VIP: An FPGA-Based Processor for Image Processing and Neural Networks

Jocelyn Cloutier, Eric Cosatto*, Steven Pigeon, François R. Boyer and Patrice Y. Simard*

Département d'Informatique et
de Recherche Opérationnelle
Université de Montréal
C.P. 6128, Succ. Centre-Ville
Montréal, H3C 3J7, Canada

*AT&T Bell Laboratories
101 Crawfords Corner Road
Holdmdel, NJ 07733, USA

E-mail: cloutier@iro.umontreal.ca

Abstract

We present in this paper the architecture and implementation of the Virtual Image Processor (VIP) which is an SIMD multiprocessor build with large FPGAs. The SIMD architecture, together with a 2D torus connection topology, is well suited for image processing, pattern recognition and neural network algorithms. The VIP board can be programmed on-line at the logic level, allowing optimal hardware dedication to any given algorithm.

1: Introduction

Even with the last decades exponential growth in performance of integrated circuits, many image processing and neural network applications still demand increased hardware speed. A first approach to increase performance is to build massively parallel computers. Their high price and difficulty to program have resulted in a very low acceptance rate. The design cycle of those computers is usually too long, and thus their technology is obsolete before they are commercially available. Consequently, users often prefer to use the latest high performance general workstation that is much less expensive and more easy to program. A second approach to solve the performance problem is to design dedicated parallel hardware for one task (or set of similar tasks). Their programmation is usually simple (or even nonexistent) while their performance/cost ratio is high. However, they are not flexible and their design cycle is long.

Over the last few years, advances in *programmable logic devices* have resulted in the commercialization of *field programmable gate arrays* (FPGA) which allow to put large numbers of programmable logic elements on a single chip. The size and speed of those circuits improve at the same rate as microprocessors' size and speed, since they rely on the same technology. In section 2, we propose an architectural framework for the *virtual image processor* (VIP) which is a parallel processor having large FPGAs as main components. In section 3, we present the first prototype of the VIP board that uses 5 large FPGAs, has 1.5 MB of static RAM and communicates through a fast PCI bus.

We are currently targeting at applications requiring a large number of simple low precision operations. Many commercially attractive applications fall into this category such as image processing and pattern recognition (e.g., recognition of fax documents, bank checks, postal addresses). Those applications are particularly well suited for FPGA implementation since a simple *processing element* (PE) may perform their most basic operations. Consequently, many instances of this PE may be fitted on one FPGA. We present in section 4 two algorithms that fall into this category. We compare the performance of the VIP board with those achieved by dedicated hardware and general processors.

2: Architecture

One of the most efficient and cost-effective architecture for parallel vector and matrix processing is the 2D systolic architecture [4, 9, 7, 8]. However, this architecture is somewhat restrictive for more general applications. We have thus preferred an SIMD (Single-Instruction Multiple-Data) architecture together with a 2D torus connection topology, which include the 2D systolic architecture.

1086-1947/96 $5.00 © 1996 IEEE
Proceedings of MicroNeuro '96

Such architecture implementation permits to compute efficiently the following basic matrix and vector operations [4]: 1D and 2D convolutions, matrix multiplication, matrix addition, matrix-vector multiplication, scalar multiplication, vector addition, etc. However, it is implicitly less reconfigurable since it has a rigid data flow [7]. We believe that this rigidity is overcome by FPGA configurability.

The SIMD architecture has a 2D torus interconnection topology of its processing elements (PE). Each PE has a local memory. As depicted in Figure 1, the VIP architecture is composed of three basic components: the SIMD controller, the processing matrix and the I/O controller. Those components are connected by a shared global bus and two control buses. The processing matrix is a set of identical PEs interconnected in a 2D grid topology (Figure 1). The I/O controller manages off-board communication and initiates memory transfers. The SIMD controller decodes and executes the program stored in its instruction memory, and read or write to its data memory. There are three distinct types of memory: instruction memory, global data memory and PE's local memory.

