HARDWARE DESCRIPTION OF DIGITAL HOPFIELD NEURAL NETWORKS FOR SOLVING SHORTEST PATH PROBLEM

Hajar Asgari, Yousef S. Kavian[*]

Abstract: The shortest path problem is an important issue in communication networks which is used by many practical routing protocols. The aim of this paper is to present an intelligent model based on Hopfield neural networks (HNNs) for solving shortest path problem and implement that on Field Programmable Gate Arrays (FPGAs) chips. The Cyclone II–EP2C70F896C6 FPGA chip from ALTERA Inc. is considered for hardware implementing and VHDL language is employed for hardware description. The synthesizing results show the proposed architecture of neuron is more efficient than relevant neuron model for chip area utilization and consequently improving the maximum operating frequency and power consumption. The proposed router core is employed to find shortest paths in ring, star and mesh communication networks and the results demonstrate the efficiency and superiority of proposed core.

Key words: *Hopfield neural network, shortest path problem, FPGA implementation, VHDL*

Received: November 17, 2013
Revised and accepted: April 18, 2014

DOI: 10.14311/NNW.2014.24.013

1. Introduction

The key responsibility of the network layer in communication networks is the routing problem [1,2] which is considered as establishing paths from source nodes to destination nodes in connecting demand matrix for different applications. There are different metrics which are considered as the objective of routing protocols and have important effects on the network performance and quality of service. The most common used metric in routing protocols is the shortest-path attempting to find the paths with minimum length. The main idea behind this metric is that using the shortest-path will result in low end-to-end delays and low resource consumptions [3].

The routing problem is a complex and multi-constraint problem which is considered as a NP-hard problem. Therefore intelligent and meta-heuristic optimization

[*]Hajar Asgari, Yousef S. Kavian – Corresponding Author, Electrical Engineering Department, Faculty of Engineering, Shahid Chamran University of Ahvaz, Ahvaz, Iran, E-mail: `y.s.kavian@scu.ac.ir`

approaches and algorithms are employed by routing protocols such as genetic algorithms [4], artificial bee colony [5], ant colonies [6], swarm optimization [7] and neural networks [8] where the Hopfield neural network (HNN) is an interesting approach [9] using energy functions for solving routing problem [10].

The HNNs proposed by Hopfield [11] have been successfully employed for practical issues such as modelling associative content addressable memories [12] for face detecting and pattern recognition and solving NP-hard optimization problems like travelling salesman problem (TSP) [13]. Furthermore, HNN is considered as an interesting optimization tool for modelling and solving routing problems in wireless and optical networks [14–17].

The hardware implementing of routing protocols for designing router and switch devices and employing them in physical layer is an important designing issue. The Application Specific Integrated Circuits (ASICs) and Field Programmable Gate Arrays (FPGAs) chips are two considering implementing environments for digital Hopfield neural networks models and applications. The ASICs are expensive with less flexibility but FPGA chips offer more flexibility by ability of reprogramming [18,19,20]. The FPGAs consist of thousands of logic gates and some configurable logic blocks which make them an appropriate solution for prototyping the digital circuits with dedicated architectures. The introduction of VHSIC Hardware Description Language (VHDL) [21] provided a modelling and simulation environment for fast prototyping digital circuits and systems on FPGA.

In this paper new hardware architecture of Hopfield neural network (HNN) for solving shortest path problem in communication networks is presented where the proposed architecture is suitable for implementing on FPGA chips.

The rest of paper is organized as follows: Section 2 presents mathematical model and concepts of digital Hopfield neural networks. The HNN model of shortest path problem is described in Section 3. The hardware implementing architecture and issues are presented in Section 4. Simulation and synthesizing results are described in Section 5 and finally the paper is concluded in Section 6.

2. Digital Hopfield Neural Network

This section describes the mathematical model of digital Hopfield neural network for implementing on FPGA. The Hopfield neural network (HNN) is considered as a recurrent and single layer neural network where output of each neuron is introduced into the inputs of other neurons in network using feedback schemes, Fig. 1.

The dynamic model of each neuron which is used to update weight and output is described as follows:

$$\frac{dX_j(t)}{dt} = -\frac{X_j(t)}{\tau} + \sum_{i=1}^{N} T_{ij} Y_i(t) + b_j \tag{1}$$

$$Y_i = f(X_i) = \frac{1}{1 - \exp(-a_i X_i)} \tag{2}$$

where $X_j(t)$ is the internal state of j-th neuron and $Y_i(t)$ is the output of i-th neuron in the network and $i \neq j$. In this model, N is the number of neurons in the network, b_j is the bias value of j-th neuron, τ is neural time constant, a_i is the i-th neuron gain and T is the network weight matrix which is a symmetric matrix $(T_{ij} = T_{ji})$.

Furthermore, when the diagonal elements are zero $(T_{ii} = 0)$, and gain of neurons is high $(a_i \to \infty)$ the computational energy of network in stable states is described as follows:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij} Y_i Y_j - \sum_{i=1}^{N} b_i Y_i \tag{3}$$

In terms of the network energy aspect, each neuron in network tends to reduct total energy of network [22, 23], therefore the Equation (1) could be rewritten as follows considering the Equation (3):

$$\frac{dX_j(t)}{dt} = -\frac{X_j(t)}{\tau} - \frac{\partial E}{\partial Y_j} \tag{4}$$

For digital implementing of Hopfield neural network on FPGA, a discrete model of Equation (4) is required which is considered for time step Δt. This Equation is a recursive one and is suitable for digital hardware implementing using memory elements:

$$X_j(k) = X_j(k-1) - \Delta t \times \frac{\partial E}{\partial Y_j} \tag{5}$$

The considered analytical model of digital Hopfield neural network always convergences to stable states where output voltages of all neurons remain constant.

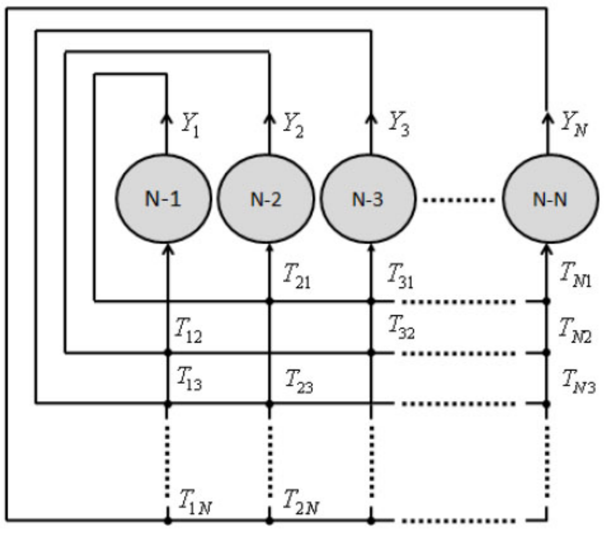

Fig. 1 *Typical Hopfield neural network with N neurons.*

3. Hopfield Neural Network Model of Shortest Path Problem

The routing problem is considered as a complex and multi-constraint problem which is the main task of network layer in communication networks. Solving and modelling routing problems require optimization of some cost functions which are subjected to set of constraints. The cost functions may be considered as energy functions which are suitable for modelling by neural networks to produce solutions through minimizing energy functions [13].

