A BIOPHYSICALLY ACCURATE FLOATING POINT SOMATIC NEUROPROCESSOR

Yiwei Zhang, José Nuñez-Yañez*, Joe McGeehan*, Edward Regan†, Stephen Kelly†*

Centre for Communications Research*, Faculty of Engineering, Faculty of Medicine and Dentistry†
University of Bristol
Email: Y.Zhang@bris.ac.uk, J.L.Nunez-Yanez@bris.ac.uk, J.P.McGeehan@bris.ac.uk

ABSTRACT

Biophysically accurate neuron models have emerged as a very useful tool for neuroscience research. These models are based on solving differential equations that govern membrane potentials and spike generation. The level of detail that needs to be presented in the model to accurately emulate the behaviour of an organic cell is still an open question, although the timing of the spikes is considered to convey essential information. Models targeting hardware are traditionally based on fixed point implementations and low precision algorithms which incur a significant loss of information. This, in turn, could affect the functionality of a bioelectronic neuroprocessor in an undefined way. In this paper, a 32-bit floating point reconfigurable somatic neuroprocessor is presented targeting an FPGA device for real-time processing. For each individual neuron, the dynamics of ionic channels are described by a set of first order kinetic equations. A dedicated CORDIC unit is developed to solve the nonlinear functions that regulate spike generation. The results have been verified using an experimental setup that combines an FPGA device and a digital-to-analogue converter.

1. INTRODUCTION

An organic neuron can be separated into four fundamental parts: somata, dendrites, axons and synapses. The soma plays the role of the central nonlinear processing unit, in which complex dynamics take place, resulting in action potential (spike) generation. The timing and the number of these action potentials are believed to carry information that is exchanged among neurons [1]. Although the literature suggests that the shape of the spike has no direct impact on neuronal coding, this shape still contains the information that regulates the change of the membrane voltage, such as the rise and fall slopes of the voltage curve, which may implicitly have an accumulating influence upon spike frequency. The complex interconnectivity among neurons is formed via a large number of synapses with a single neuron having in the order of tens of thousand synaptic connections. Although floating point arithmetic is the preferred option for the description of biophysically accurate neuron models using software approaches, such as the one used in the popular neuroscience tools NEURON [2] and GENESIS [3], hardware solutions tend to remove the biophysical accuracy at this level due to their complexity. As a result, simpler models that can be implemented using a limited amount of silicon resources are favoured.

The ultimate aim and motivation of the research presented in this paper is the realization of a bioelectronic neuroprocessor that combines artificial neurons grown in a silicon medium (FPGA) with organic cells cultivated in an electrolyte well, as illustrated in Fig. 1. For the artificial neural networks, we intend to use detailed compartmental models to account for different morphologies and neuronal plasticity. The compartments will be based on a real-time solution of the Hodgkin-Huxley model [4]. The Hodgkin-Huxley model is capable of describing the behaviour of spiking neurons accurately, taking biophysically nonlinear dynamics into consideration to mimic the spike generation of single-compartment cells [5]. To capture the significant characteristics of individual neurons, we have developed a 32-bit floating point dedicated somatic neuroprocessor which has been designed to meet the demands of high precision and real-time performance.

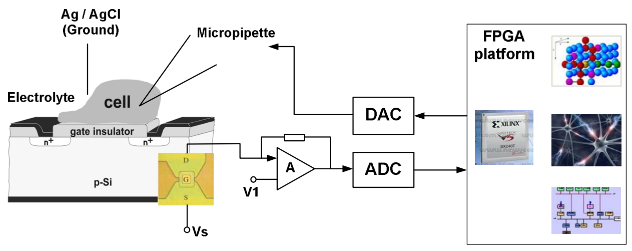

Fig. 1. Schematic of bioelectronic neuroprocessor, consisting of organic and artificial neurons.

