computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific
Journal of Computing

FAST NEURAL NETWORK FOR GENERALIZED TRIGONOMETRIC TRANSFORMATIONS SYNTHESIS

Oleh Liskevych [1], Mykhaylo Yatsymirskyy [2]

[1] State Scientific and Research Institute of Information Infrastructure,
Troleibusna Str., 11, Lviv, 79053, Ukraine, e-mail: ol@rol.com.ua, www.dndiii.lviv.ua
[2] Institute of Computer Science, Technical University of Lodz,
Wolchanska Str., 215, Lodz, 93005, Poland, e-mail: jacym@ics.p.lodz.pl, www.ics.p.lodz.pl

Abstract: *The structure of the fast hardware neural network, based on generalized trigonometric transformations algorithm is developed. The network is appointed for optimal by some given criteria transformation selection and synthesis in adaptive digital signal processing system.*

Keywords: *Digital signal processing, neural networks, generalized trigonometric transformations, unified algorithm.*

1. INTRODUCTION

Rapid development of information technologies in these latter days makes new demands to digital signal processing (DSP) methods and tools [1, 2]. Characteristic feature of DSP tasks is multiple iteration of simple primitive operations (PO), which can be usually executed in parallel. Especially it concerns fast trigonometric transformations (FTT) algorithms that are one of the most common used DSP mathematical tools [2, 3]. It is well known that DSP tasks execution on standard computers may take a lot of time. A great number of high-performance systems with application-specific architecture were developed to resolve this problem. Recently an interest in more flexible DSP systems, capable of changing their features for some range of typical applications execution, has quickened [2, 3]. Such systems are expected to provide adaptive selection of specific FTT which matches given application and input data nature best. This is important for modeling and adaptive data processing especially [2]. Among possible FTT adaptive selection methods, neural networks usage is of high interest [1, 3, 4]. One-layer linear network with N neurons, each having N inputs may be trained to perform N-point FTT [3]. This procedure will require $O(N^2)$ weighting coefficients selection. An algorithmic approach [3] to the fast neural network synthesis allows to reduce this value to $O(N \log_2 N^2)$ and even to $O(N)$.

In [3] parallel computing system for adaptive execution of the fast trigonometric transformations is proposed. The system consists of transformation execution unit (TEU, ASIC-processor) and transformations synthesis unit (TSU, neural network). Here transformation synthesis task comes to optimal transformation coefficients selection. In given work the structure of TSU hardware implementation is offered. The new U-transformation algorithm [2], with simplified structure, is implemented.

2. UNIFIED FTT HARDWARE

FTT execution system may be implemented in hardware using one of two major approaches. The first one presupposes configurable hardware usage – digital signal processors (SP) or field programmable gate arrays (FPGA). In that case several SP programs or FPGA configurations, each implementing one FTT, are stored in memory and loaded on demand. Evidently this increases hardware costs and design time. The second approach is unified algorithms usage [2, 3]. Empirically determined characteristics of both approaches are given in Table 1. Unified algorithms [2] commensurable to traditional FTT algorithms by performance are used. The following notation is implied:

k – number of transformations implemented by the system;

M_{prog} – average configuration/program storage capacity per one FTT (using the first method);

M_{coef} – average coefficients storage capacity per one FTT (using the first method);

t_{proj} – average system design time per one FTT (using the first method).

Table 1. Characteristics of universal FTT systems design methods

Specification	Configured hardware	Unified algorithms
Configuration/program storage capacity	kM_{prog}	$\leq 2M_{prog}$
Coefficients storage capacity	kM_{coef}	$\leq 3M_{coef}$
System design time	kt_{proj}	$\leq 2t_{proj}$
Single-cycle change of executed FTT	*on SP only*	*on SP and FPGA*
ASIC implementation possibility	*No*	*Yes*

As may be seen, for $k \geq 2$ the second approach reduces hardware costs and system design time while providing several specific advantages. It makes possible ASIC system implementation, which may be required for high-performance or mass production applications. Computational costs of transformation switching for FPGA-based systems are reduced also. Single-cycle change of executed FTT is of high importance for continuous data streams processing. The first approach does not provide single-cycle FTT switching on FPGA, as existing devices reconfiguration time is much longer. Thus it may be said that unified algorithms usage is the most efficient approach to FTT systems design.

