VLSI Design of a Hamming Artificial Neural Network

Diploma Thesis

Stéphane Badel

Assistant: Alexandre Schmid

Professor: Yusuf Leblebici

February 21, 2003

Microelectronics Systems Laboratory

Swiss Federal Institute of Technology, Lausanne
Acknowledgments

I would like to thank the LSM staff, and especially Prof. Leblebici for giving me the opportunity of this work. Special thanks also to Alexandre Schmid, who gave me his support and his imagination. My sincere gratitude goes to my family and friends who were always supporting me.
Contents

I Capacitive Neurons and their Applications 1

1 Brief Overview of Neural Networks 2
  1.1 Artificial neurons .......................... 2
  1.2 Neurons and threshold logic .................. 2
  1.3 Hamming networks .......................... 3

2 Capacitive Neurons 4
  2.1 Basic operation of the capacitive neuron .......... 4
  2.2 Transfer function .......................... 6

3 Design of Capacitive Hamming ANNs 8
  3.1 The quantifier .......................... 8
  3.2 The discriminator .......................... 10
    3.2.1 Principle of operation .................. 11
    3.2.2 Designing WTA and LTA units ............... 12
    3.2.3 A Discriminator with Hamming distance output ...... 12
    3.2.4 An Exemplar-ranking discriminator ............ 14

II VLSI Design of a Two-dimensional Hamming ANN for Image Processing Applications 15

4 Introduction 16

5 Precision Alignment System Application 17
  5.1 Details of the chosen application ............... 17
  5.2 High-level simulation ........................ 18
6 Design of a Test Chip

6.1 Design of the core circuit ........................................ 20
  6.1.1 2D Hamming matrix organization .......................... 20
  6.1.2 Design of the Hamming cell .................................. 21
  6.1.3 Design of the capacitive neurons ............................ 22
  6.1.4 Schematic simulation ........................................... 23

6.2 Layout ............................................................. 23
  6.2.1 Layout considerations ......................................... 24
  6.2.2 The core circuit ................................................ 25
  6.2.3 Post-layout simulations ....................................... 26

6.3 Floor planning .................................................... 27
  6.4 Full circuit layout ............................................... 29

7 Conclusion ........................................................... 30

III Appendices .......................................................... 32

A Publications .......................................................... 33

B Additional Layouts .................................................... 34
Part I

Capacitive Neurons and their Applications
1 Brief Overview of Neural Networks

1.1 Artificial neurons

Artificial neural networks are modeled after the human brain, which is composed of billions of elementary neurons and interconnections between them. The neuron are elementary, in the sense that their task consists of summing weighted inputs and eventually firing their output depending on a threshold, which is really an elementary task. It is the large number of neurons and the dense interconnections that make it possible to realize very complex functions.

Similarly, an artificial neuron (figure 1.1) is a simple unit that realizes the sum of its weighted input and a thresholding operation on the result. These neurons, when grouped together, form an artificial neural network or ANN.

1.2 Neurons and threshold logic

When the inputs and outputs of a neuron are all binary, its transfer function is a logic function. For example, a neuron with two inputs having a weight of 1 and threshold of 1.5 realizes the logic function AND, as the output fires (i.e. switches to 1) only when both inputs are 1. This way of performing logic functions is known as threshold logic, and originated back in the 1960’s — in
fact, threshold logic gates are a particular kind of neurons where the inputs and output are binary, and the thresholding function is a \textit{hard limiter}.

It has been shown that a single neuron can, by adjusting weights and threshold, perform any \textit{separable} logic function. Non-separable functions, such as exclusive-or, require more than one neuron.

\subsection{1.3 Hamming networks}

\textbf{Hamming distance} The Hamming distance is defined for binary vectors as the number of bits that differs between the two vectors and can be written as

\[ H(x, y) = \sum |x_i - y_i| = \sum (x_i - y_i)^2 \]  

(1.1)

(note that when considering binary vectors, $|x_i - y_i|$ is equal to $(x_i - y_i)^2$).

\textbf{Hamming networks} Hamming networks are a particular type of ANNs, which perform the task of matching an input vector with the most similar vector from a memorized set of exemplars. Those are typically used in pattern recognition applications, which engage speech recognition, character and handwriting recognition, etc.

The architecture of a Hamming network is depicted in figure 1.2. It is composed of two distinct parts: a quantifier and a discriminator. The quantifier processes the inputs and delivers the hamming distances to the discriminator, which in turn selects the neuron with smallest hamming distance.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{hamming_network.png}
\caption{Hamming network}
\end{figure}
2 Capacitive Neurons

Artificial neurons can be implemented in many different ways, either analog or digital. In this section, we describe a neuron implementation which is very similar to the capacitive threshold logic gate described in [7].

2.1 Basic operation of the capacitive neuron

Architecture The capacitive neuron architecture, depicted in figure 2.1 basically consists of a number of inputs — the neuron’s synapses — capacitively coupled to a common node — the neuron’s dendrite — which feeds the input of an inverter. As we will see, the capacitances perform the weighted sum operation, while the thresholding is assumed by the inverter.

