class is estimated; then, the Bayes law is used to determine the boundaries between classes in the data space, and by this way to classify new input vectors.

To estimate the probability density of data belonging to a particular class [1], the principle is to sum kernels centered on the data from the learning set available in this class:

$$\hat{p}_x(N_c, u|w_c) = \frac{1}{N_c} \sum_{i=1}^{N_c} \Phi\left(\frac{u - x_i}{h}\right), \qquad (1)$$

where $\{x_i, 1 \leq i \leq N_c\}$ denotes the samples at disposal in class $w_c$; we suppose that there are $C$ classes denoted $w_c, 1 \leq c \leq C$. The scalar parameter $h$ is called the *width factor* of the kernel. The kernel $\Phi$ is said to be radial if it is only a function of the norm of its argument. Several types of kernels $\Phi$ may be used, the most classical one being a Gaussian function:

$$\Phi\left(\frac{u - x_i}{h}\right) = \frac{1}{(\sqrt{2\pi})^d} \frac{1}{h^d} e^{-\frac{1}{2}\left(\frac{\|u - x_i\|}{h}\right)^2}, \qquad (2)$$

where $d$ is the dimension of $u$ and $x_i$. The convergence of such estimator is proved in [2]. Let us mention that the width factor $h$ may be made different for each kernel $\Phi(u - x_i/h)$; in this case the kernels are referred to

as *variable* rather than *fixed.* However, the prohibitive number of operations involved in the case of variable kernels make them rather inefficient in terms of software or hardware implementations; furthermore, tests have shown that the gain in performances between estimators based on fixed and variable kernels is limited, especially in the case of finite databases. For these reasons, only fixed kernels estimators will be considered in the following.

Let us finally mention that, even if the probability density estimators are shown to be *asymptotically* unbiased, the number of computations is reduced in practical applications by reducing the number of samples through some kind of vector quantization, for example a LVQ procedure. In order for the LVQ procedure to give an appropriate distribution of the centroids in each class, their number will be set proportional to the a priori probabilities of the respective classes.

Once the probability densities are estimated in each class, the Bayes criterion may be used to classify any new vector $x$; class $w_c$ $(1 \leq c \leq C)$ will be attributed to vector $x$ if

$$\hat{p}_x(x|w_c)\, P(w_c) \geq \hat{p}_x(x|w_i)\, P(w_i),\ 1 \leq i \leq C, \quad (3)$$

where $P(w_i)$ is the a priori probability of class $w_i$. Such classifier is mostly interesting because of its property to approximate the Bayes limits between classes, i.e. the boundaries leading to a minimum number of misclassifications in case of overlapping distributions; most other classification systems do not have this property.

# 3 A mixed architecture for neural network classifier

The main weak point of the algorithm described above resides in the number of operations involved in the computation of a class. If there are $N$ kernels, $N$ distances must be evaluated, passed through non-linear functions, and summed, before deciding the most probable class, i.e. the largest estimate of the in-class probability densities (for equal a priori class probabilities, which will be supposed in the following). The fact that all distances and all kernels may be evaluated simultaneously make this algorithm an ideal candidate for an analog parallel implementation.

The architecture presented here is made of two parts. First, an analog processor implements all operations that can be found in the kernel-based algorithm described above; we will see later how this chip can also be used for LVQ-like algorithms. This processor is algorithm-independent, provided that the algorithm only uses the resources included in the chip: values of weights can be downloaded or adapted on the chip, non-linear functions may be used or by-passed, intermediate results may be obtained (for LVQ-like algorithms),... Secondly, all operations are sequenced using an external digital architecture, which is connected to the analog processor through the input/output lines and the control ones. This part is algorithm-dependent, since different algorithms will need different sequences of operations, and the use of different output lines of the processor. The control part will be a digital finite-state machine designed according to the algorithm; it can be realized by a digital specialized or general-purpose chip, or even with discrete components on a printed-circuit board, or by FPGAs.... Combining the analog processor and the digital control part leads to an efficient architecture for classification tasks; the next part of this paper details the analog processor and the different cells involved in.

# 4 Analog processor

## 4.1 Block description of the circuit

To describe in details the analog processor and the different operations which can be realized, let us examine the functional description of figure 1.

