Journal of Information Security, 2014, 5, 154-165
Published Online October 2014 in SciRes. http://www.scirp.org/journal/jis
http://dx.doi.org/10.4236/jis.2014.54015

Scientific Research

Hardware Realization of Artificial Neural Network Based Intrusion Detection & Prevention System

Indraneel Mukhopadhyay, Mohuya Chakraborty

Department of Information Technology, Institute of Engineering & Management, Kolkata, India
Email: imukhopadhyay@gmail.com, mohuyacb@yahoo.com

Received 24 July 2014; revised 20 August 2014; accepted 13 September 2014

Abstract

In the 21st century with the exponential growth of the Internet, the vulnerability of the network which connects us is on the rise at a very fast pace. Today organizations are spending millions of dollars to protect their sensitive data from different vulnerabilities that they face every day. In this paper, a new methodology towards implementing an Intrusion Detection & Prevention System (IDPS) based on Artificial Neural Network (ANN) onto Field Programmable Gate Array (FPGA) is proposed. This system not only detects different network attacks but also prevents them from being propagated. The parallel structure of an ANN makes it potentially fast for the computation of certain tasks. FPGA platforms are the optimum and best choice for the modern digital systems nowadays. The same feature makes ANN well suited for implementation in FPGA technology. Hardware realization of ANN to a large extent depends on the efficient implementation of a single neuron. However FPGA realization of ANNs with a large number of neurons is still a challenging task. The proposed multilayer ANN based IDPS uses multiple neurons for higher performance and greater accuracy. Simulation of the design in MATLAB SIMULINK 2010b by using Knowledge Discovery and Data Mining (KDD) CUP dataset shows a very good performance. Subsequently MATLAB HDL coder was used to generate VHDL code for the proposed design that produced Intellectual Property (IP) cores for Xilinx Targeted Design Platforms. For evaluation purposes the proposed design was synthesized, implemented and tested onto Xilinx Virtex-7 2000T FPGA device.

Keywords

Artificial Neural Network, Feed Forward Multilayer ANN, Intrusion Detection & Prevention System, FPGA, VHDL, Virtex 7

How to cite this paper: Mukhopadhyay, I. and Chakraborty, M. (2014) Hardware Realization of Artificial Neural Network Based Intrusion Detection & Prevention System. *Journal of Information Security*, **5**, 154-165.
http://dx.doi.org/10.4236/jis.2014.54015

1. Introduction

Intrusion prevention is the process of performing intrusion detection and attempting to stop detected possible incidents. Intrusion Detection & Prevention Systems (IDPS) [1] are primarily focused on identifying possible incidents, logging information about them, attempting to stop them, and reporting them to security administrators. IDPS perform extensive logging of data that is related to detected events in the network. These data can then be used to confirm the validity of alerts, investigate incidents, and correlate events between the IDPS and other logging sources. Artificial neural networks (ANN) have found widespread deployment in a broad spectrum of classification, perception, association and control applications [2]. The aspiration to build intelligent systems complemented with the advances in high speed computing has proved through simulation the capability of ANN to map, model and classify nonlinear systems. Real time applications are possible only if low cost high-speed neural computation is made realizable. Towards this goal numerous works on implementation of ANN have been proposed [2].

ANNs have been mostly implemented in software. However, a disadvantage in real-time applications of software-based ANNs is slower execution compared to hardware-based ANNs. Hardware-based ANNs have been implemented as both analogue and digital circuits. Recent advances in reprogrammable logic enable implementing large ANNs on a single field-programmable gate array (FPGA) device. The main reason for this is the miniaturization of component manufacturing technology, where the data density of electronic components doubles every 18 months [3]. ANNs are biologically inspired and require parallel computations in their nature. Microprocessors and Digital Signal Processors (DSPs) are not suitable for parallel designs. Designing fully parallel modules may be possible with Application Specific Integrated Circuits (ASICs) for which the best example is Vertex 7 FPGA boards. In addition the design results in an ANN suited only for one target application. FPGAs not only offer parallelism but also flexible designs, savings in cost and design cycle [4]. Our idea is to realize an IDPS based on feed forward ANN on FPGA.

The organization of the paper is as follows. After the introduction in Section 1, Section 2 provides an overview of the proposed architecture of ANN based IDPS. Theoretical background study of feed forward ANN and IDPS implementation in FPGA have been described in Section 3. Section 4 describes the MATLAB implementation and simulation of the proposed ANN based IDPS model. Simulation environment of the proposed design followed by its performance analysis using VHDL XILINX ISE 13.2 design suite have been presented in Section 5. In Section 6 the hardware implementation of the proposed architecture on Virtex7 FPGA board followed by results are presented. Section 7 concludes the paper with some highlights on future work.

2. Proposed Architecture of ANN Based IDPS

The idea is to design an IDPS based on feed forward ANN on FPGA. The proposed IDPS architecture consists of two blocks—an Intrusion Detection System (IDS), which classifies the attack and the Intrusion Prevention System (IPS), which takes the classified signature and decides which signatures to be propagated. The block diagram shown in **Figure 1** represents the architecture of our proposed IDS. The data collected from the Internet is fed into the pre-processing unit where the raw data is formatted to make it compatible with the intrusion detection system under consideration. Subsequently data is classified as normal or attack. Normal data is allowed to pass through while on the other hand attack data is classified to the type of attack, saved in the database (Db) and an alert is raised [5]. The proposed IDS architecture may be divided into four sub-processes whose functionalities are briefly described below.