Operation The circuit operation is based on two phases scheme — namely precharge phase and evaluation phase — requiring non-overlapping clock signals, during which charge conservation applies. During the precharge phase $\phi_1$, the dendritic voltage is precharged to a reference voltage $V_{ref}$, while the synaptic nodes are all precharged to a value $V_{in}$. At end of $\phi_1$, $V_d = V_{ref}$ and $V_{in} = V_{in}$.
Then, during the evaluation phase, input voltages are applied to the synapses, eventually resulting in charge being transferred to the synaptic capacitances. At end of $\phi_2$,

$$V_{s_i} = V_i$$

Charge conservation then yields the following equation

$$\sum_{i=1}^{n} C_i (V_{ref} - V_{s_i}) = \sum_{i=1}^{n} C_i (V_d - V_i)$$

from which one can deduce the dendritic voltage at end of the evaluation phase

$$V_d = V_{ref} + \frac{1}{C_{tot}} \cdot \sum_{i=1}^{n} C_i (V_i - V_{s_i})$$

(2.1)

$$= V_{ref} + \sum_{i=1}^{n} \frac{C_i}{C_{tot}} \Delta V_s_i$$

$$= V_{ref} + \Delta V_d$$

(2.2)

where

$$C_{tot} = \sum_{i=1}^{n} C_i$$

Thus, at end of evaluation phase, the dendritic voltage is increased (or decreased) by the sum of $n$ synaptic contributions $\Delta V_d$, which are proportional not only to the corresponding variation of synaptic voltage $\Delta V_s_i$, but also to the corresponding synaptic capacitance $C_i$. Any variation in these parameters allows control of the individual weights, including negative as well as positive weights. To illustrate this process, a SPICE simulation of a capacitive neuron is shown on figure 2.2.

**Reference voltage** Although the dendrite can be precharged at any voltage, it is convenient to short-circuit the inverter to precharge the dendrite to its switching threshold. This way, no external voltage reference is necessary. Furthermore, the variation of the inverters threshold won’t affect the operation of the circuit, as the resulting voltage at end of evaluation phase is relative to the precharge voltage.
2. Capacitive Neurons

**Initial synapse voltage** As it can be seen in equation 2.1, the synaptic contributions are composed of an input-dependant part $C_i V_i$, and an input-independent part $-C_i V_{in}$, which can be considered as an offset.

### 2.2 Transfer function

Equation 2.1 expresses the relation between dendritic voltage and input voltages. To obtain the neuron’s transfer function, we must consider the output from the inverter (i.e. take into account the thresholding operation), and also establish a relation between the analog input voltages and the binary value of the input bits (for example, applying the inputs directly would result in the relation $V_i = V_{dd} x_i$, applying the inverted inputs would lead to $V_i = V_{dd} (1 - x_i)$, etc.) We would then obtain an expression of the transfer function in the form

$$
Y = \begin{cases} 
1 & \text{when } \sum w_i x_i > \text{threshold} \\
0 & \text{when } \sum w_i x_i < \text{threshold}
\end{cases}
$$

This expresses a logic function that could also be expressed in terms of the logic operators AND, OR and NOT. Thus, the resulting transfer function of the neuron depends on the threshold voltage and on the individual weights.

**Setting the weights** The weight of each input can be controlled by adequately sizing the corresponding capacitor. The actual weight is $C_i/C_{tot}$, so an additional capacitor may be added between dendrite and ground to adjust the value of $C_{tot}$ if needed.

Positive weights can be obtained by precharging a synapse to $V_{ss}$. This results in the synaptic contribution being equal to $(C_i/C_{tot}) V_{dd} x_i$. Similarly, negative weights can be obtained by inverting an input while precharging the corresponding synapse to $V_{dd}$. This results in the synaptic contribution being equal to $(C_i/C_{tot}) [V_{dd} (1 - x_i) - V_{dd}] = - (C_i/C_{tot}) V_i$

**Setting the threshold** Although the inverter switching threshold cannot be easily (nor precisely) changed, control of the threshold voltage can be achieved through the addition of a dc component, i.e. a constant input, so as to offset the resulting dendritic voltage by a value of $-V_{\text{threshold}}$. Usually, this would be done using a constant input of $V_{dd}$ or $V_{ss}$, depending on the desired sign of the offset, while the actual offset value would be set by adjusting the corresponding weight.

**An example** Suppose we want to implement the logic function $Y = x_1 x_2 x_3 + x_1 \overline{x_2}$ using a capacitive neuron. This function can also be expressed in threshold logic as (see table 2.1 for comparison of the two functions, and see [8] for details
2. Capacitive Neurons

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$2x_1 - x_2 + x_3$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

(a) (b)

Table 2.1: truth tables (a) conventional logic (b) threshold logic

on how to determine the weights)

$$Y = \begin{cases} 
1 & \text{when } 2x_1 - x_2 + x_3 > 1.5 \\
0 & \text{when } 2x_1 - x_2 + x_3 < 1.5
\end{cases}$$

which is incidentally a neuron transfer function with weights $w_1 = 2, w_2 = -1, w_3 = 1$ and threshold $\theta = 1.5$.

