Using of microprocessor NM6403 for neural net emulation

P.A. Chevtchenko[a], D.V. Fomine[a], V.M. Tchernikov[a], and P.E. Vixne[a]

[a]RC Module, 3 Eight March 4th Street, Box 166, Moscow, 125190, Russia,
tel. +7-095-152-9335, fax. +7-095-152-4661,
E-mail: fomin@module.vympel.msk.ru

ABSTRACT

This paper presents an approach of large neural net emulation by using new neuroprocessor NM6403, designed by RC Module. Model of neural net, hardware supported by neuroprocessor architecture, is discussed and some key features of NM6403, such us processing of data with variable bit length and two 64-bit external buses, are highlighted. Examples of emulation of neural nets by one neuroprocessor are shown and its performance is estimated. Proposals are also examined how to build large super parallel computing systems with NM6403 as a basic block. There are two hardware supported ways to connect neuroprocessor with other ones: using of shared memory mode on any of two external buses or two communication ports, compatible with those of DSP TMS320C4x. These abilities allow to build various parallel structures as trees, rings, grids and so on.

Keywords: neural network, neuroprocessor, vector arithmetic.

1. INTRODUCTION

Today various electronic companies produce microprocessors intended for neural net emulation. There are optoelectronical, analog, digital and hybrid neurochips [1]. The most accurate and universal data calculations are provided by digital neuroprocessor implementations due to their programmability [2]. One of the most important digital neuroprocessor feature that allows to call it a universal neurochip is a variable bit length data processing ability. There are some neurochips that operate with data the bit length of which can be software assigned in the range from 1 to 16 bit [3, 4]. Usually such processors conduct calculations over 1-bit, 8-bit and 16-bit data only. The only one known exception is Lneuro chip, designed by Philips [5]. As it was reported, this neurochip is able to process data of any bit length in the range from 1 to 8 bit. However, this neurochip has sequential (serial) data processing mode and this is the main feature that considerably reduces its performance.

The new 64-bit neuroprocessor NM6403 is a high performance microprocessor with hardware implemented matrix-by-matrix, matrix-by-vector multiplication, vector addition, saturation function for vector data and other vector operations support. This processor performs some operation over data vectors for one clock cycle. Each data vector is a 64-bit word of packed integer data. The bit length of each data in a packed data word can be any value in the range from 1 to 64 bits. The smaller is data bit length in a packed data word the larger is the number of data in a vector and thus neuroprocessor performances increases. Neuroprocessor has hardware features for multiprocessor systems support. It also can be used in various signal processing systems, not only neural net based. Nevertheless the main NM6403 purpose is neural net emulation.

2. NEURAL NETWORK MODEL

A neural net layer model emulated by NM6403 neuroprocessor is shown in Fig. 1. In general case one neural net layer contains N neuron inputs and consists of M neurons. The m-th neuron performs the multiply-accumulate operation over N data X_1, X_2,…, X_N, that feed respective neuron inputs. Also each neuron biases V_m are taken into account:

$$Y_m = V_m + \sum_{n=1}^{N} X_n \times W_{n,m} ,$$

where $W_{n,m}$ - is a weight coefficient for the n-th input in the m-th neuron (n=1,2,…,N; m=1,2,…,M). After that the saturation function F_{Q_m} with multiply-accumulate operation result Y_m is calculated for the m-th neuron:

$$OUT_m = F_{Q_m}(Y_m) ,$$

where Q_m - is a parameter of saturation function calculated for the m-th neuron. The value of Q_m is equal to the number of the significant bits in the result R_m. General saturation function supported by neuroprocessor are shown in Fig. 2. All input data, weight coefficients, bias values and resulting data are represented as two's complement.

Neuroprocessor NM6403 is a universal engine that is able to emulate a wide range of different neural net types. It allows to program neural net parameters such as, for example, number of net layers, number of neurons and number of neuron inputs at each net layer, data precision (e.g. data bit length) at each neuron input, weight coefficients precision, neuron outputs precision and saturation function parameter for each neuron.

