Journal of Neuroscience Methods, 30 (1989) 59–69
Elsevier

NSM 00993

A low-cost single-board solution for real-time, unsupervised waveform classification of multineuron recordings

Andreas K. Kreiter [1,*], Ad M.H.J. Aertsen [1] and George L. Gerstein [2]

[1] *Max-Planck-Institute for Biological Cybernetics, Tübingen (F.R.G.) and* [2] *Department of Physiology, University of Pennsylvania, Philadelphia, PA 19104-6085 (U.S.A.)*

(Received 28 February 1989)
(Accepted 9 May 1989)

Key words: Multi-unit spike train; Waveform analysis; Spike discrimination by computer; Single-board implementation; Reduced template matching

We describe a low-cost single-board system for unsupervised, real-time spike sorting of recordings from a number of neurons on a single microelectrode. The maximum number of spike classes depends on the quality of the recording; it will typically be between 2 and 5. The spike sorter communicates with a conventional microcomputer through a standard serial port (RS232). For typical firing rates as measured in the mammalian central nervous system, this set-up will accommodate up to some 10 parallel spike sorters for as many separate microelectrodes.

Introduction

Over the last decade, a gradually increasing number of laboratories have entered the field of electrophysiological multineuron studies. One of the factors contributing to this increase was the development of the recording technology necessary for simultaneous observation of the activity from a number of separate neurons. In this development we may distinguish two main components: the first is concerned with the improvement of multielectrode arrangements, the second with the optimization of signal processing techniques, aimed at utilizing differences in spike waveforms to separate the contributions from different neurons recorded on a single microelectrode (for reviews see Gerstein et al., 1983; Krüger, 1983; and Schmidt, 1984a,b). In the present paper we will concentrate on spike sorting by waveform analysis —an approach specifically aimed at studying interactions among closely neighboring neurons.

Spike sorting algorithms of different kinds have been available for some time, both in hardware and in software, both for on-line and for off-line analysis. Unfortunately, real-time versions of such 'sorting machines' were up to now invariably associated with considerable costs, prohibiting their widespread usage, especially in combination with cost-multiplying multielectrode set-ups. Furthermore, most algorithms require much operator intervention and tuning; this is particularly inappropriate in the context of multielectrode multineuron experiments where there is already more than enough to occupy the experimenter. In the following we describe a low-cost single-board system for unsupervised, real-time sorting of recordings from a number of neurons on a single microelectrode [**]. This spike sorter combines the ad-

[*] Present address: Max-Planck-Institute for Brain Research, Deutschordenstrasse 46, D-6000 Frankfurt a.M. 71, F.R.G.
Correspondence: A. Aertsen, Max-Planck-Institute for Biological Cybernetics, Spemannstrasse 38, D-7400 Tübingen, F.R.G.

[**] Negotiations are currently underway to make this spike sorter commercially available; no additional information is available pending these negotiations.