2. PREVIOUS WORK

Until now, a large number of models and methods have been developed to address the problem of understanding how neurons process data inside the brain. Neuron modeling ranges from using detailed membrane compartments to abstract black-box models based on

*Sponsor acknowledgments for Oversea Research Student Award (ORSAS) and Postgraduate Research Award of University of Bristol and funding supplied by the Micron Foundation (USA).

digital, analogue or general-purpose processor platforms [5]. Analogue neurons offer low power and minimal silicon footprint but they suffer from low levels of reconfigurability, low precision and costly ASIC implementation. Some neuron simulators, such as NEURON [2] and GENESIS [3], analyze neuron behaviour using detailed compartment models. However, their performance is well below real-time applications when targeting general-purpose computing platforms as they result in systems which use huge amounts of resources [6]. On the other hand, FPGA devices comprise a large number of logic cells with rich interconnectivity resources, similar to the 'paralleling' feature of neural networks. Additionally, FPGA devices offer flexibility and reconfigurability, thereby provide for the adaptive computation of neural networks [7].

Neuron modeling based on FPGAs has concentrated on the development of large-scale neural networks, which emphasize the scale of the network and the connectivity among neurons [7]. In order to enlarge the density of the neural network, simpler or abstract models, such as Integrate-and-Fire and cascade models, are used to describe spike generation and morphology of the system [5, 8]. Experimental results have revealed that these models cannot provide an accurate description of real neurons, as some biophysically meaningful information, such as the exponential rule, are neglected to decrease the computation load. Neuroscience research does not fully understand how the details of the cell dynamics contribute to the signal processing in neurons and which properties are essential [5]. For this reason, we have initially focused on the details available in the Hodgkin-Huxley neuron, so simplifications can be done in a controlled way.

The availability of dedicated floating-point resources in modern FPGAs means that the precision loss introduced by conventional fixed point implementations can be avoided, opening a new approach for high-performance accurate neuron modeling.

3. MATHEMATICAL METHODS OF NEURON MODELLING

In this section, a brief introduction to the Hodgkin-Huxley model is presented, which is the fundamental reference model of our neuroprocessor. The Hodgkin-Huxley model is a single-compartment isopotential model which focuses on the effect of ionic currents on spike generation. More details can be found in reference [4]. The collection of equations for the Hodgkin-Huxley model are listed in equation (1)-(8).

$$C_M \frac{dV_M}{dt} = -\overline{g_{Na}} m^3 h (V_M - E_{Na}) - \overline{g_K} p^4 (V_M - E_K) \\ -\overline{g_L}(V_M - E_L) + I_{in} \qquad (1)$$

$$\frac{du}{dt} = \alpha_u (1-u) - \beta_u u \qquad (2)$$

where C_M is the membrane capacitance, V_M is the membrane voltage, I_{in} is the stimulating current, $\overline{g_i}$ is the constant maximal conductance of the i-species ion channel, E_i is the equilibrium potential for the i-species ion. Gating variables m, h and p in Equation (1) are dimensionless which follow the first-order kinetics formula as shown in Equation (2) by replacing u with m, h or p. α and β are defined in Equation (3)-(8) with empirical parameters $a_j, b_j,$ and θ_j $(j=1,2...6)$.

$$\alpha_m = -a_1 (V_M - \theta_1) / \left[\exp\left(\frac{V_M - \theta_1}{b_1} \right) - 1 \right] \qquad (3)$$

$$\beta_m = a_2 \exp\left(\frac{V_M - \theta_2}{b_2} \right) \qquad (4)$$

$$\alpha_h = a_3 \exp\left(\frac{V_M - \theta_3}{b_3} \right) \qquad (5)$$

$$\beta_h = 1 / \left[1 + a_4 \exp\left(\frac{V_M - \theta_4}{b_4} \right) \right] \qquad (6)$$

$$\alpha_p = a_5 (V_M - \theta_5) / \exp\left(\frac{V_M - \theta_5}{b_5} - 1 \right) \qquad (7)$$

$$\beta_p = a_6 \exp\left(\frac{V_M - \theta_6}{b_6} \right) \qquad (8)$$

Equation (9) gives the integration result of the first-order kinetics equation (2) with the assumption of constant membrane voltage during each integration time step. In our work, the exponential Euler method is used to solve the differential equation (1) with integration step of 0.1ms [9]. Since many biological behaviours obey the exponential change rule, the exponential Euler, in which membrane voltage is explicitly defined, is preferred, rather than the implicit solution given by the backward Euler method [10].

