35th Annual International Conference of the IEEE EMBS
Osaka, Japan, 3 - 7 July, 2013

A Customizable Stochastic State Point Process Filter (SSPPF) for Neural Spiking Activity

Yao Xin, Will X.Y. Li, Biao Min, Yan Han, and Ray C.C. Cheung, *Member, IEEE*

***Abstract*— Stochastic State Point Process Filter (SSPPF) is effective for adaptive signal processing. In particular, it has been successfully applied to neural signal coding/decoding in recent years. Recent work has proven its efficiency in non-parametric coefficients tracking in modeling of mammal nervous system. However, existing SSPPF has only been realized in commercial software platforms which limit their computational capability. In this paper, the first hardware architecture of SSPPF has been designed and successfully implemented on field-programmable gate array (FPGA), proving a more efficient means for coefficient tracking in a well-established generalized Laguerre-Volterra model for mammalian hippocampal spiking activity research. By exploring the intrinsic parallelism of the FPGA, the proposed architecture is able to process matrices or vectors with random size, and is efficiently scalable. Experimental result shows its superior performance comparing to the software implementation, while maintaining the numerical precision. This architecture can also be potentially utilized in the future hippocampal cognitive neural prosthesis design.**

I. INTRODUCTION

Cognitive neural prosthesis design is an emerging topic in neural engineering research. For long years, we have been endeavoring to develop a silicon-based prosthetic device which can be implanted into the mammalian brain. This device is expected to perform bi-directional communications between the intact brain regions and bypass the degenerated region [1]. A well-functioning mathematical model has to be established beforehand for effective processing of neural signals. Both parametric and non-parametric modeling techniques are explored [2], [3]. Due to the low-level physical mechanisms involved and demand for *a prior* postulation of model structure in parametric modeling, we refer to non-parametric methods which are more feasible for area-constrained implantable applications. The generalized Laguerre-Volterra model (GLVM), is such a rigorous and well-functioning mathematical model based on the non-parametric modeling paradigm [4]. The GLVM, upon its successful development, is first applied to prediction of mammalian hippocampal CA1 neuronal spiking activity based on detected CA3 spike trains—by which the expected neuroprosthetic function can be achieved. The GLVM uses a weighted sum of convolution products between model inputs and the orthonormal Laguerre basis functions, passing through a threshold trigger to generate the predicted model outputs [4], [5].

Yao Xin, Will X.Y. Li, Biao Min and Ray C.C. Cheung are with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong (e-mail: yaoxin2@student.cityu.edu.hk; xyli@ee.cityu.edu.hk; biaominhk@gmail.com; r.cheung@cityu.edu.hk).

Yan Han is with the Department of Information Science and Electronic Engineering, Zhejiang University, China (e-mail: hany@zju.edu.cn).

The prediction module of the generalized Laguerre-Volterra (GLV) algorithm is straightforward to be implemented on different platforms. However, Laguerre coefficients have to be estimated beforehand using the recorded input/output data; and this estimation function is oftentimes the most computational intensive stage in the whole calculation flow. The previous silicon-based implementations of the GLVM [6], [7] are based on the single-input and single-output (SISO) model, which serves as its most simplified form. However, in real situation, a model output is oftentimes affected by the spiking activity of multiple inputs. In 2011, Li *et al.* successfully implemented the multi-input, multi-output (MIMO) GLVM on the FPGA-based reconfigurable platform, achieving a remarkable speedup in model coefficients estimation compared with traditional software-based platforms [8]–[10]. However, the tracking method they had adopted is the Steepest Decent Point Process Filter (SDPPF), which is simple in mathematical representation but sacrifices certain levels of accuracy compared to other well-established methods such as the Stochastic State Point Process Filter (SSPPF) or the Kalman Filter, thus being less effective.