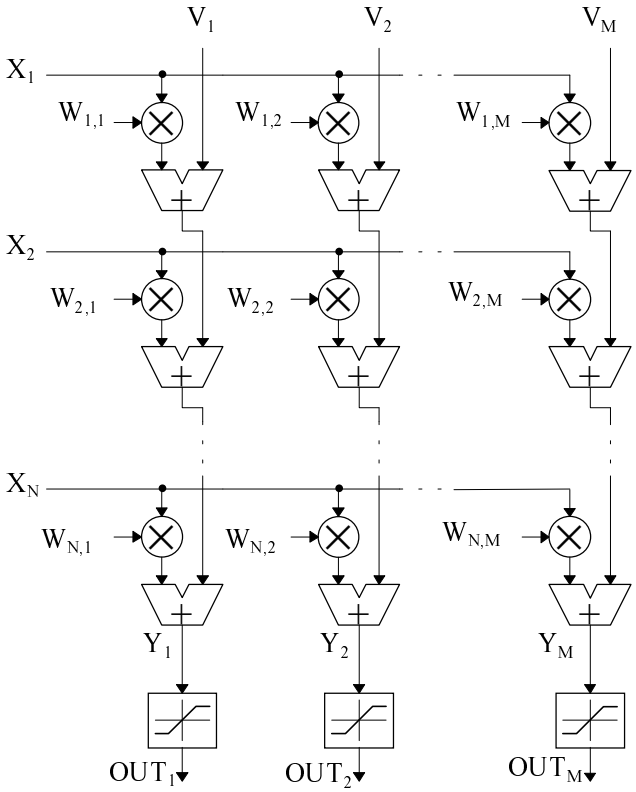

Figure 1. Neural network layer model

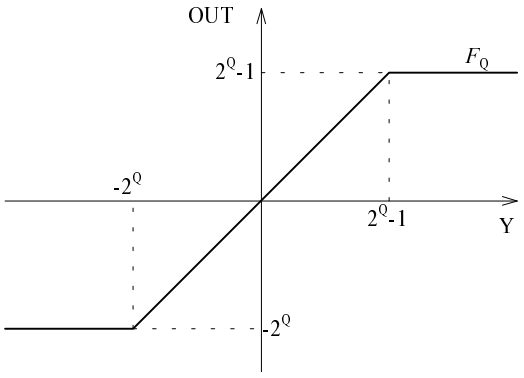

Figure 2. Saturation function, calculated by neuroprocessor

3. GENERAL APPROACH TO NEURAL NETWORK EMULATION

Single neuroprocessor NM6403 allows to emulate a neural net of practically unlimited dimensions. Emulation is performed layer by layer.

Each neural net layer is divided into sub-blocks that are processed sequentially. The division into sub-blocks is conducted according to the following rules. The set of neuron inputs of a whole layer is divided so that the total data bit length of neuron inputs group is equal to neuroprocessor data size - that is 64 bit. The set of neurons of the whole layer is divided so that the total result data bit length for group of neurons is equal to 64 bit too. That mean that neural net layer is divided by two types of sub-blocks. The set of sub-blocks is functionally different. Each sub-block of type 1 performs multiply-accumulation of data that are fed to neuron inputs of one group for each neuron that belongs to one neuron group. Each sub-block of type 2 forms outputs for all neurons that belong to one group by calculation of the saturation function with respect to multiply-accumulate results.

Fig. 1 can be used to illustrate the indicated neural net layer division. Let's imagine that each block shown in Fig. 1 performs operations over 64 bit vector data and all inputs and block-to-block connections serve for transfer of such vectors. In Fig.1 you can see that each sub-block of type 1 corresponds to a pair of devices performing multiplication and accumulation and each sub-block of type 2 corresponds to one device for saturation functions.

The emulation process of a neural net in the neuroprocessor NM6403 consists of macrooperations performed sequentially. Each of these macrooperations emulates one sub-block of a layer. Here the number of macrooperations is equal to the number of sub-blocks in the layer. The neuroprocessor processes data fed to neuron inputs in block mode - by T sets of input data in each block. When performing each macrooperation on a neural net sub-block emulation the calculations for each set of input data are done sequentially. For one run of all sub-blocks in a layer the neuroprocessor hence calculates T sets of output values of neurons, each of them corresponding to one of the input data sets. By processing of one input data block the values of weight coefficients and neurons biases do not change. The value T is program assigned and it can assume any integer value in the range from 1 to 32.

4. DATA FORMATS PROCESSED DURING NEURAL NETWORK EMULATION

Data block that is fed to n-th neuron inputs group of neural net layer (n = 1,2,…,N) is a vector $\mathbf{X}_n = \begin{pmatrix} \mathbf{X}_n^1 \\ \mathbf{X}_n^2 \\ \vdots \\ \mathbf{X}_n^T \end{pmatrix}$ the t-th element of which (t=1,2,…,T) is a vector $\mathbf{X}_n^t = (X_{n,1}^t \quad X_{n,2}^t \quad \cdots \quad X_{n,N_n}^t)$ the ν-th element $X_{n,\nu}^t$ of which (ν = 1,2,…, N_n) is a data from t-th input data set that is fed to ν-th input of n-th neuron inputs group of current neural net layer.

Vector $\mathbf{X}_n^t$ is a 64-bit word that contains N_n integer packed data. The lowest bits of $\mathbf{X}_n^t$ vector belong to first packed data $X_{n,1}^t$ and they are followed by second packed data bits $X_{n,2}^t$ and so on. The highest bits of $\mathbf{X}_n^t$ vector belong to N_n-th packed data X_{n,N_n}^t. The bit length of each data that is packed into $\mathbf{X}_n^t$ vector may be any even value in a range from 2 to 64. As a result the packed data number N_n into $\mathbf{X}_n^t$ vector may be any integer value from 1 to 32. The only restriction is that the total bit length of data packed into one $\mathbf{X}_n^t$ vector has to be equal to neuroprocessor data size - 64 bit.

