Computation, Lecture Notes in Computer Science 686, Springer-Verlag, Berlin (1993), pp. 488-493

HARDWARE DESIGN FOR SELF ORGANIZING FEATURE MAPS WITH BINARY INPUT VECTORS

S. Rüping, U. Rückert and K. Goser

University of Dortmund
Dept. of Electrical Engineering
P.O. Box 500500
W-4600 Dortmund 50
Germany

Abstract:

A number of applications of self organizing feature maps require a powerful hardware. The algorithm of SOFMs contains multiplications, which need a large chip area for fast implementation in hardware. In this paper a resticted class of self organizing feature maps is investigated. Hardware aspects are the fundamental ideas for the restictions, so that the necessary chip area for each processor element in the map can be much smaller then before and more elements per chip can work in parallel. Binary input vectors, Manhatten Distance and a special treatment of the adaptation factor allow an efficient implementation. A hardware design using this algorithm is presented. VHDL simulations show a performance of 25600 MCPS (Million Connections Per Second) during the recall phase and 1500 MCUPS (Million Connections Updates Per Second) during the learning phase for a 50 by 50 map. A first standard cell layout containing 16 processor elements and full custom designs for the most important parts are presented.

1. Introduction

Self organizing feature maps as described in [1] have applications in the fields of data analysis and pattern recognition. Especially for real time application a fast and efficient hardware is needed. Well known "von Neumann computers" work serially on each element of a self organizing feature map, so that they have to be very fast and can work only on rather small maps when time is the critical aspect.
A speedup is possible when a calculation unit is available for each element of the map or a special coprocessor is used. There are a number of hardware designs known in literature [e.g. 2, 3, 4]. Due to the mathematical operations which are necessary for the algorithm of self organizing feature maps, calculation units are rather complex and need large chip area. For increasing map size multiplexing of the available units is required to restrict the number of chips.
The basic idea of this paper is to look at a special class of self organizing feature maps. The aim is to get a most simplified hardware by making restrictions on the algorithm and the used operations. For example binary input vectors and Manhatten distance make it possible to calculate the distance between two vectors without multiplication. Chapter 2 will describe all the simplifications according to hardware aspects which are made for the presented circuits.
The investigation leeds to a hardware design that is described in chapter 3 and 4. Simulations using the hardware description language VHDL result in numbers for the performance of the hardware. This is discussed in chapter 5. Different layouts are already designed. First there is a standard cell layout containing 16 processor units for self organizing feature maps and second there are full custom layouts for the most important parts of the unit. Chapter 7 will close the paper dicussing the results.

2. Simplifications according to hardware

The algorithm of self organizing feature maps is summarized in equation 1.1. The calculation of the weight vector for the time t+1 is made with the old weight vector w(t), the distance between the old weight vector and the input vector and an adaptation factor alpha.

$$w(t+1) = w(t) + \alpha(t,x,y)\cdot(x(t) - w(t))$$

$w(t+1);\ w(t+1)$	:	new and old weight vector
$\alpha(t,x,y)$	:	adaptation factor
t	:	time
x, y	:	coordinates of the element

(1.1)

The distance between the weight vector and the input vector is very often defined by the euclidean distance which is calculated by

$$(x - w) = \sqrt{\sum_i (x_i - w_i)^2}$$

(1.2)

Due to the difficulties to realize the square root in hardware and to the fact that not the absolute value is important but the relation between all the distances on the map, most implementations use the distance

$$(x - w) = \sum_i (x_i - w_i)^2$$

(1.3)

where the square root is not used. But nevertheless multiplications are needed. A distance which is very simple to implement in hardware is shown in equation 1.4. It is called the Manhattan distance and can be realized by a modified adder and a register.

$$(x - w) = \sum_i |x_i - w_i|$$

(1.4)

Further multiplications are necessary for the product of the adaptation factor alpha and the calculated distance. A restriction to discrete values of alpha can simplify this multiplication. Alpha has a value between 0 and 1. If only the values shown in 1.5 are allowed, the operation can be handled by a shifter.

$$\text{alpha} \in \left\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \ldots..\right\}$$

(1.5)

