ISSN XXXX XXXX © 2019 IJESC

Research Article Volume 9 Issue No.1

A Spiking Neural Network Controller Design with VHDL Based on SSO Algorithm for Inverted Pendulum

Mohammed Y. Hassan[1], Ahmed Abduljabbar Mahmood[2]
Department of Control and Systems Engineering
University of Technology, Baghdad, Iraq

Abstract:
The design and simulation of the Spiking Neural Network Controller (SNNC) is presented in this paper for controlling an inverted pendulum SISO system with external disturbances. The VHDL code for implementing a SNNC using Field Programmable Gate Array (FPGA) have been proposed to control the inverted pendulum. Spike Response Model is selected in SNNC that can work with different structures as a P, PI, PD or PID like. The SNN controller has three inputs that represent the error, integral of error and derivative of error. It has one output that represent the control signal. It uses control lines to select the type of the controller, the nodes of the hidden layer, and sub-connection terminals (synapses). The weights of the SNN layers are learnt using Social-Spider Optimization algorithm with minimum Mean Square of Error (MSE) and minimum number of epochs. The design must achieve the minimum processing time, minimum FPGA chip size and as a result minimum cost. The hardware code for VHDL to implement the proposed SNNC have been generated successfully.

Keywords: Spiking Neural Network Controller, Social-Spider Optimization, Inverted pendulum, VHDL, FPGA.

1. INTRODUCTION

Spiking Neural Networks (SNNs) have obtained the interest of the researchers in many research fields in which SNNs have been successfully applied in many computational tasks, such as speech processing, robot control and navigation, as compared with conventional Artificial Neural Networks (ANNs). SNN is the third generation of artificial neural networks (ANNs). Several researchers used SNNs to solve the same problems of classical neurons, but with less iterations, which represents less processing time and faster networks. Also, SNNs used fewer nodes in compare with ANN. Thus, Spiking Neural Network Controller (SNNC) can achieve with small network size. On the other hand, FPGAs have become recently an alternative to full custom Very Large Scale Integration (VLSI) design. FPGAs have reasonable price and make the development cycle shorter in comparison to VLSI [1]. SSO algorithm is best than PSO and GA by classification performance that uses various estimation indicators and in wrapper-based feature selection [2]. In recent years, a huge attention in the using field Programmable Gate Arrays (FPGAs) to implement SNNs as a controller has grown. Xue et al. (2014) [4] designed a controller based on SNN for mobile robots. This controller was based on Self-Organized Spiking Neural Network (SOSNN) and it was implemented by an FPGA chip. The spiking neurons were adopted on the Izhikevich model and a spike timing-dependent plasticity learning was used to tune the synaptic connection. By emitting spikes of the motor neurons, the robot can be controlled on the angular velocity of the driving wheels of the mobile robot. Zbrzeski et al. in 2016 [5], presented an original bio-inspired assistive technology for real-time ventilation assistance and implemented in a digital configurable FPGA. The bio-inspired controller is applied to the closed loop diaphragmatic stimulation. This controller is a SNN inspired by the medullary respiratory network. It is robust as a classic controller and having a flexible, low-power and low-cost hardware design. Antonietti et al. in 2018 [6] developed the SNN to control NAO Robot. It was a simplified model of the neurophysiological cerebellar network, including neural connectivity, plasticity mechanisms, and simplified neuron dynamics. The Pavlovian conditioning protocol tested the model capability to learn the temporal associations between two stimuli provided to the humanoid NAO robot. The developed platform is a real neurorobot able to operate in a dynamic and noisy environment, dealing with a simple learning motor task. In this paper, the design, simulation and implementation of a SNNC in FPGA chip to work with different structures (P, PI, PD or PID like) and controls inverted pendulum SISO system is presented. The type of controller, number of hidden nodes, and number of synapses are set using external inputs. The SSO algorithm is used to obtain the best weights of the SNNC. The minimum MSE is the criterion used in obtaining the best weights between SNN layers.

2. SOCIAL SPIDER OPTIMIZATION (SSO) ALGORITHM

The Social spider optimization (SSO) is a swarm optimization algorithm that was introduced by Erik Cuevas et al. in 2013 [2]. SSO algorithm is applied in many optimization problems, but it is different than other optimization methods. It applies simple mathematical operators and does not require complex operators where it's time and space complexity are not expensive [3]. This algorithm works according to the two different search operators are known as females and males [7]. SSO converts the solution of any problem to a position in its web to starts the commutation between spiders. The spiders' communication is performed by the vibration that is transmitted when any spider moves to a new position [8]. It can summarize the SSO algorithm as the following steps:

1. Initialize the overall population randomly, the number of females (N_f) is chosen randomly with the range (65% - 90%) of the number of spiders (N_s) and number of male

spiders (N_m) are determined by using the below equations [2]:

$$N_f = \text{floor}\,[(0.9 - \text{rand} * 0.25) * N_s] \quad (1)$$

$$N_m = N_s - N_f \quad (2)$$

, where (floor) maps a real number to an integer number and (rand) is a random number generator between [0, 1].