For SIMD architecture, the same address and control signals are used by every PE (Figure 2). This architecture has many advantages. The simple PE interconnection topology is cheap (only 4 connections per PE), and very efficient for processing 2D data structures such as images. The complexity of the address buses is reduced since the same address is used by every PE. Many useful vector processing algorithms still perform optimally with such constraint.

2.1: SIMD Controler

The SIMD controller is the control unit of the VIP. It reads a program from its instruction memory and uses its data memory for storing global informations. Once an instruction is decoded, data and control signals are sent to the PEs through the global bus and a dedicated control bus. The global bus may be used to send both control and data signals. The SIMD controller also provides addresses and control to every memory during both program execution and I/O memory transfers. If configured accordingly, it exchanges status informations with the I/O controller.

2.2: Processing Matrix

The PE matrix is organized as a 2D grid. Each PE is connected in direction N, S, E and W to its 4 neighbors (Figure 1). Each PE has a local memory addressed by the SIMD controller.

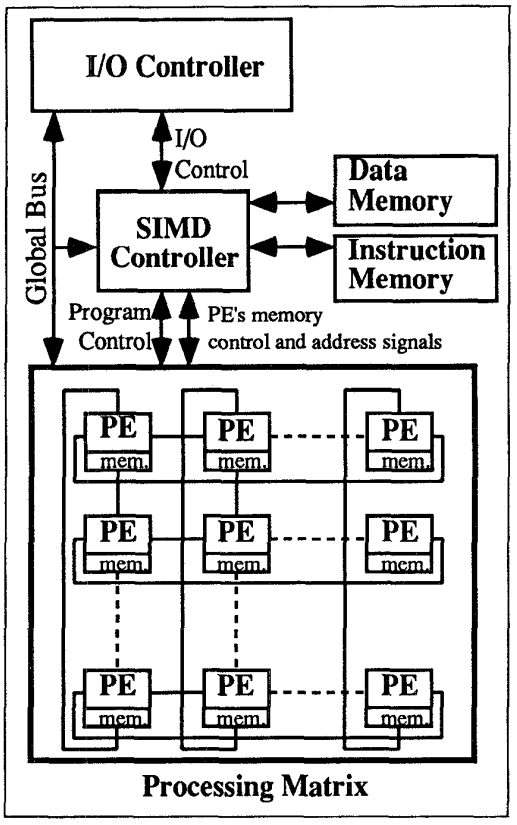

Figure 1. VIP architecture overview

2.3: I/O Controler

The I/O controller is responsible for the following operations:

- Communicating with the host.
- Exchanging status information with the SIMD controller.
- Managing data transfer between the host and the board.

Data transfers between the host and the board use the global bus to send address and data to PEs and SIMD controller.

3: VIP Implementation

We present in this section the implementation of the VIP printed circuit board based on the architecture presented in the previous section. A photography of the board is presented in Figure 6. The connection

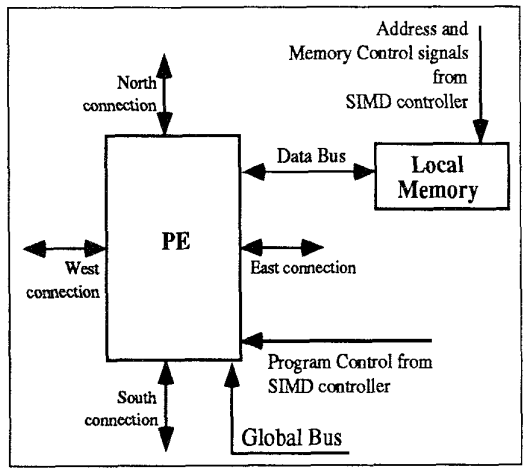

Figure 2. Detailed view of a Processing Element

to the host computer is done through a PCI bus interface. The board has a 2×2 FPGA matrix and each of the 6 memory blocks is a 20 ns static-RAM module of 64K×32 bits. The processing matrix and SIMD controller are each implemented by an SRAM-based Altera EPF81500 [1] FPGA. Each one of those FPGA has approximately 16,000 usable gates. The I/O controller is implemented by an EPROM-based Altera EPM7192 EPLD [2] (3,750 usable gates), and an AMCC S5933 PCI controller [3] with its configuration EPROM.