3.1 Mathematical Model of Shortest Path Problem

The network topology is represented as a finite, directed, and weighted graph $G(N, L, W)$ where N is the set of nodes, L is the set of connecting links in the network. The links are denoted by their end nodes, e.g. link $l_{o,e}$ for $o \in N$ as origin and $e \in N$ as ending nodes. The length of each link $l \in L$ is represented by a weighting function $W_l \in W$: $L \to R^+$ where R^+ denotes the set of positive integers. A connection request is specified by $R_{(s,d)}$ where $s, d \in N$ are source and destination nodes and the aim is to find shortest paths from source to destination node.

In shortest path problem, every link has the different cost (length) and a shortest path routing protocol selects the path that minimizes the total cost of data propagation from source to destination.

The path $P_{s,d}$ is considered as a sequence of nodes from source node s to destination node d connected by links;

$$P_{s,d} \equiv \{s, n_i, n_j, \dots, n_k, d\} \equiv \{l_{sn_i}, l_{n_in_j}, \dots, l_{n_kd}\} \tag{6}$$

The cost of each path is the sum of costs of all participating links in the path.

$$C_{P_{s,d}} = C_{l_{sn_i}} + C_{l_{n_in_j}} + \dots + C_{l_{n_kd}} \tag{7}$$

The aim of shortest path problem is to minimize the cost of connecting path between node pair (s, d) for all $s, d \in N$.

$$\text{Minimize } \left\{C_{P_{s,d}}\right\} \quad \forall s, d \in N \tag{8}$$

3.2 Hopfield Neural Network Model of Shortest Path Problem

For modelling shortest path routing problem using HNN, communication network is represented by $N \times N$ adjacency matrix which shows network connectivity where diagonal elements are removed [9]. Each element in the matrix is represented by a neuron which is described by double indices (i, j) where the row subscript i and the column subscript j denote node numbers, and a neuron at location (i, j) is characterized by its output Y_{ij} that becomes 1 when the link from node i to node j is used in the path and 0 otherwise.

The dynamic model of a typical neuron at location (i, j) is described as follows:

$$\frac{dX_{ij}(t)}{dt} = -\frac{X_{ij}(t)}{\tau} + \sum_{k=1}^{N}\sum_{l=1}^{N} T_{ijkl}Y_{kl}(t) + b_{ij} \tag{9}$$

T_{ijkl} is connection weight between neuron at location (i, j) and neuron at $(k, l), b_{ij}$ is biases value of neuron at (i, j), an important matrix that is considered in the modelling is the K matrix where the K_{ij} is 1 when the link from node i to node j doesn't exist in the path. In addition the cost of connecting link from node i to node j is denoted by G_{ij} as a finite real positive number. The cost of not exist links is assumed to be zero. In order to solve the routing problem, using the Hopfield model, first an energy function is defined whose minimization process drives the neural network in to its lowest energy state. The stable state will correspond to the routing solution. In this paper we use the same energy function given in [9]. The energy function is constructed as follows:

$$E = z_1E_1 + z_2E_2 + z_3E_3 + z_4E_4 + z_5E_5 \tag{10}$$

The energy terms are described as follows:

$$E_1 = \frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ [1mm]j\neq i}}{}^{N} G_{ij}Y_{ij} \tag{11}$$

$$E_2 = \frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N} K_{ij}Y_{ij} \tag{12}$$

$$E_3 = \frac{1}{2}\sum_{i=1}^{N}\left[\sum_{\substack{j=1\\ j\neq i}}^{N} Y_{ij} - \sum_{\substack{j=1\\ j\neq i}}^{N} Y_{ji}\right]^2 \tag{13}$$

$$E_4 = \frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N} Y_{ij}(1 - Y_{ij}) \tag{14}$$

$$E_5 = \frac{1}{2}(1 - \mathrm{Y}_{\mathrm{ds}}) \tag{15}$$

In (10) the aim of Z_1 term is to minimize the total cost of a path considering the cost of existing links, the Z_2 term prevents of non-existence links to be included in the path. The Z_3 term makes sure that if a node has been entered it will also be used by a path. The Z_4 term enforces that the state of the neural network converges to a valid route with lowest energy value. The Z_5 term is used to enforce the construction of a path, which must originate from source node s and terminate at destination node d.

The network evolutes the internal activation to reduce the overall energy and neurons update their state where $\partial E/\partial Y$ is calculated directly and is summation of all energy terms. That is a discrete time-step approach for solving the differential Equation that describes the connection between changes in activation and energy terms [24]. The update Equation for a neuron is given by:

$$\frac{dX}{dt} = -\frac{X}{\tau} - \left(z_1 \frac{\partial E_1}{\partial Y} + z_2 \frac{\partial E_2}{\partial Y} + z_3 \frac{\partial E_3}{\partial Y} + z_4 \frac{\partial E_4}{\partial Y} + z_5 \frac{\partial E_5}{\partial Y} \right) \tag{16}$$

The differential equations for updating the neuron at location (i, j) are as follow:

$$\frac{\partial E_1}{\partial Y} = \frac{1}{2} G_{ij}(1 - \delta_{id}\delta_{js}) \tag{17}$$

$$\frac{\partial E_2}{\partial Y} = \frac{1}{2} K_{ij}(1 - \delta_{id}\delta_{js}) \tag{18}$$

$$\frac{\partial E_3}{\partial Y} = \sum_{\substack{k=1 \\ [2mm]k\neq i}}^{N} (Y_{ik} - Y_{ki}) - \sum_{\substack{k=1 \\ [2mm]k\neq j}}^{N} (Y_{jk} - Y_{kj}) \tag{19}$$

$$\frac{\partial E_4}{\partial Y} = \frac{1}{2}(1 - 2Y_{ij}) \tag{20}$$

$$\frac{\partial E_5}{\partial Y} = \frac{1}{2}\delta_{id}\delta_{js} \tag{21}$$

where δ_{ij} is the Kronecker-delta operator and is 1 when $i = j$ and is 0 otherwise. By substituting (16)–(20) in (15) and comparing the corresponding coefficients of resulted Equation with (9) the connection weight and biases of HNN are as follows:

$$T_{ijkl} = z_4\delta_{ik}\delta_{jl} + z_3(\delta_{li} + \delta_{jk} - \delta_{ik} - \delta_{jl}) \tag{22}$$

$$b_{ij} = -\frac{z_1}{2} G_{ij}(1 - \delta_{id}\delta_{js}) - \frac{z_2}{2} K_{ij}(1 - \delta_{id}\delta_{js}) - \frac{z_4}{2} + \frac{z_5}{2}\delta_{id}\delta_{js} \tag{23}$$

4. Hardware Architecture and Implementing Issues

The hardware description and implementation of Hopfield neural network on FPGA for solving shortest path problem involve some important issues which are described in this section.

4.1 Neuron Architecture

The Hopfield neural network model of shortest path problem in a communication network with N nodes is a combination of $N \times (N - 1)$ single neuron blocks. Therefore designing of single neuron block is an important issue of designing Hopfield neural network models. All neurons are synchronous circuits which update their internal states synchronously using (9) or (15). Fig. 2 shows typical hardware architecture of single neuron blocks employed in Hopfield neural network. As shown the ADD and MULTIPLY operations are two main operating blocks which have important impacts on hardware resource utilization, operating frequency and power consumption. Furthermore, four dimensional (4-D) arrays of Hopfield neural network weights, described in (21), require many multipliers and adders for hardware implementation of (9) [25–27]. Implementation of (15) reduces the number of utilized resources that required in hardware implementation of HNN on FPGA.

