Applications of Real-Time Median Filtering with Fast Digital and Analog Sorters

Steven B. Leeb, *Member, IEEE,* Alfredo Ortiz, Robert F. Lepard, Steven R. Shaw, and James L. Kirtley, Jr., *Fellow, IEEE*

***Abstract*—This paper explores applications of digital and analog hardware realizations of median filters suitable for potential use in real-time control and monitoring applications in power electronic circuits and drives. Median filters have the ability to suppress impulse noise in signals, while preserving underlying edges. Filter performance is demonstrated with results from two prototypes.**

***Index Terms*—Median filters, power electronic drives.**

I. Introduciton

THIS PAPER describes fast hardware implementations of a median filter, with emphasis on possible applications to power electronics. The median filter is a discrete, nonlinear filter which can remove spike noise from a signal, while accurately preserving underlying edges. It is one example of a group of filters that make use of a sorting or ranking operation to filter a waveform. This paper focuses on the median filter as a relatively easily understood and implemented example, that richly illustrates the potential of nonlinear filtering in power electronics.

Median and median-type filters have been popular for the past 20 years in signal and image processing applications [1]–[4]. In situations where the frequency spectra of the noise and the underlying signal overlap and the noise is highly impulsive, the median filter can have remarkable advantages over linear filters. Such situations are reasonably common in the implementation of power electronic circuits and machine drive systems.

Specialized hardware architectures for implementing median-type filters have appeared in the signal processing literature [6]–[8]. Many proposed architectures for hardware median filters are not optimal for use in real-time control applications because their latency is too large or increases unreasonably with filter size. Such designs could introduce intolerable delay. This paper reviews and employs filter designs which are intended for use in control and monitoring loops in power electronic circuits. Note that this architecture is designed for use in filtering "signal strength" sources, not "power strength" inputs and outputs.

Manuscript received August 12, 1996; revised February 12, 1997. Recommended by Technical Editor R. Isermann. This work was supported in part by the Electric Power Research Institute, by the Massachusetts Institute of Technology Carl Richard Soderberg Career Development Chair, and by a Career Award from the National Science Foundation.

The authors are with the Laboratory for Electromagnetic and Electronic Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

Publisher Item Identifier S 1083-4435(97)04814-X.

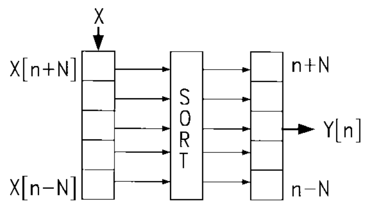


Fig. 1. Schematic diagram of a median filter.

Section II of this paper describes the median filter and some of its properties. To show the power of the median filter, a review of off-line results first presented in [5] compares outputs from the median and linear filters for a common input waveform, i.e., a switch voltage transient in a flyback converter. New off-line results in this section show the result of filtering a noisy motor tachometer signal with the median filter, suggesting one potential real-time application. Section III details some of the problems and tradeoffs associated with several traditional digital median filtering architectures. Section IV describes a digital architecture for a real-time median filter and discusses the benefits of the design. In Section V, results from an actual prototype of the design are presented. Section VI reviews the design and performance of an "analog" median filter, which performs its filtering operation without first digitizing the input samples.

II. Median Filter Basics

The median filter operates on a windowed section containing an odd number of evenly spaced, discrete samples from an input stream of data X. The window slides across the input stream, advancing one point at a time. At any instant, the output of the filter Y is the median of the data in the window. The schematic in Fig. 1 illustrates this process for a window of $2N+1$ points. This filter is often referred to as a filter of size N. In Fig. 1, the median value of the data window is determined by a sorting operation; this is not the only method for determining the median, but an adaptation of this approach provides one of the fastest possible hardware implementations. In a real-time setting, we have found it convenient to initialize the data window with zeros, although other values are possible [5], [9].

There are several ways to understand the behavior of the median filter. Studying the frequency properties of the filter is *not* typically one of the best ways, since the median filter is nonlinear. One successful approach is to consider how the