3. PARALLEL COMPUTING SYSTEM FOR ADAPTIVE TRANSFORMATION EXECUTION

Offered in [3] system (see fig. 1) is based on universal structurally homogenous fast trigonometric transformations algorithm (UFTT algorithm). UFTT algorithm manipulates real data sequences and implements fast Fourier, Hartley, cosine and sine transformations [3]. Furthermore it allows changeover to other transformations execution by proper coefficients selection. The last task is performed by TSU — fast neural network based on UFTT algorithm. TSU may be implemented both in software and hardware. Software implementation has the following benefits:

- hardware costs reduction (or possibility of larger transformation length implementation on the same device);
- training and result estimation algorithm may be modified easily.

Advantages of hardware TSU implementation are:

- higher network training performance;
- CPU load reduction (if working as a part of general-purpose computer);
- suitable for embedded systems.

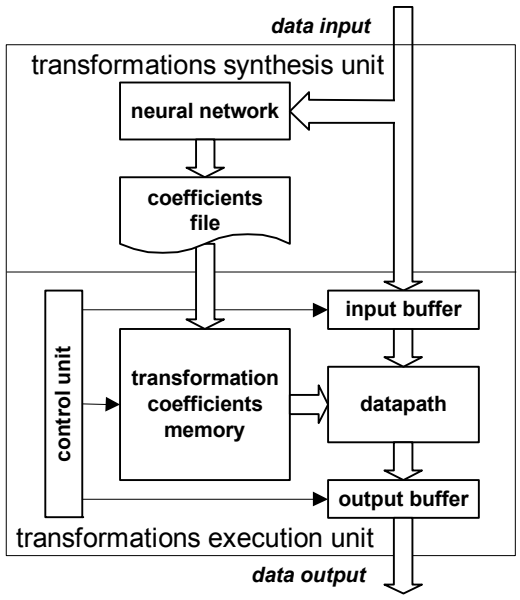

Fig. 1 - Parallel computing system for adaptive execution of fast trigonometric transformations

Thus hardware TSU implementation makes sense for stand-alone embedded systems with stringent performance and power consumption requirements, weight and dimension restrictions.

4. U-ALGORITHM FOR GENERALIZED TRANSFORMATIONS EXECUTION

U–transformation [2] is generalized FTT algorithm, a progressive advance of UFTT algorithm. It expands basic system transformations set with Walsh-Hadamar, Haar and Vilenkin-Chrestenson transformations additionally.

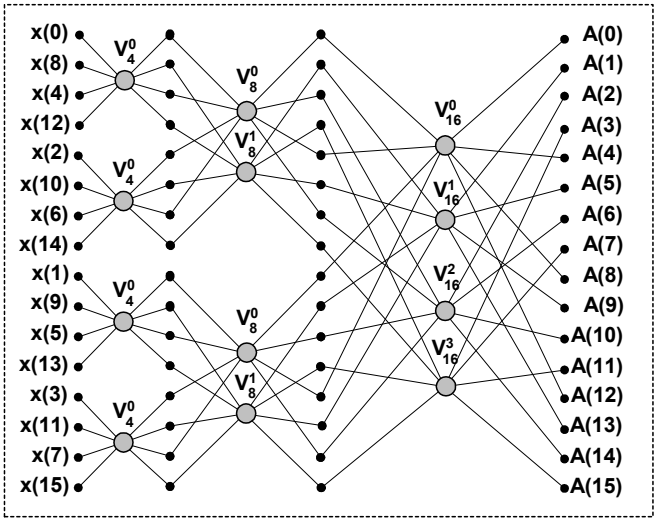

Fig. 2 - U-transformation directed graph for N=16

N-point U-transformation is determined as two equivalent $N/2$-point transformations and their join stage (see fig. 2). The latter one consists of $N/4$ 4-

point primitive operations (see fig. 3, a). PO outputs b_k are determined by equation (1).

$$b_k = \sum_{n=0}^{3} a_n V_N^p(n,k), k = 0,1,2,3 \qquad (1)$$

Here $V_N^p(n,k)$ is primitive operation number p coefficients matrix in N-point join stage; a_n – PO inputs array. N-point U-transformation includes $m=log_2N-1$ stages and $Nm/4$ PO. The structure of algorithm is simplified, comparative to UFTT, as data transposition block is eliminated [2].