The core of the system is built around $P$ identical cells, each of them being composed of memory points to store the coordinates of the centroid and its class, together with a distance calculator to compute the distance between this centroid and an input vector. Shortly, the system will work as follows. A set of $P$ centroids $p_k, 1 \leq k \leq P$ will be stored in the processor; in the case of the KBC algorithm, the coordinate of the centroid corresponds to the center of the kernel function $\Omega$. Then, when an input vector is presented to the circuit for classification, all distances between this input vector and each of the centroids are computed in a parallel way; this is the purpose of the $P$ mentioned distance computation cells.

The $P$ computed distances are then used in two ways. On one hand, they are compared to find the smallest one, in order to select the closest centroid from the input vector; this is used in LVQ-like algorithms, in the purpose of selecting the winning centroid. On the other hand, the distances serve as inputs to $P$ Gaussian-like kernel function, used in KBC algorithms, as mentioned in equation 2.

In the case of the LVQ algorithm, the selection of the winning centroid $p_a$ completes the recognition

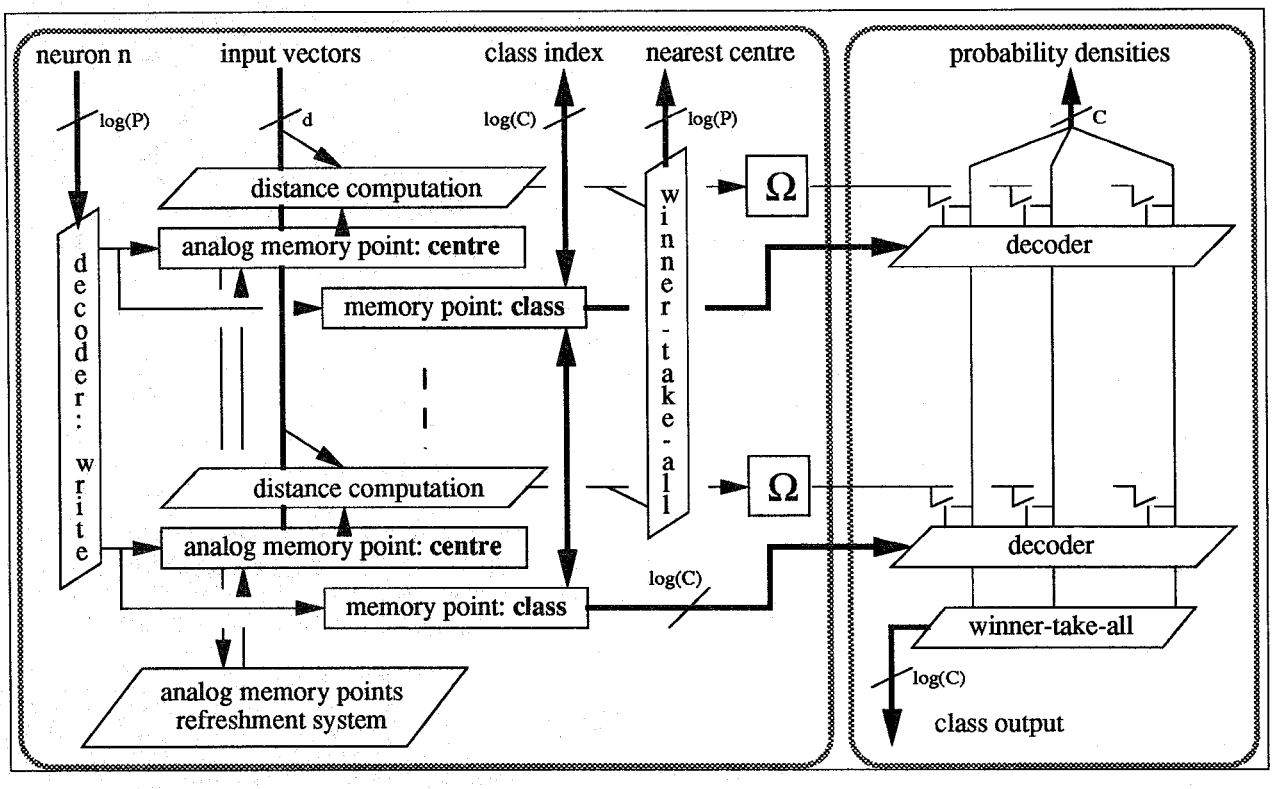


Figure 1: Functional description of the analog processor

phase of the algorithm. In the case of the KBC algorithm, the $P$ kernel outputs are summed class by class, according to equation 1, in order to estimate the probability densities of each class. The parameters of the kernels, namely their widths and shapes, may be adjusted by external commands; this will be detailed in section 4.2.4. According to the Bayes law (equation 3), classification of the input pattern is then realized by selecting the largest probability density from among the different classes.

