A VLSI Associative Processor for Neural-Like Classification Algorithms

Philippe Thissen, Michel Verleysen and Jean-Didier Legat
Université catholique de Louvain,
Microelectronics Laboratory - DICE,
3 Place du Levant, B-1348 Louvain-La-Neuve, Belgium
thissen@dice.ucl.ac.be

Abstract

This paper describes a parallel SIMD processor dedicated to classification tasks. The processor is based on a bit-serial, word-parallel associative architecture. Dedicated resources have been added in order to allow efficient implementation of neural-like algorithms for classification. A full custom VLSI implementation of this processor has been fabricated in a standard 1 μm *CMOS technology and the die size is* 55 mm^2. *The peak performance is about* 1 250 *millions of integer 8-bits additions per second for an array of 128 PEs running at 100 MHz.*

1. Introduction

In this paper, we describe a massively parallel processor adapted to classification by neural-like algorithms. The processor is based on a SIMD bit-serial, word-parallel associative architecture and implements the data path of a general classification machine.

The classification machine is composed of two parts: a control unit and an operative part as illustrated in figure 1.

- A programmable logic device (FPGA) has been chosen to implement the control unit to take advantage of the programmability and rapid reconfigurability of such device in comparison with more classical ASIC solutions. The aim of this control unit is to split up a high level instruction (like a 8-bits multiplication) into several micro-instructions directly processed by the operative part of the classification machine. The FPGA also handles the communications with the host computer which sends high level instructions.
- The operative part or data path is composed of one or several associative processors (circuits).

The following of the paper insists on the functionality and the implementation of these associative chips.

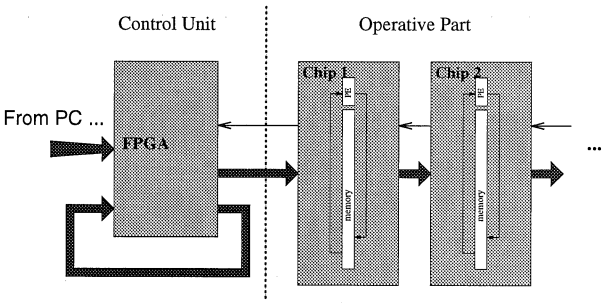

Figure 1. General view of the classification machine composed a control unit (a FPGA) and a data path (one or several associative processors).

The associative processor is made of many simple processing units working in parallel; each unit realizes simple operations like bit-serial additions or comparisons. Some specialized cells, taking as inputs the outputs of all elementary processing units, have been added to process collective computations needed by the algorithms; these dedicated resources have been chosen in order to speed up classification tasks. However this associative architecture is multi-purpose and different applications can run on this machine only by writing the related microprogram.

The operative part of the processor has been fabricated in a 1 μm CMOS technology. The elementary ALU and the static memory cell have been successfully simulated at frequencies up to 100 MHz. The size of the different blocks of operative part allows to build a parallel processor formed by 128 elementary processing units, each unit provided with 256 bits of memory, in 55 mm^2. The parallel processor is expandable and larger networks can be realized by connecting several

circuits together.

In this paper, we first briefly present the functional behavior of the associative processor; the different parts of the elementary processing unit and the operations realized by these parts are described. Results are given about the performances of this architecture for simple mathematical operations. The second part of this paper deals with the implementation of the architecture and extends the description of the elementary processing units. Finally, performances for neural algorithms are presented; as an example, the learning and the recognition phases of RCE and LVQ algorithms are illustrated on the proposed architecture.

2. Functional description of the associative processor

The associative architecture described in this section implements the data path (or operative part) of a global classification system. This system is composed of many simple processing units working in parallel; each unit includes a memory, a elementary processor and a status register. To have the parallel structure working into one common goal, some interactions exist between the elementary processors.

Figure 2 shows the global architecture of a bit-serial, word-parallel associative processor [3, 6]. Each processing element (PE) exchanges information with its associated memory (on the same horizontal line). A special 1-bit register (S) fixes the status of the elementary processor; all data transfers are made through a 1-bit data bus.