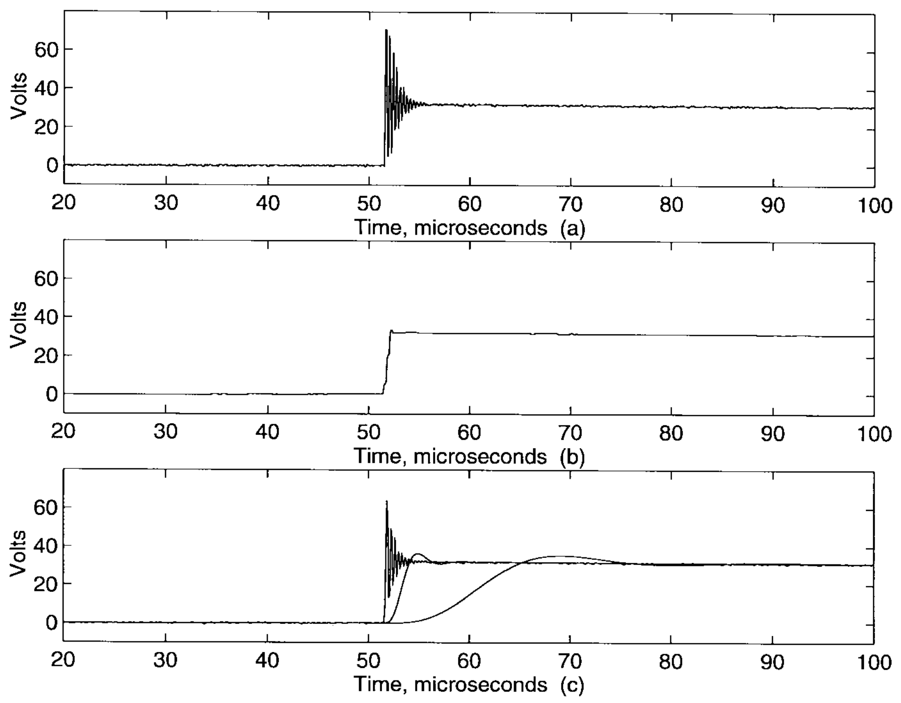


Fig. 2. Switch voltage. (a) Actual. (b) Median filtered. (c) Low-pass filtered.

median filter alters the local *geometry* or shape of a waveform. We will avoid extensive formalism here; the interested reader is directed to [3], [5], [9], and [11] for detailed explications of median filter behavior in geometric terms.

Any discrete waveform may be described as a sequential collection of constant neighborhoods, edges, impulses, and oscillations [5], [9]. Passing a signal once through a median filter will eliminate impulses and reduce oscillations. It has been shown that repetitive median filtering of a finite length signal will produce, after a finite number of repetitions, a *root* signal which is invariant to further applications of the filter [9]. Such a root signal consists only of edges and constant neighborhoods. Hence, if a waveform consists of an underlying signal of edges and constant neighborhoods corrupted with noise which consists of impulses and oscillations, the median filter will remove or reduce the noise without modifying the underlying signal.

The median filter has proven to be an invaluable off-line tool for smoothing experimental curves for comparison to simulated or theoretical data [5]. For example, Fig. 2(a) shows the switch voltage in a flyback converter when the controllable switch turns off. The "spike" on the rising edge of the step is a high-frequency ringing created by the MOSFET body capacitance and transformer leakage inductance. Fig. 2(b) shows the results of median filtering this waveform with a filter of size $N = 8$. Fig. 2(c) shows the results of filtering the waveform in Fig. 2(a) with three different fourth-order Butterworth filters, the cutoff frequencies of which span a range of values with respect to the sample frequency. The median filter is uniquely capable of removing the spike while preserving the edge.

Fig. 3(a) shows an unfiltered plot of speed versus time during the free acceleration of a 3-hp, four-pole induction motor. This plot was developed by sampling the voltage across a small dc tachometer connected to the motor. Commutator brush noise created the spikes in the waveform. Attempts to remove these spikes with a linear filter are complicated by the fact that the rate of occurrence of the spikes is a function of the motor speed, which is rapidly changing during the transient. Hence, it may be difficult or impossible in many situations to select a (nonadaptive) linear filter cutoff frequency which removes the spikes without distorting the underlying edge. In [12], for example, the authors were forced to rely on an *ad hoc* off-line scheme for removing such spikes before further processing of the waveform could proceed. Fig. 3(b) shows the results of filtering this waveform with a median filter of size $N = 4$.

The success of the median filter in the off-line processing of a variety of waveforms from power electronic circuits and motor drive systems has led us to explore potential filter architectures for use in real-time monitoring and control applications.

III. Digital Design Overview

Three broad classes of digital median filter architectures are described in [13].

- *Original order* designs maintain the window of samples in chronological order in a buffer. The median is produced by some "median finding" [14] or sorting hardware. See [6], for example.
- *Sorted order* designs maintain the window of samples in sorted order. A linked-list arrangement is used to tag each point with an "age" index to indicate its time of arrival in the window. See [7], for example.