$$u(t) = \frac{\alpha}{\alpha+\beta} + [u(t_0) - \frac{\alpha}{\alpha+\beta}] \exp[-(\alpha+\beta)(t-t_0)] \qquad (9)$$

As previously indicated, intensive computations are involved in the Hodgkin-Huxley model. In floating point format, this model can mimic the dynamics of active neurons accurately. However, how to find an area-efficient solution is a challenging problem. Modern FPGAs offer abundant hardware resources that can meet all the stringent requirements of processing speed, timing performance and computation precision for the hardware realization of the Hodgkin-Huxley model.

4. SOMATIC PROCESSOR ARCHITECTURE

The architecture of the somatic processor which combines several floating-point ALUs working in parallel is

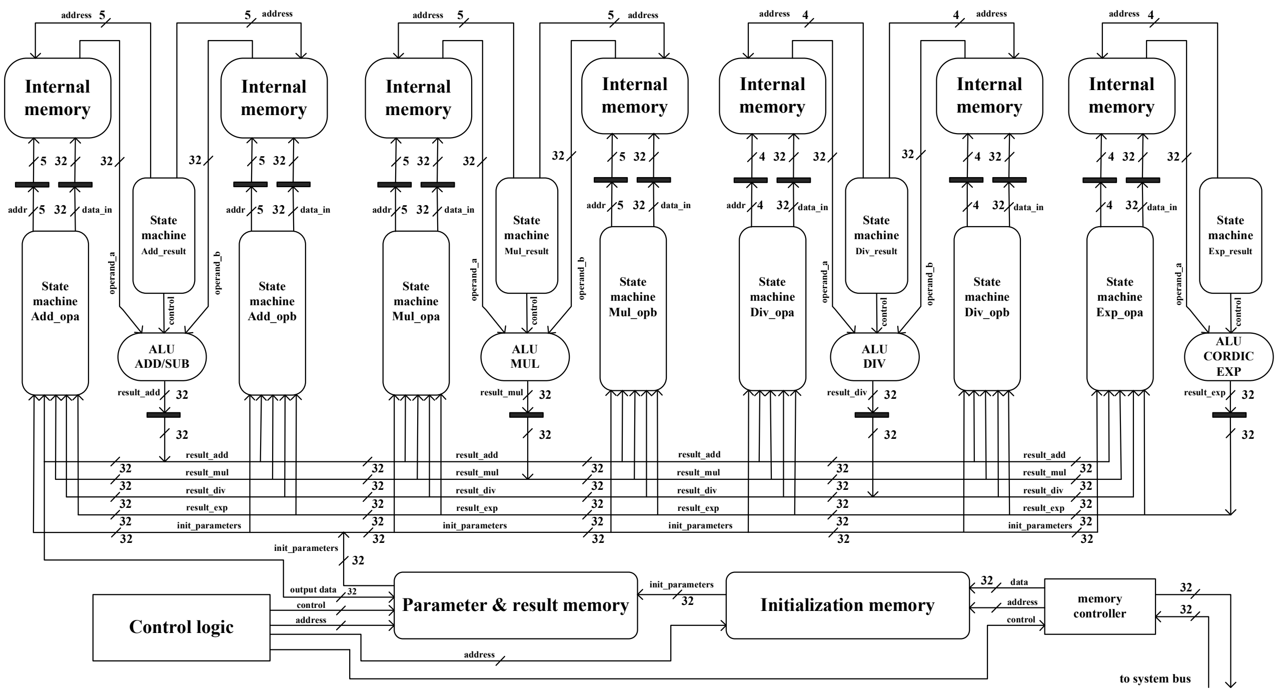

Fig. 2. Architecture of somatic neuroprocessor.