The original part of this approach resides in the structure of the associative processor well suited to neural-like classification. The associative memory and the elementary processing unit are designed according to the most frequent and complex operations involved in this task. Moreover, specialized hardware has been added to achieve collective calculations. In the following, a description of each part of the processor is given in more details.

- The memory used is a random access memory with two separate busses (read and write). At each clock period, two addresses are needed. Both addresses, coming from outside of the circuit, are propagated to all rows of memory. This two-busses architecture allows to execute two different operations on two different data at the same clock period: read and write in the memory.

- The arithmetic and logic unit (ALU on figure 2) is built to process simple data in a bit-serial way. This part of the elementary processing unit is able to execute addition, subtraction and comparison between a data coming from the memory and another one coming from the outside of the circuit, in one clock period; it has also the possibility to carry on arithmetic and logic operations on two data coming both from the memory in two clock periods. The inputs are taken in the input buffer of the ALU and the result is written in its output buffer. The operations are identical for all elementary processing units and are executed depending on the value of the status register.

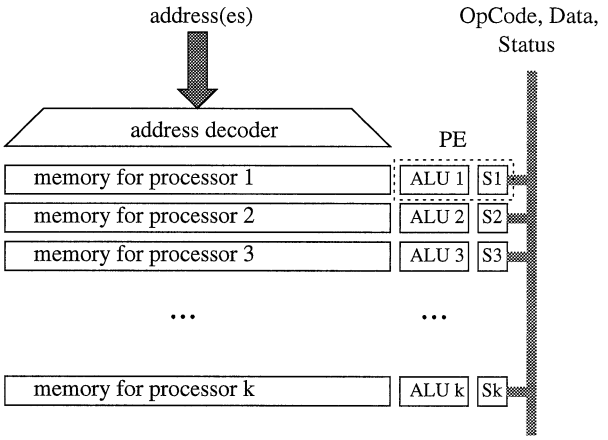

Figure 2. The bit-serial, word-parallel associative architecture.

- The status register (S) allows the current instruction to be processed or not. This status information can be loaded from and written in the memory. The result of a computation (arithmetic or logic) can be directly stored in the status register.

Figure 3 gives a schematic representation of the elementary processing unit. In one clock period, three different operations can be realized at the same time on different data:

- reading a bit from the memory and loading it in the input buffer of the ALU;

- computation of an arithmetic or logic operation on the inputs of the ALU and storage of the result in its output;

- writing the content of the output buffer in the memory.

This pipelining multiplies by 3 the number of operations realized by time unit in comparison with the clock frequency.

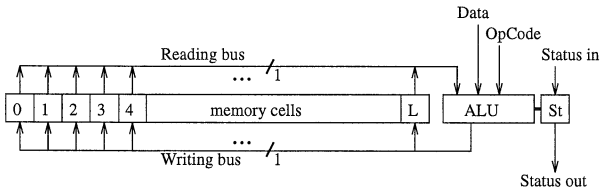

Figure 3. Schematic representation of one PE and the associated memory.

Collective functions on the outputs of all elementary processing units have been added to the associative structure. These functions, dedicated to speed up classification tasks, come from the study of classification algorithms. Two functions are particularly useful in these tasks: *the selection of the first active PE* and *the summing of all their outputs.*

- The first function (*the selection of the first active PE*) is a combinatorial block (without synchronization): a token goes through the structure (all processors) and stops when it finds the first basic PE with the state register set to 1. Then this elementary processor becomes active and all others are turned off. To avoid large delays due to a very long chain of elementary processors, a method, similar to the carry look-ahead for adders, is used to bypass the normal way of the token.

- The second collective function sums all outputs of the elementary processors. This function is useful for counting the number of active processors but also for summing the contributions of all processors to a common goal. This function is built with 1-bit full-adders; the adders are laid in a tree structure and signals go through the tree in a pipelined way. A each clock period, all data are processed to the next level in the tree; the sum is obtained in a bit-serial way at the output of the last adder of the tree.

