UDC 004.032.026

P. Tymoshchuk, S. Shatnyi
L'viv Polytechnic National University, CAD Department

A HARDWARE IMPLEMENTATION OF NEURAL CIRCUIT OF MAXIMAL/MINIMAL VALUE DISCRETE-TIME SIGNAL IDENTIFICATION

Подано апаратну реалізацію на основі програмованої користувачем вентильної матриці (ПКВМ) нейронної схеми, призначеної для ідентифікації K максимальних за значенями серед N невідомих дискретизованих сигналів, де $1 \le K < N$. Схема має низьку обчислювальну складність і складність схемотехнічної реалізації, високу швидкість опрацювання сигналів, здатністю обробляти сигнали з довільного скінченного діапазону, властивість збереження впорядкованості сигналів, а також відсутність потреби скидання і необхідної для цього схеми, що додатково підвищує швидкість опрацювання сигналів. Описано апаратну реалізацію схеми на основі ПКВМ. Пояснено структуру ПКВМ, а також її VHDL кодування. Наведено приклад моделювання, який демонструє ефективність схеми.

Ключові слова: нейронна схема опрацювання дискретизованих сигналів, реконфігурована обчислювальна архітектура, мова опису апаратного забезпечення, ВІС-технологія.

A hardware implementation in FPGA based reconfigurable computing architecture of discrete-time neural circuit that is capable of identifying the K largest/smallest of any unknown finite value N distinct inputs, where $1 \le K < N$ is presented. The circuit has low computational and hardware implementation complexity, high speed of signal processing, it is capable to process signals of any finite range, possesses signal order preserving property and does not require resetting and corresponding supervisory circuit that increases a speed of signal processing. The hardware implementation based on the results of mathematical modeling KWTA Neural Network with the FPGA-based reconfigurable computing architecture has been described. The issues of using hardware blocks combining VHDL coding have been discussed. Simulation example demonstrating the circuit performance is presented.

Key words: discrete-time neural circuit, K-winners-take-all property, FPGA hardware implementation, reconfigurable computing architecture, hardware description language, VLSI technology.

1. Introduction

Neural networks of largest/smallest signal identification are known to select K largest from N inputs, where $1 \le K < N$. When K is equal to unity, the network can distinguish the maximum/minimum from a set of N inputs [1–3]. Selection of K largest elements from a data set of N real numbers is a key task in decision making, pattern recognition, associative memories and competitive learning networks [4], [5]. Tasks of such type are naturally met in classification problems and applied for the neural network classifier development, for problem solving of pattern recognition and pattern classification [6]. Such networks are applied in telecommunications, particularly to control data packet switches [7]. Such operation has important applications in machine learning, such as k nearest neighbours classification, k-means clustering, etc. [8]. These networks are used in machine learning, in mobile robot navigation, and in feature extraction [9, 10]. Such mechanisms are applied for modeling cognitive phenomena and spiking neural networks [11], [12]. Different kinds of neural networks have been proposed to solve the above indicated problem [1–5, 8, 9, 13–23]. Hardware implementation of the neural networks of largest/smallest signal identification can be found, for instance, in [24]. Comparing with an analogue implementation, a digital hardware is more computationally precise and reliable as long as the requirements for the size and power efficiency are not high.

In this paper, a hardware implementation of FPGA based reconfigurable computing architecture of discrete-time neural circuit built in one-layer competitive architecture performing the dynamic shifting of input signals to obtain K winners is presented. The circuit that is composed of N feedforward neurons and one feedback hardlimiting neuron used to determine the desired shift of inputs is implemented in a digital hardware by adders, digital integrator, switches and external sources of voltage or current, which are appropriate for real time signal processing using VLSI technologies [23, 24]. The architecture has low computational and hardware implementation complexity, high speed of signal processing, it is capable for signal processing of any finite range and possesses signal order preserving property. The circuit does not require resetting and corresponding supervisory circuit that simplifies the hardware and increases a speed of signal processing.

2. A design of the circuit hardware implementation

A functional block diagram of discrete-time dynamical neural circuit of largest/smallest signal identification presented in [23, 24] is shown in Fig. 2, where $x^{(k)} = x((k-1)\tau)$ is an input signal dynamic shift at the discrete-time moment $t^{(k)} = (k-1)\tau$. The diagram consists of blocks of discrete-time summing $\sum$, multiplication $\times$, integration z^{-1}, signum function sgn (hardlimiter), external sources of constant signals $A, 2K, N, x^{(1)}$ and external source of controlled signal α^k. Thus, the circuit can be implemented in a modern hardware using such traditional digital components as adders, hard-limiting quantizers (switches), digital integrator, and external sources of voltage or current.

