We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

5,300
Open access books available

130,000
International authors and editors

155M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

25

Design and Implementation of Leading Eigenvector Generator for On-chip Principal Component Analysis Spike Sorting System

Tung-Chien Chen[1,2], Kuanfu Chen[2], Wentai Liu[2] and Liang-Gee Chen[1]
[1.] *Graduate Institute of Electronic Engineering, National Taiwan University Taiwan*
[2.] *Electrical Engineering Department, University of California, Santa Cruz USA*

1. Introduction

On-chip implementation of neural signal processing along with the recording circuitry can significantly reduce the data bandwidth, and is a key to enable the wireless neural recording system with a large amount of electrodes (Zumsteg et al., 2005). Without such data processing, large amount of data need to be transferred to a host computer, and typically a cable is required. In this case, patients and test subjects are restrained from free movement, which impedes the progress in fundamental neuroscience research and the advance in closed-loop neural prosthetic devices.

Several approaches to achieve the bandwidth reduction have been investigated. One example is to transmit the encoded version of the whole neural waveform by using either lossless or lossy compression algorithms. In Oweiss et al., 2003, a 30-fold data reduction is demonstrated by using wavelet transformation and variable length coding algorithms. However, more data reduction is still desired. Another approach is to detect the spike events with threshold methods, and transmit either the binary event streams or the time stamps of the detected events (Olsson & Wise, 2005, Harrison, 2003). The compression performance increases to more than 100-fold data reduction. However, the significant loss of information limits the ability of classification and sorting of the individual neuron signal sources.

A promising approach to achieve the bandwidth reduction is to extract spike features immediately after spike detection on the implant site (Oweiss et al., 1977, Letelier & Weber, 2000, Hulata et al., 2002). Only the event times and some additional features about classification are transmitted after the signal processing. This approach achieves a similar data reduction as the threshold method does while preserving the capability for the neuron-to-neuron discrimination. The principal component analysis (PCA) (Zumsteg et al., 2005, Oweiss et al., 1977) and wavelet transformation (Letelier & Weber, 2000, Hulata et al., 2002) are currently the most widely used tools in this approach.

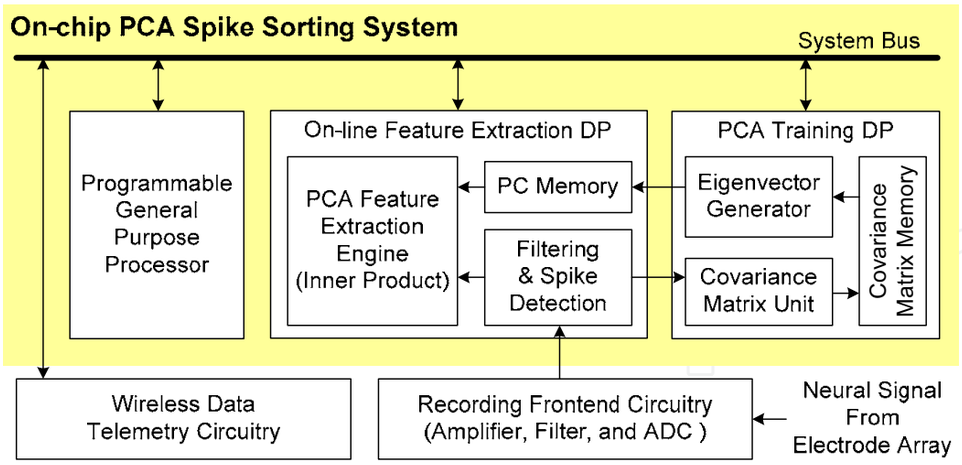

Fig. 1. The proposed on-chip system for a PCA-based spike sorting. The system has several heterogeneous processors with application specific functionalities reflecting the needs of spike sorting. Dedicated processors (DPs) are designed to accelerate the computationally intensive spike sorting algorithm. A programmable general purpose processor (GPP) is embedded for the system controlling and scheduling.

There are many architectures (Andra et al., 2002, Huang et al., 2004, Kamboh et al., 2007) that have been designed for wavelet transformation. However, there have not been seen a demonstration of prototype chip propose to achieve for wavelet-based spike sorting. In this paper, we propose to achieve the first hardware prototype in the form of an integrated circuit for PCA-based approach.

For PCA-based spike sorting, the PCA algorithm finds an optimal linear transformation which reduces d-dimensional spike waveform to h-dimensional feature scores (d>>h) in such a way that the information is maximally preserved in terms of minimum mean squared error. Note that d is the sample number of spike waveforms while the h is the desired number of principal components (PCs). In general, the spike waveforms with similar feature scores are corresponding to the same firing neuron. After the dimension reduction, the classification algorithm such as K-means (Kanungo et al., 2002, Ding & He, 2004)} or Mean-shift (Comaniciu & Meer, 2002) can be more effectively applied to sort the spikes into clusters corresponding to the different firing neurons.

The PCA feature extraction has two major phases---the parameter training phase and the on-line processing phase. In the parameter training phase, the algorithm collects the detected spikes, constructs the covariance matrix, and then calculating the corresponding eigenvectors. The major characteristic vectors, the PCs that can optimally differentiate neurons in a least square error term, are the first few eigenvectors with largest eigenvalues. In the on-line processing phase, the feature scores are extracted by projecting the detected spike waveforms on the PCs that are calculated in the training phase. The operation of inner product between the extracted spikes and the trained PCs is required for the vector projection. Note that the trained parameters may need to be updated through the periodic re-training in order to reflect the environment perturbations for long-duration experiments (Shenoy et al., 2006).

Fig. 1 shows the proposed on-chip PCA-based spike sorting system. The system has several heterogeneous processors with application specific functionalities reflecting the needs of

spike sorting. The operations on raw neural data such as noise filtering, spike detection and feature extraction require the most computation. Dedicated processors (DPs) are used to accelerate these computationally intensive tasks, utilizing customized parallel architectures and memory hierarchies. The PCA training DP collects the detected spike events and then calculates the co-variance matrix and the corresponding eigenvectors. After the training, the leading eigenvectors are transmitted to the on-line feature extraction DP and stored in the PC memory. The feature extraction DP then generates the feature scores of the following detected spike events by projecting them on the leading eigenvectors. Apart from DPs, a programmable general purpose processor (GPP) is embedded for the system controlling and scheduling. The GPP can also provide some flexibility in the algorithm development.