Moreover, the links between the ALUs and the bank of memory are programmable to some extend. In fact, each PE is able to read and/or write information from/to the memory line located just above, in front or below it. By this property, the memory of each PE can be extended up to 768 bits and systolic chain computations are feasible on this processor.

The architecture is multi-purpose and expandable. In fact, the processing unit executes basic but fundamental functions (addition, subtraction and comparison). It is expandable because the number of processing units is not fixed to a predefined value.

Each elementary processor is able to compute bit-serial additions without any interaction with the others ones. By extension, the three others basic integer operations (subtraction, multiplication and division) are feasible on this device [5]. The execution of such operations on a floating point representation is also possible by separating the number into two parts (mantissa and exponent) and processing the adequate function to each part. Table 1 illustrates the number of operations that can be executed on this architecture composed of 1024 elementary processors clocked at a frequency of 100 MHz; the results are given in million of operations (additions, multiplications ...) per second (MOPS). For floating point operations, numbers are given in a 32-bits IEEE format with a 24-bits mantissa and an 8-bits exponent. Comparison operations have the same complexity as an integer subtraction.

3. Implementation of the processor

In this part, indications are given about the hardware implementation of the processor. The two main blocks are described: the static memory and the ALU. The section is concluded by some considerations about the whole circuit and the position of the different parts of the chip.

3.1. The SRAM memory

The static RAM array is horizontally divided in banks of memory and each bank is associated to an elementary processing unit. The memory is based on a standard static RAM cell composed of 8 transistors (4 for the flip-flop and 4 for the accesses).

3.2. The elementary processing unit

The core of this architecture is the elementary processing unit. This PE receives 2 bits of data, one from the associated memory and another one, common to all PEs, from the outside of the chip. The PE processes arithmetic and/or logic operations on these data and produces a 1-bit result sent back to the memory and to the outside of the circuit.

In fact, the ALU is divided into 7 macro-cells illustrated at figure 4. To allow the pipelining of the data, latches have been added in the input buffer (block 1) and in the output selection (block 5).

The input buffer block (1) memorizes the bit from the memory associated to this PE in a D-latch. By this way, the bit read during previous clock cycle in the RAM is used for the computation during

	integer			real
	8 bits	*16 bits*	*32 bits*	*32 bits*
addition				
$X + A = C$	10 240 MOPS	5 687 MOPS	3 012 MOPS	80 MOPS
$B + A = C$	5 687 MOPS	3 012 MOPS	1 552 MOPS	80 MOPS
subtraction				
$X - A = C$	9 310 MOPS	5 390 MOPS	2 925 MOPS	80 MOPS
$B - A = C$	5 390 MOPS	2 925 MOPS	1 527 MOPS	80 MOPS
multiplication				
$X \times A = C$	397 MOPS	122 MOPS	35 MOPS	57 MOPS
$B \times A = C$	317 MOPS	92 MOPS	25 MOPS	42 MOPS
division				
$X/A = C$	132 MOPS	40 MOPS	12 MOPS	20 MOPS
$B/A = C$	112 MOPS	35 MOPS	10 MOPS	17 MOPS

Table 1. Computational power of the massively parallel processor formed by 1024 elementary processing units clocked at 100 MHz; one MOPS is equivalent to one million of operations per second.

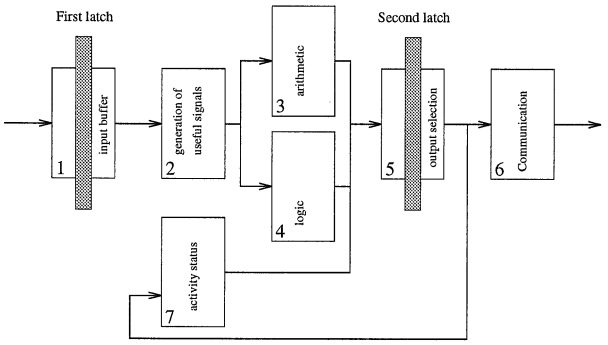

Figure 4. Schematic representation of the macro-cells inside a PE and links between these cells.

the present clock cycle. The D-latch is designed in a TSPC (True Single Phase Clock) technique [8, 9, 7] which reduces the size of the devices (less transistors) and simplifies the clock planning (no clock skew problem because only a 1-phase clock synchronizes the whole logic).