presented in the following sections. It solves the set of differential equations with tens of parameters as previously described. The specific problem posed by the exponential function is solved by a dedicated arithmetic circuit using the CORDIC algorithm [11]. In this architecture, 32-bit floating point data format is used to execute all the arithmetic operations following the IEEE 754 standard. The greatest benefit of the floating point data representation is the large dynamic range, which is suitable for high-precision neurocomputation applications.

4.1. Architecture of the Somatic Neuroprocessor

The architecture of the somatic neuroprocessor is based on 4 arithmetic-logic units (ALUs) and 11 finite state machine controllers (FSMCs) as illustrated in Fig. 2. All ALUs perform the computations in the format of a 32-bit floating point number and are realized by means of logic cells and DSP48a slices. The DSP48a core is a dedicated high-performance arithmetic component offered by Xilinx FPGAs, each consisting of two adders, a multiplier and two multiplexers [12]. The utilization of DSP48a slices improves computation performance of the high-precision ALUs. To avoid large multiplexers at input ports of the ALUs, all temporal variables are stored in internal RAMs rather than flip flops and multiplexers. The data paths among ALUs and RAMs are monitored by FSMCs and global control logic. All FSMCs are located at the input or output ports of the ALUs to fetch operands or dispatch results between internal memory and the ALUs.

The 4 ALUs perform addition/subtraction, multiplication, division and exponential functions, respectively. For each ALU, the operands are stored in their corresponding internal RAMs, and are read out in accordance with the current state of the FSMCs. All FSMCs collaborate with each other and control the sequence of computation according to a predefined state order. The state order is designed to make full use of the ALUs and reduce the whole computational duration, accounting for some cases where the operands cannot be provided to the ALUs in time. In doing so, the 4 ALUs run in parallel and are almost fully occupied during the computation with a small number of idle states.

The neuroprocessor is connected to the system bus through an initialization RAM, in which all parameters and input stimuli are stored. These parameters can be reconfigured by updating the contents of the RAM through the system bus. Once the computation is triggered, all the parameters are copied to a local parameter-and-result RAM that is updated by the neuroprocessor after each time step. The data feedback to the parameter-and-result RAM is used to compute the membrane voltage during the next time step. An array of internal memories is used to store temporary variables under the control of FSMCs. The control logic coordinates the operation among all the hardware modules.

4.2. Exponential Function Implementation

Traditionally, a look-up-table (LUT) approach is used to solve this nonlinear function using fixed point format numbers [13]. How to realize floating-point exponential functions in a way amenable to hardware poses a challenging problem. As no multipliers or dividers are required, the CORDIC algorithm is an efficient approach to implement hyperbolic functions [13], by which the exponential function can be expressed as listed in Equation (10a)-(10d).

$$e^{z} = \cosh z + \sinh z \tag{10a}$$

For hyperbolic functions, the CORDIC scheme is derived from the rotation at each iteration step:

$$X_{i+1} = X_i + d_i \cdot 2^{-i} \cdot Y_i \tag{10b}$$

$$Y_{i+1} = Y_i + d_i \cdot 2^{-i} \cdot X_i \tag{10c}$$

$$z_{i+1} = z_i - d_i \cdot \operatorname{artanh}(2^{-i}) \tag{10d}$$

where *i* is iteration integer variable, d_i contains the value of -1 and +1 to determine the direction of rotation. For convergence reason, some iteration steps (*i=4, 13, 40,…k, 3k+1*) must be repeated [13]. It should be noted that the basic CORDIC scheme is only directly applicable for those inputs whose absolute values are in the range of convergence. For hyperbolic functions, the range of convergence is $|z_{in}| \leq \theta_{max} \approx 1.1182$, where $\theta_i = artanh\ (2^{-i})$ [13]. Therefore, arbitrary input exponents need to be preprocessed until they are located in the convergence region.