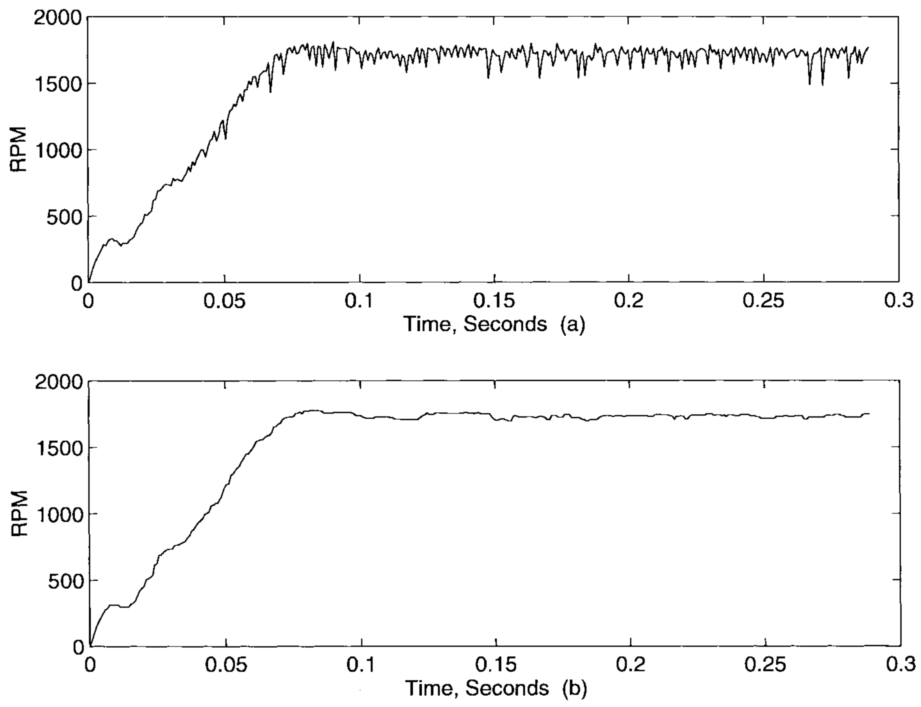


Fig. 3. Induction motor free-acceleration. (a) Actual. (b) Median filtered.

- *Bitwise* designs implement a variety of sophisticated algorithms to determine the median entry in the data window based on the assumption that the number of bits representing individual samples is relatively small. See the "stack filter" proposed in [8] for one approach in this class.

An excellent and detailed summary of the different tradeoffs between speed and complexity for these approaches, including several important variations and hybrids, may be found in [14].

Minimizing delay or *latency* is an important design constraint in the development of a hardware median filter. Referring to Fig. 1, we see that, in mathematical terms, filter operation on the data window at time n for a filter of size N may be described as

$$Y[n] = \text{Median of } (X[n-N], \cdots, X[n], \cdots, X[n+N])$$

where $X[n]$ represents a discrete input data sample and $Y[n]$ is the filter output, both at time n. In principle, the median filter is noncausal, i.e., it requires future samples of the input data stream X to compute the output value at time n. In practice, there must be a pure delay associated with an implementation of a median filter. For a filter of size N, the delay cannot be less than N sample intervals, since N "future" points are required to predict an output point. In the off-line results presented in Section II, the output waveforms have been shifted by N samples to correct for this pure delay. In a real-time setting, the delay is inescapable, and the decision to use a median filter with its nonlinear smoothing properties must be carefully balanced against other filter types, which may have linear or otherwise frequency-dependent phase characteristics. To avoid further complicating a potential control or monitoring problem, a hardware median filter of size N should, ideally, introduce no more than the inevitable N sample interval delay. Generally, we suspect that designs with excessive latency or, worse, with latencies which increase with window size or content, should be avoided.

In a fully digital implementation, where input data is digitized by an analog-to-digital converter, the sorted order architecture can be constructed to provide the best overall performance (i.e., minimum latency) [14]. In short, this design implements the equivalent of a hardware insertion sort algorithm [15]. The goal of this design is to take advantage of the fact that as the window "slides" across the data point by point, only one data value is being discarded and one added at any time. The rest of the list is already in sorted order. By using a collection of parallel comparators to simultaneously compare the incoming value to each point in the sorted list, it is possible to discard the old data point and insert the new point in a fixed number of clock cycles, regardless of the length of the list. In our digital sorted order implementation, presented in the next section, this insertion is accomplished in two clock cycles.