A multiplication with 1/2 is the same as shifting one bit to the right. Which means the restricted class of self organizing feature maps using the Manhattan distance and discrete values for alpha can be designed in hardware without using multipiers.
Another simplification that decreases the number of necessary I/O ports and the chip area can be done by restricting the components of the input vector to be binary.

$$x = (x_1, x_2, \ldots, x_n)^T \qquad x_i \in \{0,1\} \qquad \text{input vector}$$
$$w = (w_1, w_2, \ldots, w_n)^T \qquad w_i \in \{0,1,2,3,\ldots.,2^4 - 1\} \qquad \text{weight vector}$$

(1.6)

The precision of the weights is set to 4 bits. First simulations show that this seems to be a reasonable value but further investigations have to be done in this field. For the definition of the distance equation 1.7 is given. The bit of the inputvector codes the minimum and the maximum of the possible weight values.

$$|x - w| = \sum_{i=1}^{n} |x_i \cdot (2^4 - 1) - w_i|$$

(1.7)

3. Hardware design

The restrictions described in chapter 2 allow a simple hardware design with very few ports between the units. Figure 1 shows an example of a 2 by 2 map, which can easily be extended to larger map sizes. Each unit, in Figure 1 called PE (Processor Element), has a port for the row, the column, the clock and data. A three bit command bus sends the instructions to the units.

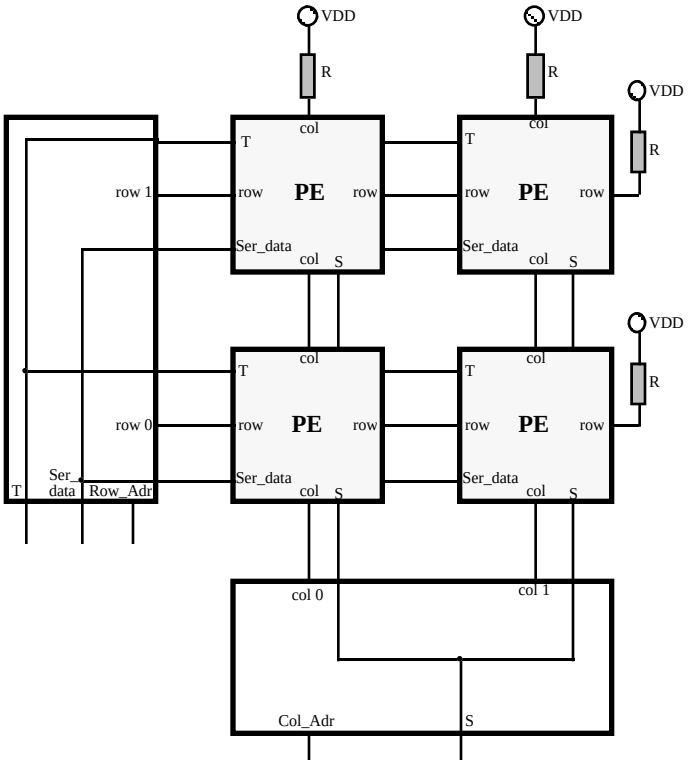

Fig. 1: Example for a 2 by 2 map

Each column and row line is connected to a seperate pull up resistor. The unit's outputs can be 'low' or can have a high impedance. So it is possible to realize a 'wired or' and use this for the minimum search. After calculation of the distance all units start to count down the distance register (see chapter 4). The unit, which is the first to be zero, put a low voltage on it's row and column line. At this time the control logic knows the minimum unit and it's position. If there are more than one unit that are the minimum at the same time, a request to the elements can solve the problem.
All units are designed in the same way. Therefore the system can easily be extended by adding new rows and columns to the map, when the control logic has the capability to handel the number of rows and columns.

4. Internal structure

The internal structure of the units is shown in Figure 2. The connections of the unit described in chapter 3 can be found above the controller. The main blocks are the calculation block, the weights, sum and alpha register and the controller. The weight register stores the weight vector and presents a component per clock cycle to the calculation block. The calculation block produces the distance of the corresponding input and weight components and add it to the sum register. During the learning phase it calculates the new weight value using the bits in the alpha register. All this is done in one clock cycle.
The input vector is send to all units via the data line. One component per clock cycle is handled and all units work in parallel. In order to decrease the chip area the input vector components are not stored in the units. They must be sent again during the learning phase. Investigations have shown that this is no disadvantage. Due to the binary type of the input vectors it is possible to send these bits as fast as the unit could read them from an internal register.
To keep the design most flexible, the adaptation factor is send before each adaptation step. That makes it possible to control the adaptation function and neighborhood size without restrictions. On the other hand all units with the same alpha can work in parallel during the adaptation step. This will be explained in more detail in chapter 5.