1. Data Collector: KDD 99 data set based on the DARPA intrusion detection programme, which is publicly accessible via MIT Lincoln Lab is first of all collected at this block.
2. Pre-processor: This block takes the original data from the MIT Lincoln Lab, extracts the required features, and converts the data set into Matlab compatible format. This basically performs the data cleaning procedure.
3. Encoder: The attributes given in the data set are converted into double data type to make it compatible with the ANN Tool box of Matlab. The authors have converted the feature variables “protocol type” with values like tcp = 0, udp = 1, icmp = 2; “error flag” with values S0 = 0, SF = 1, S1 = 2, REJ = 3, S2 = 4. The attacks in the data set have been categorized as normal = 0, DoS = 1, probe = 2, r2l = 3, u2r = 4, others = 5. The details of all 41 attributes are given in **Appendix**.
4. Neural Network Classifier: The data at the output of the encoder stage is fed into the neural network with following experimental criteria to properly classify attack types.

IPS on the other hand is a software programme that stops possible attack signatures. Therefore once the classification is done by the IDS the IPS takes the classified signature and decides which signatures to be propagated. If the signature is classified as an attack signature then the network traffic is not allowed otherwise if the signature is classified as normal then the signature is propagated as shown in **Figure 2**.

3. Theoretical Background

3.1. Feed Forward ANN

ANNs are inspired by the biological neural systems. The transmission of signals in biological neurons through synapses is a complex chemical process in which specific transmitter substances are released from the sending side of the synapse. The effect is to raise or lower the electrical potential inside the body of the receiving cell. If this potential reaches a threshold, the neuron fires. It is this characteristic of the biological neurons that the artificial neuron model proposed by McCulloch Pitts attempts to reproduce [6]. Following neuron model shown in **Figure 3** is widely used in artificial neural networks with some variations. The artificial neuron given in this figure has N inputs, denoted as x_1, x_2,…,x_N. Each line connecting these inputs to the neuron is assigned a weight, denoted as w_1, w_2,…,w_N respectively. The activation a, which determines whether the neuron is to be fired or not is given by Equation (1).

$$a = \left(\sum_{j=1}^{N} ww_j x_j \right) \tag{1}$$

A negative value for a weight indicates an inhibitory connection while a positive value indicates excitatory connection. The output, y of the neuron is given by Equation (2).

$$y = f(a) \tag{2}$$

Originally the neuron output function $f(a)$ in McCulloch Pitts model was proposed as threshold function,

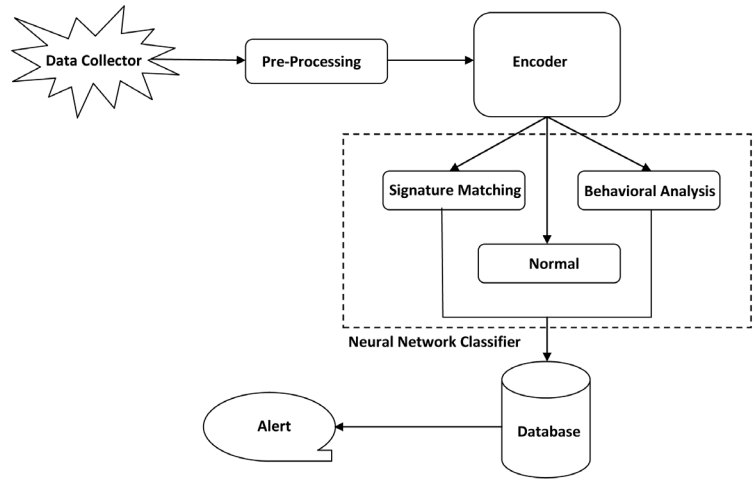

Figure 1. Architecture of intrusion detection system.

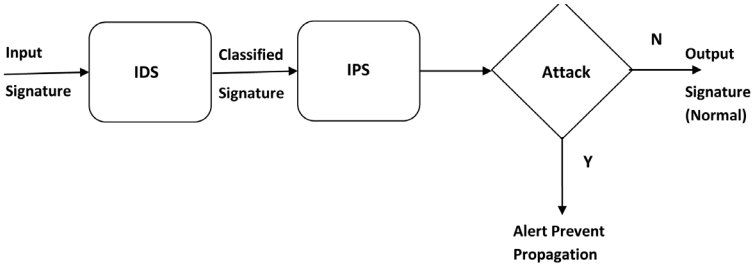

Figure 2. Proposed intrusion detection and prevention system architecture.