In realizing this on-chip PCA-based spike sorting system, the most challenging problem is to design a hardware unit to calculate leading eigenvectors. There are many algorithms to calculate eigenvectors from a covariance matrix (Golub et al., 1996, Roweis, 1998, Schilling & Harris, 2000, Sirovich, 1987), but most of them can hardly be mapped into an efficient VLSI architecture. The most well-known algorithm is the Cyclic Jacobi's method (Golub et al., 1996) based on eigenvalue decomposition (EVD). It generates all eigenvalues and eigenvectors after diagonalizing the symmetric covariance matrix. However, EVD has high computational complexity. The architecture design for matrix diagonalization is very complicated and will be expensive in silicon area. Expectation maximizing (EM) (Roweis, 1998) is proposed for PCA with less computation complexity compared to EVD method. However, it requires matrix inverse operation in both the E-step and M-step for each of the iteration, and the matrix inverse operation also cannot be efficiently implemented. Furthermore, EM algorithm may not converge to a global optimal and a good initial setup is required. The power method (Schilling & Harris, 2000) is another less computationally expensive method but can compute only the most leading eigenvector. Another snap-shot algorithm (Sirovich, 1987) also requires matrix inversion and is not hardware-friendly. Note that very small silicon area and power consumption are usually required by the implantable hardware in order to avoid neural tissue damage.

In this chapter, based on a computationally fast and hardware friendly algorithm (Sharma & Paliwal, 2007), the first VLSI architecture to calculate the leading eigenvectors is proposed for the on-chip PCA-based spike sorting system. The reminder of this paper is organized as follows. In section 2, the algorithm is introduced and then validated for the spike sorting through software simulations. In section 3, the low power and low area VLSI architecture is proposed with a flipped structure and adaptive level shifting method. Section 4 presents the implementation and fabrication results and Section 5 concludes this work.

2. Iterative Eigenvector Distilling Algorithm

In this section, a computationally fast and hardware friendly algorithm to find the desired number of leading eigenvectors is introduced. We will go through the algorithm first and then summarize the advantages of the algorithm in terms of hardware implementation. Finally, neural data are used in the software simulation to validate the algorithm for PCA-based spike sorting. For the detailed mathematic proof of the algorithm, please refer to (Sharma & Paliwal, 2007). To facilitate the description, we name this algorithm iterative eigenvector distilling algorithm.

1, Choose h, the number of eigenvectors required to estimate,
Choose r, the iteration number of eigenvector distilling,
Compute covariance matrix $\sum_{cov}$, and set p and q to 1.
2, Initialize eigenvector φ_p up, e.g. randomly.
3, Do the eigenvector distilling process :
$\varphi_p = \sum_{cov} \varphi_p$
4, Do the Gram – Schmidt orthogonalization process :
$\varphi_p = \varphi_p - \sum_{j=1}^{p-1} (\varphi_p^T \varphi_j) \varphi_j$
5, Normalize up φ_p by dividing it by its norm :
$\varphi_p = \varphi_p / \|\varphi_p\|$
6, Increase counter $q = q + 1$ and go to step 3 until q equals r.
7, Increase counter $p = p + 1$ and go to step 2 until p equals h.

Table 1. Fast PCA algorithm based on the iterative eigenvector distilling algorithm.

2.1 Algorithm Description

Table 1 depicts the fast PCA algorithm based on the iterative eigenvector distilling algorithm. "h" is the required number of the PCs. "r" is the algorithm iteration number, and "$\Sigma_{\_cov}$" is the covariance matrix calculated from the detected spike waveforms. In the beginning, the eigenvectors, "φ_p", are initialized randomly. Afterwards, the leading eigenvectors of the covariance matrix are calculated one by one in a reducing order of dominance. The calculation of each eigenvector has r iterations and each of the iterations has two procedures---the eigenvector distilling process and the orthogonal process.

The key of this algorithm is to intensify the major component on the initial eigenvector through continuously multiplying the initial eigenvector with the covariance matrix. This procedure is called the eigenvector distilling process. The most PC can be simply derived after several iterations of this distilling process. For the remaining h-1 PCs, the orthogonal process is required. In order to continuously intensify the pth PC on the initial eigenvector, the previously measured p-1 components are removed from the intermediate results of "φ_p" by the orthogonal process after every iteration of distilling process. Note that the Gram-Schmidt method is used in our orthogonal process.

This algorithm has several advantages in terms of hardware implementation. The first one is the simple math operation. This algorithm is free from eigenvalue decomposition. The matrix diagonalization, symmetric rotation, and matrix inverse are not required. Second, the algorithm exactly meets the requirement without calculating the eigenvalues as well as the remainder minor eigenvectors. This fact combined with the simple operation results in the low computation complexity. Third, the algorithm can globally converge in a few iterations without the need for any specific initial setting. Also, the algorithm has a very regular procedure. As a result, the presented algorithm is computationally efficient and hardware friendly, and is a good starting point for VLSI implementation.

2.2 Simulation Results

We realized the iterative eigenvector distilling algorithm in Matlab, and used the neural data download from Quian Quiroga to validate the algorithm for PCA-based spike sorting. The "eig" function, a standard Matlab function to generate the eigenvectors, is used as our

benchmark (Anderson et al., 1999). Note that the nonlinear energy operation (NEO) algorithm (Kim & Kim, 2000) is adopted as our spike detection method. After spike detection, the detected spike waveforms are aligned horizontally and vertically according to their peaks and 8/12 samples are used before/after the peak to represent each spike waveform. Fig. 2 illustrates the mean squared error between the benchmark and the iterative eigenvector distilling algorithm with different iterations. According to the simulation results, the eigenvector usually converges within five iterations when its eigenvalue is much larger than the eigenvalue of the previous eigenvector. Otherwise, it takes around 10 to 15 iterations for the convergence.