The SSPPF was proposed by Eden *et al.* in 2004 [11] and in 2009, Chan *et al.* [12] applied the algorithm to realize the estimation function of the GLVM, which is proven to be more effective than the SDPPF. A major improvement of SSPPF over SDPPF is the introduction of adaptive learning rate. In [8], only a constant learning rate is adopted, which can simplify the iterative computation procedure. However, brain activities of the behaving animals can be time-variant and subject to stochastic variations such as environmental changes. To be more realistic, the learning rate itself should be updated adaptively using the firing probability calculated in previous time ((9) of [8]) and the detected model output in present time. The SSPPF algorithm has thus far been only implemented in commercial software and run on a desktop setup, resulting in certain limitation in calculation process. Furthermore, there is no hardware architecture proposed for this algorithm to date to meet potential demand of portable and embedded platform for accurate coefficient estimation. Although there is an efficient hardware implementation of Kalman filter for neural ensemble decoding [13], it is only suitable for small size matrix.

In our work, we overcome the limitations of previous works and for the first time, implement the SSPPF on FPGA-based reconfigurable platform for more efficient and effective model coefficients estimation. Meanwhile, this new estimation module is practical for applying to the general framework of future cognitive prosthetic device.

978-1-4577-0216-7/13/$26.00 ©2013 IEEE

II. STOCHASTIC STATE POINT PROCESS FILTER

The general calculation flow applying the GLV algorithm can be found in [4] and [5]. Different filtering techniques can be adopted for GLV model coefficients tracking. The SSPPF is proposed to adaptively estimate model parameters in point process neural firing [11], which is defined as a linear evolution process with Gaussian errors. The time-varying parameter vector $C(k)$ and its covariance matrix $R(k)$ are updated by following recursive equations:

$$R(k)^{-1} = [R(k-1)+Q]^{-1} + \Big[\Big(\frac{\partial \log P(k)}{\partial C(k)}\Big)^T P(k)\Big(\frac{\partial \log P(k)}{\partial C(k)}\Big) - (y(k)-P(k))\frac{\partial^2 \log P(k)}{\partial C(k)\partial C(k)^T}\Big], \quad (1)$$

$$C(k) = C(k-1) + R(k)\Big[\Big(\frac{\partial \log P(k)}{\partial C(k)}\Big)^T (y(k)-P(k))\Big]. \quad (2)$$

where $P(k)$ is the firing probability of spike firing, k denotes discrete time bins, and $y(k)$ represents the new output information observed during the interval $(k-1,k]$. Applying (1) and (2) to the generalized Laguerre-Volterra model [12], the two equations can be rewritten into the below forms, as:

$$R(k) = [(R(k-1)+Q)^{-1} + k_1 M^T M]^{-1}, \quad (3)$$

$$C(k) = C(k-1) + k_2 R(k) M^T. \quad (4)$$

And k_1 and k_2 can be derived as follows:

$$k_1 = \alpha^2 P(k) + \beta[y(k)-P(k)], \quad (5)$$

$$k_2 = \alpha[y(k)-P(k)], \quad (6)$$

$$\alpha = \frac{1}{\sqrt{2\pi}P(k)} exp[-w(k)^2], \quad (7)$$

$$\beta = \frac{w(k)}{\sqrt{2\pi}P(k)} exp[-w(k)^2] + \frac{1}{2\pi P(k)^2} exp[-2w(k)^2]. \quad (8)$$

The calculation methods for $y(k)$, $P(k)$, and $w(k)$ are identical to the ones introduced in [8], which hence will not be elaborated here. Our architecture of SSPPF focuses on the calculation stages expressed by (3) and (4), wherein parallelism can be explored. A desktop computer is in charge of the other parts of calculation, and communicates with the FPGA in real time.

III. ARCHITECTURE DESIGN

The FPGA parallel architecture is built for the following reasons: 1) Traditional CPU platforms do have certain performance limitations. 2) Possible parallelism for accurate coefficient estimation using SSPPF can be explored efficiently. 3) We cannot exclude possible demand of potable and embedded platforms for real-time parameter estimation. 4) The former architecture for parameter estimation is only based on SDPPF, which compromises the accuracy.

