Design and Implementation of Cubic Spline Interpolation for Spike Sorting Microsystems

Tung-Chien Chen, Yun-Yu Chen, Tsung-Chuan Ma, and Liang-Gee Chen
Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan, Email: lgchen@cc.ee.ntu.edu.tw

***Abstract*—Accurate spike sorting is important for neuroscientific and neuroprosthetic applications. The sorting of spikes depends on the features extracted from the neural waveforms, and a better sorting performance usually comes with a higher sampling rate (SR). However for long duration experiments on free-moving subjects, the miniaturized and wireless neural recording ICs are the current trend. The compromise on sorting accuracy is usually made for the low power consumption with a lower SR. In this paper, the VLSI architecture of cubic spline interpolation is proposed to improve the power-accuracy tradeoff for the spike sorting microsystems. The window-based interpolation schedule, event-triggered processing, and two-step interpolation scheme are applied to save the memory and computation. 0.04 μW/channel is finally achieved after the implementation in 90nm process.**

I. Introduction

Spike sorting is an important tool to study neural activities and brain functions in neuroscience research [1]. It is also a key component in cortically-controlled neuroprosthetics for spinal cord injured patients [2]. Robust sorting performance is an important issue for these applications [3]. The results of the neural decoding is less significant without an accurate spike sorting. On the other hand, making miniaturized and wireless microsystems for the experiments on free-moving subjects is the current design trend [4], [5]. For these resource-constrained systems, the design issues for low power consumption is usually considered and may result in the compromise on sorting performance.

One of the design issues for the power and accuracy tradeoff is the sampling rate in the neural recorder. Since the classification of spikes depends on the features extracted from the spike waveforms, a better sorting performance usually comes with a higher sampling rate. However the high sampling rate leads to a larger power consumption for the recording, processing, and wireless telemetry circuitries, which may not be feasible for the applications. A sampling rate of 100 k sample per second (sps) is suggested in [6] for an excellent performance. However the current microsystems are usually designed with 20–40 ksps sampling rates [4].

In this paper, the cubic spline interpolation hardware is designed to improve the tradeoff between power and accuracy for spike sorting microsystems. Since most spike energy is under 6.25kHz [6], after the waveform reconstruction through the interpolation, the sorting performance with 100ksps signal resolution could be achieved even after a low sampling rate and low power consumption for the neural recorder. The remainder of this paper is organized as follows. The preliminary information about the spike sorting microsystem along with the cubic spile interpolation is introduced in Section II.

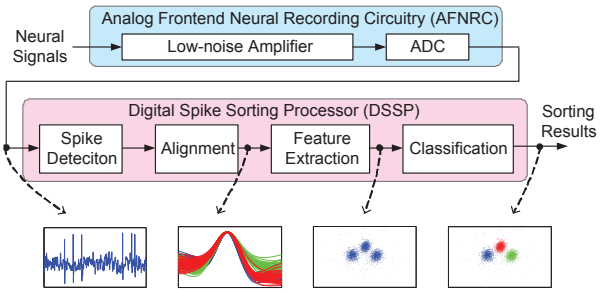

Fig. 1. The hardware operation of the neural recording and spike sorting.

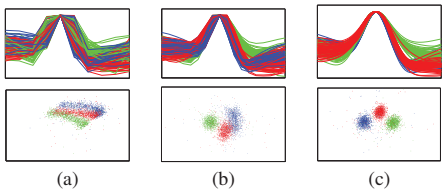

Fig. 2. The improvement of neuron cluster separation after the interpolation. The original neural signals are sampled at 12.5 ksps as shown in (a). For (b) and (c), the spike waveforms are re-aligned after the up-sampling to 25 ksps and 100 ksps with the cubic spline interpolation.

Section III describes the proposed VLSI architecture of cubic spline interpolation, and Section IV shows the implementation results. Finally, Section V concludes this work.

II. Spike Sorting and Cubic Spline Interpolation

A. Spike Sorting, Sampling Skew, and Interpolation

During the extra-cellular neural recording, an electrode usually records the signals from multiple surrounded neurons. Spike sorting is a kind of signal processing tool to differentiate which spike corresponds to which of these close-by neurons from the waveform. Figure 1 shows the microsystem for the neural recording and the spike sorting hardware. For the spike sorting, the extracted features of spikes are correlated to the shapes of the spike waveforms. As a result any waveform distortion may have great influence on the sorting performance. Sampling skew is one of the main issues resulting the waveform distortion. During the neural recording, the firing of the action potentials can hardly be synchronized with the sampling of the neural signals. Different time skews corresponding to the neural firing time are sampled for different spikes, and result in the variation of the spike waveforms as well as the degradation of the sorting performance.