2. Calculate the fitness evaluation, where the fitness function value of each spider (W_j) that represents the solution (in this paper (W_j) consists of (w_{hi}^k) and (w_{ij}^k)) is calculated using equation [2]:

$$W_j = \frac{J(Si) - worst_s}{best_s - worst_s} \quad (3)$$

, where (Si) represented the spider position and ($J(Si)$) represented the fitness value for this position. Also, $best_s$ and $worst_s$ are represented the minimum and the maximum values for the solution in the population (minimum value problem) as defined in equations (4) and (5), respectively [2]:

$$\text{best}_s = min_{i\in[1,2,3,\ldots,N]}(J_{(si)}) \quad (4)$$

$$\text{worst}_s = \max_{i\in[1,2,3,\ldots,N]}(J_{(si)}) \quad (5)$$

3. Design of the vibrations through the communal web, where the vibrations spotted by the individual (i) from member (j) are defined as [2]:

$$V_i\, b_{i,j} = w_j . e^{-d_{i,j}^2} \quad (6)$$

, where the ($d_{i,j}$) is the Euclidian distance between the spiders (i) and (j), such that $d_{i,j} = \|s_i - s_j\|$. There are three relationships of the vibrations between any pair of individuals which are used within the SSO algorithm. ($Vibci$) is the vibrations between the individual (i) and a member $c(s_c)$, that is the nearest member to (i) and has the highest weight. ($Vibbi$) which represents the vibrations between the individual (i) and a member $b(s_b)$ that is the best fitness value (best weight value) of the whole population (S). ($Vibfi$) which represents the vibrations between the individual (i) and the nearest female individual $f(s_f)$ [2].

$$Vibci = w_c . e^{-d_{i,c}^2}, \quad Vibbi = w_b . e^{-d_{i,b}^2}, \text{ and } Vibfi = w_f . e^{-d_{i,f}^2} \quad (7)$$

4. The female cooperative process. A uniform random number (r_m) is created within the range [0, 1]. When (r_m) is smaller than a threshold (PF), an attraction movement is produced; otherwise, a repulsion movement is generated as shown below [2] :

$$f_i^{t+1} = \begin{cases} f_i^t + \propto . Vibci . (s_c - f_i^t) + \beta . Vibbi . (s_b - f_i^t) \\ \quad + \delta . (rand - 0.5), \text{ with probability (PF)} \\ f_i^t - \alpha . Vibci . (s_c - f_i^t) - \beta . vibbi . (s_b - f_i^t) \\ + \delta . (rand - 0.5), \text{with probability } (1 - PF) \end{cases} \quad (8)$$

, where (t) is the iteration number and (α, β, δ, and rand) are random constant created between [0,1].

5. The male cooperative process. When the weight value of the other male's members is lower than the median value of the male population, it is considered as the median male member and it has the weight (wN_{f+m}). Changing positions of the male spider (N_m) are defined as shown in equation[2]:

$$m_i^{t+1} = \begin{cases} m_i^t + \alpha . Vibfi . (s_c - m_i^t) + \delta . (rand - 0.5), \\ \quad if\ wN_{f+i} > wN_{f+m} \\ m_i^t - \alpha . \left(\frac{\sum_{h=1}^{N_m} m_h^t . wN_{f+h}}{\sum_{h=1}^{N_m} wN_{f+h}} - m_i^t \right), \\ \quad if\ wN_{f+i} \leq wN_{f+m} \end{cases} \quad (9)$$

, where ($\frac{\sum_{h=1}^{N_m} m_h^t . wN_{f+h}}{\sum_{h=1}^{N_m} wN_{f+h}}$) represents the weighted mean of the male population (M).