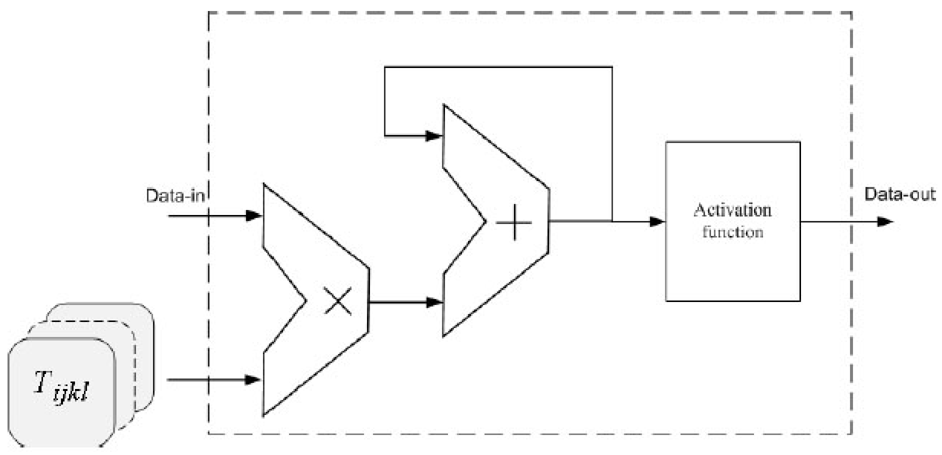

Fig. 2 *Typical hardware architecture of neuron.*

New hardware architecture of neuron is depicted in Fig. 3 where the neuron updates it's output by energy evaluation equation. Input signals of neuron component are initial state of neuron, ***X-initial***, communication network graph information indicated by ***graph*** signal, source and destination nodes, ***S_node*** and ***D_node***, K matrix calculated during setup system, ***K*** signal, coefficients values of Hopfield network that shown by z_1, z_2, z_3, z_4 and z_5, output of all neurons that saved in a register and known by ***HNN_out***, ***N_row*** and ***N_col*** signals that illustrate neuron's position among HNN neurons, ***ctrl*** and ***first_cycle*** signals that generate by controller unit block of Hopfield network setup, finally ***HNN_out[N_row][N_col]*** is output state signal of neuron. In the first operation stage (***MAX***) loaded ***X_sig*** by ***X_initial*** (when ***ctrl = '0'*** and ***first_cycle = '1'***) that lead to first neuron's output state after passing ***activation_function*** block next controller unit block changes ***first_cycle*** value to zero (***first_cycle = '0'***).

Internal state of proposed neuron is updated using Equation (15) where needed parts of equation are calculated by employing ***Y_sig***, ***N_row***, ***N_col***, ***S_node***, ***D_node***, ***graph*** and ***K*** signals and are assigned to ***DE_1, DE_2, DE_3, DE_4*** and ***DE_5*** signals. In the next posedge clock (***clk***) ***DE_s*** multiplying by corresponding coefficient signals then summation of all results produces local voltage variation of neuron, ***Delta_X*** signal, finally ***Delta_X*** plus to ***X_sig*** (previous local voltage) and make ***X_func*** (current local voltage) and the cycle continuous until ***ctrl*** signal value change to zero by controller unit block (***ctrl='0'***).

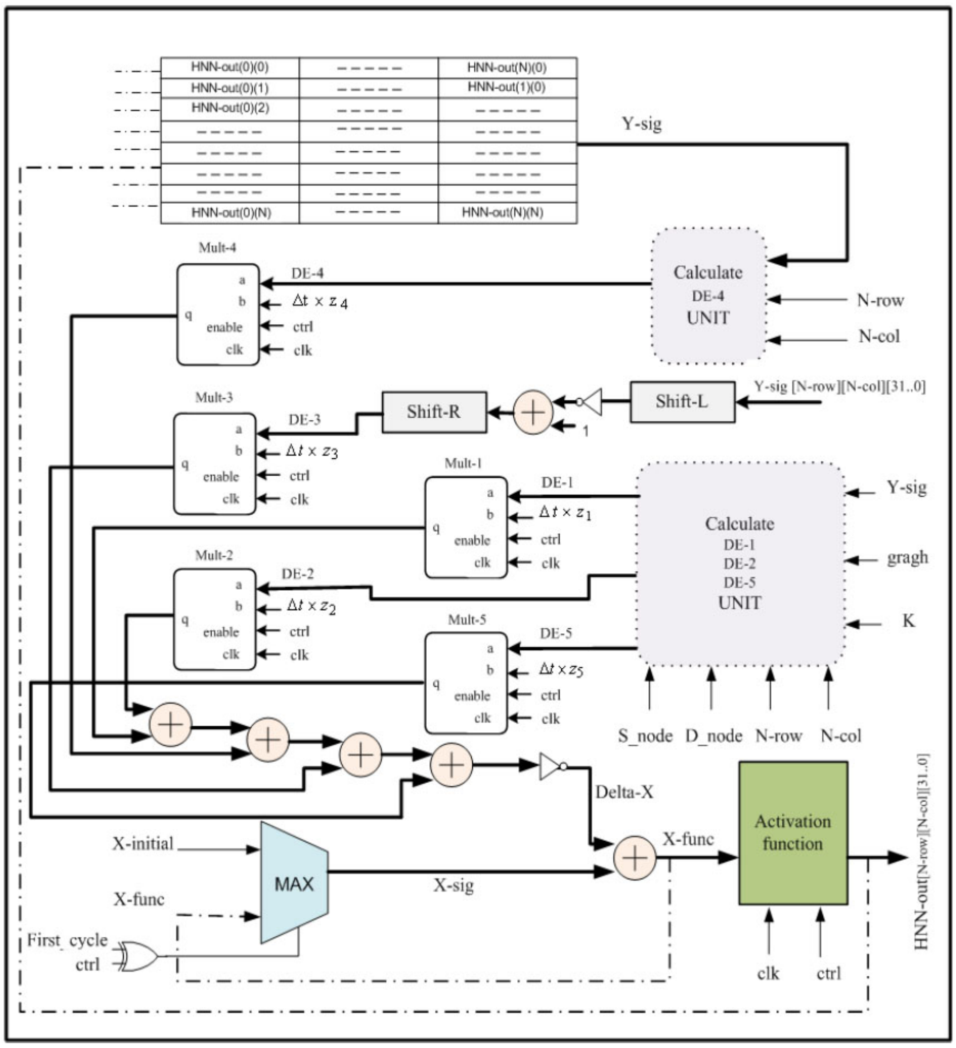

Fig. 3 *Hardware architecture of neuron.*

4.2 Activation Function

One important issue of implementing neural networks on FPGA is introducing a suitable architecture for computing activation functions. Many techniques have been introduced in literature for evaluating such elementary or nearly-elementary functions such as polynomial approximations, CORDIC algorithms, rational approximations, and table-driven methods.

