

# **A VLSI Digital Neural Processor with Variable Word Length of Operands**

*Submitted to:* "Priborostroenie". Hard-Software systems for  
neural calculation support, Thematic issue 7/96

*Authors:* Pavel Vixne, Dmitri Fomin, Vladimir Chernikov  
*Research Center "Module", Moscow, Russia*

## **Abstract**

The article describes the architecture and the instruction set of a single chip digital neuroprocessor with variable length operands. The originality of the processor lies in its ability to increase efficiency with the decrease of the operand length which permits to obtain optimum relation of precision/efficiency factors. This processor can be used for solutions of neural net tasks as well as any applications requiring hardware support of matrix/vector calculations.

### **1. Introduction**

### **2. Architecture and Instruction Set of the Neurochip**

### **3. The Architectural basis for Building Neurosystems Based on the Neurochip**

### **4. Comparative Characteristics of Architectures Based on Neurochip,**

**TMS320C40 and Intel P55C**

### **5. Conclusion**

### **6. References**

## **1. Introduction**

An analysis of the current state of affairs in the area of hardware support of neural nets [1], [2] has shown three directions of research - creation of analogue, digital and mixed neuroaccelerators. Basing ourselves on the real potential of local microelectronics, ease of use and the available ECAD, RC "Module" specialists have chosen a fully digital programmable neuroprocessor on a single chip.

### **Selection and Grounds for the Principles for the Neurochip Design**

The main pre-conditions chosen for the future neurochip were the following:

- neurochip must be programmable and have blocks specially designed for the effective implementation of neural net calculations;
- neurochip must provide calculations for the output values of as big an area of the neural net as possible during one clock cycle;
- neurochip must handle data (weights and synapses) of variable length (from 1 to 64 bits);
- neurochip must contain means for creation of complex distributed parallel computing systems containing any number of processor nodes;
- neurochip must have a sufficiently versatile instruction set capable of supporting the widely used constructions of modern high level programming languages. Any instruction (except load/unload data) must be a one clock cycle executable one.

Below we describe the architecture solutions which conform to the above requirements.

## 2. Architecture and Instruction Set of the Neurochip

The neurochip is designed to be implemented in neural nets with the following functional characteristics:

- a) Every layer of the neural net (fig.1) performs the following function:

$$Z_i = f(Y_i) = f\left(U_i + \sum_{j=1}^N X_j W_{ij}\right), (i = 1, \dots, M; j = 1, \dots, N),$$

where:

- $Z_i$  - output signal of  $i$ -neuron,
- $X_j$  -  $j$ -input signal of the layer ( $j$ -synapse),
- $U_i$  - offset of  $i$ -neuron,
- $W_{ij}$  - weight coefficient of  $j$ -input of  $i$ -neuron,
- $Y_i$  - sum of weighted inputs of  $i$ -neuron,
- $f$  - activation function,
- $N$  - number of input signals of the layer,
- $M$  - number of neurons in the layer.

- b) Operands  $Z_i$ ,  $X_j$ ,  $U_i$  and  $W_{ij}$  are represented in direct and supplementary codes and can have variable length.

- c) Number of layers in the neural net and the number of neurons and signals in each layer is random.

- d) Inputs and outputs of separate layers of the neural net can be connected in any way. I.e. output of  $k$ -layer can be connected to any input of  $i$ -layer where  $k=1, \dots, K$ ;  $i=1, \dots, K$ , and where  $K$  - number of layers of neural net.