Using SRAM-based FPGAs for implementing the PEs has major impact on the overall system performance. Usually such architecture implementation is very expensive for general purpose processing or very restricted for dedicated computing. Since each PE may be configured on-board, we may perform any dedicated function by using exactly the logic needed for its implementation. For low precision processing, it has for effect to increase the number of processing element and thus increases performance. Also, such architecture is order of magnitude faster than general purpose processors having the same cost.

This follows characteristics of low precision computation that are not handle efficiently by those processors (e.g., manipulation of 1 bit data on a 32 bit architecture).

The 128 bit vertical torus connections (two 32 bit North connections and two 32 bit South connections) are routed to a 128 pin connector header. The North-South torus connection may be established by using jumpers. This connector is also useful to provide those signals off-board for multiple-board processing.

Bus	Number of bits
Global shared bus	32 bits
Three address buses	18 bits each
Control bus between I/O and SIMD controler	23 bits
Control bus between SIMD controler and processing matrix	10 bits
2D grid connections	32 bits
Configuration signals	8 bits

Table 1. Width of each bus.

3.1: SIMD Controler

The SIMD controller is implemented by an FPGA. This implies that decoding and executing instructions may be different from one application to the other. In the convolution application presented in section 4, both data and instruction memories are used to store large instruction words, while for the character recognizer, global informations are stored in data memory. Once an instruction is decoded, data and control signals are sent to the FPGAs through the global bus and a dedicated control bus.

3.2: Processing Matrix

The processing matrix is implemented by a 2×2 FPGA matrix. Each FPGA is connected to its North, South, East and West neighbors and to a local memory as it is the case for a PE (Figure 2). Each North, South, East and West connection have 32 bits (Table 1). Conceptually, an FPGA represents a sub-matrix of the global PE matrix. The data and control signals of an FPGA are shared among every PEs of its sub-matrix.

For example, in a 2D SIMD computation, each FPGA is configured as a 2D PE matrix. The FPGA matrix then becomes a single large PE matrix. The connections between FPGAs are used for connecting adjacent PEs located in different FPGAs (Figure 3). Since a 1D topology may be mapped on a 2D topology, the board may be configured as a 1D SIMD processor. Obviously, any architecture that fit into this framework may be implemented by configuring the FPGAs.

3.3: I/O Controler

We have chosen the PCI local bus standard for the I/O interface. This selection is motivated by its widely acceptance by the PC industry and for its high transfer rate (132 MBytes/sec). The design of the bus interface