To emulate the first type neural net sub-block that conduct multiply-accumulate of n-th input data group for m-th neuron group (n=1,2,…,N; m=1,2,…,M) the neuroprocessor process following operation for t-th input data set (t=1,2,…,T):

$$\mathbf{Y}_{n,m}^t = \mathbf{Y}_{n-1,m}^t + \mathbf{X}_n^t \times \mathbf{W}_{n,m}, \qquad (1)$$

where $\mathbf{W}_{n,m} = \begin{pmatrix} W_{n,m,1,1} & W_{n,m,1,2} & \cdots & W_{n,m,1,M_m} \\ W_{n,m,2,1} & W_{n,m,2,2} & \cdots & W_{n,m,2,M_m} \\ \vdots & \vdots & & \vdots \\ W_{n,m,N_n,1} & W_{n,m,N_n,2} & \cdots & W_{n,m,N_n,M_m} \end{pmatrix}$ is a matrix, the $W_{n,m,\nu,\mu}$ element of which represents the weight coefficient of ν-th input of n-th neuron inputs group for μ-th neuron at m-th neuron group (ν=1,2,…,N_n; μ=1,2,…,M_m);

$\mathbf{Y}^t_{n,m} = (Y^t_{n,m,1} \quad Y^t_{n,m,2} \quad \cdots \quad Y^t_{n,m,M_m})$ is a vector the μ-th element $Y^t_{n,m,\mu}$ of which is a multiply-accumulation result of data that are fed to first n input groups for μ-th neuron at the m-th neuron group (μ=1,2,…,M_m):

$$Y^t_{n,m,\mu} = Y^t_{n-1,m,\mu} + \sum_{\nu=1}^{N_n} X_{n,\nu} \times W_{n,m,\nu,\mu} .$$

By the way, as a $\mathbf{Y}^t_{0,m}$, vector that are processed by first sub-block of m-th neuron group the $\mathbf{V}_m = (V_{m,1} \quad V_{m,2} \quad \cdots \quad V_{m,M_m})$ vector is used. The μ-th element $V_{m,\mu}$ of said vector is a bias of μ-th neuron from m-th neuron group (μ=1,2,…,M_m).

The 64-bit vectors $\mathbf{Y}^t_{1,m}$, $\mathbf{Y}^t_{2,m}$, … , $\mathbf{Y}^t_{N,m}$ è $\mathbf{V}_m$ are all at the same format. They contain the same number M_m of data packed into them. The bit length of each of said data has to be fixed at the integer range from N_{min} to 64. The minimal bit length value of each vector element depends on total neuron inputs number at a layer and results from the necessity to avoid arithmetical overflow while one significant bit data processing:

$$N_{min} = 1 + \left\lceil \log_2 \left(\sum_{n=1}^{N} N_n \right) \right\rceil .$$

The M_m parameter had to fall in a range from 1 to $\lfloor 64 / N_{min} \rfloor$.

The ν-th row (ν=1,2,…,N_n) of the $\mathbf{W}_{n,m}$ matrix represents the weight coefficients vector $\mathbf{W}_{n,m,\nu} = (W_{n,m,\nu,1} \quad W_{n,m,\nu,2} \quad \cdots \quad W_{n,m,\nu,M_m})$ that has the format that is same to the $\mathbf{Y}^t_{1,m}$, $\mathbf{Y}^t_{2,m}$, … , $\mathbf{Y}^t_{N,m}$ and $\mathbf{V}_m$ vectors.

The execution of operation (1) over all vectors of $\mathbf{X}_n$ data block is defined by single ***vsum*** neuroprocessor instruction. This instruction is executed by neuroprocessor during T clock cycles. The operation (1) over $\mathbf{X}^t_n$ and $\mathbf{Y}^t_{n-1,m}$ vectors is executed during t-th execution cycle(t=1,2,…,T).

The neuroprocessor configuration for required data bit length is provided by instruction of load of the control words to respective neuroprocessor control registers:

XB_n - control word that defines element borders for $\mathbf{X}^t_n$;

YB_m - control word that definess element borders for $\mathbf{Y}^t_{1,m}$, $\mathbf{Y}^t_{2,m}$, … , $\mathbf{Y}^t_{N,m}$, $\mathbf{V}_m$, $\mathbf{W}_{n,m,1}$, $\mathbf{W}_{n,m,2}$, … , $\mathbf{W}_{n,m,N_n}$ (t=1,2,…,T).