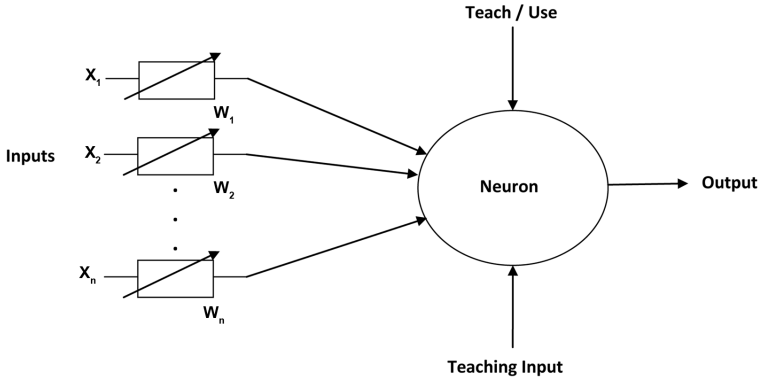

Figure 3. Artificial neural network.

however linear, ramp, and sigmoid functions are also used in different situations. The vector notation as shown in Equation (3) may be used for expressing the activation of a neuron.

$$a = w^T x \tag{3}$$

Here, the j_{th} element of the input vector x is x_j, the j_{th} element of the weight vector of w is w_j. Both of these vectors are of size N. A Neuro-computing system is made up of a number of artificial neurons and a huge number of interconnections between them. **Figure 4** shows architecture of feed forward neural network. In layered neural networks, the neurons are organized in the form of layers [7]. The neurons in a layer get inputs from the previous layer and feed their output to the next layer. These types of networks are called feed forward networks.

Output connections from a neuron to the same or previous layer neurons are not permitted. The input layer is made up of special input neurons, transmitting only the applied external input to their outputs. The last layer is called the output layer, and the layers other than input and output layers are called the hidden layers. A single layer network consists of solely input and output layers without a hidden layer. Networks with one or more hidden layers are called multilayer networks. Multilayer perceptron is a well-known feed forward layered network, on which the Back propagation learning algorithm is widely implemented. The structures, where connections to the neurons are to the same layer or to the previous layers are called recurrent networks. Hopfield and Boltzmann Machine are examples of widely used recurrent networks [8].

3.2. IDPS Implementation on FPGA

The field of network security guarantees the prevention and monitoring of unauthorized access, misuse, modification as well as denial of network accessible resources. The major goals of network security include confidentiality, data integrity, authentication and non repudiation. In the same context, intrusion detection is a security management tool which monitors network traffic for detecting possible security breaches. These security breaches attempt to compromise the confidentiality, integrity or availability of network resources and can be either from outside or inside the network concerned. In traditional networks, firewalls are used to monitor and filter incoming and outgoing packets but they cannot eliminate all security threats, nor can they detect attacks when they happen. It is like a locked gate to a treasure house that prevents the entry of thieves.

IDPS is another network processing application, which is either a software application (example Snort) [9] or a hardware device that monitors network for malicious activities such as denial of service attacks, port scans etc. These IDPS are essential network security appliances that help in maintaining the security goals in a network to a great extent. IDPS are primarily focused on identifying possible incidents, logging information about them, attempting to stop them, and reporting them to security administrators. As the main work of such systems is to monitor network traffics for suspicious activities or patterns, they can be regarded as a multiple pattern matching module. Pattern matching algorithm is thus one of the most critical in such systems and the detection of intruders is performed based on this module. At the higher level, there are management software's that configure log and display alarms. A database for a number of malicious patterns is maintained at the back end. Whenever any packet containing such malicious patterns is found during packet monitoring, the detector engine of IDPS raises an alert call to the administrator for taking necessary action against the target packet. Although the decisive fac-

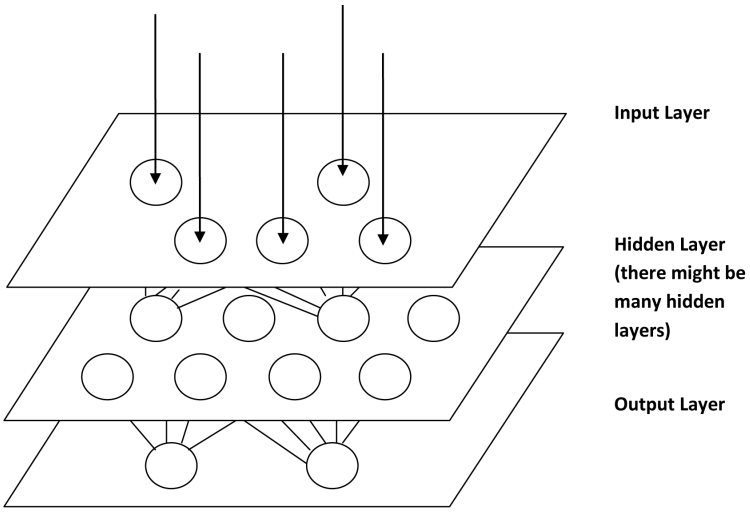

Figure 4. Feed forward neural network.

tor for an IDPS is a pattern matching algorithm, it is a highly challenging task to implement. It is because the operation of a typical IDPS involves deep packet inspection [10]. Checking every byte of an incoming packet in a network to see if it is matching with one of a set of thousands of patterns becomes a computationally intensive task. Moreover if it is a high speed network, packet inspection needs to be performed in line speed which is more challenging. Software based approach is not efficient in terms of speed and moreover parallelism cannot be exploited in case of multiple pattern matching.