Unlike most hardware designs [8], [13], the matrix size can be arbitrary in our architecture and meanwhile, it can be dynamically changed on the fly without pre-configuration. The size is only limited by on-chip memory resource. This architecture is also scalable in degree of parallelism, since the computing units are capable of scaling up. The data in our design is represented in single precision floating-point format. Fig. 1 shows the overall architecture for SSPPF. The

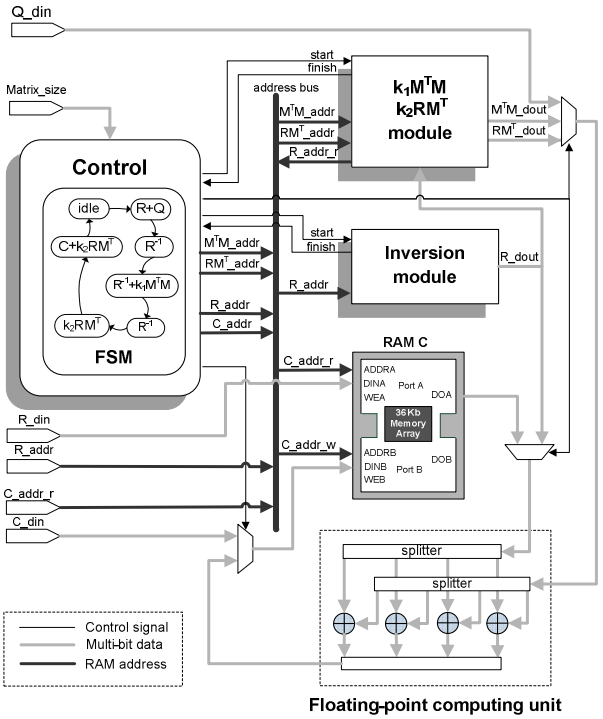


Fig. 1. Overall architecture of Stochastic State Point Process Filter.

top layer module implements two sub-modules according to major steps in SSPPF calculation: the computation of $k_1 M^T M$ and $k_2 R(k) M^T$ are conducted in $\mathrm{k_1M^TM\_k_2RM^T}$ module; the inversion of $R(k)$ is performed by matrix inversion module. Vectors and matrices involved are stored in Block RAMs which are arranged to true dual-port mode.

The computing unit consists of four floating-point operators at the top and one matrix inversion module. The intrinsic FPGA parallelism can be further explored by scaling up the computing units in parallel, with data-width for operation and storage increased accordingly. The parallel operations are performed in horizontal sweep fashion.

The matrix inversion module is shown in Fig. 2, which is based on the Gauss-Jordan elimination algorithm with partial pivoting. Two sets of operators and RAMs are needed for original and identity matrix simultaneously. The calculations in normalization and elimination phase are fully pipelined respectively. Major arithmetic operations include division, multiplication and addition. To achieve efficient pivot location, a comparator is set as the last node in the computing pipeline, directly receiving data flow from adder outputs. Pivot search is thus performed with only one extra clock in general. The found pivot row address is output for next-round row interchange when the elimination process is done.

In the $\mathrm{k_1M^TM\_k_2RM^T}$ module, the memory of vector M together with floating-point operators is shared by two different calculations, as shown in Fig. 3. We implement only a few floating-point operators in this module, since the corresponding computation is not the bottleneck affecting the overall performance.

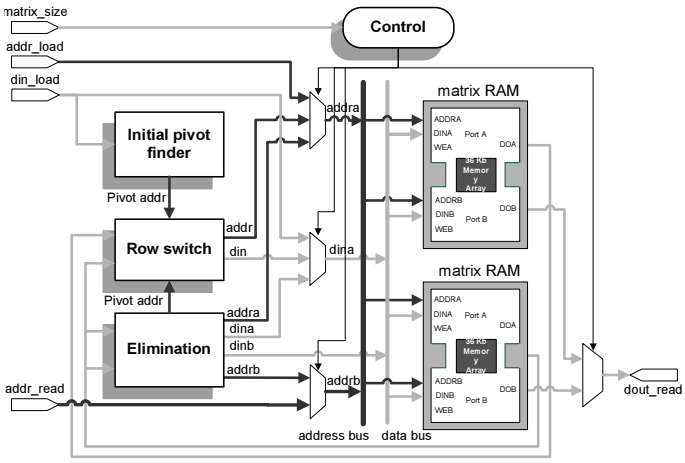


Fig. 2. Matrix inversion module.

