# VLSI Implementation of Neural Network Based Emergent Behavior Model for Robot Control

Harsh Advani, Jimmy Patel, Subhadeep Paul, and Tapas Kumar Maiti
Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, Gujarat 382007, India
email address: tapas_kumar@daiict.ac.in

***Abstract*—This paper reports the VLSI implementation of NN (Neural Network) based emergent behavior model for high-speed robot control. Augmented FSM (Finite-State Machine) is considered to implement the emergent behavior. We performed a system level simulation using our proposed model. Then, we transformed the model to RTL (Register-Transfer Level) for circuit simulation. In this study, we considered multiple-inputs and multiple-outputs NN. Our implementation method improves speed of execution and accuracy and compare the result with conventional neural network. For activation function in NN, we implemented sigmoid function with second order approximation to reduce complexity. We used walking gesture of Kondo KHR-3HV robot to verify the model.**

***Index Terms*—Emergent behavior, robot control, VLSI, Neural Network, system**

## I. Introduction

OVER the past few decades, our society is moving towards to fully automation whether it's in manufacturing, production, surveillance, security etc. Now a days for automation, AI (Artificial Intelligence) and neural networks are used for classification and identification [1], [2]. Brooks *et al*., reported a new method for the application of AI to robotics, known as emergent behavior-based robotics [3]. Lowest level behavior was represented by AFSM (Augmented Finite-State Machine). However, for a complex robotic system, knowledge based behavior is also required for full control which can be implemented as a neural network [2]. In this work, we considered a hybrid approach which is the combination of both knowledge-based behavior and sensor control priority-based behavior based on AFSM as illustrated in Fig. 1. The main objective of this work is the VLSI implementation of behavioral model with lesser delay and complexity compared to conventional implementation to speed-up the robot control. Normally, implementation of behavioral-based AI model using python or MATLAB results higher processing time (~millisecond) in robot control. However, implementation at hardware level the delay could be in nanoseconds. This work focus on the implementation of emergent behavior model at hardware level. To implement the NN, we approximated sigmoid function instead of conventional method, obtain almost same S-curve with lesser complexity. We implemented AFSM with respect to four state of sensor inputs. Session II, describes the conventional level implantation of emergent behavior model which is the combination of AFSM and NN. The RTL level implementation of NN is described in Sec. IV. Details of Sigmoid function approximation for VLSI implementation is described in Sec. III.

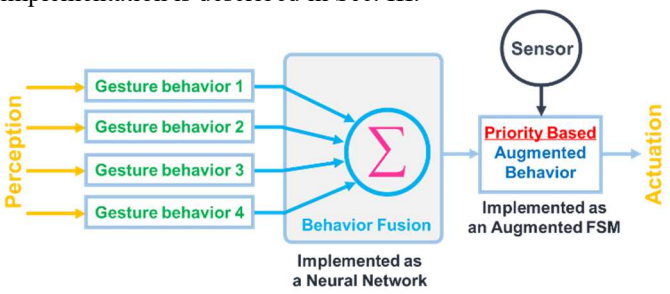


Fig. 1. Proposed behavior-based architecture implemented using a neural network (NN) and an augmented finite-state machine (AFSM).

## II. System Level Implementation of Neural Network

ANNs (Artificial Neural Networks) usually known as NNs are computing systems inspired by the biological neural network that contains animal brains [1]. In ANN, we used artificial neuron to perform different mathematical functions like transfer function, sigmoid function, summation, multiplication as depicted in Fig. 2.

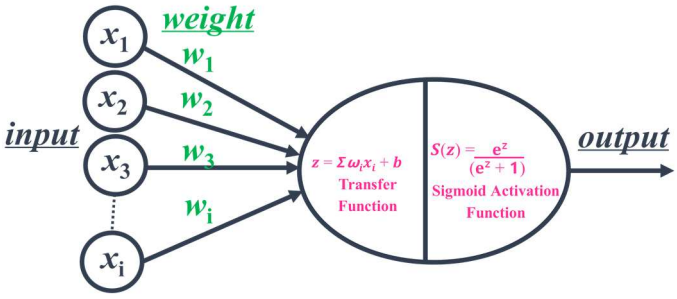


Fig. 2. A neural network consists of inputs $x_i$ is multiplied by its associated weight $w_i$. The results is then sum together ($Z$) and then subjected to Sigmoid function ($S$) to determine output.