6. The matting process. The female members and dominant males execute mating in social spider colony. Therefore; an overwhelming male (m_g) spider finds a set (E^g) of female individuals inside a specific extend (r), which mates and shapes a new offspring (s_{new}). It is significant to increase that if the set (E^g) is unfilled, then mating task is dropped. The range (r) is defined as a radius which relies upon the size of the inquiry space which is computed as follows [2] :

$$r = \frac{\sum_{j=1}^{n} (p_j^{high} - p_j^{low})}{2 * n} \quad (10)$$

Finally, the mating procedure is done. The spiders with heavier weight will probably impact the new offspring. The impact likelihood (p_{si}) of every part is allotted by the Roulette wheal method and characterized as follows where (T^g) is all the element generated by ($E_g \cup m_g$) and ($i \in T^g$) [2]:

$$p_{si} = \frac{w_i}{\sum_{j\in T^g} w_j} \quad (11)$$

3. SPIKING NEURAL NETWORK

A SNN is the third generation of ANN that was introduced by Wolfgang Mass in 1997. The structure of the SNN and the ANN is similar but the SNN has synapses between each neuron layers [9]. The SNN model used in this paper is the SRM. It is introduced by Gerstner in 1995 and due to its arithmetical simplicity, its consider the most widely contributed model [10]. There are different forms of SRM and according to the choice of suitable Spike Response Functions (SRF) [11]. It is supposed that every neuron can generate only one spike as a maximum through the interval time of the simulation and it is firing when the interior status variable is greater than or equal to the threshold value. Also, it is assumed that pre-synaptic neuron (i) fires a spike at time (t_i) and the (K^{th}) synapse transfers that spike into the post-synaptic neuron at ($t_i + d^k$). Where (d^k) is the delay correlated with the (K^{th}) synapses [11]. The SRF $(\varepsilon(t))$ termed the influence of the pre-synaptic neuron potential on the postsynaptic neuron potential [10]. There are different mathematical forms of SRF such as the hyperbolic tangent function that is used in this paper, as below [8]:

$$\varepsilon(t) = \tanh \frac{t}{\tau} \quad (12)$$

where (τ) is time constant that determines growth and decay in this SRF. The derivative of hyperbolic tangent function becomes:

$$\frac{d\varepsilon}{dx} = \frac{1}{\tau}\left(1 - tanh^2\left(\frac{t}{\tau}\right)\right) \quad (13)$$

A. Structure of the Proposed SNNC

The structure of proposed SNNC is shown in Figure (1). It is consists of three layers, the input layer (H) which has three neurons (nodes) that represent the error (e), integral of error (ie) and derivative of error (de), the hidden layer (I) which has (n) neurons (nodes) and one neuron in the output layer (J) that represents the control action (U). (W1) represents the array of weights between input and hidden neurons and (W2) represents the array of weights between hidden and output neurons. Each connection between input and hidden layers (as shown in Figure (2-A)), hidden and output layers (as shown in Figure (2-B)) consists of (k) synapses. Number of hidden units and number of synapses in each sub-connection are chosen through trial and error method.

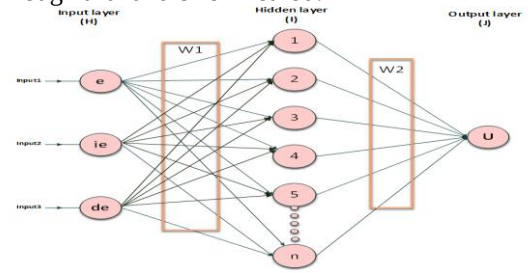

Figure.1. The structure of the proposed SNN controller.

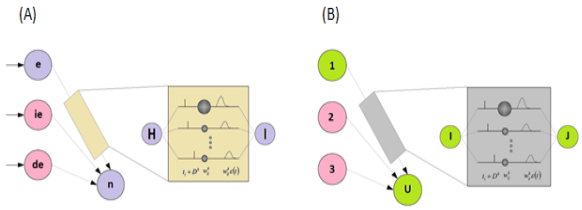

Figure.2. (A) The synaptic connections between input and hidden layers. (B) The synaptic connections between hidden and output layers.

B. The Modified SNN Training Algorithm

The feed-forward SNN comprises of three layers, input layer (H), hidden layer (I), and output layer (J). The transmitting of the data through SRM of SNN is described in the following steps [8]:

1. Data pass to the input layer neurons is encoded into spikes time and is computed using the following equation [10]:

$$t_h^f(I_{in}) = T_{max} - round\left(T_{min} + \frac{(I_{in}-I_{min})(T_{max}-T_{min})}{I_{max}-I_{min}}\right) \quad (14)$$

, where (t_h^f) is the firing time for input neurons, (T_{max}) is the maximum spike time and (T_{min}) is the minimum spike time. (I_{max}) is the largest value in the input data, (I_{min}) is the lowest value in the input data, and (I_{in}) is the present value of the input data.