A supplementary factor $P(w_i)$ is found in equation 3; it corresponds to the a priori probabilities of the classes. As in the LVQ algorithm [3], these a priori probabilities are estimated by the relative number of points in each class, condition which is realized in our circuit since we have one kernel per input point in the distribution.

In the following sections we describe the analog cells used to realize the analog processor. The sizes of all transistors have been chosen for cells designed in the MIETEC 2.4 $\mu$m technology, and for a circuit with $P = 32$ (the number of kernels), $d = 16$ (the dimension of the data space) and a precision in the memory points equal to 8 bits.

## 4.2 Description of the analog cells

### 4.2.1 Analog memory points

Analog memory points have been used to store the locations of the centroids for silicon area reasons, and to avoid non-necessary analog/digital conversions in the chip.

The principle of our analog memory point is to store a current on capacitor $C_g$ in figure 2. When switch transistor $T_s$ is on, the drain and gate of memory transistor $T_m$ are connected together, and its gate voltage adjusts to let the input current $I_{mem}$ flow through the transistor. When transistor $T_s$ is switched off, the capacitor $C_g$ will memorize the gate voltage of $T_m$ to keep the same current $I_{mem}$ flowing through the transistor.

To compensate for leakage currents effects in the blocked junction of transistor $T_m$, a refreshment system sequentially reads all analog values stored on the chip and refreshes them. The principle is the following. We consider that both the charge injection (when switching off transistor $T_s$) and the leakage current in the blocked junction make the voltage $V(C_g)$ between $V_{dd}$ and the gate of $T_m$ decrease from less than one LSB in a refreshment period T; this LSB is measured

over the whole dynamics of stored voltages on $C_g$. In figure 2, both the leakage current and the charge injection will have the same sign: the blocked junction will inject positive charges from $V_{dd}$ to $C_g$, so as the switching of transistor $T_s$. We then know the sign of the slope of $V(C_g)$. If the analog value in a memory point is now read at regular intervals $T$, and converted into the smallest digital value greater than the analog one, the memory point may be refreshed to its initial level as illustrated, keeping the stored value fixed up to a precision of one LSB. All memory points of the circuit may be refreshed by the same system, an analog-to-digital converter followed by a digital-to-analog one, provided that the period $T$ between two refreshments of the same memory point is small enough to ensure a decay in $V(C_g)$ less than one LSB; a detailed description of this system may be found in [4].

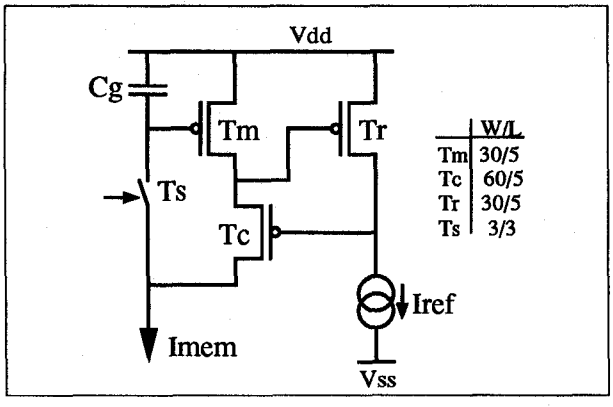


Figure 2: Regulated cascode analog memory point

Another problem is the dependency of the current in transistor $T_m$ with its drain voltage; the cell implemented on the chip is thus a regulated cascode one [5] (use of transistors $T_m$ and $T_c$). In figure 2, transistor $T_m$ operates in its linear region to reduce its transconductance $g_m$, as well as the current variation due to charge injection on $C_g$. In order to keep the drain voltage of $T_m$ as fixed as possible, one has to increase the gain of the loop formed by $T_c$ and $T_r$; they are thus both kept in saturation, and transistor $T_r$ operates in weak inversion (through a very small $I_{ref}$ current) to maximize its gain (transconductance over output conductance).