Our digital filter is similar in many respects to the sorted order architecture reported in [7]. In our design we have replaced the bit-slice architecture in [7] with a word parallel design in which all of the W bits of each data point are written, compared, or read simultaneously. This word parallel approach divides by W the number of clock cycles needed to perform an insertion cycle compared to a "bit-serial" design [14]. Readers interested in further construction details are encouraged to examine [7], in addition to the next section. In particular, [7] provides guidelines for determining and comparing silicon area for VLSI implementations which are beyond our current scope. The description of our sorted order filter architecture presented in the next section is adopted in part from [16]. See [16] for more information, including detailed schematics.

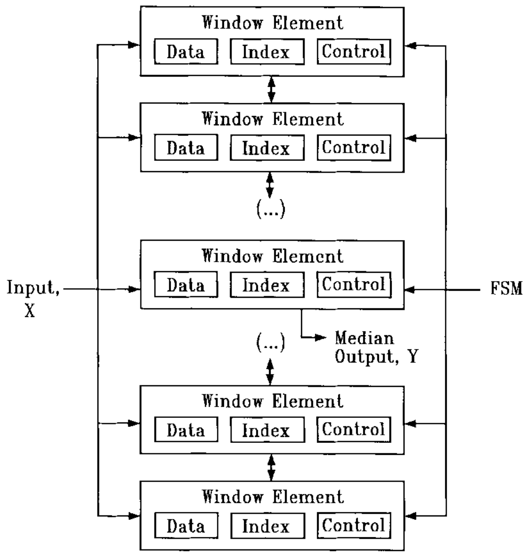


Fig. 4. Median filter block diagram.

TABLE I
ELEMENT ACTION TABLE

Element	State *Shift*	State *Load*

Top	If (index = 2N+1)
	data = data below
	index = index below
		If (data <= input)
		data = input
		index = 0
		else
		index += 1

Middle	If (index = 2N+1 or *propagated*=TRUE)
	data = data below
	index = index below
		If (data <= input and data above>input)
		data = input
		index = 0
		else if (data>input and data above>input)
		index += 1
		else
		data=data above
		index = index above+1

Bottom	If (index = 2N+1)
	do nothing
		If (data above > input)
		data = input
		index = 0
		else
		data=data above
		index = index above+1

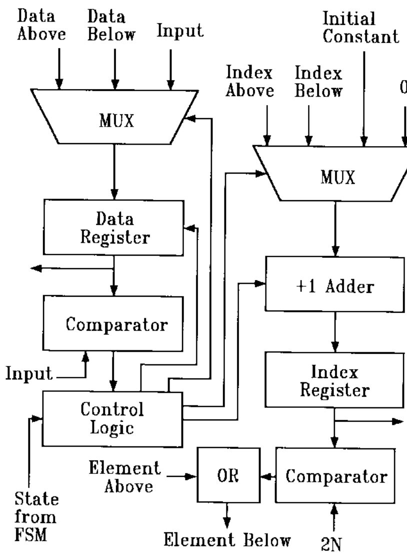


Fig. 5. Typical window element.

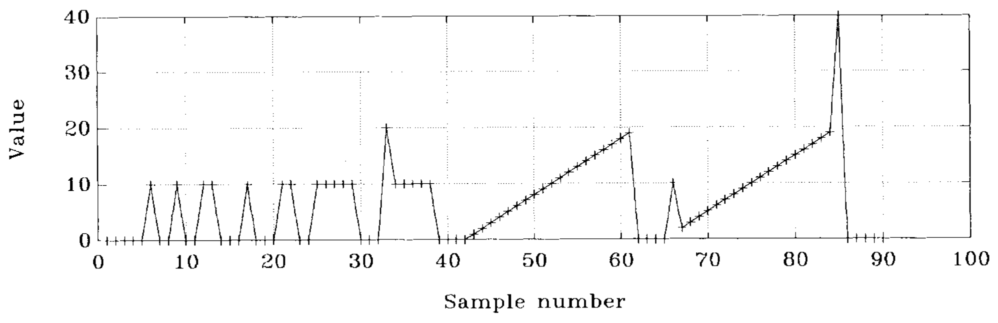
IV. DIGITAL HARDWARE IMPLEMENTATION

A block diagram of our $2N+1$ point hardware filter window is shown in Fig. 4. The actual values from the input stream X are stored in the registers labeled **Data**. The "age" of each sampled value, modulo $2N+1$ counting from zero, is stored in the auxiliary register labeled **Index** associated with each data register. A **Window Element** is a combination of the data and index registers and supporting circuitry. The top element in our design contains the largest data value in the window and the bottom element contains the smallest, with all other data in rank order in the middle elements. After an initialization in which each window element is loaded with a data value of zero and an index value ranging from zero to $2N$, the steady-state operation of this filter window is controlled by a finite-state machine (FSM) which has two states, *shift* and *load*.