The spikes have the most energy under 6.25 kHz. According to the Nyquist-Shannon sampling theory, it should be feasible to reconstruct the 100 ksps spike waveforms through the

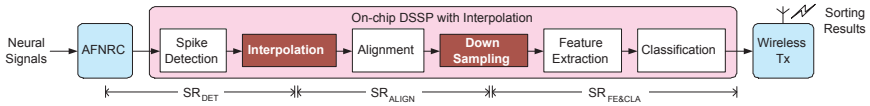

Fig. 3. The proposed on-chip spike sorting system with interpolation. SR_{DET}, SR_{ALIGN}, and $SR_{FE\&CLA}$ indicate the sampling rates of the processed data in spike detection, alignment, and feature extraction along with classification stages.

interpolation if the sampling rate in the AFNRC is higher than 12.5 ksps. An uncompromised spike sorting performance may thus be achieved even with a low sampling rate. Figure 2 shows the improvement of the neuron separation by means of the interpolation. The neural signals are originally sampled at 12.5 ksps. Then the spike waveforms are interpolated to 25 ksps and 100 ksps in Fig 2 (b) and (c). After the interpolation, the waveform can be re-aligned with less error caused by the sampling skew. This improves the separation of neuron clusters on the feature space and leads to a better sorting performance.

B. Proposed On-chip Spike Sorter with Interpolation [7]

The cubic spline interpolation is used in the proposed spike sorting microsystems to improve the power-accuracy tradeoff. Since the spike sorting is performed after the reconstruction of the high-resolution neural signals, two kinds of improvement can be expected. With a fixed sampling rate and power consumption for AFNRC, our digital system achieves a higher sorting accuracy after the interpolation. With an expected sorting accuracy, the AFNRC may consume lower power consumption with a lower sampling rate because the compensation can be made by the interpolation. As for the total power consumption, although the usage of the interpolation may increase the power of DSSP, it would release the requirement of the high sampling rate of AFNRC in some respects. The power consumed by the AFNRC chips [4] is about an order larger than the state-of-the-art DSSP designs [5], this power tradeoff between the AFNRC and DSSP would finally result in a smaller total power.

The DSSP may consume larger power after the interpolation. This penalty can be minimized if the high signal resolution is only utilized at the critical step of the spike sorting. Figure 3 shows the proposed system. The DSSP part is divided into three sections with the specific purposes. First the spike detection usually uses the energy detector and does not need detailed waveform information. Therefore the SR_{DET} along with the SR_{AFNRC} should be set as low as possible in order to save power. Afterwards, the interpolation is performed and the detected spikes are aligned with a higher SR_{ALIGN} in order to reduce the sampling skew and improve the ability of neuron separation. After the alignment, the feature extraction and classification, the most computationally intensive parts of spike sorting, are operated after the down-sampling. Since the sampling skew is minimized during the high-resolution alignment, there should be limited waveform distortion after the down-sampling and the sorting performance is kept with lower power consumption. For the detailed accuracy analysis about the spike sorting for the proposed system, please refer to [7].

C. Algorithm Review of Cubic Spline Interpolation

In this subsection, we will review the algorithm of cubic spline interpolation before the hardware implementation. Consider a collection of n data samples, y_1, y_2, ... and y_n, to perform the up-sampling with the cubic spline interpolation, n-1 piecewise third degree polynomial functions are constructed between the neighboring pairs of data points, or segments. The polynomial curve fitting for each segment can be represented by

$$Y_i(t) = a_i + b_i t + c_i t^2 + d_i t^3 \tag{1}$$

where $t \in [0,1]$ and $i = 1,2,...,n-1$. The cubic spline constrains the function value, first derivative and second derivative. The routine must ensure that Y_i, Y_i' and Y_i'' are equal at the interior nodes for the adjacent segments. The criteria are summarized as follows.

$$\begin{array}{l} Y_i(0) = y_i = Y_{i-1}(1) \\ Y_i'(0) = Y_{i-1}'(1) \\ Y_i''(0) = Y_{i-1}''(1) \end{array} \tag{2}$$

