Stack filters

Article *in* IEEE Transactions on Acoustics Speech and Signal Processing · September 1986

DOI: 10.1109/TASSP.1986.1164871 · Source: IEEE Xplore

3 authors, including:

Edward Coyle
Georgia Institute of Technology
198 PUBLICATIONS **7,597** CITATIONS

SEE PROFILE

Neal Gallagher
University of Central Asia
136 PUBLICATIONS **5,147** CITATIONS

SEE PROFILE

Stack Filters

PETER D. WENDT, MEMBER, IEEE, EDWARD J. COYLE, MEMBER, IEEE, AND NEAL C. GALLAGHER, JR., MEMBER, IEEE

Abstract—**The median and other rank-order operators possess two properties called the *threshold decomposition* and the *stacking* properties. The first is a limited superposition property which leads to a new architecture for these filters; the second is an ordering property which allows an efficient VLSI implementation of the threshold decomposition architecture.**

Motivated by the success of rank-order filters in a wide variety of applications and by the ease with which they can now be implemented, we consider in this paper a new class of filters called *stack filters*. They share the threshold decomposition and stacking properties of rank-order filters but are otherwise unconstrained. They are shown to form a very large class of easily implemented nonlinear filters which includes the rank-order operators as well as all compositions of morphological operators.

The convergence properties of these filters are investigated using techniques similar to those used to determine root signal behavior of median filters. The results obtained include necessary conditions for a stack filter to preserve monotone regions or edges in signals. The output distribution for these filters is also found.

All the stack filters of window width 3 are determined along with their convergence properties. Among these filters are found two which we have named *asymmetric median filters*. They share all the properties of median filters except that they remove impulses of one sign only; that is, one removes only positive going edges, the other removes only negative going edges, while the median filter removes impulses of both signs. This investigation of the properties of stack filters thus produces several new, useful, and easily implemented filters.

I. INTRODUCTION

MANY of the signal processing problems of greatest interest nowadays concern the suppression of noise that is non-Gaussian, nonadditive, and sometimes not even uncorrelated with the signal. The image shown in Fig. 1 provides a good example of such a situation. It shows the signal returned from a scene scanned with a laser imaging system. The scene consisted of a strongly reflective copper corner cube sitting on a table, with a forest in the background. If no noise had corrupted the return, it would have appeared as a solid black square—the copper corner cube, located somewhere against a relatively uniform gray background, representing the forest and the table.

As can be seen from the figure, however, the image actually obtained is severely corrupted. There at least three noise sources responsible for this degradation. The first, and most severe, is the speckle noise that always accompanies a coherent imaging system. It is highly correlated with the signal, is multiplicative, and non-Gaussian; it is also difficult to model analytically. The second source of noise is the fluctuation in the intensity of the laser beam caused by atmospheric turbulence and power supply variations. The third, but probably not the last, source of noise in this image is the white Gaussian noise added by the receiver. These three noise sources all combine in a way which is difficult to characterize, except to say that it is impulsive in nature, correlated with the signal, and nonadditive.

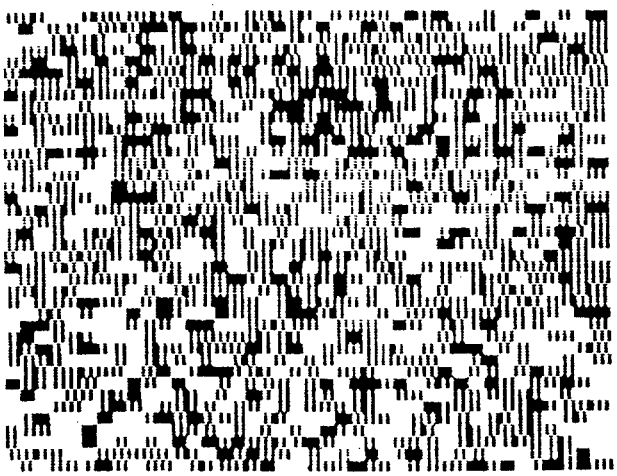


Fig. 1. Image before median filtering. This is an original two-dimensional image obtained by Dr. J. Johnson of Redstone Arsenal with an experimental laser imaging system at the far infrared wavelength of 1.2 mm. The image was obtained by scanning the laser over a scene consisting of a highly reflective copper corner cube sitting on a table with a forest in the background. The location of the cube is not apparent, however, because of speckle noise, thermal noise in the receiver, fluctuations in the laser beam's intensity, and other noise sources.

The question is: what filter should be used on the image in Fig. 1? It should not be a linear filter, since linear filters are optimal only for additive Gaussian noise, and since a linear filter would have to blur the sharp edges of the desired signal in order to remove the impulses created by the noise. In fact, all the linear filters that were applied to this image produced miserable results. Thus, a nonlinear filter should be used, but which one?

This question is more difficult than it seems. The reason is that for situations such as those in Fig. 1, in which it is very difficult to characterize the noise source, the standard nonlinear optimal filtering approaches are of little use. For instance, the minimum mean-square-error approach discussed in [1] and [2] cannot be used because it essentially assumes the noise can be completely characterized. The designer is thus left to guess a statistical description of the noise and hope that the resulting optimal

Manuscript received April 1, 1985; revised December 2, 1985.

The authors are with the School of Electrical Engineering, Purdue University, West Lafayette, IN 47907.

IEEE Log Number 8608131.

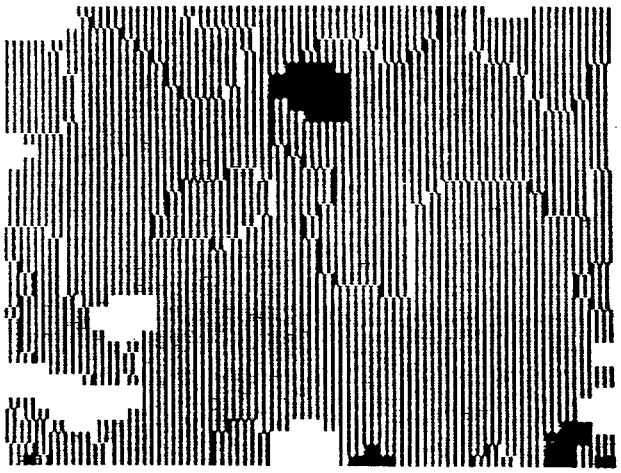


Fig. 2. Image after median filtering. Upon our suggestion, Johnson and his colleagues applied the median filter to the noise corrupted image in Fig. 1. The location of the copper cube, which was obscured in the original, is now apparent as the dark region in the center near the top. None of the other filtering techniques that were tried produced results as good as these.

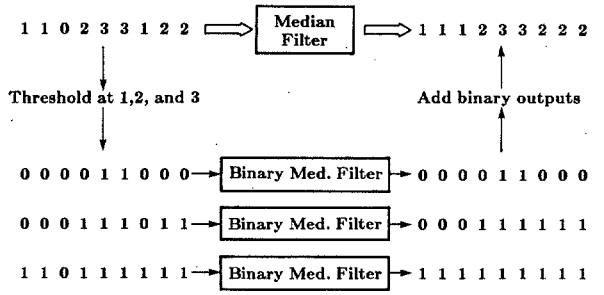


Fig. 3. Window width 3 median filtering by threshold decomposition [12], [13]. The 4-valued input at the upper left is median filtered the old way—by sorting the samples in the window—by following the heavy arrows. It is median filtered the new way—by threshold decomposition—by following the slender arrows. The binary signal on the bottom left is obtained by thresholding the input signal at level 1; the second from the bottom by thresholding at level 2, etc. The output signal is produced by adding together the binary outputs of the binary median filters at each sampling instant.

nonlinear filter is robust if the description is in error. Furthermore, except in relatively few cases, the equations specifying the optimal nonlinear filter cannot be solved. And even if they can be solved, the resulting filter is usually infinite dimensional [3] and, therefore, impossible to implement. The designer is then left to search for suboptimal truncations of the infinite dimensional optimal filter. The performance of the suboptimal filter is then difficult to determine, it may not be robust, and heroic efforts may still be required to implement it. So what should be done for the problem presented by Fig. 1?

In this particular instance, it appeared appropriate to use a median filter [4], [5]. The results obtained when this filter was applied to Fig. 1 were very good. They can be seen in Fig. 2, in which the location of the copper corner cube is apparent because the noise has been almost completely eliminated. The image retains the sharp edges that characterize the cube, although their location has been altered somewhat by the edge jitter caused by the noise. But even with the edge jitter, the performance of the median filter was not matched by any of the other filtering techniques tried on this application.

Analytical explanations of the median filter's success in applications such as the above are now becoming available [5]–[17]. Previous to these, the use of the filter was based on its proven edge preserving and impulse removing properties, its robustness, its ease of implementation, and, perhaps most importantly, the fact that it has worked so well in many different applications.

Motivated by the success and ease of implementation of the median filter, we develop, in this paper, an entire class of filters which share these characteristics. Specifically, they share the properties which allow a new, very efficient VLSI implementation of rank-order filters [14], [15]. These properties are the superposition property known as the *threshold decomposition* [12], [13], [26]–[28], [30] and an ordering property known as the *stacking property* [12], [13], [30]. Because all of these new filters share the stacking property, they are called *stack filters* [32].

After defining stack filters, we show that they can be constructed as "stacks" of positive Boolean functions. This shows that these filters, include, as a subset, all the ranked-order operators, and also represent all possible compositions of morphological filters [26], [27], [31].

We then investigate the convergence and syntactic behavior of these filters to determine when they preserve monotone signals, edges, and median filter roots, since often we desire filters which have these signals in their "passbands." The formula for the output distribution of any stack filter is also derived, since it is needed to determine the statistical behavior of these filters. Finally, we determine the behavior of all the stack filters of window width three.

As just mentioned, stack filters share the threshold decomposition property and the stacking property of rank-order operators. These properties are illustrated in Fig. 3, which is discussed in Section II when we review these defining properties of stack filters in more detail.