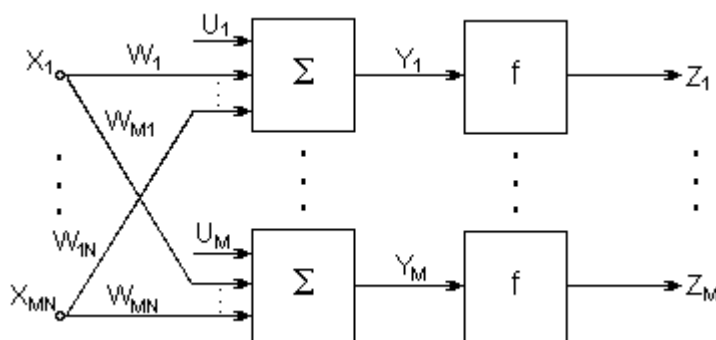
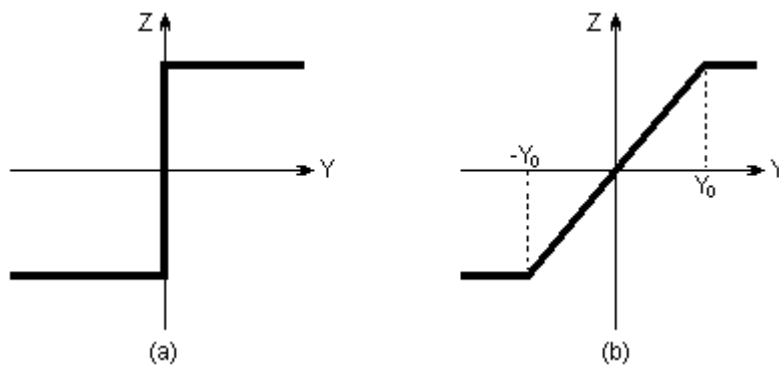


Fig.1 Neural net layer



The simplest activation functions "f":

- a) threshold, b) limit

Fig.2

The architecture of the neurochip is based on the original method of basic operations execution using operational unit (OU) which is a homogeneous calculating environment permitting to process synapses and weights of variable lengths (from 1 to 64 bits). We must note that the principle of variable length of operands is implemented in Philips L-neuro 1.0 [3], but it represents data in a serial code reduces efficiency and besides the lengths can be selected from fixed set of values only - 1, 2, 4, 8, 16. Activation function of L-neuro 1.0 is implemented outside the chip which reduces efficiency still further. L-neuro 1.0 efficiency is - 100 MCPS for one-bit operands and 26 MCPS for 8-bit.

### Weighing and Summing of Input Synapses

Weighing and summing of synapses is a complex operation requiring large hardware and time resources. That is why this operation is the main operation for the neurochip OU. Its structural organization permits weighing and summing of synapses by the method of parallel multiplication of synapses by weight coefficients.

In this case OU does in one cycle weighing of several synapses and calculation of their sums simultaneously for several neurons. I.e. during one clock cycle the OU implements a fragment of the neural net consisting of several synapses and several neurons.

The operation of OU is illustrated using an example with two neurons and three synapses (fig.3) using OU with  $m=19$  (summary length of outputs) and  $n=12$  (summary length of inputs). In this example the operands have the following lengths:  $X_1=3, X_2=4, X_3=5, U_1=10, U_2=9, Y_1=10, Y_2=9, W_{11}=5, W_{12}=4, W_{13}=3, W_{21}=4, W_{22}=3, W_{23}=2$ .

Preliminarily the OU is loaded with corresponding bit elements of weight matrix which is filled during training of the neural net. In this mode the matrix elements are the weight coefficient bits and null elements distributed in strictly determined positions. Such arrangement breaks the OU into  $P \times R$  submatrixes (fig.4), where  $P$  - number of neurons and  $R$  - number of synapses in the given fragment of the neural net. In our case  $P=2, R=3$ . Let's assume that the submatrix with coordinates  $(p, r)$  corresponds to  $p$ -neuron ( $p=1, \dots, P$ ) and  $r$ -synapse ( $r=1, \dots, R$ ). Then each submatrix given fragment of the neural net. In our case  $P=2, R=3$ . Let's assume that the submatrix with given fragment of the neural net. In our case  $P=2, R=3$ . Let's assume that the submatrix with coordinates  $(p, r)$  corresponds to  $p$ -neuron ( $p=1, \dots, P$ ) and  $r$ -synapse ( $r=1, \dots, R$ ). Then each submatrix with coordinates  $(p, l)$  calculates the function  $X_l W_{pl} + U_p$ , and each submatrix with coordinates  $(p, t)$  forms the product  $X_t W_{pt}$  and adds to it the result obtained at the output of submatrix with coordinates  $(p, t-1)$ , where  $t=2, \dots, R$ . Thus the output of submatrix  $(p, R)$  forms the result of the operation.