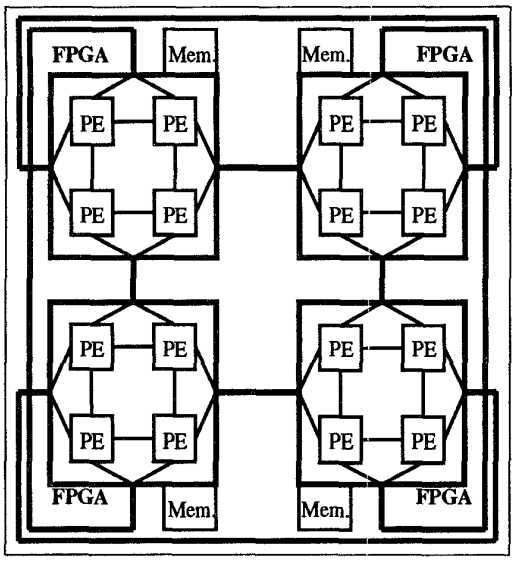

Figure 3. Mapping of a 4×4 PE matrix over the FPGA matrix

is greatly simplified by the AMCC S5933 [3] PCI controller. It is a powerful and flexible controller supporting several levels of interface sophistication. The control of the PCI interface chip is provided by an Altera EPM7192 EPROM-based programmable logic device. A 32-bit bus (multiplexed address and data) interconnects those two chips. This bus is connected to the board's global bus through a 32-bit transceiver. This transceiver has three modes: write to global bus, read from global bus and disconnect from global bus.

3.4: FPGA Configuration

Each Altera EPF81500 is on-line configurable. The reserved configuration pins for all 5 of those devices (SIMD controller and four FPGAs) are connected to the EPM7192 device (Table 1). It is thus possible to configure each FPGA at any time by providing the configuration data from the host. All FPGAs may be configured in parallel in less than 40 ms. The on-board memories are accessible only by the SIMD controller and the FPGAs. Therefore, they must be configured before any memory transfer is initiated.

3.5: Clocks

Two clock signals are available on-board. The first one is the 33 MHz PCI bus clock that is provided by the PCI controller while the second one is an on-board crystal clock. A programmable clock is generated from the EPM7192, based on those two clocks. For added flexibility, the crystal clock is mounted on a socket.

3.6: Multiple-board processing

It is possible to connect together two or more VIP boards by using their 128-pin connectors. In that case, the North and South connections of the processing matrix are routed to other boards. Some of those connections may be used as control signals to synchronize the execution on a multi-board system. This is greatly simplified by the use of the same PCI clock on each board.

3.7: The big picture

We present in this section the generic steps that are followed to perform computation on the VIP board.

1. Initially, the designer determines a program for an application and a dedicated logic design for the SIMD controller and each FPGAs.
2. This design is translated into a form that may be used to configure the corresponding FPGAs. This configuration is send to the I/O controller which supervises the configuration of each FPGA.
3. At this point, the SIMD controller and FPGAs have the capability to access each memory bank. Data transfers may be initiated by the host.
4. The I/O controller signals the SIMD controller to start program execution.
5. During program execution, status informations are transmitted to the I/O controller. SIMD processing is done in parallel to those transmissions.
6. The SIMD controller indicates to the I/O controller that the processing is done.
7. The result of the computation may be read from registers or memories as initiated by the host.

4: Application of the VIP board

We present in this section two representative applications implemented on the VIP board, comparing the achieved speed performance with those obtained by other implementations.

4.1: Convolution

We present in this section the implementation of a convolution over a binary image. The convolution process is the following. Each pixel $z_{i,j}$ in the resulting feature map is expressed as:

$$z_{i,j} = \begin{cases} 1 \text{ if } y_{i,j} > t \\ 0 \text{ otherwise} \end{cases}$$

where

$$y_{i,j} = \sum_{k=1}^{N} \sum_{n=1}^{M} f(x_{i+k,j+n}, w_{k,n})$$

and $f()$ is any 3-input Boolean function, $x_{i,j}$ is a pixel in the original image (precision: 1 bit), $w_{k,n}$ is a template pixel (precision: 2 bits), N and M are respectively the height and width of the template image. Finally, $y_{i,j}$ is the cross correlation between image field and the template and t is threshold level for a match. The interested reader is refered to [6] for more details. It is reported in this article that convolution with templates as large as 16×16 are useful for many different tasks in pattern recognition preprocessing (e.g., noise removal, printed or handwritten text differentiation).