Algorithmic approach to the fast neuronet synthesis presupposes substitution of every n-point fast algorithm PO by n neurons, each having n inputs. For U-transformation algorithm n=4 (see fig. 3).

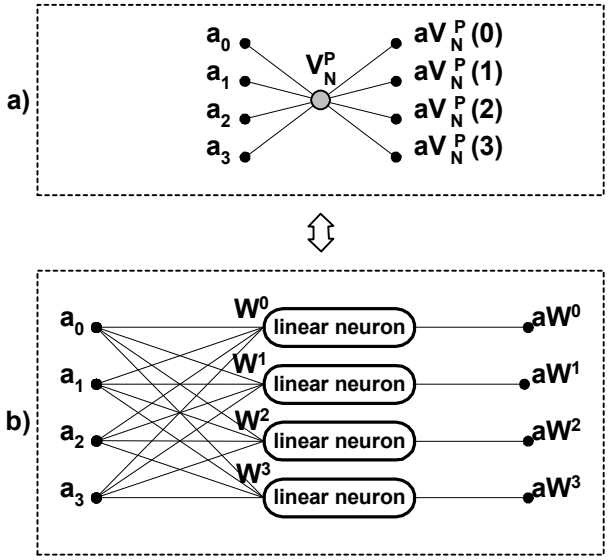

Fig. 3 - Four linear neurons (b) substituting algorithm primitive operation (a)

Four-input linear neuron excitation function y is a sum of products of neuron input signals x and corresponding weighting coefficients w (2). As evident equation (2) is special case of (1).

$$y = \sum_{i=0}^{3} x_i w_i \qquad (2)$$

The most common neuronet training algorithm is back-propagation [4]. Nowadays it is used in almost 80% of DSP neurosystems [5]. It describes output layer neuron error function δ^m as a difference between neuron excitation function y and given reference result value z (3).

$$\delta^m = z - y \qquad (3)$$

Inner layer (layers $k=1, 2, ..., m-1$) neuron error function δ^k depends on the weighting coefficients and error functions of layer $k+1$ neurons, connected with given neuron output (4).

$$\delta^k = \sum_{i=0}^{3} \delta_i^{k+1} w_i^{k+1} \qquad (4)$$

Weighting coefficient correction is governed by equation (5).

$$w_i' = w_i + \eta\, \delta\, x_i, \quad i = 0, 1, 2, 3 \qquad (5)$$

Here w and w' – current and the following value of weighting coefficient; η is training rate coefficient, $0<\eta<1$.

The structure of acquired network and U-transformation graph structure are the same. N-point FTT implementation requires Nm neurons and $4Nm$ weighting coefficients. In fact, neuron weighting coefficients play a part of FTT algorithm PO weighting coefficients here. Synthesized network characteristics and characteristics of one-layer linear network for FTT [3] are given in table 2.

Table 2. FTT neural networks characteristics

Specification	Linear network	U-network
Number of network layers	*1*	log_2N-1
Number of neurons	*N*	$N(log_2N-1)$
Number of inputs per neuron	*N*	*4*
Number of network weighting coefficients	N^2	$4N(log_2N-1)$

5. NEUROPROCESSING HARDWARE

Major advantages of neurosystems are massive parallelism and synthesis methods invariance to network dimensions and dimensions of input data [5, 6, 7]. Nowadays the following approaches are used for neurosystems with massive parallelism design [5]:

- general-purpose processors cascading;
- parallel processors usage (digital signal processors, ASICs, FPGAs);
- dedicated processors usage (bit processors, neurochips).

Neural network in hardware is a dedicated computer with SIMD architecture (single instruction – multiple data). In such a system processing element (PE) implements neuron functions. Programming here comes to weighting coefficients selection [6].

Distinctive feature of proposed network is *4*-input neurons usage. Evidently implementation of *4*-input neuron demands fewer hardware resources and much less training time, comparative to *N*-input neuron. On the other hand, most of known neurochips are not highly suitable for given network realization, as their architecture is optimized for multi-input neurons [5]. Thus optimal decision seems to be network realization on ASIC or FPGA.

Hardware implementation presupposes dividing the network into operational unit and storage unit. Storage unit capacity is determined by the network dimensions. Each neuron is represented in the memory by the structure shown on fig. 4. Here $w_0, \ldots w_3$ are weighting coefficients, y — neuron excitation function value, δ — neuron error function value. N_w, N_y and N_δ — weighting coefficients, excitation function and error function width correspondingly.

