

A Raspberry Pi-based Hardware Implementation of Various Neuron Models

Vedat Burak Yucedag, Ilker Dalkiran*

Graduate School of Natural and Applied Sciences, Electrical and Electronics Engineering, Erciyes University,
Ahmet El Biruni St., 38030, Melikgazi, Kayseri, Turkiye
vedatburakyucedag@erciyes.edu.tr; *ilkerd@erciyes.edu.tr

Abstract—The implementation of biological neuron models plays an important role in understanding the functionality of the brain. Generally, analog and digital methods are preferred during implementation processes. The Raspberry Pi (RPI) microcontroller has the potential to be a new platform that can easily solve complex mathematical operations and does not have memory limitations, which will take advantage while realizing biological neuron models. In this paper, Hodgkin-Huxley (HH), FitzHugh-Nagumo (FHN), Morris-Lecar (ML), Hindmarsh-Rose (HR), and Izhikevich (IZ) neuron models have been implemented on a standard-equipped RPi. For the numerical solution of each neuron model, the one-step method (4th order Runge-Kutta (RK4), the new version of Runge-Kutta (RKN)), the multi-step method (Adams-Bashforth (AB), Adams-Moulton (AM)), and predictor-corrector method (Adams-Bashforth-Moulton (ABM)) are preferred to compare results. The implementation of HH, ML, FHN, HR, and IZ neuron models on RPi and the comparison of numerical models RK4, RKN, AB, AM, and ABM in the implementation of neuron models were made for the first time in this study. Firstly, MATLAB simulations of the various behaviors belonging to the HH, ML, FHN, HR, and IZ neuron models were completed. Then those models were realized on RPi and the outputs of the models are experimentally produced. The errors are also presented in the tables. The results show that RPi can be considered as a new alternative tool for making complex neuron models.

Index Terms—Raspberry Pi; Hodgkin-Huxley; Hindmarsh-Rose; Izhikevich; Runge-Kutta; Adams-Bashforth-Moulton.

I. INTRODUCTION

The question of how the brain processes information has led scientists to investigate the nervous system. The nervous system is the special network structure in which neurons are connected and establish chemical and electrical bonds with each other. Neurons communicate with each other using an action potential (spike) or an explosive action potential (burst). Spike or burst behaviors can have different amplitudes, frequencies, and pattern shapes [1].

Biologically plausible neuron models such as Hodgkin-Huxley (HH) directly define the behavior of a neuron [2]. This four-dimensional model describes the ionic mechanism

and electrical current on the membrane surface of the neuron most biologically and accurately. In contrast, the mathematical complexity and the cost of implementation of the HH model are quite high. Morris-Lecar (ML) model describes oscillation in cuttlefish giant axons based on conductivity [3]. Biologically inspired neuron models such as FitzHugh-Nagumo (FHN), Hindmarsh-Rose (HR), and Izhikevich (IZ) replicate biological neuron behaviors. The FitzHugh-Nagumo (FHN) model was presented as a new model in the literature by simplifying the complexity of the HH model [4], [5]. Implementing FHN is easy since, as it has two variables. But it cannot produce sufficient action potential patterns such as bursting. The HR model obtained by the development of the FHN model can exhibit many dynamic behaviors of a biological neuron [6]. On the other hand, the IZ neuron model was defined by two simple differential equations that can mimic the biological neuron at low cost and produce a very rich membrane dynamics [7].

There are two basic techniques to implement the mathematical models of neurons, analog and digital systems. Typically, hardware implementations of neuron models are realized by using discrete elements, very large-scale integrated (VLSI) systems, field-programmable analog arrays (FPAAs), and field-programmable gate arrays (FPGAs). Lazaridis, Drakakis, and Barahona [8] realized an analog implementation of the HH neuron model using weak-inversion CMOS technology. However, the implementation did not have readjustability. Hu, Liu, Liu, Ni, and Li [9] realized the ML neuron model using basic analog elements such as a capacitor, resistance, NPN transistor, and OP-AMP. The hyperbolic cosine function was defined as half of the sum of two exponential functions with the same input value but opposite sign. Realizing the hyperbolic function is difficult and complex [9]. Khanday Kant, Dar, Zulkifli, and Psychalinos [10] presented both integer and fractional order FHN neuron models to perform CMOS circuit in the hyperbolic sine-domain (SD). Heidarapur, Ahmadi, Ahmadi, and Rahimi Azghadi [11] modified the IZ neuron model using the coordinate rotation digital computer (CORDIC) algorithm in their studies. Later, they adapted the spike timing-dependent plasticity learning algorithm (STDP) to the modified model and performed it on an FPGA. Heidarapur, Ahmadi, and Kandalaf [12] applied a set of piecewise linear approaches to the 2D HR neuron model. The neural network structure was able to display tonic spiking, tonic bursting,

Manuscript received 12 May, 2024; accepted 21 October, 2024.

This research is supported by the Erciyes University Scientific Research Projects Coordination under Grant No. FDK2022-11506; TUBITAK, the Scientific and Technological Research Council of Turkey, under Grant No. TBTK-0039-0783; 2211-A Domestic General Doctorate Scholarship Program under Grant No. 1649B032201035; 2214-A International Research Scholarship Program (For PhD students) under Grant No. 1059B142201074-2022/2.

spike latency, and class 1 and class 2 excitability behaviors on an FPGA. Even if the extra approximations are useful to reduce implementation costs, they produce inaccurate results compared to real neuron dynamics. Korkmaz, Öztürk, and Kılıç [13] realized a model of a chemical neuron, which has chemical coupling, on an FPAA. They modified the exponential function of the chemical coupling using a new approach. In this way, they reduced the coupling complexity. However, the use of four FPAA in the design makes the system more expensive [13], [14].

Although VLSI systems are fast and efficient, they do not have re-adjustable flexibility, and the development time is so long. With the development of technology, VLSI systems, which contain more transistors in the same area, still have a problem modelling large-value capacities in integrated circuits [13]. Reconfigurable platforms have more flexible structures compared to VLSI. FPAA are a preferred tool with high stability, accuracy, and rapid prototyping features. FPAA have negative features, such as limited capacity and low saturation level (+2 V). To overcome the limited capacity problem, FPAA can be linked with each other to operate in parallel. However, this significantly increases noise sensitivity and cost. In addition, since FPAA are continuous time-based systems, it is so difficult to realize discrete-time defined models on an FPAA [14]. FPGAs are digital platforms. Multiplier blocks, calculations of hyperbolic functions, footprint, and propagation delay in an FPGA cause higher costs [15], [16]. If look-up table (LUTs) structures in an FPGA are used during the calculations of hyperbolic functions, they reduce the accuracy of the output due to their limitations on memory size [15], [16]. In contrast, floating-point operations increase the accuracy of the output, but cause a considerable use of FPGA resources [14].

