33rd Annual International Conference of the IEEE EMBS
Boston, Massachusetts USA, August 30 - September 3, 2011

A Hardware-based Computational Platform for Generalized Laguerre-Volterra MIMO Model for Neural Activities

Will X. Y. Li, *Student Member*, *IEEE*, Rosa H. M. Chan, *Student Member, IEEE*, Wei Zhang, Ray C. C. Cheung, *Member*, *IEEE*, Dong Song, *Member, IEEE* and Theodore W. Berger, *Fellow, IEEE*

***Abstract*—A parallelized and pipelined architecture based on FPGA and a higher-level Self Reconfiguration Platform are proposed in this paper to model Generalized Laguerre-Volterra MIMO system essential in identifying the time-varying neural dynamics underlying spike activities. Our proposed design is based on the Xilinx Virtex-6 FPGA platform and the processing core can produce data samples at a speed of 1.33×10^6/s, which is 3.1×10^3 times faster than the corresponding C model running on an Intel i7-860 Quad Core Processor. The ongoing work of the construction of the advanced Self Reconfiguration Platform is presented and initial test results are provided.**

I. Introduction

The underlying spike transformations across brain regions and biological processes including synaptic transmissions, dendritic integrations, and spike generations are difficult to study *in vivo* because these processes are highly nonlinear, dynamical and often time-varying [1]. Adaptive filters based on the point process framework were applied to the nonlinear dynamical model developed to track time-varying systems [2][3]. However, the large number of actual neuronal units involved and long duration of experiments will require an efficient hardware model that can implement the method and reduce the computation time for intensive biological data analysis.

Berger et al. first developed a biomimetic model of the nonlinear dynamics in the hippocampal brain area of the rat and implemented it using a Field Programmable Gate Array (FPGA) [4]. Hsiao et al. later developed VLSI based neural prosthesis to replace the surgically removed hippocampal subregion and fabricated it using the TSMC 0.18um process [5]. They are proofs of concept of cognitive neural prosthesis based on the stationary single-input single-output (SISO) model. Our work is based on the time-varying multi-input multi-output (MIMO) Generalized Laguerre-Volterra Model (GLVM) which should delineate the neural dynamics in a more comprehensive and accurate way [6].

The major contribution of this paper consists of 4 parts:
1) We have developed a novel FPGA-based hardware model to prototype the GLVM. To the best of our knowledge, this work has never been reported in the previous literatures.
2) Simulation results show the hardware processing core can achieve a 3.1×10^3 times data throughput compared with the software approach.
3) Our hardware platform is extendable to a multi-FPGA-array architecture.
4) The utilization of the advanced Self Reconfiguration Platform (SRP) is proposed for further animal research.

Will X. Y. Li, Wei Zhang and Ray C. C. Cheung are with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong SAR, China (email: xiangyuli4@student.cityu.edu.hk; wezhang6@student.cityu.edu.hk; r.cheung@cityu.edu.hk).
Rosa H. M. Chan, Dong Song and Theodore W. Berger are with the Center for Neural Engineering, Department of Biomedical Engineering, University of Southern California, Los Angeles, CA 90089 USA (email: homchan@usc.edu; dsong@usc.edu; berger@bmsr.usc.edu).

II. System Identification

We propose the integration of 1) Generalized Volterra Model (GVM), 2) real-time Laguerre expansion, and 3) Steepest Descent Point Process Filter (SDPPF) to track the time-varying neural system using both natural spike inputs and outputs.

A MIMO system can be decomposed into a series of Multiple-Input, Single-Output (MISO) systems. The MISO models are identical in structure and each module projects to a separate output.

Each MISO model has physiologically plausible components which can be described by the following equations:

$$w = u(k,x) + a(h,y) + \varepsilon(\sigma) \qquad (1)$$

and

$$y = \begin{cases} 0 & \text{where } w < \theta \\ 1 & \text{where } w \geq \theta \end{cases} \qquad (2)$$