To solve the coefficients of the polynomials, a symmetric tridiagonal matrix can be derived by rearranging all these equations of the criteria:

$$\begin{bmatrix} 2 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 4 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 2 \end{bmatrix} \begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_n \end{bmatrix} = \begin{bmatrix} 3(y_2 - y_1) \\ 3(y_3 - y_1) \\ \vdots \\ 3(y_n - y_{n-1}) \end{bmatrix} \tag{3}$$

Note that D_i indicates the function's first derivative, or $Y_i'(0)$. The natural spline boundary condition is used here, and $Y_1''(0)$ and $Y_n''(1)$ is set to zeros. D_i can be solved by the multiplication of the inverse of the square matrix in eq. 3. Then the piecewise polynomial functions can be derived by substituting the data points and first derivatives into the following equations.

$$\begin{array}{l} a_i = y_i \\ b_i = D_i \\ c_i = 3(y_{i+1} - y_i) - 2D_i - D_{i+1} \\ d_i = 2(y_{i+1} - y_i) + D_i + D_{i+1} \end{array} \tag{4}$$

Finally the interpolated samples in a segment can be calculated by substituting the corresponding numbers of t into the eq. 1. For example, to up-sample the data by a factor of four, $1/4$, $1/2$, and $3/4$ are used. For detailed mathematical derivation from eq. 1 and eq. 2 to eq. 3 and eq. 4, please refer to [8].

III. VLSI Architecture Design

A. Window-based Interpolation Schedule

Low power consumption and miniaturized area are two primary issues for spike sorting microsystems. Unlike the processing in the software system, it is not feasible to store a large amount of neural data and perform cubic spline interpolation afterwards. A large memory will be required consuming significant power and area. The time delay to

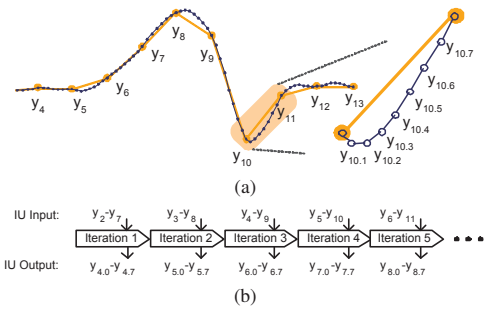

Fig. 4. The proposed window-based interpolation schedule.

collect the neural samples is not allowed under the on-the-fly processing requirement.

In order to minimize the memory requirement and achieve the on-the-fly processing, the hardware is designed to perform the interpolation window by window for the neural signals with a repeated and regular schedule. The original signals are pushed into the hardware sample by sample, and the first-in first-out (FIFO) registers would be used to store a window of the most recent neural samples. The cubic spline procedure described in Sec. II-C is repeatedly performed for the data window every time when a new original sample is input. For each iteration, the matrix operation of eq. 3 is first solved, and the coefficients of the third order polynomials are calculated as eq. 4. Since only a relatively small data window is involved for one iteration, the interpolated samples are more sensitive to the boundary conditions at two ends of the window. Therefore, only the polynomial in the middle segment of the window is adopted to generate the interpolated samples in each iteration. The entire waveform can be interpolated by repeating this procedure for the consecutive and overlapped windows of neural signals.

Under this schedule, there is a trade-off between the interpolation accuracy and the memory size regarding the length of the data window. For each iteration, if more original samples are involved for the interpolation, the polynomial in the most middle segment of the window is less influenced by the boundary conditions. The larger memory is the penalty in this case. For the optimization, according to the simulation, the window with six samples has the minimized memory size without a noticeable error propagated from the two boundaries.

Figure 4 illustrates the proposed window-based schedule. The light orange line (y_4, y_5, y_6, ...) represents the original neural samples while the deep blue line ($y_{10.1}$, $y_{10.2}$, $y_{10.3}$, ...) represents the interpolated neural samples. One window involves six original samples, and we would like to up-sample the signals by eight (i.e. from 12.5 ksps to 100 ksps) in this example. As shown in Fig. 4 (b), in the first iteration, the original samples of y_2 to y_7 are used. The interpolated samples of $y_{4.0}$ to $y_{4.7}$ are output. In the next iteration, y_8 is pushed into the FIFO, and y_5 to y_8 are used to generate $y_{5.0}$ to $y_{5.7}$.

