A DATA-FLOW LINEAR ARRAY IMPLEMENTING NEURAL NETWORK ARCHITECTURES

M. Marchesi, G. Orlandi , F. Piazza, A. Uncini*

Dip. di Elettronica ed Automatica, Università di Ancona
Via Brecce Bianche - 60131 Ancona, Italy

* Dip. INFOCOM, Università di Roma "La Sapienza"
via Eudossiana, 18 - 00183 Roma, Italy

ABSTRACT

A linear digital data-flow architecture for VLSI implementation of neural networks is proposed. It is characterized by true local connections, full expandibility and reconfigurability. It is composed of a linear array of processing elements (PE), each simulating a single neuron or a set of neurons, depending on the granularity of the actual implementation. Every PE is connected only to its two nearest neighbours through three buses. The proposed architecture is proved able to run many different neural network models. Among them, the multi-layer perceptron with Back-Propagation learning algorithm and the Counter Propagation network. Processed data are pipelined across the architecture and the computational efficiency can reach up to 100% efficiency in favourable cases.

INTRODUCTION

The main approaches to electronic hardware implementation of neural networks are analog, mixed analog-digital and fully digital VLSI circuits. Limiting our attention to digital architectures, it is known that their main implementation problems are related to the network connectivity. Present VLSI technology imposes heavy restrictions on the number of interconnections between pairs of neurons and requires local communications. Moreover, if the intercommunications among neurons are hardwired in the circuit, this is no longer reconfigurable to address different problems. Fortunately, VLSI devices operate very fast in comparison with biological neurons, a fact that allows to trade the difficulty in doing wiring against the speed of electronic devices.
In this work a linear digital data-flow architecture for VLSI implementation of neural networks is proposed. The architecture is organized as a pipeline of identical processing elements (PE). Each processor is locally connected only to its two nearest neighbours and the connections with distant processors, required by the neural models, are obtained by properly moving the data concurrently across the line. The proposed architecture presents several advantages, which make it very flexible: it has only true local connections; it can be easily expanded by simply adjoining more processors to it; it can be programmed according to the neural network model and to the number of neurons attached to each PE. In favourable cases, it has the merit of full pipelining (up to 100% efficiency).
In the next sections we will present the architecture and we will outline how various kind of neural network models can efficiently run on it.

THE LINEAR ARCHITECTURE

The proposed architecture for a digital implementation neural networks is shown in Fig. 1. It is composed of a set of elementary processors (PE), and it has the simplest form for such a set: a linear sequence where every processor communicates only with its two nearest neighbours. Given the total number of PE's, the architecture can be configured to implement a particular neural model, as it will be shown in next section. Each PE can be assigned one or many network neurons, depending on the available number of PE's, on their available data and program memory and on the model to implement. The linear array is a SIMD (Single-Instruction-stream, Multiple-Data-stream) machine, since every PE performs the same operations, but on different data. Moreover, it is implemented as a data-flow (wavefront) architecture to maximize efficiency and to eliminate synchronization problems for long structures.

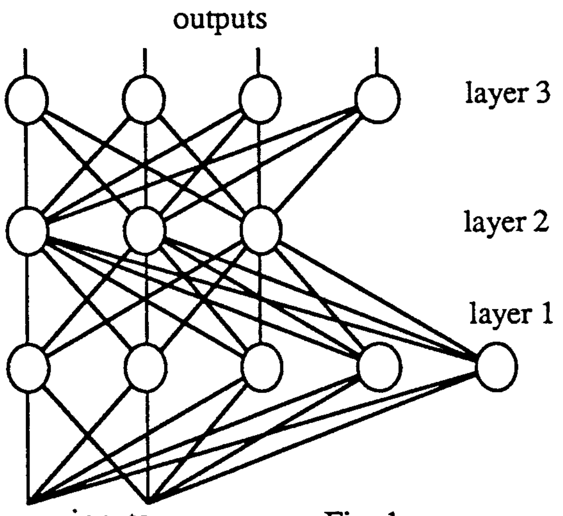


Fig. 1

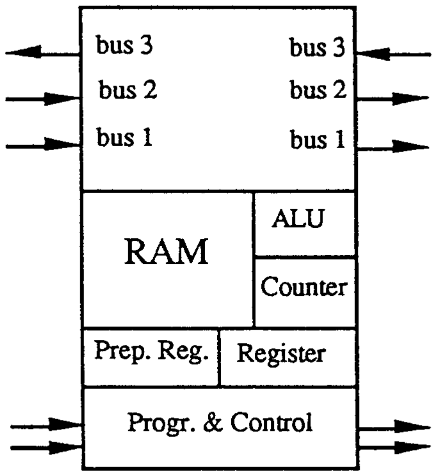
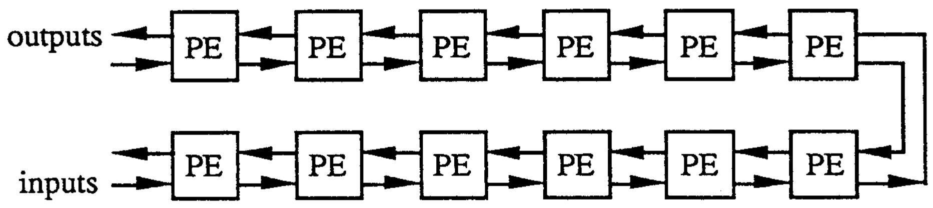