The input and output spike trains are denoted by x and y respectively. The hidden variable w represents the "pre-threshold membrane potential" of the output neuron. It is the summation of the "synaptic potential" u, the output spike-triggered "after-potential" a, and a Gaussian white noise input ε with standard deviation σ. The noise term models both the intrinsic noise of the output neuron and contributions from unobserved inputs. When w crosses the threshold θ, an output spike is generated and a feedback after-potential a is triggered which is then added to w. For the first order Volterra kernel k_1 with N inputs, the "synaptic potential" u can be expressed as

$$u(t) = k_0 + \sum_{n=1}^{N} (k_1^{(n)} * x_n)(t). \qquad (3)$$

k_0 is the baseline potential when the inputs are absent. k_1 is the impulse response of each spike from input x, which are functions of the time intervals τ between present time and the previous spikes. The feedback variable a can be expressed as

$$a(t) = (h * y)(t), \tag{4}$$

where h is the linear feedback kernel. The spike-triggered feedback captures the effects of ion channels such as calcium-activated potassium channels and also captures GABAergic feedback in the output firing patterns.

The Laguerre Expansion of the Volterra (LEV) kernel technique is used to reduce the number of open parameters to be estimated [2]. Using the LEV technique, both feedforward kernels k and feedback kernel h are expanded through L orthonormal Laguerre basis functions. Input and output spike trains x and y are convolved with j-th basis function b_j, such that the convolution products v are expressed as $v_j^{(n)} = b_j * x_n$ and $v_j^h = b_j * y$. Synaptic potential u and after potential a can be rewritten as:

$$u(t) = c_0 + \sum_{n=1}^{N} \sum_{j=1}^{L} c_1^{(n)} v_j^{(n)}(t) \tag{5}$$

and

$$a(t) = \sum_{j=1}^{L} c_h v_j^h(t). \tag{6}$$

The convolved functions v include the temporal dynamics. Another advantage of the Laguerre expansions is that the convolutions are generated in real-time. Let α_n $(0 < \alpha_n < 1)$ be the pole of the Laguerre basis functions of the n-th input x_n.

The convolved product v can also be computed iteratively at each time t [7]. Let $V^{(n)}(t)$=[$v_1^{(n)}(t) \cdots v_L^{(n)}(t)$],

$$V^{(n)}(t)A_1 = V^{(n)}(t-1)A_2 + \sqrt{1-\alpha_n^2}A_3 x_n(t). \tag{7}$$

In the above equation, $A_1 = I + \alpha_n I_+$; $A_2 = \alpha_n I + I_+$; A_3=[1 0 $\cdots$ 0]; I is an $L \times L$ identity matrix and I_+ is an upper shift matrix.

Given the recorded input and output spike trains x and y, u and a can be readily calculated based on the present values of v and the model coefficients in real-time. The estimated firing probability $P(t)$ is then calculated using the error function:

$$P(t) = 0.5 - 0.5erf\left(\frac{\theta - u(t) - a(t)}{\sqrt{2}\sigma}\right). \tag{8}$$

Without the loss of generality, θ and σ can be set to 0 and 1, respectively [2]. The model parameters to be estimated are the Laguerre coefficients C. Using the steepest descent point process filtering algorithm (SDPPF) [8], the parameter vector $C(t)$ is updated iteratively at each time step t:

$$C(t) = C(t-1) + R\left[\left(\frac{\partial \log P(t)}{\partial C}\right)' (y(t) - P(t))\right]_{C(t-1)}, \tag{9}$$

where R is learning rate. During adaptive parameter estimation, the gradient can also be generated in real-time. The derivatives with respect to the Laguerre coefficients are given as the products of v that are calculated in (7), such as $\partial u(t)/\partial c_0 = 1$, $\partial u(t)/\partial c_1^{(n)}(j) = v_j^{(n)}(t)$, $\partial a(t)/\partial c_h(j) = v_j^h(t)$. After observation of actual output spike train and prediction of firing probability in (8), R acts as the learning rate for the parameter estimations in (9). The estimated Laguerre coefficients are used to reconstruct the feedforward and feedback kernels [2].

III. Generalized Volterra Model Hardware Architecture

A major focus of this work is to utilize the intrinsic parallelism of modern FPGA device and dedicated IPs to prototype the GLVM, calculate the important coefficients for data analysis, while at the same time, achieve desirable chip resource utilization and data throughput.