The capacitance of $C_g$ must be around $1pF$ to reach a 8-bits accuracy in the stored current; to obtain this value, a supplementary capacitor realized between the two polysilicon layers of the MIETEC 2.4 $\mu$m technology is added in parallel to the gate capacitor of $T_m$. The maximum current memorized in the cell has been set to 128 $\mu A$ one LSB corresponding to 500 $nA$.

### 4.2.2 Synapse and input circuitry

The circuit of figure 3 is repeated $P \times d$ times on the chip, and connected to the $P \times d$ analog memory points described in section 4.2.1. The purpose of the circuit in figure 3 is twofold. First, it is used as input of the corresponding memory point when a current must be stored. In this mode, an external input voltage $V_{in}$ generates a current $I_{in}$ in figure 3, which is the current $I_{mem}$ in figure 2; write transistor $T_s$ is then switched on, and the current is memorized in the cell. In the second operation mode, we suppose that a current $I_{mem}$ is memorized in the cell of figure 2, and that it has to be subtracted from current $I_{in}$; we will see in the next section how this difference may be used to compute the distance between an input vector $x_i$ and a centroid $p_j$. In this mode however, the difference between currents $I_{mem}$ and $I_{in}$ may be allowed to flow out of these cells; this will also be detailed in section 4.2.3. The principle of the cascode cell in figure 3 is similar to the principle of the memory point; the sizes of transistors are given in the figure.

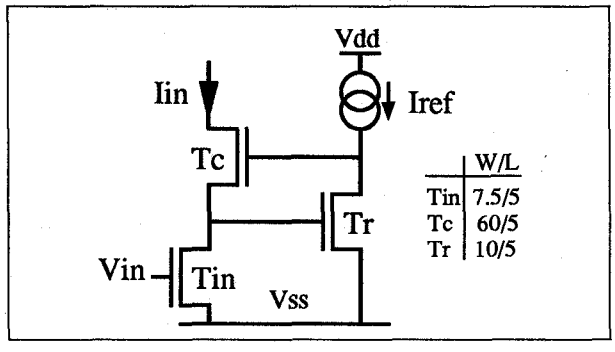


Figure 3: Regulated cascode input circuitry

### 4.2.3 Distance computation

One of the main operations that must be realized on-chip is the distance computation between a $d$-dimensional input vector and $P$ $d$-dimensional centroids. Manhattan distance has been used here for simplicity reasons, since we know that the choice of distance measure do not influence a priori the performances of a classifier [6].

To compute the Manhattan distance between an input vector $x$ and a centroid $p_i, 1 \leq i \leq P$, three operations must be realized: subtraction between $x$ and $p_j$ coordinate by coordinate, absolute value, and sum of these results over all coordinates. The subtraction has already been addressed in the previous section; when a memory point is in read mode, the difference