In Section III, we show that the stacking property implies that stack filters consist of stacks of positive Boolean functions. We then review some of the properties already known about positive Boolean functions.

In Section IV, we derive results concerning the convergence behavior of these filters. This type of syntactic analysis is necessary to determine the characteristics of these filters just as the analysis of root signals is necessary for an understanding of the operation of median filters. We thus determine the edge preservation characteristics of stack filters, and find conditions under which a stack filter will preserve edges and monotone regions in signals. We also find the rate at which these filters converge to roots when they preserve signals which are also roots of the median filter. This analysis indicates that there are many stack filters with this type of desirable behavior.

In Section V, we determine the statistical properties, primarily the output distributions, of these filters. It is

shown that the minimum sum of products expression that determines a particular stack filter, also determines its output distribution in a very simple way. This makes evaluation of the output distribution of these filters significantly simpler than it was in the past [21], [24].

In Section VI, we examine the properties of all stack filters of window width three to demonstrate the presence of filters that were not previously known. The syntactic analysis of these filters leads to two window width 3 filters we call *asymmetric median filters*. In brief, they preserve edges just like the median filters, but they have different impulse removal properties. One asymmetric median filter removes only positive going impulses; the other removes only negative going impulses.

We conclude with Section VII by providing some comments and indicating directions for future work.

II. The Threshold Decomposition and Stacking Property [12], [13], [27]

Since the threshold decomposition and stacking properties of rank-order operators will become, in the next section, the defining properties of stack filters, we review these two properties.

In brief, the threshold decomposition states what is illustrated in Fig. 3 with a window width 3 median filter; namely, that passing an M-valued discrete time signal through a rank-order filter is equivalent to the following procedure.

1) Decomposing the M-valued input signal into a set of $M - 1$ binary signals. The kth binary signal, where k is an integer in $\{1, 2, \cdots, M - 1\}$, is obtained by thresholding the input signal at the value k. It takes on the value 1 whenever the input signal is greater than or equal to k, but is zero otherwise. Note that summing these $M - 1$ binary signals always provides the original input signal; this can be seen in Fig. 3.

2) Filtering each binary signal independently with its own rank-order filter. These operations may all be performed in parallel as shown in Fig. 3. Also, each of the filters, since they have only binary signals at their inputs, are trivial to implement. Each one simply adds the number of bits in the window and compares the result to an integer r; outputting a zero if the sum is less than r, a 1 if the sum is greater than or equal to r. Each filter thus produces a binary output signal. To provide an example, if the filter's window is $b = 2r + 1$ bits wide, the filter is a median filter.

3) Adding the outputs of the binary ranked order operator one sample at a time. This addition operation is shown in Fig. 3.

The new architecture of the median filter suggested by Fig. 3 appears perfect for VLSI implementation. It is highly parallel and the thresholding operation and binary rank-order filters are both very easy to implement. However, there is a major problem in implementing the reconstruction section. As mentioned in step 3 above, this section must add together the outputs of all the binary rank-order operators. Unfortunately, if more than a few of these signals must be added together, the adder which does it might require an entire chip by itself. For instance, if the original input signal consisted of 8-bit data, there would be 255 binary signals to be added together in the reconstruction section. It thus seems that a VLSI implementation of this architecture is not very feasible.

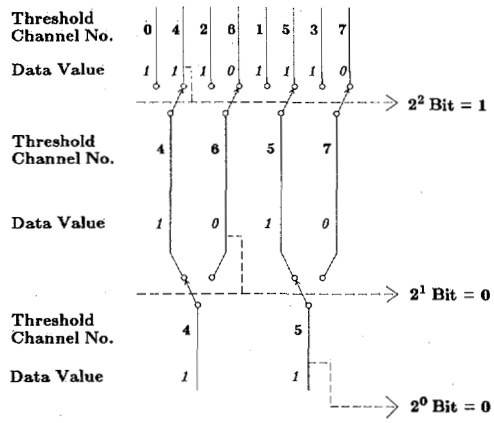


Fig. 4. Efficient reconstruction section [29]. Three-bit input data take on the values 0–7, so there are eight binary median filters. The binary output of these filters can be efficiently added by exploiting the stacking property, which says that the output value is equal to the highest threshold value which produced a binary threshold signal that was filtered to a 1 in the current window position of the filter. In the example shown above, it was the binary filter operating on data produced by thresholding the input at level 5 that produced the highest 1. The switches shown in the figure are *ganged*. This means that the set of switches controlled by the output of threshold level 4 are all thrown to the left if level 4's output is a 0; otherwise, they are all thrown to the right. The same is true of the bottom two switches, which are controlled by the output of threshold level 6.

This problem is solved by noting that the outputs of the binary ranked-order operators possess the *stacking property* [12], [13]. This says that if the binary output signals are piled on top of each other according their threshold level, the result is always a column of 1's with a column of 0's on top (see Fig. 3, for an example). The desired output is then the level just before the transition from 1 to 0 takes place. We find this level by performing a binary search on the binary filter outputs; in this way, we can quickly reconstruct the output one bit at a time, starting with the most significant bit. In fact, if the threshold levels are arranged on a silicon chip in bit-reversed order, this section can be constructed in a way which takes up very little chip area and has very few crossed metal paths [29]. This can be seen in Fig. 4, where we show the reconstruction section that would be used on a chip designed to filter 3-bit data.

Thus, by simplifying the reconstruction section, the stacking property of ranked-order operators makes possible a VLSI implementation of the threshold decomposition architecture for these filters. The VLSI layout of a rank-order filtering chip with this architecture is shown in Fig. 5. A chip with this architecture will soon enter the manufacturing stage; it will be fabricated in 5-μm NMOS technology. When completed, it will be more flexible and faster than previous implementations at this resolution.

Because of the success of this implementation—its sim-

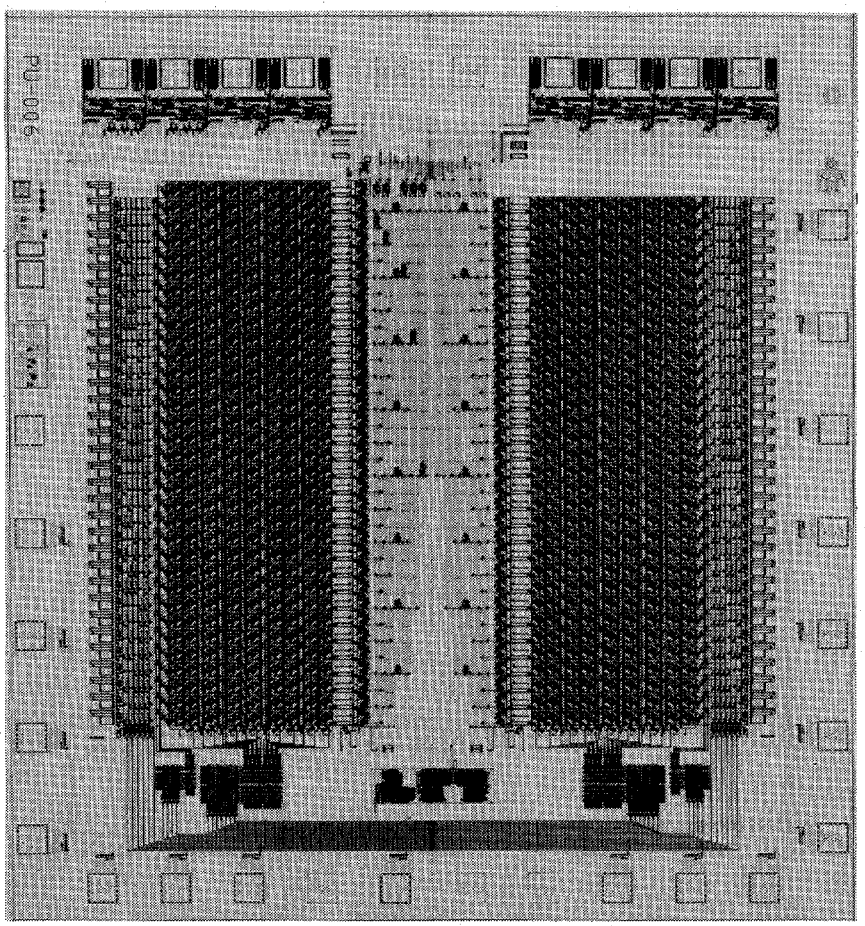


Fig. 5. VLSI layout of a rank-order filter chip [29]. This chip can perform any rank-order operation of window widths 3, 5, 7, or 9 on 6-bit input data. Its architecture is the threshold decomposition architecture shown in Fig. 3. The reconstruction section is the ornate column in the middle of the chip. Since the input signal has 6-bits, there are 63 binary rank-order filters on the chip; 32 in the stack on the left of the reconstruction section, 31 on the right.

ple architecture and the resulting fast operation—it is natural to ask what other filters have both this architecture and the stacking property. In the following section, we show that these filters are all those that can be obtained by replacing the binary median filters in Fig. 3 with positive Boolean functions. Since the stacking property is common to all these filters, we call them *stack filters*.

III. Positive Boolean Functions and Stack Filters

As mentioned in the Introduction, we want to characterize stack filters, which are all filters possessing both the stacking property and the architecture shown in Fig. 3. They are obtained by replacing each of the binary median filters in this figure with other binary filters. This binary filter or function must possess the stacking property discussed previously. Thus, our goal in this section is to determine which binary functions have this property.

Before we can do this, we must develop some notation and give the formal definitions of the threshold decomposition and the stacking property. This is done in Subsection A. In Subsection B, we give the condition which characterizes all binary functions satisfying the stacking property and discuss some of the properties of these functions.

A. Notation and Definitions

Vector notation will be used for sequences, so that an input signal s of length n will be written $\vec{s}_n = (s_1, s_2, \cdots, s_n)$. If the sequence $\vec{s}_n$ is a binary sequence, it will be marked as $\vec{s}_{b,n}$ to make this clear. When the value of n is clear from the context, it will be dropped.