2. Every hidden layer neurons is tested if it is spiked or not spiked. At most, every neuron can spike once just amid the time interval, when its membrane potential (m_i) is greater than or equals the threshold (ϑ). When neuron (i) is spiked at time (t_i^f), the algorithm goes to the next neuron $(i+1)$ in the current layer. Otherwise, the membrane potential $(m_i(t))$ is computed by the following equation [11]:

$$m_i(t) = \sum_{h=1}^{Hn}\sum_{k=1}^{Dn} w_{hi}^k(t)\varepsilon(t - t_h^f - d^k) \quad (15)$$

, where (H_n) is the number of the i/p layer neurons, (D_n) is the number of sub-synaptic, (w_{hi}^k) is the weight between neurons in the input layer and hidden layer, (d^k) denotes to the delay of the connection and $\varepsilon(t - t_h^f - d^k)$ is the response function. When $(m_i(t))$ exceeds the threshold value at instant (t), the neuron (i) will be prohibited from spiking more in the remaining time interval (T), Otherwise it will be reset in the next instant $(t+1)$. When the neurons of the hidden layer are finished, the same algorithm is repeated to the output layer (j), but in this case the spikes of the hidden layer will be the inputs to the neurons of the output layer. The membrane potential $(m_j(t))$ is calculated as [10]:

$$m_j(t) = \sum_{i=1}^{In}\sum_{k=1}^{Dn} w_{ij}^k\varepsilon(t - t_i^f - d^k) \quad (16)$$

, where (I_n) is the neurons number in the hidden layer, (t_i^f) is the firing time for hidden layer (i) and (t_j^f) is the firing time of output layer (j).

3. Finally, the output information of SNN is converted from time to real number using equation (7), which is extracted from the coding equation (14):

$$RI\left(t_j^f\right) = \frac{\left(T_{max}-t_j^f-T_{min}\right)*(I_{max}-I_{min})}{(T_{max}-T_{min})} \quad (17)$$

, where $RI\left(t_j^f\right)$ is the real information for the firing time of the output layer (j). After completing the feed-forward calculations, the error is computed using Mean Squares Error (MSE) [10]:

$$MSE = \sum_{j\in J}\left(t_j^f - t_j^d\right)^2 \quad (18)$$

, where (t_j^d) is training target spike time for output neuron (j).

The steps of training the SNN that were presented from equation (14) to equation (18) are applied to all the structure of the proposed controller for inverted pendulum system. Then initialize the number of spiders, female spiders with their positions, and male spiders with their positions. Also, we set the number of hidden neurons, sub-connection between two nodes (synapses), lower bound of the weights, and the upper bound of the weights. The sampling time is calculated for inverted pendulum model according to the time constant of the model and the weights dimensions are determined. The obtained weights from SSO between the input and the hidden layers and between the hidden and the output layers are fed through the Simulink to the SNNC, represented as a MATLAB function. The SNN parameters are chosen using trial and error method, to achieve minimum numbers of hidden neurons with minimum number of synapses and as a result minimum structure and less computational operations. The weights for each spider (which are the same weights of the SNN) are set in the SNN controlled system and MSE is calculated (as in equation (18)). The training algorithm stops when the best weights are obtained. That is, reaching the minimum MSE with minimum number of iterations. Finally, the best weights are set in the SNN controller.

4. Languages of Programming in FPGA

There are two digital hardware languages used to implement FPGA chip that are called VHDL and Verilog. The original purpose of VHDL and Verilog languages was for modeling and documenting electronic systems. They are hierarchical structural languages that describe the hardware structurally using logic blocks with their interconnections. The modules

can be written easily in VHDL or Verilog languages for testing the functionality of the logic circuits. Structural designs can be easily synthesized, as can behavioral descriptions that conform to a Register Transfer Level (RTL) coding style. However, these restrictions mean that many of the constructs available within both languages are not synthesizable, see Figure (3) [12]. There are many differences between Verilog and VHDL languages. Where Verilog is relatively easy to learn while VHDL is difficult to learn. The Verilog uses fixed data types with limited design management and reusability. The VHDL uses abstract data types with good design management and reusability. Finally, VHDL code has less gate-level timing compared with Verilog code [13].

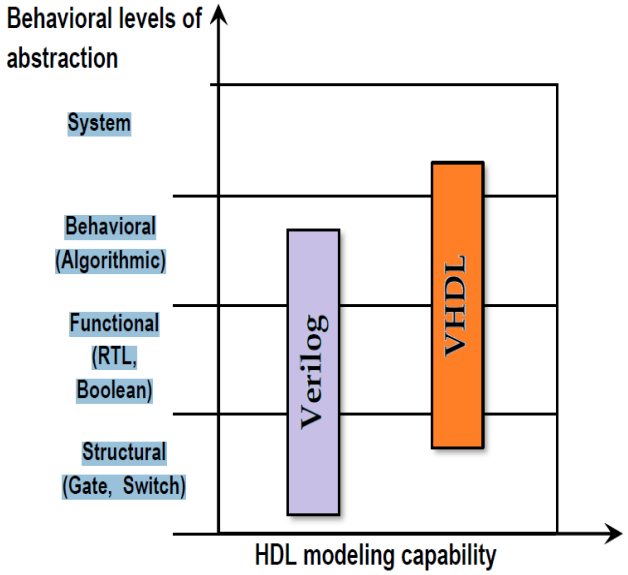

Figure.3.Modeling capability for HDL (Verilog versus VHDL) [12].