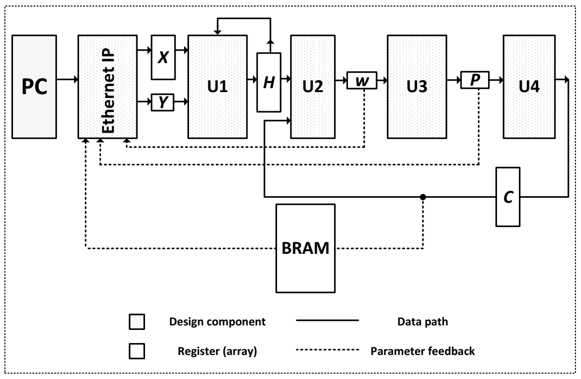

Fig. 1. Overview of the hardware architecture (U1: Convolution Unit; U2: Multiplication and Accumulation Unit; U3: Firing Probability Calculation Unit; U4: Laguerre Coefficients Updating Unit)

An overview of the hardware architecture is shown in Fig. 1. In the figure, U1-U4 are important processing units which combined forms the processing core. U1 and U2 are designed to update the pre-threshold membrane potential w. U3 calculates the firing probability P while U4 is the component which updates the Laguerre coefficients C. A $[1 + (N+1) \times L]$ register array stores the augmented horizontal vector H. The values of the input signal X and Y (Y is the feedback of neural output y) are previously stored in the *.txt* file (which is converted from the *.mat* file from Matlab). They are read and sent to the FPGA via Ethernet connection. The Ethernet IP in the FPGA receives the forward-transmitted data and relays the data to the processing core. The processing core performs DSP, calculates the important coefficients in our neural model, and stores the calculation results. The results are then sent back to the Ethernet core and back-forwarded to the PC end. The PC keeps track of the received data and saves them in regular time intervals.

The w updating unit contains $N+1$ Leading Zeros Detector (LZD) components, $N+1$ Vector Updating (VU) components and one Multiplication and Accumulation (MAC) component. Neurons in some parts of the brain, such as the hippocampus, have low firing frequency; so both X and Y could be sparse. The LZD component is designed to detect the zero elements in the input vectors. If zeros are detected, part of or the whole VU circuit, which is designed to perform the convolution function,

TABLE I
SYSTEM RESOURCE UTILIZATION

	Utilization	Available
LUTs	216,766 (75.5%)	297,600
FFs	381,371 (64%)	595,200
DSPs	2016 (100%)	2016
BRAMs	1, 6 or 12	1064

is hold. Signals will bypass the VU component and the H registers will be reset directly. This saves considerable amount of power given that the frequent updating of the VU registers will be prevented. New values of the H will be acquired after completing the convolutional routine of VU. The MAC component is of tree structure consisting of stages of adders. The products of H and C are added through stages of adder arrays. The size of the adder array shrinks by half from the utmost leaves to the root stage by stage. The value of w is acquired at the root.

U3 is designed to calculate estimated firing probability P, its gradient and $\partial \log P(t)/\partial C$ in (9). The calculation intensive stage in hardware includes the implementation of the error function *erf(x)* and the exponential function. Here, we transforms the calculation of the error function into the calculation of the exponential function by using the approximation for the error function reported in recent literature [9] as

$$erf(A) \cong \sqrt{1-e^{-\frac{4}{\pi}A^2}}. \quad (10)$$

Given that the variable in our current design model is limited to a certain range near the zero point, we can easily use the truncations of the Taylor series to do effective calculation.

IV. Results and Discussions

The design is partitioned in a sequence of two stages of pipelines. This division method is decided by the nature of the GLV algorithm itself. In the first stage, we complete updating the value of the augmented horizontal vector with the new inputs of neural spikings; in the second stage, we complete updating the Laguerre coefficients C. The calculation of the new Laguerre coefficients from the previously stored values of H and C can be divided into several time intervals during which values of important parameters such as w and P are generated and stored. In order to facilitate the later P & R, registers or register arrays are inserted into the combinational routes. Also, we assign the floating point arithmetic cores different latency values when configuring them with the Xilinx Core Generator tool.

We compare the system performance (without interface) with the software approach using C to process a same group of data. The PC running the C program boasts an Intel i7-860 Quad Core 2.8G CPU, 4GB memory and a Gigabit Ethernet interface with jumbo frame enabled. The experimental result shows that the Xilinx Virtex-6 XC6VSX475T FPGA-based hardware platform, can achieve a data throughput of 1.33×10^6 data frames/s while in software, this value can only come to 431.98 data frames/s. The FPGA resource utilization is shown in Table I.

Under our proposed hardware structure, we take 64 inputs from the X input set and 1 input from the Y input set for 1 round of convolution in 1 processing cycle. It is easy to see that we can use more FPGAs to treat more inputs from Y together with the same 64 inputs from X for higher-level parallel processing. The bandwidth cost of data transfer will be amortized and further speed up is possible.