As for relations between sequences, suppose $\vec{x}_{b,n} = (x_1, x_2, \cdots, x_n)$ and $\vec{y}_{b,n} = (y_1, y_2, \cdots, y_n)$ are two binary sequences of length n. We say that $\vec{x}_{b,n} = \vec{y}_{b,n}$ if and only if $x_i = y_i$ for all i. We say that $\vec{x}_{b,n} \leq \vec{y}_{b,n}$ if $x_i = 1$ implies $y_1 = 1$ for all i; if we also have $\vec{x}_{b,n} \neq \vec{y}_{b,n}$, we write $\vec{x}_{b,n} < \vec{y}_{b,n}$.

We also need some notation for the functions we will be considering. Because of the threshold decomposition, we will be interested mainly in binary filters; clearly a binary filter is essentially a Boolean logic function operating on the bits in the filter's window. These functions will always be denoted by italic capitals, and we will sometimes include the number of variables in the function

name. Thus, a binary function B with n inputs can be denoted Bn. If the binary function Bn is applied to the sequence $\vec{x}_{b,n}$, then the output is denoted by either $Bn(\vec{x}_{b,n})$ or $Bn(x_1, x_2, \cdots, x_n)$ depending on which is more appropriate at the time. We also warn the reader now that since one of the n's is redundant, we will often use only one.

As an example illustrating this notation for functions, consider the window width 3 median filter. If the input to this filter is restricted to binary values, it becomes a binary function which we denote $MF3$. If this filter is applied to the sequence $\vec{s}_b = (1, 0, 0)$, it produces a zero; that is, $MF3(1, 0, 0) = MF3(\vec{s}_b) = 0$. Similarly, $MF3(1, 0, 1) = 1$, while $MF3(0, 1, 0) = 0$. In general, the output of $MF3$ is a 1 if and only if at least two of the input bits are 1.

We could represent such a binary filter as a truth table; however, if we can deduce a rule defining the output of the filter, we can represent the filter more compactly as a Boolean expression. In such an expression, we represent the OR operation by addition, the AND operation by multiplication, and the COMPLEMENT of a variable x by $\bar{x}$. For example, the rule for $MF3$ implies that

$$MF3(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_1x_3. \tag{1}$$

A Boolean expression such as the one above, which represents the ORing of several AND operations, is called a *sum-of-products* expression [18]; each group of variables representing an AND operation is called a *product term*, or simply a *term*, of the expression. If the number of terms in the sum-of products expression is the minimum possible for a particular binary function, the expression is called a *minimum sum-of-products* (MSP) expression.

Finally, we need some notation for the overall filter that is implemented by placing the binary operator B at each threshold level in Fig. 6. This filter is called the stack filter S_B if B obeys the stacking property that will be defined below. For example, if each binary filter in Fig. 6 is the window width 3 binary median filter $MF3$, the whole filter is the stack filter S_{MF3}.

With all the above notation, we can now define more rigorously the threshold decomposition and stacking property discussed in the Introduction and shown in Fig. 3.

Definition: The threshold decomposition of an M-valued sequence $\vec{s}_N$ is the set of $M - 1$ binary sequences, called threshold signals, $\vec{T}1_b, \vec{T}2_b, \cdots, T(\vec{M} - 1)_b$, whose elements are defined by

$$\vec{T}i_b(j) = \begin{cases} 1 & \text{if } \vec{s}(j) \geq i \\ 0 & \text{if } \vec{s}(j) < i. \end{cases} \tag{2}$$

For an example showing a set of threshold signals and the signal they came from, see the left-hand side of Fig. 3 or Fig. 6. Note that threshold signals always obey the stacking property, which is the following.

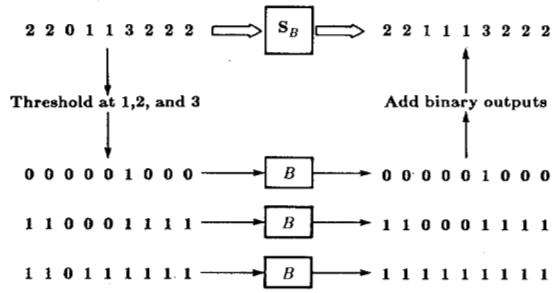


Fig. 6. A stack filter—the window width 3 asymmetric median filter. The input signal in the upper left is filtered by the stack filter S_B we call the asymmetric median filter by following the heavy arrows. The actual operations performed by S_B can be seen by following the slender arrows. The positive Boolean function which creates the above filter is

$$B(x_1, x_2, x_3) = x_1x_3 + x_2$$

in which x_i, $i = 1, 2, 3$ are the bits, in time order, appearing in the filter's window. This filter preserves edges, removes negative-going impulses (22011)→(22111), but preserves positive-going impulses (11322)→(11322).

Definition: An ordered set of L binary sequences $\vec{S}1_b, \vec{S}2_b, \cdots, \vec{S}L_b$, in which all the sequences have length n, is said to obey the stacking property if

$$\vec{S}1_b \geq \vec{S}2_b \geq \cdots \geq \vec{S}L_b. \tag{3}$$

In the Introduction, it was stated that the outputs of the binary ranked-order operators in Fig. 3 also have the stacking property. This was proven in [12] and [13]. This leads to a definition of the stacking property for binary functions as well as ordered sets of sequences

Definition: A window width n binary function Bn is said to have the stacking property if

$$Bn(\vec{x}) \geq Bn(\vec{y}) \quad \text{whenever} \quad \vec{x} \geq \vec{y}. \tag{4}$$

Recall that it is because the binary ranked-order filters have this stacking property that the reconstruction technique in Fig. 4 can be used. This reconstruction technique is what made the new VLSI implementation of these filters possible.

We are now ready to characterize the class of stackable binary operators.

B. Characterization and Elementary Properties of Stackable Binary Operators

In this section, we state a necessary and sufficient condition for a binary function to satisfy the stacking property that was found by Gilbert [19]. His interest in this property came from certain problems in digital design, but the result carries over directly to our signal processing problem. We also state some other properties of stackable binary functions. It will be seen that there are a large number of these functions, and therefore a large number of stack filters.

Theorem 1 [19]: An n-input binary function Bn will have the stacking property

$$Bn(x_1, x_2, \cdots, x_n) \geq Bn(y_1, y_2, \cdots, y_n) \quad \text{when} \quad x_i \geq y_i \ \forall i$$

if and only if Bn can be expressed as a Boolean expression which contains no complements of the input variables. □

Gilbert [19] called functions satisfying this property *frontal functions*; elsewhere they are called *positive Boolean functions*.

Applying the theorem to the operator $MF3$ in (1), we see that $MF3$ has the stacking property, whereas the function $AI3(x_1, x_2, x_3) = x_1x_2x_3$ does not since it uses complementing. Thus, a stack filter with a VLSI implementation can be created from $MF3$ but not from $AI3$.

Note that the condition which guarantees stacking is not very restrictive. As a result there are many positive Boolean functions, and since each such function defines a stack filter after it is inserted in the architecture of Fig. 3, there are just as many stack filters. In fact, there are exactly 20 window width 3 stack filters, 7581 with window width 5, and more than 2^{35} with window width 7 [19]. The exact number of these functions for any arbitrary window width is not known at this time.

The binary functions which stack—the positive Boolean functions—are a subset of unate functions [18], [20]. A unate function is any binary function that can be obtained by complementing some of the input variables of a positive Boolean function. Unate functions have the interesting property that each has a unique minimum sum-of-products (MSP) expression [18], [20]; this does not hold for all Boolean functions. Thus, each positive Boolean function, and hence, each stack filter, can be expressed in terms of a unique minimum sum-of-products Boolean expression. The unique minimum expression for the median filter of window width 3 was provided in (1).

For some positive functions, another representation is useful. As mentioned in the Introduction, the binary filters in a threshold decomposition implementation of a ranked-order filter are essentially sum-and-threshold operators. These binary ranked-order operators are particular cases of *threshold functions* [18], [20].

Definition: The n-input threshold function $T(x_1, x_2, \cdots, x_n)$ with weights $a_1, a_2, \cdots, a_n$ and threshold c is defined by

$$T(x_1, x_2, \cdots, x_n) = 1 \Leftrightarrow \sum_{i=1}^{n} a_i x_i \geq c. \tag{5}$$

If $\vec{x}_b = (x_1, x_2, \cdots, x_n)$ and $\vec{a} = (a_1, a_2, \cdots, a_n)$, we can rewrite this as

$$T(\vec{x}_b) = 1 \Leftrightarrow \vec{a} \cdot \vec{x}_b \geq c. \tag{6}$$

If $\vec{x}_b \leq \vec{y}_b$ and $a_1, a_2, \cdots, a_n \geq 0$, then $\vec{a} \cdot \vec{y}_b \geq \vec{a} \cdot \vec{x}_b$, so $\vec{a} \cdot \vec{x}_b \geq c \Rightarrow \vec{a} \cdot \vec{y}_b \geq c$. Hence, $T(\vec{x}_b) = 1 \Rightarrow T(\vec{y}_b) = 1$. Therefore, if a threshold function can be defined by nonnegative weights and a nonnegative threshold, it is a positive Boolean function.

As the rational numbers are dense in the real numbers, we can always replace the coefficients of a threshold function by arbitrarily close rational numbers, and still obtain the same Boolean function. By multiplying these rational coefficients by their least common denominator, we obtain an equivalent set of integer coefficients.

Suppose that a threshold function $T(\vec{x}_{b,n})$ has integer weights a_i and integer threshold k. If we replace the ith input to the function by a_i connected inputs, each with unit weight, we obtain the same Boolean function. Therefore, the stack filter S_T is a generalized ranked-order filter defined by

$$S_T(s_1, s_2, \cdots, s_n) = k\text{th largest element of the set } \{s_1, \cdots, s_1, s_2, \cdots, s_2, \cdots, s_n, \cdots, s_n\} \tag{7}$$

where s_i, the ith sample in the window, is repeated a_i times.