**Variable weights** The capacitances values are usually set at design time, resulting in a fixed-weights circuit. Variable weights were implemented in [10] and [9] by selectively connecting multiple capacitors in parallel, in order to obtain a variable capacitance. Another way of varying the weights would be to vary the input voltages proportionally to the weights. The input bit-input voltage relationship would then be $V_i = w_i V_{dd} x_i$, where $0 \leq w_i \leq 1$. This approach would require only one minimum capacitance per synapse, but a number of reference voltages. However, reference voltages can be shared amongst multiple synapses, eventually resulting in a more compact design.
3 Design of Capacitive Hamming ANNs

It was stated previously, a Hamming ANNs are usually composed of two stages. The first stage — the quantification layer — performs the computation of the Hamming distance between an input vector and exemplars stored in memory. The second stage — the discrimination layer — performs the selection of the best-matching exemplar (the one with smaller Hamming distance).

We will now present the design of a Hamming neuron, based on capacitive architecture presented above, during which will be introduced a novel approach in designing a discrimination layer, which deviates from the classic positive feedback architecture and exhibits a broad range of possibilities for extending the capabilities of the network.

3.1 The quantifier

Absolute and relative modes In a Hamming network, the first layer, or quantifier, performs the computation of the hamming distance between an input vector and exemplars stored in the network memory. Depending on the targeted application, Hamming distance may be substituted by another similarity score which, although not being equal to the exact Hamming distance, remains pertinent for performing vector comparison.

In [1], a pattern classifier was designed in which the input vector was simultaneously compared to all exemplar vectors. Thus, the computed Hamming distances

$$H(x, y) = \sum (x_i - y_i)^2 = \sum x_i + \sum y_i - 2 \sum x_i y_i$$

did all exhibit a common term $\sum x_i$, which depends only on the input vector and could be removed. So, the author used $S_i = -\sum y_i (2x_i - 1)$ as a similarity score, resulting in a simplified design.

This simplification turns out to be a very useful one, as pattern matching is one major application of Hamming networks. Throughout this paper, we will
Design of Capacitive Hamming ANNs

refer to this operation mode as **relative mode**, by contrast with the **absolute mode** in which exact Hamming distances are computed.

**Absolute quantifier** An absolute quantifier can be designed with a neuron whose inputs are preceded by layer of logic gates, which compute the Hamming distance between two bits, that is, output a 1 when and only when the two bits are different. In the proposed implementation (fig. 3.1), the exemplar vector is stored in a memory cell. Using XOR and XNOR gates can be considered.

![Absolute quantifier circuit](image)

(a) (b)

Figure 3.1: Absolute quantifier (a) circuit (b) sample operation

Also, the weights can assume the values +1 or −1. This leads us to consider 4 different operation modes for the absolute quantifier, in which the dendritic voltage will reach different values at end of the evaluation phase

- **XOR gate, +1 weight**: \( V_d \) will *rise* when the vectors are *different*
- **XOR gate, −1 weight**: \( V_d \) will *drop* when the vectors are *different*
- **XNOR gate, +1 weight**: \( V_d \) will *rise* when the vectors are *similar*
- **XNOR gate, −1 weight**: \( V_d \) will *drop* when the vectors are *similar*

Note that, for each of these modes, the rov voltage will either always rise or always drop from its initial value, the absolute quantifier thus operates in a *unipolar* fashion.

**Relative quantifier** In the relative quantifier, the neuron inputs are not preceded by logic gates, because this is not necessary. The function \( \sum y_i (2x_i - 1) \) can be realized with the simpler architecture of figure 3.2. This is made clear when considering the synaptic voltage \( V'_s \), at end of evaluation phase

\[
V'_s = \begin{cases} 
\frac{V_{dd}}{2} & \text{when } y_i = 0 \\
V_{dd} x_i & \text{when } y_i = 1 
\end{cases} = V_{dd} x_i + \frac{V_{dd}}{2} y_i 
\]

\[
= V_{dd} x_i y_i + \frac{V_{dd}}{2} (1 - y_i)
\]
3. Design of Capacitive Hamming ANNs

![Circuit Diagram](image)

(a) (b)

Figure 3.2: Relative quantifier (a) circuit (b) sample operation

and then the synaptic contribution $\Delta V_{s_i}$

$$\Delta V_{s_i} = \frac{C_i}{C_{tot}} \left( V_{s_i} - \frac{V_{dd}}{2} \right) = \frac{C_i}{C_{tot}} \frac{V_{dd}}{2} y_i (2x_i - 1)$$

Note that the relative quantifier operates in a bipolar fashion (see fig. 3.2b) as $\Delta V_{s_i}$ can assume positive as well as negative values.

**Hard-memorized exemplars** As it has been done in [1], exemplars can be hard-memorized when using a relative architecture. This means that the values of the exemplars are set at design time by placing a synaptic capacitor where $y_i = 1$, and no synaptic capacitor where $y_i = 0$. More details can be found in the mentioned article.