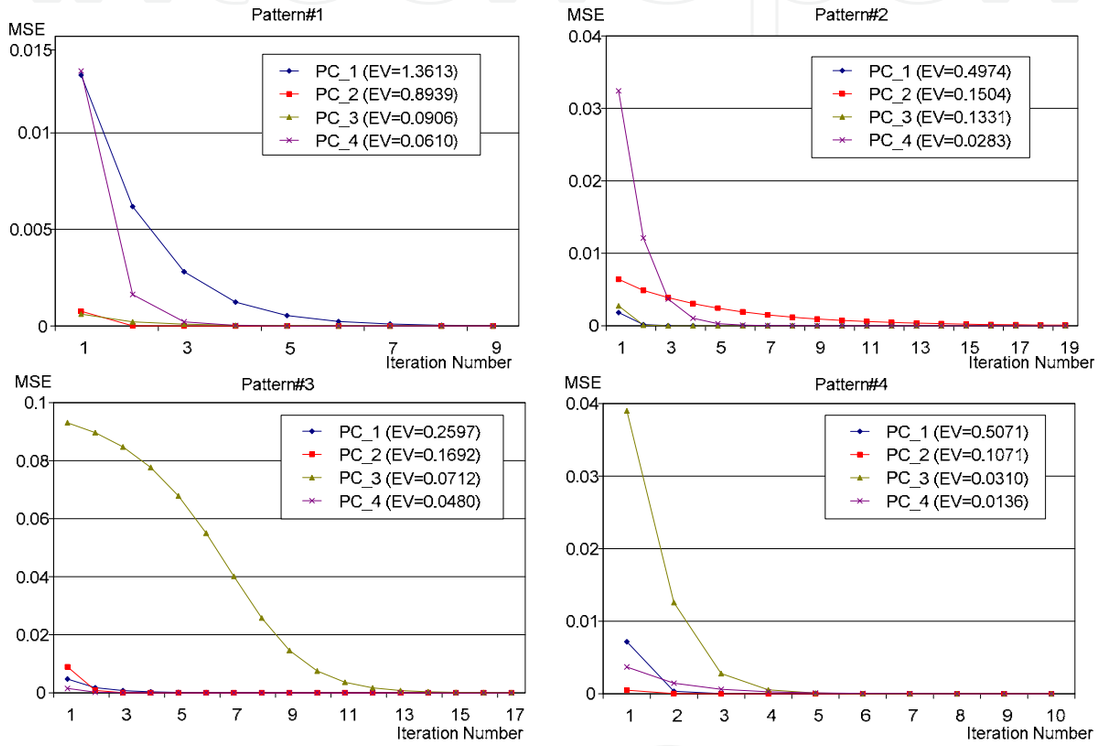

Fig. 2. Mean squared error (MSE) between the benchmark and the iterative eigenvector distilling algorithm of different iterations. EV and PC in the figures denote the abbreviation of eigenvalue and principal component. According to the simulation results, the eigenvector converges within five iterations when its eigenvalue is much larger than the eigenvalue of the previous eigenvector. Otherwise, it takes around 10 to 15 iterations for the convergence. The test patterns are downloaded from Quian Quiroga. Pattern #1 to #4 are C_Easy1_noise005, C_Easy2_noise005, C_Difficult1_noise005, and C_Difficult2_noise005.

3. Architecture Design

In this section, based on the iterative eigenvector distilling algorithm, two techniques are proposed to enable the efficient VLSI implementation. A flipped structure is used to save the power and silicon area by discarding the division and square root operations in the

orthogonalization process. The adaptive level shifting scheme is applied to achieve the highest accuracy in a fixed-point processing system with only a small bit-width. Finally, the architecture as well as the corresponding processing schedule is designed for the modified iterative eigenvector distilling algorithm.

$$
\begin{aligned}
&for(p = 1:4)\\
&\{\quad \varphi_p = [\ 1,\quad 1,\quad ....,\quad 1,\ ]^T)\\
&\quad for(i = 1:10)\\
&\quad \{\quad \varphi_p = \Sigma_{cov}\varphi_p\\
&\qquad for(j = 1:p-1)\\
&\qquad \{\quad \varphi_p = \varphi_p - (\varphi_p^T\varphi_j)\,\varphi_j \quad \}\\
&\qquad \varphi_p = \varphi_p / \|\varphi_p\| \quad \}\}
\end{aligned}
$$

Fig. 3. The pseudo-code of the original iterative eigenvector distilling algorithm. The$\Sigma_{_cov}$ is the covariance matrix of the spike waveforms, and the φ is the demanded eigenvector. Suppose the demanded number of leading PCs is four, and the iteration number for each PC is ten.

3.1 Flipped Structure

In order to clearly explain the proposed techniques, we represent the original iterative eigenvector distilling algorithm in a pseudo code format shown in Fig. 3. The "$\Sigma_{\_cov}$" is the covariance matrix of the spike waveforms, and the "φ" is the demanded eigenvector. Suppose that four leading PCs are required, and the iteration number for each PC is ten. In the original algorithm, four kinds of math operations are required---addition, multiplication, division, and square root. Generally speaking, division and square root hardware units require much more silicon area and consume much more power compared with multipliers and adders. In order to optimize the power consumption and silicon area, the flipped structure is proposed to discard these hardware-expensive operations.

First, we discard the normalization process of $\varphi_p = \varphi_p / \|\varphi_p\|$, and change the orthogonal process to $\varphi_p = \varphi_p - (\varphi_p{}^T\varphi_j / \|\varphi_j\|)\ (\varphi_j / \|\varphi_j\|)$. Then, we multiply the whole equation by $\|\varphi_j\|^2$. Since $\|\varphi_j\|^2 = \varphi_j{}^T\varphi_j$, the orthogonal process finally becomes $\varphi_p = (\varphi_j{}^T\varphi_j)\,\varphi_p - (\varphi_p{}^T\varphi_j)\,\varphi_j$. In this way, the norm of the previously calculated PC is flipped to the dividend part in the orthogonal process. The division and square root operations are thus replaced by addition and multiplication. The silicon area and power consumption are thus saved by means of reusing the uncomplicated processing units of the adders and the multipliers.

3.2 Adaptive Level Shifting Scheme

After the flipped structure, the φ can be easily represented in a fixed-point integer number during the processing. It should be advantageous since the fixed-point integer DSP system is very friendly in terms of VLSI implementation. However, the dynamic range of φ increases rapidly during the iterations. For example, suppose the input covariance matrix is a 32x32 matrix, and each entry has 16-bit precision. The dynamic range of φ is increased by 16+5 bits for every eigenvector distilling process. If the current dynamic range of φ_j is n bits, the dynamic range of φ_p is increased by (2xn+5) bits for each orthogonal process. After several iterations, the final dynamic range become prohibitively large, which impedes a low area and low power implementation.

Quantization and saturation to a pre-defined bit-level is a general solution to deal with this problem in a fixed point DSP system. When the level of the processed signal can be well predicted, this method can usually result in a good trade-off between the hardware cost and signal accuracy. However, the variation of neural signals is very large for different living individual, system setup and applications. The signal level cannot be well predicted from such high dynamic range during the iterations. Another solution is to represent all intermediary data in floating point numbers and use a floating point DSP system for calculating. The floating point system can efficiently use the full range of the limited bit-width to represent as much information as possible. However, the floating point DSP system is very complicated and not cost-efficient in terms of area per bit and power per bit.