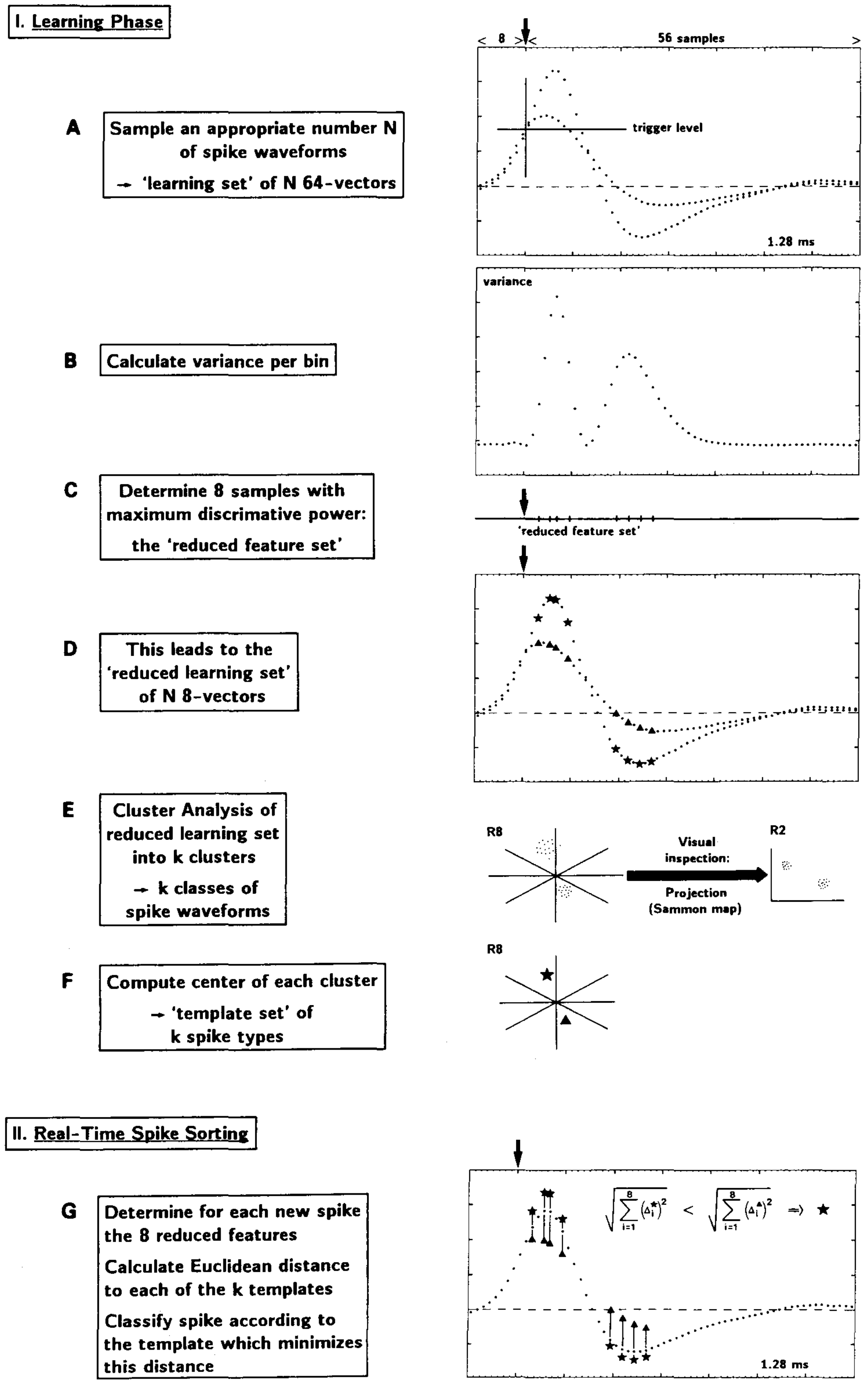
I. Learning Phase
A
Sample an appropriate number N of spike waveforms
→ 'learning set' of N 64-vectors
< 8 >< 56 samples >
trigger level
1.28 ms
B
Calculate variance per bin
variance
C
Determine 8 samples with maximum discrimative power: the 'reduced feature set'
'reduced feature set'
D
This leads to the 'reduced learning set' of N 8-vectors
E
Cluster Analysis of reduced learning set into k clusters
→ k classes of spike waveforms
R8
Visual inspection:
Projection (Sammon map)
R2
F
Compute center of each cluster
→ 'template set' of k spike types
R8
II. Real-Time Spike Sorting
G
Determine for each new spike the 8 reduced features
Calculate Euclidean distance to each of the k templates
Classify spike according to the template which minimizes this distance
$\sqrt{\sum_{i=1}^{8}(\Delta_i^{\star})^2} < \sqrt{\sum_{i=1}^{8}(\Delta_i^{\blacktriangle})^2} \Rightarrow \star$
1.28 ms

Fig. 1. Schematic description of algorithm for unsupervised, multineuron spike waveform classification.

vantages of low-cost single-board microprocessors and widespread personal computer (pc)-technology. The number of spike waveforms that may be separated depends on the extent of the differences between the waveform prototypes and on the variability within each spike class; a typical recording would yield between 2 and 5 separable neurons.

The spike sorter is based on an algorithm, recently developed by Gerstein and coworkers (Salganicoff et al., 1988; Sarna et al., 1988); its principles of operation are briefly reviewed. The spike sorter communicates with a conventional microcomputer through a standard RS232-port. For typical firing rates as measured in the central nervous system, this set-up will accommodate up to some 10 parallel spike sorters for as many separate microelectrodes.

Principle of Operation

The spike sorter algorithm can be separated into two phases: the *learning phase* (I) and the actual *sorting phase* (II); the flow of the algorithm is schematically illustrated in Fig. 1.

In the *learning phase*, the system acquires the information necessary to distinguish between the different types of spike waveforms. First, a certain number of spikes, say 100 (the 'learning set') are sampled over 1.28 ms at 50 kHz, with the selection of the samples being determined by an external trigger on the detection of a spike (e.g. by a positive-going level crossing). As a result, each waveform is described by a 64-dimensional vector: an ordered set of 64 signal values at 20 μs spacing, with the initial part of each set (in our implementation the first 8 samples) being taken from the time before the trigger occurred in order to include the ascending phase of the spike waveform (Fig. 1A). The next step is to compute separately for each of the 64 points along the time axis, the variance over the learning set: the resulting set of variances describes the time course of variance for the waveforms collected so far (Fig. 1B). In a third step, an iterative procedure selects from the original 64 sample positions those 8 positions which show the largest variance and which thus should provide highest discriminability between the different types of spike waveforms; we call this the 'reduced feature set' (Fig. 1C). In order to prevent inappropriate clustering of these 'high-discriminability points', the variance of each point is weighted in the selection process by a function (in the present implementation the square root) of the distances to all previously selected points.

The sample set for each spike waveform is thus reduced to an 8-dimensional vector (Fig. 1D); the reduced learning set of these 8 vectors is then analyzed for possible clustering (Fig. 1E) using the k-means clustering algorithm (Hartigan, 1975). The user decides how many clusters are appropriate. This is done on the basis of the f-ratio: the ratio of total partition error for k versus $k+1$ clusters; these ratios are computed for ascending k, starting with $k = 1$. Usually the f-ratio decreases drastically if k approaches the correct number of clusters; in relatively noise-free data the f-ratio increases again when k is raised further. Once the correct k has been determined, the center of mass of each cluster is calculated (Fig. 1F); each of the k cluster centers consists of 8 components and forms a template which is used for the real-time classification.