We decided to develop our own CORDIC ALU instead of using the implementation available in the Xilinx IP library because that one is limited to fixed point applications. The architecture of the CORDIC ALU, as depicted in Fig. 3, consists of two parts: an exponent preprocessing block and a CORDIC core block. In the exponent preprocessing block, if the input exponent is positive and out of the range of convergence, it is translated to a fixed point number firstly to generate an address for LUT1, where the results of floor function *p = floor(input exponent/convergence limit)* is stored in fixed point format. Then this fixed point integer *p* is used as an address signal for LUT2 that contains the product of *p* and *convergence limit* in floating point format. Finally, this product is subtracted from the original input exponent to obtain the remainder, which is sent to the CORDIC core block. In the case of a negative exponent input, its complement is used in the preprocessing step to guarantee that the address signal *p* remains positive. In the preprocessing block, only one floating-point adder and small memories of about 8KB are employed, avoiding the need of a floating-point divider for the remainder operation. The output of the CORDIC core needs to be passed to the postprocessing circuit to compensate the effect of the preprocessing block. In doing so, an arbitrary input exponent is mapped within the convergence limit, making it suitable for the basic CORDIC algorithm. The fixed point number is only used as address signals of RAMs. All the computations are performed using 32-bit floating point number to avoid precision loss during the operation. Also, the amount of the memories required by the look-up-tables in the preprocessing block is limited, because the input exponents are in a limited range for neuronal dynamics.

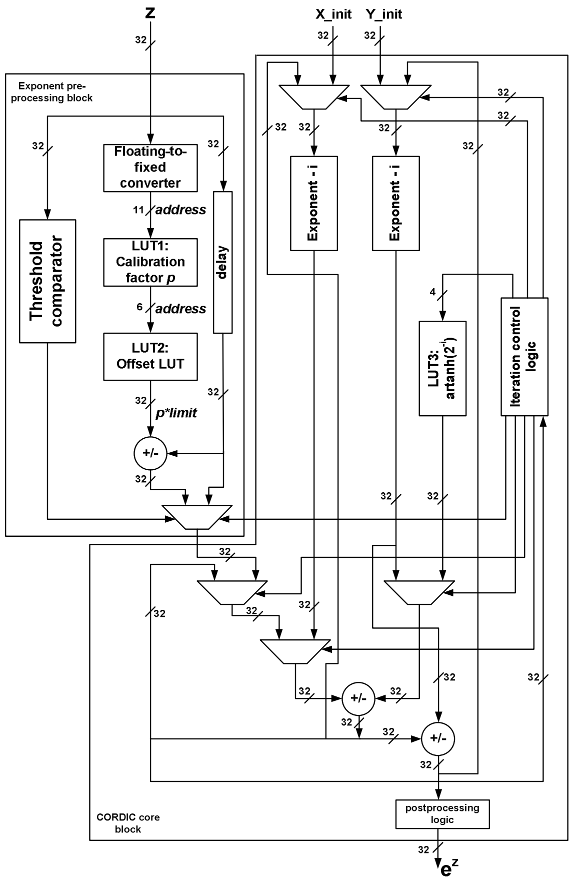

Fig. 3. 32-bit floating point iterative CORDIC ALU.

5. HARDWARE IMPLEMENTATION AND PERFORMANCE EVALUATION

In the previous sections, a 32-bit floating point dedicated somatic processor has been presented. This real-time computing platform can be implemented in any Xilinx device that supports the DSP48/DSP48a cores, such as the Spartan3aDsp or Virtex4/5 parts. The version targeting the Spartan3aDsp family has been implemented on a *xc3sd1800a* device, which achieves a clock frequency of 100MHz. The simulation temporal step is set to 0.1 ms, corresponding to 10 KHz sampling frequency in other organic cell-chip experiments [14]. The computation duration needed to output a new membrane voltage is about 2,000 cycles, depending on the execution times of the CORDIC preprocessing block. This means that a new membrane voltage can be obtained in around 20 μs so around 5 single-point real-time neurons can be mapped to a single processor. The exponential function is executed by the dedicated CORDIC hardware accelerator, in which the computation is iterated 9 times to obtain sufficient accuracy and the address signal *p* of LUT2 ranges from 0 to 63. The results of the CORDIC ALU are shown in Fig. 4.