To emulate second type neural net sub-block for m-th neuron group (m=1,2,…,M) neuroprocessor forms vector block $\mathbf{OUT}_m = \begin{pmatrix} \mathbf{OUT}^1_m \\ \mathbf{OUT}^2_m \\ \vdots \\ \mathbf{OUT}^T_m \end{pmatrix}$, the t-th element of which (t=1,2,…,T) is a vector $\mathbf{OUT}^t_m = (OUT^t_{m,1} \quad OUT^t_{m,2} \quad \cdots \quad OUT^t_{m,M_m})$, the μ-th element $OUT^t_{m,\mu}$ of which is a result of saturation function calculation $F_{Q_{m,\mu}}$ with an μ-th element $Y^t_{N,m,\mu}$ of $\mathbf{Y}^t_{N,m}$ (μ=1,2,…,M_m) vector as an input data:

$$OUT^t_{m,\mu} = F_{Q_{m,\mu}}(Y^t_{N,m,\mu}) .$$

The saturation function for each element of each vectors from $\mathbf{OUT}_m$ block is specified by ***activate*** neuroprocessor instruction. This instruction is executed by neuroprocessor during T clock cycles. During t-th calculation cycle the $\mathbf{OUT}^t_m$ (t=1,2,…,T) vector is formed. This vector has the same format as the said $\mathbf{Y}^t_{1,m}$, $\mathbf{Y}^t_{2,m}$, … , $\mathbf{Y}^t_{N,m}$ and $\mathbf{V}_m$ vectors.

Required data format for saturation function is conducted by instruction of load of control Q_m word to respective neuroprocessor control register.

5. THE NEURAL NETWORK EMULATION EXAMPLE

The neural network layer emulation process using the single NM6403 neuroprocessor can be represented by M sequentially executed procedures. Each procedure emulates one neuron group and consist of N+1 sequentially executed macrooperations. Each macrooperation emulates one neural net layer sub-block. Hence, n-th macrooperation of given procedure emulate first type sub-block that performs multiply-accumulate of data that are fed to n-th neuron inputs group with result accumulation(n=1,2,…,N). The last macrooperation of procedure emulates second type sub-block that performs saturation function calculation with the result of multiply-accumulate operation for respective neuron group as an input data.

In Fig.3 you can see an example of emulation procedure of the m-th neuron group for T sets of input data (m=2,3,…,M). The emulation procedure of the first group has some peculiarities on the phase of the first sub-block emulation (see Fig. 4). It is regarded to the necessity of load of some initial data common for all the procedures and with the absence of result of the previous neuron group emulation. result Other sub-blocks of the first group of neurons are emulated in the same way as in the other neuron groups. The examples are written in assembler of the NM6403 neuroprocessor. In order to make it simpler the procedures are described without using cycles, branches and other operators controlling the instructions flow. Each line of the procedure contains one assembler instruction and a comment to it starting with symbols "//".

The following neuroprocessor registers and blocks of internal memory are used in the considered procedures:
WOPER - memory block for storage of the weight coefficients matrix used in the operations of multiple-accumulation;
WBUF - memory block for accumulation of vectors forming the weight coefficients matrix;
WFIFO - FIFO, aimed for synchronization of the processes of weight coefficients vectors reading from the external memory and accumulation of the weight coefficients matrix in WBUF;
AFIFO - FIFO accumulating the results by execution of operations over vector blocks;
RAM - one-port memory aimed for operative storage of a vector blocks;
AR0, **AR1**, **AR4**- address registers;
VR - register for the bias vector storage;
F2CR - register for storage of the saturation functions parameters calculated for vectors of packed data;
SB1 and **SB2** - registers for storage of the control word defining the format of the vectors of packed data over which the operation of multiple-accumulation is executed;
NB1 and **NB2** - registers for storage of the control word defining the format of the result vectors for execution of operations over packed data vectors.

The contents of the registers SB1 and NB1 is used to control the process of the weight coefficients matrix accumulation in WBUF and the contents of the registers SB2 and NB2 - to control the process of execution of operations over packed data vectors. The presence of WBUF, SB1 and NB1 in the neuroprocessor provides simultaneous run of two processes: execution of multiple-accumulation of the current vector blocks and loading of weight coefficients matrix for processing of the next vector block. The neuroprocessor instruction set contains instructions of two types: scalar and vector.