The motivation of this work is the presentation of a Raspberry Pi (RPI) microcontroller/microprocessor as an alternative tool to realize neuron models. RPI can be described as a standard hardware with high adjustable ability, with which to solve complex mathematical expressions easily and quickly, without memory limitations, low thermal noise sensitivity, and low-cost production process. In addition, Python-based open source code applications can be easily implemented on RPI. In the literature, although there are several implementations of neuron models on various platforms such as discrete elements, VLSI, FPAA, and FPGA as mentioned above, no studies have been encountered using RPI.

For numerical solutions of neuron models, the fourth order Runge-Kutta (RK4) method has been generally preferred, because it is easier to simulate than the other methods [17], [18]. A few studies about Runge-Kutta New Version (RKN), Adams-Bashforth (AB), Adams-Moulton (AM), and Adams-Bashforth-Moulton (ABM) methods have been presented in [19]–[22]. It is not known which of these solution models will solve the neuron mathematical models more accurately and faster. It is also seen that more than one numerical method has never been tested for a neuron model on the same platform, such as FPGA, FPAA, and VLSI systems. In addition to RPI implementation, this study investigates the suitability of RK4, RKN, AB, AM, and ABM numerical methods for solving neuron models.

In all these contexts, various action potential behaviors of

HH, ML, FHN, HR, and IZ neuron models, which have a wide area in the literature, were obtained by numerical analysis methods in MATLAB. After discretizing each of neuron models with the RK4, RKN, AB, AM, and ABM methods, the differential equations obtained were performed on RPI by running software written in Python. The RPI has 40 general-purpose input and output (GPIO) pins and none of them has capable of analog output. Therefore, the signal received from the GPIO pin of the RPI is applied to the 12-bit digital/analog converter.

The paper is organized as follows. Section II presents HH, ML, FHN, HR, and IZ neuron models. The numerical methods and error analysis are presented in Section III. The implementation of neuron models on RPI is explained in Section IV. The discussion and conclusions are given in Sections V and VI, respectively.

II. NEURON MODELS

A. HH Neuron Model

Since the HH neuron model exhibits the properties of a biological neuron in detail, the number of parameters is numerous. Accordingly, the implementation cost is high:

$$C_M \frac{dV}{dt} = I_{input} - \bar{g}_k n^4 (V - E_K) - \bar{g}_{Na} m^3 h (V - E_{Na}) - \bar{g}_L (V - E_L), \quad (1)$$

$$\frac{dn}{dt} = a_n(V)(1-n) - \beta_n(V)n, \quad (2)$$

$$\frac{dm}{dt} = a_m(V)(1-m) - \beta_m(V)m, \quad (3)$$

$$\frac{dh}{dt} = a_h(V)(1-h) - \beta_h(V)h. \quad (4)$$

Here, C_M , I_{input} , and V represent membrane capacitance, applied external current, and the membrane potential, respectively. \bar{g}_k and \bar{g}_{Na} are the greatest conductivity of potassium and sodium ion channels, respectively, and \bar{g}_L is the largest conductivity value of leakage ions. n , m , and h represent the probabilities that the activation or inactivation gate is open. The expressions $a_n(V)$, $a_m(V)$, and $a_h(V)$ are the speed functions of the ion channel gates which change from closed to open. The expressions $\beta_n(V)$, $\beta_m(V)$, and $\beta_h(V)$ are also speed functions that determine the transition of ion channel gates which change from open to closed [2].

B. ML Neuron Model

The conductivity-based neuron model has high biological accuracy and is easy to implement. The ML neuron model consists of two differential equations [3]:

$$C \frac{dV}{dt} = I - g_{Ca} M_\infty(V)(V - E_{Ca}) - g_K N(V - E_K) - g_L (V - E_{Leak}), \quad (5)$$

$$\frac{dN}{dt} = \lambda_N(V)[N_\infty(V) - N]. \quad (6)$$

Here, V represents the membrane potential of the neuron. The parameter N represents the slow activation of K^+

channels. C is the membrane capacitance and I is the stimulus current. g_K , g_{Ca} , and g_L are the maximal conductivity of the potassium, the calcium, and the leakage current of ion channels, respectively.

C. FHN Neuron Model

The FHN neuron model is obtained by simplifying the HH neuron model. Since it has a structure that mimics the behavior of a real neuron, its biological accuracy is low. There are studies examining the effects of external forced current [23]:

$$\frac{dv}{dt} = c \left(v - u + I - \frac{v^3}{3} \right) + I_{ext}, \quad (7)$$

$$\frac{du}{dt} = \frac{v - bu + a}{c}, \quad (8)$$

$$I_{ext} = \frac{A}{2\pi f} \cos(2\pi ft), \quad (9)$$

where v , u , and I represent the membrane potential of the nerve cell, the recovery parameter, and the external current applied to the cell membrane, respectively. a and b are scaling parameters, c is a constant value. I_{ext} represents external forcing current.

D. HR Neuron Model

The HR neuron model can generate burst waveforms because it has a variable that defines the adaptation current. The model is defined by three differential equations [6]:

$$\frac{dx}{dt} = y - ax^3 - bx - z + I, \quad (10)$$

$$\frac{dy}{dt} = c - dx^2 - y, \quad (11)$$

$$\frac{dz}{dt} = \varepsilon (s(x - x_{rest}) - z). \quad (12)$$

Here, x , y , and z represent the membrane potential, the recovery parameter, and the adaptation current, respectively. b regulates the transition control between bursting and spiking and action potential frequency. ε is the control parameter of the frequency of the action potential and the number of each action potential in the behavior of the burst. s is the parameter that provides adaptation. x_{rest} represents the resting potential of the system and I represents the stimulating.

E. IZ Neuron Model

The IZ neuron model has the rich dynamics of real neurons and has a low implementation cost. The model consists of two differential equations [7]:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I, \quad (13)$$

$$\frac{du}{dt} = a(bv - u), \quad (14)$$

$$v \geq 30 \text{ mV} \Rightarrow v \leftarrow c \quad \text{and} \quad u \leftarrow u + d. \quad (15)$$

Here, v is the membrane potential of the neuron and u is

the recovery variable. a defines the time scaling of the recovery variable u at low values where the recovery is slow. b represents the sensitivity of the recovery variable u and the threshold level fluctuations of the membrane potential v . c is described as the reset value of v after an action potential. d is defined as the reset parameter of u after the action potential. The parameter I indicates the external stimulus currents.

III. RASPBERRY PI AND NUMERICAL METHODS

RPi is a digital platform that can quickly run complex mathematical expressions, unlike the FPGA and FPAA systems, without any memory limitations [24].

RPi 4 has a 1.5 GHz quad-core ARM Cortex-A72 processor architecture, which is very fast and highly energy efficient. It has a wide RAM bandwidth (2983 MBW) with 2 GB, 4 GB, and 8 GB LPDDR4 SDRAM options, but the 4 GB version was used in the study. The RPi, which supports OpenGL ES 3.x with its internal image processing unit, provides two high-resolution image outputs, 4k and 1080p. It also has external data communication at high write and read speeds with USB 3.0 support. The Python IDE interface is included as standard with the Raspberry Pi OS operating system. In this study, the solution of neuron models with numerical methods was realized with this interface [25].