At the end of the learning phase we have thus defined the two aspects of data required for the sorting of new spikes: (1) the points in the time course of the spike waveform which provide highest discriminability (*reduced feature set*); and (2) the typical values at these points for each class of spikes present in the learning set (*template set*).

Phase II finally is the *real-time sorting phase*. Upon a trigger signal indicating the arrival of a spike, we collect those 8 points on the spike waveform which correspond to the previously determined reduced feature set. The resulting 8-vector set then is classified in real-time according to the Euclidean distance to each of the k previously defined templates. The template which minimizes this distance is assigned to that particular spike waveform (Fig. 1G); in order to prevent classification of 'outlying' spikes, it is possible to set a threshold.

Implementation

For the actual implementation, it is important to note that different components of the algorithm require different speeds and priorities of execution. For instance, the acquisition of appropriate data from each incoming spike waveform and the assessment of its time of arrival should have utmost priority and guaranteed top speed. Other activities such as the classification itself do not need such high priority and still others, like the clustering algorithm, are needed only occasionally and cannot be executed in real-time with usual laboratory computers anyway.

For these reasons we decided to distribute the various tasks over different processors. Timing, sampling of spike waveforms, determination of the reduced features and real-time classification are done by a low-cost single-board system. Communication with the user, computation of clusters and cluster centers, computation of a threshold and storing of the classified data are taken care of by a host computer with mass storage (disk). Here a further advantage of a distributed system becomes apparent. Even a modern CPU is completely absorbed when continuously handling a 50 kHz data rate, and simultaneous reliable storage of data to disk would be difficult: many systems do not allow interrupts while writing to disk or they are not able to handle high interrupt rates. On the other hand, most conventional microcomputers are able to handle much more data than that delivered by 4 or 5 sorted spike trains, provided they do not have to take care of the timing as well. For this reason it is possible to have a number of spike sorters communicate in parallel with the host, and thus to service a number of microelectrodes simultaneously.

Single-board system

The single-board spike sorter system is composed of a single-board computer and additional hardware; its configuration is shown in Fig. 2. The single-board computer in our implementation contains a MC68000 CPU (10 MHz), 128 kbyte of RAM, 64 kbyte EPROM, a MC68681 which provides two RS232C interfaces, a MC68230 peripheral interface/timer and an 8-bit D/A-converter (ZN428E). The additional hardware consists of an amplifier with adjustable offset and gain (0–100 ×), a 12-bit A/D-converter (AD7572JN-12), the timing and interrupt logic, an analog delay line (RD5106A) and various devices for protection of input and output lines. The whole system, together with the necessary plugs and user controls on the front plate is mounted in a cassette which conforms to double Eurocard format; the complete system thus can be housed conveniently in a commercially available chassis with power supply. Software for interrupt handling, acquisition of learning set, calculation of reduced feature set and reduced learning set and, finally, for real-time sorting was written in assembler.

Host computer

As a host we used an AT compatible, equipped with 8 RS232-interfaces, Hercules-compatible graphics card and 30 Mbyte harddisk. Software is written in TURBO PASCAL (vs 4.0) and is able to handle a user-specified number (in our present configuration up to 8) of spike sorter systems simultaneously. Commands are available for handling file-I/O, starting learning phase, starting real-time sorting, resetting processes on the boards, transmitting thresholds, changing board numbers, data display, etc. Commands can be addressed to each spike sorter individually or to all sorters simultaneously.

For purposes of quality control of the sorting, the clustering of the learning set can be visualized by means of a mapping algorithm, projecting multidimensional data to two-dimensional space (Sammon, 1969). During data acquisition, the host receives sorted event data from the single-board system(s). Events are shown in the form of separate dot displays, and are stored in disk files. In addition, data from each learning set (reduced feature set, template set) can be stored in a separate file for later inspection. Host computer software was extended in a straightforward way to incorporate conventional methods of (quasi) on-line single- and multi-unit analysis (e.g. PSTH, auto- and cross-correlation).