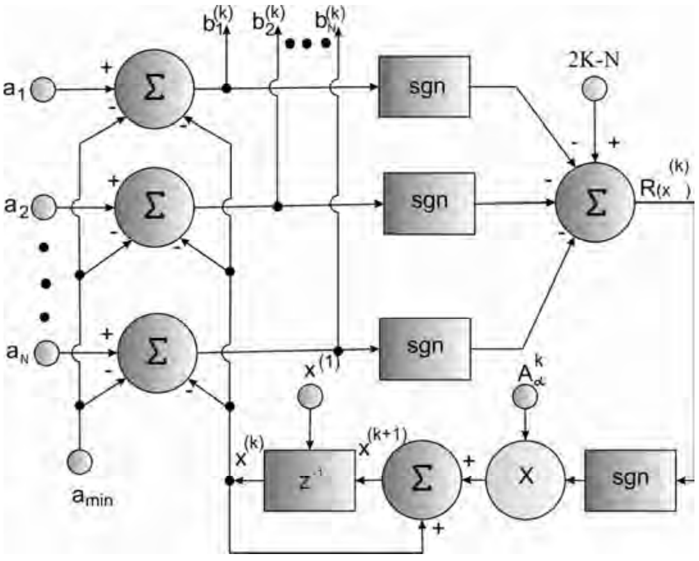


Fig. 1. Functional block diagram of discrete-time neural circuit presented in [1, 2]

Note, that output signals of N blocks sgn could be also used as outputs b of the circuit. However, in this case the only number K of winners from N inputs would be determined. No information concerning ordering of inputs which could be used further, for instance, for solving problems of sorting, classification, clustering, etc. would be available.

As one can see, from hardware implementation complexity point of view, the circuit for processing input signals located in range $a \in (0,1)$ should have one multiplier, $N+2$ adders, N+1 hardlimiting quantizers, one digital integrator, 3 external sources of constant signals and one external source of variable signal. Note that in particular case of replacing α^k with α an external source of variable signal is substituted by a source of constant which simplifies the circuit. For comparison, one of the most simple and fast competitive network proposed in [16] for processing such inputs requires two multipliers, $N+4$ summers,

$N+2$ hardlimiting quantizers, one digital integrator, 5 external sources of constant signals and one external source of variable signal. One of the most simple continuous-time competitor from [20] needs N amplifiers, N+1 adders, one integrator, N hardlimiting quantizers, and 2 external sources of constant signals. Thus, the circuit presented in Fig. 1 has less hardware implementation complexity than other analogs.

From a computational complexity point of view the circuit in the case of input signals, located in range $a \in (0,1)$ requires a consecutive performing two multiplications, $N+4$ additions/subtractions, two logic hardlimiting operations and one integrating operation on each iteration. If α^k is replaced with α, then the circuit requires only one multiplication instead of two. Competitive network from [16] requires in this case a consecutive fulfilling of two multiplications, $3N+2$ additions/subtractions, two signum function operations and one integrating operation for each iteration. Therefore the computational time of the circuit is less than that of other competitors [2, 13].