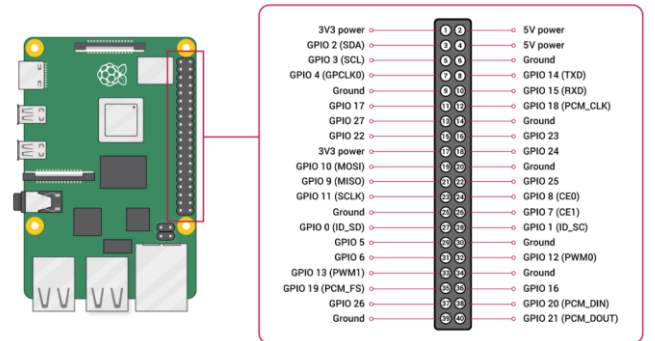


Fig. 1. Raspberry Pi 4 design and GPIO pin diagram.

The RPi 4 has 40 GPIO pins with input/output capabilities for a wide range of purposes such as serial communication pins UART TX-RX, I2C communication protocols, EEPROM SDA-SCL, and PWM. These outputs are all digital, and the RPi does not have analog output pins [25]. For analog output, the MCP4725 12-bit DAC with 6-pin SOT-23 package is used, which can be easily controlled via I2C. With a single supply voltage in the range of 2.7 V–5.5 V, the output pitch is 0 V–3.3 V and 0 V–5 V. For the negative values of the neuron, a full subtractor circuit was created with the LM741 OP-AMP. The MCP4725 has a two-wire I2C™ compatible serial interface for standard (100 kHz), fast (400 kHz), or high speed (3.4 MHz) mode. It also has a DAC that can operate between -40 °C and +125 °C ambient temperature range, ± 0.2 LSB DNL and 6 μ s fast settling time [26].

Since neuron models are defined in continuous time, their digital structure cannot be realized directly on both RPi microprocessors. These neuron models must be transformed into discrete time with various discretization methods. One-step and multi-step methods are generally used to discretize ordinary differential equations. These numerical solution methods start from an initial point and then take a short step

in time to find the next point. This process continues with the next steps to construct the solution space. One-step methods (RK4 and RKN) refer only to the previous point and its derivative to determine the current value. They also take some intermediate steps to get a higher order, but then discard all previous information before taking a second step [27]. Multi-step methods (Adams-Bashforth (AB) and Adams-Moulton (AM)) try to gain efficiency by storing and using information from previous steps rather than discarding it. As a result, multi-step methods refer to the previous points and their derivative values [28]. The predictor-corrector methods are based on estimating the solution of differential equations using the explicit formula and then correcting the estimated value using another implicit formula. Generally, AB is chosen as a predictor, and AM is selected as a corrector [28]. The expressions of numerical solution methods are given in the Appendix A.

In light of this information, RK4, RKN, explicit four-step AB, implicit three-step AM, and predictor-corrector ABM numerical methods are preferred to show that various discretization methods can be performed on RPi in this study. The RPi microprocessor can easily perform numerical solution methods which have various calculation difficulties. In addition, the numerical methods are compared in terms of design flexibility, simulation speed, simulation accuracy (with both spike and burst), and achieving dynamical behaviors of the neuron models. In this way, it has been determined which methods are more suitable for the numerical solution of neuron models.

Mean absolute error (MAE) and normalized root mean squared error (NRMSE) were calculated using implementation results. MAE is the average of the absolute differences, whereas MSE is the average of the squared differences between simulation and implementation results. NRMSE facilitates comparison between simulation and implementation of neuron models with different scales. MAE and NRMSE are formulated as follows:

$$MAE = \frac{1}{N} \sum_{i=1}^N |v_i^{simulation} - v_i^{implementation}|, \quad (16)$$

$$NRMSE = \frac{\sqrt{\frac{1}{N} \sum_{i=1}^N (v_i^{simulation} - v_i^{implementation})^2}}{(v_{i,max}^{implementation} - v_{i,min}^{implementation})}. \quad (17)$$

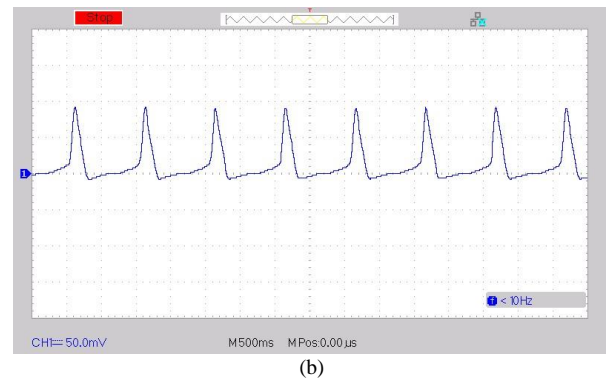
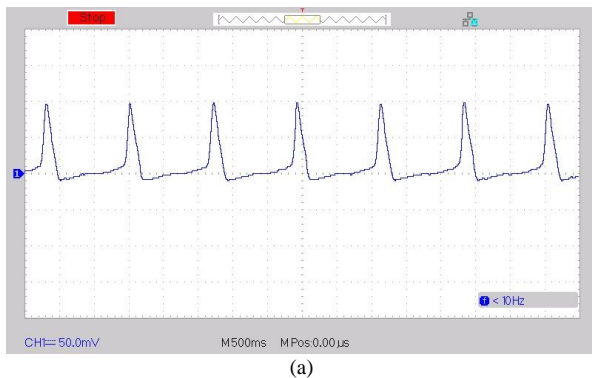


Fig. 2. Waveforms resulting from the implementation of the HH neuron model on RPi: (a) HH-Sp1 behavior with RK4 method; (b) HH-Sp2 behavior with RKN method.

The lowest error on the RPi was obtained by the AM method. However, it is the slowest among the five methods

IV. RESULTS

In this section, the implementation of HH, ML, FHN, HR, and IZ neuron models on RPi is explained using numerical methods RK4, RKN, AB, AM, and ABM. Various neuron dynamic behavior examples were obtained according to parameters determined. Each instance of each neuron model was performed with five different numerical methods. In addition, waveforms were obtained with both MATLAB simulation and the oscilloscope. For simplicity and intelligibility of the paper, not all waveforms are presented, but four states are specified for each neuron model.

A. Implementation of HH Neuron Model

The HH neuron model is implemented on the RPi using (1)–(4). Two samples' dynamics was formed with the parameters selected from the HH neuron model and named as HH-Sp. Each sample was carried out with the numerical methods RK4, RKN, AB, AM, and ABM. The parameters are determined from previous studies [2]. The error between the MATLAB simulation and RPi implementation after synthesis are presented in Table I. The membrane potential waveform is also presented in Fig. 2.

TABLE I. THE ERROR OF THE HH NEURON MODEL BETWEEN MATLAB SIMULATION AND RPI IMPLEMENTATION.