So specialized hardware approaches are required to maintain match-up with the network speed and to maintain a tight bound on worst case performance. Here, a new architecture for a pattern matching based IDPS is presented. The aim of our work is to design the ANN-based IDPS and implement it on Virtex 7 FPGA board.

4. MATLAB Implementation of the Proposed IDPS Architecture

The processing element of an ANN is the neuron. A neuron can be viewed as processing data in three steps; the weighting of its input values, the summation of them all and their filtering by sigmoid function. The summation can be calculated by a serial accumulation. For the weighted inputs to be calculated in parallel using conventional design techniques, a large number of multiplier units would be required. To avoid this, Multiplier/Accumulator architecture has been selected [11].

It takes the input serially, multiplies them with the corresponding weight and accumulates their sum in a register. The processes are synchronized to clock signal. The number of clock cycles for a neuron to finish its work, equals to the number of connections from the previous layer. The accumulator has a load signal, so that the bias values are loaded to all neurons at start-up.

To implement our proposed architecture we have selected MATLAB SIMULINK version 2010b as an environment as it supports multi-domain simulation and model-based design for dynamic and embedded systems. It provides an interactive graphical environment and a customizable set of block libraries that allows to design, simulate, implement, and test a variety of time-varying systems, including communication, control, signal processing, video processing, image processing etc. Two architectures were built in MATLAB: *single layer and double layer*. We have used 6 and 10 neurons to train the system for single layer and double layer architectures respectively. During training of the ANN we have specified some values to detect attacks e.g., Normal 0; DoS 1; Probe 2; U2R 3; R2L 4; Other 5 (yet to be discovered attacks). After training our ANN with the given KDD CUP dataset we have generated the SIMULINK ANN Attack classifier model to classify the attack and the normal data separately. **Figure 5** shows the architecture and simulation of ANN based IDPS indicating an U2R attack. Here the input signature consists of the 41 attributes of our dataset and the IDPS block consists of 5 classified attack outputs along with the normal signature output [12].

The proposed IDPS architecture consists of two blocks—the ANN based Intrusion Detection System (IDS), which classifies the attack and the Intrusion Prevention System (IPS) designed using embedded MATLAB func-

tion, which takes the classified signature and decides which signatures to be propagated as indicated by the binary values of Intrusion Prevention (IP) status. **Figure 5** shows the simulation result for U2R attack signature. After classification by the IDS as U2R attack, the IPS prevents the signatures to be propagated indicated by the IP status of "0". For normal input signature the IPS allows the input to be propagated indicated by the IP status of "1".

The performance of the system was found to be satisfactory. We had tested the system taking 10% of the data from the KDD Cup dataset [13]. **Figure 6** shows the False Positive Graph taking into account number of Input Signature (Normal) vs. Number of False positives generated. It is clear from the graph that false positive is within the expected range of 10% to 13%. It however increased to 16% on increasing the number of normal signatures beyond 100.

5. VHDL Modeling of the Proposed IDPS

The ANN-based IDPS model designed in MATLAB was transformed into VHDL model from MATLAB environment and was simulated using Xilinx ISE 13.2 ISim (ISE Simulator). ISim provides a complete, full-featured HDL simulator integrated within Xilinx ISE. HDL simulation now will be a fundamental step within our design flow with the tight integration of the ISim within our design environment. After the initial setup we passed a

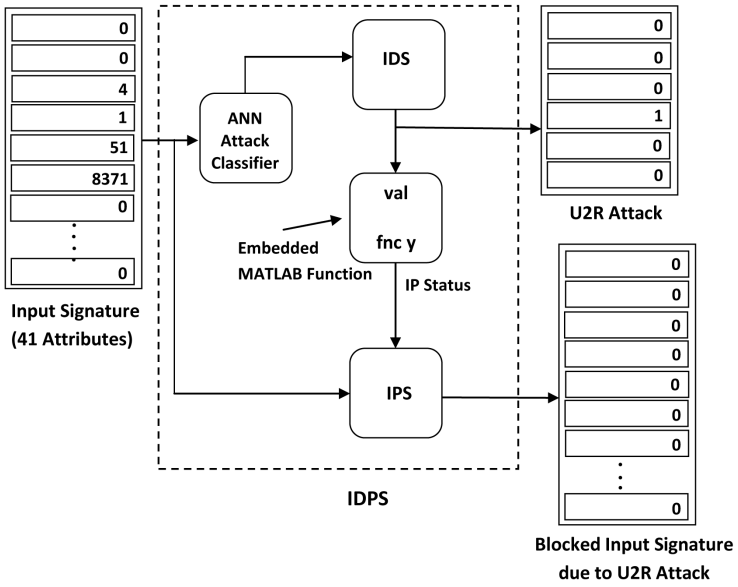

Figure 5. Proposed architecture and simulation of ANN based IDPS showing U2R attack.