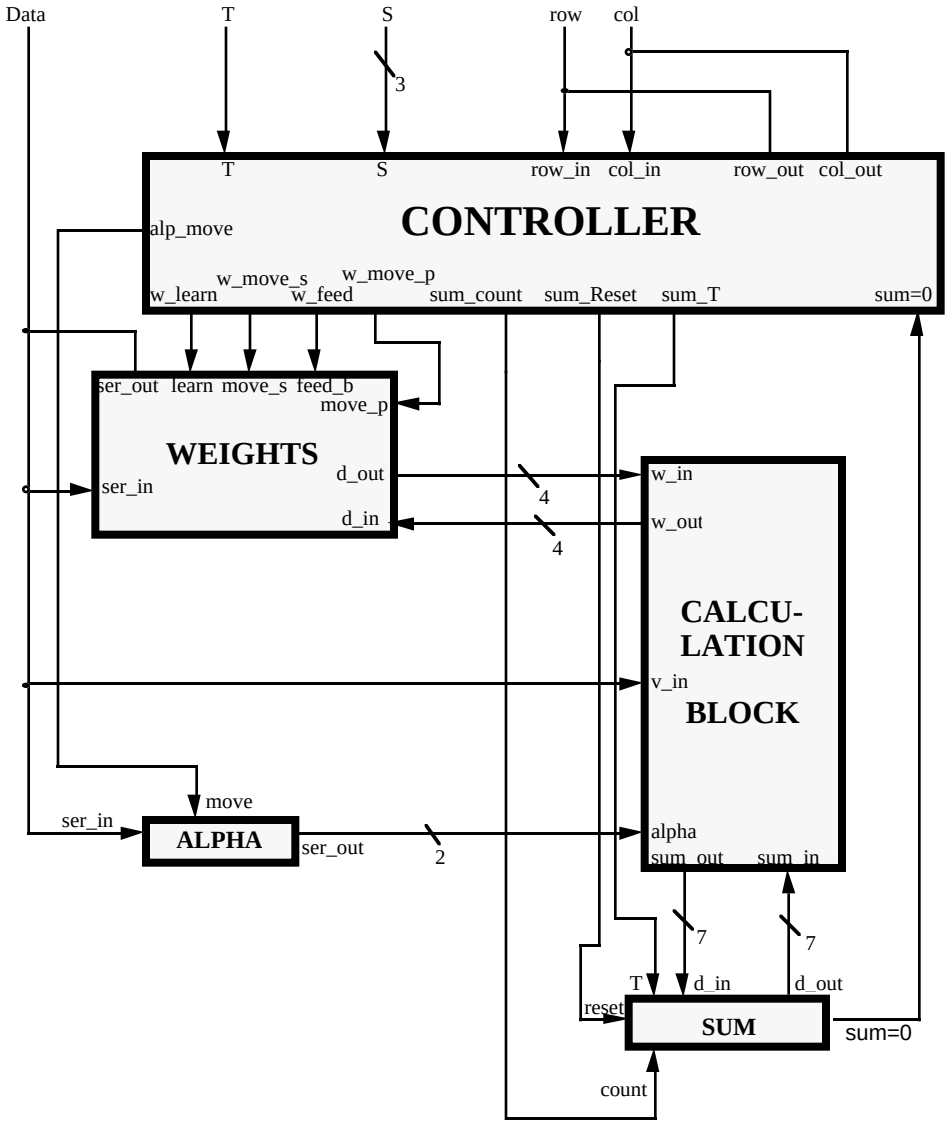

Fig. 2: The internal structure of the unit

5. Performance of the system

As mentioned before the unit is capable of processing one component per clock cycle, which means the system needs 64 clock cycles for calculating all distances on a map for an input vector with 64 components. The number of clock cycles to find the minimum depents on the minimum distance. The distance might be small for an already learned map, so for this example an average value of 36 is taken, which leads to a result of 100 clock cycles for each input vector finding the position of the most similar element. Figure 3 lists the performance for a 50 by 50 map working at 16 MHz.