HH Model		HH-Sp1	HH-Sp2	Average
RK4	MAE	2,1506	2,3079	2,2293
	NRMSE	2,8149	3,3094	3,0622
	Period [s]	1,197	0,954	1,0755
RKN	MAE	2,2124	2,2549	2,2337
	NRMSE	3,0609	3,5117	3,2863
	Period [s]	1,261	1,002	1,1315
AB	MAE	2,6957	3,9776	3,3367
	NRMSE	3,9879	6,3379	5,1629
	Period [s]	1,198	0,952	1,0750
AM	MAE	2,4275	2,2445	2,3360
	NRMSE	3,1864	3,0136	3,1000
	Period [s]	1,319	1,056	1,1875
ABM	MAE	3,3559	4,3618	3,8589
	NRMSE	5,1703	6,5923	5,8813
	Period [s]	1,307	1,050	1,1785

Considering Table I, the largest error value was obtained when using the AB numerical method. The periods of the membrane potential waveforms of the HH neuron model based on the AB numerical method are shorter than those of the other numerical methods. It is seen that although AB is a fast multi-step method, it has large error values while implementing the HH neuron.

while solving the HH neuron. The error of the numerical methods are ordered as $RK4_{error}^{HH} < AM_{error}^{HH} < RKN_{error}^{HH} <$

$AB_{error}^{HH} < ABM_{error}^{HH}$. The order of the numerical methods according to the speed of solving HH neuron is $AM_{speed}^{HH} < ABM_{speed}^{HH} < RKN_{speed}^{HH} < RK4_{speed}^{HH} < AB_{speed}^{HH}$. According to this comparison, RK4 and RKN methods, which are relatively fast and relatively have low error, can be preferred to the numerical solution method of the HH neuron. However, during RPi implementation, low-value distortions occur in the waveforms as the membrane potential increases from its lowest value to 20 mV. This situation increases, MAE and NRMSE. The reason for distortion is that environmental

parasitic effects cannot be completely prevented during the measurement process of the waveform with mV amplitude.

B. Implementation of ML Neuron Model

The ML neuron is also implemented on the RPi. [The parameters are determined from previous studies [9]. Two samples were built and named ML-Sp. Each sample was carried out with the numerical methods RK4, RKN, AB, AM, and ABM. The regular spike condition is presented in Fig. 3. The error between the MATLAB simulation and the RPi implementation after synthesis are presented in Table II.

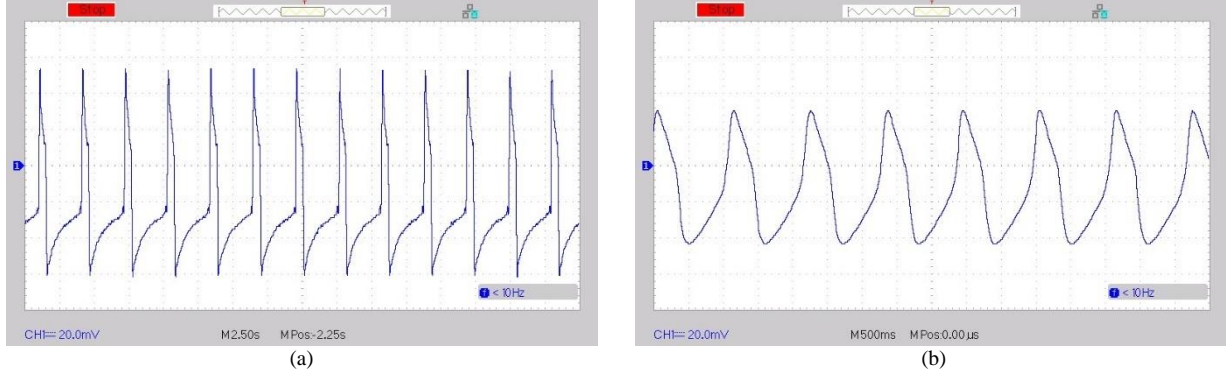


Fig. 3. Waveforms resulting from the implementation of the ML neuron model on RPi: (a) ML-Sp1 behavior with RK4 method; (b) ML-Sp2 behavior with RK4 method.

TABLE II. THE ERROR OF THE ML NEURON MODEL BETWEEN MATLAB SIMULATION AND RPi IMPLEMENTATION.

ML Model		ML-Sp1	ML-Sp2	Average
RK4	MAE	1,5676	1,2207	1,3942
	NRMSE	2,5809	1,8516	2,2163
	Period [s]	1,651	1,110	1,3805
RKN	MAE	3,9243	2,7229	3,3236
	NRMSE	5,9390	3,9767	4,9579
	Period [s]	1,706	1,155	1,4305
AB	MAE	1,9192	1,4184	1,6688
	NRMSE	3,2773	2,3417	2,8095
	Period [s]	1,654	1,112	1,3830
AM	MAE	1,9458	1,4647	1,7053
	NRMSE	3,3486	2,4703	2,9095
	Period [s]	1,875	1,265	1,5700
ABM	MAE	1,7312	1,4101	1,5707
	NRMSE	2,4287	2,2995	2,3641
	Period [s]	1,873	1,269	1,5710

Considering Table II, the largest error was obtained when the RKN method was used. The periods of the membrane potential waveforms of the ML neuron based on the RKN numerical method are larger than those of the other methods. ABM is a relatively slow one-step method with large error

values for the implementation of the neuron. The ML neuron implementation with the lowest error on the RPi was obtained using the RK4 method. The error values of the numerical methods are ordered as $RK4_{error}^{ML} < ABM_{error}^{ML} < AB_{error}^{ML} < AM_{error}^{ML} < RKN_{error}^{ML}$. The order of the numerical methods according to the speed of solving ML neuron is $ABM_{speed}^{ML} < AM_{speed}^{ML} < RKN_{speed}^{ML} < AB_{speed}^{ML} < RK4_{speed}^{ML}$. According to this comparison, the RK4 numerical method, which is fast and has lowest error, can be preferred in the numerical solution of the ML neuron.

C. Implementation of FHN Neuron Model

The differential equations given in (7)–(9) are used to implement the FHN neuron on RPi and the parameter values determined according to other studies [29]. Two samples were built and named FHN-Sp. Each sample was carried out with the numerical methods RK4, RKN, AB, AM, and ABM. The regular spike condition obtained at various values is presented in Fig. 4 and the error RPi implementation is presented in Table III.