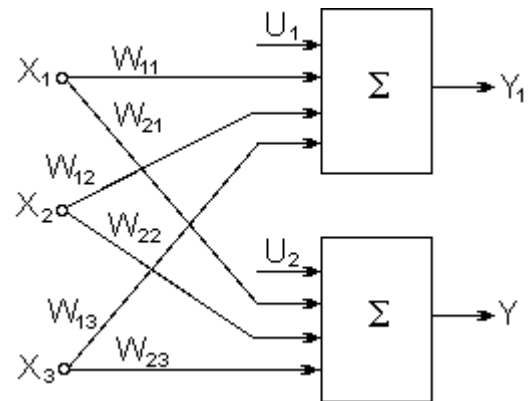


Fig.3 Layer fragments of neural net.

$$Y_p = U_p + \sum_{r=1}^R X_r \cdot W_{pr}$$

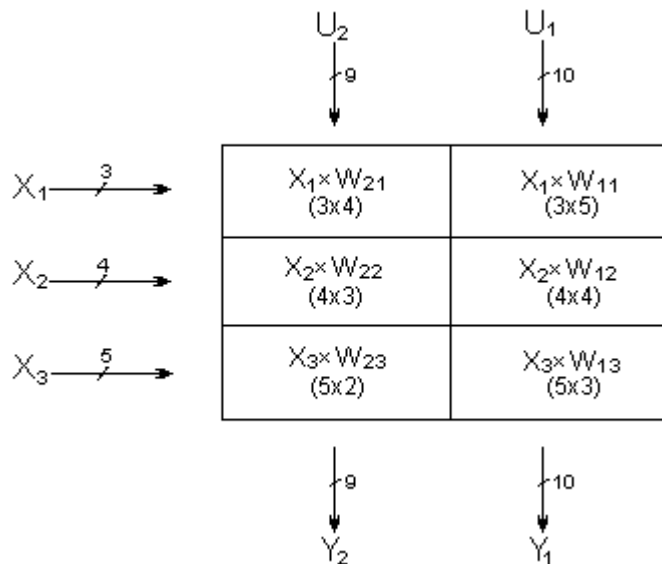


Fig.4 Dividing the matrix into sub-matrices

The necessary condition for OU operation here is the requirement that the number of bits used by each  $p$ -neuron must be less than maximum length of  $Yp$ . Only then there is no danger of shifting which happens in summing circuits across the borders of the adjacent neurons. On one side this requirement provides concatenation of several neurons in the calculating environment which is here the OU and on the other hand it eliminates the possibility of arithmetic overflow and thus an additional control. This requirement is satisfied during training of the neural net by introducing into matrix of weights some null rows in the area of upper bits of each neuron.

Thus the number of served neurons by OU depends on number of synapses, their length and weight coefficients values. Summary length of  $Yp$  must not exceed the value of  $m$ . Number of synapses processed by OU depends on their length. The summary length of  $X$  must not exceed the value of  $n$ . Tuning of the neurochip for working with specific value lengths of synapses and weights is done by loading special registers the control information before the start of work as well as during the calculations. This permits to dynamically change the the length of processed data and thus the speed of processing and its precision form layer to layer, from neuron to neuron and from synapse to synapse.

If we take synapses and weights of 8 bits then OU does 24 multiplications with carry in one clock cycle - that is 720 MCPS with 30 MHZ clock. In other words an area of neural net of 3 neurons with 8 synapses each is calculated where neuron and synapse is one byte wide.

## Neurochip structure

The general structure of the neurochip is given in fig.5 below.