In NN, each artificial neuron is connected to other artificial neuron and all artificial neurons are categorized in different layers. In this work, NN consists at least 2 layers, one input layer and one output layer but in some complex neural network there are some hidden layers to improve the accuracy of the results and reduce the errors. As each artificial neuron is connected to other artificial neuron, creates a dense and complex network, thus by adding more layers we increased complexity of the network, thereby the accuracy is increased. To implement the neural network, weight and bias values are needed which are obtained by training the network which is

performed by using TensorFlow and Keras. After training the network, we collected all the necessary parameters. Using those parameters, we tested the network, measure accuracy and error margin. Here, we have used this neural network to classify the motion of the humanoid robot such as forward motion, backward motion, right motion, and left motion. We also included sensor data responses which we collected from sensors, connected to the robot. Figure 3 shows the sensor data are corresponding to 22 different servo motor angles, used as input data for the neural network. The data we have used is 1 dimensional form.

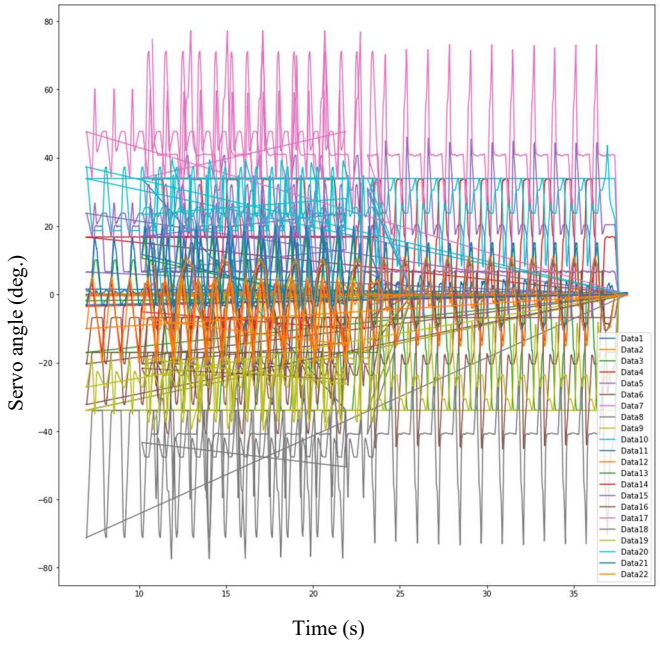


Fig. 3. Servo angles (in deg.) are obtain using Kondo KHR-3HV robot [4] simulation using VREP robot simulator [5]. Here, Data1, Data2, Data3,…… ….., and Data22 are corresponding to servo-motor number 1, 2, 3, ……., and 22, respectively.

Since, we have 22 input data, we considered 22 neurons in input layer and, the output layer consists of 4 neurons. We classified four different motions after classifying the motion, we have four different combinations of data by using that predicts next motion for the robot and, by implementing this neural network on chip, we can reduce delay and can perform prediction in real-time. Here, we avoided the hidden layers because of lesser complexity of our application if the complexity of classification is higher. NN based classified data is shown in Fig. 4. In each layer except the input layer, there are artificial neurons, and each neuron operates on two basic functions i.e., transfer function and sigmoid function. 1) In transfer function, there are two operations, multiplying weight values with input and then adding it with bias value. 2) The sigmoid activation function decides how likely the prediction is true for a given input set. As shown in Fig. 2 there are two operations, which are perform to achieve the predicted output range between 0 to 1. That's why sigmoid function is used as an activation function to get S-shaped curve which has an output range between 0 to 1. So sigmoid function has very important operation which has to be implement very precise.