**Choosing an operation mode** Choosing between absolute and relative modes is a matter of needs. If we don’t need to have the exact Hamming distance computed, we can use the relative mode which results in a slightly simpler implementation.

### 3.2 The discriminator

The type of discriminator commonly used with Hamming networks is the winner-takes-all (WTA), which selects the one exemplar with the smallest hamming distance. WTA have been implemented in [1], [9] using positive feedback devices, derived from the architecture of sense amplifiers used in memories, which pull the higher input high and the others low. This work presents a different style of discriminator. As we will see, capabilities of that discriminator extend beyond that of a WTA. It can of course be used as a WTA, but it can be also used to perform various task, such as ranking the exemplars, with few additional circuitry. We also expect better performances with this architecture, in terms of occupied area and maximum precision.
3. Design of Capacitive Hamming ANNs

3.2.1 Principle of operation

The discrimination is performed, after the evaluation phase, by simultaneously applying a perturbation to the dendrites of the neurons — which in fact results in varying the threshold of the neurons — and then gather information from the switching or non-switching of their outputs.

This is implemented by adding a perturbation input, or a threshold control input, to the neurons. As it was discussed in section 2.2, to achieve this we need to add one synapse with synaptic capacitance $C_p$, which will add a contribution $\Delta V_{sp} = \left(\frac{C_p}{C_{tot}}\right)(V_p - V_{sp0})$. The ratio of $C_p$ to $C_{tot}$ will determine the range of the resulting dendritic perturbation, or the control range of the neuron threshold, and thus needs to be adequately sized.

Sample SPICE simulations showing the application of a linear perturbation as well as an impulse perturbation are shown on figure 3.3, where the effect of the perturbation on the dendritic voltage of the neurons can be seen.

![Figure 3.3: Sample SPICE simulations (a) ramp.shape perturbation (b) pulse.shape perturbation](image)

Coping with zero Hamming distance When the input vector matches exactly one of the exemplar vectors, the corresponding neuron dendritic voltage will remain equal to $V_{ref}$, thus leaving the inverter in an unstable situation. To prevent this, we must add a supplemental offset synapse, which will offset the dendritic voltage of all the neurons by the same amount. This offset can be any value, as long as it is sufficiently large to ensure that the output of the inverter will remain stable in any case.

Dynamic range considerations When sizing the different synaptic capacitors of the neuron, care must be taken to ensure that

- the dendritic voltage will never reach above $V_{dd}$ nor below $V_{ss}$
- the capacitance $C_p$ of the perturbation synapse is be large enough to allow a sufficiently wide perturbation range

This can be analysed by considering the extreme values assumed by the dendritic voltage during the evaluation and perturbation phases. Eventually, a dummy
3. Design of Capacitive Hamming ANNs

capacitor may be added between dendrite and ground to reduce the weights. See page 13 for an example.

3.2.2 Designing WTA and LTA units

**WTA units** A WTA unit can be designed by applying a ramp-shape perturbation simultaneously to all the neurons and capturing the first switching neuron, that is, the one with smaller Hamming distance. For this to work, the neuron must operate in a mode in which the neuron with smaller Hamming distance has its dendritic voltage rise less than other neurons. We will consider a neuron operating in absolute mode, using XOR gates and positive weights. In this configuration, the dendritic voltage of the neurons will reach

\[ V_d = V_{ref} + (C_i/C_{tot}) H, \]

where \( H \) denotes the Hamming distance between the input and the exemplar. Thus, the neuron with smaller Hamming distance will have its dendritic voltage closer to \( V_{ref} \), and its output will switch first.

**LTA units** LTA (loser-takes-all) units can be designed in the same way as WTA units, by simply changing the operation mode of the neuron: using XNOR gates instead of XOR gates will turn the WTA into a LTA.

**k-WTA and k-LTA units** k-WTA and k-LTA unit (that select not one, but k winners or loosers), can also be easily implemented. A ramp-shape perturbation is still applied, but the discriminator waits until k neuron outputs have switched. Using a CTL gate, one can easily implement the function

\[
Y = \begin{cases} 
1 & \text{when } \sum x_i > k \\
0 & \text{when } \sum x_i < k
\end{cases}
\]

that will trigger the end of the discrimination phase.

3.2.3 A Discriminator with Hamming distance output

**Principle** When applying a ramp-shape perturbation, dendritic voltage amplitudes are translated into switching times, that is, a neuron with higher dendritic voltage will switch later than a neuron with lower dendritic voltage. Thus, when the switching time of the individual neurons are can be made proportional to their respective Hamming distance, which can be then determined by measuring this switching time. In fact, this is absolutely equivalent to a analog to digital conversion of the neuron dendritic voltage.

**Shape of the perturbation** The shape of the perturbation must be chosen so that the switching time remains proportional to the Hamming distance. Candidates include an analog linear perturbation, and a stair-shaped perturbation. The latter one can be easily implemented with a number of equal perturbation synapses triggered in a sequential manner; that is, summing a number of time-offset step perturbations to produce a stair perturbation.
3. Design of Capacitive Hamming ANNs

**Dynamic range considerations** We will now make some considerations on the sizing of the capacitors to achieve proper operation of the discriminator, which hold also for other operation mode with slight modifications.