Mapsize	Components	Clock Cycles	MCPS
50 x 50	64	~ 100	25600

Fig. 3: Performance during recall phase

During the recall phase all units can work in parallel. This is not possible during learning phase, when only units with the same value of alpha can calculate the new weight vectors at the same time. Figure 4 explains the process. First the minimum element adapts it's weights and switches into inactive mode. The element was addressed by the row and the column lines. Then a field of 3 by 3 units is addressed with 3 row and 3 column lines. This field can now adapt in parallel while the central unit (the minimum) is

inactive and will not adapt a second time. The working principle is used for all further rings until the whole adaptation function has been processed. Each unit switches to inactive mode after the adaptation step.

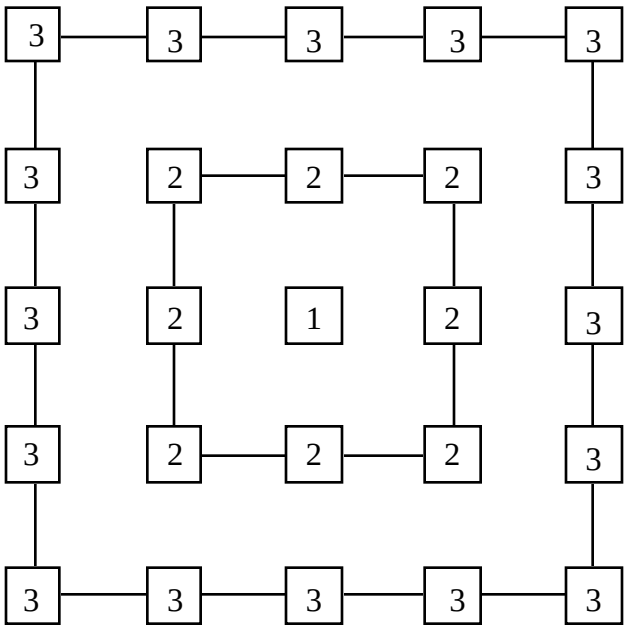

Fig. 4: Addressing the units for adaptation

VHDL simulations for a map working at 16 MHz has been used to produce the performance numbers listed in Figure 5.

Adaptation Size	Components	Clock Cyles	MCUPS
50 x 50	64	1716	1500

Fig. 5: Performance during learning phase

6. Layout of the unit

A 1.5 um CMOS technology has been used to design the different parts of the proposed unit in full custom layout. The calculation block has a size of about 0.15 mm^2. The weight register with a capacity of 256 bits (64x4) needs about 1.85 mm^2. Both are manufactured at the moment and will be tested soon. A 10 by 10 mm chip could contain about 25 units, where the majority of the area is used by the on chip memory. Further versions of the units will be designed in 0.8 um CMOS technology, so that the number of units per chip will increase.
On the other hand a standard cell layout containing 16 processing units (each with 64 bits (8x4) memory) has been developed and is shown in Figure 6. The cells are based on a 2 um CMOS process. The chip has a size of 10 by 10 mm and works with 16 MHz clock rate.

7. Discussion

A hardware design for self organizing feature maps is presented in this paper. Only a restricted class of feature maps can be handle by the hardware, but with this restrictions it is possible to simplify the design and decrease the necessary chip area. It seems, that there are a number of applications where binary data is processed. On the other hand it might be possible to find an efficient coding for continous data. This would increase the number of applications.
The hardware is very simple to extend. A chip containing full custom designed units is expected to contain a much higher number of units than the standard cell layout. Therefore large map sizes with all elements working in parallel can be built.