Implementation details

We have implemented a 2D systolic algorithm. A matrix of 8×4 PEs are assigned to each FPGA, for a total of 16×8 PEs considering all four FPGAs. The systolic algorithm computes 16×8 convolutions in parallel in $N \times M$ steps. This is done by partitioning the image in $(15+N) \times (7+M)$ pixel tiles. Each tile has a 16×8 convolution results. A tile image is shifted over the PE matrix by using the North, South and East connections. The "border" of the sliding-window is feed from the PEs' local memory. The template values are broadcasted from the global bus to each PE. At any time, every PE computes a partial sum for its convolution. Reading pixel columns and rows from memory is pipelined with the processing, and an initial latency of 3 cycle is required.

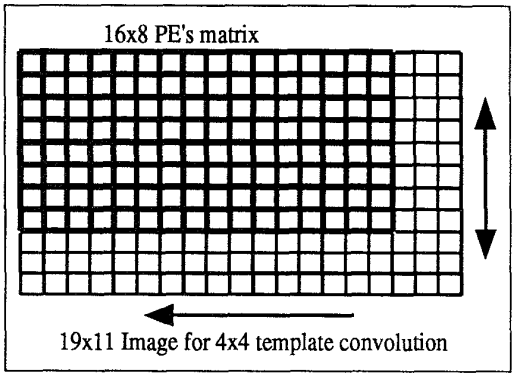

Figure 4. Tile processing for 4×4 template

Processing time

The processing steps for each tile convolution are:

1. Load a tile (1 cycle memory read, 3 cycle latency);
2. $N \times M$ computation (1 cycle each);
3. Threshold function evaluation with output $z()$ (1 cycle);
4. Write result (2 cycles memory write).

The number of cycles is thus $N \times (M+7)$ cycles per tile. The initial VIP prototype has a processing rate of 16.6 MHz for this application.

Performance comparison

This implementation speed performances are compared to those of the NET32K board [11]. We have also optimized the same algorithm for a 90 MHz Pentium computer having 32 MB of main memory and 256 KB level-2 cache. Table 2 shows a comparison between those three implementations. The Net32K board is configured to process only templates of size 16×16, this explain why no performance improvement is achieved for smaller templates. The reported performance shows that the VIP board is between 24 and 76 times faster than a 90 MHz Pentium depending on template size. Also, we show that for small template it is faster than the NET32K board, however for larger template it is 6 times slower. Considering that slow FPGAs and SRAM are used in our current implementation, those results are very impressive. The VIP board outperforms in some cases a dedicated processor. This is not a trivial achievement for programmable hardware which is much slower than custom chip implementation. Furthermore, the added flexibility of the VIP over the NET32K board permits to implement algorithms that could not be processed by this board. For example, the VIP board could process image with many level of gray which is not possible for the Net32K board.

Template size	VIP	NET32K	Pentium
4×4	1.5 ms	6.25 ms	114 ms
8×8	6 ms	6.25 ms	238 ms
16×16	24 ms	6.25 ms	961 ms

Table 2. Performance comparison between different processors for convolution (time for a convolution over a 512×512 pixel image)

4.2: Handwritten character recognition

We present in this section the implementation of an optical handwritten character recognizer. The program is based on the evaluation of a feedforward neural network. We are using a forward propagation without multiplication [5, 12]. The forward propagation for each unit i, is given by the equation:

$$x_j = f_j(\sum_i w_{ji} x_i)$$

where f is the unit function, w_{ji} is the weight from unit i to unit j, and x_i is the activation of unit i. For our implementation, we consider the following resolution and format of neural network variables. The weights are 8 bit fixed point numbers with 6 bit fractional value. The states are 3 bit floating point values with 1 bit mantissa (m), and 2 bits exponent (e). The corresponding real numbers are $-1^m \times 2^{-e}$. Each neuron has the same unit function which is a discretized version of a sigmoid function. The output function is a state value as defined above.

Let define the *shift-operation* $a \ll b$ where a is a fixed point number with same resolution as weights and $b = (m, e)$ is a floating point number. We have

$$(a \times b) = (a \ll b) = -1^m a 2^{-e}$$

which is implemented by a barrel shifter. The sign inversion is performed by the adder used for the *multiply-accumulate* operation. This implementation permits to reduce the hardware by about 40% over one using an 8 bit multiplier. This is done without any degradation of recognizer performance.