The generation of useful signal block (2) prepares the data to be processed. This block inverts eventually one signal in case of subtraction or comparison; it memorizes the bit coming from the memory and sends it one clock cycle later in place of the external input, when computations involve two data from the memory. In this case two clock periods are needed to perform one computation.

The arithmetic block (3) is composed of a full adder and a D-latch to memorize the carry when processing serial additions or subtractions.

The logic block (4) has been designed to perform fast comparisons. All data are compared from the MSB to the LSB. A 2-bits result warns of the status of the comparison operation (finish or not) and its decision (for example greater or not).

The output selection block (5) selects one from the four possible outputs: one from the arithmetic block, two from the logic block and one from the activation status block. A 4 to 1 multiplexer memorizes one of these outputs in a D-latch. This output bit will then be written back to the memory at the next clock period. An inactivate PE will have its output low all the time.

The communication block (6) sends signal through a 1-bit bus outside of the chip. The output of each PE is connected to a global bus which imposes a high logic level if there is at least one PE output high and a low logic level only when all the PEs outputs are low. This option is very powerful for searching extremum and others comparison operations.

The activity status block (7) fixes the activity of each PEs. The same instructions are received by all PEs but they are conditionally processed depending on the activity status. In fact, the associative processor has 3 different operating modes.

- In *normal mode*, the activity of each PE is stored in a status register.
- In *forced mode*, all PEs are active independently of the bit in the status register.
- In *one active mode*, only one PE is activate and all others are turned off. This possibility is used to get sequentially to each PE and thus to each cell of memory.

The output of the PE can be stored in the status register and the value memorized in this register is able to be sent out of the PE and written back in the memory.

3.3. General view of the circuit

A prototype circuit of this associative architecture has been designed in a $1\,\mu m$ CMOS technology. The whole circuit is composed of 128 elementary processing units with 256 bits of static RAM each. Figure 5 gives a schematic representation of this parallel processor (left) and the chip micro photograph (right). Without pads, the dimensions of this prototype circuit are $6.5\,mm$ high on $6.3\,mm$ wide. With the 60 pads and the guard rings, the dimension are $7.5\,mm$ high $7.5\,mm$ wide, that is a surface of $55\,mm^2$. The circuit has been packaged in a 84 pins PGA socket.

4. Performances of classification algorithms running on this processor

Some neural-like classification algorithms have been slightly adapted to fit into this processor. In such algorithms, a lot of computations can be realized in parallel without any interaction between the different processing units. During the learning phase, the network is progressively built according to some rules to represent the input distribution of patterns; then the classification phase compares an input pattern to all stored patterns and makes some computations on these distances.

Two types of neural-like classification algorithms have already been implemented into this parallel processor: RCE and LVQ algorithms.

4.1. The RCE algorithm

The RCE algorithm proposed by Reilly, Cooper and Elbaum [4] is an incremental model of neural classifiers. During the learning, an input vector is presented to the network and neurons fire if this input vector belongs to their region of influence. Neurons leading to misclassification are modified (the size of their region of influence is decreased) and a new neuron is added if the input element is not well classified.

Each elementary processing unit in the architecture represents a neuron and performs the whole computation of this neuron; the center and the radius of the concerned neuron are stored in the memory associated to the processing unit. The radius of the elementary units leading to misclassification are modified and new elementary units are added if necessary. To simplify the computation, the Manhattan distance metric is used.