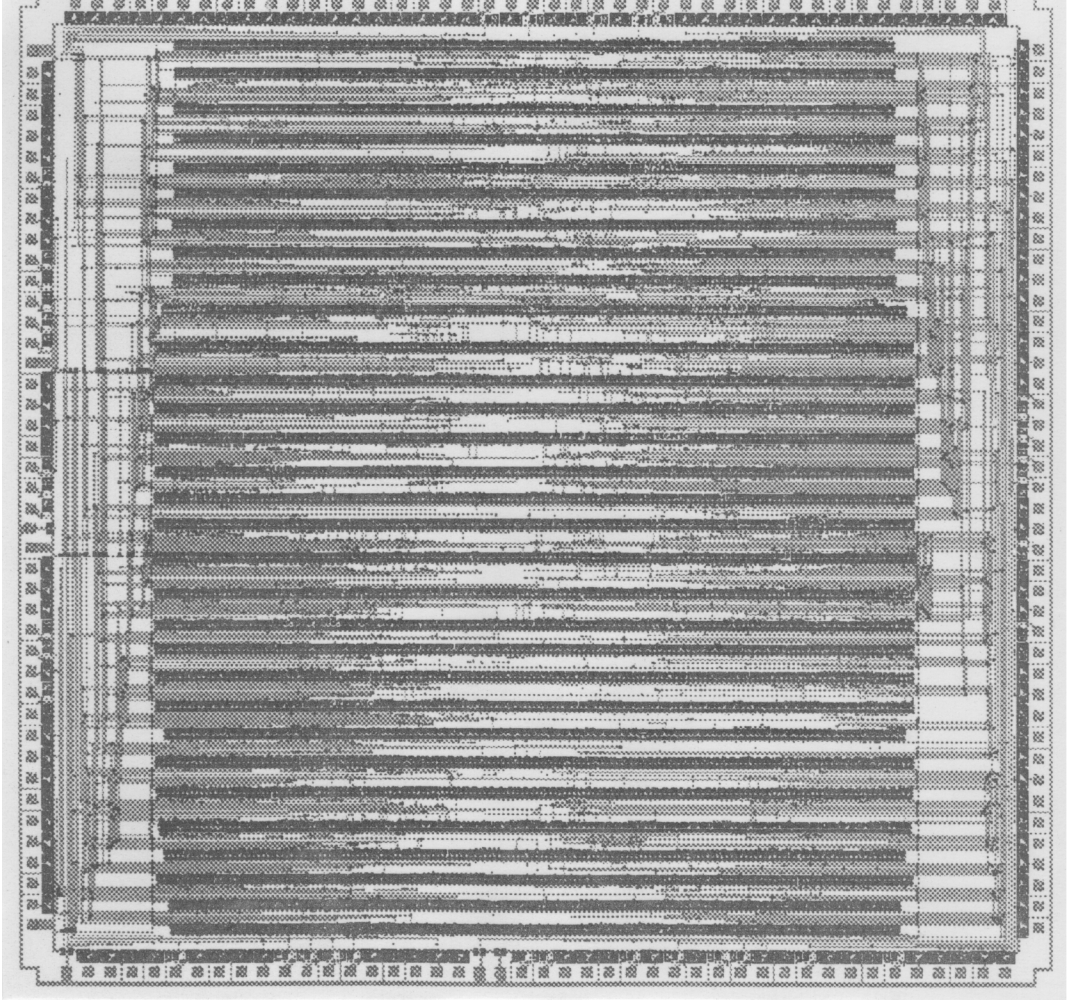

Fig. 6: Standard cell layout of 16 units (10x10 mm, 2 um CMOS)

A software for simulating the restricted class of self organizing feature maps is tested at the moment and will be used to investigate the necessary weight precision and other specifications of the maps.

8. Acknowledgement

This work has been partly supported by the german ministry of research and technology BMFT, contract number 01-IN 103 B/O.

9. References

[1] T. Kohonen. *Self-Organization and Associative Memory*. Springer Verlag Heidelberg New York Tokio, 1984

[2] D. Hammerstrom, N. Nguyen. *An Implementation of Kohonen's Self-Organizing Map on the Adaptive Solutions Neurocomputer*. Proceedings of Artificial Neural Networks 1991, pp. 715 - 720. North-Holland, 1991

[3] V. Tryba. *Selbstorganisierende Karten: Theorie, Anwendung und VLSI-Implementierung*. (in German). Dissertation an der Universität Dortmund, Abteilung Elektrotechnik. 1992

[4] U. Ramacher, U. Rückert, J.A. Nossek. Proceedings of the 2nd International Conference on Microelectronics for Neural Networks. Kyrill &Method Verlag, München, 1991