B. Event-triggered Interpolation Scheme

Since the neurons do not keep firing the action potentials all the time, the event-triggered processing is proposed to reduce the computation complexity and the corresponding power consumption. Here the event is defined as the coming

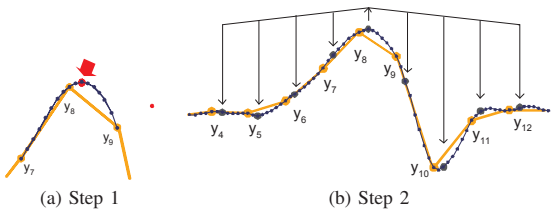

Fig. 5. The proposed two-step interpolation scheme with down-sampling.

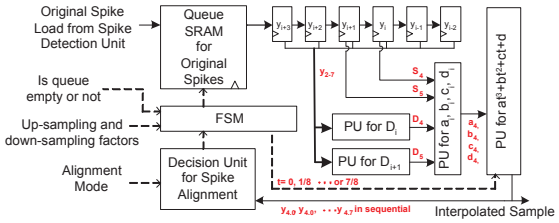

Fig. 6. The proposed architecture performing the cubic spline interpolation, alignment, and down-sampling operations for the system shown in Fig. 3.

of the spike. In the proposed spike sorting microsystem, the interpolation is performed after the spike detection. Since the signal characteristics are extracted only for the neural events, the interpolation engine can be turned on after the detection of the spikes. Compared to the always-on interpolation, this scheme can save morn than 95% computation complexity if the firing rate of the neurons is about 20 spikes per second.

C. Two-stage Interpolation with Downsampling

In the proposed system, down-sampling is performed after the interpolation and alignment. Some of the interpolated samples are not used for the subsequent processing tasks and will be discarded. Based on this observation, the two-stage interpolation is proposed to further reduce the computation and the corresponding power consumption. The idea is to firstly perform the interpolation locally only for the alignment regions in order to find the new alignment point. After the re-alignment, referred to the new alignment point, only the corresponding samples that will be used after the down-sampling are interpolated for the entire spike waveform.

Figure 5 shows an example of the two-step interpolation. In this example, the system does interpolation for the raw signals by the factor of eight, and then performs the down-sampling by the factor of eight as well after the re-alignment according to the peak of the spike. In the first step as shown in Fig. 5 (a), only two segments beside the peak, the original alignment point, of the spike are interpolated. The new peak as the new alignment point is refined with a higher sampling rate. In the second step, since the new alignment point is found, and we can know the exact locations of the samples that will be required after the down-sampling. Therefore, the hardware performs the interpolation only for those useful samples as shown in Fig. 5 (b). In this example, only 33% of samples are finally interpolated compared to the traditional case that performs the interpolation for the whole spike waveform.

D. Final Architecture

Figure 6 shows the architecture performing the interpolation, alignment, and down-sampling operations for the system

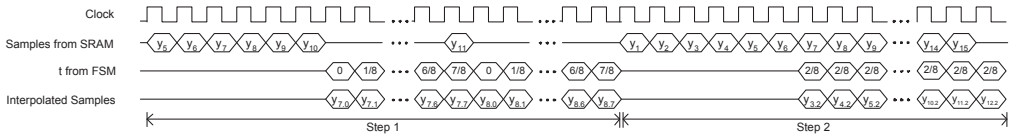

Fig. 7. The cyclic schedule of the proposed architecture for the example shown in Fig. 5.

shown in Fig. 3. This architecture can support the proposed window-based interpolation schedule as well as the event-triggered and two-step interpolation schemes. After the spike detection, the detected spikes of multiple channels are fed into the "Queue SRAM for Original Spikes." When the queue is empty, the FSM keeps waiting for the next event, and the processing units (PUs) are turned off by a gated clock. If there are spikes in the queue, the hardware performs the interpolation for them in a FIFO fashion. The outputs are the spike waveforms aligned and down-sampled according to the hardware configuration. Several configurations are designed and can be programmed by the users. The spikes are allowed to be aligned according to their peaks, slopes, or purely points crossing the threshold value defined for the spike detection. The up-sampling and down-sampling factors can be set to two, four, or eight separately.