An adaptive level shifting scheme is proposed to optimize the hardware in terms of processing accuracy per hardware cost. The idea is to use the floating point concept in a fixed point DSP system. It is realized by dynamically increasing the quantization level according to the signal level until the limited bit-width can completely represent the quantized signals for each processing step. Fig. 4 (a) shows the pseudo code of the proposed flipped structure combining with the proposed adaptive level shifting scheme. After the eigenvector distilling process and the orthogonalization process, the level check and shift procedure is applied to adaptively compress the dynamic range according to the current level. The level check and shift procedure is shown in Fig. 4 (b). "bw" is the pre-defined bit-width of the system outputs of the final eigenvectors. During the level check and shift procedure, φ_p is continuously rounded by 2 until it can be completely represented in "bw" bits.

```
for(p = 1 : 4)
{   φp = [ 1,  1,  ....,  1, ]
    for(i = 1 : 10)
    {   φp = Σcov φp
        Level_Check_And_Shift(φp)*
        for(j = 1 : p − 1)
        {   φp = (φj^T φj) φp − (φp^T φj) φj
            Level_Check_And_Shift(φp)*}}}
```

(a)

```
* Level_Check_And_Shift(φp) :
While(max(φp) ≥ 2^(bw−1) || min(φp) < −2^(bw−1))
{   φp = (φp + 1) >> 1   }
```

(b)

Fig. 4. (a) The pseudo-code of the modified iterative eigenvector distilling algorithm with the flipped structure and the adaptive level shifting scheme. For the flipped structure, the normalization process is discarded, and the norm of the calculated PC, $||\varphi_j||$, is flipped to the dividend part in the orthogonal process. The division and square root operations are thus replaced by addition and multiplication operations. After the processing of the eigenvector distilling or the orthogonalization, the level check and shift procedure is applied to compress the dynamic range of the intermediate results according to their signal levels. (b) The level check and shift procedure. During the procedure, φ_p is continuously rounded by 2 until it can be completely represented in a pre-defined bit-width. The adaptive level shifting scheme optimizes the hardware in terms of processing accuracy per hardware cost by using the full range of the limited bit-width to represent as much information as possible. Note that the "max(*)" function extracts the largest value in the input vector while the "min(*)" function extracts the smallest value.

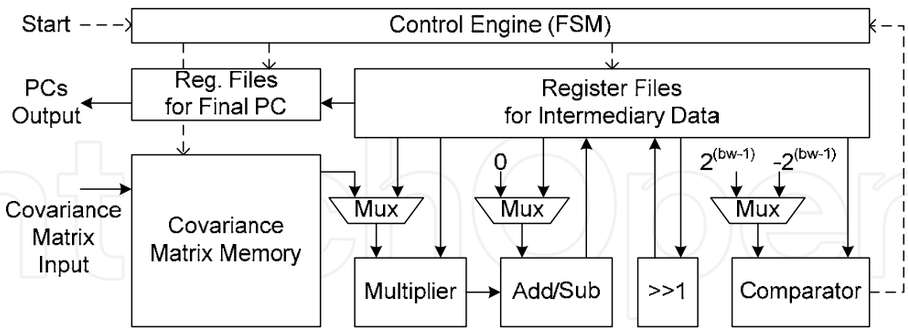

Fig. 5. The block diagram of the proposed architecture for the leading eigenvector generation. The multiplier and adder (also used as a subtractor) units are used for the eigenvector distilling process and the orthogonalization process. The right-shift and comparator units are used for the level check and shift procedure. The whole algorithm is folded into these four processing units and processed sequentially. All the intermediary data are stored in the register files. The control engine is responsible for the scheduling and resource allocation.

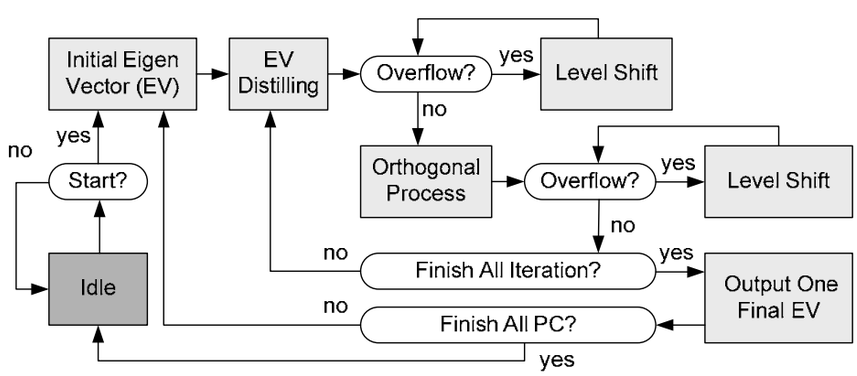

Fig. 6. The main finite state machine in the control engine. Each eigenvector distilling state takes (nxn) cycles. The orthogonal process for each pre-calculated eigenvector takes (nx4) cycles. Each overflow checking and level shift states take n cycles.

3.3 Architecture Design

Based on the modified algorithm, the block diagram of the proposed architecture for the leading eigenvector generation is shown in Fig. 5. The input is a covariance matrix of the detected spike waveforms. The outputs are several leading eigenvectors of the covariance matrix, or the PCs of the detected spike waveforms. Four major processing units are implemented. The multiplier and adder (also used as a subtractor) units form a multiply-accumulate (MAC) structure and are used for the eigenvector distilling process and the orthogonalization process. The right-shift and comparator units are used for the level check and shift procedure. The whole algorithm is folded into these four processing units and processed sequentially. All the intermediary data are stored in the register files. The control

engine constructed of finite state machines (FSMs) is responsible for the scheduling and resource allocation during the processing.