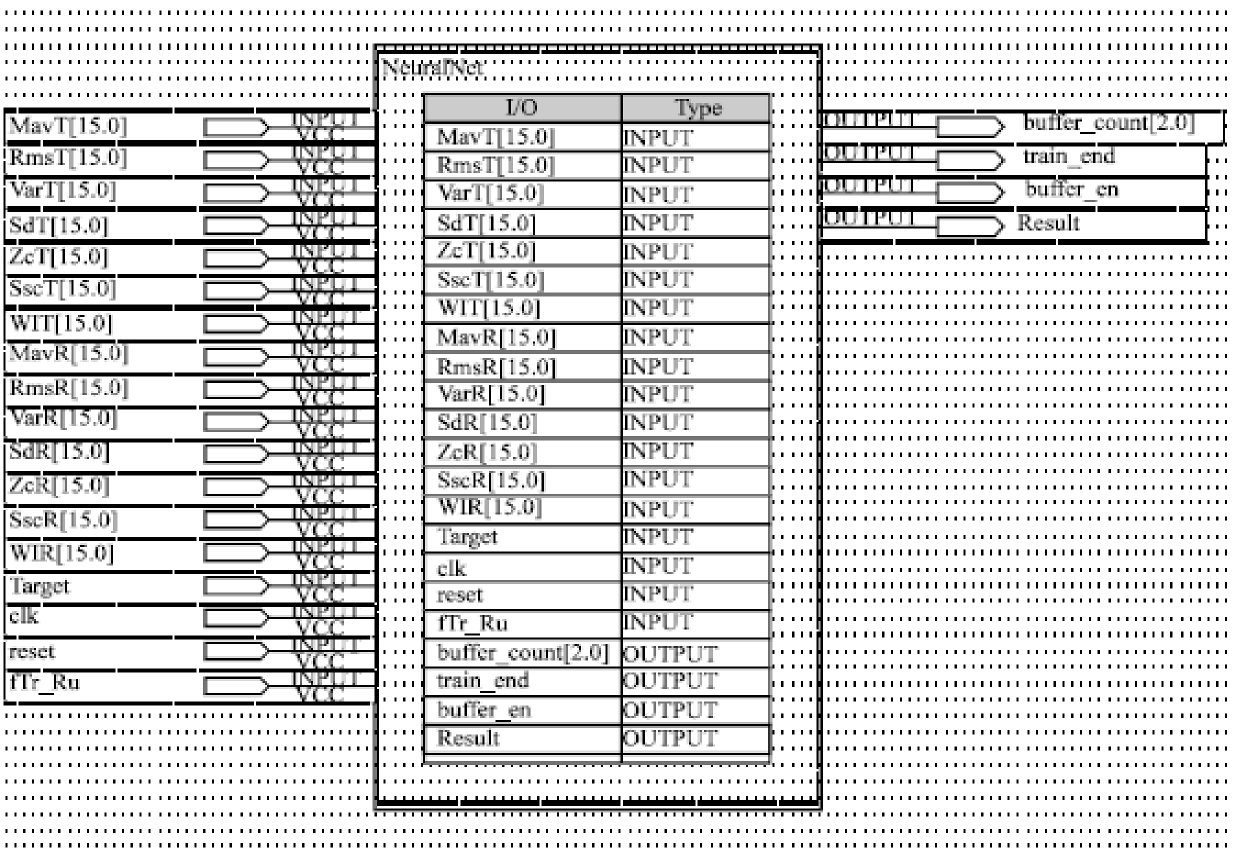
A resolution ability of the circuit is infinite theoretically, i.e. if inputs are distinct, then the circuit can always identify them. Practical resolution ability can be limited by precision of the circuit hardware implementation. Since proposed circuit can process correctly any distinct inputs, therefore the accuracy performance of the circuit is the same as that of other analogs.

A functioning of the circuit is independent on initial condition $x^{(1)}$ which can accept arbitrary value in range [0,A]. Therefore, the circuit does not require periodical resetting $x^{(1)}$ for repetitive signal processing, additional supervisory circuit for resetting, and spend additional processing time on this operation [8, 16, 18, 19]. This allows to simplify the hardware and increase a speed of signal processing that is important for real time operation. An important advantage of the circuit is that in contrast to other analogs [8, 16, 20] it possesses an order preserving property of input signals.

An implementation of many recurrent neural networks is usually simulated using software, but the processing speed in this case can be not fast enough to meet demands of real time. Therefore, microprocessors and digital signal processing are not suitable for parallel designs. Let us implement the neural circuit presented in [23, 24] in up-to-date digital hardware. The digital implementation comparatively to continuous-time analogs demonstrate a more high precision of signal processing, better repeatability, lower noise sensitivity, better testability, higher flexibility and reliability, as well as compatibility with other types of preprocessors [25]. The up-to-date digital neural network hardware implementations can be classified as FPGA-based implementations, DSP-based implementations, and ASIC-based implementations. Since DSP-based implementation is sequential, it does not preserve the parallel architecture of the circuit functional block-diagram presented in Fig. 1. ASIC implementation can be used for the circuit hardware realization, although it does not offer re-configurability by the user in order, for instance, to improve the circuit performance. Due to the relatively high capacity, high density, short design cycle and short time to market when using EDA tools, FPGA becomes the most applicable microelectronic technology in many recent applications [26]. The size and speed evaluation of FPGA reveal its low cost in terms of logic and memory [27–30]. Moreover, the FPGA implementation achieves a comparable accuracy with the traditional solutions based on general-purpose computers. To implement the circuit in a hardware, the FPGA based reconfigurable computing architecture is quite suitable because the parallel structure of FPGA matches the topology of the circuit and offers flexibility in reconfiguration. An FPGA as an implementation hardware can be chosen because it combines the reprogrammability advantage of general purpose processors with the parallel processing and speed advantages of customer hardware. Such computational characteristics of the circuit as modularity and dynamic adaptation can be realized in FPGA hardware. The circuit architecture and training algorithm can be implemented on a FPGA chip performing an on-line training. Therefore, using FPGA, the circuit can be implemented through parallel computing in a real-time hand-tracking system [31, 32].