Interrupt levels

The single board's CPU MC68000 provides 7

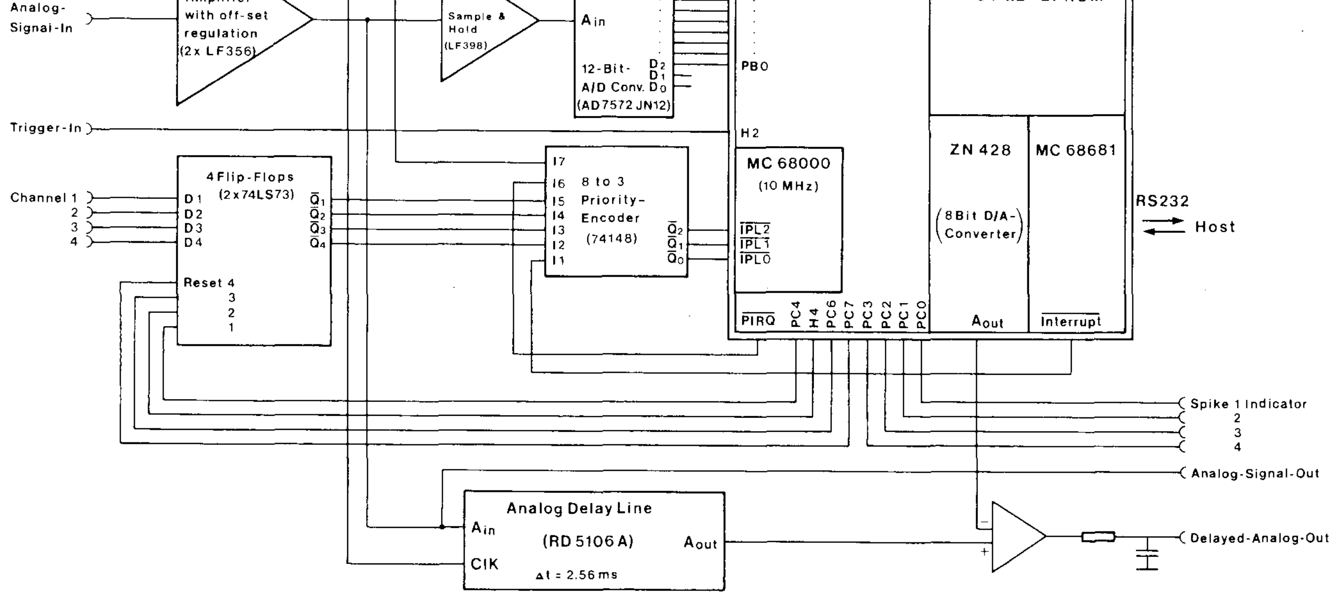

Fig. 2. Schematic circuit diagram of the single-board spike sorter system. The diagram shows the most important parts of the circuitry, but omits details like protection and buffer circuitry for input and output lines. The thick frame delineates the standard microprocessor circuitry from the additional electronics. The frequency generator in the upper left of the diagram provides the 200 kHz clock for the analog delay line, as well as an internal source for the 50 kHz sampling clock. One switch selects either the internal or an external clock source (Clock-In); another one selects which of the two provides the Clock-Out signal. At each positive-going transition of the clock signal, the timing logic generates a wide pulse and a narrow pulse. The wide pulse activates the highest level (level 7) interrupt via the interrupt priority encoder. At the same time the narrow pulse initiates both a new conversion cycle of the A/D converter (ADC) and the output of the last conversion result to ports A and B of the MC68230; furthermore, it sets, via an AND gate, the 'Sample and Hold' circuit to hold mode. The ADC sets its Busy-output low, which initiates reading of the output data by the MC68230, and, via the AND gate, causes persistence of the hold mode of the 'Sample and Hold' circuit until conversion is done after about 12 μs. The analog input signal is amplified and corrected for offset by standard operation amplifier circuitry. The signal is then carried to the 'Sample and Hold' circuit, the 'Analog-Signal-Out' plug, and the analog delay line. An analog subtraction circuit subtracts the output of the D/A converter from the delayed signal to provide the final output to the 'Delayed-Analog-Out' plug. The 'Trigger-In'-signal, indicating the occurrence of a spike, activates the H2 input of the MC68230 to initiate an interrupt request via the MC68230's PIRQ output and the level 6 input to the interrupt priority encoder. Inputs on channels 1, 2, 3, or 4 set the corresponding Flip-Flop to cause a level 5, 4, 3, or 2 interrupt request; resetting is done by the appropriate interrupt service routine via the H4 and 3 port C lines used as outputs. Four other port C lines are used to signal the identification of a spike waveform.

levels of interrupt with different priority. The highest, non-maskable interrupt (level 7) is activated by the 50 KHz clock; switches select between internal and external clock input. Independently, it is possible to connect one of the two sources to the clock output. These features provide maximum flexibility in creating synchronized configurations of multiple spike sorter systems. Upon a level 7 interrupt, the CPU increments the 32-bit time counter by 2 and loads a new 10-bit A/D value from the MC68230 into the next 2 bytes of a 64 kbyte ringbuffer (the procedure of incrementing the time counter in steps of 2 enables us to use this time counter also as address pointer for the 64 kbyte ringbuffer, with the address being determined on the basis of the least significant 16 bits of the time counter); this A/D value was latched from the A/D converter into the MC68230 by the interrupt signal about 6 μs earlier. The same signal instructs the 'sample and hold' circuit to freeze the momentary signal value and initiates a new conversion cycle in the A/D converter (cf. Fig. 2). The level 6 interrupt is activated by the trigger signaling the detection of a spike; time of occurrence and position of the waveform samples in the above-mentioned ringbuffer are stored into a second ringbuffer. The level 2, 3, 4 and 5 interrupts can be activated by low-to-high transitions at 4 additional digital input channels; the CPU then writes a specific 6-byte datum (containing channel number and time of occurrence) to the output stream, the third ringbuffer in the system. These additional 4 inputs can be used to signal other events from the experiment, e.g. a stimulus trigger or an animal's behavioral action. A level 1 interrupt is activated when the RS232 interface receives a command byte from the host; the CPU enters the command interpretation routine and checks whether the received byte is a valid command. If not, the ascii-character 'nak' is sent to the host; on a valid command the CPU branches to the appropriate command routine. This routine, if necessary, first waits for additional information, then sends an 'ack'-character to inform the host that the command (and possibly necessary data) have been received and that the command will be executed. Some commands, e.g. the command to perform real-time classification, allow level 1 interrupts; most others, which have a rather short execution time, do not. With no current interrupt, the system is at level 0. Possible commands (level 1 interrupts) are: send an 'ack' character, read board number, acquire learning set with a user-specified number of spikes and send the reduced learning set (together with the reduced feature set for diagnostic purposes) to the host, reset the time counter, read a new template set, read distance threshold, start real-time spike sorting, and stop current process.