This is similar to a generalization of the median filter suggested by Nodes [9]. In particular, if $a_i = 1$ for all i, S_T defines an $(n - k)$th-order filter; if n is odd, and $k = \lfloor n/2 \rfloor$, S_T defines a median filter.

Note that only a threshold function that can be defined by positive weights and a positive threshold is positive; any negative weight corresponds to a complemented variable in the MSP expression for the threshold function. However, it is true that all threshold functions are unate.

Also, not every positive function is a threshold function. All positive functions of three variables are threshold functions; however, of the 7581 positive functions of five variables or less, only 3287 are threshold functions [20]. This means that there are many stack filters whose outputs cannot be expressed as simple functions of the input samples.

There are two basic operations that will map one positive Boolean function to another positive function. First, if we permute the input variables to a positive function the new function is clearly positive; in particular, this is true if we reverse the order of the variables. Second, if we complement all the inputs to a positive function B, and then complement the total function, we obtain a new function called the *dual* of B, which we denote B^d. This operation is equivalent to replacing AND operations by OR operations, and vice versa, in an AND-OR expression for B; hence, if B is positive, B^d is positive. In particular, the dual of a positive threshold function is another positive threshold function.

These operations correspond to basic transformations of stack filters. Obviously, reversing the order of the variables in a positive function B reverses the time sense of the stack filter S_B. Replacing B in a threshold decomposition by its dual function will reverse the sense of the M levels; if the original filter produces an output sequence $\{z(t)\}$ for an input $\{s(t)\}$, then the new filter will produce an output $\{M - 1 - z(t)\}$ for an input $\{M - 1 - s(t)\}$.

In this section, we have stated a necessary and sufficient condition, due to Gilbert [19], for a binary function to possess the stacking property. Such functions are called positive Boolean functions, and a stack filter consists of a "stack" of them, as shown in Fig. 6. Because of the large number of positive Boolean functions, there are a large number of stack filters. The threshold functions with

nonnegative coefficients, which form the generalized rank-order operators are also stack filters. Also, since each positive Boolean function, and therefore, the stack filter it defines, has a unique minimum sum-of-products (MSP) Boolean expression, each stack filter has a simple, unique functional definition. Finally, we reviewed two ways of producing new positive functions from a given positive function; we will use these mappings later in a classification of the stack filters of window width 3.

IV. Deterministic Properties of Stack Filters

Since median and ranked-order filters are nonlinear, we cannot reduce the analysis of any of these filters by considering the filtering of an input signal and the filtering of noise separately. Instead, we must approach the problem syntactically, and ask which features of a noisy input signal a filter preserves, and which it eliminates.

For a median filter, this approach led to the idea of a root [5], [8]–[11] which is any signal that is invariant to filtering by that filter. Essentially, the roots of a median filter are the "passband" signals of the filter. Furthermore, just as a linear filter can estimate a signal within its passband that is corrupted by noise outside the passband, so a median filter can estimate any root buried in noise. In fact, any signal of finite length will converge to a root in a finite number of passes of a median filter; the resultant root may not be the same as the original uncorrupted root, but it will be close.

As stack filters are generalizations of median filters, we can analyze them in a similar manner. Here, we derive conditions for the existence of invariant signals, and for the convergence of arbitrary signals to invariant signals under repeated passes of a stack filter. As the structure of median filter roots is well known, our results focus on other filters that preserve these signals.

Because of the threshold decomposition, we need only consider binary filters. Hence, each basic result will be in terms of strings of zeros and ones. As a Boolean expression is a convenient way to represent a binary filter, we will restate each basic result in terms of these expressions; in a couple of cases, we give corollaries in terms of threshold functions as well.

Before we present these results, we rigorously define root signals, and introduce some other basic concepts and notation that we will need in this section.

Definition: Let $\vec{s} = (s(1), s(2), \cdots, s(L))$ be an M-level signal of length L. Then, for a filter of window width $2N + 1$, the corresponding *appended signal* is $\vec{s} = (s(1), \cdots, s(1), s(2), \cdots, s(L-1), s(L), \cdots, s(L))$, where $s(1)$ and $s(L)$ are repeated $N + 1$ times.

If a signal has finite length, we need to append samples to the ends of the signal so that the output of a stack filter is defined for all positions of the window on the signal; if there were no appended samples, the filter window would overshoot the signal when centered on samples near the ends.

Definition: The *roots* of a stack filter S_B are all the appended signals that are invariant under filtering by S_B. The roots of a median filter of window width $2N + 1$ are all the appended signals consisting of monotonic regions alternating with constant regions of at least $N + 1$ samples.

Definition: A signal is *locally monotone* with respect to a window of width $2N + 1$ if every sequence of $2N + 1$ consecutive samples in the signal is monotonic.

For example, the roots of a median filter of window width $4N + 1$ appear locally monotone to a window of width $2N + 1$.

Notation: To simplify our representation of binary sequences, we shall use 0^m and 1^m to denote 0 or 1 repeated m times.

In fact, every stack filter must have some roots, as every stack filter preserves constant signals. If a positive Boolean function Bn is not identically 0 or 1, then $Bn(0^n) = 0$ and $Bn(1^n) = 1$. Hence, the binary filter based on Bn will preserve all constant binary signals and have the following result.

Theorem 2: Suppose that S_B is an M-level stack filter, based on a nontrivial positive Boolean function B. Then S_B will preserve all constant signals. □

If B is identically 0, then S_B will preserve only constant zero-valued signals, whereas if B is identically 1, S_B will only preserve signals of value $M - 1$.

To find some nontrivial roots of stack filters, we start by finding which stack filters preserve increasing signals. Then, it is simple to find those that preserve decreasing signals, and those that preserve median filter roots.

Proofs of the more complex results that follow are in the Appendix.

Theorem 3: Suppose that an M-level stack filter S_B, based on a positive Boolean function B, has window width $2N + 1$. Then, S_B preserves all increasing (nondecreasing) signals if and only if $B(0^N 1^{N+1}) = 1$ and $B(0^{N+1}1^N) = 0$. □

We mentioned before that many binary filters are better represented by Boolean expressions than by truth tables. Hence, a restatement of theorem 3 in terms of such expressions is valuable.

Corollary 1: Let S_B and B be as in theorem 3. Then S_B preserves all increasing M-level signals if and only if

a) every term in the MSP form of $B(x_1, x_2, \cdots, x_{2N+1})$ that contains only variables from $\{x_{N+1}, x_{N+2}, \cdots, x_{2N+1}\}$ must contain x_{N+1}; and

b) there exists at least one such term. □

For example, for a window width of 3, the stack filter based on $B(x_1, x_2, x_3) = x_1 + x_2x_3$ will preserve increasing signals.

As mentioned before, generalized ranked-order filters decompose into threshold functions. The next corollary follows immediately from the conditions of theorem 3 and the definition of a threshold function.

Corollary 2: Let S_T be an M-level stack filter of window width $2N + 1$, based on a threshold function T with weights $a_1, a_2, \cdots, a_{2N+1} \geq 0$ and threshold $c \geq 0$. Then S_T preserves all increasing M-level signals if and only if $\sum_{i=N+2}^{2N+1} a_i < c \leq \sum_{i=N+1}^{2N+1} a_i$. □

By reversing the order of the variables in the previous

three results, we obtain similar results about filters that preserve decreasing signals.

From the definition of the roots of a median filter, we see that the roots of a median filter of window width $4N + 1$ will appear locally monotone to a window of width $2N + 1$. Hence, the window sequences for the median filter roots are exactly those for increasing and decreasing signals, and we have the following result.

Theorem 4: Let S_B be an M-level stack filter of window width $2N + 1$. Then S_B preserves the roots of a median filter of window width $4N + 1$ if and only if S_B preserves all M-level monotonic signals. □

Using the preceding results, we can classify all stack filters of any window width according to the locally monotone signals that they preserve.

A stack filter may preserve other signals that are not locally monotone. For example, many of the roots of a median filter of window width $2N + 1$ are not locally monotone with respect to that window size. Also, the previous theorems do not address the convergence properties of stack filters; they do not guarantee that any appended signal will converge to an invariant signal under repeated filtering by a stack filter.

Note that if a stack filter of window width $2N + 1$ preserves the roots of a median filter with a window width greater than $4N + 1$, it must preserve the roots of a median filter of window width $4N + 1$; the possible window sequences are the same for both sets of roots. Also, if a stack filter preserves the roots of a median filter of window width less than $4N + 1$, it must preserve all median filter roots for larger window widths, including $4N + 1$. Therefore, if a stack filter preserves the roots of any median filter, it must satisfy theorem 4.

We now consider whether or not such a filter preserves the roots of median filters with window widths less than $2N + 1$.

Theorem 5: Suppose that a stack filter S_B of window width $2N + 1$ is based on a positive Boolean function B. Then, S_B preserves the roots of a median filter of window width $2N' + 1$ $(0 \le N' \le N)$ if and only if B satisfies the conditions

a) $B(0^m 1^{N'+1} 0^{2N-N'-m}) = 1$ $N - N' \le m \le N$,

b) $B(1^{mN'-1} 1^{2n-N'-m}) = 0$ $N - N' \le m \le N$. □

Essentially, the conditions of the theorem guarantee that each binary filter in S_B will preserve all constant regions of at least $N' + 1$ ones or zeros. Hence, each binary filter will preserve all binary roots of the median filter of window width $2N' + 1$, and S_B preserves all M-level roots of the same filter.

Given a positive Boolean function of n variables $B(\vec{x}_{b,n})$, and a binary window sequence $\vec{y}_{b,n}$, $B(\vec{y}_{b,n}) = 1$ if and only if the assignment $x_i = y_i$ $i = 1, 2, \cdots, n$ causes at least one term in the MSP expression of B to be 1. Hence, we can obtain a version of theorem 5 in terms of a Boolean expression.