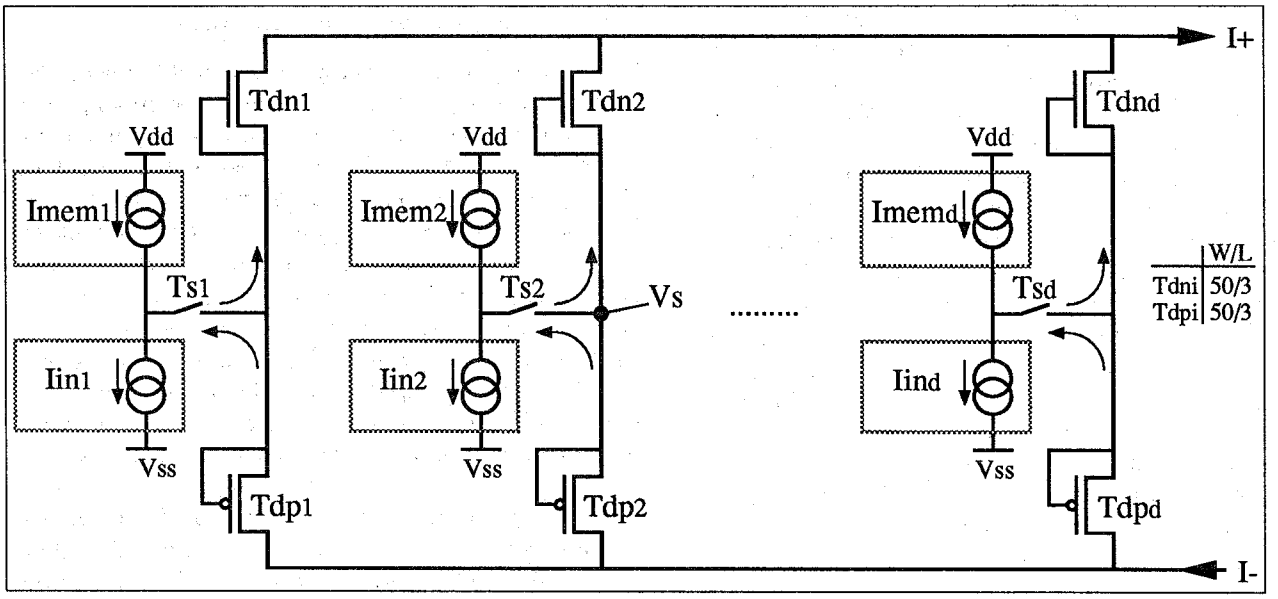


Figure 4: Computation of the Manhattan distance between the input vector and one centroid

between the memorized currents $I_{memi}$ and the input currents $I_{ini}$ in figure 4 ($1 \leq i \leq d$) is allowed to flow out from the cells (transistors $T_{si}$ are switched on). This current may either be positive or negative; depending on its sign, it is directed to one of the two summation current lines in figure 4. The difference between these two sums must finally be computed by a set of current mirrors in order to complete the implementation of the distance computation. Voltages on lines $I_+$ and $I_-$ are kept fixed through simple operational amplifiers.

### 4.2.4 Kernel functions

Recent developments in the theory of KBC algorithms [7] have shown that the quality of probability density estimations can be greatly improved by adjusting two kinds of parameters in the Gaussian kernels. The first one is classically its width factor, but a second one, which must be adjusted depending on the dimension of the data space, determines the tail curvature of the Gaussian function, i.e. the rate at which the kernel function drops off. We show in this section two ways of implementing kernel functions.

Figure 5 shows the differential pair used to realize the first type of Gaussian-like kernel. Let us first mention that the exact kernel shape is not critical for the approximation of probability densities as soon as two such parameters can be adjusted; moreover, only half of the Gaussian function has to be realized, since its argument is always positive (distances). We thus use the non-linear characteristics of a differential pair to evaluate the Gaussian-like functions.

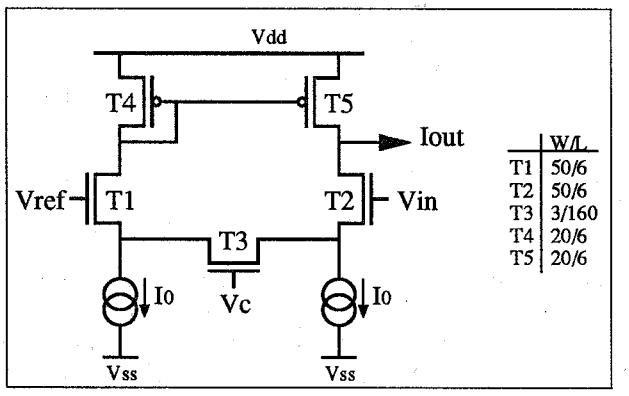


Figure 5: Kernel Gaussian-like function

In figure 5, the input voltage $V_{in}$ is generated by flowing the argument of the kernel function, namely the difference between currents $I_+$ and $I_-$ in figure 4, into a transistor in its linear region. The width of the kernel is determined by voltage $V_{ref}$, while its curvature is adjusted by modifying voltage $V_c$ which acts on the conductance of transistor $T_3$.

Figure 6 respectively shows a simulation of the kernel Gaussian like function for only one $V_C$ and $V_{ref}$ ranging from 0.5 to 2.5 V, and for $V_{ref}$ fixed to 1.5 V and a sweep of $V_C$. The chip is currently under test and the measurements are not yet available.