Real-time sorting

The flow of the actual real-time spike sorting routine is illustrated in Fig. 3. When sorting is started, all interrupt levels are enabled. The system enters a loop, at the start of which the CPU first searches the third ringbuffer for data to be sent to the host. If data are available and the MC68681 is ready to send, one byte is transmitted. Then, the second ringbuffer is tested for its contents. When a spike marker is found, the CPU tests whether all corresponding data are available in the first ringbuffer. If not, the CPU returns to the start of the loop; if all necessary data are available, the CPU identifies the spike as described above. If the euclidean distance to the nearest cluster center is larger than the threshold, the spike is assigned the number 0 ('not classified'); if not, it is assigned the number of the nearest template. This spike id, together with the time of occurrence, is written to the third ringbuffer. In addition, for spike classes 1–4, a short pulse is set at the corresponding digital output line; furthermore the D/A converter is set to 0.5 V times the spike id and this voltage is subtracted from the output of the 2.56 ms delay line. The resulting analog output can be used to display the delayed spike waveforms on a storage oscilloscope, arranged according to their classification; this feature allows for continuous quality control of the sorting process.

Data format

Data for an event transmitted by the spike sorter to the host consist of 6 bytes in the following format: byte 1 contains the board number; byte 2 contains either in its lower 4 bits the spike

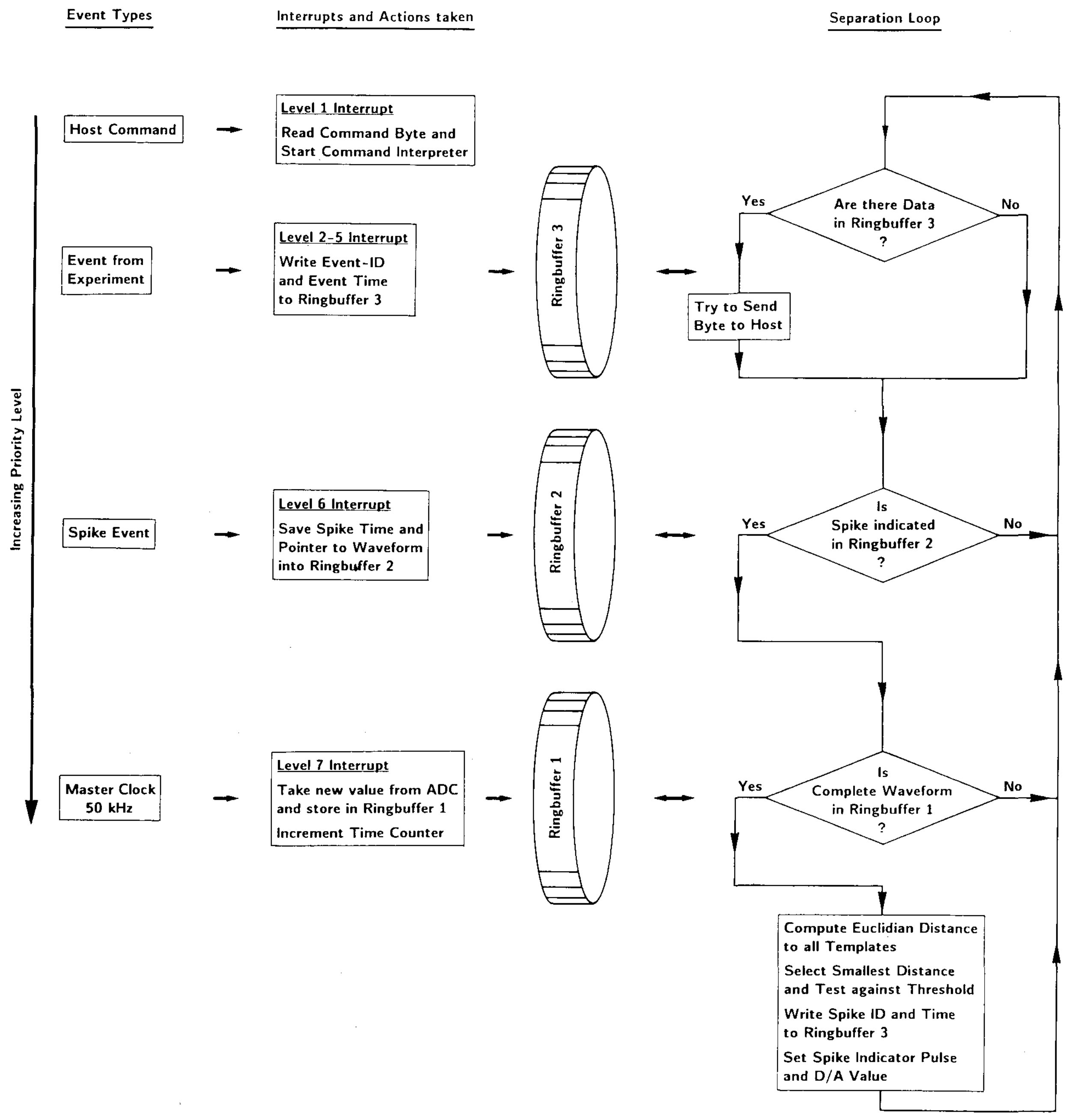
Event Types
Interrupts and Actions taken
Separation Loop
Increasing Priority Level
Host Command
Level 1 Interrupt
Read Command Byte and Start Command Interpreter
Event from Experiment
Level 2-5 Interrupt
Write Event-ID and Event Time to Ringbuffer 3
Spike Event
Level 6 Interrupt
Save Spike Time and Pointer to Waveform into Ringbuffer 2
Master Clock 50 kHz
Level 7 Interrupt
Take new value from ADC and store in Ringbuffer 1
Increment Time Counter
Ringbuffer 3
Ringbuffer 2
Ringbuffer 1
Are there Data in Ringbuffer 3 ?
Yes
No
Try to Send Byte to Host
Is Spike indicated in Ringbuffer 2 ?
Yes
No
Is Complete Waveform in Ringbuffer 1 ?
Yes
No
Compute Euclidian Distance to all Templates
Select Smallest Distance and Test against Threshold
Write Spike ID and Time to Ringbuffer 3
Set Spike Indicator Pulse and D/A Value