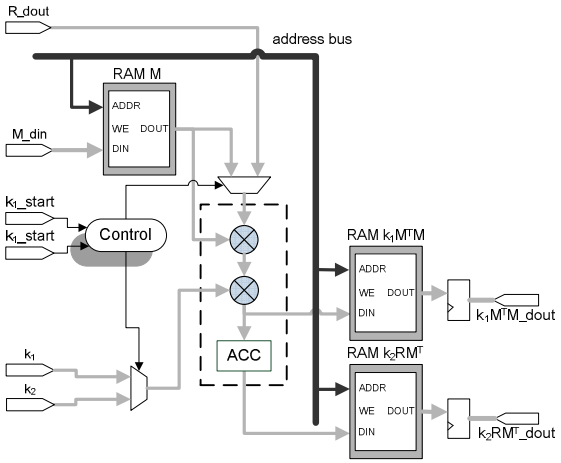


Fig. 3. $k_1M^TM\_k_2RM^T$ module.

IV. EXPERIMENTAL RESULTS

The architecture are synthesized, placed and routed in Xilinx Virtex-6 (XC6VLX240T-1) FPGA. Table I summarizes the resources occupied by the whole architecture and the main modules. We set the upper limit of matrix (vector) size N to 256 under consideration of available on-chip memory. Except for Block RAMs, the resource usage is rather small.

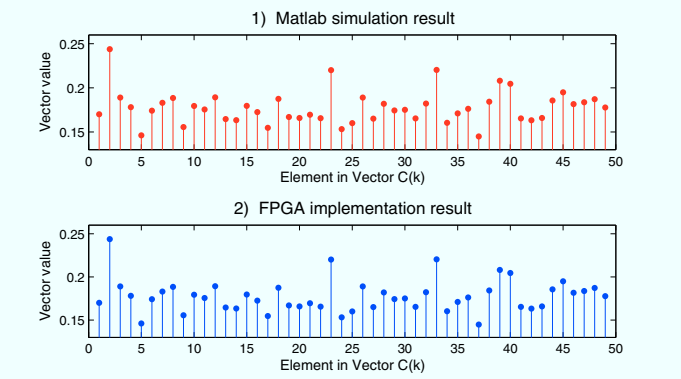
To verify the functionality of the SSPPF architecture, two sets of synthetic experimental data are taken as the initial input into the hardware. The vector $C(k)$ size N is set to be 49 and 139 respectively, which accounts for the coefficients to be estimated in a MISO model, and under certain conditions : 1) second-order self kernel / cross kernel are applied, and all kernels (besides the 0th) have 5 inputs, 2) number of Laguerre basis functions L is set at 3 [5], [8]. Meanwhile, the Laguerre coefficients estimation in Matlab version of SSPPF algorithm is simulated as a reference. The estimated vector $C(k)$ of one iteration are showed in Fig. 4.

Analysis is also done to evaluate the error between our adopted single precision and double precision floating-point realization. The FPGA design is compared with Matlab realization with double precision. Input initial coefficient vector and covariance matrix are randomly produced. We select absolute maximum of k_1 and k_2 under certain conditions to evaluate possible maximal error. 100 times of independent experiments to update $C(k)$ are done to calculate the average maximal Mean Error (ME) and Mean Squared Error (MSE) of FPGA implementation results for each iteration of SSPPF update. The results are presented in Fig. 5. Although maximal ME and MSE value both increase because of the cumulated error with growing size N, our design maintains the accuracy with MSE under 2×10^{-8}.