Scalar instructions are usual for RISC-processors and executed with one per clock cycle performance. Scalar instructions of two types are used in the presented example:
<register> = <variable name>; - assigning of the variable address to the respective register;
<register> = [<variable name>]; -transfer of the variable from the external memory to the respective register.
Vector instructions are intended to process operations over vector blocks of packed data. The assembler mnemonic for these instructions is terminated by "**;**" symbol and may contains some operators divided by commas or ***with*** reserved word.

```
//******* Emulation of the first sub-block of m-th neuron group *******
.branch;
nb1 = [YBm];                          // Load control word YBm into NB1 register
sb1 = [XB1];                          // Load control word XB1 into SB1 register
rep N1 wfifo = [ar4++], ftw;          // Load data for W1,m matrix from memory into WFIFO
                                      // and their reload from WFIFO into WBUF
wtw;                                  // Reload W1,m data from WBUF into WOPER
ar0 = X2;                             // Load X2 data block address into AR0 register
vr = [Vm];                            // Load bias data Vm into VR register
rep T [ar1++] = afifo with vsum, ram, vr;
                                      // Store OUTm-1 from AFIFO into external memory,
                                      // calculate vector Y1,m^t = X1^t × W1,m + Vm
                                      // (vector X1^t is taken from external memory)
                                      // and store them into AFIFO (t=1,2,…,T)

//******* Emulation of the second sub-block of m-th neuron group *******

sb1 = [XB2];                          // Load control word XB2 into SB1 register
rep N2 wfifo = [ar4++], ftw;          // Load data for W2,m matrix from memory into WFIFO
                                      // and their reload from WFIFO into WBUF
wtw;                                  // Reload W2,m data from WBUF into WOPER
rep T data = [ar0++] with vsum, data, afifo;
                                      // Calculate Y2,m^t = X2^t × W2,m + Y1,m^t
                                      //(vector X2^t is taken from external memory)
                                      // and store them into AFIFO (t=1,2,…,32)
      ⋮
//******* Emulation of the N-th sub-block of m-th neuron group *******

sb1 = [XBN];                          // Load control word XBN into SB1 register
rep NN wfifo = [ar4++], ftw;          // Load data for WN,m matrix from memory into WFIFO
                                      // and their reload from WFIFO into WBUF
wtw;                                  // Reload WN,m from WBUF into WOPER
rep T data = [ar0++] with vsum, data, afifo;
                                      // Calculate YN,m^t = XN^t × WN,m + YN−1,m^t
                                      //(vector XN^t is taken from external memory)
                                      // and store them to AFIFO (t=1,2,…,32)

//******* Emulation of the (N+1)-th sub-block of m-th neuron group *******

f2cr = [Qm];                          // Load control word Qm into F1CR register
ar1 = OUTm-1;                         // Load data block address OUTm-1 into AR1 register
rep T with vsum, 0, activate afifo;
                                      // Calculate OUTm^t = FQm(YN,m^t),
.wait;                                // with following store them into AFIFO (t=1,2,…,32)
```

Figure 3. The m-th neuron group emulation procedure (m=2,3,…,M)

```
//******* Emulation of the first sub-block of the first neuron group *******

ar4 = W1,1;                          // Load W1,1 matrix data address into AR4 register
nb1 = [YB1];                         // Load control word YB1 into NB1 register
sb1 = [XB1];                         // Load control word XB1 into SB1 register
rep N1 wfifo = [ar4++], ftw;         // Load data for W1,1 matrix from memory into WFIFO
                                     // and their reload from WFIFO into WBUF
wtw;                                 // Reload W1,1 from WBUF into WOPER
ar1 = OUT1;                          // Load OUT1 data block address into AR1 register
ar0 = X1;                            // Load X1 data block address into AR0 register
vr = [V1];                           // Load bias vector V1 into VR register
rep T data, ram = [ar0++] with vsum ,data, vr;
                                     // Calculate Y1,1t = X1t × W1,1 + V1
                                     // Load vectors X1t from memory into RAM,
                                     // and store them into AFIFO (t=1,2,…,32)
```

Figure 4. The first sub-block of the first neuron group emulation procedure

Each instruction operator defines one vector operation to execute. All operations that are specified at one vector instruction are executed simultaneously.

Vector instruction may has a prefix "**rep <K>**" that means that each operator of this instruction will be executed K times. So, the single vector instruction is equivalent to parametrical cycle the body of which contains one instruction and repeat count is equal to K.

The following vector instruction operators are used in the examples presented in Fig. 3 and 4:
wfifo = [ar<i>++] - transfer to WFIFO of the weight coefficients vector from the external memory addressed by the register ARi using the autoincrement method of addressing;
ftw - transfer of the weight coefficients matrix from WFIFO to WBUF (the operation is execution for 32 clock cycles independently on the parameter T value in the prefix "rep <T>");
wtw - simultaneous transfer of the weight coefficients matrix from WBUF to WOPER, of the register SB1 contents to the register SB2 and of the register NB1 contents to the register NB2 (this operation is executed for one clock cycle);
data = [ar<i>++] - reading of the packed data vector from the external memory addressed by the register ARi using the autoincrement method of addressing;
data, ram = [ar<i>++] - reading of the packed data vector from the external memory addressed by the register ARi using the autoincrement method of addressing and its recording to RAM;
[ar<i>++] = afifo - transfer of the packed data vector from AFIFO to the external memory addressed by the register ARi using the autoincrement method of addressing;
vsum, ram, vr - addition of the vector stored in the register VR with the production of the vector read from RAM by the weights matrix stored in WOPER and recording of the result vector to AFIFO;
vsum, data, afifo - addition of the vector read from AFIFO with the production of the vector read from the external memory with the help of one of the operators described above by the weights matrix stored in WOPER and recording of the formed resulting vector to AFIFO;
vsum, 0, activate afifo - calculation of the saturation function with regard to the elements of the vector read from AFIFO and recording of the result vector to AFIFO.