Neural network architecture

Figure 5 shows the neuron states in the network after processing the pattern "A". The first hidden layer (h1) performs a convolution with kernel 5×5 over the 32 × 32 pixel input (3 bit per pixel), and generates 4 feature maps. The second layer (H2) makes a subsampling of those 4 feature and reduces the resolution by four. The third layer (H3) performs a set of convolution (similarly to first layer) by combining different feature maps and results in 12 new maps. Layer (H4) perform the same operation as layer (H2) but on smaller feature maps. Finally layer (H5) is a fully connected neural network connected with 73 outputs corresponding to letters, digits and punctuation. All weighs in the network were learned from examples using the backpropagation algorithm.

Implementation details

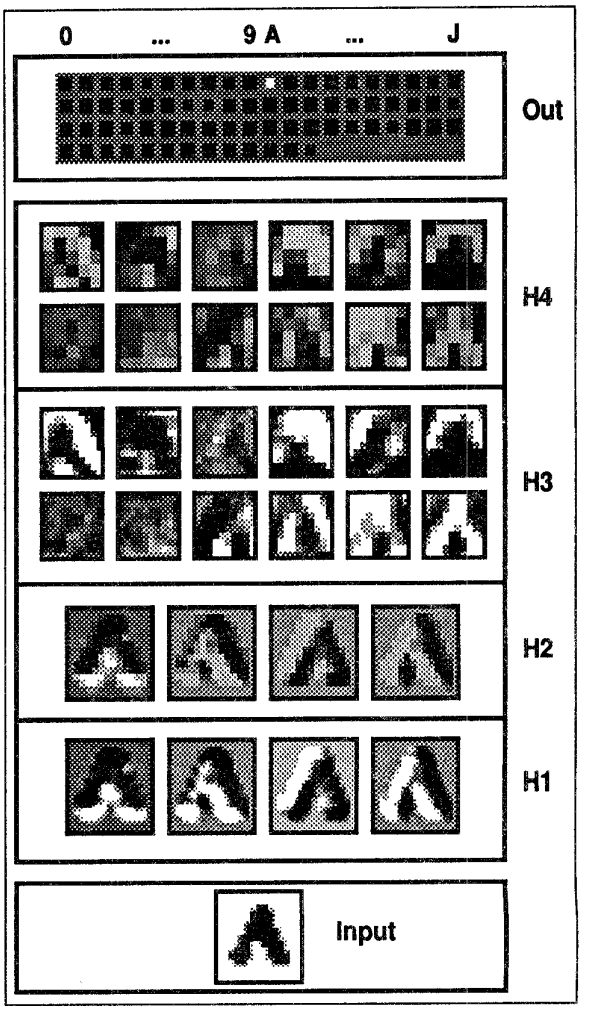

Figure 5. Architecture of the neural network for handwritten character recognition

The forward pass algorithm was implemented with a 2D systolic network of 12 × 12 PEs (6 × 6 PEs per FPGA). Only a square sub-matrix of 8×8 is able to perform a shift-accumulate operation and a discretized sigmoid function. The other PEs are used only to shift informations around. It is able to perform at a frequency of 16.6 MHz. The processing is time-multiplexed over the layers of the network. Each feature map is diced into *tiles* that fits into the 12 × 12 PEs of the 2D systolic network. For example, the 32 × 32 input is diced into 16 tiles (9 of size 12 × 12, 6 of size 12×8 and 1 of size 8 × 8). It is important that the resulting feature map for a tile is not larger than 8 × 8 since only 8 × 8 convolutions may be computed per tile.