3. An architecture of the circuit hardware implementation using FPGA

Let us use FPGA chip Altera Cyclone III EP3C16 of MultiCore and Softprocessor technology. We use HDL Language since with it help we can develop input\outputs of the system, logical blocks, arithmetical blocks etc. In simplified form, the description of the components in VHDL consists of an interface specification and an architecture specification [23, 33, 34]. The VHDL model for the design is designed with three modules in one package. The top entity module is the main module to manage and control its other components (Fig. 2). Simulation of the VHDL code has to be carried out for two reasons after describing a digital system in VHDL. First, we need to verify whether the VHDL code correctly implements the intended design. Second, it is necessary to verify if the design meets its specifications. The simulation is used to test the VHDL code by writing test bench models [34]. Top entity consists of description of existing system inputs and outputs that are galvanically connected to the appropriate port circuits of the FPGA IC. According to the description presented in Fig. 2, defined maximum and minimum level signals are being processed in the working mode. In case of processing data array values there should be described the procedures for booking the corresponding memory cells which in turn should be clocking by the common clock source. The proposed design consists of neuron architecture, activation signed function problem solving and network architecture. As the data flow inside the system require parallel computational, multiplier/accumulator architecture has been selected for implementation as shown in Fig. 3. The architecture gets the input serially, multiplies them with the corresponding weight value and accumulates their sum in a shift register. The processes are synchronized by clock signal which is the oscillator source for the whole circuit design. Implementation of fully parallel network is possible with use of FPGAs. Such the network is fast but inflexible because of a number of primitive arithmetic blocks must be equal to the number of connections in the network.



I/O	Type
MavT[15.0]	INPUT
RmsT[15.0]	INPUT
VarT[15.0]	INPUT
SdT[15.0]	INPUT
ZcT[15.0]	INPUT
SscT[15.0]	INPUT
WIT[15.0]	INPUT
MavR[15.0]	INPUT
RmsR[15.0]	INPUT
VarR[15.0]	INPUT
SdR[15.0]	INPUT
ZcR[15.0]	INPUT
SscR[15.0]	INPUT
WIR[15.0]	INPUT
Target	INPUT
clk	INPUT
reset	INPUT
fTr_Ru	INPUT
buffer_count[2.0]	OUTPUT
train_end	OUTPUT
buffer_en	OUTPUT
Result	OUTPUT