The NM6403 neuroprocessor pipeline allows to start next scalar or vector instruction execution in parallel when previous vector instruction not finish yet. It is possible when the neuroprocessor resources that is necessary for next vector instruction execution are not used by previous ones. In general, neuroprocessor allows to execute up to five vector and one scalar instructions simultaneously. At the example represented at Fig. 3 all instructions into one large bracket "{" at the left of instructions are executed at the background of previous vector instruction. The parallel instruction execution is allowed by **.branch;** assembler directive until the **.wait;** assembler directive is mentioned.

When high speed memory used which allows one per clock cycle read/write cycle, each scalar instruction is executed at one clock cycle, each vector instruction that contains "**ftw**" operator is executed 32 clock cycles (clock cycles number not depends on possibly presented prefix "rep <K>"), and each vector instruction without "**ftw**" operator is executed K clock cycles. Generally, neural network emulation for T input data sets requires approximately $34 \times (N+1) \times M \times \lceil T/32 \rceil$ clock cycles. The best results can be achieved when K parameter for instructions that contains vector multiply-accumulate operation has the value of 32. Using of the smaller K parameter value results in neuroprocessor computation stalls during the new weight coefficients matrix load to WBUF and partial performance degradation.

Unfortunately, the paper size prevent us to illustrate some other neuroprocessor features such as element mapping into the vectors of packed data. This feature allows to perform more powerful data packing in vectors that are formed for each layer of emulated neural net. This operation may be executed simultaneously with saturation function calculation and results in output vectors number shrinking.

6. MULTIPROCESSOR SYSTEMS BASED ON NEUROPROCESSOR NM6403

Neuroprocessor NM6403 comprises the following features for multiprocessor system support:

- two byte wide I/O communication ports CP0 and CP1. Each port allows point-to-point data transfer between neuroprocessor and external device with the same port with the throughput of 20 Mbytes/s;
- two programmable external memory interfaces with 64-bit external data buses local and global. Each programmable interface supports up to three multiprocessor system configurations. Some neuroprocessors connected to the same data bus are able to exchange data via shared memory mechanism. Shared memory operations for two neuroprocessors may be provided without any external controller. The peak throughput for one common data bus is up to 400 Mbytes/s.

Examples of multiprocessor system based on neuroprocessor NM6403 are shown in Fig. 5 and Fig. 6. These are linear (circular) and matrix (toroidal) structures. When communication ports are used to exchange information between neuroprocessors the CP0 port of neuroprocessor 1 must be connected to CP1 port of neuroprocessor 2. When shared memory data exchange is used then the mentioned shared memory must be local for one neuroprocessor and global for the other neuroprocessor.

It is necessary to note that the neuroprocessor NM6403 communication ports are functionally and electronically compatible with DSP TMS320C4x by Texas Instruments [7]. Various modifications of this DSP have from four to six communication ports. That make it possible to use DSP as commutation element in a more complicated neuroprocessor based multiprocessor. Multiprocessor system of K neuroprocessors NM6403 will emulate a neural net by K times faster then single neuroprocessor. In the best case each sub-block of each neural net layer must be emulated by its own neuroprocessor.