After the architecture is constructed, the next step is to do the scheduling and resource allocation. Fig. 6 shows the main FSM of the control engine. Suppose each spike waveforms has n samples, and $\Sigma_{\_cov}$ is an nxn matrix while φ is an nx1 vector. During the state of eigenvector distilling, "$\Sigma_{\_cov}$" and "φ_p" are input to MAC and the new "φ_p" is stored back to the register file. Because of the serial processing, every eigenvector distilling state takes nxn cycles. During the orthogonal process, "$\varphi_j^T\varphi_j$" is first computed, and two inputs of MAC are both "φ_j ". Afterwards, "φ_p " and "φ_j " are input to MAC for "$\varphi_p^T\varphi_j$". Then, "$\varphi_j^T\varphi_j$" and "φ_p" are input for "$(\varphi_j^T\varphi_j)\ \varphi_p$". As the final step in the orthogonal process, the MAC is initialized with "$(\varphi_j^T\varphi_j)\ \varphi_p$". "$\varphi_p^T\varphi_j$" and "$\varphi_j$" are input with the subtraction mode. After this orthogonalization process, the "φ_j" component is removed from "φ_p". The result is also stored back to the register files. Note that the orthogonal process to remove each pre-calculated eigenvector, "φ_j", takes nx4 cycles. During the overflow checking state, "φ_p" is input to the comparator and compared with $2^{(bw-1)}$ and $-2^{(bw-1)}$. The checking result is fed back to the control engine. If an overflow occurs, the FSM will enter the level shift state, and "φ_p" is input to the right shift engine to quantize the signal by 2. This procedure will continue until the overflow checking fails. It takes n cycles to pass each overflow checking and level shift state.

Spec.: Max. Bit Width		Entire Core		Covariance Matrix Memory	Processing Units + Register Files
I/O	Internal Circuit	Area (μm^2)	Power (μW)	Area (μm^2)	Area (μm^2)
2	11	45996	93	17667	26465
4	17	70027	150	20859	47303
6	23	95054	204	25007	68182
8	29	119956	255	29154	88937
9	32	132021	282	31228	98929
10	35	145624	308	33302	110457
12	41	171074	364	37449	131760
14	47	196965	417	41597	153503
16	53	222950	469	45744	175341

Table 2. Synthesized results of different processing accuracy.

Spec.: samples per Spike Waveform	Entire Core		Covariance Memory Matrix	Processing Units + Register Files
	Area (μm^2)	Power (μW)	Area (μm^2)	Area (um^2)
16	79209	152	18135	59322
32	132021	282	31228	98929
64	255495	521	75481	178071

Table 3. Synthesized results of different sample number per spike waveform.

4. Implementation Results

In the previous section, the first VLSI architecture is designed to generate the leading eigenvectors for the PCA-based spike sorting. However, defining the hardware

specifications in the spike sorting system to meet the application requirements under the minimum hardware cost is still an opened issue. In this section, we use 90 nm 1P9M process to synthesize the proposed leading eigenvector generator for various specifications. Through the simulation, the PCs generated by our verilog hardware model are compared to those generated by the standard Matlab function. Combining our hardware with the software-based classification algorithm, we also demonstrate the tradeoff between the sorting performance and the hardware cost. Finally, this eigenvector generation unit is integrated with other processors to complete a PCA-based spike sorting system and fabricated in .35 μm 2P4M process. We hope that the report in this section acts as a good reference for those who intend to define and implement a closed-loop neural prosthesis in the future.

4.1 Synthesized Results

There are four hardware parameters that can be specified in this design. The first one is the accuracy, including the bit width of input covariance matrix and the bit width of output PCs. The second one is the sample number of spike waveforms. The third and fourth are the required PC number and the iteration number for eigenvector distilling process. With a given operation frequency, the silicon area and power consumption are highly influenced by the first and second parameters, while the processing capability is influenced by the second, third, and fourth parameters. The processing capability is defined as the number of channels that can be trained in PCA algorithm within a period of time.

Table 2 reports the synthesized results of different processing accuracies. The input bit-width specifies the precision of the given covariance matrix while the output bit-width specifies the precision of the required PCs. The sample number of spike waveforms is fixed to support as large as 32 samples while the PC number and iteration number can go up to four and 128 respectively. Note that if the input/output (I/O) bit-width is n, the maximum bit-width of internal circuit is (3xn+5) which happens after the orthogonal process. The size of the on-chip static random access memory is (32x32xn) bits to store the covariance matrix. The area and power are reported in 90 nm 1P9M process at 1MHz operation frequency. When the bit width goes high, the area of covariance matrix memory, register files, and processing units increase in order to store and process more data. The hardware costs almost linearly increase in this case.

Table 3 reports the synthesized results of different sample numbers of spike waveforms. This time the I/O bit-width is fixed to 9 bits. The PC number and iteration number can still go up to 4 and 128 respectively. If the sample number of spike waveform is m, the size of the on-chip static random access memory is (mxmx9) bits. When the sample number of spike waveforms goes high, the dimensions of $\Sigma_{\_cov}$ and φ increase. This fact increases the area of the covariance matrix memory and the register files. The area cost also linearly increases in this case.

Table 4 shows the hardware capability with different hardware parameters. The processing capability is defined as the number of channels that can be trained in PCA algorithm within one minute and the required seconds that can train 1000 channels. The number is reported for the worst case (which requires the maximum cycles for level checking and shifting) and with iteration number of 20, required PC number of 4, and 1MHz operation frequency. In the maximum specification of 64 samples per spike and 16 bits bit-width of each input spike

sample and output eigenvector sample, our hardware can perform PCA parameter training for 90 channels within one minute. It requires 666 seconds to train 1000 channels.

Samples per Spike (#)	I/O Bit-width (bit)	Required Cycles for each Channel (#)	Channel Number per Minute (#)	Required Time to train 1000 Channels (sec)
64	16	666k	90	666
32	16	246k	244	246
32	9	192k	312	192
16	9	73k	822	73

Table 4. The hardware capability with different hardware parameters.