Corollary 3: Let S_B and B be as above, and $0 \le N' \le N$. Then S_B preserves the roots of a median filter of window width $2N' + 1$ if and only if

a) $\forall m$ such that $N - N' \le m \le N$, there is a term in the MSP form of B that depends only on variables from $\{x_{m+1}, x_{m+2}, \cdots, x_{m+N'+1}\}$; and

b) $\forall m$ such that $N - N' \le m \le N$, there is no term in the MSP form of B that depends only on variables from the set $\{x_1, x_2, \cdots, x_m\} \cup \{x_{m+N'+2}, x_{m+N'+3}, \cdots, x_{2N-N'-m}\}$. □

We can also derive a version of theorem 5 in terms of threshold functions.

Corollary 4: Let S_T be a stack filter of window width $2N + 1$, based on a threshold function $T(x_1, x_2, \cdots, x_{2N+1})$ with weights $a_1, a_2, \cdots, a_{2N+1} \ge 0$ and threshold $c \ge 0$. Then, S_T preserves the roots of a median filter of window width $2N' + 1$, $(0 \le N' < N)$ if and only if

$$\sum_{i=m+1}^{m+N'+1} a_i \ge c' \qquad N - N' \le m \le N$$

where

$$c' = \max\left(c, \sum_{i=1}^{2N+1} a_i - c - 1\right). \qquad \square$$

This corollary follows by expressing the two conditions of theorem 5 in terms of a threshold function. This gives two inequalities in terms of the coefficients of the threshold hold function; the constant c' arises because there are two inequalities.

Finally, we consider the convergence properties of stack filters. In particular, we have found a sufficient condition for every appended input to a stack filter to converge to a median filter root in a finite number of passes of the filter. As mentioned before, this is a stronger result than the previous theorems about the preservation of median filter roots; it is possible that, for some input signal, the output of a stack filter will eventually cycle through a set of signals ad infinitum, and never converge. It is also possible that the filter has roots other than those of a median filter.

Theorem 6: Let S_B be an M-level stack filter of window width $2N + 1$, based on a positive Boolean function B. Then every appended M-level signal will converge to a root of a median filter of window width $2N + 1$ after a finite number of passes of S_B if

a) $B(0^N 1^{N+1}) = 1$, but for $\vec{x}_b < 0^N 1^{N+1}$, $B(\vec{x}_b) = 0$;

b) $B(1^N 0^{N+1}) = 0$, but for $\vec{x}_b > 1^N 0^{N+1}$, $B(\vec{x}_b) = 1$;

c) $B(1^{N+1} 0^N) = 1$, but for $\vec{x}_b < 1^{N+1} 0^N$, $B(\vec{x}_b) = 0$; and

d) $B(0^{N+1} 1^N) = 0$, but for $\vec{x}_b > 0^{N+1} 1^N$, $B(\vec{x}_b) = 1$. □

The four conditions guarantee that the constant regions at the ends of the appended signal will grow into rootlike regions on repeated passes of the filter, and that each rootlike region must grow inward by at least one sample position on each pass. When the two regions meet, the whole signal is a root of the median filter.

Immediately, we have a limit on the number of passes to a root for these filters; the limit is the same as that of Gallagher and Wise for median filters [6].

Corollary 5: Let S_B be a stack filter of window width $2N + 1$ that satisfies the conditions of the previous theorem. Then, any signal of unappended length L will, when appended, converge to a root of a median filter of window width $2N + 1$ in at most $\lceil (L - 2)/2 \rceil$ passes of S_B. □

Finally, we obtain a version of theorem 6 in terms of the MSP form of the Boolean function B.

Corollary 6: Let S_B be a stack filter of window width $2N + 1$, based on a positive Boolean function B. Then every appended M-level signal will converge to a root of a median filter of window width $2N + 1$ after a finite number of passes of S_B if

1) $x_1 x_2 \cdots x_{N+1}$ is a term in the MSP form of B;

2) $x_{N+1} x_{N+2} \cdots x_{2N+1}$ is a term in the MSP form of B;

3) for each $i = 1, 2, \cdots, N + 1$, there is a term in the MSP form of B that contains only x_i and variables from $\{x_{N+2}, x_{N+3}, \cdots, x_{2N+1}\}$; and

4) for each $i = N + 1, N + 2, \cdots, 2N + 1$, there is a term in the MSP form of B that contains only x_i and variables from $\{x_1, x_2, \cdots, x_N\}$. □

In fact, the median filter is not the only filter that satisfies theorem 6 or corollary 6. For $N > 1$, the positive function

$$\begin{aligned} B(x_1, \cdots, x_{2N+1}) = {} & (x_1 x_2 \cdots x_N) \\ & \cdot (x_{N+1} + x_2 + \cdots + x_{2N+1}) \\ & + (x_{N+2} x_{N+3} \cdots x_{2N+1}) \\ & \cdot (x_1 + x_2 + \cdots + x_{N+1}) \end{aligned} \tag{8}$$

defines another filter that satisfies the conditions of corollary 6. It is easy to see that there are many other such filters. (For $N = 1$, the only such filter is the median filter.) All these filters have exactly the same roots and, for each filter, every signal converges to a root; thus, these filters can differ only in the rate at which they filter the signal to the root.

V. Statistical Properties of Stack Filters

As mentioned before, for ranked-order filters, we cannot reduce the analysis of the filtering of a signal plus noise to two separate subproblems for signal and noise. This is also true for stack filters. So, to analyze the noise reduction offered by a stack filter, we must compute the distribution function for each filtered output sample.

Several authors [14], [21], [22] have considered the distribution of the output of a median or ranked-order filter, whose input is i.i.d. noise, or i.i.d. noise plus a known signal. Here, we present two methods to calculate the distribution of the output of a general stack filter with similar inputs.

Let S_B be a stack filter with window width $2N + 1$, based on a positive Boolean function B. If the number of levels M is large, we can assume a continuous range of input values. This allows the analysis to proceed more smoothly, because we can then use the distribution function F rather than a set of multinomial probabilities. The same assumption has been made by previous authors [21].

Suppose that input to S_B is a known signal plus i.i.d. noise, i.e.,

$$v_i(\omega) = s_i + n_i(\omega) \tag{9}$$

where $n_i(\omega)$ is an i.i.d. noise sequence defined $\forall i$, $i = 1, 2, \cdots$, and for all outcomes ω in the underlying sample space. We also assume that the noise has distribution function $F(\cdot)$.

Suppose the MSP form of B is

$$B(x_1, x_2, \cdots, x_{2N+1}) = \Pi_1 + \Pi_2 + \cdots + \Pi_m \tag{10}$$

where

$$\Pi_p = x_{j(p,1)} x_{j(p,2)} \cdots x_{j(p,k(\Pi_p))} \tag{11}$$

is the pth product term in the MSP expression, and the $j(p, q)$ are the indexes of the variables in Π_p in increasing order; also, if Π is any single product of the x_j, let $k(\Pi)$ be the number of variables in Π.

Let $z_i(\omega)$ be the output sequence generated by the stack filter for the input $v_i(\omega)$ and suppose that its first-order distribution is $F_z(z:i) = P(\{\omega : z_i(\omega) \le z\})$. To calculate $F_z(z:i)$, we consider z to be a threshold in the stack filter S_B. The binary sequence created by the thresholding operation at level z is then given by

$$x_j(\omega) = \begin{cases} 1 & \text{if } v_{i+j-N-1}(\omega) \ge z \\ 0 & \text{else} \end{cases} \tag{12}$$

and the output at position i of the binary filter B acting on this threshold sequence is a 1 if and only if the output for the stack filter S_B is greater than or equal to the threshold level z at this position. In other words,

$$B(x_1, x_2, \cdots, x_{2N+1}) = \begin{cases} 1 & \text{if } z_i \ge z \\ 0 & \text{else} \end{cases}. \tag{13}$$

This implies that

$$\begin{aligned} 1 - F_z(z:i) &= P(B(x_1, x_2, \cdots, x_{2N+1}) = 1) \\ &= P(\text{at least one } \Pi_i \text{ in } (10) = 1) \\ &= P(x_{j(p,1)} = x_{j(p,2)} = \cdots \\ &= x_{j(p,k(\Pi_p))} = 1 \quad \text{for some } p) \\ &= P\left(\bigcup_{p=1}^{m} \bigcap_{l=1}^{k(\Pi_p)} \{\omega : v_{j(p,l)+i-N-1}(\omega) \ge z\} \right). \end{aligned} \tag{14}$$

If we denote the intersection of events corresponding to the term Π_p by A_p, then we want $P(\cup_{p=1}^{m} A_p)$. In calculating probability, we must count every intersection of the A_i only once; we do this by successively adding or subtracting the probability associated with each possible intersection.

Also, we know that $P(\{\omega : v_l(\omega) \ge z\}) = 1 - F(z - s_l)$. Therefore, the probability of the event $\Pi_{p_1} = \Pi_{p_2} = \cdots = \Pi_{p_k} = 1$ is

$$\prod_{x_j \in \Pi_{p_1}\Pi_{p_2}\cdots\Pi_{p_k}} (1 - F(z - s_{i+j-N-1}))$$

where $x_j \in \Pi$ means that the arithmetic product is taken over all indexes j for which x_j is a variable in the Boolean product Π. If Π is a product of smaller products, this is equivalent to saying that x_j is a variable in at least one of these smaller products.