Fig. 2. Top entity for VHDL model of the network

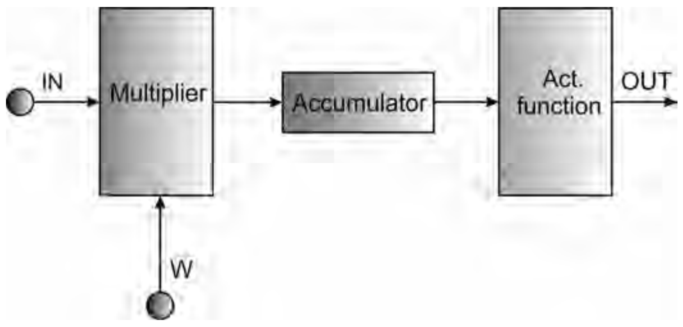
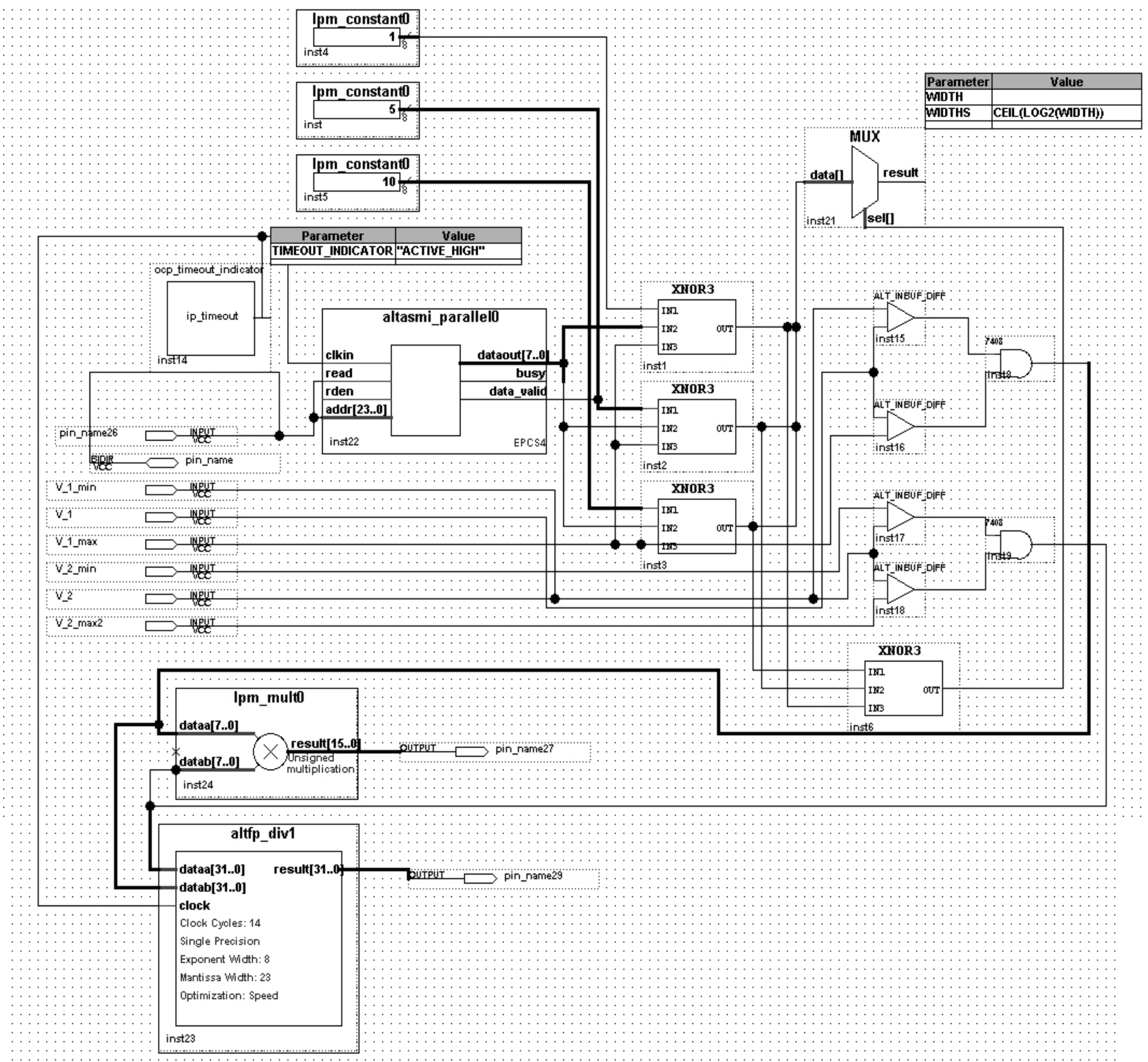


Fig. 3. Multiplier/accumulator architecture

All the modules for the designed model are compiled for the Analysis and Synthesis check, Place and Route (Fitter) check, Assembler check and Classic Timer Analyzer check by using the Quartus II software. Simulation is the most important part for VHDL based hardware modeling flow. It is also one of the most difficult part not for the mechanism of the processes but because it needs to evaluate all possible processes and failure modes and requires to test them carefully. Fig. 4 shows a design of iterative KWTA network based on Fig.1. The design includes schematic description of the basic elements used in the project. For processing descrete-time sampled signals eight-bit unidirectional interface connected to external analog-to-digital converter is provided. Since the magnitude of the signal level should not exceed the level of applied voltage, there is realized a voltage limiter that runs the algorithm of voltage divider. The design contains summers, multipliers, integrators, switchers and external signal sources which are described in top entity of the project. Data transaction are implemented by using serial interface protocol RS-232. The main restrictions are the bandwidth of the channel and error corrections. To eliminate this disadvantage specialized interfaces for data exchanging can be used as well as EEPROM FLASH-storage for preservation and archiving of transitional and final results of the scheme can be employed. For this purpose external controller FLASH-drive has to be implemented and included to the overall scheme of operation. In this case, all intermediate data will be stored in memory during the experiment with minimal time latency for later analysis. A limitation of this method is impossibility of operation scheme configuration, analysis of the results in real time and operation monitoring of the system.