We emphasize the work of the construction of the hardware IP library. It is the first step towards our future Self Reconfiguration Platform (SRP) [10]. The coming animal research may require real-time analysis of data under different application models. The hardware structure is also preferred to be real-time adaptive to the model changes and dynamically reconfigure itself to meet the new application needs, while at the same time, achieve the optimal compromise with regard to speed, resource utilization and power consumption. In the SRP, specific application information is transmitted to the FPGA via the external configuration access port (ECAP). And some specific circuits on the logic array can read the information and control other parts of the FPGA via the internal configuration access port (ICAP). It utilizes the resources of the hardware IP library to partially and dynamically reconfigure the device to best meet the application demands.

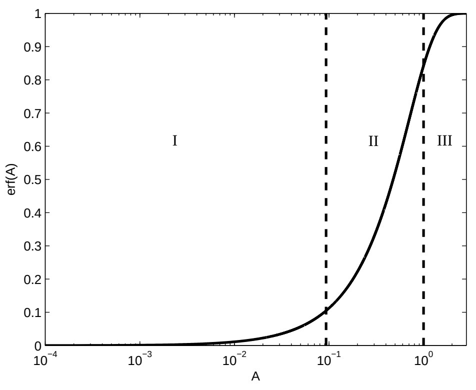

Fig. 2. Region division strategy for error function calculation. A is the input parameter.

The following attempt serves as a good example as an initial testing of the SRP. We may notice that when implementing the error function calculation circuit, we use Taylor series expansion (TSE) as the mathematical grounds. It is a very efficient approach, but one notable defect is that the chip resource utilization will increase dramatically when the data range expands. This is due to the fast adding up of items of the Taylor series. One counterplan is to use BRAM based partial table lookup, which is an advanced measure of the full LUT that write all the entries of the (x, erf(x)) mapping into a BRAM with proper data step size. We divide the error function curve into 3 regions as shown in Fig. 2. In Region I we use linear fit. This saves storage space. Instead of BRAM, here a

TABLE II
PROCESSING TIME OF EACH STAGE

	Parameter	Time
Hardware	T_1	7.50ms
	T_2	5.37ms
	$T_d + T_r$	2.4s
Software (C)	T_A	23.15s
	T_B	23.15s

simple DSP multiplier can achieve. In the Region II, we use traditional full LUT. And in Region III, we also use LUT; but as the gradient of the curve decreases to a much smaller scale, the data step size in the LUT could be relaxed without much precision lost. For absolute error less than 10^{-4}, the optimal point to separate Region I and II is (0.0932, erf(0.0932)) and the corresponding linear approximation is *g(A) = 1.1254A* for $A <| 0.0932 |$.

So after compilation, translation, mapping and P&R, the TSE based implementation and BRAM based implementation serve as two different bitstreams which are pre-stored in our host PC. We can first program the FPGA using the TSE architecture. After processing 10,000 data frames, the FPGA will send a request to the host PC for the reprogramming bitstream which is based on the partial LUT technique. Then the bitstream will be downloaded to the FPGA via Ethernet connection and the device will be reconfigured. The same 10,000 data frames will be processed afterward. We calculate the total processing time T_{total} of this procedure to be $T_{total} = T_1 + T_d + T_r + T_2$. T_1 and T_2 are the processing time of the TSE model and the partial LUT based model respectively; T_d is the downloading time and T_r the reconfiguration time as shown in Table II.

Also, we can calculate the minimal data size for processing if the hardware approach still takes an advantage. Let us assume that when the FPGA is reconfigured, the software will also shift from using one algorithm to another, namely, algorithm A to algorithm B (in our case, A and B are the same). If the hardware approach is preferred, the following equation must be satisfied:

$$T_1 + T_d + T_r + T_2 < T_A + T_B. \tag{11}$$

T_A and T_B are the execution time of the software. Let us assume the downloading and reconfiguring time do not change (and actually they vary little), we will have:

$$\frac{D}{T_{p1}} + \frac{D}{T_{p2}} + T_d + T_r < \frac{D}{T_{pA}} + \frac{D}{T_{pB}} \tag{12}$$

T_p is for data throughput. Based on the above equation and analysis, we can calculate the worst case data size for hardware implementation is $D_{min} = 33.67kB$, that equals to only 518 data frames. Also, from the above analysis, we can see that the hardware platform is faster than the software model in our general applications.