Fig. 3. Flow chart of the real-time spike sorting routine, handling the different events (spikes, triggers from the experiment, host commands), signalled to it by the interrupt service routines.

id or in its upper 4 bits the id of the activated digital input channels (id 1–4, where 1 corresponds to interrupt level 5, 4 to level 2); and bytes 3–6 contain time, with byte 3 being the most significant byte.

Events are stored by the host into a disk file, consisting of records which describe type and timing of the events; full-time resolution of 20 μs is conserved. A record consists of 2 integers, followed by 4 bytes. In case of a spike, the first integer gives spike id; in case of an activated digital input channel (e.g. a stimulus onset), the second integer gives input id. In each case the remaining integer is set to zero. The 4 bytes code time of occurrence, with byte 1 being the most significant and byte 4 the least significant part.

Performance and Limitations

Performance and limitations of the spike sorter algorithm, implemented on a PDP-11/02, were described in detail elsewhere (Sarna et al., 1988). The authors compared the present algorithm to the commonly used amplitude–time window device and a principal components device (Abeles and Goldstein, 1977; Eggermont et al., 1983). From their evaluation we quote: "We conclude that the simple amplitude–time window device is least able to cope with multineuron data. Such devices mix and miss waveform classes to a completely unacceptable extent, given the fact that alternative devices do exist.... When contrasted with a principal components sorting device, the new device turns out to be approximately equal in the accuracy, although requiring far less complex and expensive hardware for its realization, and far less 'tuning' by the operator".

Event rates

An obvious limitation of the spike sorter, and one which it shares with most other sorter procedures (including the ones mentioned above), is that the system is not designed to classify overlapping spikes on the same microelectrode (coincident spikes on different electrodes are obviously no problem). This evidently puts an absolute upper limit to the attainable maximum event rates. On the output side, maximum event rates are further limited by the speed of communication from single board system(s) to the host.

The system is able to recognize a trigger event, marking the occurrence of a spike, in a time much below the actual duration of the spike waveform. From this point of view, it is no problem to detect spikes which follow each other very rapidly, until they start to overlap, where classification becomes impossible. The time required for classification grows with the number of classes present, and is for 4 classes approximately 0.65 ms per spike. When spikes enter at intervals below this value, the CPU will still be concerned with classification of the preceding one(s). It is important to note, however, that classification of a spike need not necessarily be immediately after its occurrence: since the first ringbuffer has a capacity of 32,768 values (i.e. over 650 ms at 50 kHz), more than 650 ms are available before classification is due. All this implies that on the input side, maximum event rates are basically not limited by the duration of the classification process.

On the output side, event rates are limited by the baudrate of the RS232-interface and the size of the output queue (third ringbuffer). In the present version, the RS232 operates at 9600 baud; consequently it takes about 7 ms to transmit the 6 bytes defining an event and, hence, the capacity of the RS232 line is 142 spikes/s. This, however, is a continuous rate; since the sorter board is able to store data of 6144 classified events in its output cue, bursts at much higher rates can be accommodated and problems will only arise if the event rate exceeds 142 spikes/s over long periods of time. In addition, by changing one line in the single-board system's source code, the RS232's baudrate can be raised to 19,200 or 38,400, thereby doubling or even quadrupling the maximum continuous spike rate.

To summarize, when asking for the highest spike rate each single spike sorter system can handle, one should distinguish between the maximal long-term average spike rate and the maximal rate that may occur during a burst of spikes. The long-term average spike rate for each individual sorter is limited by the serial port capacity to a maximum of 142 spikes/s at 9600 baud, 284 at 19,200 baud, or 568 at 38,400 baud. Maximal spike rates during bursts may be very much higher, however, they are basically only limited by the requirement of having non-overlapping spike waveforms. In practice, this would allow for burst rates of over 1000 spikes/s.