The combination of low order polynomials approximation and not very large-scale look up table is an interesting method [28] using a piecewise linear approximation function $y = f(t) = m \times t + h$ considering m and h parameters. If the segments are chosen wisely, the sigmoid can be calculated using only shift and add operations. However, this method has a limited accuracy, with no possibility for improving (really by increasing degree of polynomial approximation to tow order or further it can be improved a little). Using the nearly odd property of the activation function, $f(-t) = 1 - f(t)$, the number of the segments can be decreased to half of the desired range. The non-saturation range is between -8 and 8, so the approximation segments only needs operate between 0 and 8. That requires all 15 integer bits and all 16 fractional bits to be included in the computations. Any numbers not in that range are considered to be in saturation and assigned an output value of 1. Block diagram of implementing a typical piecewise linear approximation function extracted from [29] is depicted in Fig. 4.

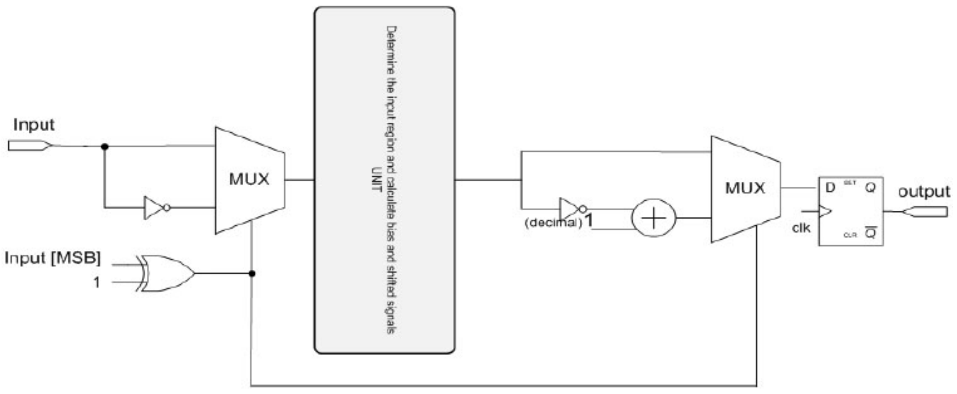
Input
Input [MSB]
1
MUX
Determine the input region and calculate bias and shifted signals UNIT
(decimal) 1
MUX
D
Q
clk
Q̄
output

Fig. 4 *Hardware architecture of activation function.*

4.3 Number presentation

Another issue is determining the numerical precision format that allows an optimum trade-off between precision and implementation areas. Standard single or double precision floating-point representations minimize quantization errors while requiring significant hardware resources. Less precise fixed-point representation may require less hardware resources with more quantization errors. Implementing a multi-layer perceptron (MLP) on FPGA using both fixed and floating point precision demonstrated that the fixed-point MLP implementation is 12 times faster in speed, over 13 times smaller in area, and achieves far greater processing density compared to the floating-point FPGA-based MLP [28]. In this paper numbers are represented in 32-bit fixed-point fashion; 1 bit for sign part, 16 bits for integer part and 15 bits for fractional part.

5. Simulation and Synthesizing Results

This section presents the implementing schemes of digital HNN on FPGA which is intended for real-time applications. The Cyclone Π-EP2C50F672C6 FPGA chip from ALTERA Inc. is considered for hardware implementing and VHDL language is employed for hardware description.

5.1 FPGA based HNN setup

The proposed architecture for FPGA implementation of Hopfield neural network model for solving shortest path problem is shown in Fig. 5. The Controller unit is considered as heart of system for generating internal data signals such as Initial state of neurons, S_node, D_node and Net parameter and controlling signals (Ctrl and first_cycle) for neurons operating. The Controller unit receives the information of Source and Destination nodes, the network Graph information and synchronizing clock (clk) for starting the operating of hardware neurons core in Hopfield network block after receiving an Enable signal. The Hopfield network block consists of $N \times (N-1)$ single hardware neurons for modelling shortest path problem in communication networks with N nodes. Here we consider two different models of neurons; a typical model shown in Fig. 2 and a new optimized model shown in Fig. 3. The two different models of neurons are employed in different experiments for a comparison study on hardware utilization, power utilization and maximum operating frequency.

After getting the information of communication network such as number of nodes and connecting link costs normalized in the range of $[0,1]$ then the required parameters are calculated considering the number of nodes in communication network and required neurons in Hopfield network. The HNN starts its operation, then Controller unit starts network operation by determining neurons initial states and assigned ctrl and first_cycle signals to one, when the HNN starts its operation and it's neuron output calculate for first time first_cycle signal value is changed to zero then the HNN operation continues until network reach to a stable state when further iterations do not change the output value of neurons (when ctrl='0').

5.2 Experimental examples

To evaluate the proposed architecture a network topology shown in Fig. 6 is considered. The proposed network consists of 4 nodes and 5 links. The shortest path between node pair (0, 2) is proposed where $S = 0$ and $D = 2$. The HNN model of proposed network includes 12 neurons. Initial state of all neurons is randomly considered in the range of [0,1], constant τ for all neuron is unit, $\tau = 1$, and the other network parameters are considered as follows; $z_1 = 950$, $z_2 = 2500$, $z_3 = 1500$, $z_4 = 475$, $z_5 = 2500$, $\Delta t = 10^{-5}$.

As shown in Fig. 6, the path $P_{0,2} = \{0,1,2\}$ was found by HNN as the shortest path and the cost (length) of shortest path is $C_{P_{0,2}} = 0.875$ which the minimum between the cost of all possible paths. Fig. 7 shows the timing diagram. As shown after 2558 iterations the shortest path was found.

As a comparison study, the performance of proposed neuron shown in Fig. 3 is compared by the typical neuron, Fig. 2, for interesting hardware implementing

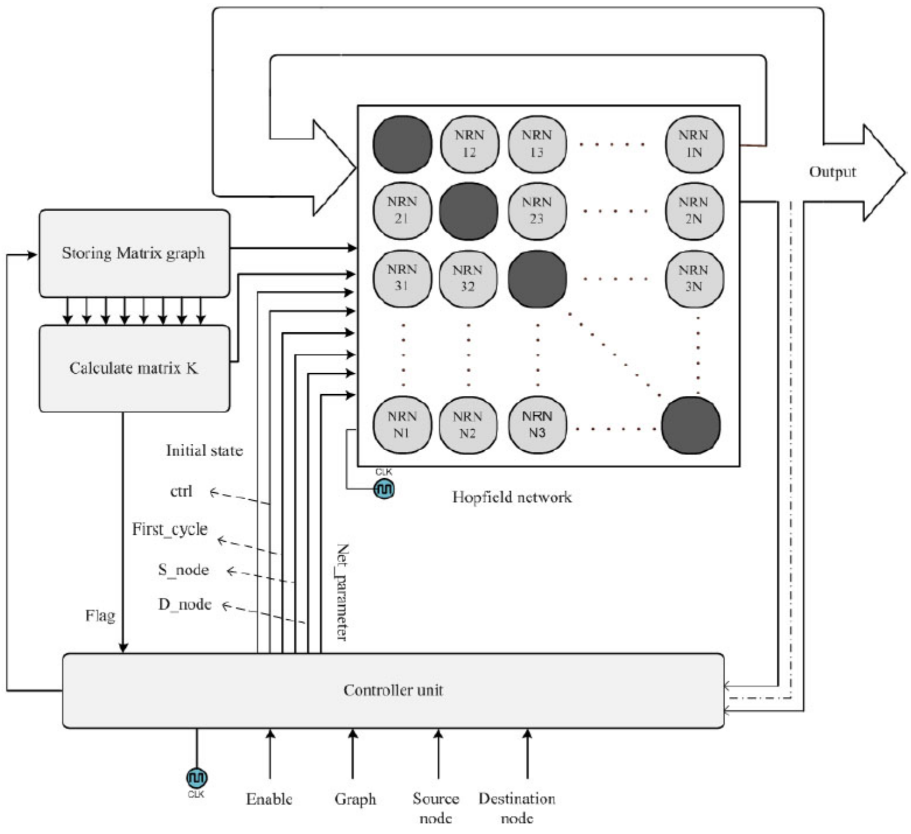

Fig. 5 *The block diagram of implemening Hopfield neural network for solving shortest path routing problem.*