Parameter	Value
WIDTH	
WIDTHS	CEIL(LOG2(WIDTH))

Parameter	Value
TIMEOUT_INDICATOR	"ACTIVE_HIGH"

Fig.4. Hardware-description design of functional block diagram of discrete-time KWTA neural circuit presented in [21, 22]

4. Simulation results of the circuit

Let us consider concrete example adopted from [21, 22] with corresponding simulations which demonstrate signal processing by described above FPGA based hardware implementation of discrete-time neural circuit.

Example. Let it be necessary to identify the four largest inputs, that is K=2, from vector $a=[-1,0.7,-0.3,-0.8,0.2]$, i.e. $N=5$ having been used the FPGA based hardware implementation of neural circuit described in [21, 22]. Let us take for this circuit $A_{\min}=-1$, $A=2$, an initial condition $x^{(1)}=A$ and a decaying coefficient $\alpha=0.7$. Let us determine the discrete-time trajectories of the shift $x^{(k)}$ and the outputs $b_i^{(k)}, i=1,2,3,4,5$. Such trajectories in normalized units are shown Fig. 3. As one can see, in the steady state, the components $b_2>0,\ b_5>0$ correspond to the two largest components of vector a – the winners and the remaining ones, $b_1<0, b_3<0, b_4<0$ correspond to the losers. A convergence of the search process to a steady state is achieved in $m=6$ iterations.

Analyzing the data, we can conclude that the results of numerical experiments confirm mathematical calculations and modeling presented in [21] but it turned out that while collecting the required information from technical device identifying the number of iterations to convergence of search process fails, due to low bandwidth to data exchange process between FPGA-based system and PC with running software on it.