Let us define the following values:

- \( \Delta V_u \) = dendritic voltage swing corresponding to Hamming distance of 1
- \( \Delta V_o \) = offset (common to all neurons)
- \( \Delta V_e \) = maximum dendritic voltage swing during evaluation phase
- \( \Delta V_p \) = maximum dendritic voltage swing during discrimination phase
- \( \Delta V_s \) = amplitude of a perturbation step
- \( \Delta V_l \) = lower margin
- \( \Delta V_h \) = higher margin

We have

\[
\begin{align*}
\Delta V_u &= (C_s/C_{tot}) V_{dd} \\
\Delta V_o &= (C_o/C_{tot}) V_{dd} \\
\Delta V_e &= n (C_s/C_{tot}) V_{dd} \\
\Delta V_p &= p (C_p/C_{tot}) V_{dd} \\
\Delta V_s &= (C_p/C_{tot}) V_{dd}
\end{align*}
\]

Choosing \( C_o = \frac{1}{2} C_s \) seems obvious. Next, we determine the perturbation capacitance \( C_p \) and the number \( p \) of those capacitances. We want to apply a perturbation such that each step corresponds to a specific Hamming distance, that is, neurons switching during a single step all have the same Hamming distance. To achieve this, we must set \( C_p = C_s \), so the amplitude of one step corresponds to a Hamming distance of 1. Also, we will need \( p = n + 1 \) perturbation steps to test all the possible Hamming distances, as the first step corresponds to Hamming distance of 0 (neurons with zero Hamming distance will have a dendritic voltage of \( V_{ref} + V_o \) ad thus will switch during the first perturbation step).

We must also ensure that no dendritic voltage will reach above \( V_{dd} \) or below \( V_{ss} \). Expressions for the margins are given by

\[
\begin{align*}
\Delta V_l &= V_{dd}/2 + \Delta V_o - \Delta V_p = V_{dd}/2 - (n + \frac{1}{2}) (C_s/C_{tot}) V_{dd} \\
\Delta V_h &= V_{dd}/2 - \Delta V_o - \Delta V_e = V_{dd}/2 - (n + \frac{1}{2}) (C_s/C_{tot}) V_{dd}
\end{align*}
\]
with

\[ C_{\text{tot}} = nC_s + C_o + pC_p + C_{\text{parasitics}} = \left( 2n + \frac{3}{2} \right) C_s + C_{\text{parasitics}} \]

Thus, the margins are always positive, and can be adjusted by adding a “parasitic capacitor” between row and ground.

### 3.2.4 An Exemplar-ranking discriminator

Another application is the ranking of exemplars, from the best-matching to the worse-matching. This can also be achieved with few additional circuitry. The basic operation and neuron sizing are exactly the same as for the discriminator with Hamming distance output, however, an additional rank counter is used. When one or more switchings occur during a perturbation step, the corresponding neuron are assigned the rank value stored in the counter, and the counter is incremented. This way, if several neurons have the same Hamming distance, they will all have the same rank.
Part II

VLSI Design of a Two-dimensional Hamming ANN for Image Processing Applications
4 Introduction

In this part we will describe the full-custom design of a 2D Hamming artificial neural network for image processing applications, using an original matrix structure based on the capacitive neuron described in the first part of this paper. We will describe an application of this ANN in a precision alignment system, and validate this application using high-level simulations.
5 Precision Alignment
System Application

5.1 Details of the chosen application

The targeted application is the use of the 2D-ANN circuit inside a precision alignment system, in which an input pattern must be precisely aligned on a previously stored pattern. The 2D-ANN will be used to deliver Hamming distances between rows and columns of the two patterns, which can be processed by peripheral units to generate control signals to the positioning system. Benefitting from the parallel processing capabilities of the ANN, an entire image can be processed in one cycle of a few tenths on nanoseconds, thus allowing very fast operation.

The block-level diagram of a the precision alignment system is depicted in figure 5.1a. It consists of an imager device (camera) feeding the ANN with the actual pattern information, which in turn outputs row and column Hamming distances. These are processed by a digital processing unit, which provides the control signals to the positioning system (motors). Figure 5.1b shows the envisaged external structure of the ANN.

Figure 5.1: Precision alignment system block diagram
5. Precision Alignment System Application

**Perspective** Future developments could eventually lead to the integration of CMOS imager devices as well as peripheral processing units inside the ANN chip, thus turning it into a stand-alone precision alignment control system.

### 5.2 High-level simulation

Before starting the lengthy process of designing a chip, we have done high-level simulations in order to validate the targeted application, that is, make sure the circuit will fit the needs of such an application. The simulations were done using MATLAB, and have proven that the hamming distance informations provided by the ANN are sufficient to achieve alignment of the input pattern over the stored pattern, using very simples algorithms.