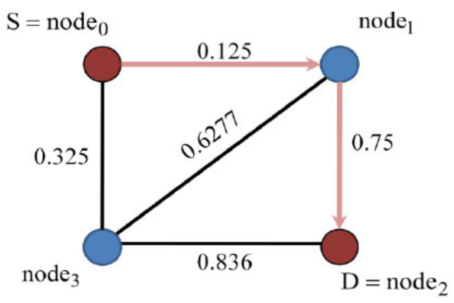

Fig. 6 *4-node network topology test-bench.*

issues. The results are described in Tab. I. As shown proposed architecture uses 19% and 2% of total logic elements and registers respectively where typical architecture uses 38% and 7% of logic elements and registers. The typical architecture use all

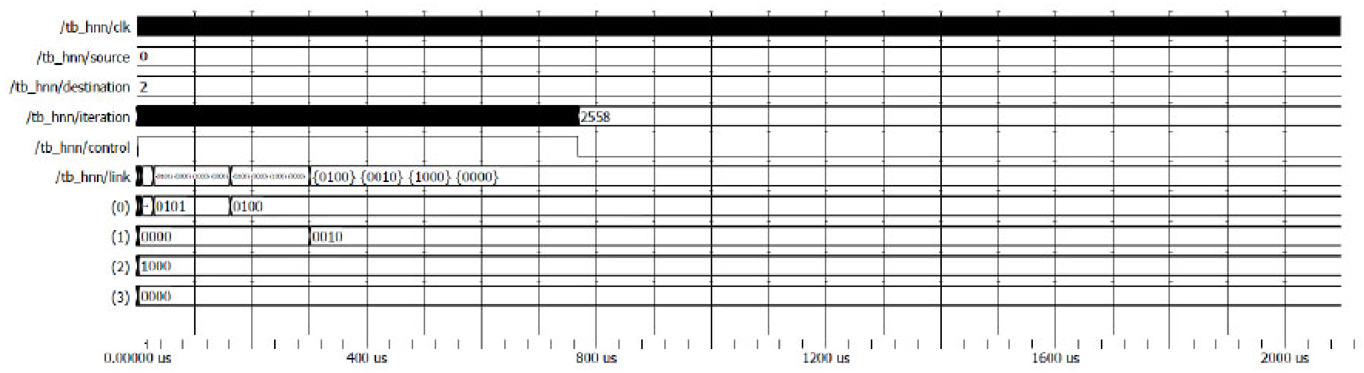

Fig. 7 *Timing diagram of 4-node network topology.*

Criteria	**Typical Neuron Architecture**	**Proposed Neuron Architecture**
Total Logic Elements	25675/68416 (38%)	12903/68416 (19%)
Dedicated logic Registers	5024/68416 (7%)	1330/68416 (2%)
Embedded Multiplier 9-bit Elements	300/300 (100%)	96/300 (32%)
Frequency (MHz)	25	27.3
Power Dissipation (mW)	225	224.2

Tab. 1 *Comparison results.*

multipliers on FPGA chip, 100%, where proposed architecture uses just 32% of multipliers which means 68% register saving.

The performance of proposed HNN architecture was also evaluated for two main important topologies; the ring topology (6 nodes, 6 links, node degree 2) and the star topology (6 nodes, 6 links, node degree 2), shown in Fig. 8 and Fig. 9 respectively. These topologies were considered as many applications use these practical topologies and other complex mesh networks could be considered as a combination of some star and ring topologies. In ring topology, the shortest path between node pair $(1, 3)$ is proposed where the path $P_{1,3} = \{1, 2, 3\}$ was found by HNN as the shortest path and the cost (length) of shortest path is $C_{P_{1,3}} = 1.5$ which the minimum between the cost of all possible paths. In star topology, the shortest path between node pair $(1, 3)$ is proposed where the path $P_{1,3} = \{1, 5, 3\}$ was found by HNN as the shortest path and the cost (length) of shortest path is $C_{P_{1,3}} = 0.665$ which the minimum between the cost of all possible paths.

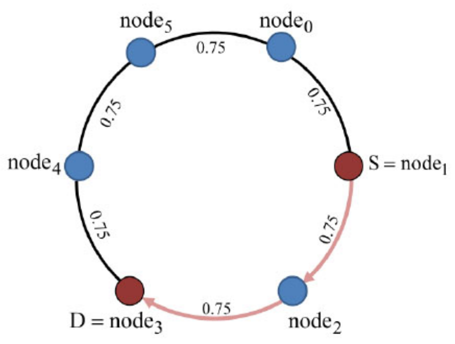

Fig. 8 *The ring network topology.*

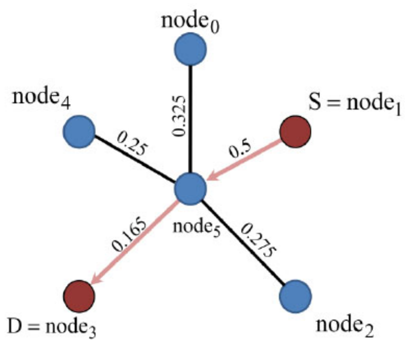

Fig. 9 *The star network topology.*

Figs. 10 and 11 show the initial-state and steady-state timing diagrams of HNN for Ring topology, respectively. As shown after 2914 iterations the HNN reached the steady state. The timing diagram of HNN for Star topology is shown in Fig. 12 where after 2409 iterations the HNN reached the steady state. The hardware utilization summary, maximum operating frequency and power dissipation for both star and ring topologies are described in Tab. II. As shown in Tab. II both networks have the same performance due to the same number of nodes and consequently the same number of neurons. The energy evolution for both Star and Ring topologies for finding shortest path between node pair $(1, 3)$ is shown in Fig. 13.

To evaluate the proposed architecture on real-world communication networks, the pan-European optical network [30] shown in Fig. 14 is considered. The network includes 18 nodes and 35 connecting links. The shortest path between node pair $(5, 12)$ is proposed; $S = 5$ and $D = 12$. The HNN model for the pan-European network includes 306 neurons.