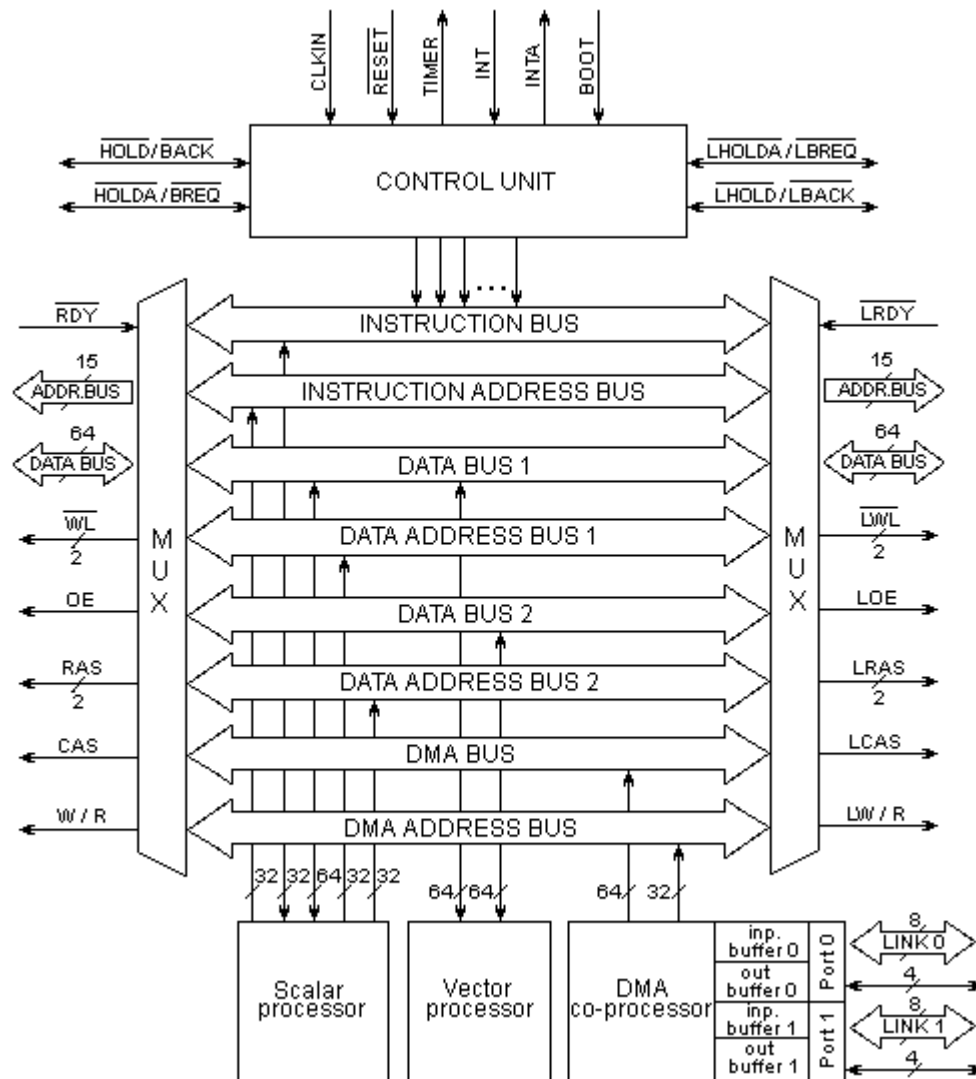


Fig.5 General structure of neurochip

The neurochip contains:

- *Vector processor* which permits basic operations in neural nets processing data bases where data is represented as vectors with elements which could have variable lengths. The main block of the vector processor is the Operational Unit (OU) described above;
- *Scalar processor* used for instruction address calculation, control of their selection, calculation of addresses of operands and weights coefficients when working with memory and also to support the scalar operations on data;
- two identical programmable interfaces with local and global busses (*MUX*) which can work with two external memories thru 64 bit wide data busses. The memory can be both DRAM and SRAM. The type of memory is determined during initialisation;
- two communication ports with *DMA coprocessors* supporting memory access when using duplex one byte links (Link0 and Link1) thru communication ports (Port0 and Port1) working in the background;
- *Control unit* does the general control, arbitrates the use of the external memory and supports 8 interrupts – one external and 7 internal including the interrupts from system timer at the conclusion of the link exchange. User can use step by step interrupt during debugging.