Finally, the host computer, with a typical AT-like processing speed, should not pose any limitations on maximally attainable event rates for up to 8 spike sorters in parallel, communicating through as many RS232 interfaces (assuming that during data acquisition the user is suitably modest in his demands regarding on-line data analysis).

'Diagnostic' display

The diagnostic display of spike waveforms on the oscilloscope at a level corresponding to their classification has one limitation. Whenever two spikes occur within the time interval it takes for the oscilloscope beam to finish a sweep across the screen, the first waveform will be displayed correctly, i.e. superimposed on the other ones of its class; the second one, however, will be displayed within the same sweep, i.e. shifted to the right and not necessarily at the appropriate level. Note that this limitation, *only* refers to the oscilloscope display; in particular it does *not* affect proper classification of the second waveform by the spike sorter itself.

In view of the fact that classification is not always performed immediately upon arrival of the spike, another limitation of the proper oscilloscope display of spike waveforms would seem to reside in the finite time span of the analog delay line (2.56 ms). However, given the short time it takes to classify a spike (typically 0.65 ms, see above), problems would only occur when spike intervals become so short that waveforms start to overlap anyway, in which case the compound event is marked by the spike sorter as 'unclassified', as is indeed (and correctly) shown on the oscilloscope display.

Discussion

In the present paper we have described an efficient and low-cost implementation of a recently proposed algorithm for unsupervised, real-time sorting of multineuron recordings. As shown before (Sarna et al., 1988), the performance of this type of sorter is clearly much better than that of the usual time–amplitude window discriminators and certainly comparable to that of the most advanced type of spike sorters, the principal components sorter. The major advantage of our sorter above the principal components machine is two-fold: (1) its minimal need for user intervention; and (2) its considerably lower price, due to the combination of low-cost single-board microprocessors and widespread pc technology. These advantages, combined with the fact that a single host computer can handle a number of sorters simultaneously, lead to a further advantage above current spike sorting techniques. It enables the experimenter to combine the two major types of multi-unit recording in a single set-up at a feasible price: recording with multiple electrodes, each electrode being serviced by its own spike sorter system. This allows one to investigate short-range and long-range interactions within groups of neurons simultaneously on a larger scale than hitherto possible.

Our implementation of the actual classification of a spike waveform uses the computation of distance in 8-space to assign a given spike to one of the different templates. Salganicoff et al. (1988) mentioned as a possible improvement to incorporate the actual size and shape of the clusters of waveforms in the 8-space into this distance measure. This would indeed be an improvement if the noise on the spike waveforms were such that the space filled by one or more of the spike classes deviates significantly from a hypersphere and/or if the radii of these hyperspheres were to differ appreciably. The usual assumption is that this noise is additive and uncorrelated to the spike waveform. Such noise, however, would tend to extend a cluster equally in all dimensions, i.e. it would indeed result in a hypersphere. Moreover, such noise would cause all hyperspheres to be of approximately equal size. In other words, the straightforward distance criterion used in our implementation is in fact the logical consequence of the assumption of additive, uncorrelated noise. It is only for the cases of other types of noise (e.g. multiplicative) that a modified criterion, incorporating the actual cluster shape and size would improve classification. Such a case arises, for instance, when during a spike burst the size of a neuron's action potentials decreases systematically. Certainly the design of our sorter would allow for such modification fairly easily.

Like all other current spike sorters, the present one is not able to separate spikes that overlap in time. Should such overlap occur in a systematic fashion (i.e. with an approximately constant delay), then these spike configurations would show up as a separate class of spike waveforms, provided they occur often enough to be adequately represented in the learning set. Much more likely, however, is that such overlap shows variable delay, and thus does not lead to a separate category, but rather to a 'halo' of individual occurrences. In the sorting phase such multispike configurations would be usually so different from each of the templates that they would tend to be sorted as non-classifiable waveforms. Summarizing, the above arguments suggest that the issue of separating overlapping waveforms, although admittedly not addressed by the current spike sorter, is not the most severe problem in multiunit recordings.

Possible improvements

From a technical point of view, the present spike sorter is a software-based system, and an enormous amount of the sorter processor's power is used to continuously sample the analog data stream. This job could be done equally well by a 'direct memory access' (DMA) device. Freed of this sampling task, the processor would need only a fraction of the time for the remaining tasks. This, in turn, implies that a single processor could easily handle more than one analog data stream and thus support multielectrode arrangements. A further enhancement could be achieved by using discontinuous, rather than continuous sampling: upon each occurrence of a spike a short DMA transfer should be initiated, its duration covering only the time interval needed to describe the spike waveform. This would require the means for delaying the analog signal by a few hundred microseconds to include the rising flank of the action potential. Clearly such measures would increase the expense of hardware somewhat, but more important, they would further improve the cost–benefit relation. Especially for large-scale multielectrode arrangements, currently already incorporating tens of electrodes (e.g. Krüger, 1981) such methods seem to be mandatory in order to avoid an instrumental explosion with tens of sorters (possibly with multiple hosts) in a single experimental set-up.