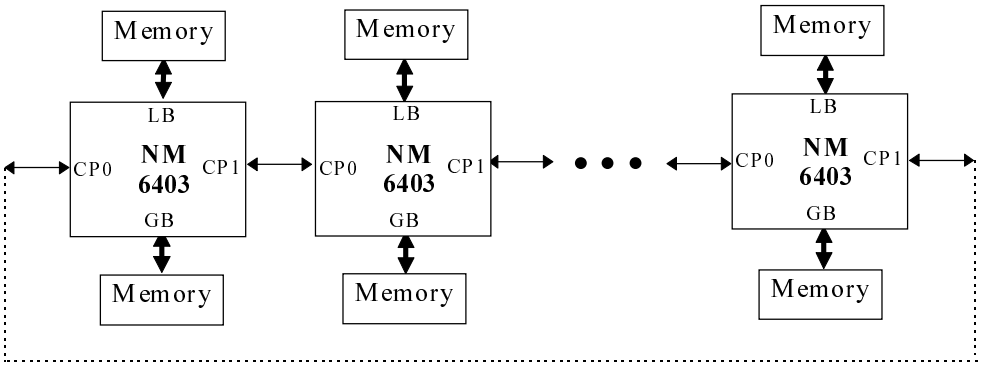

Figure 5. Linear(cyclic) multiprocessor systems based on neuroprocessor NM6403

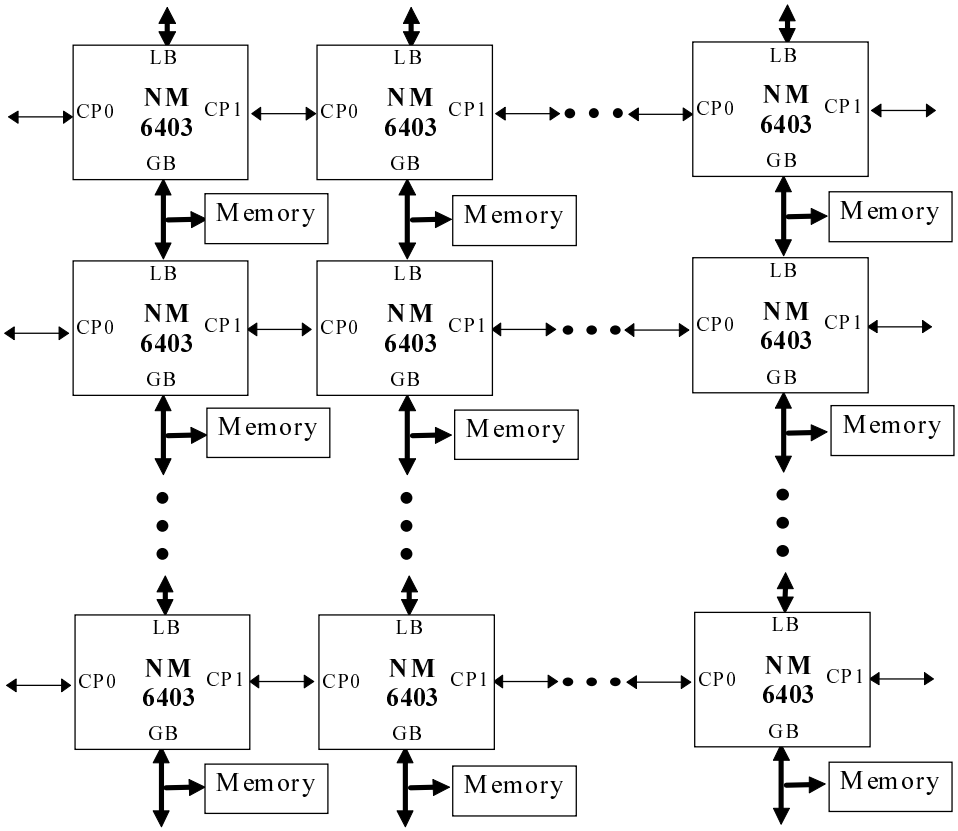

Figure 6. Matrix(toroidal) multiprocessor system based on neuroprocessor NM6403

7. NEUROPROCESSOR PERFORMANCE

The main operation that is used for neural net emulation is multiply-accumulate. So the most commonly used performance estimator for neuroprocessors is the number of Connections Per Second (CPS) [7]. CPS usually means the number of multiply-accumulate operations per second. The peak value of CPS for the neuroprocessor NM6403 largely depends on the input data bit length N_X and weight coefficients bit length N_W:

$$CPS = \left\lfloor \frac{64}{N_X} \right\rfloor \times \left\lfloor \frac{64}{N_X + N_W + \left\lceil \log_2 \frac{64}{N_X} \right\rceil} \right\rfloor \times f \qquad \text{when } N_X,>1 \text{ è } N_W>1,$$

$$CPS = \left\lfloor \frac{64}{N_X} \right\rfloor \times \left\lfloor \frac{64}{N_X + \left\lceil \log_2 \frac{64}{N_X} \right\rceil} \right\rfloor \times f \qquad \text{when } N_X,> 1 \text{ è } N_W = 1,$$

$$CPS = 32 \times \left\lfloor \frac{64}{N_W + 5} \right\rfloor \times f \qquad \text{when } N_X = 1 \text{ è } N_W > 1,$$

$$CPS = 1024 \times f \qquad \text{when } N_X = N_W = 1,$$

where $f = 50*10^6$ Hz - neuroprocessor clock rate.

The results of neuroprocessor NM6403 CPS estimation for some most commonly used values of N_X and N_W are represented in Fig. 7 in the form of histograms. These results show the high performance of the neuroprocessor. In the table below you can find CPS information concerning some known neuroprocessors. The more detailed neuroprocessor analysis information can be found in [8].