**Correction in translation** MATLAB simulations were done using a very simple algorithm, which basically divides the pattern in two horizontal and two vertical halves. The corresponding hamming distances are summed, and the direction of the next move is determined by the gradient of the Hamming distances, that is, a move will be made towards the direction of smallest Hamming distance. Simulations with 16x16 patterns have shown rapid convergence when applying this algorithm with an initially translated pattern, as it can be seen on figure 5.2 where the image (a) represents stored pattern and the initial translated pattern, and image (b) shows the resulting aligned patterns. Depending on the initial position, 5 to 10 cycles were necessary to achieve convergence.

![Simulation of the translation correction algorithm](image)

**Figure 5.2:** Simulation of the translation correction algorithm (a) initial situation (b) final situation

![Simulation of the translation-rotation correction algorithm](image)

**Figure 5.3:** Simulation of the translation-rotation correction algorithm (a) initial situation (b) convergence of the translation algorithm (c) final situation
5. Precision Alignment System Application

**Correction in rotation** The preceding algorithm has been extended to process rotated patterns. A simple extension has proved to be able perform correction in both translation when the patterns were symmetrical along both x and y axes. The basic idea is that, for symmetrical patterns, when convergence of the translation algorithm is achieved and the Hamming distances are still not zero, then a correction in rotation needs to be applied. This algorithm has also proved to be successful with MATLAB simulations, as shown in figure 5.3.

**Extended algorithms** Extended algorithms have also been successfully tested which can achieve more rapid convergence, deal with noisy or corrupted patterns, and to a certain extent with non-symmetric patterns.
6 Design of a Test Chip

A full-custom test chip has been designed in CMOS 0.35\(\mu\) technology, which contains only the Hamming matrix without any peripheral circuitry. A secure design methodology has been adopted, so as to maximize the chances of ending with a working chip. In this perspective, complementary transmission gates have always been used, and transistors have been oversized when a risk was present.

6.1 Design of the core circuit

The core circuit consists in the 16x16 Hamming matrix, with the following functionalities:

- internal storage of two 16x16 patterns, accessed by 16 data inputs and 16 selection inputs
- application of a common analog perturbation signal to all neurons through an analog input
- output of the 32 digital neuron output signals
- external neuron command signals

The circuit is not stand-alone, as the outputs of the neurons must be processed by external circuitry. This will allow the test of the Hamming matrix with different post-processing implementations.

6.1.1 2D Hamming matrix organization

The two dimensional Hamming matrix (fig. 6.1) basically consists of a 16x16 array of basic cells, which are simultaneously applied to the synapses of two capacitive neurons, one along rows of the array and one along the columns. The basic cell performs the comparison (Hamming distance) between two bits belonging each to one of the two stored patterns. These bits are stored inside the basic cell, i.e. the memories are integrated inside the architecture of the Hamming matrix.
6. Design of a Test Chip

6.1.2 Design of the Hamming cell

Architecture The Hamming cell is basically composed of two memory cells to store the patterns and a XOR gate to perform the comparison, whose output is connected to the synapses of one row neuron and on column neuron. This architecture is depicted in figure 6.2. Additionally, two selection lines and two data lines are used for writing pattern data to the memory cells.

Memory cells design and sizing The inverters composing the memory cells are designed with minimum sized transistors. In the employed 0.35\mu technology, this means a size of \((W/L)_n = \frac{0.6\mu}{0.3\mu}\) for the NMOS transistors. Simulation of the inverters showed that the scaling factor for PMOS transistors is approximately 3.33, thus \((W/L)_p = \frac{2.0\mu}{0.3\mu}\).

Transmission gates sizing The transmission gates must be sized large enough to guarantee proper writing of data to the memory cells. Practically, this means that the memory cell must be pulled up above \(V_{dd}/2\), when writing a one, or pulled down below \(V_{dd}/2\), when writing a zero. Not taking into account the positive feedback, the computed dimensions are [6] \((W/L)_n = \frac{0.75\mu}{0.3\mu}\) and \((W/L)_p = \frac{1.9\mu}{0.3\mu}\).

XOR gate design and sizing The XOR gate has been designed using a common 8-transistors architecture. As the memory cells naturally delivers the data bit and its complementary, no additional inverters were needed. Minimum-size devices have been used, as for the memory cells. Note that the fan-out of this gate has an influence on timing, as it will provide the current to charge the synaptic capacitors. However, timing is not much an issue for this test chip, even though the simulations will show a fast operation anyway.
6.1.3 Design of the capacitive neurons

The capacitive neurons basically operate on 16 inputs, to which we add one perturbation input and one offset input.

**Synapses** According to the considerations of §3.2.3, weights have been set to

- +1 for the regular synapses
- +1/2 for the offset synapse
- −17 for the perturbation synapse

Then, the value of the capacitance $C_s$ of regular synapses has been chosen to 40fF, in order to achieve comfortable matching. Consequently, $C_o=20fF$, and $C_p=680fF$.

Regular synapses and offset synapse have positive weights, and thus are precharged to $V_{ss}$ through a NMOS transistor, which is of minimum size in order to reduce charge injection. Perturbation synapses are not internally precharged, this will be done by the external circuitry as they are directly connected to the analog pad.