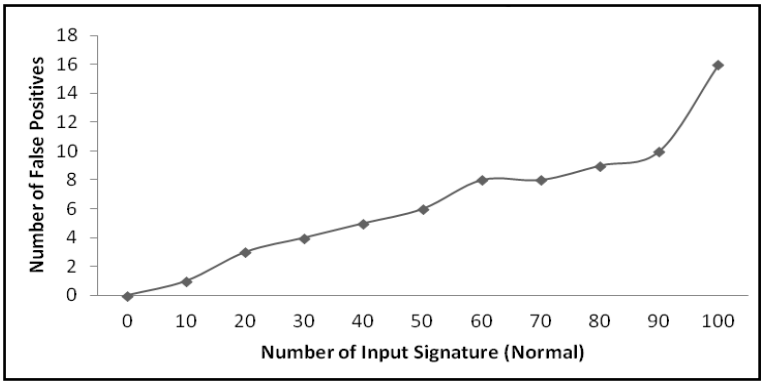

Figure 6. False positive graph.

normal packet through the design to check the behavior of the model. As predicted the output was same as the result obtained using MATLAB SIMULINK. We then passed different signatures and recorded the behavior of the design. Based on attack signatures we got different simulation results which have been presented in **Table 1** with respect to the kind of attack that was detected and subsequently prevented.

We used two types of models for your simulation. Model 1 takes in floating point input values, the actual values as obtained from KDD CUP dataset as signature of an attack. The result is depicted in **Table 1(a)**. In Model 2 we had rounded off the floating point input values to its nearest integer values so as to implement the proposed design on FPGA. Keeping that in mind we trained our ANN model and then executed our simulation output, which has been depicted in **Table 1(b)**. For both the simulations we have used the same set of input data. Due to the conversion of floating point values to integer values there was some loss of precision in the case of simulation results.

As per the model that was simulated the result has been shown with respect to the highest value in the simulation column. According to our simulation environment that was set up classifier detects the signature with the highest value in the simulation and based on the highest value the signature is classified into either Normal, DoS, Probe, U2R, R2L or Other.

If we look deeply into **Table 1(a)** we can see that simulation 1 and simulation 3 show the corresponding values for DoS signature, simulation 2 show the corresponding values for R2L attack signature, simulation 4 and simulation 5 show the corresponding values for normal signature as designed in our model, and simulation 6 shows the corresponding value for Probe attack signature.

When we execute the same dataset after converting the data to integer we have the following observations. Simulation 1 and simulation 3 show the corresponding values for DoS signature (same as **Table 1(a)**), simulation 2 shows the corresponding values for a normal signature which denotes false positive, and simulation 4 and simulation 5 show the corresponding values for normal signature. However simulation 4 also shows Probe attack which means false negative as designed in our model, and simulation 6 shows the corresponding value for Probe attack signature. The discrepancy in the values of **Table 1(a)** and **Table 1(b)** has been further discussed in this paper.

For comparison purpose we have designed, simulated and implemented the proposed architecture both as a single layer and as a double layer ANN model. Comparison of the both is shown in **Table 2**. When comparing

Table 1. (a) Simulation results for floating point values. (b) Simulation results for integer values.

(a)

	Simulation 1	Simulation 2	Simulation 3	Simulation 4	Simulation 5	Simulation 6
Normal	0	0	0	1	1	0
DoS	1	0	1	0	0	0
Probe	0	0	0	0	0	1
R2L	0	1	0	0	0	0
U2R	0	0	0	0	0	0
Others	0	0	0	0	0	0

(b)

	Simulation 1	Simulation 2	Simulation 3	Simulation 4	Simulation 5	Simulation 6
Normal	0	1	0	1	1	0
DoS	1	0	1	0	0	0
Probe	0	0	0	1	0	1
R2L	0	0	0	0	0	0
U2R	0	0	0	0	0	0
Others	0	0	0	0	0	0

single layer with double layer ANN model with same set of signatures using floating point as well as integer values we got 100% correct result for double layer ANN model in floating point mode as against 83.33% in integer mode. Similarly for single layer ANN model in floating point mode we got 96.66% correct result as against 70% for integer mode. This discrepancy is due to precision error while rounding off the floating point values to integer ones. Similarly there has been discrepancy between single and double layer ANN model with respect to False Positives and False Negatives as given in **Table 2**.

The proposed architecture allows only the packets of normal signature to pass through the system and stops any other attack signature [14]. Intrusion prevention is the process of performing intrusion detection and attempting to stop detected possible incidents. Intrusion detection and prevention systems (IDPS) are primarily focused on identifying possible incidents, logging information about them, attempting to stop them, and reporting them to security administrators. In our proposed architecture once the attack signature has been detected by the IDS, the IPS comes into play and the signature is stored in our database, compared with a reference value and IP status is set as binary value of "1" to pass the normal signature and a binary value of "0" to block the attack signature. **Figure 7** shows the simulation result for a normal signature where IP status is set to "1" to pass the traffic between 1 ps and 2 ps interval. The figure also shows dos attack during the interval 2 ps to 3 ps where IP Status goes low to block the traffic.

6. FPGA Implementation of the Proposed ANN Based Architecture

After simulation of the proposed architecture in ISim we have synthesized, implemented and tested the model onto Virtex-7 2000T FPGA device. The system requirement is Intel Core 2 Duo at 2.5 GHz with 4 GB RAM. Virtex-7 2000T combines four smaller FPGAs into a single package by placing them on a special silicon interconnection pad to deliver 6.8 billion transistors in a single large chip. The interposer provides 10,000 data pathways between the individual FPGAs—roughly 10 to 100 times more than would usually be available on a board to create a single FPGA [15]. We have done a comparative analysis between single layer and double layer ANN implementation on Virtex7 2007 T with respect to resource parameters as shown in **Table 3**.