Another implementation of a Gaussian kernel,

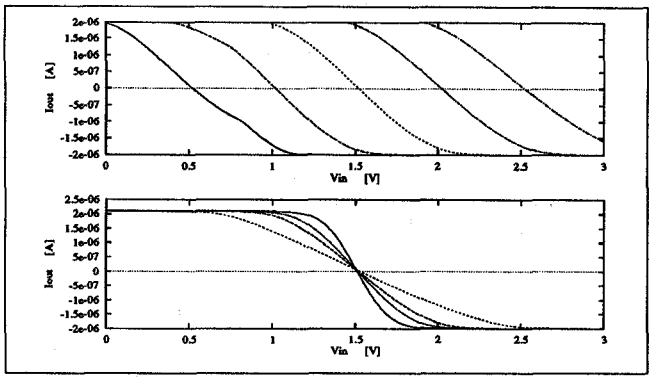


Figure 6: Simulation of the kernel Gaussian like function: influence of $V_{ref}$ and of $V_C$

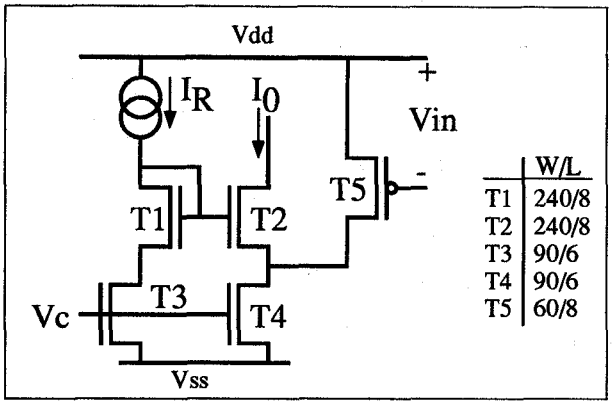


Figure 7: Kernel Gaussian function

which respects more closely the equation of a Gaussian kernel (2) is also proposed here. The principle of the circuit is illustrated in figure 7. The Gaussian function is constructed into two steps. First, a MOS transistor in saturation $T_5$ performs the square function of the input voltage; secondly, a negative exponential circuit realizes the Gaussian function.

To understand how the circuit of figure 7 works, let vus first suppose that transistors $T_3$ and $T_4$ act as resistors, identified by $R_1$ and $R_2$. The reference current $I_R$ is set small enough to put transistors $T_1$ and $T_2$ in weak inversion. If we note $V_{D1}$, $V_{D2}$, $V_{S1}$, $V_{S2}$, and $V_G$ respectively the drain voltages of $T_1$ and $T_2$, their source voltages, and their common gate voltage, we have if $V_{D1}$ and $V_{D2}$ are much greater than $u_T$:

$$I_R = I_S\, e^{\frac{V_{G1}}{n u_T}} e^{-\frac{V_{S1}}{u_T}} \tag{4}$$

$$I_0 = I_S\, e^{\frac{V_{G1}}{n u_T}} e^{-\frac{V_{S2}}{u_T}} \tag{5}$$

where $u_T = kT/q$ and $n$ the subthreshold slope. Dividing both expressions and substituting $V_{S1}$ and $V_{S2}$ leads to

$$I_0 = I_R e^{\frac{R_1 I_R - R_2 - I_0 - R_2 I_1}{u_T}} \tag{6}$$

Finally, assuming that $I_1 \gg I_R$, and then $I_0 \ll I_R$, which can be easily guaranteed since $I_R$ is small as before, we have

$$I_0 \approx I_R e^{-\frac{R_2 I_1}{u_T}} \tag{7}$$

The output current $I_0$ decreases exponentially with current $I_1$, which was the expected behavior; the value of $R_2$ determines the exponential constant.

The value of $R_2$ is then determined by transistor $T_4$, working in strong inversion and in its linear region; assuming that the drain voltage of $T_4$ is small, its output conductance is controlled by $V_C$ and given by:

$$g_{ds4} = \frac{1}{R_4} = \beta_N (V_C - V_{TN}) \tag{8}$$

where $g_{ds4}$ is the transconductance of transistor $T_4$, $V_{TN}$ the threshold voltage of N-type transistors and $\beta_N$ their conductance parameter. Transistor $T_5$ is saturated and in strong inversion too, and current $I_1$ is then given by:

$$I_1 = \frac{\beta_P}{2\lambda}(V_{in} - |V_{TP}|)^2 \tag{9}$$

where $V_{TP}$ is the threshold voltage of P-type transistors, $\beta_P$ their conductance parameter, and $\lambda$ takes the substrate effect into account. Combining 7, 8 and 9 leads to:

$$I_0 = I_R e^{-\frac{\beta_P (V_{in} - |V_{TP}|)^2}{2 n u_T \beta_N (V_C - V_{TN})}} \tag{10}$$

which has the desired Gaussian form; $V_C$ controls the parameter of the Gaussian function, and the constant $V_{TP}$ can be compensated by an offset voltage shifting the center of the function.