5. DESIGN AND SIMULATION FOR THE PROPOSED SNNC

This controller has three inputs and one output. The first input is the error, the second input is the integral of error and the third input is the derivative of error. The SNN output represents the control action signal and feeds the inverted pendulum system that must be controlled. The block diagram of the proposed SNN controller is shown in Figure (4).The parameters that are used in the SNNC are the weights between input and hidden layers (W1), the weights between hidden and output layers (W2), two control lines (c1 and c2) to select the type of the controller as listed in Table (1), three control lines (h1, h2 and h3) to select the number of the hidden neurons and three control lines (s1, s2 and s3) to select the number of the synapses between any two neurons as listed in Table (2). The SNNC that was simulated with inverted pendulum system is showing in Figure (5).

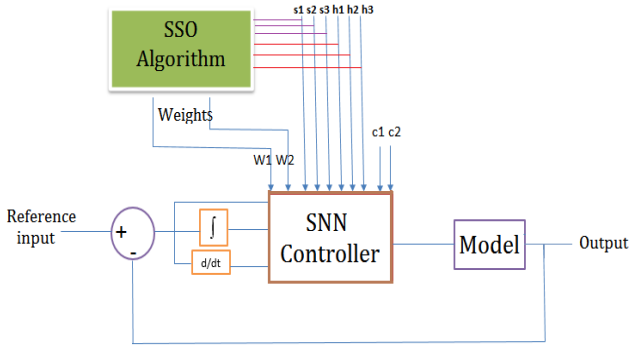

Figure.4. Block diagram for the proposed controller.

Table.1. Selection lines for the type of controller.

CONTROL SIGNALS C2 C1	CONTROLLER TYPE
0 0	P-LIKE SNN
0 1	PD-LIKE SNN
1 0	PI-LIKE SNN
1 1	PID-LIKE SNN

Table.2. Selection lines for the number of synapses and hidden nodes used in SNN.

SELECTION LINES S3 H3	S2 H2	S1 H1	NO. OF SYNAPSES/ NO. OF HIDDEN NODES
0	0	1	ONE
0	1	0	TWO
0	1	1	THREE
1	0	0	FOUR
1	0	1	FIVE
1	1	0	SIX
1	1	1	SEVEN

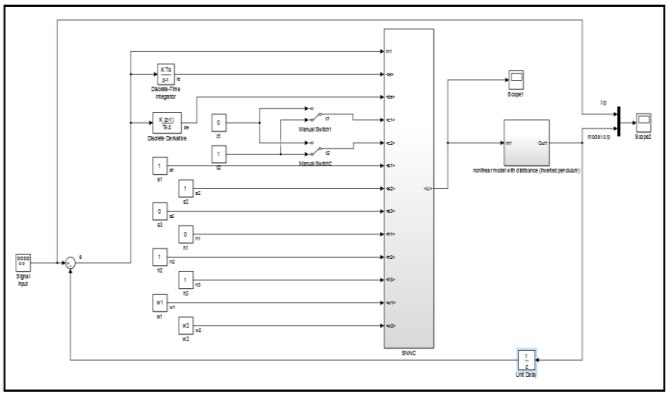

Figure .5. Simulink structure of P/PI/PD/PID-like SNNC.

A. Nonlinear Inverted Pendulum Model

The inverted pendulum system has a unique characteristic that has been considered as a prototype of nonlinear systems whose control is known to be not easy [14]. We apply it to a nonlinear Single-Input Single-Output (SISO) inverted pendulum system with external disturbances to verify the effectiveness and superiority of the proposed SNNC. Figure (6) shows the inverted pendulum system.