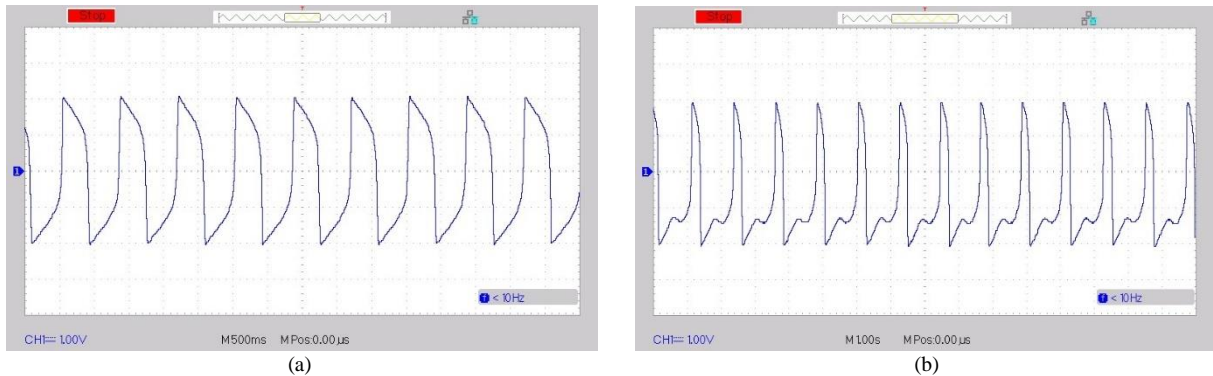


Fig. 4. Waveforms resulting from the implementation of the FHN neuron model on RPi: (a) FHN-Sp1 behavior with AB method; (b) FHN-Sp2 behavior with AB method.

Considering Table III, the largest error was obtained when the ABM method was used. The periods of the membrane

potential waveforms of the FHN neuron based on the ABM method are greater than those of the other numerical methods. ABM is a relatively slow one-step method with large error values for model implementation. The FHN neuron implementation with the lowest error on the RPi was obtained by the AM method. However, it is slow in solving the FHN neuron. The error values of the numerical methods are ordered as $AM_{error}^{ML} < AB_{error}^{ML} < RK4_{error}^{ML} < RKN_{error}^{ML} < ABM_{error}^{ML}$. The order of the numerical methods according to the speed of solving differential equations is $ABM_{speed}^{ML} < AM_{speed}^{ML} < RKN_{speed}^{ML} < AB_{speed}^{ML} < RK4_{speed}^{ML}$. According to this comparison, the AB method, which is fast and relatively has low error

values, can be preferred in the numerical solution of the FHN neuron.

D. Implementation of HR Neuron Model

The next neuron implemented is HR. Two samples, one of which was chaotic, were constructed with selected parameters and named HR-Sp. Each sample was carried out with the numerical methods RK4, RKN, AB, AM, and ABM. The burst and chaotic conditions obtained at the various values of the parameters are presented in Fig. 5. The error values of RPi implementation after synthesis are presented in Table IV.

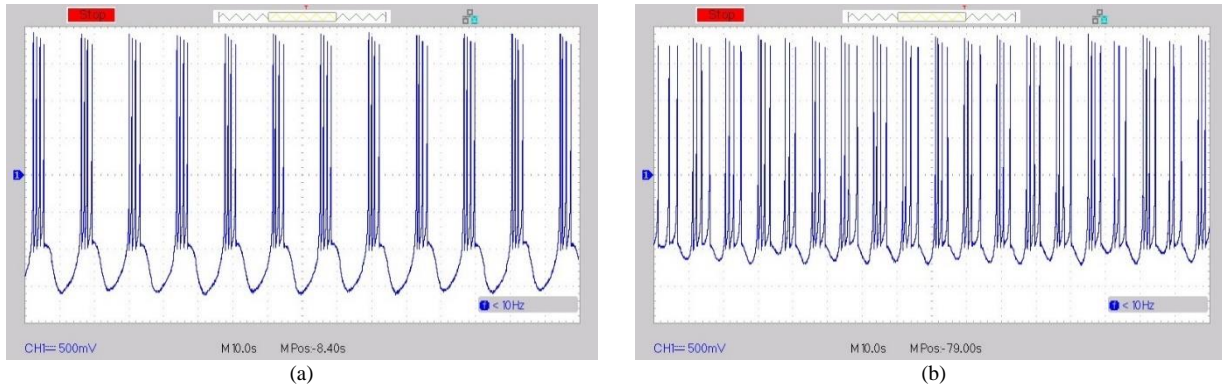


Fig. 5. Waveforms resulting from the implementation of the HR neuron model on RPi: (a) HR-Sp1 (Burst of 4 Spike) behavior with RK4 method; (b) HR-Sp2 (Chaotic) behavior with AB method.

TABLE IV. THE ERROR OF THE HR NEURON MODEL BETWEEN MATLAB SIMULATION AND RPI IMPLEMENTATION.

HR Model		HR-Sp1 (Burst of 4 Spike)	HR-Sp2 (Chaotic)	Average
RK4	MAE	0,0347	0,0785	0,05
	NRMSE	2,9527	7,8779	4,5971
	Period [s]	13,910	--	16,27
RKN	MAE	0,0787	0,0499	0,0692
	NRMSE	5,4939	4,6018	5,0564
	Period [s]	14,350	--	16,785
AB	MAE	0,0815	0,0430	0,0556
	NRMSE	7,9339	3,3252	4,9339
	Period [s]	13,690	--	16
AM	MAE	0,0583	0,0635	0,0588
	NRMSE	5,9089	6,4861	5,8892
	Period [s]	15,590	--	18,345
ABM	MAE	0,0889	0,0422	0,0583
	NRMSE	8,9312	3,1453	5,4639
	Period [s]	15,360	--	17,9325

Considering Table IV, the largest error was obtained when using the AM method. The periods of the membrane potential waveforms of the HR neuron based on AM method are larger than those of the other methods. AM is a slow multi-step method with large error values for the implementation of the

HR neuron. The HR neuron implementation with the lowest error value on the RPi was obtained with the RK4 method. It is also fast in solving the HR neuron. The error values of the numerical methods are ordered as $RK4_{error}^{HR} < AB_{error}^{HR} < RKN_{error}^{HR} < ABM_{error}^{HR} < AM_{error}^{HR}$. The order of the numerical methods according to the speed of solving differential equations is $AM_{speed}^{HR} < ABM_{speed}^{HR} < RKN_{speed}^{HR} < RK4_{speed}^{HR} < AB_{speed}^{HR}$. According to this comparison, RK4 and AB numerical methods, which are relatively fast and relatively have low error values, can be preferred in the numerical solution of the HR neuron model.

E. Implementation of IZ Neuron Model

The last model which is implemented on the RPi is the IZ neuron. The parameter values of the differential equations that are given in (13)–(15) were obtained. Two samples were formed and defined by the expression IZ-Sp. Each sample was carried out with the numerical methods RK4, RKN, AB, AM, and ABM. The error between RPi implementation after synthesis is presented in Table V. The different membrane behaviors are presented in Fig. 6.

TABLE V. THE ERROR OF THE IZ NEURON MODEL BETWEEN MATLAB SIMULATION AND RPI IMPLEMENTATION.