The neurochip uses the following internal busses:

- *Instruction bus* - 32 bit wide for instructions fetching from the memory;
- *Instruction address bus* - 32 bit wide for the address of the selected instruction;
- *Data bus 1* and *Data bus 2* 64 bit wide each to work with operands and load weight coefficients into the vector processor;
- *Data address bus 1* and *Data address bus 2* 32 bits each;
- *DMA bus* - 64 bit wide to have exchanges with links in DMA mode;
- *DMA address bus* - 32 bit wide.

Memory busses have a continuous addressing. The access to one of the two memory busses is determined by higher bit of the selected address. This permits to have data and instructions anywhere in the memory. The memory exchange with each one of them can be done in 64 or 32 bits when the lower bit of the calculated address points to the half of 64 bit word with which to work.

### **The main registers**

There is the following set of 32 bit registers in the neuroprocessor:

- eight *address* registers AR0-AR7 used for memory access; the address register AR7 has an alternative name SP (stack pointer). This register can be used to access the memory using stack principle;
- eight *general* registers GR0-GR7, to keep data read from memory and the intermediate results of the calculations;
- register *PC (Program Counter)*, to determine the address of the next instruction to be executed. The program can read and write PC register, i.e. dynamic control over the sequence of instruction execution;
- register *PSWR (Program Status Word Register)* holds the information on the current state of the processor including flags, processor resources (f.e. local and global bus information), and the current state of the interrupt masks. All register fields can be read. Mask fields can be written;
- two 64 bit registers to point the length of synapses and weights.

### **Neurochip Instruction Set**

Neurochip has 32 bit instructions which can be divided into 4 main groups: scalar processor instructions, vector processor instructions, control instructions and transfer instructions.

Scalar processor instruction set is so organised that the basic operations (i.e. general register operations) can be executed together with address registers operations.

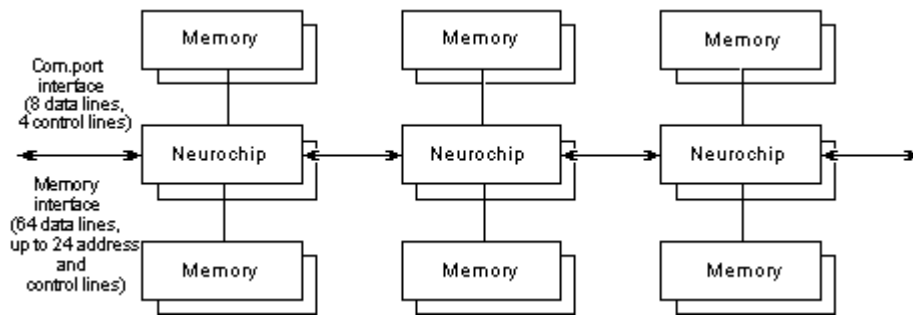
### **Interrupt system**

Neurochip has one external and a number of internal interrupts: two timer interrupts, four communication ports interrupts, step-by-step interrupt for debugging mode.

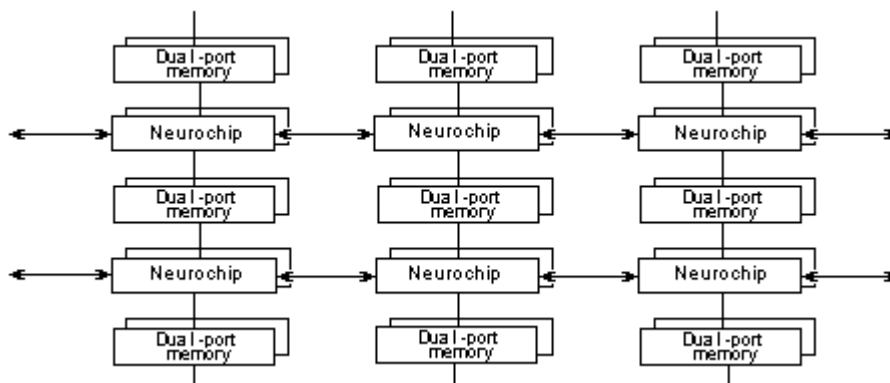
## **3. The Architectural Basis for Building Neurosystems Based on the Neurochip**