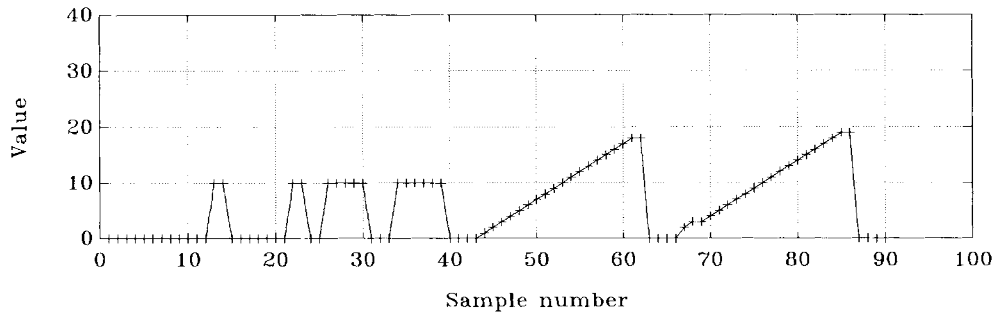
The *shift* operation prepares the window elements to receive a new input value from the analog-to-digital converter. During the *shift* state two processes occur: the value from the *data* register of the middle element is read into an output register as the median value, and each element compares the value of its *index* register with the constant $2N$. The element which matches contains the oldest point; an index of zero indicates the newest point. The result of the index age comparison in each element is OR-gated with the result of the index comparison from the element above. This composite result propagates to the element below. All elements below the oldest point, determined by the propagated index age comparison, shift the values of their sampled data and index registers up one element, eliminating the oldest point by writing over it, and freeing the bottom sampled data and index registers.

The *load* operation performs an insertion sort to place a fresh input value from the analog-to-digital converter into its correct position in the window. The fresh input value is broadcast to all of the window elements. Each window element compares the input value with the value in its *data* register. At some "break point" in the window, all of the elements below the break point will be less than the input value, and all of the elements above the break point will be greater than the input value. The break point is unique in that the element immediately below it compares low to the input value and the element above it compares high; this condition is easily tested in parallel for all pairs of window elements. All elements below the break point shift down, opening an element in which the new data point from the input bus is inserted. The *index* register for this new point is set to zero. All other index registers are incremented by one. When the *load* operation is complete, the machine returns to the *shift* state and the process begins again. A new output point, the median data value in the current window, is made available after every two finite-state machine cycles (i.e., clock cycles), regardless of the size of the window or the type of waveform being filtered.

Fig. 5 shows a schematic of a typical window element which operates under the two-state scheme described above. Slight modifications must be made for the top and bottom



(a)



(b)

Fig. 6. Test input and output.

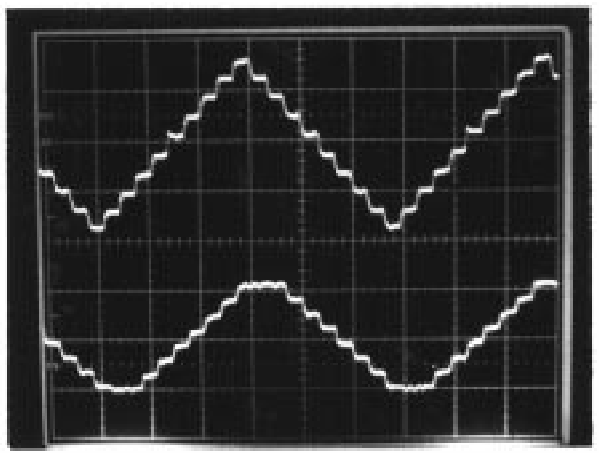


Fig. 7. Triangle-wave input.

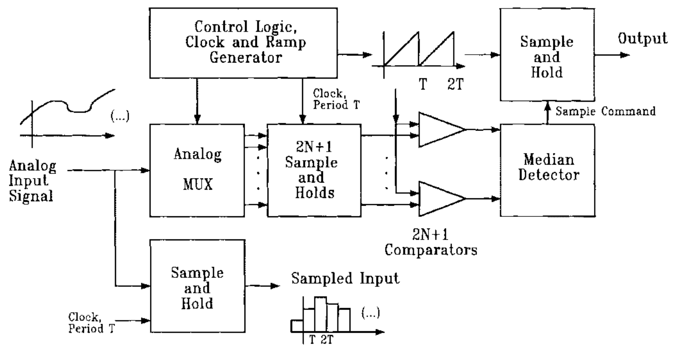


Fig. 8. Analog filter block diagram.

elements, since these elements do not have elements above or below, respectively. An action table summarizing the behavior of the top, bottom, and middle elements during the *shift* and *load* states is shown in Table I. The *propagated* signal, checked during the *shift* state for each middle element, refers to the value of the propagation chain that indicates the presence of an index match with the constant $2N$ above an element.