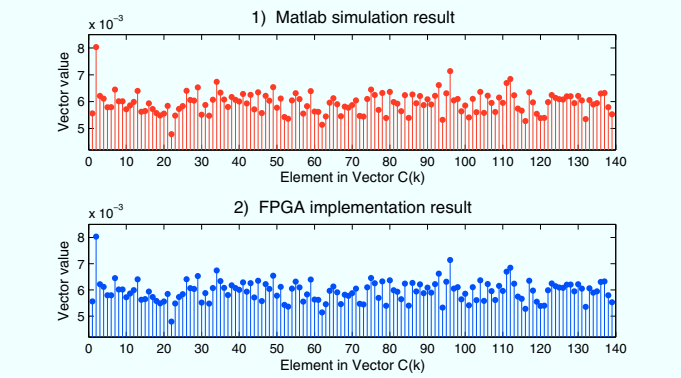
For performance evaluation, the FPGA architecture is compared with the software running on commercial CPU platform. The software implementation of SSPPF algorithm in C code has been compiled in Visual Studio 2010 and

TABLE I
RESOURCE UTILIZATION OF ARCHITECTURE DESIGN

	$k_1M^TM\_k_2RM^T$	Matrix Inversion	Overall Design
Slice LUTs	2344 (1%)	12068 (8%)	16667 (11%)
Slice Registers	2905 (1%)	14139 (4%)	19035 (6%)
RAMB36E1	57 (13%)	118 (28%)	176 (42%)
RAMB18E1	2 (1%)	1 (1%)	3(1%)
Max Frequency	261.575 MHz	220.653 MHz	208.247 MHz



(a) $N = 49$



(b) $N = 139$

Fig. 4. Filter result of FPGA implementation compared with Matlab simulation.

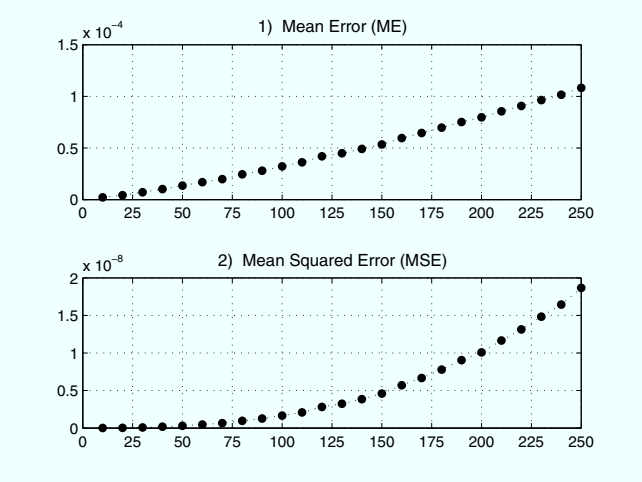


Fig. 5. Mean Error and Mean Squared Error of the FPGA results with different matrix size N.

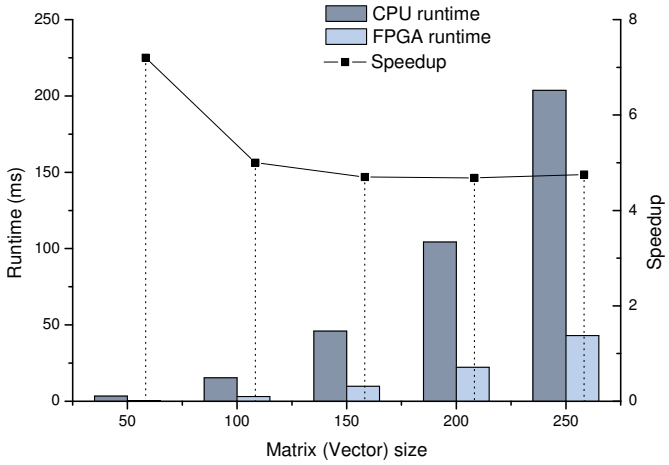


Fig. 6. Execution time comparison between hardware and software platform, with different matrix size N.

executed on the platform of Intel Core i5-250M @ 2.50GHz with 4GB RAM. Different datasets with coefficient number ranging from 50 to 250 are executed in software. FPGA architecture is driven by its possible maximal frequency clock in this test. Fig. 6 shows the execution time comparison between two platforms with several different matrix sizes. The FPGA-based hardware platform can achieve up to 7 times speedup compared to the software platform in calculation efficiency, as shown in Fig. 6. The speedup would be much more significant with exploration of higher degree of parallelism in future work.