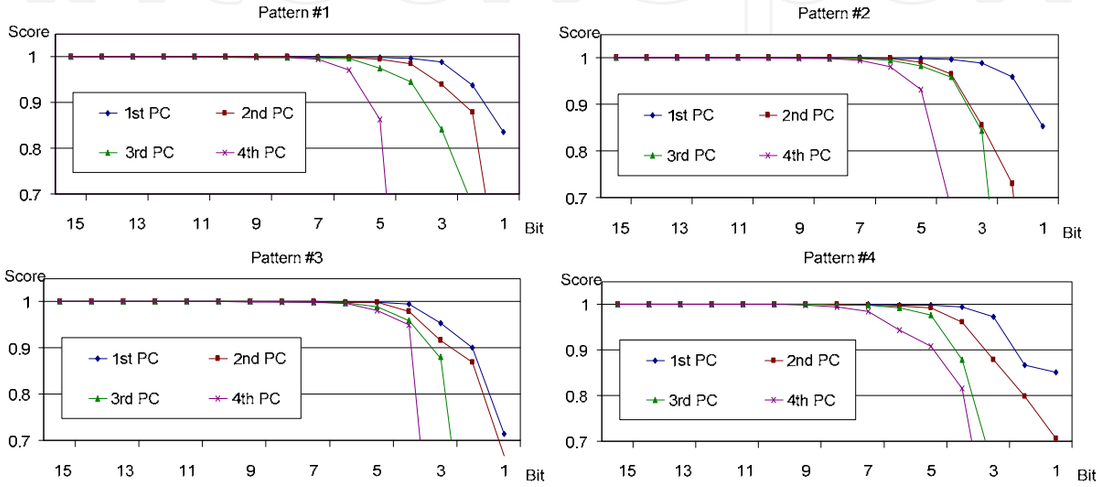

Fig. 7. The comparison between the PCs generated by the software Matlab model using floating point operations and our hardware Verilog model using fixed point operations. We use the correlation parameter as the similarity score. The simulation results show that the hardware with 9-bit precision is the cost-minimized hardware without affecting the accuracy of the output PCs.

4.2 Precision Analysis

As the synthesized results, the larger bit-width leads to the larger chip area. The precision analysis is made here in order to find the cost-minimized hardware without affecting the accuracy of the output PCs. The experimental data and the algorithm setup are the same as those used in Section 2.2. The benchmark is also the standard Matlab "eig" function. The only difference is that we use the hardware Verilog model instead of the software Matlab model to realize the leading eigenvector distilling algorithm. After the spike detection and alignment, the covariance matrix of the detected spike waveforms is calculated and quantized into n-bit precision. This n-bit fixed point covariance matrix is then fed into our hardware Verilog model. The output PCs are also n-bit fixed point number. Note that the iteration number for each PC is set to 20 in this analysis. Fig. 7 shows the comparisons between the PCs generated by the standard Matlab function and our hardware verilog models. We use the correlation function as the similarity score, and the equation is shown as follows:

$$\varphi^{T}_{Verilog}\ \varphi_{Matlab}\ /\ norm(\varphi_{Verilog})\ x\ norm(\varphi_{Matlab})$$

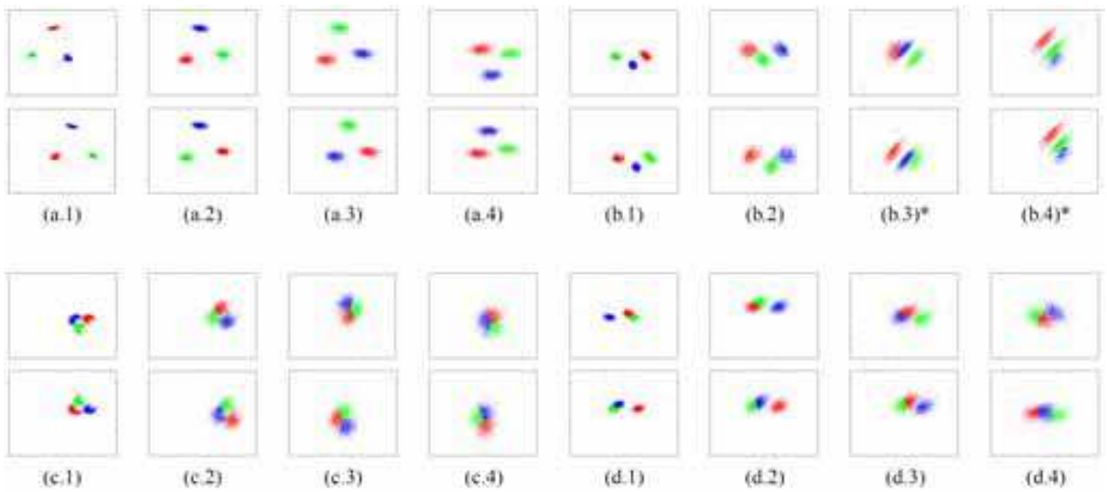

Fig. 8. Subjective comparison of the sorting performance with the PCs generated by our hardware Verilog model and the software Matlab model. The neural sequences #a.1~a.4, #b.1~b.4, #c.1~c.4, and #d.1~d.4 are C_Easy1_noise005~020, C_Easy2_noise005~020, C_Difficult1_noise005~020, C_Difficult2_noise005~020 from Quian Quiroga. The NEO-based spike detection, PCA-based feature extraction, and K-means classification (*watershed-based classification algorithm for b.3 and b.4) algorithms are used. For each neural sequence, the upper figure uses the software Matlab model for the eigenvector generation. 32-bit floating-point number is used to represent the input calculated covariance matrix, and the output PCs. The lower figure uses the hardware verilog model. 9-bit fixed-point precision is used instead. As the results, the PCs generated by the verilog model can achieve almost the same sorting performance compared with the Matlab model. The corresponding objective comparison is shown in Table 5.

Neural Data	a.1	a.2	a.3	a.4	b.1	b.2	b.3	b.4
Matlab (float)	98.24%	98.74%	98.34%	98.02%	98.27%	96.15%	89.02%	84.48%
Verilog (9bit)	98.24%	98.71%	98.39%	97.58%	98.15%	94.85%	90.08%	79.59%
Neural Data	c.1	c.2	c.3	c.4	d.1	d.2	d.3	d.4
Matlab (float)	96.57%	87.47%	78.03%	68.50%	93.37%	83.33%	76.81%	69.10%
Verilog (9bit)	96.87%	85.83%	78.28%	69.01%	93.33%	80.74%	75.61%	63.56%

Average: Matlab(32-bit floating point)/Verilog(9-bit fixed point) = 95.16%/94.45%

Table 5. Objective comparison of the sorting performance between the verilog and matlab models.

The simulation results show that the hardware with 9-bit precision is the cost-minimized hardware without affecting the accuracy of the output PCs.