The PUs and the FIFO register array in Fig. 6 are designed to support the window-based cubic spline interpolation. Six-pipelined FIFO registers are used to store a window of six original samples, y_{i+3}, y_{i+2}, ... and y_{i-2}, loaded from the queue SRAM. The curve between y_i and y_{i+1} is the targeted segment to be interpolated. As for the processing, the first-stage PUs calculate D_i and D_{i+1} by solving the eq. 3. Afterwards, the second-stage PU derives the coefficients of a_i, b_i, c_i, and d_i according to eq. 4. Then, the last-stage PU generates the interpolated samples by substituting the corresponding values of t into eq. 1. The red words in the Fig. 6 denote the example of the first iteration in Fig. 4 (b). The inputs are y_2 to y_7 while the outputs are $y_{4.0}$ to $y_{4.7}$ if t is set to 0, $1/8$, $2/8$, ... and $7/8$ in a sequential order.

The FSM controls the SRAM and PUs with the two-step interpolation scheme. The cyclic schedule is shown in Fig. 7. This example is for the case shown in Fig. 5. The first step refines the alignment point for spikes in a higher sampling rate. After the six cycles to load the window of original samples, the FSM generates all possible values for "t," and the interpolation is thoroughly performed for two segments centered by the original alignment point. The decision unit checks the new alignment point on-line after the interpolation PUs and forwards the result to the FSM. In the second step, only the samples required after the down-sampling are interpolated for the entire spike. The FSM generates the corresponding "t" value according to the new alignment point to produce the samples required after the down-sampling.

IV. Implementation Results

The proposed architecture is implemented in verilog and synthesized with UMC 90 nm 1P9M low-leakage CMOS process. Table I summarizes the implementation results. The hardware requires 0.23 mm^2 area including 27.7k logic gates and a 11.5kb SRAM. The SRAM is able to queue 32 spikes with 45 samples/spike in maximum and 8 bits/sample. The hardware can handle about 2600 spikes per second in a realtime with 1MHz operation frequency under the worst case of 45 samples per spike and upsampling by a factor of 8 without any down-sampling. This specification should be able to perform on-line processing for 128 channels with the spike firing rate of 20 spikes/channel. The corresponding power consumption is 5.6 μW in the worst case. The average power consumption for each channel is 0.04 μW. This overhead is almost negligible compared to the saving in AFNRC and other parts of DSSP utilizing the traditional processing procedure without the interpolation.

TABLE I
Implementation Results in 90 nm CMOS Process

Process	90 nm 1P9M CMOS
Supply Voltage	1.0 Volt
Maximum Operation Frequency	20 MHz
Core Area	0.16 mm^2
Power Consumption	5.60 μW (@1MHz,1Volt)
Computation Capability	Realtime processing for 128 channels, $20\frac{spikes}{channel\times sec}$, and $45\frac{samples}{spike}$ (@1MHz)

V. Conclusion

In this paper, the on-chip spike sorting systems with cubic spline interpolation is proposed to strike the tradeoff between the sorting accuracy and power consumption. In the VLSI hardware for on-line cubic spline interpolation, the window-based processing schedule is designed to save the memory while the event-triggered and two-step interpolation schemes are proposed for the computation reduction. 0.04 μW/channel power is finally required after the implementation in 90nm CMOS process.

References

[1] M. Velliste and et al., "Cortical control of a prosthetic arm for self-feeding," *Nature*, vol. 453, pp. 1098–1101, 2008.
[2] Z. Zumsteg and et al., "Power feasibility of implantable digital spike sorting circuits for neural prosthetic systems," *IEEE Trans. on Neural Syst. and Rehabilitation Eng.*, vol. 13, no. 3, pp. 272–279, 2005.
[3] M.D. Linderman and et al., "Signal processing challenges for neural prostheses," *IEEE Signal Process. Mag.*, vol. 25, no. 1, pp. 18–28, 2008.
[4] R. R. Harrison and et al., "A low-power integrated circuit for a wireless 100-electrode neural recording system," *IEEE J. Solid State Circuits*, vol. 42, no. 1, pp. 123–133, 2007.
[5] V. Karkare and et al., "A 130-gw, 64-channel spike-sorting dsp chip," in *Proc. of IEEE Asian Solid-State Circuits Conference*, 2009, pp. 289–292.
[6] T. J. Blanche and N. V. Swindale, "Nyquist interpolation improves neuron yield in multiunit recordings," *J. of Neurosci. Methods*, vol. 155, no. 1, pp. 81–91, 2006.
[7] Y.-Y. Chen and et al., "Accuracy and power tradeoff in spike sorting microsystems with cubic spline interpolation," in *Proc. of IEEE Int. Symp. on Circuits and Syst.*, 2010, pp. 1508–1511.
[8] R. H. Bartels and et al., *An Introduction to Splines for Use in Computer Graphics and Geometric Modelling*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.