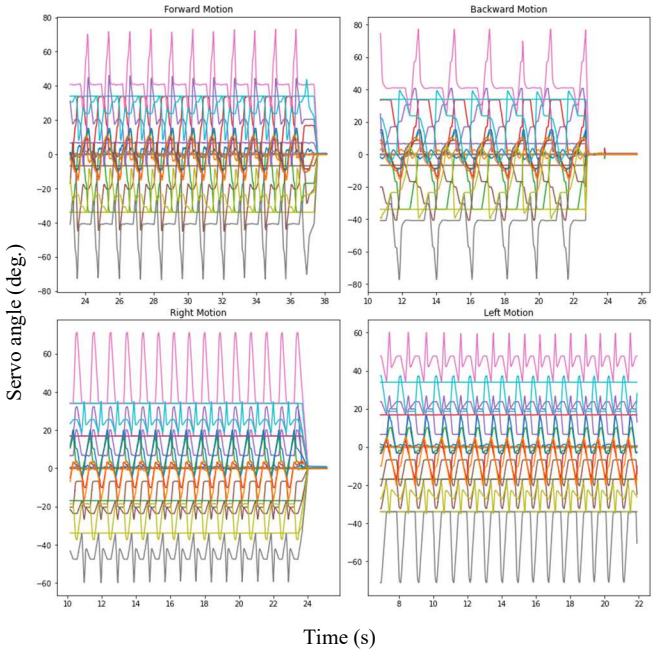


Fig. 4 Classified data corresponding to humanoid robot forward, backward, right, and left movement. The classification is done using developed NN model.

## III. Implementation of Sigmoid Function

Sigmoid function is one type of activation function which is mainly used in ANN [6], [7]. A Sigmoid function is a mathematical function having a characteristic "S" shaped curve or sigmoid curve and normally the output range of this function is from 0 to 1. Because of the shape of the curve and the output range, it is used in NNN. The mathematical equation for the sigmoid function is [6]

$$S(z) = \frac{1}{1+e^{-z}} = \frac{e^{z}}{e^{z}+1} \tag{1}$$

Two basic operations are observed in the equation which are an exponential component and division. Direct implementation of these two operations at hardware level is quite complex. Therefore, instead of direct implementing (1), we can approximate it up to second order which is described below.

### *A. Sigmoid Function Approximation*

Now, as we have seen that sigmoid function contains various complicated operations such as exponential and division operation which makes hardware level implementation complicated. So, to make it easier we have proposed an efficient way of implementing sigmoid function using second order approximation. The proposed function contains operations such as multiplication, addition and subtraction which are not complex as compared to exponential and division operations. The proposed equation of the sigmoid function using second order approximation is mentioned in (2). The result of second-order approximation is almost accurate, also the complexity of the equation is reduced significantly as it requires multiplication, addition, and subtraction operation instead of exponential, so the implementation of this equation is simpler compared to the conventional one. According to our discussion

on the proposed equation, the flow chart for the equation is depicted in Fig.5.

$$S(z)=\begin{cases} 0 & :z\le -4 \\ \frac{1}{2}\left(\frac{z}{2^2}+1\right)^2 & :-4<z<0 \\ 1-\frac{1}{2}\left(\frac{z}{2^2}-1\right)^2 & :0\le z<4 \\ 1 & :z\ge 4 \end{cases} \quad (2)$$

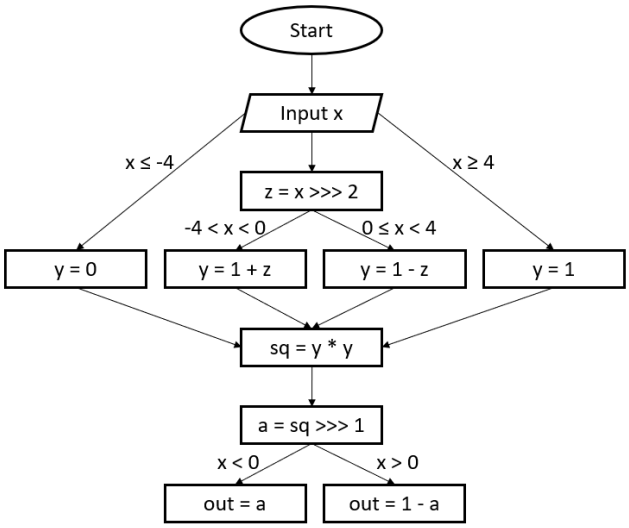


Fig.5 Illustrates the flow chart of Sigmoid approximation. The Sigmoid function is implemented, corresponding to this flow chart.