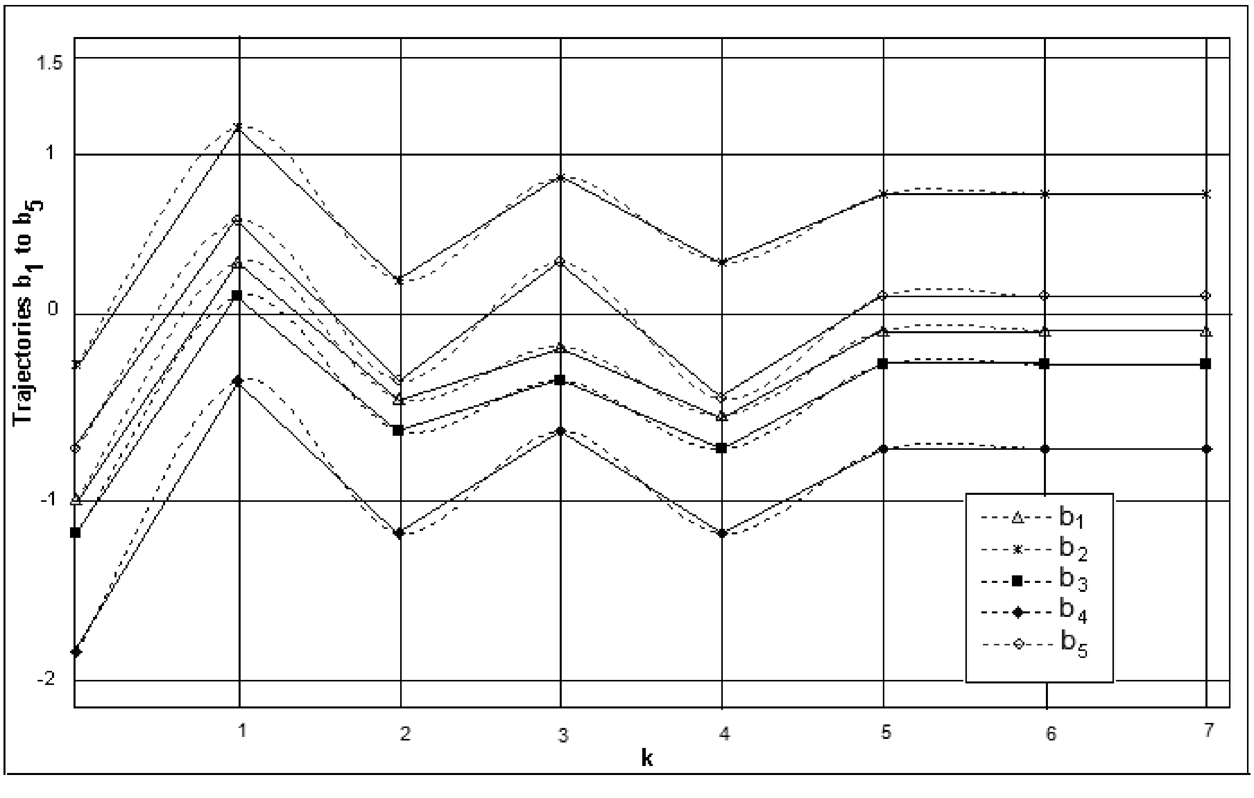


Fig. 5. The trajectories of the outputs $b_i^{(k)}, i=1,2,3,4,5$ *of the FPGA based hardware implementation of neural circuit presented in [21, 22]*

Conclusions

A hardware implementation by FPGA of discrete-time neural circuit of largest/smallest signal identification designed based on input signal dynamic shifting is described. The circuit is stable and convergent to correct operation in finite number of iterations. The hardware implementation complexity of the circuit is less than that of other competitive networks. The circuit is capable to process any finite value distinct signals and possesses signal order preserving property. Periodical resetting and corresponding supervisory circuit for repetitive signal processing are not necessary. Described circuit is better to use if it is necessary to have simple circuit with a high resolution ability, high speed of signal processing of wide range, independency on initial conditions and signal ordering preserving property. The described circuit that is implemented based on serial integral circuits is suitable for various applications.