Performance comparison

Only the first three layers were executed on the VIP board (86% of the connections). The remaining layers require higher precision and were computed on the host. The total execution time for the three layers are presented in Table 3. We compare the obtained performance our implementations to those of the ANNA board [10, 11] and a dual-processor SPARC 10 workstation [11]. Loading and storing transfer operations were included in the execution time of the VIP board. The VIP board is respectively 1.36 and 16 times faster than the ANNA and SPARC implementations. This is due mainly to the improved parallelism and 2D systolic network that is well suited for processing 2D data structure. For example, at each step of the convolution, 144 state values of 3 bits are exchanged between PEs, giving an impressive bandwidth of 900 MBytes/sec.

ANNA board	16 MHz VIP	SPARC 10-41X2
1.02 ms	0.75 ms	12.3 ms

Table 3. Comparison of execution time for recognition of one character on different hardware

4.3: Conclusion

We have presented in this paper the architecture and implementation of the Virtual Image Processor having large FPGAs as main building blocks. This has for impact of having maximal flexibility for the processor logic design. The VIP has a SIMD "virtual" architecture with a 2D interconnection network that is well suited for implementing 2D systolic networks.

We used two applications to compare the speed performance of the VIP board with other dedicated and general hardware designs. The VIP board is order of magnitude faster than general processor implementation. Furthermore, it has speed performances similar to those obtained by dedicated hardware. Those results are excellent, considering our approach is much more general and much cheaper than dedicated hardware. Also, the initial prototype reported here uses the slowest version of the EPF81500 FPGA available from Altera. We are planning to use faster chips and faster memory in the second prototype of the VIP board. This should permit to increase performance by a factor from 2 to 3.

We anticipate that more and more coprocessors will use an approach similar to ours. The advantage of such approach are lower cost and fast execution time. The price to pay for the added flexibility is the time taken for designing an hardware implementation for each new algorithm. We conjecture that current advances in hardware/software codesign should reduce the time taken for such design.

Acknowledgments

The authors would like to give many thanks for the support and funding of the Adaptive System Research Department research team at AT&T Bell Laboratories in Holmdel, NJ. We express our gratitude particularly to Hans Peter Graf, Larry D. Jackel and John Denker. This work was also funded by NSERC (Canadian government) under grant OGPIN-007.

References

[1] Altera inc. *FLEX 8000: Programmable Logic Device Familly.* Altera inc., March 1995. Data Sheet, ver. 6.

[2] Altera inc. *MAX 7000: Programmable Logic Device Familly.* Altera inc., March 1995. Data Sheet, ver. 3.

[3] AMCC inc. *S5930-S5933 PCI Controlers.* AMCC inc., Spring 1995. Technical reference.

[4] J. Beichter, U. Ramacher, and H. Klar. Vlsi design of a neural signal processor. In *Silicon Architectures for Neural Nets*, pages 245–260. Elsevier Science Publishers B.V. (North-Holland), 1991.

[5] J. Cloutier and P. Simard. Hardware implementation of the backpropagation without multiplication. In *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pages 46–55, September 1994.

[6] E. Cosatto and H. Graf. Net32k high speed image understanding system. In *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pages 413–421, September 1994.

[7] P. Ienne. Architecures for neuro-computers: Review and performance evaluation. Technical Report 93/91, École polytechnique fédérale de Lausanne, January 1993.

[8] H. Kung and C. Leiserson. Systolic arrays (for vlsi). In I. S. Duff and G. Stewart, editors, *Sparse Matrix Proceedings.* Knoxville: Academic Press, 1979.

[9] D. Mammerstrom. Image processing and pattern recognition hardware. Tutorial at Neural Information Processing Systems conference, November 1994.

[10] E. Säckinger, B. E. Boser, J. Bromley, E. LeCun, and L. D. Jackel. Application of the anna neural network chip to high-speed character recognition. *IEEE Trans. on Neural Networks*, 3(3), May 1992.

[11] E. Säckinger and H. Graf. A system for high-speed pattern recognition and image analysis. In *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pages 364–371, September 1994.