Table 2. Comparison between single & double layer ANN based IDPS.

	Single Layer ANN		Double Layer ANN	
	Floating Point Values	Integer Values	Floating Point Values	Integer Values
Dataset Result (%)	96.66	70	100	83.33
False Positive (%)	10	6	3	10
False Negative (%)	10	10	23	10

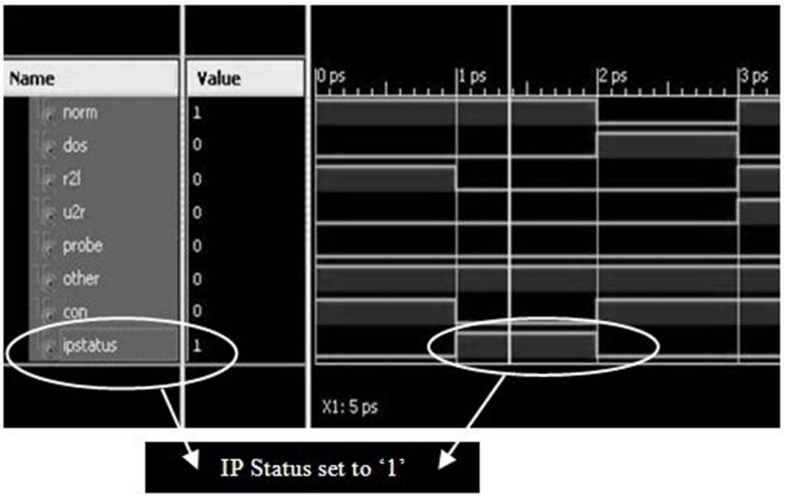

Figure 7. Simulation of normal signature.

The register-transfer level (RTL) schematics sub-part for the single layer and double layer ANN models of IDS are shown in **Figure 8** and **Figure 9** respectively. RTL abstraction is used in hardware description languages (HDLs) like Verilog and VHDL to create high-level representations of a circuit, from which lower-level representations and ultimately actual wiring can be derived. Design at the RTL level is a typical practice in modern digital design [16].

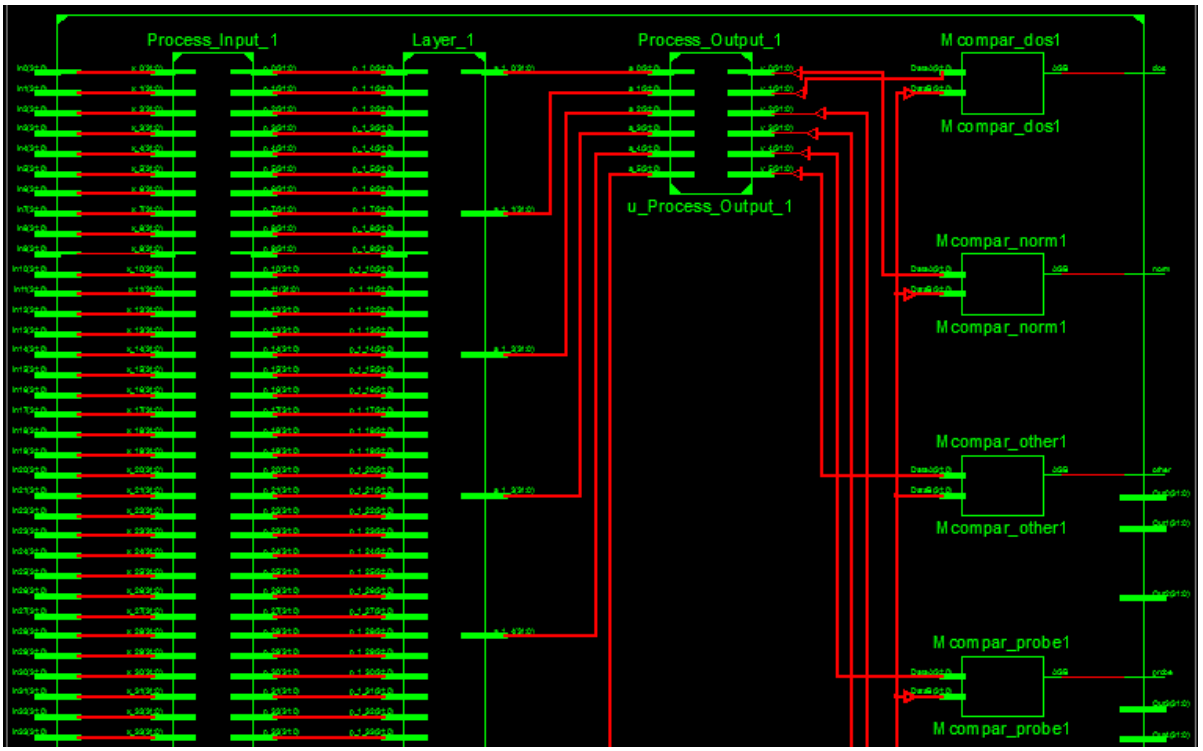

Figure 8. RTL schematic sub-part of single Layer ANN Model.