References

1. Lippmann R. P. A comparison of Hamming and Hopfield neural nets for pattern classification / R. P. Lippmann, B. Gold, and M. L. Malpass // MIT Lincoln Laboratory Technical report TR-769 (1987) 1–37. 2. Majani E., Erlanson R. and Abu-Mostafa Y. On the K *-winners-take-all network, in: Advances in Neural Information Process. Syst. D. S. Touretzky, Vol. 1 (Kaufmann, San Mateo, 1989) 634–642. 3. Tymoshchuk P. A winner-take-all circuit using neural networks as building blocks / P. Tymoshchuk and E. Kaszkurewicz // Neurocomputing 64 (2005) 375–396. 4. Urahama K. K-Winner-take-all circuit with 0(n) complexity / K. Urahama and T. Nagao // IEEE Trans. on Neural Networks 6 (1995) 776–778. 5. Yen J. C. A new* k *-Winners-take all neural network and its array architecture / J. C. Yen, J. I. Guo, and H.-C. Chen // IEEE Trans. on Neural Networks 9 (1998) 901–912. 6. Kwon T. M. and Zervakis M. A parallel sorting network without comparators: A neural-network approach, in: Proc. Int. Joint Conf. on Neural Networks, Vol. 1 (1992) 701–706. 7. Bihn L. N. A neural-network contention controller for packet switching networks / L. N. Bihn and H. C. Chong // IEEE Trans. on Neural Networks 6 (1995) 1402–1410. 8. Liu S. A simplified dual neural network for quadratic programming with its KWTA application / S. Liu and J. Wang // IEEE Trans. on Neural Networks, 17 (2006) 1500–1510. 9. DeSouza G. N. Vision for mobile robot navigation: a survey / G. N. DeSouza and A. C. Zak // IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 24, no. 2, pp. 237–267, Feb. 2002. 10. O'Reilly R. C. Computational explorations in cognitive neuroscience: understanding the mind by simulating the brain / R. C. O'Reilly and Y. Munakata. – Cambridge, MA: MIT Press, 2000. 11. A. Lazar Fading memory and time series prediction in recurrent networks with different forms of plasticity / A. Lazar, G. Pipa, and J. Triesch // Neural Networks, vol. 20, № 3, pp. 312–322, Apr. 2007. 12. Wolfe W. J. K-Winner networks / W. J. Wolfe, D. Mathis, C. Anderson, J. Rothman, M. Gotler, G. Bragy, R. Walker, G. Duane, and G. Alaghband // IEEE Trans. on Neural Networks 2 (1991) 310–315. 13. Perfetti R. On the robust design of k-winners-take-all networks / R. Perfetti // IEEE Trans. on Cir. and Syst.-II: Analog and Digit. Sign. Process., CAS-42 (1995) 55–58. 14. Yen J. C. and Chang S. A new first-* k *-winners neural network, in: Proc. of the ISANN (1997) D-01-D-06. 15. Yang J. F. A Dynamic K-Winners-Take-All Neural Network / J. F. Yang and C. M. Chen // IEEE Trans. on Syst., Man and Cyb. 27 (1997) 523–526. 16. Calvert B. D. Another* K *-winners-take-all analog neural network / B. D. Calvert and C. A. Marinov // IEEE Trans. on Neural Networks 4 (2000) 829–838. 17. Marinov C. A. Performance analysis for a* K *-winners-take-all analog neural network: basic theory / C. A. Marinov and B. D. Calvert // IEEE Trans. on Neural Networks 14 (2003) 766–780. 18. Marinov C. A. Stable computational dynamics for a class of circuits with 0(N) interconnections capable of KWTA and rank extractions / C. A. Marinov and J. J. Hopfield // IEEE Trans. on Cir. and Syst. I: Fundamental Theory and Applications 52 (2005) 949–959. 19. Hu X. An improved dual neural network for solving a class of quadratic programming problems and its k-winners-take-all application / X. Hu and J. Wang // IEEE Trans. on Neural Networks, 19 (2008) 2022–2031. 20. Wang J. Analysis and design of a k-winners-take-all model with a single state variable and the Heaviside step activation function / J. Wang // IEEE Trans. Neural Networks, vol. 21, № 9, pp. 1496–1506, Sept. 2010. 21. Тимощук П. Математична модель нейронної схеми типу "K-Winners-Take-All" обробки дискретизованих сигналів / П. Тимощук // Компютерні системи проектування. Теорія і практика. – 2010. – № 685. – С. 45–50. 22. Тимощук П. Оптимізація нейронної схеми ідентифікації максимальних дискретизованих сигналів / П. Тимощук // Оптимізація виробничих процесів і* *технічний контроль у машинобудуванні та приладобудуванні. – 2010. – № 679. – С. 107–112. 23. Hui-Ya Li Hardware Implementation of k-Winner-Take-All Neural Network with On-Chip Learning / Hui-Ya Li, Chien-Min Ou // 13th IEEE International Conference on Computational Science and Engineering, 2010. 24. Cichocki A. Neural Networks for Optimization and Signal Processing / A. Cichocki and R. Unbehauen. – New York: John Wiley and Sons, 1993. 25. Sahoolizadeh H. A FPGA implementation of neural/wavelet face detection system / Sahoolizadeh H., & Keshavarz A. // Australian Journal of Basic and Applied Sciences, 2010. – 4(3), 379–388. 26. Reyneri L. M. Implementation issues of neuro-fuzzy hardware: going towards HW/SW codesign / Reyneri L. M. // IEEE Transactions on Neural Networks, 2003. – 14(1), 176–194. 27. Muthuramalingam A. Neural network implementation using FPGA: issues and application / Muthuramalingam A., Himavathi S., & Srinivasan E. // International Journal of Information Technology, 2008. – 4:2, 95–101. 28. Krips M. FPGA implementation of a neural network for a real-time hand tracking system / Krips M., Lammert T., & Kummert A. // Proceedings of the 1st IEEE International Workshop on Electronic Design, Test and Applications, 2002. – 29–31, 313–317. 29. Lin C.-J. FPGA implementation of a recurrent neural fuzzy network with on-chip learning for prediction and identification applications / Lin C.-J., & Lee C.-Y. // Journal of Information Science and Engineering, 2010. – 25, 575–589. 30. Zheng M. FPGA implementation of a neural network control system for a helicopter / Zheng M., Tarbouchi M., Bouchard D., & Dunfield J. // Proceedings of the 7th WSEAS International Conference on Neural Networks, 2006. – 7–10. 31. Haitham K. A. Design artificial neural network using FPGA / Haitham K. A., & Esraa Z. M. // IJCSNS International Journal of Computer Science and Network Security, 2010. – 10(8), 88–92. 32. Muthuramalingam A. Neural Network Implementation Using FPGA: issues and Application / A. Muthuramalingam, S. Himavathi, E. Srinivasan // Intarnational Journal of Information Technology, 2008. 33. Ali H. Design Artificial Network Using FPGA / H. Ali, E. Mohammed // IJCSNS International Journal of Computer Science and Network Security. – 2010. – vol. 10, № 8.*