IZ Model		IZ-Sp1 (Fast Spike)	IZ-Sp2 (Burst of 3 Spike)	Average
RK4	MAE	1,6548	2,1416	1,8982
	NRMSE	6,8243	5,4824	6,15335
	Period [s]	5,792	34,550	20,171
RKN	MAE	1,7588	2,5794	2,1691
	NRMSE	6,9163	6,8508	6,88355
	Period [s]	5,894	34,860	20,377
AB	MAE	1,5471	3,4937	2,5204
	NRMSE	6,3682	8,1993	7,28375
	Period [s]	5,804	34,890	20,347
AM	MAE	1,5293	3,3799	2,4546
	NRMSE	5,1136	8,6370	6,8753
	Period [s]	6,020	35,430	20,725

IZ Model		IZ-Sp1 (Fast Spike)	IZ-Sp2 (Burst of 3 Spike)	Average
ABM	MAE	1,3827	3,9585	2,6706
	NRMSE	5,6117	9,1965	7,4041
	Period [s]	5,982	35,300	20,641

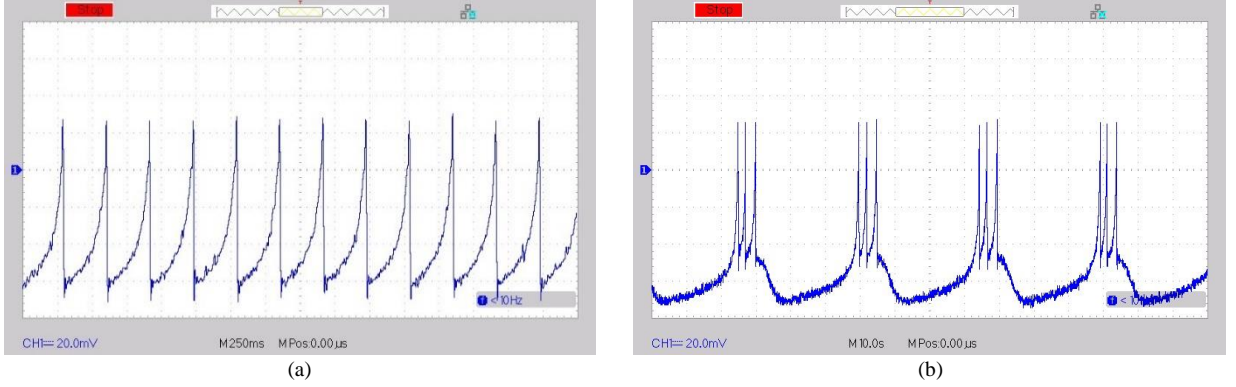


Fig. 6. Waveforms resulting from the implementation of the IZ neuron model on RPi: (a) IZ-Sp1 (Fast Spike) behavior with RK4 method; (b) IZ-Sp2 (Burst of 4 Spike) behavior with RKN method.

Considering Table V, the largest error value was obtained when using the ABM method. The periods of the membrane potential waveforms of the IZ neuron obtained by the AM method are of great value compared to those obtained by other methods. AM is a slow one-step method with large error values for the implementation of the IZ neuron. The IZ neuron implementation with the lowest error on the RPi was obtained with the ABM method. However, it is slow in solving the IZ neuron. The error values of the numerical methods are ordered as $RK4_{error}^{IZ} < AM_{error}^{IZ} < RKN_{error}^{IZ} < AB_{error}^{IZ} < ABM_{error}^{IZ}$. The order of the numerical methods according to the speed of solving differential equations is $AM_{speed}^{IZ} < ABM_{speed}^{IZ} < RKN_{speed}^{IZ} < AB_{speed}^{IZ} < RK4_{speed}^{IZ}$.

According to this comparison, RK4 and RKN method, which is relatively fast and has relatively low error values, can be preferred in the numerical solution of the IZ neuron. However, on RPi implementation, low-value distortions occur on the waveform as the membrane potential increases from its lowest value to -50 mV.

V. DISCUSSION

In this study, both the implementation of HH, ML, FHN, HR, and IZ neuron models and the comparison of the obtained results of the RK4, RKN, AB, AM, and ABM numerical methods on RPi was made for the first time. According to the results presented, the action potential patterns of both simulations and implementations were matching. It is divided into some categories for comparison of these numerical methods, such as achieving the dynamic behaviors of the neuron with these numerical methods, design flexibility, simulation speed, and simulation accuracy. Thus, in Table VI, the performance comparison of the numerical methods is summarized instead of the exact results. While (+++++) shows high performance, (+++) represents good performance, and (++) means average acceptability.

To determine the performance of RPi in neuron implementation, various neuron dynamics behaviors such as regular spike, fast spike, burst, and chaotic were formed with different parameters. RPi was able to perform all samples

using numerical methods RK4, RKN, AB, AM, and ABM.

TABLE VI. A QUALITATIVE COMPARISON FOR NUMERICAL METHODS USED FOR IMPLEMENTATION OF NEURON ON RPI.

	RK4	RKN	AB	AM	ABM
To achieve the dynamic behaviors of the neuron	+++++	+++++	+++++	+++++	+++++
Design flexibility	+++++	++++	++++	+++	++
Simulation speed	+++++	++++	+++++	++	++
Simulation accuracy (with spike)	+++++	++++	+++	+++++	++
Simulation accuracy (with burst)	+++++	+++++	++++	++	+++

However, the design flexibility of numerical methods is different. RK4 and RKN are easy to design because they do not keep the previous information in memory and use the current information. For the AB numerical method to solve the differential equation, the first four terms must be known in advance. It gradually predicts other terms based on these terms, which can be determined using methods such as RK4 or RKN. The design of the AB is relatively more difficult, as additional methods are used. The AM numerical method, on the other hand, has an algorithm that corrects previously known terms and creates new terms. That is, unlike the AB, all terms must be known beforehand. Therefore, the AM numerical method consists of two stages: determining the terms using RK4 or RKN and correcting the determined terms. This complicates the design of the AM method. In the ABM numerical method, the predictive method AB is used in determining the terms, and the AM method is used in the corrective stage. The ABM is the most difficult method to design, as the AB and the AM are used together. The difficulty of designing numerical methods directly affects the simulation speed. The RK4 method is the fastest as it finds the new term in four steps with random starting points. Another method that finds the new term in four steps is AB. Although the first four terms must be known beforehand, it is faster than the five-step RKN method. Step 5 had a negative effect on the simulation speed. AM and ABM are the slowest methods, where all terms need to be known beforehand.

TABLE VII. SCORING TABLE FOR THE PERFORMANCE OF NUMERICAL METHODS IN NEURON MODEL REALIZATION ON RPI.