The sigmoid function is used as an activation function in neural network, as it gives output between the range 0 and 1. Using the proposed sigmoid function, we avoided the exponential operation, also the accuracy is very high. Figure 6 shows the comparison between conventional sigmoid function and proposed sigmoid function. We observed that the curve of the proposed sigmoid function is S-shaped, almost identical to the conventional one.

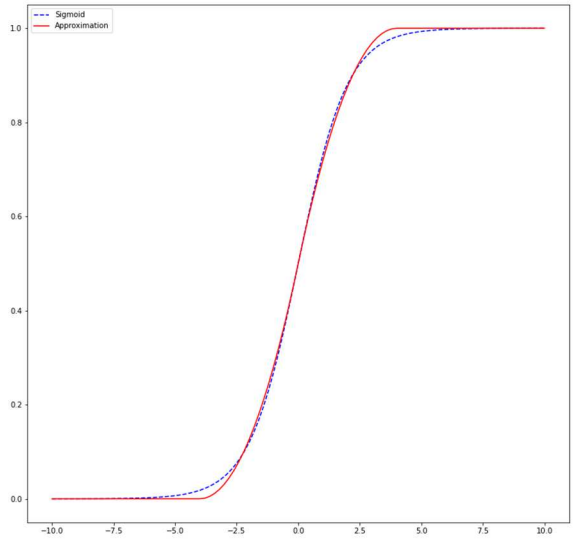


Fig. 6 Comparison between conventional sigmoid function and proposed sigmoid function.

*B. Transfer Function*

Transfer functions are mathematical functions that models the output $y(t)$ (a time varying function) of a system with respect to input $x(t)$. Here, it's important to note that the input has a notation that denotes its pattern, so does the output. A transfer function basically tries to find the mathematical relationship between the input pattern and the desirable pattern (output).

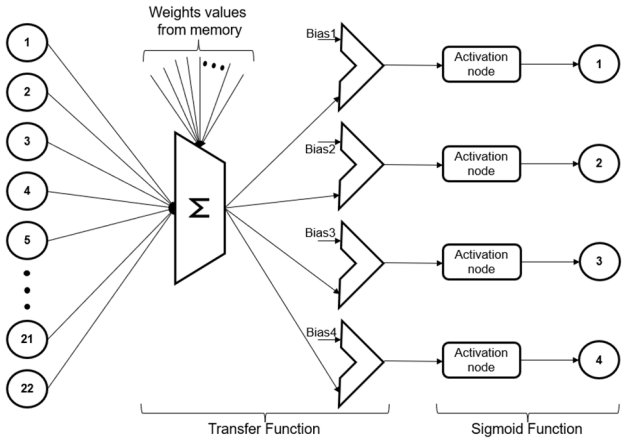


Fig.7 Architectural representation of Artificial Neural Network (ANN).

Figure 7 shows that the transfer function collect multiplication of the input data and weights values and add them and feed to activation function in this case it feeds the output to sigmoid function for activation of the neuron. Equation (3) describes this function,

$$Z = B + \sum_{i=0}^{n} w_i * x_i \quad (3)$$

Here, $Z$ is the final output which feeds to sigmoid function. We considered $n$ (1, 2, 3, ……, $n$) number of inputs, corresponding weight values $w_i$ varies from $w_1$ to $w_n$ same for input data, varies from $x_1$ to $x_n$. Here $B$ is the bias value which is fixed for each neuron. The bias and weight values are obtained after completing the training of the network.

## IV. Hardware Implementation

Hardware implementation of sigmoid function requires adder, comparator, registers, and multiplier [8], [9], [10]. Figure 8 shows the data path for the digital logic design.

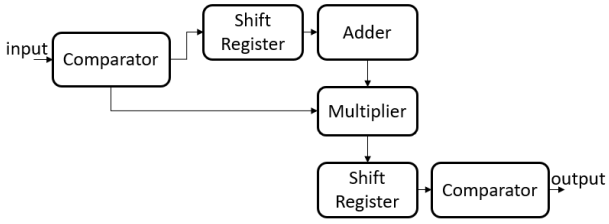


Fig.8 Architecture of proposed sigmoid function for hardware (FPGA) level implementation.