Only the top and bottom elements, therefore, require special control coding. A filter of arbitrary size may be constructed by stacking sufficient “generic” middle elements between a top and bottom pair. For simplicity, analog signals are level shifted prior to analog-to-digital conversion in our prototype, so the lowest-valued digital sample stored in the window is zero. Hence, our prototype does not handle negative sample values digitally; all register values are positive binary numbers. The median output of the filter can be inverse shifted to restore the original bias.

In the next section, we present results from a prototype filter.

V. Experimental Results

To study the validity and potential performance of the proposed design, we constructed a hardware prototype of a median filter of size one (a window size of three points). First, to test the hardware sorter, the prototype was used to filter digital data stored in an electrically programmable read-only memory (EPROM). A section of the digitized input waveform is shown in Fig. 6(a). The precise sample locations are indicated with + symbols. The edges and constant neighborhoods in the signal are corrupted with spike noise. Fig. 6(b), the output of the filter, shows that the noise is removed by the prototype filter with its three-point window.

Fig. 7 shows the results of presenting a 35 kHz triangle wave as the input to the filter test hardware. The top trace in Fig. 7 shows the digitized samples fed as input to the median

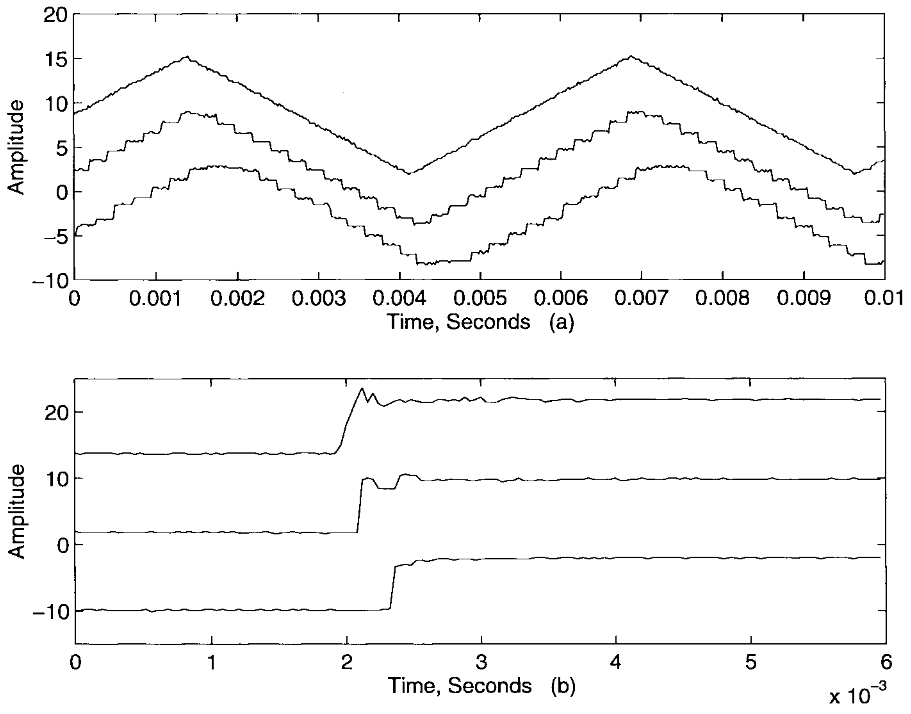


Fig. 9. (a) Triangle inputs. (b) Step inputs.

filter. For this $N = 1$ filter, the crest and trough samples of this triangle wave are part of signal oscillations where the wave is changing directions. We expect, therefore, that these oscillations will be reduced in the filtered output. The bottom trace in Fig. 7 shows the output of the median filter. As expected, the crest and trough samples have been eliminated. The filter sampling rate for this experiment was 575 kHz. We have had similar successes operating our hardware filter at sample rates as high as 1.23 MHz, near the limit of our digital-to-analog conversion hardware. Even at this reasonably fast sample rate, we saw no signs in our breadboarded prototype of performance degradation which would prevent the filter from running at even higher sample rates.