Speed (Scoring from 1 to 5)						
	HH	ML	FHN	HR	IZ	Average
RK4	4	5	4	4	5	4,4
RKN	3	3	3	3	3	3
AB	5	4	5	5	4	4,6
AM	1	1	1	1	1	1
ABM	2	2	2	2	2	2
Error (Scoring from 1 to 5)						
	HH	ML	FHN	HR	IZ	Average
RK4	3	4	2	5	1	3
RKN	4	1	3	3	2	2,6
AB	1	2	4	4	3	2,8
AM	5	5	5	1	4	4
ABM	2	3	1	2	5	2,6
Overall (Speed + Error (Scoring from 2 to 10))						
	HH	ML	FHN	HR	IZ	Average
RK4	7	9	6	9	6	7,4
RKN	7	4	6	6	5	5,6
AB	6	6	9	9	7	7,4
AM	6	6	6	2	5	5
ABM	4	5	3	4	7	4,6

The speed and error values of the numerical methods are compared. Table VII has been prepared to understand which numerical method gives better results in the realization of the neuron model on RPi. Here, scores were made between 1 and 5, with the best result being 5 points and the worst result being 1 point. Considering the average values, the numerical methods that solve the neuron models in the fastest way are RK4 and AB. The numerical methods that solve with the highest accuracy are RK4 and AM. However, it should be noted that while AM is best at achieving spike behavior, it is worst at achieving burst behavior. When speed and error are considered together, it is understood that RK4 and AB are the best numerical methods for the realization of neuron models on RPi, with a score of 7,4 (The order of performance here is as follows: 2–4 very bad, 4–5 bad, 5–7 good, 7–10 best). Another preferred method is RKN. AM and ABM are not suitable methods to realize neuron models on the RPi.

VI. CONCLUSIONS

Researchers have proposed various neuronal models that mimic neural activity for a better understanding of brain structure. They differ from each other by various features such as membrane dynamics richness, biological accuracy, and complexity of their expressions [1]–[7]. Different platforms have been developed for digital and analog realization of the models [8]–[16]. These platforms have the following disadvantages:

1. VLSI platforms do not have the flexibility to be reconfigurable. The design needs to be recreated to apply the parameter change to the platform. In addition, with reduced transistor area, it becomes more sensitive to noise and is highly affected by changing environmental conditions [13].
2. Although FPAA platforms are reconfigurable, their capacity is limited. For neuron model realization, multiple FPAA elements need to be connected in parallel. As a result, they become more susceptible to noise [14].
3. In FPGA platforms, multiplier blocks, computation of hyperbolic functions, and propagation delays lead to high

costs. Even if computational costs are tried to be reduced with various approaches, the actual dynamics of the neuron is distorted, and the accuracy decreases. In addition, in some applications, computational costs become more difficult due to the high use of FPGA resources [15].

4. Neuron models are mathematical approximations defined in continuous time. On FPAA and FPGA platforms, continuous time definitions must be converted to discrete time. The Euler and Runge-Kutta methods are preferred for their simple expressions. However, they produce results with a certain amount of error from the actual result [14]–[16]. The suitability of other methods such as RKN, AB, AM, and ABM to solve neuron models with complex dynamics such as burst and chaotic has not yet been determined.

In this study, HH, ML, FHN, HR, and IZ neuron models are implemented with the Raspberry Pi 4 (RPi 4) microprocessor platform, which has a standard hardware, powerful processor architecture, facilitates multi-input operations thanks to its high RAM capacity, open source code, and has recently attracted the attention of researchers. In addition, the suitability of the RK4, RKN, AB, AM, and ABM mathematical methods for solving neuron models is compared. It has been demonstrated that RPi 4 is capable of realizing various neuron membrane dynamics such as spike, burst, and chaotic with high accuracy, without any memory limitation, with little effect on noise, with the solution methods compared. On the other hand, the RK4 and AB methods were found to be the most suitable for solving neuron models, while the ABM method was relatively inadequate; the AM method solved spike dynamics with high accuracy, while the AM method solved burst dynamics with high error. In light of the results obtained, the scope of the suitability of mathematical solution methods for neuron models has been expanded, and it is understood that RPi 4 is a new digital platform for neuron realization (the code is freely available for non-commercial use from: <https://github.com/vedatburakyucedag/A-Raspberry-Pi-Based-Hardware-Implementations-of-Variou-Neuron-Models.git>).

APPENDIX A

In this appendix, the numerical methods RK4, RKN, AB, AM, and ABM are described.

Fourth-order Runge-Kutta (RK4):

$$y_{i+1} = y_i + \frac{1}{6}h[k_1 + 2k_2 + 2k_3 + k_4], \quad (18)$$

$$k_1 = f(x_i, y_i), \quad (19)$$

$$k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right), \quad (20)$$

$$k_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2\right), \quad (21)$$

$$k_4 = f(x_i + h, y_i + hk_3). \quad (22)$$

Runge-Kutta New Version (RKN):

$$y_{i+1} = y_i + \frac{1}{6}h[k_1 + 2k_2 + 2k_3 + k_5], \quad (23)$$

$$k_1 = f(x_i, y_i), \quad (24)$$

$$k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right), \quad (25)$$

$$k_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2\right), \quad (26)$$

$$k_4 = f(x_i + h, y_i + hk_3), \quad (27)$$

$$k_5 = f\left(x_i + \frac{3}{4}h, y_i + \frac{1}{32}h(5k_1 + 32k_2 + 13k_3 - k_4)\right), \quad (28)$$

Four-step explicit Adams-Bashforth method (AB):

$$y_{i+1} = y_i + \frac{1}{24}h[55k_1 - 59k_2 + 37k_3 - 9k_4], \quad (29)$$

$$k_1 = f(x_i, y_i), \quad (30)$$

$$k_2 = f(x_{i-1}, y_{i-1}), \quad (31)$$

$$k_3 = f(x_{i-2}, y_{i-2}), \quad (32)$$

$$k_4 = f(x_{i-3}, y_{i-3}). \quad (33)$$

Three-Step Implicit Adams-Moulton Method (AM):

$$y_{i+1} = y_i + \frac{1}{24}h[9k_1 + 19k_2 - 5k_3 + k_4], \quad (34)$$

$$k_1 = f(x_{i+1}, y_{i+1}), \quad (35)$$

$$k_2 = f(x_i, y_i), \quad (36)$$

$$k_3 = f(x_{i-1}, y_{i-1}), \quad (37)$$

$$k_4 = f(x_{i-2}, y_{i-2}). \quad (38)$$

Adams-Bashforth-Moulton Method (ABM)

As a first step, the next value is estimated using the AB method. This value is then corrected by the AM method.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