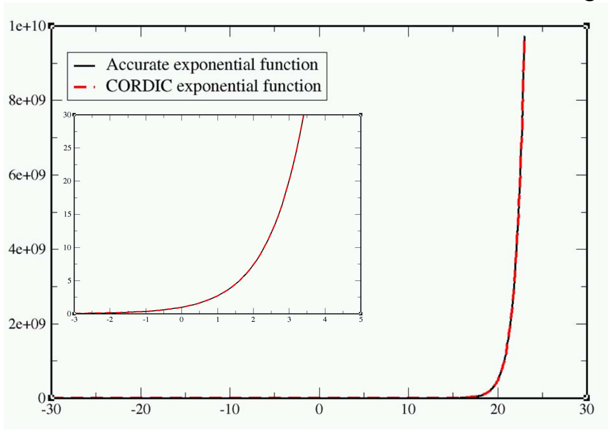

Fig. 4. Result of the CORDIC ALU. The solid line depicts accurate values computed using the C reference software model. The dashed line was computed by the CORDIC ALU. The two waveforms almost overlap each other.

Fig. 5 shows the spike generation result of the somatic neuroprocessor. The inset waveform is the result of reference model implemented in ANSI C. The output of the hardware realization is sampled at the interface between the FPGA board and the DAC board. The outputs are encoded in the range from 0V to +2V due to a postprocessing stage for data conversion in terms of the features of a 16-bit DAC device (*Texas Instruments DAC5682z*). In this experiment, a stimulating current pulse with amplitude of 15nA was injected from 10ms to 30ms. The surface area of the neuron is assumed to be 0.1mm^2. The logic resource utilization is reported in Table 1. Neuron parameters are summarised in Table 2. For one soma compartment with the simulation duration of 100ms, membrane voltage processing takes about 20ms with 100MHz system clock frequency, while the ANSI C model running on Intel Pentium4 1.8 GHz CPU and 1GB RAM PC spends 592ms on the same simulation. This performance implies that the P4 implementation is well below real-time requirements. The somatic processor can be used to implement several somatic compartments and if several processors run in parallel, a neural network can be constructed on the FPGA. The population of neurons and neuron compartments can grow as the amount of available hardware resources increases.

Table 1. Result of logic resource utilization

Mapping to device: xc3sd1800a-4fg676	
Number of Slice Flip Flops	5,862 out of 33,280 (17.6%)
Number of 4 input LUTs	6,490 out of 33,280 (19.5%)
Number of DSP48a	4 out of 84 (4%)

Table 2. The significant parameters of the HH model

Para	Value	Para	Value	Para	Value
C_M	1.0 μF/cm^2	a_1	0.1	a_4	1.0
E_{Na}	45.0 mV	b_2	-10.0	b_4	-10.0
E_K	-82.0 mV	θ_1	-45.0	θ_4	-40.0
E_{Cl}	-59.0 mV	a_2	4.0	a_5	0.01
E_{rest}	-70.0 mv	b_2	-18.0	b_5	-10.0
$\overline{g}_{Na}$	120.0 mS/cm^2	θ_2	-70.0	θ_5	-60.0
$\overline{g}_K$	36.0 mS/cm^2	a_3	0.07	a_6	0.125
$\overline{g}_{Cl}$	0.3 mS/cm^2	b_3	-20.0	b_6	-80.0
Area	0.1 mm^2	θ_3	-70.0	θ_6	-70.0