Fig. 2



A PE (Fig. 2) is provided of 3 input and 3 output unidirectional data buses, two directed from left to right and one in the opposite direction; each bus has a validation signal that flags if the data carried on the bus is valid. Every processor has also some activation and control lines which are used to signal its ready or busy state and which let it to be activated on the left side (forward mode) or on the right side (backward mode). A reset line is also present to reset the PE in a known state.
A PE is also provided of a RAM memory holding the weights of the neurons and other information needed to the neural model, including the look-up tables used to compute functions needed by the model, such as the activation function. The size of the memory limits the number of neurons attached to each PE and consequently th maximum size of the network. Some registers, a few pre-programmed registers used to configure the network, programmable counters which make the neuron able to perform the correct sequence of operations, as well as a control unit and an arithmetic and logic unit able to perform add and multiply operations are also contained in each PE.
The input of the network are fed one after the other in pipelined way into the left side of its first PE and propagate forwards from left to right. The outputs get out from the right side of the last PE of the network. It is also possible to propagate data backwards from right to left.

RUNNING DIFFERENT NEURAL MODELS ON THE ARCHITECTURE

The original neural model the linear array has been studied to implement is the multi-layer perceptron (MLP) with back-propagation (BP) learning algorithm. If each PE is assigned to one neuron, the neurons belonging to various layer are contiguous and disposed, from left to right starting form the first PE, according to the order of the layer (Fig. 3a). Initially, the number and the width of the layers are programmed by setting some registers of the PE's, and the starting interconnection weights are stored in each PE. In the feedforward mode the input samples feed the array and serially move across the PE's of the first layer, where each neuron takes its input, multiplies it with the corresponding weight, accumulates the result in the a register, passes the sample to its right side through the first bus and activates the next neuron of the array. After receiving all inputs, the first PE computes its output and propagate it concurrently with the last input through the second bus. When the first output arrives to the end of the first layer, it should be fed into the second layer in the same way the inputs were fed into the first layer. A new wavefront can be pipelined in the snake after the first, with some limitations. In fact, in the hypothesis that all the layer have the same length, the input data need not to be separated and a full pipelining is achieved, with all neurons computing at every step. If layer lengths differ, the length of a single input sequence must be at least equal to the longest layer of the net. If the longest layer is not the input one, enough no-op's (input data with valid1 off) have to be inserted between input sequences to satisfy this constraint. In the most favourable case, a 100% pipelining efficiency can be achieved.