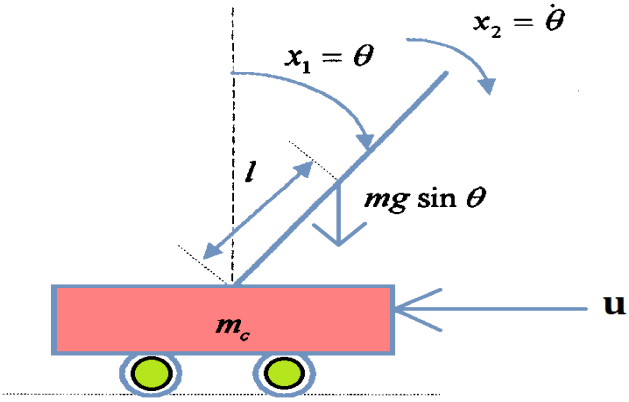

Figure.6. The inverted pendulum system.

The nonlinear differential equation of the single inverted pendulum can be given by [15]:

$$\dot{x}_1 = x_2 \quad (19)$$

$$\dot{x}_2 = \frac{g\sin x_1 - mlx_2^2\cos x_1 \sin x_1/(m_c+m)}{l\left(\frac{4}{3} - m\cos^2 x_1/(m_c+m)\right)} + \frac{\cos x_1/(m_c+m)}{l\left(\frac{4}{3} - m\cos^2 x_1/(m_c+m)\right)} u(t) + d(t) \quad (20)$$

Where (x_1) represents the angular position of the pendulum (rad), (x_2) represents the angular speed of the pole (rad/sec), (g) is the gravity acceleration (m/sec^2), (m_c) is the mass of the cart (kg), (m) is the mass of pole (kg), (l) is the half length of the pendulum pole (m), (u) represents the control input and $(d(t))$ denotes the external disturbances. The simulation parameters are as follows [15]:

$m_c = 1kg, m = 0.3kg,\ l = 0.5m\ and\ g = 9.8m/sec^2$.

The external disturbance of $\{d(t) = 0.01sin2t\}$ is applied to the inverted pendulum system and the position tracking reference input is (0.3sinwt), where (w) equals to (6 rad/sec) and the sampling time is TS=0.001sec. To train this model, the following SNN parameters are selected: τ= 7msec, ϑ = 0.8, T_{max}= 9msec and T_{min}=1 msec. The delay interval of the connection is 2 msec. Thus; the synaptic delays are from (1 to 3) msec. For SSO algorithm, the following parameters are selected: Ns= 50 spiders, the upper and lower initial weights are within (-50, 50). After several trials, it was found that the best structure of SNN contains six neurons in hidden layer with three synapses for each node. As a result, there are 72 weights to be trained. The closed loop controlled system is selected to work as a PID-like SNN controller (c1=1 and c2=1). The best weights obtained from SSO algorithm are listed in Table (3). The MSE reaches the best value (0.1335) after (90) iterations. Figure (7) shows the control signal (the output of SNNC) of the inverted pendulum system that was controlled by the PID-like SNN controller. Figure (8) shows the angular position responses of the inverted pendulum which are exactly tracking the reference input that was applied. Therefore; the proposed SNN controller guarantees to control the system.

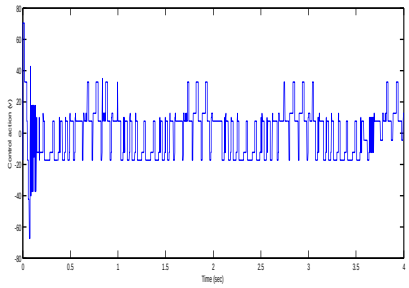

Figure .7. The control action of the inverted pendulum system that was controlled by the PID-like SNN controller.

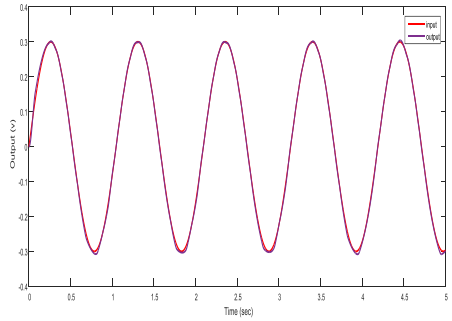

Figure.8. The input-output responses of the inverted pendulum system that was controlled by the PID-like SNN controller.

Table.3. the best weights for snnc for control an inverted pendulum.

w1 (:,:,1)=[42.6803 26.5726 27.3331 -20.9186 -23.8871 -23.2845 32.0021 -8.4035 -38.1886]	w1 (:,:,2) = [-36.4005 4.0047 34.8802 -20.2583 17.8603 34.9845 7.8522 18.3829 49.8394]
w1 (:,:,3) = [-2.6906 -8.6128 31.4090 12.8679 15.5132 30.0269 -23.2928 7.3580 -10.3885]	w1 (:,:,4) = [33.4459 43.5622 22.9469 11.0763 -22.2882 -24.6892 5.7131 -30.3702 -22.3560]
w1 (:,:,5) = [-23.0648 -2.1672 -16.4141 -26.4734 9.5837 -25.9788 -6.2076 3.7079 12.8783]	w1 (:,:,6) = [0.4714 8.2885 9.3848 -31.3674 -46.4987 7.7122 -13.0544 -9.9014 10.1425]
w2 = [8.9884 19.8057 36.7264 -16.4706 18.3447 29.1239 16.9872 13.2994 -15.0626 21.3109 -14.4446 -0.2203 21.5875 36.4610 -25.2386 0.6163 -11.8150 -9.7447]	