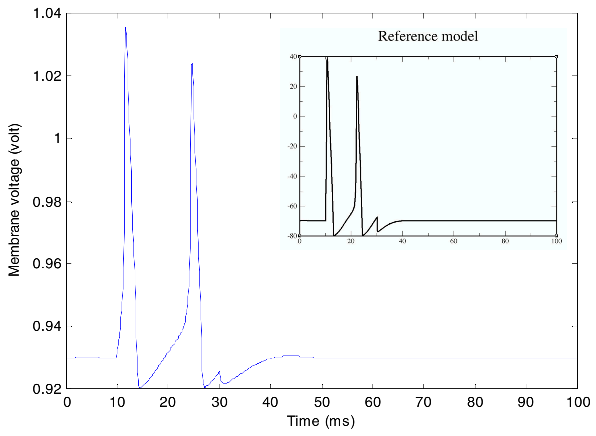

Fig. 5. FPGA output waveform. The inset figure was computed using reference model in ANSI C.

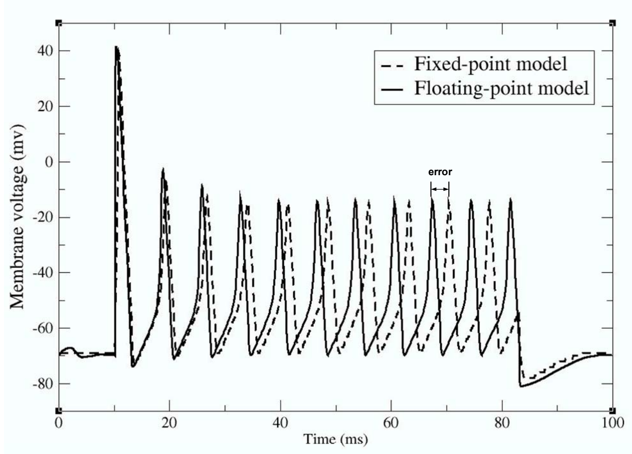

Fig. 6. Comparison of fixed-point and floating-point format. The solid line represents 32-bit floating point model; the dashed line is 32-bit fixed point model in the Q(32,15) format.

An experiment to analyze the effects of data precision has been conducted and the results are shown in Fig. 6. The solid line represents the 32-bit floating point model and the dashed line is the 32-bit fixed point model. It can be seen that the fixed-point error is accumulated, leading to a significant difference of the location of the spikes. Since when and where the action potentials occur defines the functional behaviour of the neuron, the impact of this error could alter the principles of neural information processing, specially on large-scale neural networks with complex interconnecting patterns as it is also suggested in [8].

6. CONCLUSION AND FUTURE WORK

In this paper, the architecture of a somatic neuroprocessor that mimics spike generation in organic neurons has been presented. The proposed system solves the Hodgkin-Huxley neuron model using 32-bit floating point precision. This model can be used as a basic building block to simulate complex neurons made up of hundreds of compartments or to construct neural network with each neuron represented by several compartments. In order to reduce hardware usage, dedicated arithmetic units are used in the design. Consequently, in comparison with reduced neuron models or fixed point algorithms, this system provides a more accurate description of neuronal dynamics and can meet the real-time performance requirements of a bioelectronic neuroprocessor. Furthermore, the Hodgkin-Huxley model can be used to further study neuron behaviour by keeping its mathematical form but simultaneously interpreting the terms in Equation (1) in different ways [15]. For example, the channel conductance can be replaced by inhibitory, excitatory and passive channels, rather than categorizing them with ionic species. This approach makes this neuroprocessor suitable for further analysis of neuronal dynamics, especially for synaptic modeling.

An experiment setup under development at the Centre for Communications Research (CCR) at Bristol is shown in Fig. 7, which combines real and artificial neurons communicating through a MEA (MicroElectrode Arrays) device. The MEA is a neuron sensor comprising neuron-to-silicon interface and analogue processing circuitry. The future work involves the extension of the artificial neural network by adding dedicated synaptic and dendritic neuroprocessors.