We have discussed earlier in Sec. II is that that an activation function plays a crucial role in NN, which makes it necessary for the hardware implementation to be accurate, with minimal errors. The hardware level implementation is performed for different number of input bits i.e., higher the number of bits,

higher is the accuracy, as well as precision. During testing, we have considered 8-bit, 16-bit, and 32-bit floating numbers as an input to the proposed sigmoid function, one at a time. Of course, 32-bit floating input would have higher precision, also the shape of the curve is sharper compared to the other two inputs i.e., 8-bit and 16-bit, but at the cost of more memory space. In 8-bit floating input, 4 bits are used for representing fraction. In 16-bit, 8 bits are used for representing fraction. In 32-bit, 14 bits are used for the same. Hence, 8-bit floating input is having precision of $6.25*10^{-2}$, for 16-bit it becomes $3.9*10^{-3}$ and for 32-bit it becomes $6.1*10^{-5}$. For output the case is different, 8-bit, 16-bit and 28-bit are used to represent fraction, also the precision comes out to be $3.9 * 10^{-3}$, $1.52 * 10^{-5}$, and $3.72 * 10^{-9}$ respectively for 8-bit, 16-bit, and 32-bit floating inputs. Hence, we have proposed the hardware design of sigmoid function with 32-bit floating input for higher precision. Also, we have analyzed different constraints such as time and delay for our proposed design.

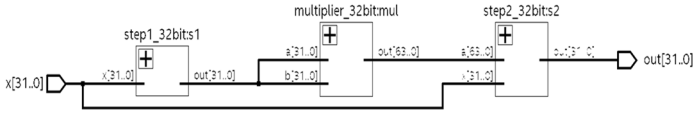


Fig.9 RTL overview of proposed sigmoid function

The RTL shown in the Fig. 9, is an overview of the proposed sigmoid function, there are three parts in the RTL Design, each one having its own functionality. Part-I in the RTL Design performs $\left(\frac{z}{2^2}+1\right)$ or $\left(\frac{z}{2^2}-1\right)$ i.e., either of the two, this requires a comparator, a shift-register and adders, Part-II performs a squaring operation; hence a multiplier is required, and Part-III performs subtraction if the input is greater than 0, otherwise the input is bypassed directly to the output, which are illustrated in Figs.10 (a)-(c), respectively. Simulated output corresponding to the sigmoid function is depicted in Fig. [11].