Our realization of the real-time spike sorter does not include the actual detection of a spike, but rather uses an external trigger from a spike detector device, and sets the times of sampling relative to that input. Usually such a device is available in the standard electrophysiological laboratory, and is based on simple crossing of a selected amplitude value. This method is unfortunately subject to baseline error due to noise, which then in turn affects the location of samples on the waveform. Clearly, this jitter of spike position relative to the samples degrades the feature set and, hence, the template set and classification performance.

These problems associated with time jitter of the spike detection could be counteracted in the learning phase as well as in the sorting phase, but preferably in both. In the sorting phase one could try to find a more optimal alignment of each new incoming spike waveform with each of the templates by shifting it over a few time steps in both directions and trying to optimize the fit. This would basically multiply the time needed for classification (0.65 ms in the case of 4 classes) by the number of shifts considered. Since the time needed for classification proved not to limit the maximum allowable spike rate, such modification would be easily tolerable. This, however, does not address the more fundamental problem that the templates themselves may be degraded by the time jitter of spike detection. A possible approach to improve the templates during the learning phase might be to proceed as follows: after defining the different classes of spike waveforms and calculation of the templates, all spikes of each class could be shifted in time to obtain optimal fit with the corresponding template. After this alignment, one could either simply calculate new templates by averaging the newly aligned spikes within each class, or, alternatively, first determine an improved reduced feature set over all newly aligned spikes in the learning set and only then calculate new templates. Here too this would be done by averaging the newly aligned spikes within each class, now, however, at the new feature positions. This procedure could be reiterated until some criterion (e.g. on successive changes of the template) is met. Neither of these alternatives would involve any reclustering; consequently host intervention would not be required and the whole process could be carried out by the single-board computer.

An important issue during the learning phase is finding the appropriate distribution of reduced feature positions over the spike waveforms. These positions are selected successively on the basis of the corresponding variances, weighted by a function of the distances to already selected points in order to prevent undesired clustering of feature positions. The optimal function for weighting these distances might differ depending on various signal characteristics (e.g. noise level, waveform types), possibly related to recording area and/or animal species. Different choices of weight functions were in fact mentioned by Salganicoff et al.. (1988). However, they also suggested, after trying a number of alternatives, that the actual choice of the weight function should not be too critical: "Although no systematic tests and comparisons were made, it is likely that satisfactory performance can be obtained with a range of procedures." In our realization we implemented the square root of the distance as weight function: we provided the 63 necessary function values explicitly in the form of a table, for reasons of computation speed. In order to enable the experimenter to choose between different weight functions, one could implement the option to download this table from the host computer, which could easily generate a table for each desired function. This option would certainly increase flexibility of the system with respect to optimal feature selection.

What about miracles?

Finally, we want to give a very strong warning. A spike sorter is a machine to separate spikes on the basis of their waveform. This could in principle be done by a human being. One should bear in mind that, when the spike signals are so noisy that many of one's careful sorting decisions are turned into nonsense, the spike sorter will generate the same nonsense, only at a considerably higher rate.

Acknowledgements

The realization of the real-time spike sorter was carried out at the Max-Planck-Institute for Biological Cybernetics, Tübingen, F.R.G.; it was initiated during a sabbatical stay of one of the authors (G.L.G.) at this Institute. We wish to thank Michael Erb for helpful discussions regarding the design, Claudia Martin-Schubert for skilful assistance in the preparation of the figures, and Shirley Würth for correcting the English.

References

Abeles, M. and Goldstein Jr., M.H. (1977) Multispike train analysis, Proc. IEEE, 65: 762–773.

Eggermont, J.J., Epping, W.J.M. and Aertsen, A.M.H.J. (1983) Stimulus dependent neural correlations in the auditory midbrain of the grassfrog (*Rana temporaria L.*), Biol. Cybern., 47: 103–117.

Gerstein, G., Bloom, M., Espinosa, I., Evanczuk, S. and Turner, M. (1983) Design of a laboratory for multi-neuron studies, IEEE Trans. Syst. Man Cybern. SMS-13: 668–676.

Hartigan, J.A. (1975) Clustering Algorithms, Wiley, New York.

Krüger, J. (1981) Simultaneous recording with 30 microelectrodes in monkey visual cortex, Exp. Brain Res., 41: 191–194.

Krüger, J. (1983) Simultaneous individual recordings from many cerebral neurons: techniques and results, Rev. Physiol. Biochem. Pharmacol., 98: 177–233.

Salganicoff, M., Sarna, M., Sax, L. and Gerstein, G.L. (1988) Unsupervised waveform classification for multi-neuron recordings: a real-time, software-based system. I. Algorithms and implementation, J. Neurosci. Methods, 25: 181–187.

Sammon Jr., J.W. (1969) A nonlinear mapping for data structure analysis, IEEE Trans. Comput. C-18: 401–409.

Sarna, M.F., Gochin, P., Kaltenbach, J., Salganicoff, M. and Gerstein, G.L. (1988) Unsupervised waveform classification for multi-neuron recordings: a real-time, software-based system. II. Performance comparison to other sorters, J. Neurosci. Methods 25: 189–196.

Schmidt, E.M. (1984a) Instruments for sorting neuroelectric data: a review, J. Neurosci. Methods 12: 1–24.

Schmidt, E.M. (1984b) Computer separation of multi-unit neuroelectric data: a review, J. Neurosci. Methods, 12: 95–111.