Hardware Neural Network with MC431 DSP Board

Cosmin Cernăzanu, Ştefan Holban

"Politehnica" University of Timişoara, Pța.Victoriei no.2, RO-300006 Timişoara
cosmin.cernazanu@ac.upt.ro, stefan@aspc.cs.utt.ro

Abstract: The paper is a description of the way in which one neural network was built on a Single-DSP PCI Board. The board was designed for software evaluation and system prototyping on NeuroMatrix® NM6403 DSP. In the first part of the paper, we will point out the importance of the hardware neural network in different lines of business and we will explain which these lines are. Further on, we will present a summary of the main features of the NM6403 board, followed by the implementation of a feedforward neural network. The network has one input layer with 20 neurons, one hidden layer with 512 neurons and an output layer with 20 neurons. We should mention that the neural network described in this paper was used for testing. The results achieved with the help of this network are encouraging and they open new possibilities of study for the future.

Keywords: hardware, artificial neural network, Single-DSP PCI board, NeuroMatrix

1 What is Neural Network?

An *artificial neural network* is a model that emulates the biologic neural network. A neural artificial network is made up of thousands of artificial neurons; elements of non-linear processing that operate in parallel. [1]

The main characteristics of the neural networks are the same with those of the human brain that is:

- capacity of learning

- capacity of generalizing

If trained adequately, the artificial neural networks would be capable of giving correct answers even for the set-entries different from those they have already been used to, as long as they don't differ too much. This generalization is made automatically as a result of their structure and not as a result of human intelligence which is included in a program as in the case of the expert systems.

2 Why Hardware Neural Network (HNN)?

In the early years of neural network research, it was assumed that implementation in special hardware would be the future of neural network development. Such hardware, in particular, would probably be analogue and involve multiple parallel processing elements.

However, in present, the majority of Artificial Neural Network (ANN) applications in commercial use are implemented in software, and run on a conventional single processor general purpose computer. This fact is mainly due to software's flexibility; particularly important in using comparatively new and unknown technology, where conditions may be somewhat experimental.

Meanwhile, the development of HNN has been slowed down and had only modest commercial success. However, specialized hardware (which can either support or replace the software) offers appreciable advantages in some situations. It is worth looking in greater depth at the features of HNN because of their potential to change market opportunities.

In what follows, we will try to point out the most common reasons for using HNNs are:

- Speed – is considered to be the main reason for using HNN because most of the applications can be speeded up by using the specialized hardware. If we have ANN with a large number of neurons, not even the fastest sequential processor will be able to provide a real-time response. Parallel processing with multiple simple processing elements can provide significant speedups.

- Cost – although this was considered to be one of the problems for decreasing HNN, in mass-production industry the situation is reverse. In that sector we have found the need for cheap dedicated devices, such as for speech recognition in consumer products.

- Reliability – it is known the fact that a hardware implementation can offer greater reliability in operation, in the sense of reduced probability of equipment failure.

- Special operating conditions – for those applications that require limited physical size or weight.

- Secrecy – better protection to reverse engineering.

Even with those characteristics, the HNN, in particular those with multiple parallel processing elements optimized to run neural network algorithms, are still struggling to find markets for commercial success.

3 MC431 DSP Board

MC431 is a Single-DSP PCI board designed for software evaluation and system prototyping on NeuroMatrix® NM6403 DSP. NeuroMatrix® NM6403 is a high performance dual core processor with combination of VLIW/SIMD architectures. The architecture includes two main units: 32-bit RISC Core and 64-bit VECTOR co-processor to support vector operations with elements of variable bit length. There are two identical programmable interfaces to work with any memory types as well as two communication ports hardware compatible with TI DSP TMS320C4x which permit to build multi- processor systems.

As you can see in Figure 1, the processor NM6403 has four main channels to transmit data to/from peripheral devices.

Global and local data buses are used to access external memory. Memory blocks connected to the global bus are referred to as global memory.

In addition to external memory, the processor can also send and receive data through two communication ports hardware compatible to TMS320C4x.

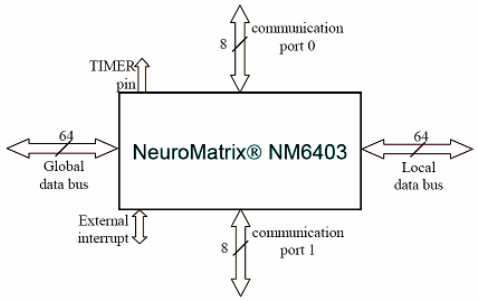

Figure 1
NM6403 External Processor Interface

Processor NM6403 consists of the following internal blocks, see Figure 2:

- Scalar RISC Core
- Vector execution unit
- DMA–coprocessors, managing communication ports
- Timers
- Global and Local Memory Interface Units managed by control registers to access different types of external memory.