Figs. 15(a), 15(b), 15(c) and 15(d) show the output state of all neurons versus the number of iterations. The output of all neurons at initial state is shown in Fig. 15(a) where all neurons begin their operation from 0.5 approximately. After starting network operation, the output of neurons changes and some converge to

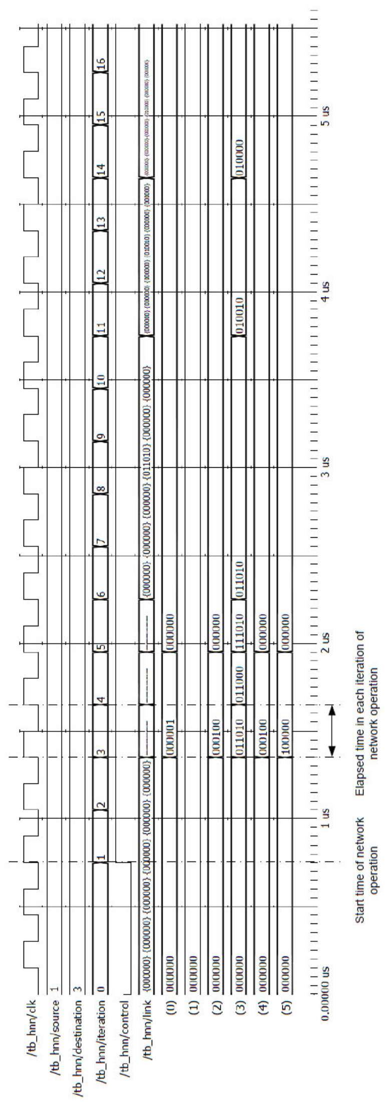

Fig. 10 *Timing diagram for ring topology during initial times.*

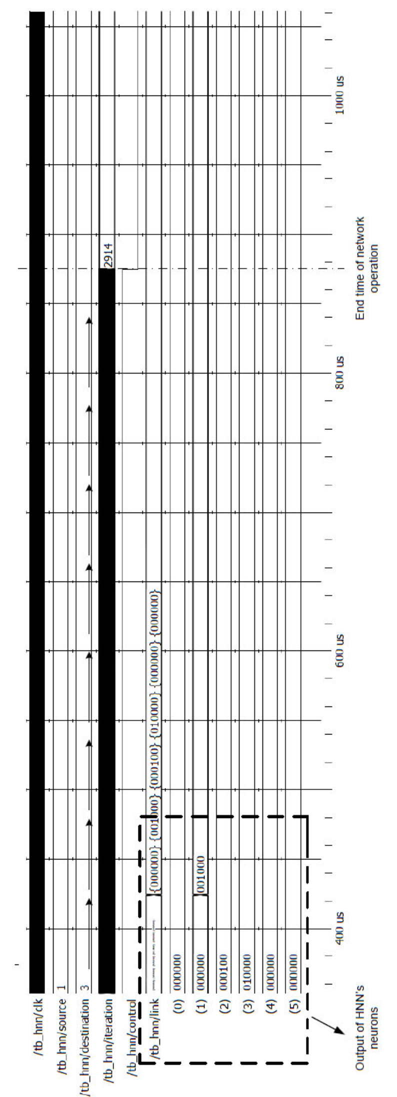

Fig. 11 *Timing diagram for ring topology during steady state.*

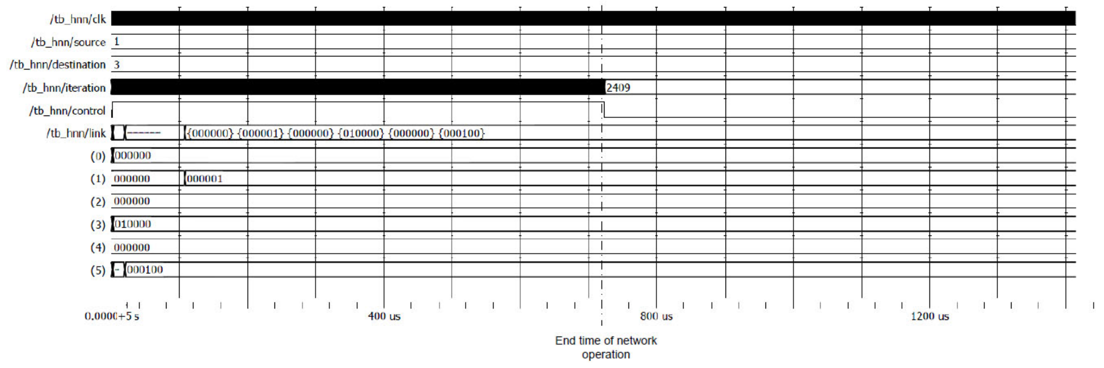

Fig. 12 *Timing diagram for star topology.*

Criteria	Ring Topology	Star Topology
Total Logic Elements	39485/68416(58%)	39485/68416(58%)
Dedicated logic Registers	3274/68416(5%)	3274/68416(5%)
Embedded Multiplier 9-bit Elements	240/300(80%)	240/300(80%)
Frequency (MHz)	21.6	21.6
Power Dissipation (mW)	239.39	239.39

Tab. II *Resource utilization summary.*

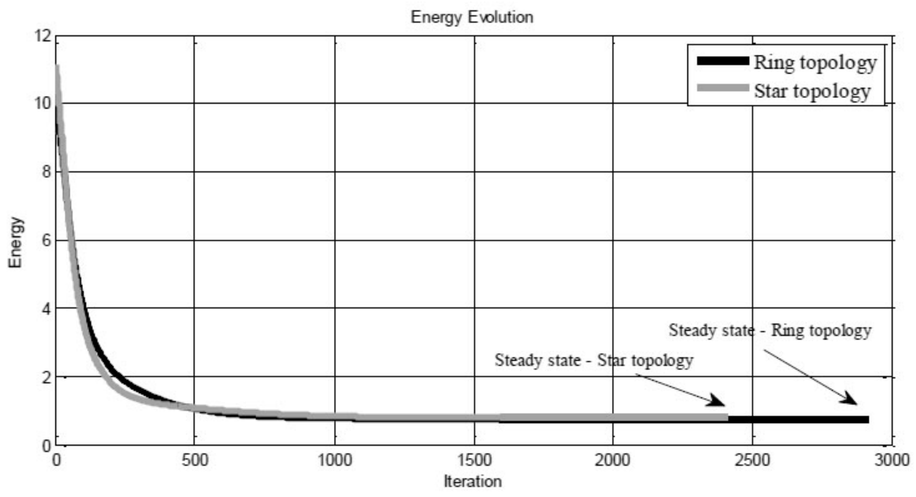

Fig. 13 *The energy evolution for both star and ring topologies.*

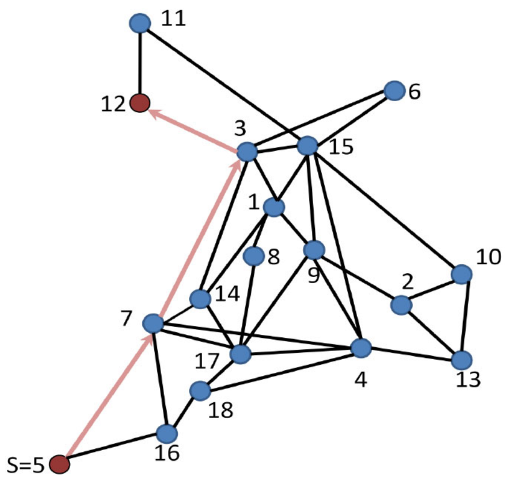

Fig. 14 *The pan-European network topology.*

zero and the output of some others become greater than 0.5 to determine the shortest path between source and destination nodes which has the minimum length, shown in Figs. 15(b) and 15(c).