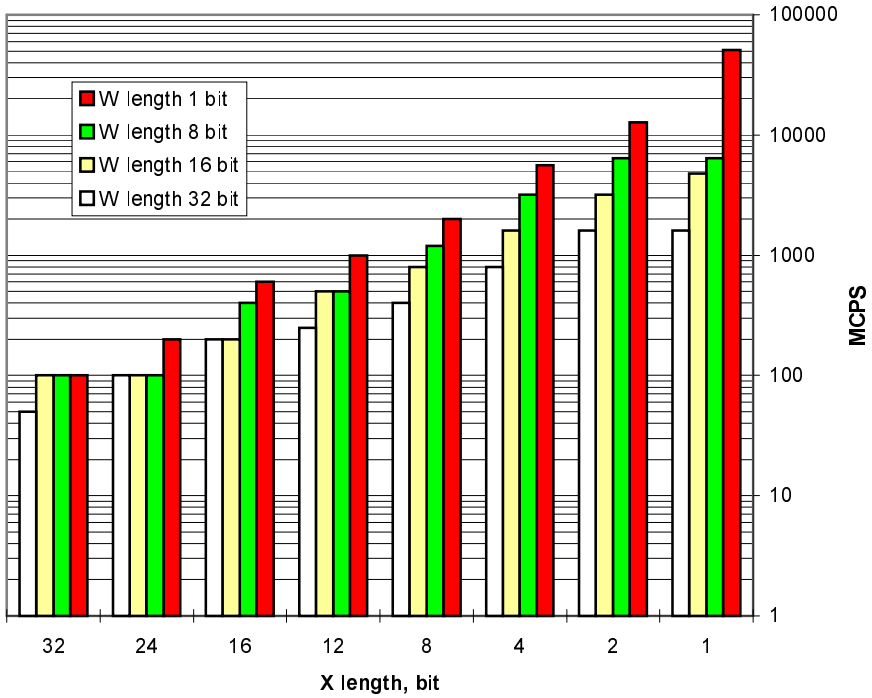

Figure 7. Neuroprocessor performance depending on bit length of input data and weight coefficients

Table 1. Digital Neuroprocessors Performance Measures

Name	***Configuration***	***CPS*** [1)]	***CPSPW*** [2)]	***CPPS*** [3)]	***CUPS*** [4)]	***Patterns/s***
Nuralogix, NLX-420	32-16, 8-bit mode	10M	20k	640M	na	20k
Hecht-Nielson, 100 NAP	4-chips, 2M wts, 16bit Mantissa	250M	125	256G	64M	na
Hitachi, WSI	576 neuron Hopfield	138M	3,7	9,9G	na	na
Inova, N64000	64-64-1, 8bit mode	871M	3.4k, 128k	55.7G	220M	100k
IBM, ZISC036	64 8bit el. vectors	na	na	na	na	250k
MCE, MT19003	4-4-1, 32MHz	32M	32M	6.8G	na	140k
Micro Devices, MD-1220	8-8	8.9M	1.1M	142M	na	139k
Nestor/Intel, Ni1000	256 5bit el. vectors	na	na	na	na	40k
Philips, Lneuro-1	1-chip, 8bit mode	26M	26k	1.6G	32M	na
Siemens, MA-16	1-chip, 25MHz	400M	15M	103G	na	40k
RC Module, NM6403	**8bit mode, 50MHz**	**1200M**	**150M**	**76.8G**	**na**	**na**

Notes: 1) CPS - Connection-Per-Second.

2) CPSPW = CPS/Nw , Nw - number of synapses per neuron.

3) CPPS - Connection Primitives per Second, CPPS = CPS*Bw * Bs, were Bw, Bs - word length of weight and synapse.

4) CUPS - Connection-Update-Per-Second.

8. CONCLUSION

The new neuroprocessor NM6403 has an unique ability to perform variable bit length data processing and increase calculation performance when data bit length is reduced. This gives the programmer wide possibilities to find the optimum between data precision and calculation performance. The performance at 50 MHz clock rate ranges from 50 MCPS (32-bit weights and 32-bit input data) to 51.2 GCPS (1-bit weights and input data).

The high performance and the powerfull interface allow to use the neuroprocessor NM6403 in neural network accelerators for PC and to apply it as the base element for large parallel neurocomputer systems.

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. P. Ienne, G. Kuhn, “Digital Systems for Neural Networks”, In P.Papamichalis and R.Kerwin, editors, *Digital Signal Processing Technology*, volume CR57 of *Critical Reviews Series*, p. 314-45, SPIE Optical Engineering, Orlando, Fla., 1995.
3. NLX420 Data Sheet, June 1992, Neurologix, Inc., 800 Charcot Av., Suite 112, San Jose, Ca. USA.
4. D. Hammerstrom, “A VLSI Architecture for High Performance, Low Cost, On-chip Learning”, Proc. Int. Joint Conf. On Neural Networks IJCNN`90, June 1990, vol.II, pp.537-544, San Diego, Ca, USA
5. N.Mauduit, M.Duration, J.Gobert, “Lneuro 1.0: A Piece of Hardware LEGO for Building Neural Network Systems”, IEEE Trans. On Neural Networks, vol.3, no. 3, pp. 414-422, May 1992.
6. Vixne P.E., Fomine D.V., Tchernikov V.M. “A VLSI Digital Neural Processor with Variable Word Length of Operands”, PRIBOROSTROENIE, *Hard-Software systems for neural calculation support*, thematic issue, 1996, v.39, №7, с.13-21.
7. TMS 320C4x User’s Guide, August 1993, Texas Instruments Inc., USA.
8. C. S. Lindsey, Th. Lindblad, “Survey of Neural Network Hardware”, Proc. SPIE Vol. 2492, p. 1194-1205, Applications and Science of Artificial Neural Networks, Steven K. Rogers; Dennis W. Ruck; Eds. , March 1995.