The Vector execution unit, which is the main feature of NeuroMatrix NM6403, operates concurrently with scalar RISC-core and two DMA-coprocessors. This 64-

bit engine provides highly parallel operations, allowing simultaneous execution of up to 2048 operations in a single clock cycle. Its architecture gives flexibility of choice in performance/accuracy ratio. Depending on data size, vectors are from one to sixty-four elements long. [2]

The Vector Unit operations are performed on multiple data elements by a single instruction. This is often referred to as SIMD (single instruction, multiple data) parallel processing. Processor NM6403 has a 64-bit external memory interface. It allows access to one 64-bit word per transaction at each memory bus. Depending on memory access time up to two 64-bit words per cycle can be transmitted.

RISC-core is used for address calculations and for basic operations over general-purpose registers. It is also used to prepare data for the Vector Unit.

RISC-core contains a primary register file that contains eight address registers and eight general-purpose registers. In addition to the primary register file there are also a peripheral control register file and a vector register file.

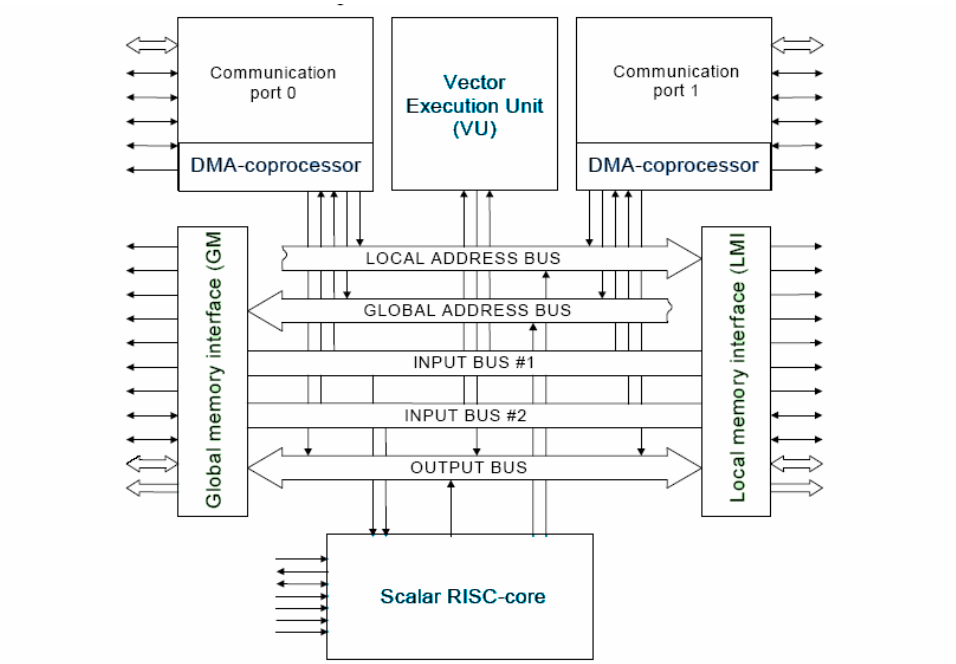

Figure 2
NM6403 Block Diagram

4 Implementation of Neural Network on the NeuroMatrix® NM6403 Processor

We have chosen a feed forward propagation network. The sample neural net has the following parameters:

- the net has got 20 inputs, one hidden layer of certain number of neurons (about hundreds) and the output layer of 12 neurons;
- input and output values of the neurons are represented as 64-bit signed integers;
- weight coefficients of the neurons are represented as 16-bit signed integers, packed in fours into 64-bit words;
- the output characteristic of a neuron (that is, the dependence of the output value from a weighted sum of the inputs) is defined as an integer function with a non-linear (sigmoid) real prototype, whose output values lay in range from -32767 to 32767. At the argument values less than -32768 or greater than 32767, the function takes the margin values, accordingly -32767 or 32767.

The neuron output is calculated as follows: the input values are multiplied by corresponding weight coefficients and these products are added together. [3] The obtained sum is used as an argument for certain function, called threshold function or activation function, to obtain the neuron's output value, see Figure 3.

As a rule, this function is chosen non-descending and bound in output value to a certain range. In order to achieve better trainability when using gradient methods, the function (or its real-value prototype in case of discrete values) is often chosen to be smooth and non-linear.

The neurons of the same layer of neural network share one vector of inputs. Tuning of weight coefficients called training changes the network output behavior.