The architectural features of the neurochip which permit it to be used for building parallel neurosystems are two byte wide duplex high speed communication ports which are hardware compatible with ports of TMS320C40 and support for the shared use of memory. Combing

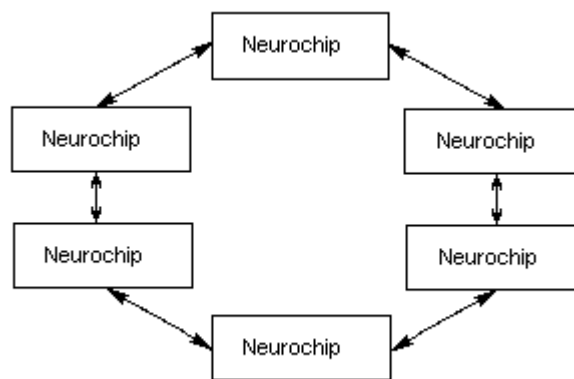
neurochips thru different communication schemes permits to implement a large variety of neural net configurations. Fig.6 shows some examples of neural net systems based on the neurochip.



a)



b)



c)

Fig.6 Examples of neural nets based on the neurochip:  
a) two-direction conveyor (for matrix operations, emulation of direct propagation neural nets and other conveyor operations),  
b) two dimensional grid structure (for matrix operations and emulation of direct propagation neural nets,  
c) two directional ring (for emulating different neural nets including back propagation and multilayer ones)

Apart from the given examples one can implement calculating networks of practically any configuration using TMS320C40 as the switching element.

Fig.7 shows some examples of such systems:

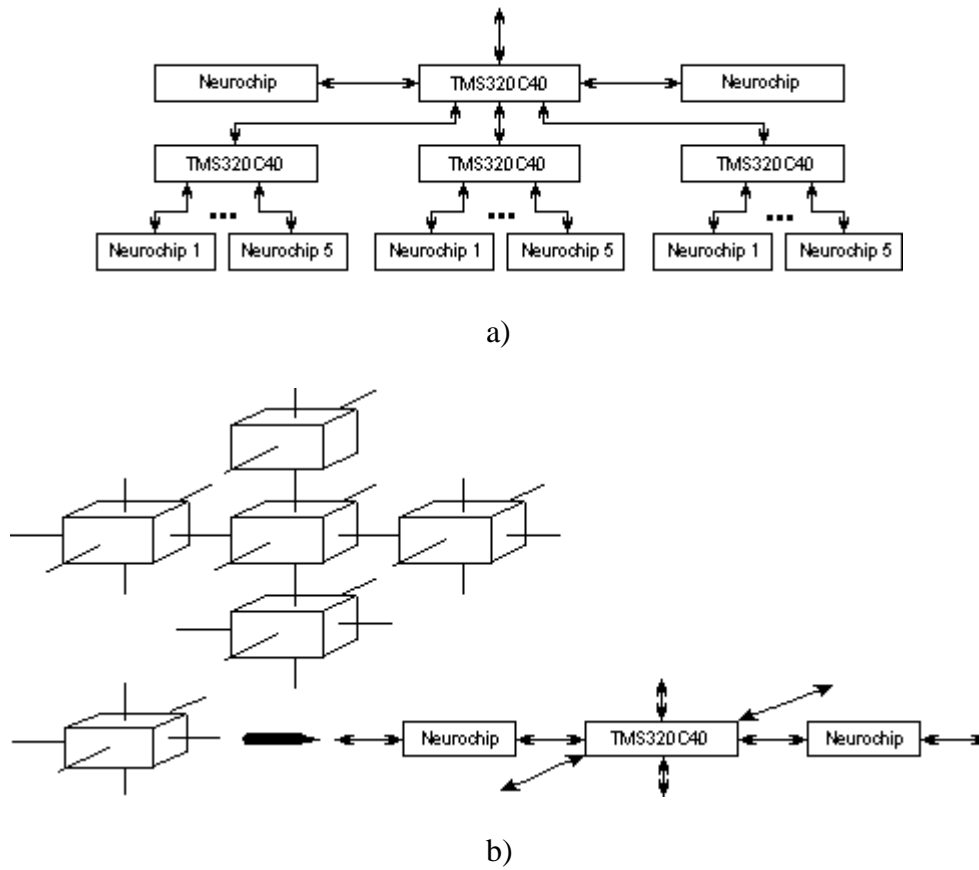


Fig.7 Examples of neural nets based on the neurochip and TMS320C4x as switching element:  
a) tree structure (for emulating multilayer neural nets and image recognition tasks),  
b) three dimensional grid (for three dimensional neural net emulation and for image recognition).

Memory interface used determines three main groups of architectures for systems based on the neurochip (see fig.8):

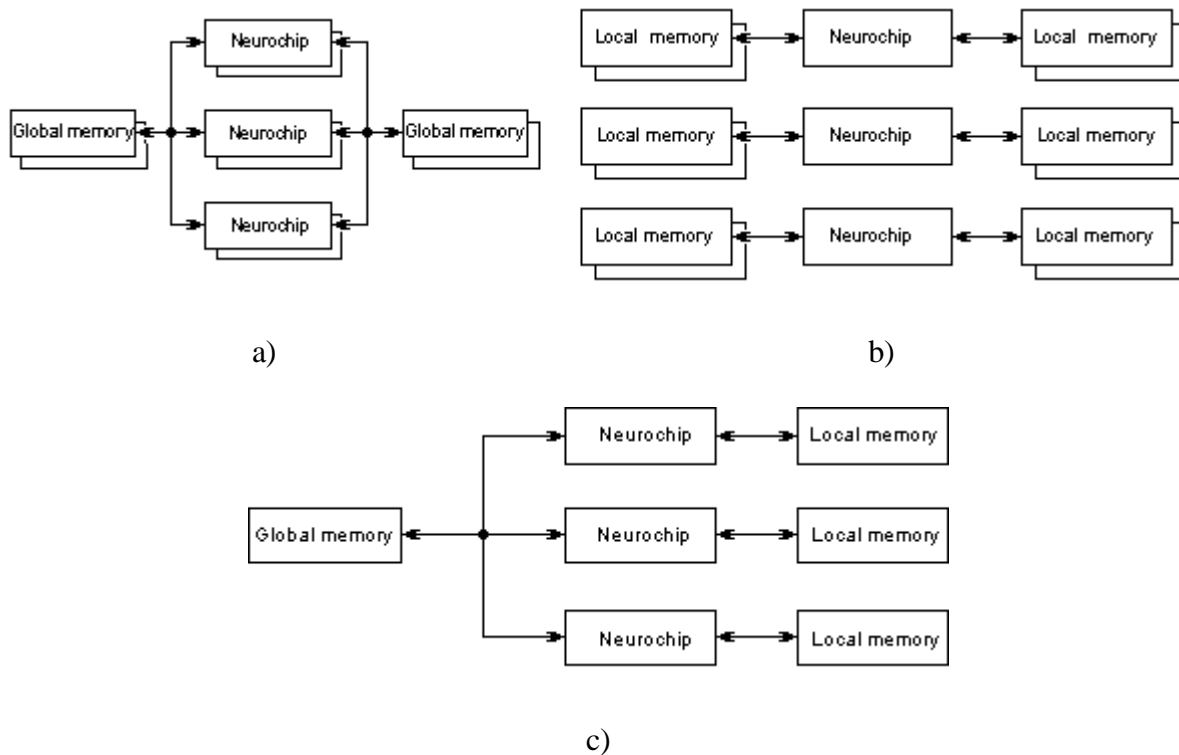


Fig.8. Neural nets with shared and distributed memory:

- a) architecture with shared memory (global memory accessible to all neurochips),
- b) architecture with distributed memory (each neurochip has its own memory and the chips link thru communication ports),
- c) mixed architecture (each neurochip has a local memory but can access common memory used by other neurochips)

When few neurochips are used in the system a mixed architecture approach can be used but if the number of chips is large the expense of organizing shared access to common memory is too big. So in that case we recommend a distributed memory approach.

#### 4. Comparative Characteristics of Architectures Based on Neurochip, TMS320C40 and Intel P55C

For comparison purposes we have taken commercially available processors which can do several additions (TMS320C4x) [4] and multiplications with carry (Intel P55C) [5] in one clock cycle.

The neurochip architecture is aimed to solve specific class of tasks - matrix-matrix type, and is very effective here (40 times better in cycles efficiency). The vector-matrix type tasks do not use effectively all the potential of the neurochip operational unit of the vector processor but still it is 4 to 8 times faster than on the other processors. And finally vector-vector operations are comparable in efficiency to TMS320C40 and are worse compared to Intel P55C.

##### 1. Matrix-matrix operations (in cycles):

- o Product of 8 bit matrixes:
  - 43 times faster than TMS320C40 not less than 20 times compared to Intel P55C.
- o Folding operation with mask on the 8 bit value matrix (in clock cycles):
  - 6-20 times faster than TMS320C40;

- 3-20 times faster than Intel P55C, depending on the size of the mask.
- 2. Vector-matrix operation (in clock cycles):
  - o multiplication of vector by matrix (8 bit elements):
    - 6 times faster than TMS320C40;
    - 4-8 times faster than Intel P55C.
- 3. Vector-vector operations (in cycles):
  - o Scalar product of vectors (8 bit elements):
    - same as TMS320C40;
    - approximately same as Intel P55C.

If we assume that the main neural net operation is multiplication of vector by matrix we have 4 to 6 times better efficiency. For image processing the main operation is folding and we can assume the increase in efficiency 3 to 10 times depending on the size of the mask

## 5. Conclusion

The proposed neurochip being a universal type can be used as a basic element for building neuroaccelerators for PC's, neurocomputer systems, hardware support for matrix operations of any size and for digital signal processing. The possibility to process the variable length data permits to increase efficiency by decreasing length of operands and thus to control optimum relation of precision/efficiency. The neurochip can be used as a stand alone system due to the comprehensive instruction set, powerful address arithmetic, high speed communication ports compatible with the well known processor as well as the programmable interface for external memory access.

The limits of the magazine article did not give us the opportunity to give a more detailed description of the neurochip architecture and to reveal such important aspects as the structure of conveyor instructions, original schematic solutions used in the design of arithmetic units (activation function, random bit length of operands and the carry methods, etc.). The article did not touch the very modern ECAD technologies we have used when designing and manufacturing the neurochip but we hope that we could do that in our future articles.

## 6. References

1. Jan N.H Heemskerk. Neurocomputers for Brain-Style Processing. Design, Implementation and Application. PhD thesis. Unit of Experimental and Psychology Leiden University, The Netherlands. 1995
2. Clark S. Lindsey, Thomas Lindblad. Survey of neural network hardware. Physics Dept.-
3. Frescati, Royal Institute of Technology. Sweden. SPIE Vol. 2492. May 1995.
4. Mauduit 1992: N. Mauduit, M. Duranton, J. Gobert, "L-Neuro 1.0: A Piece of Hardware LEGO for Building Neural Network Systems", IEEE Trans. of Neural Networks, vol.3, no.3, pp.414-422, May 1992.
5. Texas Instruments. TMS320C4x User's Guide. 1993.
6. Intel Corporation. MMX-Technology Overview.