B. Conversion of SNNC into VHDL and Verilog Codes

The proposed SNNC program were written by MATLAB, after that converted into VHDL using HDL coder. There are many restrictions and limitations in MATAB functions when using HDL coder technique, like trigonometric functions, do-while, for loops, do-while, continue, break, if statement, floating point and variable size matrix. Therefore; these limitations were solved by designing MATLAB functions that overcome the HDL coder restrictions. As a result, the VHDL code become small and the implementation achieves minimum size of FPGA chip and as a result, less execution time. The steps for generating VHDL code for SNN controller are listed in the following:

- Take the MATLAB/Simulink of the proposed SNNC and put it inside the best weights that were obtained from the SSO algorithm and the best SNN parameters that were obtained from trial and error method.
- Specify the bits number for input and output which must be fixed point precision.
- Select from MATLAB/Simulink window, code then choose HDL code then select generate HDL.

The block diagram of the proposed SNN controller for the nonlinear inverted pendulum model after reducing the numbers of inputs became as in Figure (9). This block contains MATLAB function (Previous SNN controller) with discrete time integral and derivative blocks. The derivative block is not supported for HDL coder. Therefore, it is constructed by subtracting the delay from the input and dividing the answer by the sampling time (TS). In fact, this block diagram is converted into HDL coder. The HDL Code generation check report is presented in Figure (10). Also, the samples of the generated VHDL code for the controller that controls the inverted pendulum model are shown in Figure (11). It can be noticed from the check report that there are no errors, warnings, and messages.

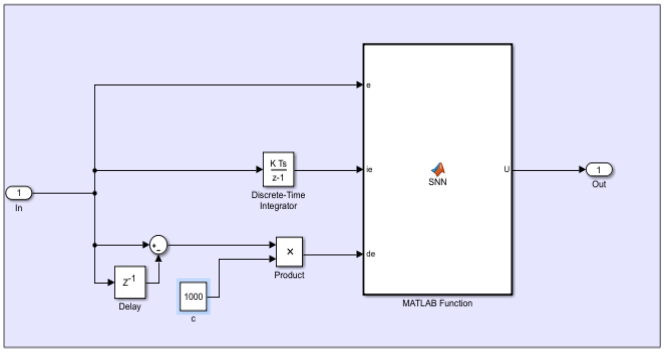

Figure.9. General block diagram of the proposed SNNC for the nonlinear inverted pendulum model that was converted into HDL coder.

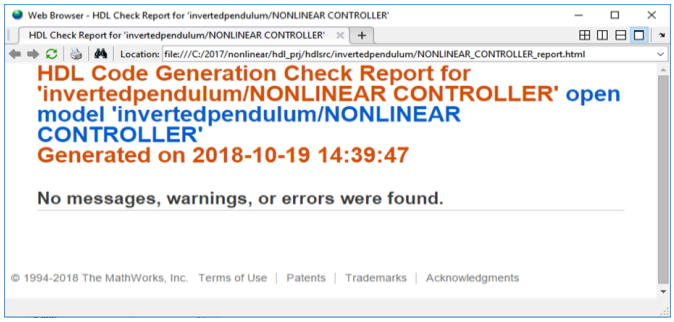

Figure.10. Check report for the linear model for the generated HDL Code.

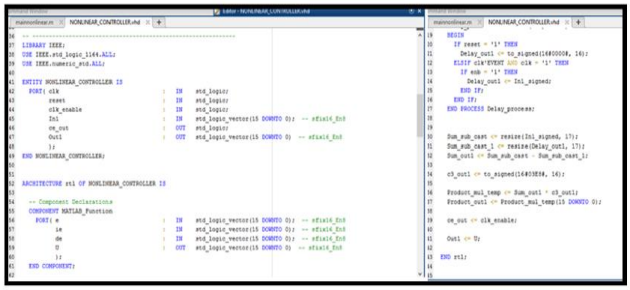

Figure.11. Sample of the generated VHDL code for the SNN controller that controls the inverted pendulum model.