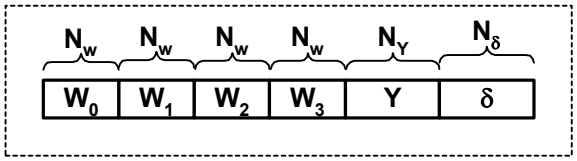

Fig. 4 - Data structure representing neuron in the memory

The Network for *N*-point transformation synthesis includes log_2N-1 layers and each layer consists of *N* neurons [3]. Implementation of such a network requires *M* memory bits, where *M* is calculated by the equation (6).

$$M = N(\log_2 N - 1)(4N_w + N_y + N_\delta) \qquad (6)$$

Operational unit may be PE array or matrix, the size of which is determined by available hardware resources and system performance requirements.

Each PE implements four neurons and may work in two modes. Operative mode implements equation (2), while correction mode implements equations (3, 4, 5). Each neuron requires 4 multiplications and 3 additions in operative mode.

Correction mode requires 5 multiplications and 4 additions per inner layer neuron and 1 addition per output layer neuron. Notice that the magnitude of η is limited here to values 2^{-j}, where $j = 0, 1, 2, \ldots$ This allows logical shift usage as a substitute for multiplication by η.

The most common operational unit design method used in DSP is algorithm graph mapping into hardware. Since network training is iterative process, and number of neurons in all of network layers is the same, mapping of one network layer into PE array is advisable.

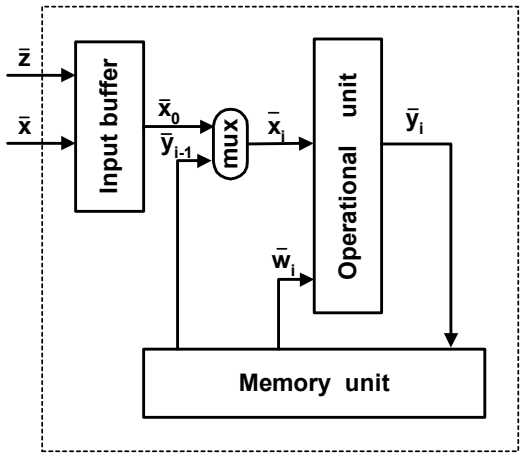

Fig. 5 - Transformation synthesis network (operative mode).

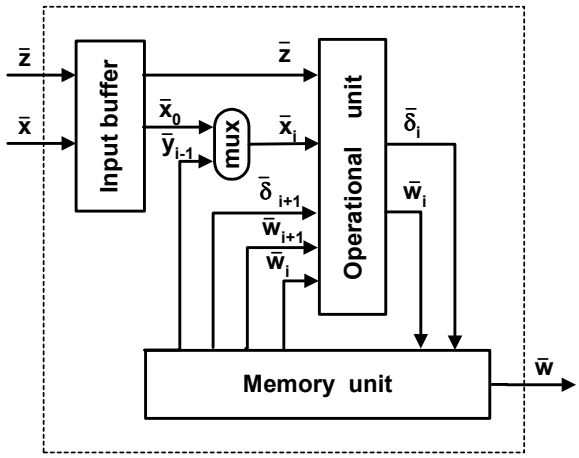

Fig. 6 - Transformation synthesis network (correction mode).

In that case operational unit for *N*-point transformation may include $k=1,2, \ldots, N/4$ PE. All of log_2N-1 layers will be processed sequentially in operative mode. Correction mode provides network training in the reverse layer sequence. The structure of network in operative and correction modes is shown on fig. 5 and fig. 6 correspondingly.

Using known methods network could be trained to perform specific FTT or to synthesize a new one, which matches given application better [3]. Furthermore network may be used for the synthesis of transformation with prescribed characteristics. The choice of appropriate FTT depends on desired appearance of the signal in the frequency domain usually. For example in data compression transformations concentrating signal energy in narrow frequency band are preferred. Developed network may be used to synthesize transformation, which yields spectral signal image of prescribed form (constant, linear, quadratic, etc) for given type of the signal. That will be done if the reference signal for network training matches desired signal accurate to constant [3].