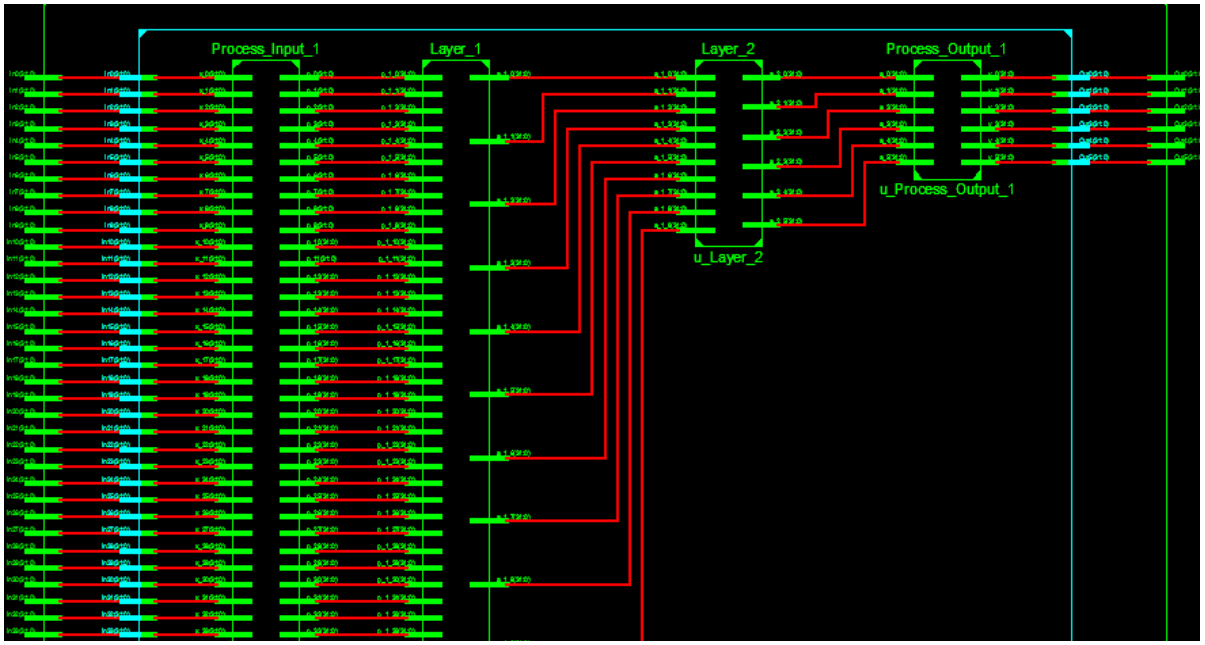

Figure 9. RTL schematic sub-part of double layer ANN Model.

Table 3. Comparative analysis between Single & Double Layer ANN based IDPS Model with respect to Resources used on a Virtex7 FPGA board.

	Single Layer Integer ANN Model	% Age of Resources Used	Double Layer Integer ANN Model	% Age of Resources Used	Resources Available in Virtex 7 FPGA
Memory Usage	120 MB		194 MB		
Time Required	24 sec		2 min, 21 sec		
No. of Slice LUT's	8337	0	14169	1	1221600
Number of Fully Used LUT-FF Pair	0	0	0	0	8336
Number of Bonded IOB's	793	66	989	83	1200
Number of DSP48E1's	6	0	865	40	2160

7. Summary and Future Work

The proposed IDPS model using artificial neural network was verified in MATLAB environment using Simulink. The design was later converted to VHDL environment using Xilinx Project Navigator 13.2 and simulated using ISim. In the next step the proposed design was implemented on Virtex-7 2000T FPGA. In all the cases the performance of the design was within the acceptable range.

Future work will include further classification of different types of attacks to further subtypes as and when signatures are available and then integrate it to the proposed architecture so that the model can be robust in nature. Next step will be to evaluate the performance of the proposed architecture in real-time environment.

In synopsis, we have provided a better understanding of the design principles and implementation techniques for building high-speed, reliable, robust and scalable ANN-based network intrusion detection and prevention system that is highly useful for mankind and organizations (e.g., military, defense etc.,) in today's world of network defenselessness.

References

[1] Scarfone, K. and Mell, P. (2007) Guide to Intrusion and Prevention System (IDPS). Department of Commerce, National Institute of Standard and Technology, Technology Administration. http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf

[2] Widrow, B., Rumelhart, D.E. and Lehr, M.A. (1994) Neural Networks: Applications in Industry, Business and Science. *Communications of the ACM*, **37**, 93-105. http://dx.doi.org/10.1145/175247.175257

[3] Muthuramalingam, A., Himavathi, S. and Srinivasan, E. (2008) Neural Network Implementation Using FPGA: Issues and Application. *The International Journal of Information Technology*, **4**, 86-92.

[4] Tommiska, M.T. (2003) Efficient Digital Implementation of the Sigmoid Function for Reprogrammable Logic. *IEEE Proceedings of Computers and Digital Techniques*, **150**, 403-411.