Simulations of our architecture on the RCE algorithm have been carried out on a bidimensional database containing 1000 elements divided into two separated classes (a circle and an external ring). The data accuracy has been truncated respectively to 5 and 16 bits. Table 2 gives the number of vectors that can be learned per second, and the percentage of time used for the distance computation on it. Values illustrated in the *"Simulations"* column represent the mean value for 1000 presentations of the whole learning set; those indicated in the *"Theory"* column represent a theoretically computed minimum number of vectors learned/classified per second.

During the generalization phase of this algorithm, three operations are performed. First, the distance between the input pattern and all the stored patterns are computed and only the neurons which include the input pattern are set active. Then, the class of the vector to classify is randomly set to the class of one of the active patterns. Finally, it must be verified that all active processors propose the same class; if not, no solution exists for the given input. Table 2 gives the theoretically computed and simulated number of vectors that can be classified per second.

4.2. The LVQ algorithm

The Learning Vector Quantization (LVQ) method, proposed by Kohonen [1, 2] is an adaptative vector quantization algorithm which is used for classification purposes by adding an information of class to each quantized centroid.

During the learning phase, centroids are associated to elementary processors. At each presentation of an input pattern, the nearest centroid - in terms of Manhattan distance - is moved towards or away from the input stimulus depending on the concordance or not of their classes. Because of the move of only one centroid at each learning step, only one elementary processor will be active while all others are turned off; the parallel computation power of this associative processor is thus spoiled and the performances drastically de-

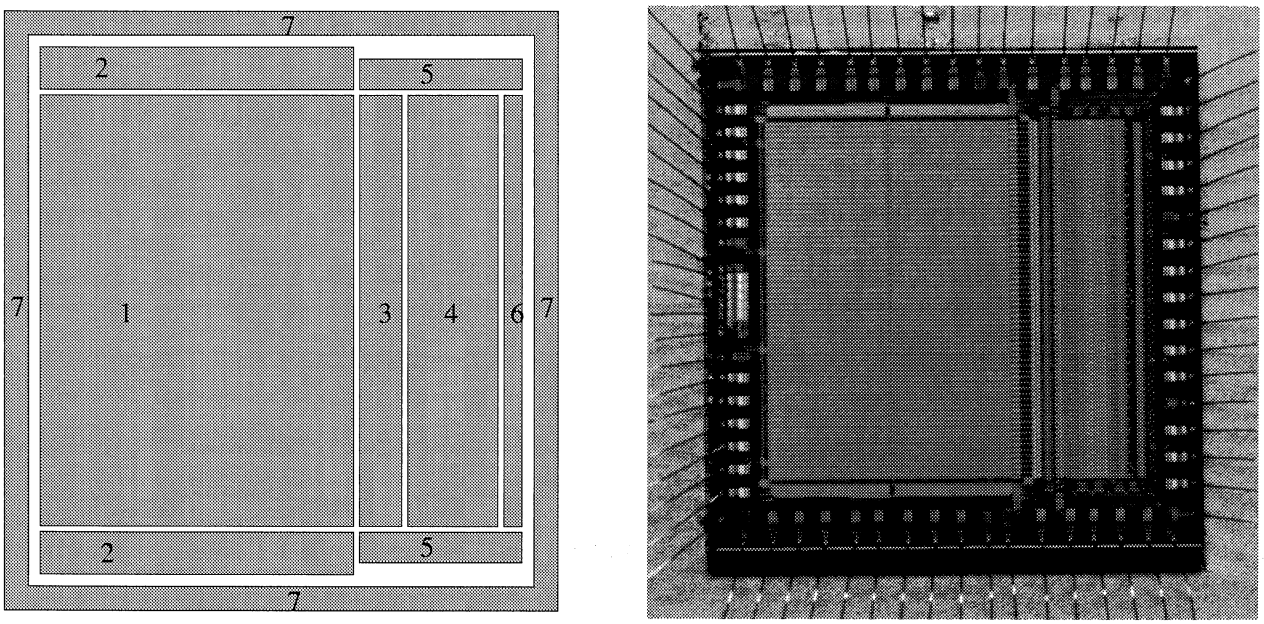

Figure 5. Schematic representation of the prototype circuit (left part). The different parts are: 1. the memory array; 2. the reading decoder (top) and the writing one (bottom); 3. the writing and reading RAM circuits and links between RAM and ALUs; 4. the ALUs; 5. command signals; 6. common functions; 7. pads, power supply and guard rings. Micro photograph of the circuit (right part).

crease for this specific algorithm. Table 2 illustrates the number of learning vectors (presentation of an input pattern and adaptation of the network) that can be achieved per second, and the percentage of time spent in distance computation.