Combined with the classification algorithm, we also demonstrate the sorting performance with the PCs generated by our hardware Verilog model, and compare it with the software Matlab model in Fig. 8. We adapt the K-means algorithm (Kanungo et al., 2002, Ding & He, 2004), the traditional classification algorithm for spike sorting, to classify most of the neural data on the PCA feature space. For data b.3 and b.4, because the K-means algorithm cannot come out with a reasonable result, another algorithm based on the watershed segmentation algorithm (Wang, 1998) is used. For each neural sequence in Fig. 8, the upper figure indicates the sorting results with the Matlab model while the lower figure is with the

Verilog model. Note that 9-bit precision is used in our final hardware. As the results, the PCs generated by the 9-bit fixed point verilog model can achieve almost the same sorting performance compared with the floating point Matlab model. The objective comparison is shown in Table 5.
In the modified eigenvector distilling algorithm, the normalization process is discarded with the proposed flipped structure. In this case, the output PCs are the orthogonal bases but not the unit vectors. That means the PCs generated by our hardware are the scaled version of the original PCs. However, our adaptive level shifting scheme uses the same bit-width to maximally represent the eigenvectors after the adaptive quantization. These orthogonal but non-orthonormal PCs will thus have similar scaling factors, and lead to almost the same classification results with the K-means algorithm as shown in Fig. 8 and Table 5.

4.3 Fabrication Results

The proposed eigenvector generator is integrated with other processors to complete the PCA-based spike sorting system shown in Fig. 1. As the first prototype chip (Chen et al., 2009), the system is fabricated in .35 μm 2P4M CMOS process for its lower cost. Figure 9 shows the chip micrograph. Table 6 describes the detailed chip specification. The chip size is 28.32 mm^2 with 51.1 k logic gates and 83.5 kb SRAMs. The chip is able to perform NEO-based spike detection and PCA-based feature extraction for 16 recording channels in a realtime. The power consumption is 4.11 mW with 5 volt supply voltage and 3.2MHz/400kHz operation frequency for the GPP/DPs.
Figure 10 demonstrates the functional capability of this chip. The 16-channel neural samples are input to the chip through the NI card device. The PCA training DP calculates the covariance matrix and the corresponding eigenvectors from the detected spikes. After the training, the resultant PCs are stored in the on-chip memory. This training procedure is sequentially performed channel by channel for the 16 recording channels. The embedded GPP is used to control the training and re-training schedule. After the training, the on-line 16-channel spike sorting DP uses these PCs to extract features from the following detected spikes. After the processing, the spike features and the corresponding timing information are output from the chip, recorded by the NI card device, parsed by the computer, and then displayed visually on the screen.

5. Conclusion

In this chapter, the VLSI architecture for leading eigenvector generation was designed for the on-chip PCA-based spike sorting system. The iterative eigenvector distilling algorithm is used because of its simple and regular nature. The proposed flipped structure enables the low area and low power implementation, while the adaptive level shifting scheme optimizes the accuracy and area trade-off. According to the synthesized results with specification of four PCs/channel, 32 samples/spike and 9 bits/sample, the proposed hardware can train 312 channels per minute at 1MHz operation frequency and consumes 132k μm^2 silicon area and 282 μW power in 90 nm process. This eigenvector generation unit is finally fabricated together with other processors in .35 μm process to complete the on-chip 16-channel PCA-based spike sorting system resulting in a 28.32 mm^2 chip area and 4.11 mW power consumption.

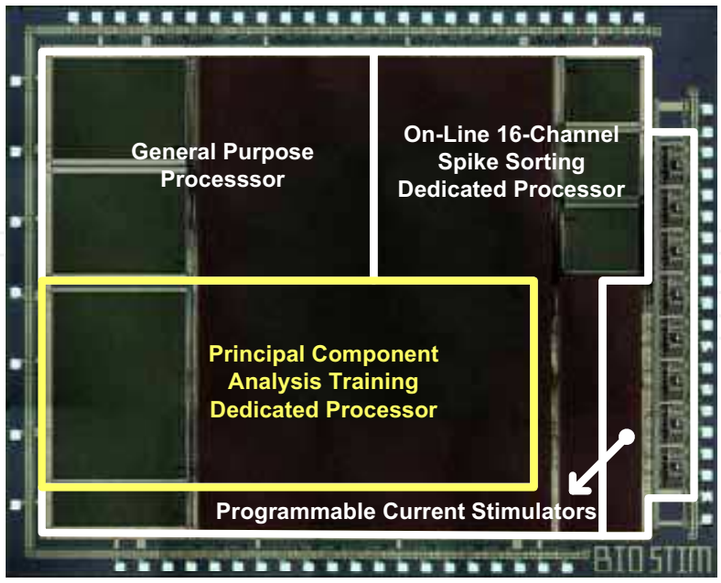

Fig. 9. Chip micrograph of the PCA-based spike sorting system. The proposed eigenvector generator is integrated with other processors to complete the PCA-based spike sorting system shown in Fig. 1. The system is fabricated in .35 μm 2P4M CMOS process.

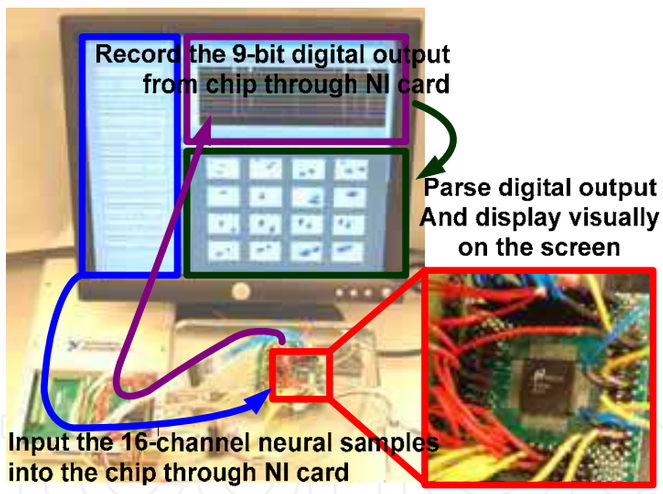

Fig. 10. Chip functional demonstration. The 16-channel neural samples are input to the chip through the NI card device. After the PCA training and the on-line feature extraction, the extracted spike features and the corresponding timing information are output from the chip, recorded by the NI card device, parsed by the computer, and then displayed visually on the screen.

Fabrication Process	0.35 μm 2P4M CMOS
Silicon Area	4.8x5.9 mm²
Logic Gate	51.1k
On-chip Memory	83.5kb
Max. Frequency	40MHz
Max. Voltage	5V
Power Consumption	4.11 mW*

*Power consumption for 16-channel PCA-based spike sorting. 3.2 MHz/400kHz frequencies are used for GPP and DPs respectively.

Table 6. Chip specifications of the PCA-based spike sorting system.

6. References

Zumsteg, Z.; Kemere, C.; Odriscoll, S.; Santhanam, G.; Ahmed, R.; Shenoy, K. & Meng, T. (2005). Power feasibility of implantable digital spike sorting circuits for neural prosthetic systems. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Vol. 13, No. 3, pp. 272–279, 1534-4320.