Fig. 15(d) shows the final state of all neurons where, instead of two neurons representing source and destination nodes, two more neurons are in ON-situation which are connecting inter-mediate nodes. All of these nodes and corresponding links constitute required loop, and shortest path is generated from these links. The path $P_{5,12} = \{5, 7, 3, 12\}$ is considered as a sequence of nodes from source to destination nodes connected by links.

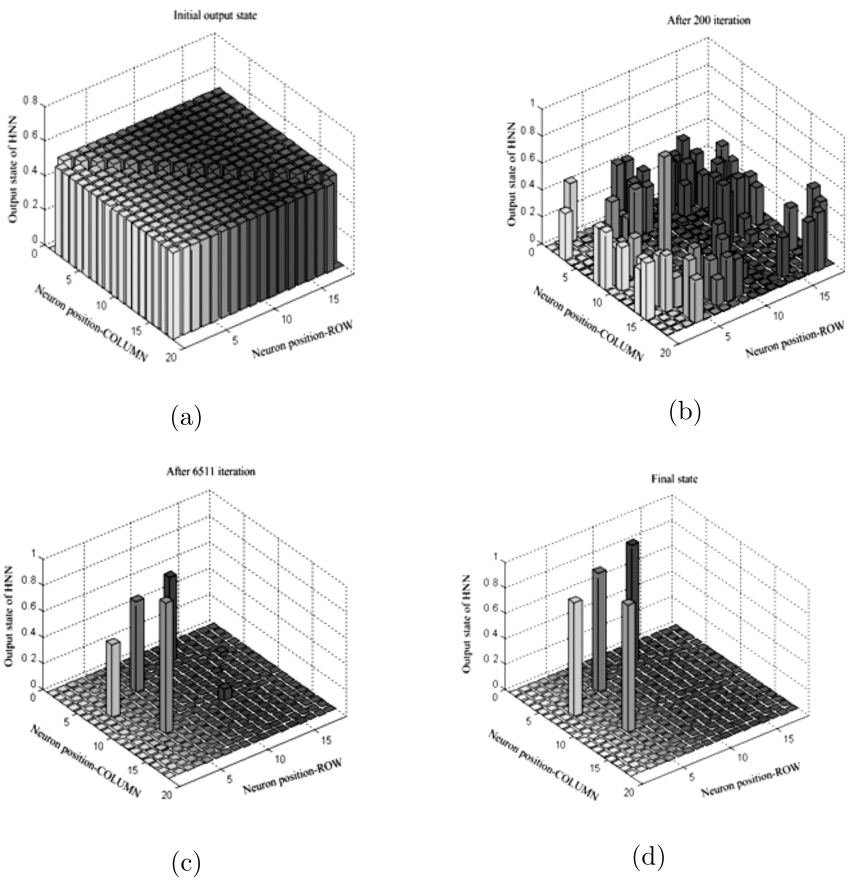

Fig. 15 *Evolution of output state of all neurons in the HNN (a)-Initial state (b)-After 200 iterations (c)-After 3000 iterations (d)-Steady state (After 10116 iterations).*

The timing diagram is shown in Fig. 16. The evaluation of the network overall energy is presented in Fig. 17 where the energy of network is decreased versus the number of iterations until network find the stable state and shortest path.

6. Conclusion

The Hopfield neural network is a recurrent and single layer neural network which is suitable for modelling and solving routing problem in communication networks. In this paper a digital hardware FPGA based implementation of Hopfield neural network for solving shortest path problem in communication networks was proposed. Modelling shortest path problem in a network with n nodes used $n\times(n-1)$ neurons where states of neuron outputs were updated using energy evaluation functions. The Cyclone II-EP2C70F896C6 FPGA chip from ALTERA Inc. was considered