VI. Analog Hardware Implementation and Results

Like the digital median filter, an "analog" median filter operates on input data collected at discrete sample instants. However, input data is *not* digitized before the filtering operation. Instead, the filter operates directly on sampled and held values of the input waveform (see [17] and [18] for further discussion and an example). A block diagram of the analog hardware implemenation constructed for our experiments is shown in Fig. 8.

The analog filter is essentially an "original-order" design. A total of $2N + 1$ discrete, analog samples collected T s apart are stored in a ring buffer formed from $2N + 1$ sample-and-hold units, as shown in Fig. 8. The samples are stored in chronological order counting back from the most current sample stored in the ring. To generate an output sample, the input samples are compared to a ramp by a bank of comparators. The ramp rises from the lowest anticipated input value to the highest every T s. As the ramp value rises each period, the comparators change state or "fire" as the ramp rises above the value of the particular input sample under comparison. The median detector, a finite-state machine implemented by a programmable array logic device, monitors the comparators and generates a pulse each period when $N+1$ comparators have fired. This pulse latches the current value of the ramp into an output sample-and-hold. This value is the median value of the data in the input ring buffer.

Notice that fresh output samples are not necessarily latched into the output sample-and-hold at precisely the same instant during every clock period. Instead, the output is loaded when the ramp has passed the median value of the data in the input buffer. Therefore, although the output data will be updated once during each ramp period of T s duration, the analog filter does not exhibit the precisely synchronous behavior of the digital implementation. Also, notice that it is relatively easy, in comparison to the fully digital filter, to modify the filter to handle negative-valued input signals. Different input ranges can be accomodated by level shifting the ramp, so that its lowest and highest values encompass the full input range.

For convenience in examining the performance of the analog filter, an additional sample-and-hold, as shown in Fig. 8, was used to store samples of the input waveform for comparison to the filtered output waveform. Our prototype analog output filter could easily be configured to operate with a three-, five-, or seven-point window. Real-time results from the five-point window filter are shown in Fig. 9. Fig. 9(a) shows the result of filtering a triangle-wave input signal. Fig. 9(b) shows the result of filtering a step with an oscillation on the rising edge. In both cases, the top traces show the original analog input waveforms. The next traces down show the sampled versions of the input waveforms. The bottom traces show the filtered output signals. The waveforms have been offset from each other for clarity.

VII. Conclusions

This paper has presented hardware median filter designs which meet the stringent requirements necessary for control and monitoring applications. The results from the hardware prototypes substantiate the expected performance of the filter. Several important problems remain in the development of specific applications.

A point mentioned previously that cannot be overemphasized is that the median filter is nonlinear. Standard linear analysis techniques for the design of feedback control systems, for example, cannot be directly applied if a median filter is incorporated in the closed loop. Developing satisfactory techniques for determining stability and other properties of systems which contain nonlinear building blocks like the median filter is an open and active area of research.

Both the digital and analog prototypes were constructed with discrete components and integrated circuits. With only the use of readily available, off-the-shelf components, the analog filter was substantially easier to construct for a given window size than the digital implementation. The complex but highly modular structure of the digital filter naturally points toward a custom VLSI implementation, an approach taken by many researchers studying nonlinear filtering in the signal-processing fields. In situations where the input data is presented digitally, e.g., in a real-time digital controller, the digital median filter might be desirable. The commercial success of fast floating point units for modern digital signal-processing applications is well known. We suspect that, in the long run, the power of nonlinear filtering may generate a demand for the development of commercially available chip packages for nonlinear operations, such as sorting.

Acknowledgment

The authors gratefully acknowledge Prof. W. C. Karl, Prof. G. C. Verghese, K. O. Prakah-Asante, and Dr. L. A. Jones for valuable assistance and discussions. Essential hardware for this project was made available through generous donations from the Intel Corporation and Tektronix.

References

[1] G. R. Arce and N. C. Gallagher, Jr.,“BTC image coding using median filter roots,” *IEEE Trans. Commun.*, vol. COM-31, pp. 784–793, June 1983.

[2] P. A. Maragos and R. W. Schafer, “Morphological skeleton representation and coding of binary images,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 1228–1244, Oct. 1986.

[3] R. T. Hoctor and S. A. Kassam, “An algorithm and a pipelined architecture for order-statistic determination and L-filtering,” *IEEE Trans. Circuits Syst.*, vol. 36, pp. 344–352, Mar. 1989.

[4] I. Pitas and A. N. Venetsanopoulos, *Nonlinear Digital Filters: Principles and Applications*. Norwell, MA: Kluwer, 1990.