**Inverters** The inverters are best kept small, since enlarging them increases also the parasitic capacitance of the dendrite. However, this effect is small compared to the capacitance contributed by interconnections. Moreover, enlarging the inverters will decrease the time needed to precharge the synapses. We decided to size them so as to attain a precharge time approximately equal to the
precharge time of the synapses. This has been done by simulation and led to
the following sizes

\[
\left( \frac{W}{L} \right)_n = \frac{5.0 \mu m}{0.3 \mu m} \quad \text{for the 2 NMOS transistors}
\]

\[
\left( \frac{W}{L} \right)_p = \frac{16.7 \mu m}{0.3 \mu m} \quad \text{for the 2 PMOS transistors}
\]

A number of progressively sized buffers have been inserted at the output of the
inverters to increase the driving capability.

![Neuron schematic](image)

**Figure 6.3: Neuron schematic**

### 6.1.4 Schematic simulation

Following is a sample simulation of one complete row of the Hamming matrix.
The schematic used consisted of one capacitive neuron, with 16 Hamming cells
as inputs. Capacitances of 500pF were added at each column nodes, to account
for the missing parts of the circuit. Other capacitances were added at nodes
where long metal lines would have to be drawn. Perturbation has been realized
as an analog stair shape, with 17 steps, so each step would correspond to one
different Hamming distance. Hamming distance value was swept from 0 to 16,
resulting in 17 row voltage curves and 17 corresponding output curves. Note
that at each perturbation step, the row voltage with corresponding Hamming
distance reaches below the inverter threshold, resulting in the output switching
as expected.

### 6.2 Layout

In this section, the full layout of the test chip is presented and discussed. The
technology used was a 0.35μ CMOS with 3 metal layers and a poly-poly capacitor module.
6. Design of a Test Chip

6.2.1 Layout considerations

Signals routing As a general guideline, the first metal layer and, to a smaller extent, the polysilicon layer, were used for routing of signals over small distances (inside cells). Circuit-wide routing has mostly been done with second and third metal layers, the former being used for horizontal routing and the latter for vertical routing.

The rows and columns of the Hamming matrix have been laid out with long polysilicon lines, POLY2 for the rows and POLY1 for the columns. The lines were made as thin as possible, to reduce parasitic capacitance (as well as coupling, see below) and were enlarged whenever a synaptic capacitor was needed. This method caused in fact numerous violation of the design rules.

Coupling between rows and columns As the rows and columns intersect each other, a capacitive coupling occurs between them. Practically, this means that a capacitor is formed from the intersection of the row and column polysilicon lines. This coupling can be theoretically analysed by considering each coupling capacitor as an additional synapse. Each row neuron would then have 16 more synapses, the input of which would be the row voltage of the 16 columns. In the same way, each column neuron would receive contribution from the 16 row voltages. We can thus express the variation of dendritic voltage during evaluation phase for row i as the sum of regular synaptic contributions and a coupling contribution

\[
\Delta V_{R_i} = \frac{1}{C_{tot}} \sum_{j=1}^{n} C_{ij} \Delta V_{ij} + \frac{C_c}{C_{tot}} \sum_{j=1}^{n} \Delta V_{C_j} = \frac{1}{C_{tot}} \sum_{j=1}^{n} C_{ij} \Delta V_{ij} + \delta V_c
\]

Figure 6.4: SPICE simulation of one row of the matrix

![Figure 6.4: SPICE simulation of one row of the matrix](image-url)
with \( i \) the column index, \( j \) the row index, \( n \) the number of rows and columns, \( C_c \) the value of the coupling capacitor and \( C_{ij} \) and \( \Delta V_{ij} \) the synaptic capacitance and synaptic voltage variation at row \( i \) and column \( j \) of the matrix. The first term represents the offset, which is common to all rows and columns. The second term is contribution from all input synapses, and the last term is the contribution due to coupling. Note that \( C_c \) is included in the total capacitance, which is now \( C_{tot} = \left( 2n + \frac{3}{2} \right) C_s + nC_c \). Similarly, for column \( j \),

\[
\Delta V_{C_j} = \frac{1}{C_{tot}} \sum_{i=1}^{n} C_{ij} \Delta V_{ij} + \frac{C_c}{C_{tot}} \sum_{i=1}^{n} \Delta V_{R_i}
\]

\[
= \frac{1}{C_{tot}} \sum_{i=1}^{n} C_{ij} \Delta V_{ij} + \delta V_c
\]

By summing \( V_{R_i} \) over all rows and \( V_{C_i} \) over all columns, we obtain

\[
\sum_{i=1}^{n} \Delta V_{R_i} = \frac{1}{C_{tot}} \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} \Delta V_{ij} + n \frac{C_c}{C_{tot}} \sum_{j=1}^{n} \Delta V_{C_j}
\]

\[
\sum_{j=1}^{n} \Delta V_{C_j} = \frac{1}{C_{tot}} \sum_{j=1}^{n} \sum_{i=1}^{n} C_{ij} \Delta V_{ij} + n \frac{C_c}{C_{tot}} \sum_{i=1}^{n} \Delta V_{R_i}
\]