[5] Mukhopadhyay, I., Chakraborty, M. and Chakarbarti, S. (2011) A Comparative Study of Related Technologies of Intrusion Detection & Prevention Systems. *Journal of Information Security*, **1**, 28-38. http://dx.doi.org/10.4236/jis.2011.21003

[6] Savran, A. and Ünsal, S. (1999) Hardware Implementation of a Feed forward Neural Network Using FPGAs'. Department of Electrical and Electronics Engineering, Ege University.

[7] Haykin, S. (1999) Neural Networks—A Comprehensive Foundations. Prentice-Hall International, New Jersey.

[8] Rai, C.S. and Singh, A.P. (2006) A Review of Implementation Techniques for Artificial Neural Networks. University School of Information Technology, GGS Indraprastha University, Delhi.

[9] (2014) SNORT User Manual. http://www.snort.org/

[10] Chang, R.-I, Lai, L.-B., Su, W.-D., Wang, J.-C. and Kouh, J.-S. (2007) Intrusion Detection by Back Propagation Neural Networks with Sample-Query and Attribute-Query. *International Journal of Computational Intelligence Research*, **3**, 6-10.

[11] Dharmapurikar, S., Krishnamurthy, P., Sproull, T., Lockwood, J. and Speeds, L. (2004) Deep Packet Inspection Using Parallel Bloom Filters.

[12] Mukhopadhyay, I., Chakraborty, M., Chakrabarti, S. and Chatterjee, T. (2011) Back Propagation Neural Network Approach to Intrusion Detection System. *The Proceeding of International Conference on Recent Trend in Information*

System (*ReTIS*-11), Kolkata, 21-23 December 2011, 303-308.

[13] KDD Cup (1999) Computer Network Intrusion Detection. http://www.sigkdd.org/kdd-cup-1999-computer-network-intrusion-detection

[14] Mukhopadhyay, I., Chakraborty, M. and Chatterjee, T. (2012) Artificial Neural Network Modeling of Intrusion Detection & Prevention System. *IEM International Journal of Management & Technology*, **2**.

[15] Manners, D. (2011) Electronics Weekly. Xilinx Launches 20m ASIC Gate Stacked Silicon FPGA.

[16] Vahid, F. (2010) Digital Design with RTL Design, Verilog and VHDL. 2nd Edition, John Wiley and Sons, 247.

Appendix

	Feature	Description	Type		Feature	Description	Type
1	Duration	Duration of the connection	Cont.	22	Is guest login	1 if the login is a “guest” login; 0 otherwise	Disc.
2	Protocol type	Connection protocol (e.g. tcp, udp)	Disc.	23	Count	Number of connections to the same host as the current connection in the past two seconds	Cont.
3	Service	Destination servfice (e.g. telnet, tfp)	Disc.	24	srv count	Number of connections to the same service as the current connection in the past two seconds	Cont.
4	Flag	Status flag of the connection	Disc.	25	serror rate	% of connections that have “SYN” errors	Cont.
5	Source bytes	Bytes sent from source to destination	Cont.	26	srv serror rate	% of connections that have “SYN” errors	Cont.
6	Destination bytes	Bytes sent from destination to source	Cont.	27	rerror rate	% of connections that have “REJ” errors	Cont.
7	Land	1 if connection is from to the same host; 0 otherwise	Disc.	28	srv rerror rate	% of connections that have “REJ” errors	Cont.
8	Wrong fragment	Number of wrong fragments	Cont.	29	same srv rate	% of connections to the same service	Cont.
9	Urgent	Number of urgent packets	Cont.	30	diff srv rate	% of connections to the different services	Cont.
10	Hot	Number of “hot” indicators	Cont.	31	srv diff host rate	% of connection to different hosts	Cont.
11	Failed login	Number of failed logins	Cont.	32	dst host count	Count of connections having the same destination host	Cont.
12	Logged in	1 if successfully loggedin; 0 otherwise	Disc.	33	dst host srv count	Count of connections having the same destination host and using the same service	Cont.
13	# compromised	Number of “compromised” conditions	Cont.	34	dst host same srv rate	% of connections having the same destination host and using the same service	Cont.
14	Root shell	1 if root shell is obtained; 0 otherwise	Cont.	35	dst host diff srv rate	% of different service on the current host	Cont.
15	Su attempt	1 if “su root” command attempt; 0 otherwise	Cont.	36	dst host same src port rate	% of connections to the current host having the same src port	Cont.
16	# root	Number of “root” accesses	Cont.	37	dst host srv diff host rate	% of connections to the same service coming from different hosts	Cont.
17	# file creations	Number of file creation operations	Cont.	38	dst host serror rate	% of connections to the current host that have an S0 error	Cont.
18	# shells	Number of shell prompts	Cont.	39	dst host srv serror rate	% of connections to the current host and specified service that have an S0 error	Cont.
19	# access files	Number of operations on access control files	Cont.	40	dst host rerror rate	% of connections to the current host that have an RST error	Cont.
20	# outbound cmds	Number of outbound commands in an ftp session	Cont.	41	dst host srv reeror rate	% of connections to the current host and specified service that have an RST error	Cont.
21	Is hot login	1 if the login belongs to the “hot” list; 0 otherwise	Disc.				