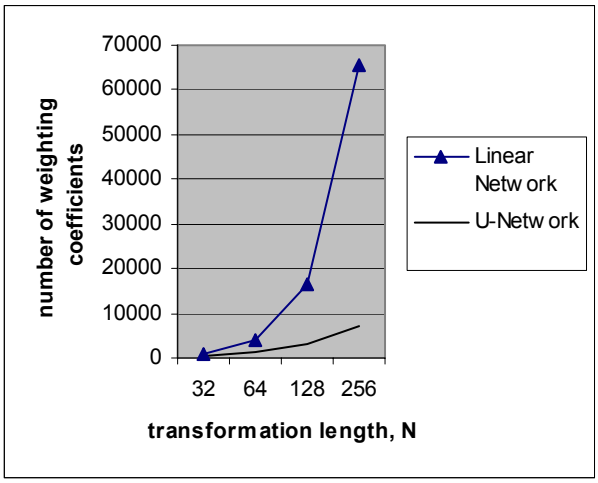

Fig. 7 - Comparison of the weighting coefficients number in modeled networks

Usually network training time is proportional to the number of weighting coefficients. Comparison of examined networks weighting coefficients number for standard transformations length is given in fig. 7. As evident, higher transformation length results in greater training time reduction. For example U-network for 256-point FTT provides almost tenfold training time reduction comparative to one-layer linear network.

6. CONCLUSION

Developed fast hardware neural network implements fast Fourier, Hartley, Walsh-Hadamar, Haar, Vilenkin-Chrestenson, cosine and sine transformations of the real data sequences. It may be used for new, optimal by some given criteria, transformation synthesis also. The network is appointed for FPGA or ASIC implementation and is supposed to be used as a transformation synthesis unit in adaptive DSP system.

7. REFERENCES

[1] V. Grytsyk. Information-analytic systems based on neural network technologies and structures (in Ukrainian). *Lviv Polytechnic National University bulletin "Computer engineering and information technologies"*. Lviv, 2000. - № 392. pp. 32–35.

[2] M. Yatsymirskyy. Discrete orthogonal transformations modeling (in Ukrainian). *Proceedings of Ukrainian-Polish symposium MIMUS`2002*. Lviv-Brjuhovychi, 2002. pp. 195-199.

[3] M. Yatsymirskyy, R. Liskevych, O. Liskevych. Parallel computing system for adaptive execution of the fast trigonometric transformations. *Proceedings of Ukrainian-Polish symposium MIMUS`2002*. Lviv-Brjuhovychi, 2002. pp. 131-136.

[4] R. Tkachenko. Neural networks and the elements of adaptive systems (in Ukrainian). *Lectures abstract. Lviv Polytechnic National University*. Lviv. p. 104

[5] V. Shahnov, A. Vlasov, A. Kuznetsov, Ju. Poliakov. Neurocomputers. Architecture and implementation (in Russian). *ChipNews Journal*, 2000, №№ 6-10.

[6] A. Dorogov, A. Alekseev, D. Butorin. Neural networks with the structure of fast algorithm (in Russian). *Proceedings of the VI All-Russian workshop "Neuroinformatics and its applications"*. Krasnoyarsk. 1998. p. 53.

[7] U. Lisovik, O. Lipchanskiy. The implementation of the neural network for the classification problem. *International Scientific Journal of Computing*, Vol. 3, Issue 2. Ternopil 2004.

Oleh Liskevych. *Was born in 1980. In 2001 graduated from the Lviv Polytechnic National University. Master of Computer Engineering. Currently employed as Junior Research Engineer of the State Scientific and Research Institute of Information Infrastructure (Lviv, Ukraine). Scientific interests: unified algorithms of trigonometric transformations and their hardware description language (HDL) models; fast neural networks. Author of 12 papers.*

Mykhaylo Yatsymirskyy. *Was born in 1955. In 1978 graduated from the Faculty of Mathematics of the Lviv University with M. Sc. degree. Received his Ph. D. in 1990 and the D. Sc. degree (habilitation) in 1998. Professor since 2002. In 1976 began working for the Institute of Radiotechnology in Lviv, first as independent developer-engineer, mathematician-engineer, then research collaborator and finally the head of the Institute. Since 1995 was employed as assistant professor at the department of Computer Techniques and Information Technology of the Lviv Polytechnic National University. Since 1999 - associate professor of the Technical University of Lodz, Poland. Scientific interests: methods, algorithms and models of signal and image processing, technical and medical diagnostics. Author of 120 papers (including one monograph and 8 patents). Supervisor of 5 doctoral dissertations.*