In case that the host PC and the SRP are not in the same location, then remote runtime reconfiguring is needed. In this case we can use the Xilinx Partial Reconfiguration Tool Kit (XPART) which is built on top of the ICAP API. The XPART is hardware independent are can be compiled for an embedded OS running on the SPR like uClinux. This offers our engineers a good opportunity to request remote bitstream using the HTTP or FTP protocol, debug on a standard workstation and manipulate the bitstream. This is also a research focus of our future work. The SRP shows great potential and will facilitate the analysis of neural spike train data due to its great adaptability and extensibility. The processing ability of the hardware platform will help us explorer the highly nonlinear, dynamical and time-varying biological processes of the animal brain.

V. ACKNOWLEDGEMENT

This work is supported by the City University of Hong Kong Start-up Grant 7200179.

REFERENCES

[1] M. D. Linderman, G. Santhanam, C. T. Kemere, V. Gilja, S. O'Driscoll, B. M. Yu, A. Afshar, S. I. Ryu, K. V. Shenoy, and T. H. Meng, "Signal processing challenges for neural prostheses," *IEEE Signal Processing Magazine*, vol. 25, pp. 18–28, 2008.

[2] R. H. M. Chan, D. Song, and T. W. Berger, "Tracking temporal evolution of nonlinear dynamics in hippocampus using time-varying volterra kernels," *Proceedings of the 30th IEEE EMBS Annual International Conference*, vol. 54, pp. 4996–4999, 2008.

[3] R. H. M. Chan, D. Song, A. Goonawardena, S. Bough, J. Sesay, R. E. Hampson, S. A. Deadwyler, and T. W. Berger, "Tracking hippocampal population nonlinear dynamics in rats learning a memory-dependnet task," *Proceedings of the 33rd IEEE EMBS Annual International Conference.*

[4] T. W. Berger, A. Ahuja, S. H. Courellis, S. A. Deadwyler, G. Erinjippurath, G. A. Gerhardt, G. Gholmieh, J. J. Granacki, R. Hampson, M. C. Hsiao, J. LaCoss, V. Z. Marmarelis, P. Nasiatka, V. Srinivasan, D. Song, A. R. Tanguay, and J. Wills, "Restoring lost cognitive function: hippocampal-cortical neural prostheses," *IEEE Engineering in Medicine and Biology Magazine*, vol. 24, pp. 30–44, 2005.

[5] M. C. Hsiao, C. H. Chan, V. Srinivasan, A. Ahuja, G. Erinjippurath, T. P. Zanos, G. Gholmieh, D. Song, J. D. Wills, J. LaCoss, S. Courellis, A. R. Tanguay, J. J. Granacki, V. Z. Marmarelis, and T. W. Berger, "VLSI implementation of a nonlinear neuronal model: a 'neural prosthesis' to restore hippocampal trisynaptic dynamics," *Proceedings of the 28th IEEE EMBS Annual International Conference*, pp. 4396–4399, 2006.

[6] D. Song, R. H. M. Chan, V. Z. Marmarelis, R. E. Hampson, S. A. Deadwyler, and T. W. Berger, "Nonlinear dynamic modeling of spike train transformations for hippocampal-cortical prostheses," *IEEE Transactions on Biomedical Engineering*, vol. 54, pp. 1053–1066, 2007.

[7] C. Boukis, D. P. Mandic, A. G. Constantinides, and L. C. Polymenakos, "A novel algorithm for the adaptation of the pole of laguerre filters," *IEEE Signal Processing Letters*, vol. 13, pp. 429–432, 2006.

[8] U. T. Eden, L. M. Frank, R. Barbieri, V. Solo, and E. N. Brown, "Dynamic analysis of neural encoding by point process adaptive filtering," *Neural Computation*, vol. 16, pp. 971–998, 2004.

[9] M. Nandagopal, S. Sen, and A. Rawat, "A note on the error function," *Computing in Science & Engineering*, vol. 12, no. 4, pp. 84–88, 2010.

[10] B. Blodget, P. James-Roxby, E. Keller, S. McMillan, and P. Sundararajan, "A self-reconfiguring platform," *Proceedings of the 13th International Conference on Field Programmable Logic and Applications (FPL'03)*, pp. 565–574, 2003.