Figure 8 shows the measured output of the circuit for $V_C$ varying between 1 and $4V$ (with a step of $1V$), and for an input voltage between 0 and $5V$. The cell has been realized in UCL SOI (Silicon-On-Insulator) $3\mu m$ technology.

## 5 Conclusion

We described in this paper the analog implementation of a kernel-based classifier, based on estimation of in-class probability densities. The global architecture of the system was detailed, together with the implementation of some specific blocks, including measurements on the Gaussian kernels realized in Silicon-On-Insulator technology. Such analog processor may

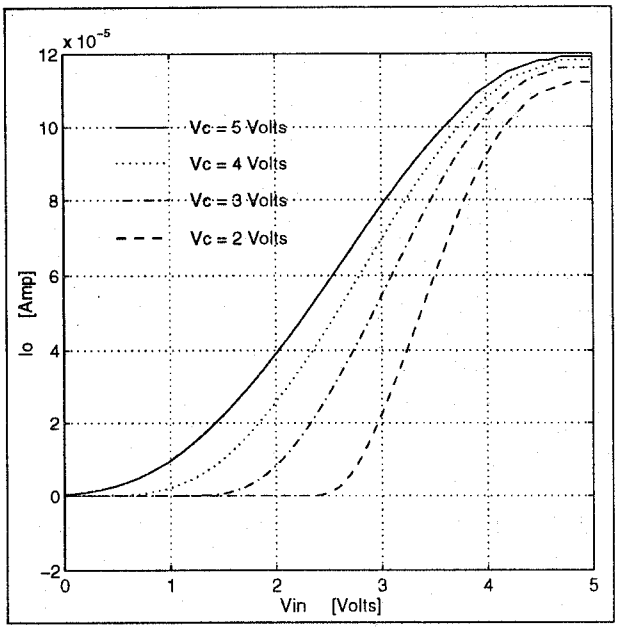


Figure 8: Chip measurements of Kernel Gaussian function

be used in any classification system when speed and portability are necessary, together with the high performances of Bayes classifiers. Work still to achieve concern the implementation of a large system (only sparse cells were realized and tested up to now) and the programming of a digital controller to sequence the operations in the processor.

## Acknowledgments

Part of this work has been funded by the ESPRIT-BRA project 6891, ELENA-Nerves II, supported by the Commission of the European Communities (DG XIII). Michel Verleysen is a Senior Research Assistant of Belgian National Fund for Scientific Research (FNRS). Philippe Thissen is working towards the Ph.D. degree in microelectronics under an IRSIA (Institut pour l'Encouragement de la Recherche Scientifique dans l'Industrie et l'Agriculture) fellowship.

## References

[1] E. Parzen, "On the estimation of a probability density function and the mode," *Ann. Math. Stat.*, vol. 27, pp. 1065–1076, 1962.

[2] T. Cacoullos, "Estimation of a multivariate density," *Annals of Inst. Stat. Math.*, vol. 18, pp. 178–189, 1966.

[3] T. Kohonen, *Self-Organization and Associative Memory*. Berlin: Springer-Verlag, 1989. 3rd Edition.

[4] D. Macq, J. Legat, and P. Jespers, "Analog storage of adjustable synaptic weights," in *Proceedings of the SPIE conference on Applications of Artificial Neural Networks, Orlando (USA)*, pp. 712–718, April 1992.

[5] C. Toumazou, J. Hugues, and D. Pattulo, "Regulated cascode switched-current memory cell," *Electronics Letters*, vol. 26, pp. 303–305, March 1990.

[6] M. Verleysen, P. Thissen, J. Voz, and J. Madrenas, "An analog processor architecture for neural network classifier," *IEEE Micro*, vol. 14, pp. 16–28, June 1994.

[7] P. Comon, J. Voz, and M. Verleysen, "Estimation of performance bounds in supervised classification," in *ESANN94-European Symposium on Artificial Neural Networks* (M. Verleysen, ed.), (Brussels, Belgium), pp. 37–42, D facto publications, April 1994.