This system can be solved and lead to the (quite surprising) conclusion that the coupling contribution \( \delta V_c \) is equal for all rows and all columns, and depends only on the sum of synaptic contributions over the whole matrix.

\[
\delta V_c = \frac{C_s C_c}{C_{tot}^2} - \frac{1}{1 - n \left( C_c / C_{tot} \right)} \sum_{i} \sum_{j} \Delta V_{ij} \leq \frac{C_s C_c}{C_{tot}^2} - \frac{1}{1 - n \left( C_c / C_{tot} \right)} n^2 V_{dd}
\]

Knowing the worst-case coupling contribution, we can compute a minimum value for \( C_s \)

\[
C_s \geq \left( \frac{n}{2n + \frac{3}{2}} \right)^2 \frac{V_{dd}}{\delta V_c,_{\text{max}}} - \frac{n}{2n + \frac{3}{2}} C_c
\]

which, for our circuit, was less than the chosen 40pF value.

### 6.2.2 The core circuit

The layout of the core circuit (fig 6.5) was basically made from one basic cell, which comprised the Hamming cell and the corresponding row and column synapses, which was repeated as a 16x16 array to form the core of the matrix. This cell has determined the pitch of the array (30\( \mu \)m), and further layouts were accommodated to fit these dimensions. The total area occupied by the core is less than 1mm\(^2\).

**The Core cell** The Core cell layout comprises the Hamming cell, the synapses with their capacitors and many circuit-wide lines, including the row and column dendrites themselves (horizontal and vertical polysilicon lines), data to load the memories and various control signals. This has all been laid out such that, when several cells are placed side-by-side, the lines propagate through the structure.
The cell has been laid out as square as possible, so the resulting width and height of the matrix would be equal. Selection lines for the memory cells were routed horizontally with MET2, while bitlines and other controls signals were routed vertically with MET3. Numerous substrate contacts were placed when possible. The resulting size of the cell is 30µm x 30µm.

Other cells All other cells have been designed to be placed one the sides of the array, so their maximum height (or width) was fixed by the size of the core cell. They all decline in two different variations, one for the rows and one for the columns. The layouts can be found in appendix B

6.2.3 Post-layout simulations

Post-layout simulations have been done on one row of the matrix, as for the schematic simulations. They exhibit slightly more parasitics, due to capacitive couplings, but the circuit operation is still very good.
6. Design of a Test Chip

6.3 Floor planning

Packaging  Due to the large number of input and outputs signals, we planned to package the circuit in a 100 pins PGA package, with a maximum die size of 100mm². 94 pads have been used:

- 16 data inputs (to the memory cells)
- 16 selection inputs (selection of the memory cells)
- 1 $X/Y$ input (writing to X or Y memory cell)
- 1 $\Phi_1$ input
- 1 $\Phi_2$ input
- 1 analog perturbation input
- 32 outputs

Figure 6.6: Layout of the core cell
6. Design of a Test Chip

Figure 6.7: Post-layout simulation of one row of the matrix

- 16 pads for pad alimentation
- 4 pads for core circuit alimentation
- 4 pads for input inverter alimentation
- 2 separate pads for analog pad alimentation

**Inverted inputs**  $\Phi_1$, $\Phi_2$ and all data inputs have to be inverted, so large inverters have been laid out next to the corresponding pads. These buffers have dedicated pads for alimentation.

**Selection signals**  Separate selection signals have had to be generated from the input selection signals and $X/Y$ signal. This has been done using standard AND and NAND cells (in fact, the only standard cells used in this design) which have been placed next to the core circuit in order to reduce their load capacitance.
6. Design of a Test Chip

6.4 Full circuit layout

Figure 6.8: Full circuit layout
7 Conclusion

During this work, a circuit has been designed. The full design-flow has been experimented, from the schematic entry to the post-layout simulation and the floorplanning. The circuit is original, it presents a completely new architecture of discriminator for Hamming neural networks, which offers a broad range of possibilities. An application in precision alignment systems has been proposed and validated. Perspectives include the manufacturing and test of the chip, and eventually the design of a fully integrated stand-alone Hamming ANN circuit.
References


Part III

Appendices
Appendix A  Publications

Two articles were written and submitted for publication during this work, one of which has already, at the time of writing, been accepted.

Accepted for publication  Stéphane Badel, Alexandre Schmid and Yusuf Leblebici, VLSI Realization of a Two-Dimensional Hamming Distance Comparator for Image Processing Applications, 11th European Symposium on Artificial Neural Networks, ESANN’2003, Bruges, Belgium, April 2003.

Submitted for publication  Stéphane Badel, Alexandre Schmid and Yusuf Leblebici, A VLSI Hamming Neural Network with k-Winner-Take-All and k-Loser-Take-All Capability, International Joint Conference on Neural Networks, IJCNN’03, Portland, OR, July 2003.
Appendix B  Additional Layouts

Buffers
Appendix B. Additional Layouts

Offset cell
Appendix B. Additional Layouts

**XOR gate**
Appendix B. Additional Layouts

Memory cell
Appendix B. Additional Layouts

Zoom on part of the core