Finally, we sum over all possible sets of indexes p_i of product terms, and obtain

$$\begin{aligned} F_z(z:i) = 1 &- \sum_{p=1}^{m} \prod_{l=1}^{k(\Pi_p)} (1 - F(z - s_{j(p,l)+i-N-1}) \\ &+ \sum_{\{p_1,p_2\}} \prod_{x_j \in \Pi_{p_1}\Pi_{p_2}} (1 - F(z - s_{i+j-N-1})) \cdots \\ &+ (-1)^k \sum_{\{p_1,p_2,\cdots,p_k\}} \prod_{x_j \in \Pi_{p_1}\Pi_{p_2}\cdots\Pi_{p_k}} \\ &\cdot (1 - F(z - s_{i+j-N-1})) \cdots \\ &+ (-1)^m \prod_{x_j \in \Pi_1\Pi_2\cdots\Pi_m} (1 - F(z - s_{i+j-N-1})). \end{aligned} \tag{15}$$

If $s_i \equiv 0$, we obtain a result for i.i.d. inputs

$$\begin{aligned} F_z(z) = 1 &- \sum_{p=1}^{m} (1 - F(z))^{k(\Pi_p)} \\ &+ \sum_{\{p_1,p_2\}} (1 - F(z))^{k(\Pi_{p_1}\Pi_{p_2})} \cdots \\ &+ (-1)^k \\ &\sum_{\{p_1,p_2,\cdots,p_k\}} (1 - F(z))^{k(\Pi_{p_1}\Pi_{p_2}\cdots\Pi_{p_k})} \cdots \\ &+ (-1)^m (1 - F(z))^{k(\Pi_1\Pi_2\cdots\Pi_m)}. \end{aligned} \tag{16}$$

As an example, we consider the median filter of window width 3, which is based on the Boolean function $x_1x_2 + x_1x_3 + x_2x_3$. If we let $\Pi_1 = x_1x_2$, $\Pi_2 = x_1x_3$, and $\Pi_3 = x_2x_3$, then $\Pi_i\Pi_j = \Pi_1\Pi_2\Pi_3 = x_1x_2x_3$ for $i, j = 1, 2, 3$, $i \neq j$. Therefore, $k(\Pi_i) = 2$ and $k(\Pi_i\Pi_j) = k(\Pi_1\Pi_2\Pi_3) = 3$ otherwise. Using (15), we find that, for an i.i.d. input,

$$\begin{aligned} F_z(z) &= 1 - 3(1 - F(z))^2 + 3(1 - F(z))^3 - (1 - F(z))^3 \\ &= 1 - 3(1 - F(z))^2 + 2(1 - F(z))^3 \\ &= 1 - 3 + 6(F(z)) - 3(F(z))^2 + 2 \\ &\quad - 6F(z) + 6(F(z))^2 - 2(F(z))^3 \\ &= 3(F(z))^2 - 2(F(z))^3. \end{aligned} \tag{17}$$

For the median filter of window width 3, we also know that

$$\begin{aligned} F_z(z) &= P(\text{at least 2 samples in window} \leq z) \\ &= P(\text{exactly 2 samples in window} \leq z) \\ &\quad + P(\text{all samples in the window} \leq z) \\ &= 3(F(z))^2 (1 - F(z)) + (F(z))^3 \\ &= 3(F(z))^2 - 3(F(z))^3 + (F(z))^3 \\ &= 3(F(z))^2 - 2(F(z))^3. \end{aligned} \tag{18}$$

This is the same as the previous result. Alternatively, we can consider that

$$F_z(z:i) = P(B(x_1, x_2, \cdots, x_{2N+1}) = 0) \tag{19}$$

$$= P(B^d(\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_{2N+1}) = 1). \tag{20}$$

We then use the MSP form of the dual function B^d

$$B^d(y_1, y_2, \cdots, y_{2N+1}) = \Phi_1 + \Phi_2 + \cdots + \Phi_n \tag{21}$$

where the Φ_i are the product terms in the MSP expression. Using the same notation as before

$$\begin{aligned} F_z(z:i) = &\sum_{p=1}^{n} \prod_{l=1}^{k(\Phi_p)} F(z - s_{j(p,l)+i+N-1}) \\ &- \sum_{\{p_1,p_2\}} \prod_{x_j \in \Phi_{p_1}\Phi_{p_2}} F(z - s_{i+j-N-1}) \cdots \\ &+ (-1)^{k+1} \sum_{\{p_1,p_2,\cdots,p_k\}} \prod_{x_j \in \Phi_{p_1}\Phi_{p_2}\cdots\Phi_{p_k}} \\ &\cdot F(z - s_{i+j-N-1}) \cdots \\ &+ (-1)^{n+1} \prod_{x_j \in \Phi_1\Phi_2\cdots\Phi_n} F(z - s_{i+j-N-1}). \end{aligned} \tag{22}$$

For i.i.d. inputs

$$\begin{aligned} F_z(z) = &\sum_{p=1}^{n} (F(z))^{k(\Phi_p)} \\ &- \sum_{\{p_1,p_2\}} (F(z))^{k(\Phi_{p_1}\Phi_{p_2})} \cdots \\ &+ (-1)^{k+1} \\ &\sum_{\{p_1,p_2,\cdots,p_k\}} (F(z))^{k(\Phi_{p_1}\Phi_{p_2}\cdots\Phi_{p_n})} \cdots \\ &+ (-1)^{n+1} (F(z))^{k(\Phi_1\Phi_2\cdots\Phi_n)}. \end{aligned} \tag{23}$$

For a stack filter which is based on a threshold function T, it is probably simpler to express T in its MSP form, and then use the previous expressions for the output distribution, than it would be to derive a closed-form expression in terms of the weights and threshold of T.

VI. Examples and Applications—Window Width 3

There are exactly 20 positive functions of 3 variables, 7581 of 5 variables, and an unknown but very large number of positive functions of 7 variables [19]; therefore, it is only practical to enumerate the stack filters with window width 3. We list the positive functions according to the roots of the filters they produce, and discuss the filters' convergence properties. Among these filters, there are several that are very useful.

One important point is that all positive functions of 3 variables are threshold functions [20], [23]. Therefore, all the corresponding stack filters are generalized ranked-order filters, as in (6).

In the following classification, we list the stack filters of window width 3 according to their roots; for each filter, we give the positive Boolean function B and the sample function of the stack filter.

Stack Filters Which Preserve Only Constant Roots

$B(x_1, x_2, x_3)$	$S_B(s_1, s_2, s_3)$	Comments
0	0	filters every input to 0
1	$M - 1$	filters every input to $M - 1$
x_1	s_1	right-shift operator
x_3	s_3	left-shift operator
x_1x_3	min (s_1, s_3)	
$x_1 + x_3$	max (s_1, s_3)	
$x_1x_2x_3$	min (s_1, s_2, s_3)	minimum filter
$x_1 + x_2 + x_3$	max (s_1, s_2, s_3)	maximum filter

Stack Filters Which Preserve Only Increasing Roots

$B(x_1, x_2, x_3)$	$S_B(s_1, s_2, s_3)$
x_2x_3	min (s_2, s_3)
$x_1 + x_2$	max (s_1, s_2)
$x_1 + x_2x_3$	2nd largest element of $\{s_1, s_1, s_2, s_3\}$
$x_1x_3 + x_2x_3$	2nd smallest element of $\{s_1, s_2, s_3, s_3\}$

Stack Filters Which Preserve Only Decreasing Roots

This class of filters is obtained from the previous class by reversing the order of the variables.

$B(x_1, x_2, x_3)$	$S_B(s_1, s_2, s_3)$
x_1x_2	min (s_1, s_2)
$x_2 + x_3$	max (s_2, s_3)
$x_3 + x_1x_2$	2nd largest element of $\{s_1, s_2, s_3, s_3\}$
$x_1x_2 + x_1x_3$	2nd smallest element of $\{s_1, s_1, s_2, s_3\}$

Stack Filters Which Preserve Median Filter Roots and/or Other Roots

$B(x_1, x_2, x_3)$	$S_B(s_1, s_2, s_3)$	Comments
x_2	s_2	the identity function; preserves everything
$x_1x_2 + x_1x_3 + x_2x_3$	median (s_1, s_2, s_3)	the median filter
$x_2 + x_1x_3$	2nd largest element of $\{s_1, s_2, s_2, s_3\}$	eliminates negative-going impulses
$x_1x_2 + x_2x_3$	2nd smallest element of $\{s_1, s_2, s_2, s_3\}$	eliminates positive-going impulses

We can show that each of these filters causes every input to converge to one of its roots. After eliminating the first two trivial filters, and the identity function, we have 17 filters to consider. Of these, we already know that the maximum and minimum filters make every input converge to a constant signal [24], and that the median filter filters every input to a root [5]. Finally, by using the dual and time-reversal operations, we can reduce the problem even further; the action of all the filters can be determined by examining the filters based on the functions x_1, x_1x_3, x_2x_3, $x_1x_3 + x_2x_3$, and $x_2 + x_1x_3$.

$B(x_1, x_2, x_3) = x_1$. Clearly, S_B will filter any appended signal to a constant signal by extension of the constant region at the left end of the signal.

$B(x_1, x_2, x_3) = x_1x_3$: if $x_2 = 0$, then $B(x_1, x_2, x_3) = 0$ if and only if $x_1 = 0$ or $x_3 = 0$; if $x_2 = 1$, then $B(x_1, x_2, x_3) = 1$ if and only if $x_1 = x_3 = 1$.

Hence, in a binary signal, each region of two or more zeros will extend in both directions at each pass, whereas any region of two or more ones will shrink by two bits at each pass. Curiously, any oscillation $\cdots$ 01010 $\cdots$ that is bounded at one end by a region of ones will grow in the direction until the region of ones disappears.

Still, any signal will converge to a constant signal. If one end bit of a signal is 0, then the constant region of zeros at that end of the appended signal will extend inward at each pass. If the signal is all ones, then it will be preserved. However, if it has an end bit of 1, and a 0 bit somewhere, the constant region of ones at the end of the appended signal must shrink at each pass, until the end bit becomes a zero; at that point, the new end region of zeros will extend inward at each pass. Therefore, unless a binary signal contains no zeros, it will eventually converge to a zero signal.