- [1] E. M. Izhikevich, *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting*. The MIT Press, 2007. DOI: 10.7551/mitpress/2526.001.0001.
- [2] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve", *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952. DOI: 10.1113/jphysiol.1952.sp004764.
- [3] C. Morris and H. Lecar, "Voltage oscillations in the barnacle giant muscle fiber", *Biophysical Journal*, vol. 35, no. 1, pp. 193–213, 1981. DOI: 10.1016/S0006-3495(81)84782-0.
- [4] R. FitzHugh, "Impulses and physiological states in theoretical models of nerve membrane", *Biophysical Journal*, vol. 1, no. 6, pp. 445–466, 1961. DOI: 10.1016/S0006-3495(61)86902-6.
- [5] J. Nagumo, S. Arimoto, and S. Yoshizawa, "An active pulse transmission line simulating nerve axon", *Proceedings of the IRE*, vol. 50, no. 10, pp. 2061–2070, 1962. DOI: 10.1109/JRPROC.1962.288235.
- [6] J. L. Hindmarsh and R. M. Rose, "A model of neuronal bursting using three coupled first order differential equations", *Proceedings of the Royal Society B: Biological Sciences*, vol. 221, no. 1222, pp. 87–102, 1984. DOI: 10.1098/rspb.1984.0024.
- [7] E. M. Izhikevich, "Simple model of spiking neurons", *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, 2003. DOI: 10.1109/TNN.2003.820440.
- [8] E. Lazaridis, E. M. Drakakis, and M. Barahona, "Full analogue electronic realisation of the Hodgkin-Huxley neuronal dynamics in weak-inversion CMOS", in *Proc. of 2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2007, pp. 1200–1203. DOI: 10.1109/IEMBS.2007.4352512.
- [9] X. Hu, C. Liu, L. Liu, J. Ni, and S. Li, "An electronic implementation for Morris-Lecar neuron model", *Nonlinear Dynamics*, vol. 84, no. 4, pp. 2317–2332, 2016. DOI: 10.1007/s11071-016-2647-y.
- [10] F. A. Khanday, N. A. Kant, M. R. Dar, T. Z. A. Zulkifli, and C. Psychalinos, "Low-voltage low-power integrable CMOS circuit implementation of integer- and fractional-order FitzHugh-Nagumo neuron model", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 7, pp. 2108–2122, 2019. DOI: 10.1109/TNNLS.2018.2877454.
- [11] M. Heidarpur, A. Ahmadi, M. Ahmadi, and M. Rahimi Azghadi, "CORDIC-SNN: On-FPGA STDP learning with Izhikevich neurons", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 7, pp. 2651–2661, 2019. DOI: 10.1109/TCSI.2019.2899356.
- [12] M. Heidarpur, A. Ahmadi, and N. Kandalaf, "A digital implementation of 2D Hindmarsh-Rose neuron", *Nonlinear Dynamics*, vol. 89, no. 3, pp. 2259–2272, 2017. DOI: 10.1007/s11071-017-3584-0.
- [13] N. Korkmaz, I. Öztürk, and R. Kılıç, "The investigation of chemical coupling in a HR neuron model with reconfigurable implementations", *Nonlinear Dynamics*, vol. 86, no. 3, pp. 1841–1854, 2016. DOI: 10.1007/s11071-016-2996-6.
- [14] N. Dahasert, I. Öztürk, and R. Kiliç, "Implementation of Izhikevich neuron model with field programmable devices", in *Proc. of 2012 20th Signal Processing and Communications Applications Conference (SIU)*, 2012, pp. 1–4. DOI: 10.1109/SIU.2012.6204544.
- [15] E. L. Graas, E. A. Brown, and R. H. Lee, "An FPGA-based approach to high-speed simulation of conductance-based neuron models", *Neuroinformatics*, vol. 2, no. 4, pp. 417–436, 2004. DOI: 10.1385/NI:2:4:417.
- [16] P. Pourhaj, D. H.-Y. Teng, K. Wahid, and S.-B. Ko, "A novel scalable parallel architecture for biological neural simulations", in *Proc of 2010 IEEE International Symposium on Circuits and Systems*, 2010, pp. 3152–3155. DOI: 10.1109/ISCAS.2010.5537951.
- [17] S. Valadez-Godínez, H. Sossa, and R. Santiago-Montero, "On the accuracy and computational cost of spiking neuron implementation", *Neural Networks*, vol. 122, pp. 196–217, 2020. DOI: 10.1016/j.neunet.2019.09.026.
- [18] G. de Alteriis and C. M. Oddo, "Tradeoff between accuracy and computational cost of Euler and Runge Kutta ODE solvers for the Izhikevich spiking neuron model", in *Proc. of 2021 10th International IEEE/EMBS Conference on Neural Engineering (NER)*, 2021, pp. 730–733. DOI: 10.1109/NER49283.2021.9441070.
- [19] A. Tutueva, T. Karimov, and D. Butusov, "Semi-implicit and semi-explicit Adams-Bashforth-Moulton methods", *Mathematics*, vol. 8, no. 5, 2020. DOI: 10.3390/math8050780.
- [20] R. Siciliano, "The Hodgkin-Huxley Model: Its Extensions, Analysis and Numerics", 2012.
- [21] M. Xiao, "Stability analysis and Hopf-type bifurcation of a fractional order Hindmarsh-Rose neuronal model", in *Advances in Neural Networks – ISNN 2012. ISNN 2012. Lecture Notes in Computer Science*, vol. 7367. Springer, Berlin, Heidelberg, 2012, pp. 217–224. DOI: 10.1007/978-3-642-31346-2_25.
- [22] N. H. Sweilam and T. A. Assiri, "Numerical simulations of some real-life problems governed by ODEs", in *Numerical Simulation - From Brain Imaging to Turbulent Flows*. IntechOpen, 2016. DOI: 10.5772/63958.
- [23] A. Adili and B. Wang, "Random attractors for stochastic FitzHugh-Nagumo systems driven by deterministic non-autonomous forcing", *Discrete and Continuous Dynamical Systems - B*, vol. 18, no. 3, pp. 643–666, 2013. DOI: 10.3934/dcdsb.2013.18.643.
- [24] V. B. Yucedag and I. Dalkiran, "Raspberry Pi implementation of the Wilson-Cowan neural network with chemical synapse", in *Proc. of 2023 Innovations in Intelligent Systems and Applications Conference (ASYU)*, 2023, pp. 1–6. DOI: 10.1109/ASYU58738.2023.10296705.
- [25] *Raspberry Pi Documentation*, Raspberry Pi Foundation Inc., Cambridge, UK, 2009.
- [26] *MCP4725 Data Manual*, Microchip Technology Inc., Arizona, ABD, 2009.
- [27] J. C. Butcher, "Implicit Runge-Kutta processes", *Mathematics of Computation*, vol. 18, no. 85, pp. 50–64, 1964. DOI: 10.1090/S0025-5718-1964-0159424-9.
- [28] S. Kumar, R. Kumar, R. P. Agarwal, and B. Samet, "A study of fractional Lotka-Volterra population model using Haar wavelet and Adams-Bashforth-Moulton methods", *Mathematical Methods of Applied Sciences*, vol. 43, no. 8, pp. 5564–5578, 2020. DOI: 10.1002/mma.6297.
- [29] S. Vaidyanathan, "Anti-synchronization of the FitzHugh-Nagumo chaotic neuron models via adaptive control method", *International*

Journal of PharmTech Research, vol. 8, no. 7, pp. 71–83, 2015.
[Online]. Available:

<https://www.researchgate.net/publication/283566545>



This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution 4.0 (CC BY 4.0) license (<http://creativecommons.org/licenses/by/4.0/>).