For the classification phase (once the representation of the input space has been built), the power of this parallel architecture is much more exploited and the results obtained are very impressive as shown in table 2. The classification algorithm consists in computing all distances between the input pattern and the stored centroids, and choosing the smallest of these distances. The class of the input vector is then selected as the one of the nearest centroid.

5. Conclusion

We described in this paper a parallel SIMD architecture dedicated to classification and pattern recognition applications. The SIMD machine is based on a bit-serial, word-parallel associative processor. One bit-column of each word is processed simultaneously and only one ALU is implemented by word. This machine is optimized for neural net time consuming computations and many algorithms can be processed by writing their associated microprogram.

Such a processor, operating at a frequency of 100 MHz and composed of 1024 PEs, is able to process up to 10 billions of 8 bits additions per second and to class 400 000 8-bits vectors with neural algorithms like RCE or LVQ.

6. Acknowledgments

Philippe Thissen is working towards the Ph.D. degree in microelectronics under a FRIA (Fonds pour la Formation à la Recherche dans l'Industrie et dans l'Agriculture) fellowship. Michel Verleysen is a Research Fellow of Belgian FNRS (Fonds National pour la Recherche Scientifique).

References

[1] T. Kohonen. Improved versions of the learning vector quantization. In *Proceedings of the International Joint Conference on Neural Networks, San Diego*, volume I, pages 545–550, June 1990.

[2] T. Kohonen. Statistical pattern recognition revisited. In *Advanced Neural Computer*, pages 137–144, 1990.

[3] B. Parhami. Associative memories and processors: an overview and selected bibliography. *Proceedings of the IEEE*, 61(6):722–730, 1973.

	Learning				Classification			
RCE	*Simulations*		*Theory*		*Simulations*		*Theory*	
	Distance	*Total*	*Distance*	*Total*	*Distance*	*Total*	*Distance*	*Total*
5 bits	51 %	1 000 000	41 %	800 000	65 %	1 278 000	43 %	1 200 000
16 bits	58 %	450 000	43 %	332 000	69 %	536 000	63 %	490 000
LVQ								
5 bits	18 %	358 000	16 %	308 000	58 %	1 142 000	46 %	909 000
16 bits	10 %	80 000	6 %	43 400	51 %	398 000	30 %	238 000

Table 2. Number (theoretical and simulated) of vectors learned or classified per second with the RCE algorithm (top table) and with the LVQ algorithm (bottom table).

[4] D. Reilly, L. Cooper, and C. Elbaum. A neural model for category learning. *Biological Cybernetics*, 45(1):35–41, 1982.

[5] I. Scherson, D. Kramer, and B. Alleyne. Bit-parallel arithmetic in massively-parallel associative processor. *IEEE Transactions on Computers*, 41(10):1201–1210, 1992.

[6] S. Yau and H. Fung. Associative processor architecture—a survey. *Computing Surveys*, 9(1):3–27, 1977.

[7] J. Yuan. High speed circuit techniques for pipelining and for one-clock-cycle decision, 1994. Lecture 3, EUROCHIP Advanced Course, no. 4.

[8] J. Yuan, I. Karlsson, and C. Svensson. A true single-phase-clock dynamic CMOS circuit technique. *IEEE Journal of Solid-State Circuits*, 22(5):899–901, October 1987.

[9] J. Yuan and C. Svensson. High-speed CMOS circuit technique. *IEEE Journal of Solid-State Circuits*, 24(1):62–70, February 1989.