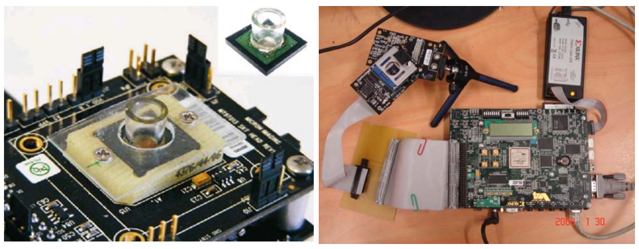
(a) (b)

Fig. 7. Photograph of the experiment setup under development. (a) the neuron cultivation well and the MEA board; (b) the prototype experimental setup, consisting of a MEA neuron sensor board, a FPGA board and a connection board.

7. REFERENCES

[1] W. Gertner and W.M.Kistler, "Spiking neuron models: single neurons, populations, plasticity," *Cambridge University Press,* 2002.

[2] M. L. Hines, N. T. Carnevale, "The NEURON simulation environment", *Neural Computation*, vol. 9, No. 6, pp. 1179-1209, Aug. 1997

[3] JM Bower, D Beeman, **"**The book of GENESIS: exploring realistic neural models with the GEneral NEural SImulation System", *Springer-Verlag New York, Inc*., 1998.

[4] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *Journal of Physiology*, Vol. 117, pp. 500–544 (1952).

[5] Andreas V. M. Herz, Tim Gollisch, "Modeling single-neuron dynamics and computations: a balance of detail and abstraction," *Science*, vol. 314. no. 5796, pp. 80 – 85, Oct 2006.

[6] R Brette, M Rudolph, T Carnevale, M Hines, "Simulation of networks of spiking neurons: A review of tools and strategies", *Journal of Computational Neuroscience*, vol. 23, No. 3, pp. 349-398, July 2007

[7] Jim Harkin, F. Morgan, "Reconfigurable platforms and the challenges for large-scale implementations of spiking neural networks," in *Conf. Field Programmable Logic and Applications*, 2008.

[8] L.P. Maguire, T.M. McGinnity, "Challenges for large-scale implementations of spiking neural networks FPGAs," *Neurocomputing***,** vol. 71, issues 1-3, pp. 13-29, Dec 2007.

[9] SW Hughes, M Lőrincz, DW Cope, V Crunelli, "NeuReal: An interactive simulation system for implementing artificial dendrites and large hybrid networks," *Journal of Neuroscience Methods*, vol. 169, issue 2, pp. 290-301, Oct. 2007.

[10] Michael V. Mascagni and Arthur S. Sherman , Numerical Methods for Neuronal Modeling in *Methods in Neuronal Modeling: From Ions to Networks*, edited by Cristof Koch and Idan Segev, *MIT* Press, pp 569-606, 1998.

[11] R Andraka, "A survey of CORDIC algorithms for FPGA based computers", *International Symposium on Field Programmable Gate Arrays*, pp. 191 - 200, 1998.

[12] *Xilinx* UG431 XtremeDSP *DSP48A* for *Spartan-3A* DSP FPGAs User Guide, *www.xilinx.com/support/ documentation/user_guides/ug431.pdf*

[13] DR Llamocca-Obregón, CP Agurto-Ríos, "A fixed-point implementation of the neural logrithm based on the expanded hyperbolic CORDIC algorithm," *Lat. Am. Appl. Res*. vol. 37, no.1, Mar. 2007.

[14] T. Schoenauer, S. Atasoy, N. Mehrtash, and H. Klar, "NeuroPipe-Chip: A digital neuro-processor for spiking neuralnetworks," *IEEE Transactions Neural Networks*, vol. 13, issue 1, pp. 205-213, Jan 2002.

[15] F. C. Hoppensteadt, "An introduction to the mathmatics of neurons*", Cambridge University Press*, 1986