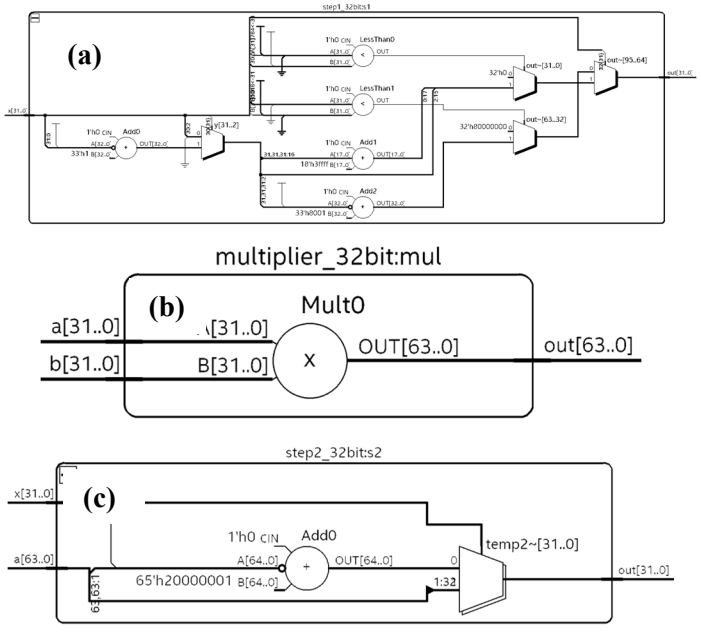


Fig.10 (a) RTL of Part-I, (b) RTL of Part-II, and (c) RTL of Part-III

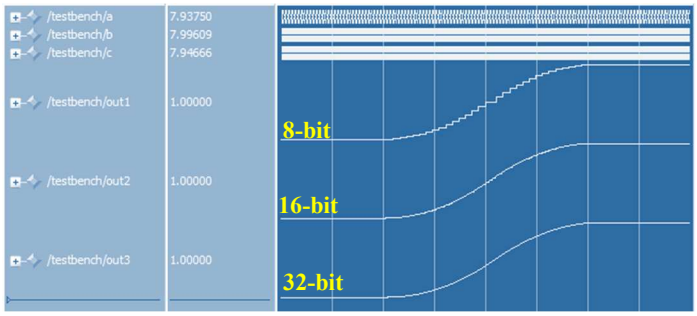


Fig.11 Simulated output of proposed sigmoid function with various precision. Higher precision is obtained for the 32-bit floating input.

## V. Conclusion

For hardware implementation of the proposed model, we combined the NN with AFSM for identifying the next motion off the humanoid robot [11], [12], [13]. Also, we classified robot motion which is implemented in the hidden layer. We optimized several mathematical operations like multiplication, addition, and subtraction to optimize entire neural network.


## Acknowledgement

The work has been supported by The Gujarat Council on Science and Technology (DST-GUJCOST), Govt. of Gujarat, India.


## References


[1] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th US ed., Pearson, Apr. 2020.

[2] R. R. Murphy and R. C. Arkin, Introduction to AI Robotics, 1st ed., MIT Press, Jan. 2001.

[3] R. A. Brooks, Cambrian Intelligence-The Early History of the New AI. MIT Press, Cambridge, MA, 1999.

[4] Specification of KHR-3HV Ver.2 Robot, Dec. 2018, [online] Available: https://kondo-robot.com/product/03178.

[5] M. F. E. Rohmer and S. P. N. Singh, ""V-rep: a versatile and scalable robot simulation framework", *Int. Conf. on Intel. Robo Sys. (IROS)*, pp.1321-1326, Nov. 2013, Tokyo, Japan.

[6] C.-W. Lin and J.-S. Wang, "A Digital Circuit Design of Hyperbolic Tangent Sigmoid Function for Neural Networks", *IEEE Int. Sym. on Cir. and Sys. (ISCAS)*, pp. 1-4, 2008, Seattle, WA, USA.

[7] S. Satyanarayana and H. P. Graf, "A Reconfigurable VLSI Neural Network", *IEEE J. of Solid-State Cir.*, pp. 67-81, vol. 27, Iss. 1, Jan 1992.

[8] A. H. Namin, K. Leboeuf, R. Muscedere, H. Wu, and M. Ahmadi, "Efficient Hardware Implementation of the Hyperbolic Tangent Sigmoid Function" *IEEE Int. Symp. on Circ. and Sys. (ISCAS)*, pp. 1-4, May 2009, Taipei, Taiwan

[9] I. Tsmots, V. Rabyk, and O. Skorokhoda, "Hardware Implementation of Sigmoid Activation Functions using FPGA", *IEEE 15th Int. Conf. on the Exper. of Des. Appl. of CAD Sys. (CADSM)*, pp. 1-4, Mar. 2019, Polyana, Ukraine.

[10] S. Xing and C. Wu, "Implementation of A Neuron Using Sigmoid Activation Function with CMOS", *IEEE Int. Conf. on Integr. Cir. and Microsys. (ICICM)*, pp. 1-4, Oct. 2020, Nanjing, China.

[11] P. Treleaven, M. Pacheco, and M. Vellasco "VLSI Architectures for Neural Network", pp. 1-4, May 1989, Portland, OR, USA

[12] A. Bermak and A. Bouzerdoum, "VLSI Implementation of a Neural Network Classifier Based on the Saturating Linear Activation Function", *Int. Conf. on Neural Info. Process. (ICONIP)*, pp. 1-4, Nov. 2002, Singapore

[13] H. M. El-Bakry and N. Mastorakis "A Simple Design and Implementation of Reconfigurable Neural Networks*", Int. Joint Conf. on Neural Networks (IJCNN)*, Jun 2009, Atlanta, GA, USA.