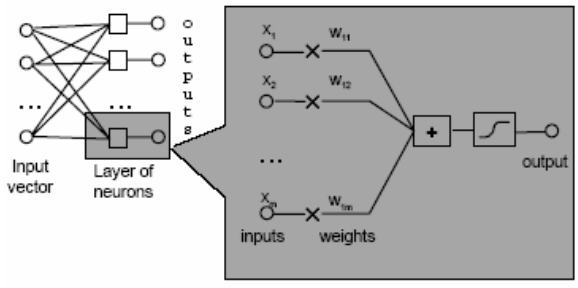

Figure 3
Neural Network layer

5 Calculation of Weighted Sums

The vector-processing unit of the NeuroMatrix®NM6403 processor is designed to execute a vector-by-matrix product, which is a common operation for matrix calculations. In our example, the configuration of the Active Matrix of NM6403 is one column and four rows filled with 64-bit words of input data: [4]

rep 4 wfifo = [ar4++], ftw;

...

wtw;

The corresponding chunks of weights matrix of four 16-bit values packed into 64-bit word are multiplied by these values with respect to sign. The four products are summed to a 64-bit value that is added to the already accumulated sum (Fig. 4).

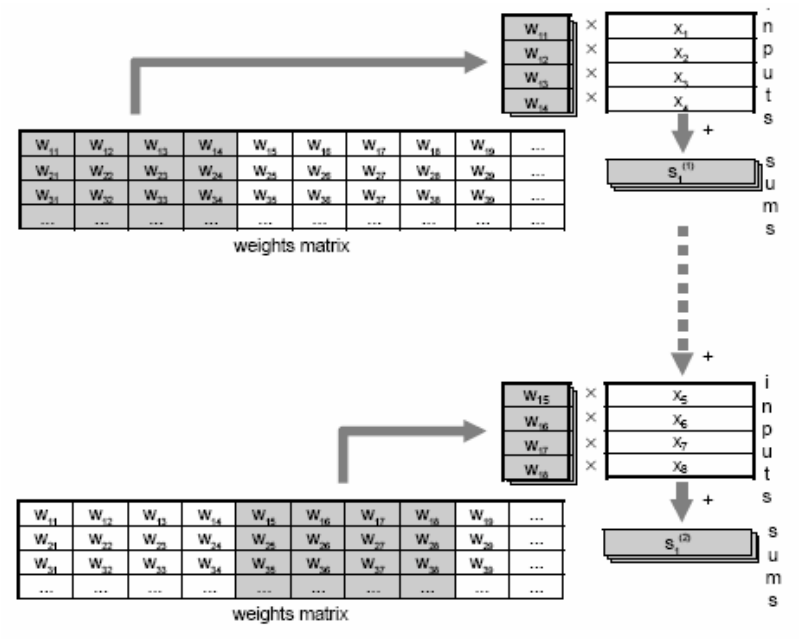

Figure 4
Scheme for weighted calculation

These four multiplications and the summation of five values are performed during one step of vector operation which, being multi-step, calculates several (up to 32) such partial sums for several neurons due to the fact that the input vector is the same for every neuron in the hidden layer:

rep 32 data = [ar0 += gr0] with vsum ,data, 0;

(for the first four input values),

rep 32 data = [ar0 += gr0] with vsum ,data, afifo;

(For the forthcoming input values, adding accumulated sums left by the previous such operation in the results queue).

In order to avoid overheads for saving and restoring of partial sums from memory, the sums of all the neurons processed by one such operation are added to the result of the next iteration and do not leave the vector device.

Thus, consequently changing fours of input values in the multiplier matrix, complete weighted sums are calculated for, say for certainty, 32 neurons.

During the next iteration, the sums of 32 more neurons are calculated, and so on until the neurons of the whole layer are processed.

6 Calculation of Threshold Function

One would note that the threshold function becomes nearly constant for the arguments with large magnitude. Therefore we can select a range of integer argument values and retrieve function values within this range from a table. If the arguments are out of the range then the function value return is equal with marginal function value.

The hardware saturation function that is implemented in NeuroMatrix® NM6403 is suitable to truncate values to the range:

rep 32 with activate afifo + 0;

The control register *f1cr* is initialized such that saturation function replaces the values that are less than -32768 or greater than 32767 with -32768 or 32767 accordingly, and leaves the values from -32768 to 32767 unchanged. After hardware saturation is applied we use table of 65536 numbers to retrieve function values corresponding to all the possible arguments. The table is filled with sigmoid function values converted to integers (1):

$$f = 32767 \cdot \frac{1 - e^t}{1 + e^t}, t = -\frac{x}{2048}, \quad (1)$$

where x runs the range from -32768 to 32767.

Conclusions

We must add that this network was build only for experimental used. It hasn't been made yet a statistics that can show the aspects concerning the performance of the network regarding the other forms of training.

We only measure the time of training for that network and the result for 512 hidden neurons was:

- 20480 number of element multiplications,

- 19140 cycles spent,

- 0.55 msec time of execution.

References

[1] Teodorean Gavril, "Rețele neuronale artificiale", Ed. Cluj-Napoca, 1995

[2] RC Module. NeuroMatrix®NM6403. Architectural overview

[3] Negnevitsky, M. (2002). "Artificial Intelligence: A Guide to Intelligent Systems", Addison Wesley, England

[4] RC Module. NeuroMatrix®NM6403. Assembly language overview