6. CONCLUTIONS

In this paper, the SNNC is designed to control the inverted pendulum system. It is possible to work with different structures; as a P, PI, PD or PID like. It is using position tracking reference input signal is (0.3sinwt) and an external disturbance of {d(t)=0.01sin2t} is applied to the inverted pendulum system. The design is performed with small network size and as a result fast feed-forward calculations in SNN as compared with second generation neural network. Therefore; the hardware codes for SNNC becomes small and then need small FPGA chip size. SSO algorithm succeeded to train and find the (72) best weights of the SNN with minimum number of iterations and more speed to reach the target goal as compared with different training algorithm for SNN. The MSE reaches the best value (0.1335) after (90) iterations. Moreover, Simulation results proved the efficiency of suggested controller to reach accurate responses with minimum Mean Squared Error, small structure and minimum number of epochs under disturbances conditions. . Also the program is written with minimum steps and it avoided the restrictions and limitations in HDL coder to perform minimum FPGA chip size in implementation. The hardware code for VHDL to implement the proposed SNNC have been generated successfully.

7. REFERENCES

[1].M. Maganda, "A High Performance Hardware Architecture for Multilayer Spiking Neural Networks", Ph.D. Thesis, Department of Computer Science, National Institute for Astrophysics, Optics and Electronics, Tonantzintla, 2009.

[2]. H. M. Zawbaa, E. Emary, A. E. Hassanien, and B. Parv, "A Wrapper Approach for Feature Selection Based on Swarm Optimization Algorithm Inspired in the Behavior of the Social-Spider", 7th International Conference on Soft Computing and Pattern Recognition, pp. 25-30, 2015.

[3]. S. Cawley, F. Morgan, B. McGinley, S. Pande, L. McDaid, S. Carrillo, and J. Harkin, "Hardware Spiking Neural Network Prototyping and Application", Genetic Programming and Evolvable Machines, Vol. 12, No. 3, pp. 257–280, 2011.

[4]. F. Xue, W. Wang, N. Li, and Y. Yang, "FPGA Implementation of Self-Organized Spiking Neural Network Controller for Mobile Robots", Advances in mechanical Engineering, Hindawi Publishing Corporation, 2014.

[5]. A. Zbrzeski, Y. Bornat, B. Hellin, R. Siu, J. Abbas, R. Jung and S. Renaud, "Bio-Inspired Controller on an FPGA Applied to Closed-Loop Diaphragmatic Stimulation", Frontiers in Neuroscience, Vol. 10, 2016.

[6]. M. J. Pearson C. Melhuish, A. Pipe, M. Nibouche, I. Gilhesphy, K. Gurney, B. Mitchinson, "Design and FPGA Implementation of an Embedded Real-Time Biologically Plausible Spiking Neural Network Processor", IAS laboratory in University of the West of England, IEEE, 2005.

[7]. Y. Oniz, O. Kaynak, and R. H.Abiyev, "Spiking Neural Networks for the Control of a Servo System", International Conference on Mechatronics (ICM), pp. 94-98, Feb. 27-March 1, 2013.

[8]. F. Morgan, S. Cawley, J. Harkin, B. McGinley, L. McDaid and S. Pande, "An Evolvable NoC-Based Spiking Neural Network Architecture", Signals and Systems Conference (ISSC 2009), IET Irish, Dublin, pp. 1-6, 2009.

[9]. W. Maass, "Networks of Spiking Neurons the Third Generation of Neural Network Models", Neural Networks, Vol. 10, No. 9, pp. 1659-1671, 1997.

[10]. B. Ginley, P. Rocke, F. Morgan and J. Maher, “Reconfigurable Analogue Hardware Evolution of Adaptive Spiking Neural Networks Controllers”, In Proceedings GECCO’08, pp. 289–290, 2008.

[11]. K. L. Rice, M. A. Bhuiyan, T. M. Taha, C. N. Vutsinas and M. C. Smith, "FPGA Implementation of Izhikevich Spiking Neural Networks for Character Recognition", Proceeding of Reconfig‘09, Mexico, pp. 451–456, Dec. 2009.

[12]. Maxfield Clive, " FPGAs: Instant Access", Amsterd Jam, 2008.

[13]. Altera Coporation, DE2 Development and Education Board Manual, Version 1.4, Altera Corporation, 2006.

[14]. J. Noh, G. Lee, H. Choi and S. Jung, "Robust Control of a Mobile Inverted Pendulum System using RBF Neural Network Controller", IEEE ROBIO, pp. 1932-1937, 2008.

[15]. H. Pang and Q. Yang, "Optimal Sliding Mode Control for a Class of Uncertain Nonlinear Systems Based on Feedback Linearization", Robust Control, Theory and Applications, pp. 141-162, 2001.