[5] W. C. Karl, S. B. Leeb, L. A. Jones, J. L. Kirtley, and G. C. Verghese, “Applications of a class of nonlinear filters to problems in power electronics,” in *Proc. IEEE Power Electronics Specialists Conf.*, June 1990, pp. 35–42.

[6] K. Oflazer, “Design and implementation of a single-chip 1–D median filter,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 1164–1168, Oct. 1983.

[7] G. R. Arce and P. J. Warter, “A median filter architecture suitable for VLSI implementation,” in *Proc. 23rd Allerton Conf. Commun., Control, Computing*, Oct. 1984, pp. 172–181.

[8] E. J. Coyle, “The theory and vlsi implementation of stack filters,” in *VLSI Signal Processing*. New York: IEEE Press, 1986.

[9] G. R. Arce, N. C. Gallagher, and T. A. Nodes, “Median filters: Theory for one- and two-dimensional filters,” in *Advances in Computer Vision and Image Processing*. Greenwich, CT: JAI Press, 1986.

[10] N. C. Gallagher, Jr. and G. L. Wise, “A theoretical analysis of the properties of median filters,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 1136–1141, Dec. 1981.

[11] P. Maragos and R. W. Schafer, “Morphological filters—Part II: Their relations to median, order-statistic, and stack filters,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 1170–1184, Aug. 1987.

[12] B. Szabados, J. H. Dableh, G. M. Obermeyer, R. E. Draper, and R. D. Findlay, “Measurement of the torque-speed characteristics of induction motors using an improved new digital approach,” *IEEE Trans. Energy Conversion*, vol. 5, pp. 565–571, Sept. 1990.

[13] G. R. Arce, P. J. Warter, R. E. Foster, “Theory and VLSI implementation of multilevel median filters,” presented at the Int. Symp. Circuits Systems, June 1988.

[14] D. S. Richards, “VLSI Median Filters,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 145–153, Jan. 1990.

[15] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes in C*. Cambridge, U.K.: Cambridge Univ. Press, 1988.

[16] A. Ortiz, “Flash filter: Median filtering by insertion sort,” B.S. thesis, Dep. Elect. Eng. Comput. Sci., MIT, Cambridge, MA, May 1990.

[17] J. S. Li and W. H. Holmes, “Analog implementation of median filters for real-time signal processing,” *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1032–1033, Aug. 1988.

[18] J. P. Fitch, E. J. Coyle, and N. C. Gallagher, “The analog median filter,” *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 94–102, Jan. 1986.

Steven B. Leeb (S’89–M’91) received the B.S., M.S., E.E., and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, MA, in 1987, 1989, 1990, and 1993, respectively.

Since 1993, he has been a Faculty Member with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, where he currently serves as the Carl Richard Soderberg Assistant Professor of Power Engineering in the Laboratory for Electromagnetic and Electronic Systems. He is concerned with the design, analysis, development, and maintenance processes for all kinds of machinery with electrical actuators, sensors, or power electronic drives.

Dr. Leeb is a member of the IEEE Power Engineering Society and a Junior Faculty Fellow of the Massachusetts Institute of Technology Leaders for Manufacturing Program. He is also a member of Tau Beta Pi and Eta Kappa Nu.

Alfredo Ortiz received the B.S. degree in electrical engineering from the Massachusetts Institute of Technology, Cambridge, MA, in 1990.

Robert F. Lepard received the B.S. and M. Eng. degrees from the Massachusetts Institute of Technology, Cambridge, MA, in 1996.

He is currently serving in the U.S. Air Force as a Second Lieutenant in the Aeronautical Systems Command, Wright-Patterson Air Force Base, Dayton, OH.

Steven R. Shaw received the B.S. and M. Eng. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, MA, in 1995 and 1997, respectively.

He is currently a Graduate Research Assistant in the Laboratory for Electromagnetic and Electronic Systems, Massachusetts Institute of Technology.

James L. Kirtley, Jr. (S'69–M'71–SM'80–F'91) received the B.S. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, MA, in 1968 and 1971, respectively.

In 1971, he joined the faculty of the Massachusetts Institute of Technology, where he is currently a Professor of Electrical Engineering.

Dr. Kirtley is Vice Chairman and Technical Paper Coordinator of the IEEE Power Engineering Society Electric Machinery Committee, a member of the Synchronous Machinery and Machinery Theory Subcommittees, and a member of the Editorial Board of Electric Machines and Power Systems. He was Chairman of the 1990 International Conference on Electric Machines. He is a member of the International Conference on Large High Voltage Electric Systems (CIGRE) and a Registered Professional Engineer in Massachusetts.