$B(x_1, x_2, x_3) = x_2x_3$: as $S_B = \min(s_2, s_3)$, we see that any negative peak in the input signal will extend to the left on repeated passes of the filter, until it encounters the trailing edge of a lower peak. The final root corresponds to the output of a negative peak detector with no decay, operating on the input from right to left.

$B(x_1, x_2, x_3) = x_1x_3 + x_2x_3$: if $x_2 = 0$, the output is 0 if and only if $x_1 = 0$ or $x_3 = 0$; if $x_2 = 1$, the output is 1 if and only if $x_3 = 1$. On each pass of the binary filter, each zero region in a binary signal will extend to the left by at least one bit—except for isolated zeros which will move to the left on repeated passes of the filter. An isolated zero will stop moving when it meets a larger region of zeros; this, and the extension of such regions to the left, guarantees that the final binary root will be increasing. Hence, S_B will filter any multilevel signal to an increasing signal.

$B(x_1, x_2, x_3) = x_2 + x_1x_3$: if $x_2 = 1$, the output is 1 no matter what x_1 and x_3 are; if $x_2 = 0$, then $B(x_1, x_2, x_3) = 0$, if and only if $x_1 = 0$ or $x_3 = 0$. Hence, the binary filter will eliminate the 0 in 101, but it will preserve every other bit in a signal. S_B will eliminate all negative-going impulses in a signal in one pass, but preserve all other features of the signal; it follows that the output after one pass is a root of S_B.

Since these five filters make every input converge, the other filters must do the same.

Note that the root sets of the filters based on $x_2 + x_1x_3$ and $x_1x_2 + x_2x_3$ are supersets of the roots of the median filter. The first filter eliminates negative impulses, but preserves positive impulses, whereas the second filter does the opposite. The median filter preserves all appended signals with no impulses.

Because of these properties, we call the two stack filters based on the Boolean functions $x_2 + x_1x_3$ and $x_1x_2 + x_2x_3$ *asymmetric median filters*. Fig. 7 (see also Fig. 6) shows the results of filtering a binary signal with a median filter and each of the two asymmetric median filters. Note that each filtered signal is a root of the corresponding filter.

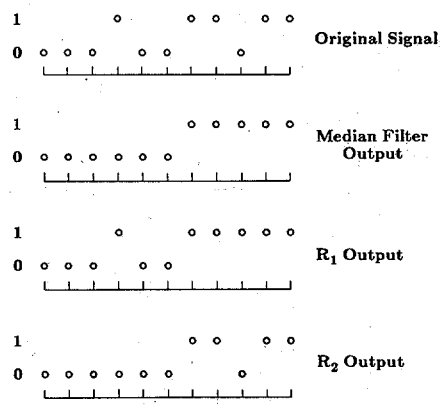


Fig. 7. Filtering of a binary signal with a median filter of window width 3, a positive-impulse preserving asymmetric median filter (R_1), and a negative-impulse preserving asymmetric median filter (R_2).

The asymmetric median filters may be useful in several areas. One could use them to eliminate spurious bright spots in an image, while preserving the rest of the image. Furthermore, since they tend to eliminate impulses of one sign, but preserve those of the opposite sign, they could be used to process geological or biomedical signals, in which large impulses of one sign are often significant features of the signals [25].

The filters that preserve only monotonic signals would be useful in eliminating gray-level gradients in an image. Using one of these filters to filter the rows of an image to roots, we obtain a gradient estimate for each row that contains essentially no local detail of the image. On subtracting each root from the original row, we eliminate the gradient, but preserve the details of the image.

VII. Conclusion

In this paper we have introduced a new class of filters called stack filters. They offer the advantage of efficient VLSI implementation along with the ability to perform a very large number of useful nonlinear operations. Even in the class of window width 3 stack filters we found several very useful filters.

Much research remains to be done on these filters.

1) The results presented in this paper concerning the convergence and invariance properties of these filters are probably only a small part of what is actually true; following this line of investigation should be very fruitful.

2) What is the optimal stack filter for a particular situation? When are they optimal relative to other types of filters? We have already begun to answer these questions [16], [17]; the results promise to provide significant advances in image processing.

3) Of course, generalizations are possible. There is no fundamental reason for requiring the same Boolean function on each level in the stack filter. Any set of filters which have the stacking property relative to each other could be used. These filters, which might be called nonhomogeneous stack filters, would provide a great deal of flexibility in nonlinear filter design.

The list could continue, but the point has been made—this is a very rich area.

Appendix—Proofs

Proof of Theorem 3: All increasing M-level signals decompose into binary signals of the forms $\cdots 0000 \cdots$ or $\cdots 1111 \cdots$ or $\cdots 00001111 \cdots$.

If we consider a decomposition of S_B into M-1 binary filters B, we see that the bit sequence in the window of any binary filter will always be of the form

$$0^k 1^{2N+1-k}.$$

If $k \le N$, then the central bit is 1, whereas if $k \ge N+1$, the central bit is 0. Furthermore, if $k_1 < k_2$, then $0^{k_2}1^{2N+1-k_2} < 0^{k_1}1^{2N+1-k_1}$. Since $\vec{x} \le \vec{y} \Rightarrow B(\vec{x}) < B(\vec{y})$, we have the following.

If $B(0^N1^{N+1}) = 1$ and $B(0^{N+1}1^N) = 0$, then

$$B(0^k1^{2N+1-k}) = 0 \quad k \ge N+1 \qquad \text{(a)}$$

$$= 1 \quad k \le N. \qquad \text{(b)}$$

Therefore, B preserves all binary threshold signals for any increasing input, so S_B preserves all increasing M-level inputs.

Conversely, if S_B preserves all increasing M-level signals, then (a) and (b) hold. In particular, for $k = N$, $B(0^N1^{N+1}) = 1$, and for $k = N+1$, $B(0^{N+1}1^N) = 0$. □

Proof of Corollary 1: From theorem 3, we need only prove that (a) and (b) in the corollary are equivalent to:

$$B(0^N1^{N+1}) = 1$$

and

$$B(0^{N+1}1^N) = 0.$$

If (a) and (b) are true, then $x_{N+1} = 0$ and $x_{N+2} = x_{N+3} = \cdots = x_{2N+1} = 1 \Rightarrow B(x_1, x_2, \cdots, x_{2N+1}) = 0$, i.e., $B(0^{N+1}1^N) = 0$. Note that, in this case, any term that does not depend only on $\{x_{N+1}, x_{N+2}, \cdots, x_{2N+1}\}$ will be 0. Also, (a) and (b) imply that $x_{N+1} = x_{N+2} = \cdots = x_{2N+1} = 1 \Rightarrow B(x_1, x_2, \cdots, x_{2N+1}) = 1$, i.e., $B(0^N1^{N+1}) = 1$.

Conversely,

$$B(0^N1^{N+1}) = 1 \Rightarrow \exists \vec{y} = (y_1, y_2, \cdots, y_{2N+1}) \le 0^N1^{N+1} \text{ such that } B(\vec{y}) = 1$$

$$\Rightarrow \exists \vec{y} = 0^N y_{N+1} y_{N+2} \cdots y_{2N+1} \text{ such that } B(\vec{y}) = 1$$

$$\Rightarrow \exists \text{ terms in the MSP form of } B \text{ that depend only on } \{x_{N+1}, x_{N+2}, \cdots, x_{2N+1}\}$$

$$B(0^{N+1}1^N) = 0 \Rightarrow \text{no term of } B \text{ depends only on } \{x_{N+1}, x_{N+2}, \cdots, x_{2N+1}\} \text{ can be 1 if } x_{N+1} = 0$$

$$\Rightarrow \text{all terms of } B \text{ that contain only variables from}$$

$$\{x_{N+1}, x_{N+2}, \cdots, x_{2N+1}\} \text{ must contain } x_{N+1}. \quad \square$$

Proof of Theorem 5:

($\Rightarrow$): obvious, as all constant regions of length $N' + 1$ or more are preserved,

($\Leftarrow$): by the properties of positive Boolean functions,

(a) $\Rightarrow$ all constant regions of 1's of length $\geq N' + 1$ are preserved,

(b) $\Rightarrow$ all constant regions of 0's of length $\geq N' + 1$ are preserved.

Hence, B preserves all binary roots of a median filter of window width $2N' + 1$, so S_B must preserve all M-level roots of the same filter. $\square$

Proof of Corollary 4: When expressed in terms of a threshold function with weights $a_1, a_2, \cdots a_{2N+1}$ and threshold c, the two conditions of theorem 5 become

$$\sum_{i=m+1}^{m+N'+1} a_i \geq c \qquad N - N' \leq m \leq N$$

and

$$\sum_{i=1}^{m} a_i + \sum_{i=2N+1}^{m+N'+1} + 2a_i < c \qquad N - N' \leq m \leq N.$$

From the last condition, we obtain

$$\sum_{i=1}^{2N+1} a_i - \sum_{i=m+1}^{m+N'+1} a_i < c$$

$$\Leftrightarrow \sum_{i=m+1}^{m+N'+1} a_i > \sum_{i=1}^{2N+1} a_i - c$$

$$\Leftrightarrow \sum_{i=m+1}^{m+N'+1} a_i \geq \sum_{i=1}^{2N+1} a_i - c - 1$$

for all possible values of m.

The result of the corollary follows immediately. $\square$

Proof of Theorem 6: Suppose (a)–(d) hold. We consider the convergence to be a growth of rootlike regions from the ends of the appended signal. The end bits of a binary signal, plus the appended bits, form constant regions of length $N + 1$ at each end of the signal, which are rootlike. If we can show that these end regions always extend inward on each pass, except if the whole signal is a root, then we have shown that every appended signal will converge to a root in a finite number of passes.