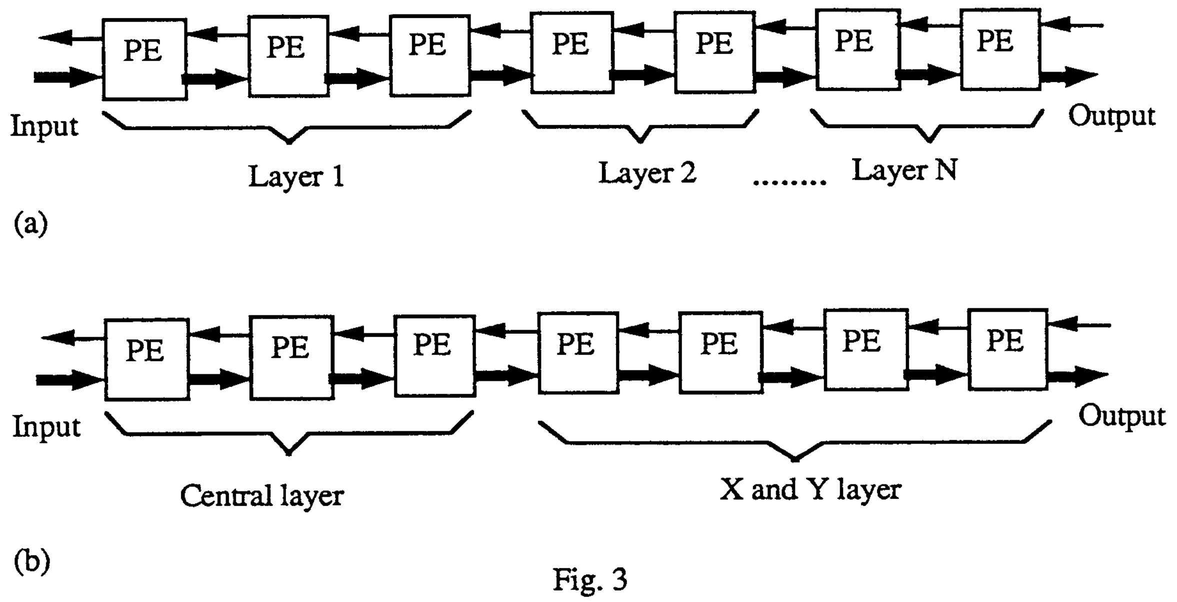


Fig. 3

The backward phase is divided in a forward and a backward move. The forward move is carried out by letting the array to operate in forward mode, without pipelining anymore input patterns. The backward move starts when the structure has computed all the outputs of the last layer. However, the differeces between computed and expected outputs are externally computed and fed into the array using the backward bus, and the atructure is activated by its right side. These values propagate through the last layer where the proper quantities of BP algorithm [1] are computed. At the end of the last layer (left side), they are fed into the penultimate layer and so-on. The adaptation of the weights of the neurons of a layer is performed concurrently with the computation of the quantities of the preceding layer. A more detailed description of BP on the array can be found in [2].
Another neural network model that can be implemented on the array is the counter-propagation network (CPN) [3]. In the CPN the neurons are divided in three layers: one central layer and two side layers. The central layer performs the task of classifying the inputs in M different classes, being composed by M neurons. The two side layers (X-layer and Y-layer) are composed of n and m neurons, respectively. Their purpose is to optimally reconstruct an input pair (x,y) to the

network.
Implementing CPN with a linear array of PE's is a non trivial problem since operations like the computation of the maximum output of central neurons and its notification to all neurons are inherently sequential and involve passing data forward and backward the array. Moreover, the weight updating in central layer is performed by only one neuron for every input, blocking the whole net until it is accomplished. For these reasons, one may expect that pipelining efficiency as high as with the BP model cannot be achieved. The proposed implementation, however, takes full advantage of the three buses and exploits as far as possible parallel execution of operations. It is shown in Fig. 3b; its inputs are first elaborated by the central layer, whose neurons are thus assigned to the first PE's of the array. The subsequent PE's are assigned two side neurons each, thus enabling receiving and passing data in couples through buses. X and Y data pass in couples through buses also in the central layer.
During CP, the components of input vectors X and Y are fed in couples into the first PE and then propagated forwards through the whole network. Each neuron computes the weight sum of its inputs and stores it in a register. When the last input is fed to any neuron of central layer, it ends the computation of its weighted sum and the computation of the maximum weighted sum can be accomplished very naturally passing forward through the array the value of the highest weighted sum of the preceeding neurons in one bus and the index of the corresponding neuron on the second bus. This pair of data immediately follows the last pair of inputs and can be compared by every neuron with its freshly computed weighted sum. The higher between these two sums is passed forward, together with the proper neuron index. After this phase, every central neuron starts the updating of its weights as it were the winner, but holding in memory the old weight values. This updating can be accomplished because a second copy of the input pairs (that is needed for the updating is sent forward through the array after the first. When the highest weighted sum of inputs arrives to the last neuron of central layer the maximum match is computed for all central neurons, and this information is then forwarded both forward to X and Y layers and backward to the neurons of the central layer, to update their weights. The neuron of side layers, that have already received and stored the proper input values, can thus compute their outputs, adjust their weights and then properly send forwards their outputs through the array up to the network output. Concurrently, when the winner neuron of central layer recognizes its index, it can end and validate its weight updating. The other neurons simply discard the updated weights.
From the proposed structure may also be derived the ring architecture presented in [4]. Moreover other well-known network models (Hopfield, Recurrent Networks, Kohonen) can easily be implemented on such structure.

REFERENCES

[1] D.E. Rumelhart, J.L. McClelland, and the PDP Research Group, "Parallel Distributed Processing (PDP): Exploration in the Microstructure of Cognition (Vol. 1)", MIT Press, Cambridge, Massachusetts, 1986.

[2] F. Piazza, M. Marchesi and G. Orlandi, "A Digital 'Snake' Implementation of the Back-Propagation Neural Network", Proc. IEEE ISCAS '89, Portland, OR, May, 1989, pp. 2185-2188.

[3] R. Hecht-Nielsen, "Counterpropagation Networks", Proc. ICNN 87 Inter. Conf. on Neural Networks, San Diego (CA), Vol. II, pp.19-32, 1987.

[4] F. Piazza, M. Marchesi, G. Orlandi and A. Uncini, "An Efficient Mapping of Neural Networks to a Linear Architecture of Coarse-Grained Processors", Proc. IEEE ISCAS '90, New Orleans, USA, May, 1990.

This work was supported in part by the Consiglio Nazionale delle Ricerche of Italy and in part by the Ministero della Pubblica Istruzione of Italy.