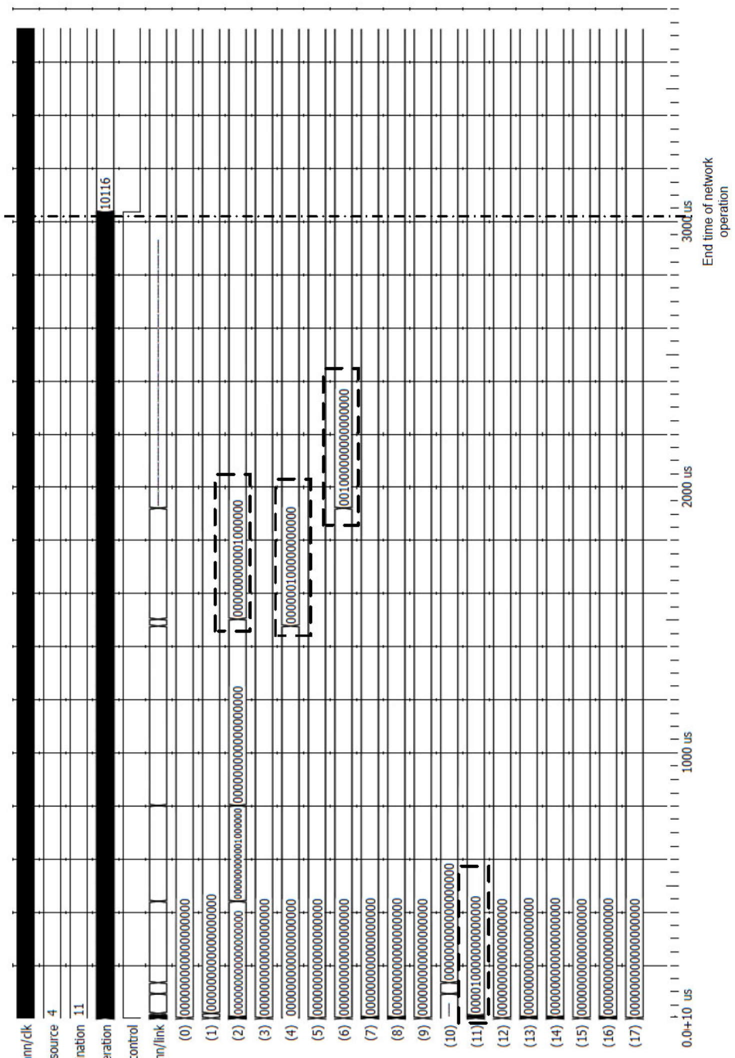

Fig. 16 *Timing diagram of pan-European network.*

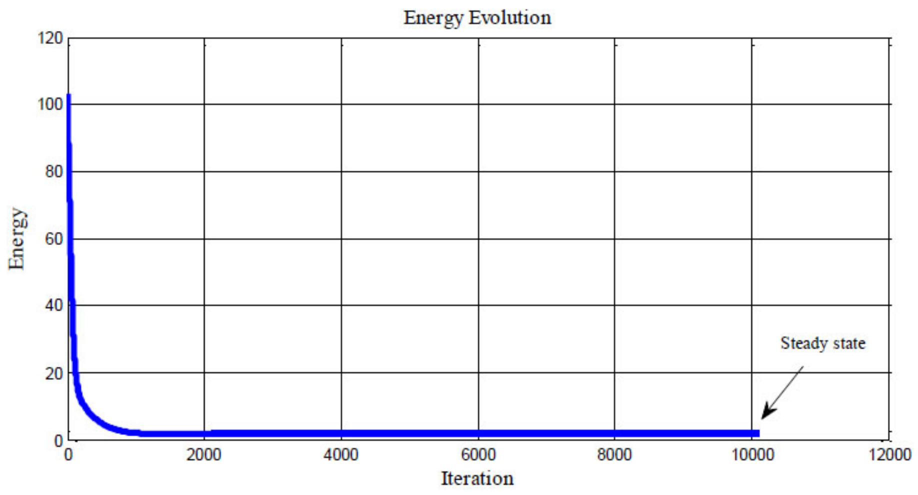

Fig. 17 *Energy evolution of pan-European network..*

for hardware implementing and VHDL language was employed for hardware description. The results demonstrated that the proposed approach is efficient for solving shortest path problem in communication networks. Furthermore, multiple processing cores could be employed in parallel architectures for implementing practical routing protocols and algorithms in large-scale networks.

References

[1] Strub S. M.: Routing in Communications Networks. Prentice Hall, United States, 1995.

[2] Korzun D., Gurtov A.: Structured Peer-to-Peer Systems: Fundamentals of Hierarchical Organization, Routing, Scaling, and Security. Springer, 2012.

[3] Dargie W., Poellabauer C.: Fundamentals of wireless sensor networks : theory and practice. John Wiley and Sons, 2010.

[4] Kavian Y. S., Rashvand H. F., Ren W., Leeson M. S., Hines E. L., Naderi M.: RWA problem for designing DWDM networks – delay against capacity optimisation, Electron. Lett., **43**, 2007, pp. 892–893.

[5] Kavian Y. S., Rashedi A., Mahani A., Ghassemlooy Z.: Routing and wavelength assignment in optical networks using Artificial Bee Colony algorithm. Optik, **124**, 2013, pp. 1243–1249.

[6] Sim K. W., Sun W. H.: Ant colony optimization for routing and load-balancing: survey and new directions. IEEE Transactions on Systems, Man and Cybernetics,Part A: Systems and Humans **33**, 2003, pp. 560–572.

[7] Bastos-Filho C. J. A., Schuler W. H., Oliveira A. L. I., Vitorino L. N.: Routing algorithm based on Swarm Intelligence and Hopfield Neural Network applied to communication networks. Electronics Letters, **44**, 2008, pp. 995–997.

[8] Zhang Y., Wu L., Wei G., Wang S.: A novel algorithm for all pairs shortest path problem based on matrix multiplication and pulse coupled neural network. Digital Signal Processing, **21**, 2011, pp. 517–521.

[9] Araújo F., Ribeiro B., Rodrigues L.: A Neural Network for Shortest Path Computation. IEEE Transactions on Neural Networks, **12**, 2001, pp. 1067–1073.

[10] Mehmet Ali M. K., Kamoun F.: Neural networks for shortest path computation and routing in computer networks. IEEE Transactions on Neural Networks, **4**, 1993, pp. 941–953.

[11] Hopfield J. J.: Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences, 1982, pp. 2554–2558.

[12] Ionescu L. M., Mazare A. G., Serban G., Barbu V., Constantin A.: FPGA implementation of an associative Content Addressable Memory. Appled Electronics (AE), IEEE International Conference, 2011, pp. 1-4.

[13] Tank D. W., Hopfield J. J.: Simple neural network optimization networks: An A/D converter, signal decision circuit and a linear programming circuitries transaction on circuit and systems. IEEE Tranansactions on Circuits and Systems, **33**, 1986, pp. 535–541.

[14] Park D. C., Choi S. E.: A neural network based multi-destination routing algorithm for communication network. IEEE World Congress on Computational Intelligence, **2**, 1998, pp. 1673–1678.

[15] Bastos-Filho C. J. A., Santana R. A., Silva D. R. C., Martins-Filho J. F., Chaves D. A. R.: Hopfield neural networks for routing in all-optical networks. 12th International Conference Transparent Optical Networks (ICTON10), 2010, pp. 1–4.

[16] Jain S., Sharma J. D.: Delay bound multicast routing using Hopfield neural network. International Journal of computer Theory and Engineering, **2**, 2010, pp. 384–389.

[17] Kojic N. S., Zajeganovic Ivancic M. B., Reljin I. S., Reljin B. D.: New algorithm for packet routing in mobile ad-hoc networks. 7th International Conference on Hybrid Intelligent Systems, **20**, 2010, pp. 9–16.

[18] Kowalska T. O., Kaminski M.: FPGA Implementation of the Multilayer Neural Network for the Speed Estimation of the two-Mass Drive System. IEEE Transactions on Industrial Informatics, **7**, 2011, pp. 436–445.

[19] Himavathi S., Anitha D., Muthuramalingam A.: Feedforward neural network implementation in FPGA using layer multiplexing for effective resource utilization. IEEE Transactions on Neural Networks, **18**, 2007, pp. 880–888.

[20] Girau B., Tiseeerand A.: MLP computing and learning on FPGA using on-line arithmetic. Journal on System research and Information Science, Special issue on Parallel and Distributed Systems for Neural Computing, **9**, 2000, pp. 2–4.

[21] Navabi Z.: VHDL: Analysis and Modeling of Digital Systems. McGraw-Hill, New York, 1993.

[22] Hopfield J. J.: Neurons with graded response have collective computational properties like those of two-state neurons. Proc. Nat. Acad. Sci, **81**, 1984, pp. 3088–3092.

[23] Haykin S.: Neural networks. A Comprehensive Foundation, New York, Macmillan, 1994.

[24] Fausett L. V.: Fundamentals of Neural Networks. Englewood Cliffs, Prentice Hall, 1994.

[25] Jimenez R., Sanchez-Raya M., Gomez-Galan J. A., Flores J. L., Duenas J. A., Martel I.: Implementation of a neural network for digital pulse shape analysis on a FPGA for on-line identification of heavy ions. ELSEVIER, Nuclear Instruments and Methods in Physics Research A, **674**, 2012, pp. 99–104.

[26] Nedjah N., daSilva R. M., Mourelle L. M., daSilva M. V. C.: Dynamic MAC-based architecture of artificial neural networks suitable for hardware implementation on FPGAs. ELSEVIER, Neurocomputing, **72**, 2009, pp. 2171–2179.

[27] Hikawa H.: FPGA implementation of self organizing map with digital phase locked loops. Neural Networks **18**, 2005, pp. 514–522.

[28] Omondi A. R., Rajapakse J. C.: FPGA implementations of Neural Networks. Springer, 2006, p. 21–200.

[29] Reynolds P. D.: Algorithm implementation in FPGAs demonstrated through neural network inversion on the SRC-6e. MSC Thesis, Waco-Texas, 2005, pp. 24–53.

[30] Kavian S. Y., Strobel O., Mahani A., Rejeb R.: Fuzzy future demand uncertainty management in optical networks. The Mediterranean Journal of Computers and Networks, **7**, 2011, pp. 273–280.