If the left bit of an unappended signal is 0, then the window centered on the first 1 (if it exists) will be of the form

$$0^N 1 x_{N+2} x_{N+3} \cdots x_{2N+1}.$$

If (a) holds, the filter will leave the first 1 unchanged if and only if $x_{N+2}, x_{N+3}, \cdots, x_{2N+1} = 1$, i.e., the 1 is at the left end of a region of at least $N + 1$ 1s. Otherwise, it will become a 0. Similarly, (b) covers the case when the leftmost bit of the signal is a 1, while (c) and (d) relate to the right end of the appended signal.

Note that, for constant signals, none of the conditions apply, but a constant signal is already a trivial root of a median filter.

Together, (a)–(d) imply that, until the whole signal is a root, the end root regions must extend inward on each pass. $\square$

Proof of Corollary 6: We need to show that the conditions of the corollary are equivalent to the conditions of theorem 6. Considering those conditions in turn, the following is so.

a) $B(0^N 1^{N+1}) = 1$, but for $\vec{x} < 0^N 1^{N+1}$, $B(\vec{x}) = 0$,

$\Leftrightarrow \exists$ term in the MSP form of B that depends only on $\{x_{N+1}, x_{N+2}, \cdots, x_{2N+1}\}$, but none that depends on a proper subset thereof,

$\Leftrightarrow x_{N+1}, x_{N+2}, \cdots, x_{2N+1}$ is a term in the MSP form of B.

Similarly,

c) $B(1^{N+1}0^N) = 1$, but for $\vec{x} < 1^{N+1}0^N$, $B(\vec{x}) = 0$,

$\Leftrightarrow x_1 x_2 \cdots x_{N+1}$ is a term in the MSP form of B.

b) $B(1^N 0^{N+1}) = 0$ but for $\vec{x} > 1^N 0^{N+1}$, $B(\vec{x}) = 1$,

$\Leftrightarrow B(1^N 0^{N+1}) = 0$, but $B(1^N 0^k 1 0^{N-k}) = 1$ $\forall k = 0, 1, \cdots, N$,

$\Leftrightarrow \forall i = N + 1, N + 2 \cdots, 2N + 1, \exists$ term in the MSP form of B that contains only x_i and variables from $\{x_1, x_2, \cdots, x_N\}$.

Similarly,

d) $B(0^{N+1}1^N) = 0$, but for $\vec{x} > 0^{N+1}1^N$, $B(\vec{x}) = 1$,

$\Leftrightarrow \forall i = 1, 2, \cdots, N + 1, \exists$ term in the MSP form of B that contains only x_i and variables from $\{x_{N+2}, x_{N+3}, \cdots, x_{2N+1}\}$. $\square$

References

[1] V. Krishnan, *Nonlinear Filtering and Smoothing: An Introduction to Martingales, Stochastic Integrals and Estimation*. New York: Wiley, 1984.

[2] A. Segal, "A Martingale approach to modeling, estimation, and detection of jump processes," Inform. Syst. Lab., Stanford Univ., Stanford, CA, Tech. Rep. 7050-21, Aug. 1973.

[3] V. E. Benes, "Exact finite dimensional filters for certain diffusions with nonlinear drift," *Stochastics*, vol. 5, pp. 65–92, 1981.

[4] J. W. Tukey, "Nonlinear (nonsuperposable) methods for smoothing data," in *Congr. Rec. 1974 EASCON*, p. 673.

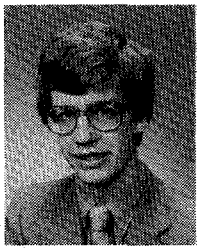
[5] N. C. Gallagher and G. L. Wise, "A theoretical analysis of the properties of median filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 1136–1141, Dec. 1981.

[6] T. S. Huang and G. J. Yang, "Median filters and their applications to image processing," School Elec. Eng., Purdue Univ., West Lafayette, IN, Tech. Rep. EE 80-1, Jan. 1980.

[7] A. C. Bovik, T. S. Huang, and D. C. Munson, Jr., "Image restoration using order-constrained least-squares methods," in *Proc. ICASSP*, 1983, pp. 828–831.

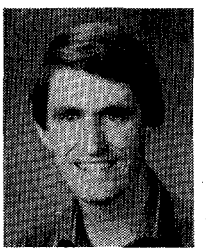
[8] T. A. Nodes and N. C. Gallagher, "Two-dimensional root structures and convergence properties of the separable median filter," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 1350–1365, Dec. 1983.

[9] G. R. Arce and N. C. Gallagher, Jr., "State description for the root-signal set of median filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 894–902, Dec. 1982.
[10] J. P. Fitch, E. J. Coyle, and N. C. Gallagher, "Root properties and convergence rates of median filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 230–240, Feb. 1985.
[11] P. D. Wendt, E. J. Coyle, and N. C. Gallagher, "Some convergence properties of median filters," *IEEE Trans. Circuits Syst.*, to appear.
[12] J. P. Fitch, E. J. Coyle, and N. C. Gallagher, "Median filtering by threshold decomposition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 1183–1188, Dec. 1984.
[13] ——, "Threshold decomposition of multidimensional ranked-order operations," *IEEE Trans. Circuits Syst.*, vol. CAS-32, pp. 445–450, May 1985.
[14] F. Kuhlman and G. L. Wise, "On the spectral characteristics of median filtered independent data," *IEEE Trans. Commun.*, vol. COM-29, pp. 1374–1379, Sept. 1981.
[15] M. Basseville and A. Benveniste, "Design and comparative study of some sequential jump detection algorithms for digital signals," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 521–535, June 1983.
[16] E. Coyle, "On the optimality of rank order operators," in *Proc. XX Annu. Conf. Info. Sci. Syst.*, Princeton, NJ, Mar. 1986.
[17] ——, "On the optimality of rank order filters and stack filters," submitted to *IEEE Trans. Acoust., Speech, Signal Processing*.
[18] T. A. Nodes and N. C. Gallagher, "Theoretical results on the properties of median type filters," School Elec. Eng., Purdue Univ., West Lafayette, IN, Tech. Rep.-EE 82-32, Sept. 1982.
[19] E. N. Gilbert, "Lattice-theoretic properties of frontal switching functions," *J. Math. Phys.*, vol. 33, pp. 57–67, Apr. 1954.
[20] S. Muroga, *Threshold Logic and Its Applications*. New York: Wiley, 1971.
[21] T. A. Nodes and N. C. Gallagher, "The output distribution of median type filters," *IEEE Trans. Commun.*, vol. COM-32, pp. 532–541, May 1984.
[22] E. Ataman, V. K. Aatre, and K. M. Wong. "Statistical properties of median filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 1073–1075, Oct. 1981.
[23] P. M. Lewis and C. L. Coates, *Threshold Logic*. New York: Wiley, 1967.
[24] T. A. Nodes and N. C. Gallagher, "Median filters: Some modifications and their properties," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 739–746, Oct. 1982.
[25] M. Basseville and A. Benveniste, "Design and comparative study of some sequential jump detection algorithms for digital signals," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 521–535, June 1983.
[26] P. A. Maragos and R. W. Schafer, "A unification of linear, median, order statistics, and morphological filters under mathematical morphology," in *Proc. ICASSP'85*, Tampa, FL, Mar. 1985.
[27] J. Serra, *Image Analysis and Mathematical Morphology*. New York: Academic, 1982.
[28] J. P. Fitch, E. J. Coyle, and N. C. Gallagher, "Analysis and implementation of median type filters," School Elec. Eng., Purdue Univ., West Lafayette, IN, Tech. Rep. EE 84-20, July 1984.
[29] R. G. Harber, S. C. Bass, and G. L. Neudeck, "VLSI implementation of a fast rank-order algorithm," in *Proc. ICASSP'85*, Tampa, FL, Mar. 1985.
[30] J. P. Fitch, E. J. Coyle, and N. C. Gallagher, "Median filtering by threshold decomposition," in *Proc. 1984 Princeton Conf. Inform. Sci., Syst.*, Princeton Univ., Princeton, NJ, Mar. 1984.
[31] J. P. Fitch, "Software and VLSI algorithms for generalized ranked order filters," submitted to *IEEE Trans. Circuits Syst.*
[32] P. D. Wendt, E. J. Coyle, and N. C. Gallagher, "Stack filters: Their definition and some initial properties," in *Proc. 1985 Conf. Inform. Sci. Syst.*, Baltimore, MD, Mar. 1985.



Peter D. Wendt (M'84) was born in Fielding, New Zealand, in February 1957. He received the B.Sc. with honors in pure mathematics from the University of Adelaide, Adelaide, Australia, in 1978, and the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1981 and 1985, respectively.

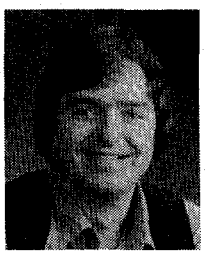
In June 1985 he joined the Electronics Laboratories of the General Electric Corporate Research and Development Center in Schenectady, NY. His technical interests include quantization theory, nonlinear filters, speech and image processing, and DSP architectures.



Edward J. Coyle (S'79–M'82) was born in Philadelphia, PA, on April 22, 1956. He received the B.S. degree in electrical engineering from the University of Delaware, Newark, and the M.S. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, in 1980 and 1982, respectively.

Since August 1982 he has been an Assistant Professor of Electrical Engineering at Purdue University, West Lafayette, IN. His research interests include nonlinear digital signal processing and the modeling and analysis of computer communication networks.

Dr. Coyle is a member of Phi Kappa Phi, Tau Beta Pi, and Eta Kappa Nu.



Neal C. Gallagher, Jr. (S'72–M'75) received the Ph.D. degree in electrical engineering in 1974 from Princeton University, Princeton, NJ.

After being a member of the Faculty of Case Western Reserve University, Cleveland, OH, he joined Purdue University, West Lafayette, IN, in 1976, where he is currently a Professor of Electrical Engineering. He is the President of Gallagher Associates, Inc., doing research and system design in microwave holography, diffractive optics, guidance systems, and signal processing.