V. CONCLUSIONS

The first Stochastic State Point Process Filter hardware architecture is proposed and implemented on FPGA. The architecture is capable of handling a wide dynamic range of data, while processing arbitrary size of coefficient vector without any pre-configurations. The upper bound of vector size is only limited by on-chip memory resource. The functionality of our architecture has been validated; error analysis results show that the architecture can effectively maintain numerical accuracy. Finally, the processing speed is compared between the hardware-based platform and the previous software-based platform. Experimental result indicates that the hardware platform can achieve up to 7 times speedup comparatively. In the future, the SSPPF computing architecture would be incorporated into our previously established reconfigurable framework for more efficient and accurate GLVM coefficients estimation.

REFERENCES

[1] T. W. Berger, D. Song, R. H. M. Chan, and V. Z. Marmarelis, "The Neurobiological Basis of Cognition: Identification by Multi-Input, Multioutput Nonlinear Dynamic Modeling," *Proceedings of the IEEE*, vol. 98, no. 3, pp. 356 –374, March 2010.

[2] "Brain in Silicon.", Stanford University, USA. [Online]. Available: http://brainsinsilicon.stanford.edu

[3] S. Hill and H. Markram, "The Blue Brain Project," in *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, Aug. 2008, p. clviii.

[4] D. Song, R. Chan, V. Marmarelis, R. Hampson, S. Deadwyler, and T. Berger, "Nonlinear Dynamic Modeling of Spike Train Transformations for Hippocampal-Cortical Prostheses," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 6, pp. 1053 –1066, June 2007.

[5] D. Song, R. H. M. Chan, V. Z. Marmarelis, R. E. Hampson, S. A. Deadwyler, and T. W. Berger, "Nonlinear modeling of neural population dynamics for hippocampal prostheses," *Neural Networks*, vol. 22, pp. 1340–1351, 2009.

[6] T. Berger, A. Ahuja, S. Courellis, S. Deadwyler, G. Erinjippurath, G. Gerhardt, G. Gholmieh, J. Granacki, R. Hampson, M. C. Hsaio, J. LaCoss, V. Marmarelis, P. Nasiatka, V. Srinivasan, D. Song, A. Tanguay, and J. Wills, "Restoring lost cognitive function," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 24, no. 5, pp. 30 – 44, Sept.-Oct. 2005.

[7] M. C. Hsiao, C. H. Chan, V. Srinivasan, A. Ahuja, G. Erinjippurath, T. Zanos, G. Gholmieh, D. Song, J. Wills, J. LaCoss, S. Courellis, A. Tanguay, J. Granacki, V. Marmarelis, and T. Berger, "VLSI Implementation of a Nonlinear Neuronal Model: A "Neural Prosthesis" to Restore Hippocampal Trisynaptic Dynamics," in *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*, 30 2006-Sept. 3 2006, pp. 4396 –4399.

[8] W. X. Y. Li, R. H. M. Chan, W. Zhang, R. C. C. Cheung, D. Song, and T. W. Berger, "High-Performance and Scalable System Architecture for the Real-Time Estimation of Generalized Laguerre-Volterra MIMO Model From Neural Population Spiking Activity," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 4, pp. 489 –501, Dec. 2011.

[9] W. Li, R. Chan, D. Song, T. Berger, and R. Cheung, "A dual mode fpga design for the hippocampal prosthesis," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, 28 2012-Sept. 1 2012, pp. 4579 –4582.

[10] R. Chan, D. Song, A. Goonawardena, S. Bough, J. Sesay, R. Hampson, S. Deadwyler, and T. Berger, "Tracking the changes of hippocampal population nonlinear dynamics in rats learning a memory-dependent task," in *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, Aug.30-Sept.3 2011, pp. 3326 –3329.

[11] 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, no. 5, pp. 971–998, May 2004.

[12] R. Chan, D. Song, and T. Berger, "Nonstationary modeling of neural population dynamics," in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, Sept. 2009, pp. 4559 –4562.

[13] X. Zhu, R. Jiang, Y. Chen, S. Hu, and D. Wang, "FPGA implementation of Kalman filter for neural ensemble decoding of rat's motor cortex," *Neurocomputing*, vol. 74, no. 17, pp. 2906 – 2913, 2011.