Oweiss, K. G.; Anderson, D. J. & Papaefthymiou, M. M. (2003). Optimizing signal coding in neural interface system-on-a-chip modules. *Proceedings of 25th Annual Conference IEEE Engineering in Medicine and Biology Society*, Vol. 3, pp. 2216–2219, 0-7803-7789-3.

Olsson, R. H. & Wise, K. D. (2005). A three-dimensional neural recording microsystem with implantable data compression circuitry. *IEEE Journal of Solid-State Circuits*, Vol. 40, No. 12, pp. 2796–2804, 0018-9200.

Harrison, R. R. (2003). A low-power integrated cicuit for adaptive detection of action potentials in noisy signals. *Proceedings of 25th Annual Conference IEEE Engineering in Medicine and Biology Society*, Vol. 4, pp. 3325–3328, 0-7803-7789-3.

Oweiss, K. G.; Anderson, D. J. & Papaefthymiou, M. M. (1977). Multispike train analysis. *Proceedings of the IEEE*, Vol. 65, No. 5, pp. 762–773, 0018-9219.

Letelier, J. C. & Weber, P. P. (2000). Spike sorting based on discrete wavelet transform coefficients. Journal of Neuroscience Methods, Vol. 101, No. 2, pp. 93–106, 0165-0270.

Hulata, E.; Segev, R. & Ben-Jacob, E. (2002), A method for spike sorting and detection based on wavelet packets and Shannon's mutual information. Journal of Neuroscience Methods, Vol. 117, No. 1, pp. 1–12, 0165-0270.

Andra, K.; Chakrabarti, C. & Acharya, T. (2002). A VLSI architecture for liftingbased forward and inverse wavelet transform. *IEEE Transactions on Signal Processing*, Vol. 50, No. 4, pp. 966–977, 1053-587X.

Huang, C. T.; Tseng, P. C. & Chen, L. G. (2004). Flipping structure: an efficient VLSI architecture for lifting-based discrete wavelet transform. *IEEE Transactions on Signal Processing*, Vol. 52, No. 4, pp. 1080–1089, 1053-587X.

Kamboh, A. M.; Raetz, M.; Oweiss, K. G. & Mason, A. (2007). Area-power efficient VLSI implementation of multichannel SWT for data compression in implantable neuroprosthetics. *IEEE Transactions on Biomedical Circuits and Systems*, Vol. 1, No. 2, pp. 128–135, 1932-4545.

Kanungo, T.; Mount, D. M.; Netanyahu, N. S.; Piatko, C. D.; Silverman, R. & Wu, A. Y. (2002). An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 7, pp. 881–892, 0162-8828.

Ding, C. & He, X. (2004). K-means clustering via principal component analysis. *Proceedings of International Conference on Machine Learning,* pp. 225–232, 1-58113-828-5, Banff, Alberta, Canada, July 2004, ACM, New York.

Comaniciu, D. & Meer, P. (2002). Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 5, pp. 603–619, 0162-8828.

Shenoy, K.; Santhanam, G.; Ryu, S.; Afshar, A.; Yu, B.; Gilja, V.; Linderman, M. Kalmar, R.; Cunningham, J.; Kemere, C.; Batista, A.; Churchland, M. & Meng, T. (2006). Increasing the performance of cortically controlled prostheses. *Proceedings 28th Annual Conference IEEE Engineering in Medicine and Biology Society*, pp. 6652–6656, 1-4244-0032-5.

Golub, G. H. & Van Loan, C. F. (1996). *Matrix Computations*. Johns Hopkins University Press, 0-8018-5414-8, Baltimore, USA.

Roweis, S. (1998). Em algorithms for PCA and SPCA. *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems*, pp. 626–632, 0-262-10076-2, Denver, Colorado, United States, 1998, MIT Press, Cambridge, MA, USA.

Schilling, R. J. & Harris, S. L. (2000). *Applied Numerical Methods for Engineers Using Matlab and C*. Brooks/Cole Publishing Company, 0-5343-7014-4, Pacific Grove, CA, USA.

Sirovich, L. (1987). Turbulence and the dynamics of coherent structures. *Quarterly of Applied Mathematics*, Vol. 45, pp. 561–571, 0033-569X.

Sharma, A. & Paliwal, K. K. (2007). Fast principal component analysis using fixed-point algorithm. *Pattern Recognition Letters*, Vol. 28, No. 10, pp. 1151–1155, 0167-8655.

Quian Quiroga, R. Simulated extracellular recordings. http://www2.le.ac.uk/departments/engineering/research/bioengineering/neuroengineering-lab.

Anderson, E.; Bai, Z.; Bischof, C.; Blackford, S.; Demmel, J.; Dongarra, J.; Du Croz, J.; Greenbaum, A.; Hammarling, S.; McKenney, A. & Sorensen, D. (1999). *LAPACK User's Guide*, Society for Industrial and Applied Mathematics, 0-89871-447-8, Philadelphia, USA.

Kim, K. H. & Kim, S. J. (2000). Neural spike sorting under nearly 0-db signal-to-noise ratio using nonlinear energy operator and artificial neural-network classifier. *IEEE Transactions on Biomedical Engineering*, Vol. 47, No. 10, pp. 1406–1411, 0018-9294.

Wang, D. (1998). Unsupervised video segmentation based on watersheds and temporal tracking. *IEEE Transactions on Circuits System on Video Technology*, Vol. 8, No. 7, pp. 539–546, 1051-8215.

Chen, T. C.; Chen, K.; Yang, Z.; Cockerham, K. & Liu, W. (2009) A biomedical multiprocessor soc for closed-loop neuroprosthetic applications. *Proceedings of IEEE International Solid-State Circuits Conference*, Vol. 25, pp. 434–435.

New Developments in Biomedical Engineering
Edited by Domenico Campolo

ISBN 978-953-7619-57-2
Hard cover, 714 pages
Publisher InTech
Published online 01, January, 2010
Published in print edition January, 2010

Biomedical Engineering is a highly interdisciplinary and well established discipline spanning across engineering, medicine and biology. A single definition of Biomedical Engineering is hardly unanimously accepted but it is often easier to identify what activities are included in it. This volume collects works on recent advances in Biomedical Engineering and provides a bird-view on a very broad field, ranging from purely theoretical frameworks to clinical applications and from diagnosis to treatment.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen