Threshold Boolean Filters

Ki Dong Lee and Yong Hoon Lee

***Abstract*—A class of nonlinear digital filters, called the threshold Boolean filter (TBF), is introduced. The TBF is defined by a Boolean function on the binary domain and is a natural extension of stack filters. Multilevel representations of a TBF corresponding to a Boolean function are derived; a TBF can be represented either as a sum of *"local minimum—local maximum"* terms or as an adaptive linear combination of ordered input data. It is shown that TBF's may be neither translation invariant nor scale invariant and that any TBF can be expressed as a linear combination of stack filters. A subclass of TBF's, called linearly separable (LS) TBF's, defined by the threshold logic is introduced as a direct extension of weighted-order statistic (WOS) filters. Implementation and design of a TBF and an LS TBF is investigated. The procedure for designing TBF's (LS TBF's) is shown to be considerably simpler than designing stack (WOS) filters, and the former can outperform the latter at marginal increase in computational cost. Finally, experimental results are presented to illustrate the performance characteristics of TBF's and LS TBF's.**

I. Introduction

A NUMBER of digital filters possess the threshold decomposition property [1]. The output of a filter with threshold decomposition can be obtained by decomposing an input signal into a set of binary signals, carrying out the filtering operation separately on each binary signal and then by summing up the results. To be specific, consider a nonrecursive filter whose output $Y(n)$ is denoted by $Y(n) = F(\mathbf{X}(n))$, where $\mathbf{X}(n) \equiv (X_1(n), X_2(n), \ldots, X_N(n))$ is the input vector within a window at time n, $X_j(n)$ is the jth input sample from the left of the window, and N is the window size. Assume that the input is $(M+1)$ valued, i.e., $X_i(n) \in \{0, 1, \ldots, M\}$. If this filter obeys the threshold decomposition, then the output $Y(n)$ can be expressed as

$$Y(n) = F(\mathbf{X}(n)) = \sum_{\ell=1}^{M} F(I_\ell(\mathbf{X}(n))) \tag{1}$$

where $I_\ell(\mathbf{X}(n)) \equiv (I_\ell(X_1(n)), I_\ell(X_2(n)), \ldots, I_\ell(X_N(n)))$ is the thresholding operator defined as $I_\ell(x) = 1$ if $x \geq \ell$ and 0 otherwise. Linear FIR and IIR filters and nonlinear filters such as stack filters [2]—which include median [3], rank order [4], and weighted order statistic (WOS) [5] filters as special cases—linear combination of order statistics (LOS) filters [6], [7] and linear combination of weighted order statistics (LWOS) filters [8] obey the threshold decomposition property. In addition, multilevel morphological filters are defined on the basis of the threshold decomposition [9], [10].

Manuscript received October 30, 1992; revised October 11, 1993. This work was supported by Korea Ministry of Science and Technology. Portions of this work were presented at the Canadian Conference on Electrical and Computer Engineering, Toronto, Canada, September 1992.

K. D. Lee is with the Image and Media Laboratory, GoldStar Co., Ltd., Seoul, Korea.

Y. H. Lee is with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Taejon Korea.

IEEE Log Number 9401904.

The filters satisfying the threshold decomposition can be fully specified on the binary domain by a truth table or by an extended truth table [8] that lists all possible binary input vectors and the corresponding output values. The (extended) truth table representation is useful for analyzing and implementing the nonlinear filters. In the case of stack filtering, the truth table representation reduces to a positive Boolean function performing only logical OR and logical AND operations.

The concept of threshold decomposition leads to the class of filters that are defined on the binary domain by $f_\ell(\cdot)$ and whose multilevel representation is given by

$$Y(n) = \sum_{\ell=1}^{M} f_\ell(I_{\ell-L}(\mathbf{X}(n)), \ldots, I_\ell(\mathbf{X}(n)), \ldots, I_{\ell+L}(\mathbf{X}(n))) \tag{2}$$

where L is a nonnegative integer. In this filter, the binary vectors from $2L+1$ threshold levels nearest to the level ℓ are input to the binary domain operator $f_\ell(\cdot)$, which may vary depending on ℓ. Generalized stack filters [11] and microstatistic filters [12] are defined following this approach. When $L = 0$ and $f_\ell(\cdot) = f(\cdot)$ for all ℓ is a positive Boolean function, the class of filters defined by (2) reduces to stack filters.

When a filter is specified on the binary domain and its multilevel representation is given by (2), a natural question arises: Can we directly express such a filter on the multilevel without using the threshold decomposition? The answer to this question is affirmative for the cases associated with linear FIR, stack, and LWOS filters, where $L = 0$ and $f_\ell(\cdot) = f(\cdot)$ for all ℓ. For example, any filter that is specified by an extended truth table on the binary domain and by (2) on the multilevel with $L = 0$ and $f_\ell(\cdot) = f(\cdot)$ for all ℓ can be expressed directly as an LWOS filter if the filter produces zero output value for the zero input vector [8]. On the other hand, the direct multilevel expressions of the generalized stack and microstatistic filters are generally unknown.

In this paper, we focus our attention on the filters that are specified by a Boolean function $f(\cdot)$ on the binary domain and defined by (2) with $L = 0$ and $f_\ell(\cdot) = f(\cdot)$ for all ℓ on the multilevel. These filters will be referred to as the *threshold Boolean filters* (TBF's). The class of TBF's includes stack filters as a special case. We shall develop multilevel TBF representations and investigate the properties, implementation, and design of TBF's. It will be shown that a TBF, which

1053-587X/94$04.00 © 1994 IEEE

outperforms the optimal stack filter in [25], can be obtained following the procedure for designing the stack filter.

As a very interesting and useful subclass of TBF's, we will introduce what we call the *linearly separable* (LS) TBF. The LS TBF is defined by an LS Boolean function called the threshold logic [17], [18]. Since a WOS filter is defined by an LS Boolean function with nonnegative coefficients [5], the LS TBF is a direct extension of the WOS filter. It will be seen that the method for designing WOS filters can be applied to design LS TBF's.

The organization of this paper is as follows. In Section II, the TBF is defined by a Boolean function on the binary domain, and its multilevel representation based on true vectors of the Boolean function is derived. In Section III, we develop three alternative TBF representations: two of them are based on the sum-of-product (SOP) expression of a Boolean function, and the third one is represented in terms of ordered input data. It is shown that the LS TBF can be expressed succinctly on the multilevel by using the ordered input data. Some properties of the TBF are investigated in Section IV, where the relation among TBF's, rank order, and LOS filters is discussed. In Section V, we consider implementations of the TBF. In Section VI, procedures for designing a TBF and an LS TBF are described, and the performance of the TBF and LS TBF is examined through computer simulation. Finally, Section VII presents conclusions.

II. Threshold Boolean Filters

A TBF, which is denoted by $\mathrm{TBF}_f(\mathbf{X})$, is defined as

$$\mathrm{TBF}_f(\mathbf{X}) = \sum_{\ell=1}^{M} f(I_\ell(\mathbf{X})) \tag{3}$$

where $f(\cdot)$ is a Boolean function, and $\mathbf{X} = (X_1, \ldots, X_N)$ is an input vector. Here, as well as in the rest of this paper, the time index n is dropped from $\mathbf{X}(n)$ and $X_j(n)$ to simplify the notation. When $f(\cdot)$ is a positive Boolean function, the corresponding multilevel expression is obtained by exchanging logical OR and logical AND operations of $f(\cdot)$ with maximum and minimum operations, respectively; the filter is a stack filter. For example, for $f(\mathbf{x}) = x_1x_2 + x_2x_3$, $N = 3$, where the multiplication and addition represent logical AND and OR operations, respectively, we get $\mathrm{TBF}_f(\mathbf{X}) = \max\{\min\{X_1, X_2\}, \min\{X_2, X_3\}\}$. This is possible since every positive Boolean function commutes with thresholding [13], and the maximum and minimum operations become the logical OR and the logical AND, respectively, for binary inputs. On the other hand, for $f(\cdot)$, which is a Boolean function with logical negations (complements), the corresponding $\mathrm{TBF}_f(\cdot)$ cannot be obtained directly because such a Boolean function does not commute with thresholding, and the multilevel operator that reduces to the logical negation for binary inputs does not exist. The multilevel representations of a TBF, which are obtained in the following subsection, will show that the logical negation is closely related to the multilevel minus ($-$) operation.

The TBF in (3) becomes the LS TBF when $f(\mathbf{x})$ is an LS Boolean function, which is defined as

$$f(x_1, \ldots, x_N) = \begin{cases} 1, & \text{if } \sum_{i=1}^{N} w_i x_i \geq T \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where the weights w_i and the threshold T are real numbers. If the weights and the threshold are limited to be nonnegative and $\sum_{i=1}^{N} w_i \geq T$, then an LS Boolean function becomes positive, and the corresponding LS TBF reduces to a WOS filter. We shall see that an LS TBF is simpler to implement than general TBFs.

It should be pointed out that there are some well-known multilevel operators that can be thought of as TBF's. In fact, if a multilevel operator $F(\cdot)$ can be expressed as in (1), and it produces a binary output for any binary input vector, then the filter is a TBF. The example below illustrates this.

Example 1: Consider the range estimator [14] $F(\mathbf{X}) = X_{(1)} - X_{(N)}$, where $X_{(1)}$ and $X_{(N)}$ are the maximum and the minimum, respectively, among $\{X_1, \ldots, X_N\}$. It is straightforward to see that this estimator obeys the threshold decomposition, and obviously, the estimator yields binary outputs for binary input values. The Boolean function corresponding to this with $N = 3$ is $f(\mathbf{x}) = x_1\bar{x}_2 + x_2\bar{x}_3 + x_3\bar{x}_1$. □

In a similar manner, we can see that the quasi-ranges [15] and the absolute difference between ordered data are TBF's.

In the following, the multilevel TBF representations are derived after introducing some notations and definitions.

A. Notations and Definitions

It is assumed that the input samples $\{X_1, \ldots, X_N\}$ are $(M+1)$ valued: $X_i \in D$ for all i and $\mathbf{X} \in D^N$, where $D = \{0, 1, \ldots, M\}$, which is a set of nonnegative integers. The jth largest sample among the input samples $\{X_1, \ldots, X_N\}$ is denoted by $X_{(j)}, j = 1, \ldots, N$. If two input values are identical, say, $X_i = X_j$, then either of them will be considered to be a larger one. We define $X_{(0)} = M$ and $X_{(N+1)} = 0$ so that $\mathbf{X}_{(N+1)} \leq X_{(N)} \cdots \leq X_{(1)} \leq X_{(0)}$ holds. The difference $X_{(j)} - X_{(j+1)}$ is denoted by C_j, $0 \leq j \leq N$. Note that $C_0 = M - X_{(1)}$ and $C_N = X_{(N)}$. For some positive integer a and b, the set of integers $S[a, b] \equiv \{a, a+1, \ldots, b\}$, which is a subset of D, is defined. Here, $S[a, b] \equiv \emptyset$ if $a > b$. $|S[a, b]|$ denotes the number of elements in $S[a, b]$, that is, $|S[a, b]| = b - \mathbf{a} + 1$ if $b \geq a$, and 0 otherwise.

Now, consider a Boolean function $f(\mathbf{x}) = \pi_1(\mathbf{x}) + \pi_2(\mathbf{x}) + \cdots + \pi_P(\mathbf{x})$ expressed as a SOP, where $\mathbf{x} = (x_1, \ldots, x_N)$ denotes a vector of N binary entries, and $\pi_j(\mathbf{x})$ represents the jth product of $f(\mathbf{x})$. A binary vector, say, $\mathbf{x}_0$, is called a true (false) vector of $f(\cdot)$ if $f(\mathbf{x}_0) = 1(0)$. The set of all true vectors of $f(\mathbf{x})$ is denoted as $V(f)$. Similarly, the set of all true vectors of $\pi_j(\mathbf{x})$ is expressed as $V(\pi_j)$. Note that $V(f) = \bigcup_{j=1}^{P} V(\pi_j)$. The products $\pi_i(\mathbf{x})$ and $\pi_j(\mathbf{x})$ are said to be *mutually exclusive* if $V(\pi_i)$ and $V(\pi_j)$ are disjoint, i.e., $V(\pi_i) \cap V(\pi_j) = \emptyset$. As an example, consider $\pi_1(\mathbf{x}) = x_1\bar{x}_2$, $\pi_2(\mathbf{x}) = x_1x_3$, and $N = 3$. Then, $V(\pi_1) = \{(1,0,0), (1,0,1)\}$, $V(\pi_2) = \{(1,0,1), (1,1,1)\}$, and $\pi_1(\mathbf{x})$ and $\pi_2(\mathbf{x})$ are not mutually exclusive. The binary vectors $(1, 1, \ldots, 1)$ and $(0, 0, \ldots, 0)$ are denoted by $\mathbf{1}$ and $\mathbf{0}$,

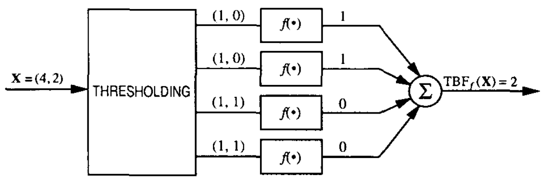


Fig. 1. Filtering operation of the TBF with $f(\mathbf{x}) = x_1\bar{x}_2$.

respectively. Finally, the number of 1's in a binary vector $\mathbf{x}$, which is the Hamming weight, is denoted by $w_H(\mathbf{x})$.

B. A Multilevel Representation of the TBF in Terms of True Vectors

Following from (3), the TBF can be expressed as shown in (5) at the bottom of the page, where $\ell \in S[1, M]$. Once the set of true vectors $V(f)$ is known, a multilevel representation can be obtained in a straightforward manner, as illustrated in the following simple example.

Example 2: Suppose that $f(\mathbf{x}) = x_1\bar{x}_2$, $N = 2$. The set of true vectors $V(f) = \{(1,0)\}$. Consider (3). At a level ℓ, the binary input vector becomes a true vector if and only if (iff) $X_2 < \ell \leq X_1$. Assume $X_2 < X_1$. Then, $f(I_\ell(\mathbf{X})) = 1$ for all $X_2 < \ell \leq X_1$ and following from (5), $\text{TBF}_f(\mathbf{X})$ is equal to the number of ℓ's for which $I_\ell(\mathbf{X}) = (1,0)$. Thus, $\text{TBF}_f(\mathbf{X}) = X_1 - X_2$. When $X_2 \geq X_1$, the binary input vector at each level is not a true vector, and thus, $\text{TBF}_f(\mathbf{X}) = 0$. Combining these, $\text{TBF}_f(\mathbf{X}) = \max\{0, X_1 - X_2\}$. Fig. 1 illustrates this result for $\mathbf{X} = (4,2)$. For this input, $\text{TBF}_f(\mathbf{X}) = 2$. Note that in Fig. 1, the same output is obtained through the threshold decomposition. □

A multilevel representation corresponding to an arbitrary Boolean function is presented next.

Proposition 1 (Representation for a Single True Vector): Suppose that a Boolean function $f(\cdot)$ has only one true vector, say, $\mathbf{v} = (v_1, \ldots, v_N)$. Then

$$\text{TBF}_f(\mathbf{X}) = \max\{0, \min\{X_j \mid j \in B_1(\mathbf{v})\} - \max\{X_j \mid j \in B_0(\mathbf{v})\}\}, \tag{6}$$

where $\min\emptyset \equiv M$, $\max\emptyset \equiv 0$, and $B_0(\mathbf{v})(B_1(\mathbf{v}))$ is the set of all indices of $v_i, i = 1, 2, \ldots, N$, which are zero (one).

Proof: Let $\mathbf{v} = \mathbf{1}(B_0(\mathbf{v}) = \emptyset)$. Then, $I_\ell(\mathbf{X}) = \mathbf{1}$ iff $1 \leq \ell \leq \min\{X_1, \ldots, X_N\}$ and $\text{TBF}_f(\mathbf{X}) = \min\{X_1, \ldots, X_N\}$. Now, let $\mathbf{v} = \mathbf{0}(B_1(\mathbf{v}) = \emptyset)$. Then, $I_\ell(\mathbf{X}) = \mathbf{0}$ iff $\max\{X_1, \ldots, X_N\} < \ell \leq M$ and $\text{TBF}_f(\mathbf{X}) = M - \max\{X_1, \ldots, X_N\}$. Finally, let $\mathbf{v} \neq \mathbf{1}, \mathbf{0}$. For this case, $I_\ell(\mathbf{X}) = \mathbf{v}$ iff $\max\{X_j \mid j \in B_0(\mathbf{v})\} < \ell \leq \min\{X_j \mid j \in B_1(\mathbf{v})\}$. Since $\text{TBF}_f(\mathbf{X})$ is equal to the number of ℓ's for which $I_\ell(\mathbf{X}) = \mathbf{v}$, (6) is obtained. □

For $f(\mathbf{x})$ in Example 2, we get $\mathbf{v} = (1,0)$, $B_0(\mathbf{v}) = \{2\}$, and $B_1(\mathbf{v}) = \{1\}$; $\text{TBF}_f(\mathbf{X})$ is obtained through (6).

Proposition 2 (Representation Based on True Vectors): Suppose that a given Boolean function $f(\cdot)$ has K true vectors, $V(f) = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K\}$. Let $\text{TBF}_{f_i}(\mathbf{X})$ be the TBF output obtained by (6) when the Boolean function $f_i(\cdot)$ has a single true vector $\mathbf{v}_i$. Then, the multilevel representation of $f(\mathbf{x})$ is

$$\text{TBF}_f(\mathbf{X}) = \sum_{i=1}^{K} \text{TBF}_{f_i}(\mathbf{X}). \tag{7}$$

Proof: From (5), $\text{TBF}_f(\mathbf{X})$ can be written as $\text{TBF}_f(\mathbf{X}) = \sum_{i=1}^{K}[\textit{number of } \ell\textit{'s for which } I_\ell(\mathbf{X}) = \mathbf{v}_i]$, but $[\textit{number of } \ell\textit{'s for which } I_\ell(\mathbf{X}) = \mathbf{v}_i] = \text{TBF}_{f_i}(\mathbf{X})$. This completes the proof. □

Example 3: Suppose $f(\mathbf{x}) = x_1\bar{x}_2 + x_1x_3$ when $N = 3$. Then, we have $V(f) = \{(1,0,0), (1,0,1), (1,1,1)\}$. Let $\mathbf{v}_1 = (1,0,0)$, $\mathbf{v}_2 = (1,0,1)$ and $\mathbf{v}_3 = (1,1,1)$. Then, from (6), we get $\text{TBF}_{f_1}(\mathbf{X}) = \max\{0, X_1 - \max\{X_2, X_3\}\}$, $\text{TBF}_{f_2}(\mathbf{X}) = \max\{0, \min\{X_1, X_3\} - X_2\}$, and $\text{TBF}_{f_3}(\mathbf{X}) = \min\{X_1, X_2, X_3\}$; $\text{TBF}_f(\mathbf{X}) = \sum_{i=1}^{3} \text{TBF}_{f_i}(\mathbf{X})$. □

Since a TBF is expressed as a function of local minimum and local maximum operations, it can be specified by using an *ordering-output* table [13] that lists all possible input orderings and the corresponding outputs. The ordering-output table specifying the TBF in Example 3 is shown in Table I. Note that the table indeed specifies the TBF: the output of the TBF can always be obtained from the table without reference to the multilevel representation of the TBF. When two or more orderings can be thought of as the ordering of a given input vector (this happens when some input values are equal), any one of the orderings can be chosen, and the output associated with the selected ordering becomes the output of the filter. For example, in Table I, if $X_3 = X_2 < X_1$, then either $X_2 \leq X_3 \leq X_1$ or $X_3 \leq X_2 \leq X_1$ can be the input ordering. The outputs associated with the former and the latter, respectively, are X_1 and $X_1 - X_2 + X_3$, which are equivalent in this case because $X_2 = X_3$. The ordering-output table provides some insights into the behavior of the TBF. In Table I, the TBF selects one of the inputs as its output for all input orderings except for the ordering "$X_3 \leq X_2 \leq X_1$." The output for the ordering "$X_3 \leq X_2 \leq X_1$" is $X_1 - X_2 + X_3$. Note that only X_2, which is the only complemented literal in $f(\mathbf{x})$, has a minus ($-$) sign. In Section III, we shall see that only complemented literals can have a minus ($-$) sign in output representations of the ordering-output table.

The multilevel representation in (7) is somewhat inconvenient to use because the number of true vectors K is usually large. Next, we shall show that simpler multilevel representations of a TBF can be obtained from Boolean expressions without explicitly considering the true vectors of a Boolean function.

$$\begin{aligned}\text{TBF}_f(\mathbf{X}) &= [\textit{number of level } \ell\textit{'s for which } f(I_\ell(\mathbf{X})) = 1] \\ &= [\textit{number of level } \ell\textit{'s for which } I_\ell(\mathbf{X}) \textit{ is a true vector of } f(\mathbf{x})]\end{aligned} \tag{5}$$

TABLE I
ORDERING-OUTPUT TABLE FOR $f(\mathbf{x}) = x_1\bar{x}_2 + x_1x_3$

orderings	outputs
$X_1 \le X_2 \le X_3$	X_1
$X_1 \le X_3 \le X_2$	X_1
$X_2 \le X_1 \le X_3$	X_1
$X_2 \le X_3 \le X_1$	X_1
$X_3 \le X_1 \le X_2$	X_3
$X_3 \le X_2 \le X_1$	$X_1 - X_2 + X_3$

III. ALTERNATIVE MULTILEVEL REPRESENTATIONS OF THE TBF

In this section, multilevel TBF representations are derived from SOP Boolean expressions. In addition, another multilevel TBF expressed in terms of ordered input data is obtained.

A. Multilevel TBF from Sum of Products Representation

Consider a Boolean function $f(\cdot)$, which is given by the sum of P products $f(\mathbf{x}) = \pi_1(\mathbf{x}) + \pi_2(\mathbf{x}) + \cdots + \pi_P(\mathbf{x})$. Suppose each product $\pi_j(\mathbf{x})$ consists of q_j uncomplemented and r_j complemented literals. Then, it can be expressed as

$$\pi_j(\mathbf{x}) = x_{p(j,1)}x_{p(j,2)}\cdots x_{p(j,q_j)}\bar{x}_{n(j,1)}\bar{x}_{n(j,2)}\cdots\bar{x}_{n(j,r_j)},\quad j = 1,2,\ldots,P$$

where $p(\cdot,\cdot)$ and $n(\cdot,\cdot)$ denote indices of uncomplemented and complemented literals, respectively, and $1 \le q_j + r_j \le N$.

The Boolean function $f(\mathbf{x})$ produces 1 whenever at least one of the P products produces 1 for the binary input vector $\mathbf{x}$, and the product $\pi_j(\mathbf{x})$ produces 1 iff $\mathbf{x}$ satisfies

$$x_{p(j,1)} = x_{p(j,2)} = \cdots = x_{p(j,q_j)} = 1 \quad (8.a)$$

$$x_{n(j,1)} = x_{n(j,2)} = \cdots = x_{n(j,r_j)} = 0. \quad (8.b)$$

In other words, each true vector of $\pi_j(\mathbf{x})$ should satisfy both (8.a) and (8.b). Let $m_j(\mathbf{X})$ denote the minimum among the input samples corresponding to uncomplemented literals of $\pi_j(\mathbf{X})$, and let $M_j(\mathbf{X})$ denote the maximum among input samples corresponding to complemented literals of $\pi_j(\mathbf{X})$, that is, $m_j(\mathbf{X}) = \min\{X_{p(j,1)},\ldots,X_{p(j,q_j)}\}$ and $M_j(\mathbf{X}) = \max\{X_{n(j,1)},\ldots,X_{n(j,r_j)}\}$. We set $m_j(\mathbf{X}) = M$ if $q_j = 0$ and $M_j(\mathbf{X}) = 0$ if $r_j = 0$. Now, we derive multilevel representations from SOP Boolean expressions.

Proposition 3 (Representation for a Single Product Term): Suppose that a Boolean function has only one product term $f(\mathbf{x}) = \pi_1(\mathbf{x})$. Then

$$\text{TBF}_f(\mathbf{X}) = \max\{0, m_1(\mathbf{X}) - M_1(\mathbf{X})\}. \quad (9)$$

Proof: In the threshold decomposition, the binary vector $I_\ell(\mathbf{X}) = (x_1,\ldots,x_N)$ is generated by thresholding the input vector $\mathbf{X} = (X_1,\ldots,X_N)$ at level ℓ, where x_k will be 1 iff $\ell \in S[0,X_k]$ and 0 iff $\ell \in S[X_k+1,M]$. Therefore, $I_\ell(\mathbf{X})$ satisfies (8.a) iff $\ell \in \bigcap_{l=1}^{q_1} S[0,X_{p(1,l)}]$ and satisfies (8.b) iff $\ell \in \bigcap_{l=1}^{r_1} S[X_{n(1,l)}+1,M]$, and thus, $\pi_j(I_\ell(\mathbf{X})) = 1$ iff $\ell \in \left(\bigcap_{l=1}^{q_1} S[0,X_{p(1,l)}]\right) \cap \left(\bigcap_{l=1}^{r_1} S[X_{n(1,l)}+1,M]\right)$. Note that $\bigcap_{l=1}^{q_1} S[0,X_{p(1,l)}] = S[0,m_1(\mathbf{X})]$ and $\bigcap_{l=1}^{r_1} S[X_{n(1,l)}+1,M] = S[M_1(\mathbf{X})+1,M]$. Now, $\pi_1(I_\ell(\mathbf{X})) = 1$ iff $\ell \in S[0,m_1(\mathbf{X})] \cap S[M_1(\mathbf{X}),M] = S[M_1(\mathbf{X})+1,m_1(\mathbf{X})]$ and (9) follows from (5). □

Note that (9) becomes (6) when $\pi_1(\mathbf{x})$ in Proposition 3 has only one true vector (this happens whenever the number of literals of $\pi_1(\mathbf{x})$, $q_1 + r_1$, is N). In general, $\pi_1(\mathbf{x})$ has $2^{N-(q_1+r_1)}$ true vectors, and the TBF associated with $f(\mathbf{x}) = \pi_1(\mathbf{x})$ can also be expressed by using (7):

$$\text{TBF}_{\pi_1}(\mathbf{X}) = \max\{0, m_1(\mathbf{X}) - M_1(\mathbf{X})\} \quad (10.a)$$

$$= \sum_{i=1}^{K} \text{TBF}_{f_i}(\mathbf{X}), \quad (10.b)$$

where $\text{TBF}_{\pi_1}(\mathbf{X})$ is a multilevel representation of $\pi_1(\mathbf{x})$, $K = 2^{N-(q_1+r_1)}$, and $\text{TBF}_{f_i}(\mathbf{X})$ was defined in Proposition 2. In general, (10.a) is simpler than (10.b) because $K \ge 1$ and each term in (10.b) is given by (6).

Proposition 4 (Representation Based on SOP): Suppose that a given Boolean function $f(\cdot)$ has P products $f(\mathbf{x}) = \pi_1(\mathbf{x}) + \cdots + \pi_P(\mathbf{x})$. Then

$$\text{TBF}_f(\mathbf{X}) = \left|\bigcup_{i=1}^{P} S[M_i(\mathbf{X})+1, m_i(\mathbf{X})]\right|, \quad (11)$$

where $|A|$ is the number of elements in a set A.

Proof: Since $f(\cdot)$ is the logical sum of P products, $f(I_\ell(\mathbf{X}))$ produces 1 iff $\ell \in \bigcup_{i=1}^{P}(S[0,m_i(\mathbf{X})] \cap S[M_i(\mathbf{X})+1,M]) = \bigcup_{i=1}^{P} S[M_i(\mathbf{X})+1,m_i(\mathbf{X})]$. Equation (11) follows from (5). □

For the case of stack filters, since $f(\cdot)$ is a positive Boolean function, $r_i = 0$ and $M_i(\mathbf{X}) = 0$ for all i, and thus, $S[M_i(\mathbf{X})+1,m_i(\mathbf{X})]$ is simplified to $S[M_i(\mathbf{X})+1,m_i(\mathbf{X})] = S[1,m_i(\mathbf{X})]$, and $\bigcup_{i=1}^{P} S[M_i(\mathbf{X})+1,m_i(\mathbf{X})] = \bigcup_{i=1}^{P} S[1,m_i(\mathbf{X})] = S[1,\beta]$, where $\beta = \max\{m_j(\mathbf{X}), j = 1,2,\ldots,P\} = \max\{\min\{X_{p(j,l)}, l = 1,2,\ldots,q_j\}, j = 1,2,\ldots,P\}$. Therefore, we have $\text{TBF}_f(\mathbf{x}) = \beta$, which is consistent with the well-known fact that the output of a stack filter can be represented as the maximum of local minima.

It should be pointed out that $|\bigcup_{i=1}^{P} S[M_i(\mathbf{X})+1,m_i(\mathbf{X})]| \ne \sum_{i=1}^{P} |S[M_i(\mathbf{X})+1,m_i(\mathbf{X})]|$ unless $S[M_i(\mathbf{X})+1,m_i(\mathbf{X})], i = 1,\ldots,P$ are disjoint from each other. Since $|S[M_i(\mathbf{X})+1,m_i(\mathbf{X})]| = \max\{0,m_i(\mathbf{X}) - M_i(\mathbf{X})\} = \text{TBF}_{\pi_i}(\mathbf{X})$, then

$$\text{TBF}_f(\mathbf{X}) \ne \sum_{i=1}^{P} \text{TBF}_{\pi_i}(\mathbf{X}) \quad (12)$$

unless $S[M_i(\mathbf{X})+1,m_i(\mathbf{X})]$ are disjoint. The example below illustrates (11) and (12).

Example 4: Consider $f(\mathbf{x}) = x_1\bar{x}_2 + x_1x_3$ again. In this case, $P = 2$. If we let $\pi_1(\mathbf{x}) = x_1\bar{x}_2$ and $\pi_2(\mathbf{x}) = x_1x_3$, then $q_1 = n_1 = 1, q_2 = 2$, $n_2 = 0$, $p(1,1) = 1, n(1,1) = 2, p(2,1) = 1$, and $p(2,2) = 3$. Now, we get $m_1(\mathbf{X}) = X_1, m_2(\mathbf{X}) = \min\{X_1,X_3\}, M_1(\mathbf{X}) = X_2$, and $M_2(\mathbf{X}) = 0$. Thus, $S[M_1(\mathbf{X})+1,m_1(\mathbf{X})] = S[X_2+1,X_1]$ and $S[M_2(\mathbf{X})+1,m_2(\mathbf{X})] = S[1,\min\{X_1,X_3\}]$; $\text{TBF}_{\pi_1}(\mathbf{X}) = |S[X_2+1,X_1]| = \max\{0,X_1-X_2\}$ and $\text{TBF}_{\pi_2}(\mathbf{X}) = |S[1,\min\{X_1,X_3\}]| = \min\{X_1,X_3\}$. From (11), $\text{TBF}_f(\mathbf{X}) = |S[X_2+1,X_1] \cup S[1,\min\{X_1,X_3\}]|$. Since

$\pi_1(\mathbf{x}) \cdot \pi_2(\mathbf{x}) = x_1\bar{x}_2x_3$ and $\pi_1(I_\ell(\mathbf{X})) \times \pi_2(I_\ell(\mathbf{X})) = 1$ iff $\ell \in S[X_2+1, \min\{X_1, X_3\}]$, we get $S[X_2+1, X_1] \cap S[1, \min\{X_1, X_3\}] = S[X_2+1, \min\{X_1, X_3\}]$. Therefore, $\text{TBF}_f(\mathbf{X}) = |S[X_2+1, X_1]| + |S[1, \min\{X_1, X_3\}]| - |S[X_2+1, \min\{X_1, X_3\}]| = \max\{0, X_1 - X_2\} + \min\{X_1, X_3\} - \max\{0, \min\{X_1, X_3\} - X_2\}$. Obviously, $\text{TBF}_f(\mathbf{X}) \neq \text{TBF}_{\pi_1}(\mathbf{X}) + \text{TBF}_{\pi_2}(\mathbf{X})$. The TBF representation in this example yields the ordering-output table in Table I, and it is equivalent to that of Example 3. □

The TBF expression in (11) is inconvenient to use because evaluating the intersections among $S[M_i(\mathbf{X})+1, m_i(\mathbf{X})], i = 1, \ldots, P$ is tedious. In general, the number of possible intersections is $\sum_{i=2}^{P} \binom{P}{i} = 2^P - (P+1)$, and the TBF expression in (11) may have $2^P - (P+1) + P = 2^P - 1$ terms. This indicates that the expression in (11) may be lengthier than that in (7). Next, we obtain a simpler expression by finding a condition under which $S[M_i(\mathbf{X})+1, m_i(\mathbf{X})]$'s are disjoint.

Lemma 1: If $\pi_i(\mathbf{x}), i = 1, \ldots, P$ are mutually exclusive, then $S[M_i(\mathbf{X})+1, m_i(\mathbf{X})], i = 1, \ldots, P$ are disjoint from each other.

Proof: Consider the sets of true vectors $V(\pi_i)$ and $V(\pi_j)$. Assume $\pi_i(\mathbf{x})$ and $\pi_j(\mathbf{x})$ are mutually exclusive so that $V(\pi_i) \cap V(\pi_j) = \emptyset$. From the proof of Proposition 3, we know that $I_\ell(\mathbf{X}) \in V(\pi_i)$ iff $\ell \in S[M_i(\mathbf{X})+1, m_i(\mathbf{X})]$ and $I_\ell(\mathbf{X}) \in V(\pi_j)$ iff $\ell \in S[M_j(\mathbf{X})+1, m_j(\mathbf{X})]$. Suppose $S[M_i(\mathbf{X})+1, m_i(\mathbf{X})] \cap S[M_j(\mathbf{X})+1, m_j(\mathbf{X})] \neq \emptyset$. Then, there exists a level ℓ for which $I_\ell(\mathbf{X}) \in V(\pi_i)$ and $I_\ell(\mathbf{X}) \in V(\pi_j)$. Thus, $V(\pi_i) \cap V(\pi_j) \neq \emptyset$, which is a contradiction. This completes the proof. □

Proposition 5 (Representation Based on SOMEP): Consider $f(\mathbf{x}) = \pi_1(\mathbf{x}) + \cdots + \pi_P(\mathbf{x})$. If $\pi_i(\mathbf{x})$ are mutually exclusive, then

$$\text{TBF}_f(\mathbf{X}) = \sum_{i=1}^{P} \text{TBF}_{\pi_i}(\mathbf{X}) = \sum_{i=1}^{P} \max\{0, m_i(\mathbf{X}) - M_i(\mathbf{X})\}. \quad (13)$$

This proposition is a direct consequence of Lemma 1. A SOP Boolean expression consisting of mutually exclusive product terms is called the *sum of mutually exclusive products* (SOMEP). An arbitrary Boolean function can be easily converted into a SOMEP form; an algorithm for the conversion is presented in Appendix. The expression in (13) clearly shows that only complemented literals can have $(-)$ sign in the output representation of a TBF. The number of mutually exclusive products of any Boolean function, which is P in (13), is always less than or equal to the number of its true vectors K. Therefore, (13) is preferable to (7). In addition, it is usually simpler than (11).

Example 5: Consider again $f(\mathbf{x}) = x_1\bar{x}_2 + x_1x_3$. By applying the algorithm in Appendix, we obtain a SOMEP expression $f(\mathbf{x}) = x_1\bar{x}_2 + x_1x_2x_3$. From (13), $\text{TBF}_f(\mathbf{X}) = \max\{0, X_1 - X_2\} + \min\{X_1, X_2, X_3\}$. Comparing this result with those of Examples 3 and 4 indicates that (13) can provide a simpler expression than can (7) and (11). □

An alternative to (13) can be obtained from the product-of-sums (POS) Boolean expressions. The TBF representation based on POS expression is similar to that in (13) and will not be considered further.

B. Representing TBF's in Terms of Ordered Input Data

Consider $S[1, M]$, which is the range of summation in (3), or equivalently the range of level ℓ's in (5). We decompose $S[1, M]$ into a union of subintervals: $S[1, M] = \bigcup_{i=0}^{N} S[X_{(i+1)}+1, X_{(i)}]$ for any input $\mathbf{X}$. Note that $\{S[X_{(i+1)}+1, X_{(i)}], i = 0, 1, \ldots, N\}$ are disjoint since $X_{(i)} \geq X_{(i+1)}$ and $S[X_{(i+1)}+1, X_{(i)}] = \emptyset$ when $X_{(i)} = X_{(i+1)}$. Thus, (3) can be written as

$$\text{TBF}_f(\mathbf{X}) = \sum_{\ell \in S[1,M]} f(I_\ell(\mathbf{X})) = \sum_{i=0}^{N} \sum_{\ell \in S[X_{(i+1)}+1, X_{(i)}]} f(I_\ell(\mathbf{X})), \quad (14)$$

where in the case of $X_{(i)} = X_{(i+1)}$, $\sum_{\ell \in S[X_{(i+1)}+1, X_{(i)}]} f(I_\ell(\mathbf{x})) = 0$ since $S[X_{(i+1)}+1, X_{(i)}] = \emptyset$. In the following, we introduce two lemmas that lead us to another TBF representation.

Lemma 2: If $X_{(i)} > X_{(i+1)}$, $i = 0, 1, \ldots, N$ for an input $\mathbf{X}$, then

$$I_\ell(\mathbf{X}) = I_{X_{(i)}}(\mathbf{X}) \quad (15)$$

for all $\ell \in S[X_{(i+1)}+1, X_{(i)}]$.

Proof: For all $\ell \in S[X_{(i+1)}+1, X_{(i)}]$, $I_\ell(X_j) = 1$ if $X_j \geq X_{(i)}$ and 0 if $X_j \leq X_{(i+1)}$; thus, (15) follows. □

Lemma 3: For any input $\mathbf{X}$

$$\sum_{\ell \in S[X_{(i+1)}+1, X_{(i)}]} f(I_\ell(\mathbf{X})) = f(I_{X_{(i)}}(\mathbf{X}))C_i \quad (16)$$

where $C_i = X_{(i)} - X_{(i+1)}, i = 0, 1, \ldots, N$.

Proof: Assume that $X_{(i)} > X_{(i+1)}$. Then, from Lemma 2, $\sum_{\ell \in S[X_{(i+1)}+1, X_{(i)}]} f(I_\ell(\mathbf{X})) = \sum_{\ell \in S[X_{(i+1)}+1, X_{(i)}]} f(I_{X_{(i)}}(\mathbf{X})) = f(I_{X_{(i)}}(\mathbf{X}))(X_{(i)} - X_{(i+1)}) = f(I_{X_{(i)}}(\mathbf{X}))C_i$. If $X_{(i)} = X_{(i+1)}$, then $C_i = X_{(i)} - X_{(i+1)} = 0$, and both sides of (16) are zero. Thus, (16) holds. □

Now, we express the TBF in terms of C_i's.

Proposition 6 (Representation Based on Ordered Input): The output of a TBF with a Boolean function $f(\mathbf{x})$ can be expressed as

$$\text{TBF}_f(\mathbf{X}) = \sum_{i=0}^{N} f(I_{X_{(i)}}(\mathbf{X}))C_i. \quad (17)$$

This result is obtained by using (16) in (14). Note that $I_{X_{(i)}}(\mathbf{X})$ in (17) is dependent on the input ordering. Thus, (17) shows that a TBF can be expressed as an adaptive linear combination of ordered input data.

Example 6: For $f(\mathbf{X}) = x_1\bar{x}_2 + x_1x_3$, $\mathrm{TBF}_f(\mathbf{X}) = \sum_{i=0}^{3} f(I_{X_{(i)}}(\mathbf{X}))C_i$ from (17). This expression cannot be simplified further unless input orderings are given. Consider an input ordering $0 < X_2 < X_1 = X_3 < M$. Then, $f(I_M(\mathbf{X})) = f(0,0,0) = 0$, $f(I_{X_{(1)}}(\mathbf{X})) = f(I_{X_{(2)}}(\mathbf{X})) = f(1,0,1) = 1$, and $f(I_{X_{(3)}}(\mathbf{X})) = f(1,1,1) = 1$. Thus, $\mathrm{TBF}_f(\mathbf{X}) = C_1 + C_2 + C_3 = (X_3 - X_1) + (X_1 - X_2) + X_2 = X_3$ where we set $X_{(1)} = X_3$ and $X_{(2)} = X_2$.

Note that the same result is obtained when we set $X_{(1)} = X_1$ and $X_{(2)} = X_3$. It can be seen that the expression derived in this example also produces the ordering-output table in Table I. □

For the case of positive Boolean functions (stack filters), (17) can be simplified further as shown below.

Proposition 7 (Stack Filters): The output of a TBF with a positive Boolean function $f(\mathbf{x})$ can be expressed as

$$\mathrm{TBF}_f(\mathbf{X}) = X_{(m)} \tag{18}$$

where $m = \min\{i \mid f(I_{X_{(i)}}(\mathbf{X})) = 1, i = 0,1,\ldots,N\}$.

Proof: If $f(I_{X_{(i)}}(\mathbf{X})) = 1$ for some $i, 0 \leq i \leq N$, then $f(I_{X_{(k)}}(\mathbf{X})) = 1$ for all $k = i, i+1,\ldots,N$ due to the stacking property of a positive Boolean function [2]. Thus, $\mathrm{TBF}_f(\mathbf{X}) = C_m + C_{m+1} + \cdots + C_N = (X_{(m)} - X_{(m+1)}) + (X_{(m+1)} - X_{(m+2)}) + \cdots + (X_{(N)} - 0) = X_{(m)}$. □

Note that $f(I_{X_{(m)}}(\mathbf{X})) = 1$ and $f(I_{X_{(m)}+1}(\mathbf{X})) = 0$ in (18). Due to the stacking property, $f(I_\ell(\mathbf{X})) = 0$ for all $\ell \geq X_{(m)} + 1$. Therefore, $\mathrm{TBF}_f(\mathbf{X}) = X_{(m)} = \max\{\ell \mid f(I_\ell(\mathbf{X})) = 1\}$, which is a result derived in [2].

Example 7: Suppose $f(\mathbf{x}) = x_1x_3 + x_2$ and the input ordering is $0 < X_3 < X_1 < X_2 < M$. Since $f(I_M(\mathbf{X})) = f(0,0,0) = 0$ and $f(I_{X_{(1)}}(\mathbf{X})) = f(I_{X_2}(\mathbf{X})) = f(0,1,0) = 1$, we get $m = 1$ and $\mathrm{TBF}_f(\mathbf{X}) = X_{(1)} = X_2$. □

In (17) and (18), evaluating the Boolean expression $f(I_{X_{(i)}}(\mathbf{X}))$ is often burdensome because $f(\mathbf{x})$ is usually very long and cumbersome. For the case of LS TBF's, the evaluation of $f(\cdot)$ is not required, as shown below.

Proposition 8 (Linearly Separable TBF): Consider an LS Boolean function $f(\mathbf{x})$, which is defined as (4). The LS TBF corresponding to the $f(\mathbf{x})$ can be expressed as

$$\mathrm{TBF}_f(\mathbf{X}) = \sum_{i=0}^{N} I_T(R_i)C_i, \tag{19}$$

where $R_0 \equiv 0$, $R_i = \sum_{j=1}^{i} w_{(j)}$ for $i = 1,\ldots,N$, $w_{(j)}$ is the weight associated with $X_{(j)}$ and $I_T(R_i) = 1$ if $R_i \geq T$ and 0 otherwise.

Proof: From (15), we get $(y_1,\ldots,y_N) \equiv I_\ell(\mathbf{X}) = I_{X_{(i)}}(\mathbf{X})$ for all $\ell \in S[X_{(i+1)} + 1, X_{(i)}] \neq \emptyset$, $i = 0,1,\ldots,N$, where $y_{(k)} = 1$ if $k \leq i$ and 0 otherwise. Thus, $f(I_{X_{(i)}}(\mathbf{X})) = I_T(\sum_{j=1}^{N} w_jy_j) = I_T(\sum_{j=1}^{N} w_{(j)}y_{(j)}) = I_T(w_{(1)} + w_{(2)} + \cdots + w_{(i)}) = I_T(R_i)$, and (19) follows from (17). If $S[X_{(i+1)} + 1, X_{(i)}] = \emptyset$, i.e., $X_{(i)} = X_{(i+1)}$, $i = 0,1,\ldots,N$, then $f(I_{X_{(i)}}(\mathbf{X}))$ does not matter in (17) since $C_i = 0$. This completes the proof. □

Example 8: Suppose $f(\cdot)$ is an LS Boolean function with $T = 0$ and $(w_1,w_2,w_3) = (-1,1,-1)$. Note that (19) is not simplified further unless the input ordering is specified. Consider an input ordering $X_1 \leq X_2 \leq X_3$. Then, we get $w_{(1)} = w_3 = -1$, $w_{(2)} = w_2 = 1$, and $w_{(3)} = w_1 = -1$. From (19), $\mathrm{TBF}_f(\mathbf{X}) = I_0(0)C_0 + I_0(-1)C_1 + I_0(-1+1)C_2 + I_0(-1+1-1)C_3 = C_0 + C_2 = (M - X_{(1)}) + (X_{(2)} - X_{(3)})$. □

The implementation of an LS TBF using (19) should be simpler than that of a TBF using (17). The computational complexity associated with these filtering will be discussed in Section V. For WOS filters having nonnegative w_i and T, we can see that if $R_i \geq T$ for any $i = 0,1,\ldots,N-1$, then $R_j \geq T$ for all $j = i+1,\ldots,N$. Therefore, (19) can be reduced to

$$\mathrm{TBF}_f(\mathbf{X}) = \sum_{i=k}^{N} C_i = X_{(k)} \tag{20}$$

where $k = \min\{i \mid R_i \geq T, i = 0,1,\ldots,N\}$. The expression in (20) is a definition of the WOS filter [5].

In general, $f(I_{X_{(i)}}(\mathbf{X}))$ in (17) produces 0 or 1, depending on the input $\mathbf{X}$. For some special cases, however, it is independent of $\mathbf{X}$, and the output of a TBF may be represented as

$$\mathrm{TBF}_f(\mathbf{X}) = \sum_{i=0}^{N} b_iC_i, \tag{21}$$

where b_i are binary constants. A TBF that can be expressed as in (21) will be called a *fixed* TBF. Next, we present a sufficient condition for $f(\mathbf{x})$ being a Boolean function of a fixed TBF.

Proposition 9 (Fixed TBF): If a Boolean function $f(\cdot)$ satisfies $f(\mathbf{x}) = f(\mathbf{y})$ whenever binary input vectors $\mathbf{x}$ and $\mathbf{y}$ have the same number of 1's, i.e., $w_H(\mathbf{x}) = w_H(\mathbf{y})$, then the TBF corresponding to $f(\cdot)$ becomes a fixed TBF in (21), and each coefficient b_i, $i = 1,\ldots,N$ is given by the output of the Boolean function $f(\mathbf{x})$ with $\mathbf{x}$ having i 1's, i.e., $w_H(\mathbf{x}) = i$.

Proof: If $C_i \neq 0$, i.e., $X_{(i)} > X_{(i+1)}$, then $w_H(I_{X_{(i)}}(\mathbf{X})) = i$, and (21) follows from (17). When $C_i = 0$, (21) holds regardless of b_i. This completes the proof. □

Example 9: Consider $f(\mathbf{x}) = x_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + x_1x_2x_3$. For this case, $f(\mathbf{x}) = b_i$ with $b_1 = b_3 = 1$ and $b_0 = b_2 = 0$. From (21), therefore, we get $\mathrm{TBF}_f(\mathbf{X}) = \sum_{i=0}^{3} b_iC_i = C_1 + C_3 = X_{(1)} - X_{(2)} + X_{(3)}$. □

The class of fixed TBF's encompasses the rank-order filters; a rank-order filter selecting $X_{(k)}$, where k is a fixed integer between 1 and N, is a fixed TBF with b_i equal to 0 if $i \leq k-1$ and 1 otherwise. On the other hand, a fixed TBF with $b_0 = 0$ is a particular case of the LOS filter that is defined as *LOS* $(\mathbf{X}) = \sum_{i=1}^{N} a_iX_{(i)}$, where a_i are real numbers. An LOS filter becomes a fixed TBF iff it produces either 0 or 1 for any binary input vector. Alternatively, we can say that an LOS filter becomes a fixed TBF iff it can be expressed as in (21). Note that the coefficients $\{a_i\}$ of an LOS filter should be either 0 or 1 or -1 when the filter is a fixed TBF.

Both the expressions in (13) and (17) are useful for investigating the behavior of TBF's. In the following section, some properties of the TBF are derived by using these expressions.

TABLE II
ORDERING-OUTPUT TABLE FOR $f(\mathbf{x}) = x_1\bar{x}_2 + \bar{x}_1\bar{x}_3$

orderings	output=max$\{0, X_1 - X_2\} + \{M - \max\{X_1, X_3\}\}$
$X_1 \le X_2 \le X_3$	$0 + (M - X_3) = M - X_3$
$X_1 \le X_3 \le X_2$	$0 + (M - X_3) = M - X_3$
$X_2 \le X_1 \le X_3$	$X_1 - X_2 + (M - X_3) = M - X_3 + X_1 - X_2$
$X_2 \le X_3 \le X_1$	$X_1 - X_2 + (M - X_1) = M - X_2$
$X_3 \le X_1 \le X_2$	$0 + (M - X_1) = M - X_1$
$X_3 \le X_2 \le X_1$	$X_1 - X_2 + (M - X_1) = M - X_2$

IV. PROPERTIES OF THE TBF

The TBF does not obey the superposition principle, and it is nonlinear; moreover, some TBF's are neither *scale-invariant* nor *translation-invariant*. In what follows, the translation and scaling of $\mathbf{X}$ will be denoted by $\mathbf{X}+c = (X_1+c, \ldots, X_N+c)$ and $a\mathbf{X} = (aX_1, \ldots, aX_N)$, respectively, where c and a are constants. Note that if $a\mathbf{X}+c \in D^N$, then $\text{TBF}_f(a\mathbf{X}+c) \in D = \{0, 1, \ldots, M\}$ since $f(I_\ell(\mathbf{X}))$ in (3) produces either 0 or 1. The following lemma is useful for investigating scale- and translation-invariant properties of TBF's.

Lemma 4: The TBF representation in (17) can be rewritten as

$$\sum_{i=0}^{N} f(I_{X_{(i)}}(\mathbf{X}))C_i = f(\mathbf{0})C_0 + \sum_{i=1}^{N-1} f(I_{X_{(i)}}(\mathbf{X}))C_i + f(\mathbf{1})C_N. \quad (22)$$

Proof: It suffices to show that $f(I_{X_{(0)}}(\mathbf{X}))C_0 = f(\mathbf{0})C_0$ and $f(I_{X_{(N)}}(\mathbf{X}))C_N = f(\mathbf{1})C_N$, which are obvious if $C_0 = 0$ and $C_N = 0$, respectively. If $C_0 > 0$, i.e., $X_{(0)} = M > X_{(1)}$, then $I_{X_{(0)}}(\mathbf{X}) = \mathbf{0}$. We have $I_{X_{(N)}}(\mathbf{X}) = \mathbf{1}$ for any cases. This completes the proof. □

In the properties stated below, we shall see that some TBF's are neither translation invariant nor scale invariant.

Property 1 (Translation-Invariance): Suppose that $\mathbf{X}+c \in D^N$. Then

$$\text{TBF}_f(\mathbf{X}+c) = \text{TBF}_f(\mathbf{X}) + c(f(\mathbf{1}) - f(\mathbf{0})). \quad (23)$$

Proof: Let $\mathbf{X}' = \mathbf{X}+c$ and $C_i' = X_{(i)}' - X_{(i+1)}'$. Then, $C_0' = M - (X_{(1)} + c)$, $C_N' = X_{(N)} + c$, and $C_i' = (X_{(i)} + c) - (X_{(i+1)} + c) = C_i$ for $i = 1, \ldots, N-1$. Since $I_{X_{(i)}'}(\mathbf{X}') = I_{X_{(i)}}(\mathbf{X})$, (22) yields $\text{TBF}_f(\mathbf{X}+c) = \text{TBF}_f(\mathbf{X}') = f(\mathbf{0})C_0' + \sum_{i=1}^{N-1} f(I_{X_{(i)}'}(\mathbf{X}'))C_i' + f(\mathbf{1})C_N' = f(\mathbf{0})(M - X_{(1)} - c) + \sum_{i=1}^{N-1} f(I_{X_{(i)}}(\mathbf{X}))C_i + f(\mathbf{1})(X_{(N)} + c) = f(\mathbf{0})(M - X_{(1)}) + \sum_{i=1}^{N-1} f(I_{X_{(i)}}(\mathbf{X}))C_i + f(\mathbf{1})X_{(N)} + c(f(\mathbf{1}) - f(\mathbf{0})) = \text{TBF}_f(\mathbf{X}) + c(f(\mathbf{1}) - f(\mathbf{0}))$. □

Property 1 indicates that a TBF is translation invariant iff $f(\mathbf{1}) = 1$ and $f(\mathbf{0}) = 0$. If $f(\mathbf{1}) = f(\mathbf{0})$, then $\text{TBF}_f(\mathbf{X}+c) = \text{TBF}_f(\mathbf{X})$, and thus, dc components of the input are completely removed, and the TBF is not translation invariant. TBF's with the dc-removal property may be useful for applications such as edge detection and DPCM. In fact, the edge detectors in [15] and [16] are TBF's with the dc-removal property. The examples below present some other TBFs that are not translation invariant.

Example 10: Consider the range estimator in Example 1. Obviously, this estimator removes dc components. This can be also seen by considering its Boolean function $f(\mathbf{x}) = x_1\bar{x}_2 + x_2\bar{x}_3 + x_3\bar{x}_1$. Since $f(\mathbf{1}) = f(\mathbf{0}) = 0$, the estimator has the dc-removal property and is not translation invariant. □

Example 11: Consider $f(\mathbf{x}) = x_1\bar{x}_2 + \bar{x}_1\bar{x}_3$. Since $f(\mathbf{1}) = 0$ and $f(\mathbf{0}) = 1$, the TBF corresponding to $f(\mathbf{x})$ satisfies $\text{TBF}_f(\mathbf{X}+c) = \text{TBF}_f(\mathbf{X}) - c$. In fact, since $x_1\bar{x}_2$ and $\bar{x}_1\bar{x}_3$ are exclusive, (13) gives $\text{TBF}_f(\mathbf{X}) = \max\{0, X_1 - X_2\} + (M - \max\{X_1, X_3\})$ and $\text{TBF}_f(\mathbf{X}+c) = \max\{0, (X_1+c) - (X_2+c)\} + (M - \max\{X_1+c, X_3+c\}) = \max\{0, X_1 - X_2\} + (M - \max\{X_1, X_3\}) - c$. Therefore, $\text{TBF}_f(\mathbf{X}+c) = \text{TBF}_f(\mathbf{X}) - c$. The reason why this equality holds becomes obvious if we consider the ordering output table in Table II. Each output in the table is either $M - X_j$ or $M - X_3 + X_1 - X_2$, and thus, the equality should hold. □

If $f(\cdot)$ is an LS Boolean function defined by (4), $f(\mathbf{1}) = 1$ iff $\sum_{i=1}^{N} w_i \ge T$, and $f(\mathbf{0}) = 0$ iff $T > 0$. Therefore, an LS TBF is translation invariant iff $\sum_{i=1}^{N} w_i \ge T > 0$.

Property 2 (Scale Invariance): Suppose $a\mathbf{X} \in D^N$. Then

$$\text{TBF}_f(a\mathbf{X}) = a\text{TBF}_f(\mathbf{X}) + M(1-a)f(\mathbf{0}). \quad (24)$$

Proof: Let $\mathbf{X}' = a\mathbf{X}$ and $C_i' = X_{(i)}' - X_{(i+1)}'$. Then, $C_0' = M - aX_{(1)}$, $C_N' = a\mathbf{X}_{(N)}$, and $C_i' = aX_{(i)} - aX_{(i+1)} = aC_i$ for $i = 1, \ldots, N-1$. Since $I_{X_{(i)}'}(\mathbf{X}') = I_{X_{(i)}}(\mathbf{X})$, (22) yields $\text{TBF}_f(a\mathbf{X}) = \text{TBF}_f(\mathbf{X}') = f(\mathbf{0})C_0' + \sum_{i=1}^{N-1} f(I_{X_{(i)}'}(\mathbf{X}'))C_i' + f(\mathbf{1})C_N' = f(\mathbf{0})(M - aX_{(1)}) + \sum_{i=1}^{N-1} f(I_{X_{(i)}}(\mathbf{X}))aC_i + f(\mathbf{1})aX_{(N)} = af(\mathbf{0})(M - X_{(1)}) + a\sum_{i=1}^{N-1} f(I_{X_{(i)}}(\mathbf{X}))C_i + af(\mathbf{1})X_{(N)} + Mf(\mathbf{0}) - aMf(\mathbf{0}) = a\text{TBF}_f(\mathbf{X}) + M(1-a)f(\mathbf{0})$. □

This property indicates that a TBF is scale invariant only when $f(\mathbf{0}) = 0$. Thus, an LS TBF is scale invariant if $T > 0$. Note that if $f(\mathbf{0}) = 1$, the corresponding TBF is neither translation invariant nor scale invariant (for example, see the TBF in Example 11). Combining Properties 1 and 2, we draw the following conclusion.

Corollary 1: Let $a\mathbf{X}+c \in D^N$. Then

$$\text{TBF}_f(a\mathbf{X}+c) = a\text{TBF}_f(\mathbf{X}) + c \quad (25)$$

iff $f(\mathbf{1}) = 1$ and $f(\mathbf{0}) = 0$.

Since any positive Boolean function $f(\mathbf{x})$ satisfies $f(\mathbf{1}) = 1$ and $f(\mathbf{0}) = 0$, all stack filters are translation and scale invariant. Next, we derive some properties that are often useful for obtaining the multilevel TBF representations in (7) and (13).

TABLE III
ORDERING-OUTPUT TABLE FOR $f(\mathbf{x}) = x_1x_2 + x_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3$

orderings	output=$(\mathrm{TBF_f(X)})$	output=$(\mathrm{TBF_f(M-X)})$
$X_1 \le X_2 \le X_3$	$X_1 + 0 + 0 = X_1$	$M - X_2 + (X_2 - X_1) + 0 = M - X_1$
$X_1 \le X_3 \le X_2$	$X_1 + 0 + (X_2 - X_3) = X_1 + X_2 - X_3$	$M - X_2 + (X_3 - X_1) + 0 = M - (X_1 + X_2 - X_3)$
$X_2 \le X_1 \le X_3$	$X_2 + 0 + 0 = X_2$	$M - X_1 + 0 + (X_1 - X_2) = M - X_2$
$X_2 \le X_3 \le X_1$	$X_2 + (X_1 - X_3) + 0 = X_1 + X_2 - X_3$	$M - X_1 + 0 + (X_3 - X_2) = M - (X_1 + X_2 - X_3)$
$X_3 \le X_1 \le X_2$	$X_1 + 0 + (X_2 - X_1) = X_2$	$M - X_2 + 0 + 0 = M - X_2$
$X_3 \le X_2 \le X_1$	$X_2 + (X_1 - X_2) + 0 = X_1$	$M - X_1 + 0 + 0 = M - X_1$

When a multilevel representation of a Boolean function $f(\mathbf{x})$ is given, the multilevel representation corresponding to the *dual* of $f(\mathbf{x})$ can be obtained easily.

Property 3 (Duality): Let $f^d(\mathbf{x})$ denote the dual function of a Boolean function $f(\mathbf{x})$, $f^d(\mathbf{x}) = \bar{f}(\bar{\mathbf{x}})$. Then

$$\mathrm{TBF}_{f^d}(\mathbf{X}) = M - \mathrm{TBF}_f(M - \mathbf{X}). \qquad (26)$$

Proof: Suppose $C_j > 0$, $j = 0, 1, \ldots, N$ for an input $\mathbf{X}$. Then, we have $X_{(0)}, X_{(1)}, \ldots, X_{(j)} > X_{(j+1)}, \ldots, X_{(N)}, X_{(N+1)}$. Let $f'(\mathbf{x}) = f(\bar{\mathbf{x}})$ and $\mathbf{X}' = M - \mathbf{X} = (M - X_1, \ldots, M - X_N)$. Then, we also have $X'_{(0)}, X'_{(1)}, \ldots, X'_{(j)} < X'_{(j+1)}, \ldots, X'_{(N)}, X'_{(N+1)}$. Thus, we obtain $\overline{I_{X_{(j)}}(\mathbf{X})} = I_{M - X_{(j+1)}}(M - \mathbf{X}) = I_{X'_{(N-j)}}(\mathbf{X}')$, and from (17), $\mathrm{TBF}_{f'}(\mathbf{X}) = \sum_{i=0}^{N} f(\overline{I_{X_{(i)}}(\mathbf{X})})C_i = \sum_{i=0}^{N} f(I_{X'_{(N-i)}}(\mathbf{X}'))C_i = \sum_{i=0}^{N} f(I_{X'_{(i)}}(\mathbf{X}'))C_{N-i}$, which also holds for $C_i = 0$. Since $C_{N-i} = X_{(N-i)} - X_{(N-i+1)} = (M - X_{(i)}) - (M - X_{(i+1)}) = X'_{(i)} - X'_{(i+1)} \equiv C'_i$ for all i, $\mathrm{TBF}_{f'}(\mathbf{X}) = \sum_{i=0}^{N} f(\overline{I_{X_{(i)}}(\mathbf{X})})C_i = \sum_{i=0}^{N} f(I_{X'_{(i)}}(\mathbf{X}'))C'_i = \mathrm{TBF}_f(\mathbf{X}') = \mathrm{TBF}_f(M - \mathbf{X})$. Since $\mathrm{TBF}_f(\mathbf{X}) + \mathrm{TBF}_{\bar{f}}(\mathbf{X}) = M$, $\mathrm{TBF}_{f^d}(\mathbf{X}) = \mathrm{TBF}_{\bar{f}'}(\mathbf{X}) = M - \mathrm{TBF}_{f'}(\mathbf{X}) = M - \mathrm{TBF}_f(M - \mathbf{X})$. □

If $f(\cdot)$ is *self dual*, $f(\mathbf{x}) = \bar{f}(\bar{\mathbf{x}})$, and then, (26) yields $\mathrm{TBF}_f(M - \mathbf{X}) = M - \mathrm{TBF}_f(\mathbf{X})$. Thus, we can say that a TBF corresponding to a self-dual Boolean function commutes with a signal inversion. Since Boolean functions corresponding to weighted median (WM) filters are self dual [18], all WM filters commute with a signal inversion.

Example 12: Consider a self-dual Boolean function $f(\mathbf{x}) = x_1x_2 + x_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3$. Since the three products of $f(\mathbf{x})$ are mutually exclusive, (13) yields $\mathrm{TBF}_f(\mathbf{X}) = \min\{X_1, X_2\} + \max\{0, X_1 - \max\{X_2, X_3\}\} + \max\{0, X_2 - \max\{X_1, X_3\}\}$. Thus, $\mathrm{TBF}_f(M - \mathbf{X}) = \min\{(M - X_1), (M - X_2)\} + \max\{0, (M - X_1) - \max\{(M - X_2), (M - X_3)\}\} + \max\{0, (M - X_2) - \max\{(M - X_1), (M - X_3)\}\} = M - \max\{X_1, X_2\} + \max\{0, \min\{X_2, X_3\} - X_1\} + \max\{0, \min\{X_1, X_3\} - X_2\}$. The ordering-output table in Table III shows $\mathrm{TBF}_f(M - \mathbf{X}) = M - \mathrm{TBF}_f(\mathbf{X})$. □

A filter that is expressed as an absolute difference between two stack filters is a TBF because it yields either 0 or 1 for binary inputs and obeys the threshold decomposition. The Boolean function of such a filter is derived in the following property.

Property 4 (Absolute Difference Between Two Stack Filters): Consider $\mathrm{TBF}_h(\mathbf{X}) = |\mathrm{TBF}_f(\mathbf{X}) - \mathrm{TBF}_g(\mathbf{X})|$, where $f(\mathbf{x})$ and $g(\mathbf{x})$ are positive Boolean functions ($\mathrm{TBF}_f(\mathbf{X})$ and $\mathrm{TBF}_g(\mathbf{X})$ are stack filters). Then, the Boolean function $h(\mathbf{x})$ of $\mathrm{TBF}_h(\mathbf{X})$ is given by $h(\mathbf{x}) = f(\mathbf{x})\bar{g}(\mathbf{x}) + \bar{f}(\mathbf{x})g(\mathbf{x})$.

Proof: Let $h_1(\mathbf{x}) = f(\mathbf{x})\bar{g}(\mathbf{x})$, $F = \mathrm{TBF}_f(\mathbf{X})$, and $G = \mathrm{TBF}_g(\mathbf{X})$. Then, $F = \max\{\ell \mid f(I_\ell(\mathbf{X})) = 1\}$ and $G = \max\{\ell \mid g(I_\ell(\mathbf{X})) = 1\}$. Since $h_1(I_\ell(\mathbf{X})) = f(I_\ell(\mathbf{X}))\bar{g}(I_\ell(\mathbf{X})) = 1$ only when $G < \ell \le F$, we get $\mathrm{TBF}_{h_1}(\mathbf{X}) = \max\{0, F - G\}$. Letting $h_2(\mathbf{x}) = \bar{f}(\mathbf{x})g(\mathbf{x})$, we get similarly $\mathrm{TBF}_{h_2}(\mathbf{X}) = \max\{0, G - F\}$. Since $f(\mathbf{x})\bar{g}(\mathbf{x})$ and $\bar{f}(\mathbf{x})g(\mathbf{x})$ are mutually exclusive, $\mathrm{TBF}_h(\mathbf{X}) = \mathrm{TBF}_{h_1}(\mathbf{X}) + \mathrm{TBF}_{h_2}(\mathbf{X}) = \max\{0, F - G\} + \max\{0, G - F\} = |F - G| = |\mathrm{TBF}_f(\mathbf{X}) - \mathrm{TBF}_g(\mathbf{X})|$. □

Example 13: Consider $\mathrm{TBF}_h(\mathbf{X}) = |$ *median* $\{X_1, X_2, X_3\}$ − *median* $\{X_4, X_5, X_6\}|$, which is the *difference-of-medians* operator [16]. Since the positive Boolean functions for median$\{X_1, X_2, X_3\}$ and median$\{X_4, X_5, X_6\}$ are $f(\mathbf{x}) = x_1x_2 + x_2x_3 + x_3x_1$ and $g(\mathbf{x}) = x_4x_5 + x_5x_6 + x_6x_4$, respectively, we get $h(\mathbf{x}) = f(\mathbf{x})\bar{g}(\mathbf{x}) + \bar{f}(\mathbf{x})g(\mathbf{x})$ from Property 4. The Boolean functions corresponding to the range or quasiranges [15] $X_{(i)} - X_{(N-i+1)}, 1 \le i \le \lfloor \frac{N}{2} \rfloor$ can be obtained in a similar manner. □

In general, a filter expressed as a linear combination of stack filters is not a TBF. It can be shown, however, that any TBF can be represented as a linear combination of stack filters. This result is derived by using the definition and the lemmas presented below.

Definition 1 (Stacking Property of True Vectors): Consider $V(f)$, which is the set of true vectors of a Boolean function $f(\mathbf{x})$. Let $V(f) = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$, and let $W_i(f)$ be the set of binary vectors that are greater than or equal to the true vector $\mathbf{x}_i, i = 1, \ldots, K$. Here, a binary vector $\mathbf{y} = (y_1, \ldots, y_N) \ge \mathbf{x} = (x_1, \ldots, x_N)$ if $y_j \ge x_j$ for all j. The set $V(f)$ is said to have the *stacking* property iff $V(f) = \bigcup_{i=1}^{K} W_i(f)$. □

For example, for $V(f) = \{(0,0,1), (1,0,1)\}$, we have $W_1(f) = \{(0,0,1), (0,1,1), (1,0,1), (1,1,1)\}$, and $W_2(f) = \{(1,0,1), (1,1,1)\}$ when $\mathbf{x}_1 = (0,0,1)$ and $\mathbf{x}_2 = (1,0,1)$. Thus, $V(f) \ne W_1(f) \cup W_2(f)$, and $V(f)$ does not possess the stacking property. The union of sets $\bigcup_{i=1}^{K} W_i(f)$ will be called the *stacking set* of $f(\mathbf{x})$. Of course, $\bigcup_{i=1}^{K} W_i(f) \supseteq V(f)$. The difference between the stacking set $\bigcup_{i=1}^{K} W_i(f)$ and $V(f)$ will be called the *complementary set* of $f(\mathbf{x})$ and denoted by $C(f)$: $\bigcup_{i=1}^{K} W_i(f) = V(f) \cup C(f)$ and $V(f) \cap C(f) = \emptyset$. For $V(f) = \{(0,0,1), (1,0,1)\}$ in the above example, the complementary set $C(f) = \{(0,1,1), (1,1,1)\}$. Obviously, $C(f) = \emptyset$ when $V(f)$ has the stacking property. Since a Boolean function $f(\mathbf{x})$ is positive iff $f(\mathbf{x}) = 1$ means

$f(\mathbf{y}) = 1$ for all $\mathbf{y} \geq \mathbf{x}$, we can say that $V(f)$ has the stacking property iff the corresponding Boolean function $f(\cdot)$ is positive.

Lemma 5: If $C(f) \neq \emptyset$, then $\min\{w_H(\mathbf{x}) \mid \mathbf{x} \in C(f)\} > \min\{w_H(\mathbf{x}) \mid \mathbf{x} \in V(f)\}$, where $w_H(\mathbf{x})$ is the number of 1's in $\mathbf{x}$.

Proof: Consider $W_i(f)$, which is the set of all binary vectors $\mathbf{y} \geq \mathbf{x}_i \in V(f)$, where $w_H(\mathbf{y}) > w_H(\mathbf{x}_i)$ if $\mathbf{y} \neq \mathbf{x}_i$. Since $C(f) = \{\mathbf{y} \mid \mathbf{y} \in \bigcup_{i=1}^{K} W_i(f)$ and $\mathbf{y} \notin V(f)\}$, we get the desired result. □

Now, we are ready to show that an arbitray TBF can be decomposed into a linear combination of stack filters.

Property 5 (TBF as a Sum of Stack Filters): Any $\mathrm{TBF}_f(\mathbf{X})$ can be expressed as a linear combination of stack filters:

$$\mathrm{TBF}_f(\mathbf{X}) = \sum_{i=1}^{Q} (-1)^{i-1} \mathrm{TBF}_{f_i}(\mathbf{X}), \tag{27}$$

where $1 \leq Q \leq N+1$, and each $f_i(\cdot)$ is a positive Boolean function with true vectors $V(f_i) = V(f'_{i-1}) \cup C(f'_{i-1}), i = 1, \ldots, Q$, where $f'_0(\mathbf{x}) = f(\mathbf{x})$ and $f'_i(\mathbf{x})$ is a Boolean function with $V(f'_i) = C(f'_{i-1})$.

Proof: Suppose that $f(\mathbf{x})$ is not positive. We determine the Boolean functions $f_1(\mathbf{x})$ and $f'_1(\mathbf{x})$ so that $V(f_1) = V(f) \cup C(f)$ and $V(f'_1) = C(f)$, respectively. The Boolean function $f_1(\mathbf{x})$ becomes positive because $V(f) \cup C(f)$ has the stacking property. Since $V(f) \cap C(f) = \emptyset$, we get, from (13), $\mathrm{TBF}_{f_1}(\mathbf{X}) = \mathrm{TBF}_f(\mathbf{X}) + \mathrm{TBF}_{f'_1}(\mathbf{X})$, and thus

$$\mathrm{TBF}_f(\mathbf{X}) = \mathrm{TBF}_{f_1}(\mathbf{X}) - \mathrm{TBF}_{f'_1}(\mathbf{X}). \tag{28}$$

If $f'_1(\mathbf{x})$ is positive, then we stop here, and $Q = 2$. If not, by repeating the procedure for obtaining (28), we decompose $\mathrm{TBF}_{f'_1}(\mathbf{X})$ into $\mathrm{TBF}_{f_2}(\mathbf{X})$ and $\mathrm{TBF}_{f'_2}(\mathbf{X})$, where $\mathrm{TBF}_{f_2}(\mathbf{X})$ is a stack filter. Continuing in this manner, we eventually get a positive Boolean function $f'_n(\mathbf{x})$, $n \leq N$. This is true because $\min\{w_H(\mathbf{x}) \mid \mathbf{x} \in C(f_i)\} > \min\{w_H(\mathbf{x}) \mid \mathbf{x} \in V(f_i)\}$ (Lemma 6), and the maximum number of 1's in $\mathbf{x}$ is N. The worst case, $n = N(Q = N+1)$, occurs when $\mathbf{0} \in V(f)$ and $\min\{w_H(\mathbf{x}) \mid \mathbf{x} \in C(f_N)\} = N$, i.e., $f'_N(\mathbf{x}) = x_1 x_2 \cdots x_N$. By setting $f_{n+1}(\mathbf{x}) = f'_n(\mathbf{x})$, we get $\mathrm{TBF}_f(\mathbf{X}) = \mathrm{TBF}_{f_1}(\mathbf{X}) - \mathrm{TBF}_{f_2}(\mathbf{X}) + \cdots (-1)^n \mathrm{TBF}_{f_{n+1}}(\mathbf{X})$. □

Example 14: For $f(\mathbf{x}) = x_1\bar{x}_2 + \bar{x}_2 x_3 + x_1 x_3$, $V(f) = \{(0,0,1), (1,0,0), (1,0,1), (1,1,1)\}$, and $C(f) = \{(0,1,1), (1,1,0)\}$. From $V(f_1) = V(f) \cup C(f)$ and $V(f'_1) = C(f)$, we obtain $f_1(\mathbf{x}) = x_1 + x_3$ and $f'_1(\mathbf{x}) = x_1 x_2 \bar{x}_3 + \bar{x}_1 x_2 x_3$, respectively. Since $f'_1(\mathbf{x})$ is not positive, we consider $C(f'_1) = \{(1,1,1)\}$. From $V(f_2) = V(f'_1) \cup C(f'_1)$ and $V(f'_2) = C(f'_1)$, we get $f_2(\mathbf{x}) = x_1 x_2 + x_2 x_3$ and $f'_2(\mathbf{x}) = x_1 x_2 x_3$, respectively, where $f'_2(\mathbf{x})$ is positive. Therefore, we set $f_3(\mathbf{x}) = f'_2(\mathbf{x})$ and $\mathrm{TBF}_f(\mathbf{X}) = \mathrm{TBF}_{f_1}(\mathbf{X}) - \mathrm{TBF}_{f_2}(\mathbf{X}) + \mathrm{TBF}_{f_3}(\mathbf{X})$, with $f_1(\mathbf{x}) = x_1 + x_3$, $f_2(\mathbf{x}) = x_1 x_2 + x_2 x_3$, and $f_3(\mathbf{x}) = x_1 x_2 x_3$. □

Property 5 suggests that a TBF be realized as a parallel connection of stack filters. When Q is reasonably small and an efficient algorithm for stack filtering is available, the parallel structure may be preferable to direct implementations of a TBF.

V. Implementation of the TBF

The TBF representations developed in Sections II and III lead to various implementations of a TBF. Among the representations, the one in (17) generally results in more efficient implementation than the others because the number of additions in (17) is always $N+1$, whereas K and P in (7) and (13) vary depending on Boolean functions and can be as large as $O(2^N)$. Therefore, in this section, we focus our attention to the implementation of a TBF based on (17), develop an algorithm for realizing a TBF, and compare the computational complexities of TBF"s, LS TBF's, and stack and WOS filters.

The output of a TBF is obtained from (17) through the following steps:

Step 1. Sort the data within the window and evaluate $C_i, i = 0, 1, \ldots, N$.

Step 2. Obtain $I_{X_{(i)}}(\mathbf{X})$ for each i.

Step 3. Evaluate $\sum_{i=0}^{N} f(I_{X_{(i)}}(\mathbf{X})) C_i$.

In Step 1, the input data inside a window can be sorted by applying well-known algorithms such as BUBBLESORT, MERGESORT, QUICKSORT [19], [20] and the running sorting algorithms in [21]. Among the sorting algorithms, the running sorting requires less computation and is preferable to the others when a TBF is realized on a general-purpose computer. On the other hand, BUBBLESORT has been used in the practical design of sorting circuits [22]–[24] due to its modular structure. In Step 2, the direct computation of $I_{X_{(i)}}(\mathbf{X})$'s requires N^2 multilevel comparisons. The computational load can be reduced significantly if we use the information on time indices of the sorted input data. For each input $X_j, j = 1, \ldots, N$, we define its location vector $\mathbf{r}_j = (r_j^1, \ldots, r_j^N)$, where $r_j^k = 1$ if $k = j$ and 0 otherwise. In addition, the location vector of $X_{(i)}$, which is denoted by $\mathbf{r}_{(i)}$, is defined as $\mathbf{r}_{(i)} = \mathbf{r}_j$ when $X_{(i)} = X_j$. For example, when $N = 3$, $\mathbf{r}_1 = (1,0,0)$, $\mathbf{r}_2 = (0,1,0)$, and $\mathbf{r}_3 = (0,0,1)$. If $X_2 < X_1 < X_3$, then $\mathbf{r}_{(1)} = \mathbf{r}_3 = (0,0,1)$, $\mathbf{r}_{(2)} = \mathbf{r}_1 = (1,0,0)$, and $\mathbf{r}_{(3)} = \mathbf{r}_2 = (0,1,0)$. Now, consider a set of binary vectors $\{\mathbf{z}_i, i = 0, 1, \ldots, N\}$ defined as $\mathbf{z}_i = \mathbf{r}_{(1)} + \mathbf{r}_{(2)} + \cdots + \mathbf{r}_{(i)}$, $i = 1, \ldots, N$, and $\mathbf{z}_0 = \mathbf{0}$, where $+$ performs componentwise addition. Note that $\mathbf{z}_i$ has i 1's and $N-i$ 0's and $\mathbf{z}_N = \mathbf{1}$. In the example described above, $\mathbf{z}_1 = (0,0,1)$, $\mathbf{z}_2 = (1,0,1)$, and $\mathbf{z}_3 = (1,1,1)$. The vector $\mathbf{z}_i$ can be evaluated recursively by using $\mathbf{z}_i = \mathbf{z}_{i-1} + \mathbf{r}_{(i)}$ with $\mathbf{z}_1 = \mathbf{r}_{(1)}$. The relation between $\mathbf{z}_i$ and $I_{X_{(i)}}(\mathbf{X})$ is observed below.

Observation 1: If $X_{(i)} > X_{(i+1)}$, then $I_{X_{(i)}}(\mathbf{X}) = \mathbf{z}_i$.

Proof: The kth element of $I_{X_{(i)}}(\mathbf{X})$ is equal to 1 iff $X_k \geq X_{(i)}$. The number of 1's in $I_{X_{(i)}}(\mathbf{X})$ is greater than or equal to i, and it becomes i when $X_{(i)} > X_{(i+1)}$. The kth element of $\mathbf{z}_i$ is equal to 1 iff $X_k \in \{X_{(1)}, X_{(2)}, \ldots, X_{(i)}\}$, and the number of 1's in $\mathbf{z}_i$ is always i. Thus, $I_{X_{(i)}}(\mathbf{X}) = \mathbf{z}_i$ when $X_{(i)} > X_{(i+1)}$. □

When $X_{(i)} = X_{(i+1)}$, $I_{X_{(i)}}(\mathbf{X})$ is different from $\mathbf{z}_i$ but $f(I_{X_{(i)}}(\mathbf{X}))C_i = f(\mathbf{z}_i)C_i$ because $C_i = X_{(i)} - X_{(i+1)} = 0$. Consequently, we obtain the following equality that is very

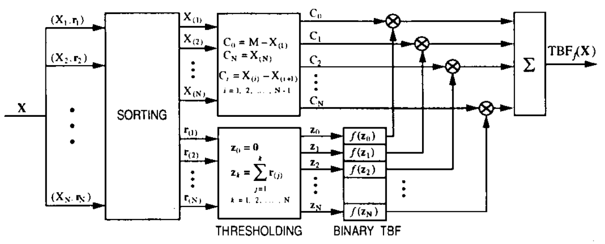


Fig. 2. Implementation of TBF's.

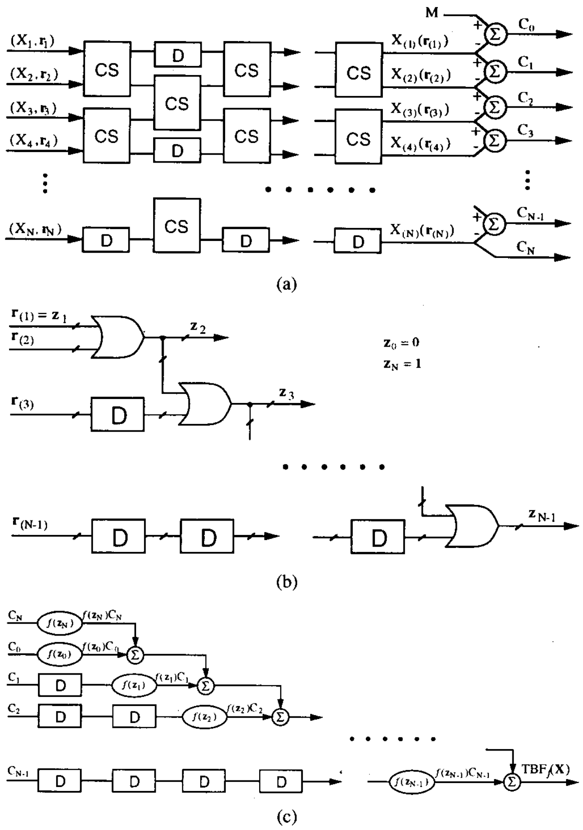


Fig. 3. (a) Sorting block and calculation of C_i's. *CS* and *D* denote the compare-and-swap operation and delay line, respectively; (b) generation of $\mathbf{z}_i$'s from the location vectors $\mathbf{r}_{(j)}$'s. The OR gate denotes N-bit parallel OR operation, and $\mathbf{z}_0$ and $\mathbf{z}_N$ are set to 0 and 1, respectively; (c) calculation of multilevel TBF output.

useful for realizing Steps 2 and 3.

$$\sum_{i=0}^{N} f(I_{X_{(i)}}(\mathbf{X}))C_i = \sum_{i=0}^{N} f(\mathbf{z}_i)C_i. \tag{29}$$

Evaluation of $\mathbf{z}_i, i = 1, \ldots, N$ requires N^2 binary additions (or logical OR operations), which are much simpler than N^2 multilevel comparisons. The price for this simplicity is the storage space for the N N-bit location vectors. In (29), the output is calculated by summing C_i's associated with $f(\mathbf{z}_i) = 1$. The Boolean function is implemented either by a combinatorial logic or by a look-up table. The complexity of a combinatorial logic depends heavily on the Boolean function, whereas a look-up table always requires the storage space of 2^N bits.

The algorithm flow based on (29) is shown in Fig. 2. Each input sample X_j is paired with an N-bit binary location vector $\mathbf{r}_j$ and then sorted. The sorted pairs $X_{(i)}$'s and $\mathbf{r}_{(i)}$'s are used to calculate C_i's and $\mathbf{z}_i$'s, respectively. The output is obtained by evaluating (29). The structures for hardware realization of each of the basic modules in Fig. 2 are depicted in Fig. 3. The sorting block, which is based on BUBBLESORT, and the block calculating C_j's are shown in Fig. 3(a), where the CS element denotes *compare-and-swap* operation, which swaps input sample values and their location vectors if the input value at the bottom is greater than that at the top. In Fig. 3(b), the block for calculating $\mathbf{z}_i$ is depicted, where each OR gate denotes parallel OR operations for N bits. $\mathbf{z}_0$ and $\mathbf{z}_N$ can be preset to 0 and 1, respectively, and $\mathbf{z}_1 = \mathbf{r}_{(1)}$. Fig. 3(c) shows the block evaluating $\sum_{i=0}^{N} f(\mathbf{z}_i)C_i$. Here, $f(\mathbf{z}_i)$'s are obtained sequentially for pipelining. The realization of $f(\mathbf{z}_i)$ can be simplified if we use a look-up table that lists all possible input vectors and the corresponding outputs. This is because each $\mathbf{z}_i$ has i 1's, and the number of possible inputs for $f(\mathbf{z}_i)$ is $\binom{N}{i}$. The total size of the look-up tables realizing N Boolean functions $f(\mathbf{z}_i), i = 1, \ldots, N$ is the same as that realizing one Boolean function $f(\mathbf{x})$ having 2^N possible binary input vectors. In essence, by implementing one look-up table with 2^N binary inputs and the corresponding outputs, the effect of implementing N Boolean functions can be achieved. It is noted that the structure in Fig. 2 and 3 is highly suitable for pipelining, and it can produce a TBF output at every clock; the rate is limited by the maximum delay among CS processor, adder, and the Boolean function.

When implementing stack filters, the algorithm in Fig. 2 can be simplified. Since the output of a stack filter is obtained by finding the minimum among i's for which $f(\mathbf{z}_i) = 1$ (see (18)), the $2N + 2$ additions for calculating C_i's and (29) are unnecessary, and stack filters are somewhat simpler to implement than TBF's. As a useful alternative to the algorithm based on (18), one may consider the bit-serial algorithm [32] for realizing stack filters. This algorithm, however, cannot be used for TBF's because it exploits the positivity of the Boolean function.

In practice, it is difficult to implement TBF's and stack filters with large values of N because the complexity of $f(\cdot)$ increases exponentially as a function of the window size N. As noted before, the evaluation of $f(\cdot)$ is not required in LS TBF's and WOS filters, and for a large N, their implementation becomes considerably simpler than that of TBF's and stack filters.

The structure for realizing the LS TBF in (19) is shown in Fig. 4. The input data paired with the weights (X_j, w_j) are sorted to produce $(X_{(i)}, w_{(i)})$, $i = 1, \ldots, N$. Then, $R_i = \sum_{j=1}^{i} w_{(j)}$ is evaluated and compared with the threshold T. The output of the LS TBF is obtained by summing the C_i's corresponding to R_i's that are greater than or equal to T. In WOS filtering, the output is obtained by finding the minimum among i's for which $R_i \geq T$ (see (20)). Thus, the $2N + 2$ additions for calculating C_i's and (19) are unnecessary.

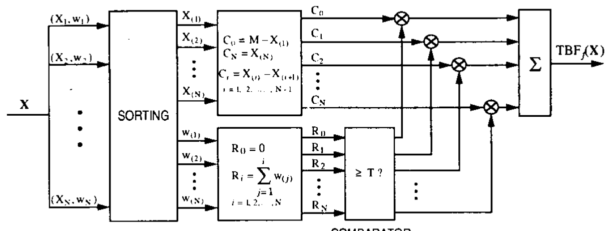


Fig. 4. Implementation of LS TBF's.

VI. Design and Application of the TBF

One of the advantages gained by extending stack (WOS) filters to the class of TBF's (LS TBF's) is the simplicity in designing the TBF (LS TBF). In this section, we discuss how the design procedures for stack filters in [25] and WOS filters in [27] are simplified to design TBF's and LS TBF's and examine the performance of these filters through computer simulation.

A. Designing a TBF under the MAE Criterion

Coyle and Lin [25] introduced a novel technique for obtaining an optimal stack filter under the mean absolute error (MAE) criterion. They showed that an optimal stack filter or, equivalently, an optimal positive Boolean function can be designed by using linear programming. In [8], it is pointed out that a Boolean function yielding a smaller MAE than an optimal positive Boolean function minimizing the MAE can be found easily without using linear programming. The procedure for finding such a Boolean function, which is, in fact, a design procedure for TBF's, is summarized below.

Suppose that the input process $X(n)$ is a noise corrupted version of some desired signal $S(n)$. To estimate the signal, a filtering operation is carried out over a window process $\mathbf{X}(n)$, which is formed by $\mathbf{X}(n) = (X_1(n), \ldots, X_N(n)) \equiv (X(n - N_1), \ldots, X(n), \ldots, X(n + N_2))$, where $N = N_1 + N_2 + 1$ is the window size. We want to find the TBF that best estimates the signal. The MAE for a TBF estimate is given by, dropping the time indices for notational simplicity

$$\begin{aligned}\mathrm{MAE}_f &= E\{|S - \mathrm{TBF}_f(\mathbf{X})|\} \\ &= E\left\{\left|\sum_{\ell=1}^{M}[I_\ell(S) - f(I_\ell(\mathbf{X}))]\right|\right\} \\ &\leq \sum_{\ell=1}^{M} E\{|I_\ell(S) - f(I_\ell(\mathbf{X}))|\} \equiv \mathrm{SMMAE}_f \quad (30)\end{aligned}$$

where the last term is called the *sum of micro* MAE (SMMAE) [8]. If the Boolean function $f(\cdot)$ is positive, the SMMAE is equal to the MAE [25]. Let $f^*(\cdot)$ be the positive Boolean function for which the corresponding stack filter is optimal under the MAE criterion. Then, there always exists a Boolean function, say $f^o(\cdot)$ such that $\mathrm{MAE}_{f^o} \leq \mathrm{SMMAE}_{f^o} \leq \mathrm{SMMAE}_{f^*} = \mathrm{MAE}_{f^*}$, and the $f^o(\cdot)$ can be found following a simplified version of the design process for $f^*(\cdot)$.

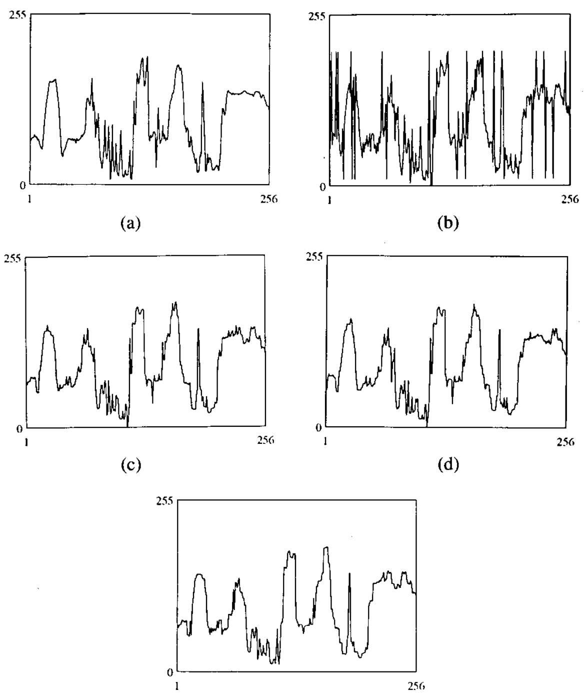


Fig. 5. (a) Original signal; (b) noisy signal; (c) output of the stack filter; (d) output of the TBF; (e) output of the LS TBF.

The optimal stack filter $f^*(\cdot)$ is designed as follows [25]:

$$\underset{f}{\text{minimize}}\, J(f) = \sum_{j=1}^{2^N} c_j f(\mathbf{x}_j) \quad (31.a)$$

$$\text{subject to } f(\mathbf{x}_i) \leq f(\mathbf{x}_j) \text{ if } \mathbf{x}_i \leq \mathbf{x}_j \quad (31.b)$$

$$f(\mathbf{x}_j) = 0 \text{ or } 1 \text{ for all } j \quad (31.c)$$

where c_j is constants depending on input statistics, $f(\cdot)$ is a Boolean function, and $\mathbf{x}_j$ is the jth input vector among 2^N possible binary input vectors and $\mathbf{x}_i \leq \mathbf{x}_j$ if every element of $\mathbf{x}_i$ is smaller than or equal to the corresponding element of $\mathbf{x}_j$. In designing $f^o(\cdot)$, the constraint in (31.b), which guarantees the positivity of Boolean functions, is unnecessary. The TBF $f^o(\cdot)$ is obtained by minimizing $J(f)$ in (31.a) under the constraint in (31.c). Now, the minimization for $f^o(\cdot)$ is trivial; $f^o(\cdot)$ is given by

$$f^o(\mathbf{x}_j) = \begin{cases} 0, & \text{if } c_j > 0 \\ 1, & \text{if } c_j \leq 0 \end{cases} \quad (32)$$

for every $j, 1 \leq j \leq 2^N$.

B. Designing an LS TBF

The design method for stack filters in [25] cannot be applied to designing WOS filters. So far, for WOS filtering, only suboptimal design procedures have been proposed [27], [31]. An LS TBF can be designed following the procedures for WOS filter design. In this subsection, we shall obtain a suboptimal LS TBF using the method in [27].

Consider an LS Boolean function $f(\cdot)$ defined by (4). Let $\tilde{\mathbf{x}}^\ell$ and $\mathbf{w}$ be the augmented input and weight vectors, respectively, defined by $\tilde{\mathbf{x}} = (x_1^\ell, \ldots, x_N^\ell, -1)^t$ and $\mathbf{w} = (w_1, \ldots, w_N, T)^t$, where $x_i^\ell = I_\ell(X_i)$, and w_j's and T are the real-valued weights and the threshold value, respectively, associated with $f(\cdot)$. Then, the LS TBF can be expressed as

$$\mathrm{TBF}_f(\mathbf{X}) = \sum_{\ell=1}^{M} U(\mathbf{w}^t\tilde{\mathbf{x}}^\ell) \tag{33}$$

where $U(\cdot)$ is the unit step function defined as $U(X) = 1$ if $X \geq 0$ and 0 otherwise. Using (33), SMMAE_f is written as

$$\begin{aligned}\mathrm{SMMAE}_f &= \sum_{\ell=1}^{M} E\{|s^\ell - U(\mathbf{w}^t\tilde{\mathbf{x}}^\ell)|\} \\ &\approx \sum_{\ell=1}^{M} E\{|s^\ell - \mathbf{w}^t\tilde{\mathbf{x}}^\ell|^2\}\end{aligned} \tag{34}$$

where $s^\ell = I_\ell(S)$ (see (30)), and the last expression is obtained by approximating the unit step function to a linear function and utilizing the fact that the absolute and square errors are equivalent in the binary domain. Now, the LS TBF minimizing the approximation of the SMMAE in (34) can be obtained as follows:

$$\underset{\mathbf{w}}{\mathrm{minimize}}\, J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^t\Psi\mathbf{w} - \mathbf{w}^t\Phi \tag{35}$$

where $\Psi = \sum_{\ell=1}^{M} E\{\tilde{\mathbf{x}}^\ell(\tilde{\mathbf{x}}^\ell)^t\}$ and $\Phi = \sum_{\ell=1}^{M} E\{s^\ell\tilde{\mathbf{x}}^\ell\}$. Obviously, the solution to this problem is given by

$$\mathbf{w}^o = \Psi^{-1}\Phi. \tag{36}$$

In the case of WOS filtering, the problem in (35) should be solved under the constraints that every element of $\mathbf{w}$ is nonnegative and $\sum_{i=1}^{N} w_i \geq T$. Thus, there is no closed-form solution to the problem, and designing WOS filters requires more work.

C. Experimental Results

In order to assess the performance of the TBF and the LS TBF, these filters are designed using the methods described in the previous subsections and applied to suppress additive Gaussian and impulsive noise superimposed on 1-D and 2-D signals. The Gaussian noise has zero-mean and variance 100, and impulses occur with probability 0.1. Signal values corrupted by impulses are set to either 200 or 20. The statistics required for designing the filters are estimated from the signals under consideration.

Fig. 5(a) illustrates a 1-D signal taken from horizontal lines of the *Lena* image having 256 × 256 pixels. The noisy signal is shown in Fig. 5(b). The TBF, LS TBF, stack, and WOS filters with window size 9 were designed using the statistics estimated from the signals in Fig. 5(a) and (b). The resulting LS TBF and WOS filter turned out to be identical; the parameters of these filters are given by $(w_1, \ldots, w_9) =$ (0.09945, 0.07319, 0.09849, 0.19047, 0.37809, 0.18271, 0.08561, 0.06762, 0.11132) and $T = 0.65053$. To see the potential filtering capabilities, the designed filters were applied to the noisy signal in Fig. 5(b). The outputs of the filters are illustrated in Fig. 5(c)–(e), and the corresponding MAE's that have been calculated from the filtered and the original signals are listed in Table IV. It is seen that the TBF preserves more details and has smaller MAE compared with the others.

TABLE IV
MAE VALUES ESTIMATED FROM THE 1-D SIGNALS IN FIG. 5

	MAE
	$G(0,100), P_e = 0.1$
Stack Filter	7.67
TBF	6.52
LS TBF (WOS Filter)	8.67

TABLE V
MAE VALUES ESTIMATED FROM THE IMAGES IN FIG. 6

	MAE
	$G(0,100), P_e = 0.1$
Stack Filter	7.9422
TBF	7.9170
LS TBF (WOS Filter)	7.9074

Fig. 6(a) and (b) illustrate the original 512 × 512 "bridge over stream" image and the noisy image, respectively. To enhance the noisy image, the TBF, LS TBF, stack, and WOS filters with a 3 × 3 square window were designed. The statistics required to design the filters are estimated from the upper left quarter of images 6(a) and (b). Again, the LS TBF and the WOS filter turned out to be identical. The images obtained by applying the designed filters to the noisy image are shown in Fig. 6(c)–(e), and the corresponding MAE values are listed in Table V. Visually, the image filtered by the LS TBF looks better than the other images. Furthermore, rather surprisingly, the MAE associated with the LS TBF is smaller than the others. The superiority of the LS TBF in filtering performance is based on the following fact. The number of parameters to be estimated for the problem in (35), $(N+1)^2 + (N+1)$, is considerably smaller than 2^N, which is the number of c_j's in (31). Therefore, roughly speaking, the parameters estimated for (35) are more accurate than those for (31). The LS TBF's and WOS filters may be preferred to TBF's and stack filters when the input statistics are unknown and have to be estimated.

VII. CONCLUSION

A new class of nonlinear filters called TBF's has been introduced. A TBF is defined by a Boolean function on the binary domain. It has been shown that the logical negation on the binary domain produces the minus (−) operation on the multilevel. In particular, the TBF can be expressed either as a sum of "*local minimum − local maximum*" terms on the multilevel or as an adaptive linear combination of ordered input data.

While all stack filters are translation and scale invariant, TBF's may be neither translation nor scale invariant. The class of TBF's includes some known operators such as the range estimator, quasi-ranges, and the difference of median estimates that are employed for edge detection in image processing. An interesting property indicating that any TBF can be expressed as a linear combination of stack filters has been derived.

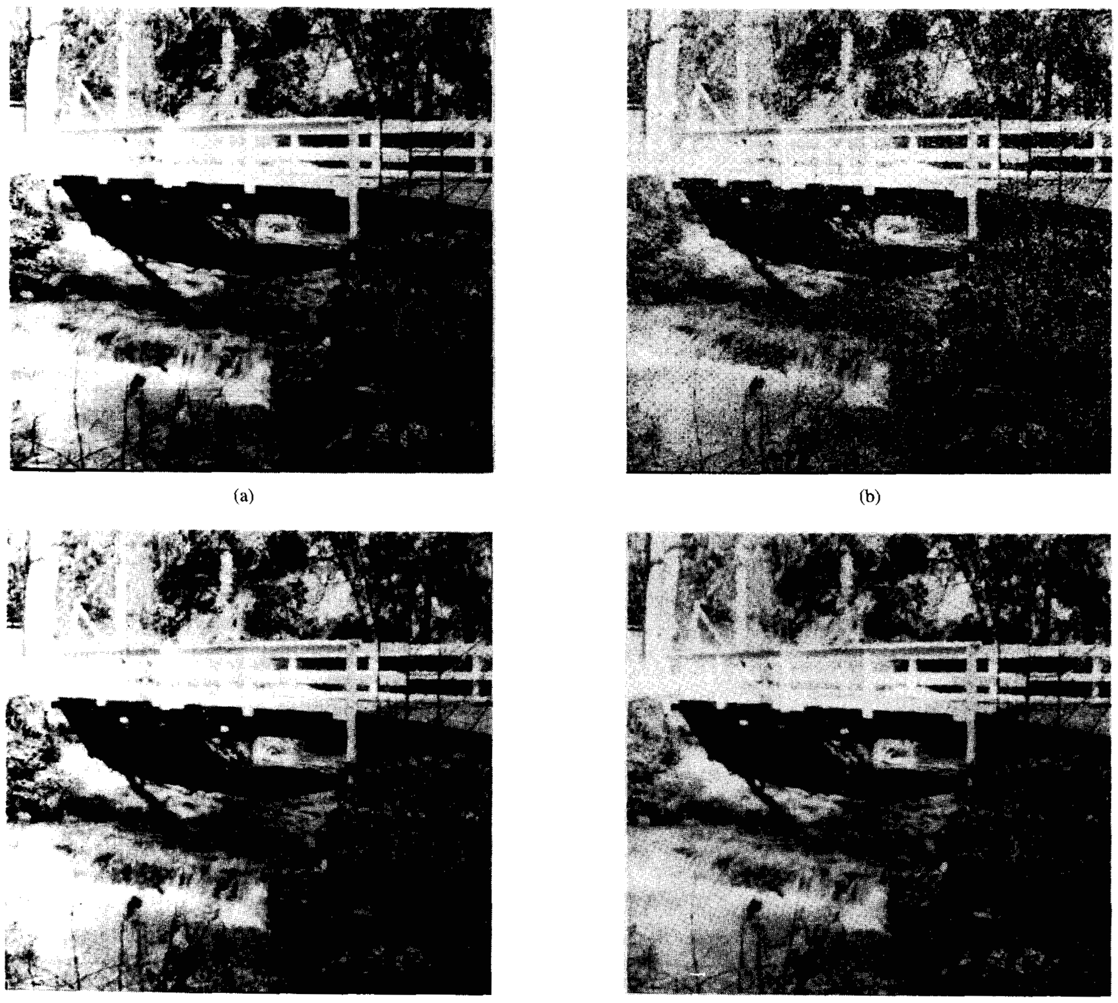


Fig. 6. (a) Original image; (b) noisy image; (c) output of the stack filter; (d) output of the TBF.

As a useful special case of TBF's, the LS TBF has been introduced. The LS TBF, which is defined by the threshold logic, is a direct extension of WOS filters. LS TBF's are much simpler to implement than TBF's and may be preferred to TBF's in practical applications.

Implementation and design of the TBF and LS TBF has been investigated. It was observed that the procedure for designing TBF's (LS TBF's) is considerably simpler than designing stack (WOS) filters and that the former can outperform the latter at the expense of a slight increase in computational complexity.

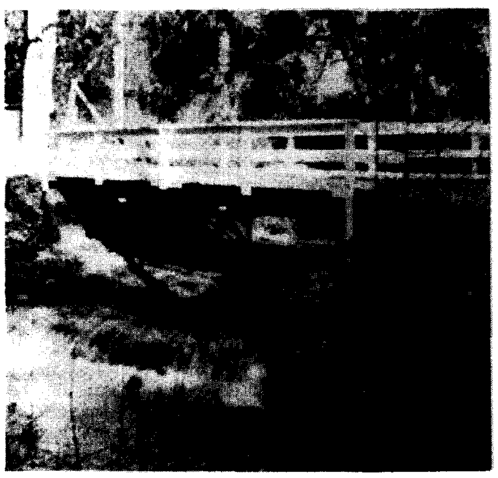
Experimental results indicate that LS TBF's can be superior to TBF's in filtering peformance when the filters are designed based on estimated input statistics. All LS TBF's designed in the experiments for reducing noise turned out to be WOS filters.

There are some interesting topics for further research. They are described as follows: 1) *Statistical analysis of the properties of TBF's.* 2) *Optimization of TBF's and LS TBF's.* Recently, a design method for TBF's under the mean square error criterion was presented in [30]. An interesting question to be addressed is the following: Under what conditions does an optimal TBF reduce to an LS TBF, a stack, or a WOS filter? 3) *Extending TBF's to filters that are defined by an arbitrary function $f(\cdot)$ (not necessarily a Boolean function) on the binary domain.* The class of filters introduced by this extension is the class of all filters possessing the threshold decomposition property. 4) *Further extension of TBFs to the filters defined by (2), which include generalized stack, microstatistic, and generalized WOS [28] filters as special cases.*

Appendix

SOMEP: an algorithm to get *SOMEP* expressions from a given Boolean function

Suppose $f(\mathbf{x})$ is a given Boolean function. Consider the set of its true vectors $V(f) = \{\mathbf{v}_i = (v_i^1, \ldots, v_i^N) \mid f(\mathbf{v}_i) =$



(e)

Fig. 6 *(continued).* (e) Output of the LS TBF.

$1, i = 1, \ldots, 2^N\}$. Then, a SOMEP expression of $f(\mathbf{x})$ is obtained through the following steps.

Step 1. Find a pair of true vectors $\mathbf{v}_i, \mathbf{v}_j \in V(f)$ such that $v_i^k \neq v_j^k$ for some k and $v_i^l = v_j^l$ for all $l \neq k$. Then, update $V(f)$ by replacing $\mathbf{v}_i$ and $\mathbf{v}_j$ with the reduced one, which is given by $(v_i^1, \ldots, v_i^{k-1}, \times, v_i^{k+1}, \ldots, v_i^N)$, where $\times$ denotes a *don't-care* condition.

Step 2. Repeat Step 1 until no pair of true vectors in $V(f)$ can be reduced.

Step 3. To each true vector $\mathbf{v}_m = (v_m^1, \ldots, v_m^N)$ in the resultant set $V(f)$, assign the product $\pi_m(\mathbf{x}) = x_1' x_2' \cdots x_N'$, where $x_i' = x_i$ if $v_m^i = 1$, $x_i' = \bar{x}_i$ if $v_m^i = 0$, and $x_i' = 1$ if v_m^i is a don't-care.

Step 4. Then, the product $\pi_m(\mathbf{x})$'s are mutually exclusive, and their sum is a SOMEP expression of the given Boolean function $f(\mathbf{x})$.

Proof: In the above algorithm, true vectors of $f(\mathbf{x})$ are partitioned into groups, each of which is represented by a vector with don't-cares. Since there is no intersection between the groups, the products representing the groups are mutually exclusive. □

Although the SOMEP algorithm can be easily implemented in a computer program, the minimal SOMEP expression (in the sense that it cannot be further reduced) resulting from the above algorithm may depend on the choice of combining pairs in Step 1. An example illustrating the SOMEP algorithm is presented below.

Example A.1: Suppose $N = 4$ and $f(\mathbf{x}) = x_1x_3x_4 + \bar{x}_2x_4$. Then

$$\begin{aligned} V(f) &= \{(0,0,0,1),(0,0,1,1),(1,0,0,1),\\ &\qquad (1,0,1,1),(1,1,1,1)\}\\ &= \{(0,0,\times,1),(1,0,0,1),(1,0,1,1),(1,1,1,1)\}\\ &= \{(0,0,\times,1),(1,0,\times,1),(1,1,1,1)\}\\ &= \{(\times,0,\times,1),(1,1,1,1)\}. \end{aligned} \tag{A.1}$$

Thus, we have a SOMEP expression: $f(\mathbf{x}) = \bar{x}_2x_4 + x_1x_2x_3x_4$. Choosing different pairs to combine, we obtain another SOMEP form:

$$\begin{aligned} V(f) &= \{(0,0,0,1),(0,0,1,1),(1,0,0,1),\\ &\qquad (1,0,1,1),(1,1,1,1)\}\\ &= \{(0,0,\times,1),(1,0,0,1),(1,0,1,1),(1,1,1,1)\}\\ &= \{(0,0,\times,1),(1,0,0,1),(1,\times,1,1)\}. \end{aligned} \tag{A.2}$$

The resulting SOMEP form is given by $f(\mathbf{x}) = \bar{x}_1\bar{x}_2x_4 + x_1\bar{x}_2\bar{x}_3x_4 + x_1x_3x_4$. □

References

[1] J. P. Fitch, E. J. Coyle, and N. C. Gallagher Jr., "Median filtering by threshold decomposition," *IEEE Trans. Acoust. Speech Signal Processing,* vol. ASSP-32, pp. 1183–1188, Dec. 1984.

[2] P. D. Wendt, E. J. Coyle, and N. C. Gallagher Jr., "Stack filters," *IEEE Trans. Acoust. Speech Signal Processing,* vol. ASSP-34, pp. 898–911, Aug. 1986.

[3] T. S. Huang, Ed., *Two-Dimensional Digital Signal Precessing II: Transform and Median Filters, Topics in Applied Physics.* New York: Springer-Verlag, 1981, vol. 43.

[4] T. A. Nodes and N. C. Gallagher, Jr., "Median filters: Some modifications and their properties," *IEEE Trans. Acoust. Speech Signal Processing,* vol. ASSP-30, pp. 739–746, Oct. 1982.

[5] O. Yli-Harja, J. Astola, and Y. Neuvo, "Analysis of the properties of median and weighted median filters using threshold logic and stack filter representation," *IEEE Trans. Signal Processing,* vol. 39, pp. 395–410, Feb. 1991.

[6] A. C. Bovik, T. S. Huang, and D. C. Munson, Jr., "A generalization of median filtering using linear combinations of order statistics," *IEEE Trans. Acoust. Speech Signal Processing,* vol. ASSP-31, pp. 1342–1350, Dec. 1983.

[7] Y. H. Lee and S. A. Kassam, "Generalized median filtering and related nonlinear filtering techniques," *IEEE Trans. Acoust. Speech Signal Processing,* vol. ASSP-33, pp. 672–683, June 1985.

[8] J. Song and Y. H. Lee, "Linear combination of weighted order statistic filters: an extension of stack filters," in *Proc. 26th Ann. Conf. Inform. Sci. Syst.* (Princeton, NJ), Mar. 1992.

[9] 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.

[10] H. J. A. M. Haijmans, "Theoretical aspect of gray-level morphology," *IEEE Trans. Patt. Anal. Machine Intell.,* vol. 13, pp. 568–582, June 1991.

[11] J.-H. Lin and E. J. Coyle, "Minimum mean absolute error estimation over the class of generalized stack filters," *IEEE Trans. Acoust. Speech Signal Processing,* vol. 38, pp. 663–678, Apr. 1990.

[12] G. R. Arce, "Micro-statistic in signal decomposition and the optimal filtering problem," *IEEE Trans. Signal Processing,* vol. 40, no. 8, Aug. 1992.

[13] A. T. Fam and Y. H. Lee, "Selection filters and commutativity with memoryless nonlinearities," in *Proc. IEEE Int. Symp. Circuits Syst. ISCAS-90,* May 1990, pp. 1743–1746.

[14] H. A. David, *Order Statistics.* New York: Wiley, 1981.

[15] I. Pitas and A. N. Venetsanopoulos, "Edge detectors based on order statistics," *IEEE Trans. Patt. Anal. Machine Intell.,* vol. PAMI-8, pp. 538–550, July 1986.

[16] A. C. Bovik and D. C. Munson, "Edge detection using median comparisons," *Comput. Graphics Image Process.,* vol. 33, pp. 377–389, 1986.

[17] P. M. Lewis and C. L. Coates, *Threshold Logic.* New York: Wiley, 1967.

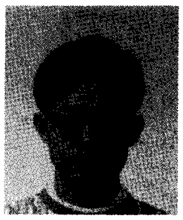
[18] S. Muroga, *Threshold Logic and Its Applications.* New York: Wiley, 1971.

[19] D. E. Knuth, *The Art of Computer Programming.* Reading, MA: Addison-Wesley, 1973, vol. 3.

[20] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms.* New York: Computer Science Press, 1978.

[21] I. Pitas, "Fast algorithms for running ordering and max/min calculation," *IEEE Trans. Circuits Syst.,* vol. 36, pp. 795–804, June 1989.

[22] 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.
[23] N. Demassieux, F. Jutand, M. Saint-Paul, and M. Dana, "VLSI architecture for a one chip video median filter," in *Proc. IEEE ICASSP* (Tampa, FL), Mar. 1985, pp. 1001–1004.
[24] L. A. Christopher, W. T. Mayweather III, and S. S. Perlman, "A VLSI median filter for impulse noise elimination in composite or component TV signals," *IEEE Trans. Consumer Electron.,* vol. 34, pp. 262–267, Feb. 1988.
[25] E. J. Coyle and J.-H. Lin, "Stack filters and the mean absolute error criterion," *IEEE Trans. Acoust. Speech Signal Processing,* vol. 36, pp. 1244–1254, Aug. 1988.
[26] B. Zeng, M. Gabbouj, and Y. Neuvo, "A unified design method for rank order, stack and generalized stack filters based on classical Bayes decision," *IEEE Trans. Circuits Syst.,* vol. 38, pp. 1003–1020, Sep. 1991.
[27] L. Yin, J. T. Astola, and Y. A. Neuvo, "Adaptive stack filtering with application to image processing," *IEEE Trans. Signal Processing,* vol. 41, pp. 162–184, Jan. 1993.
[28] ——, "A new class of filters-neural filters," *IEEE Trans. Signal Processing,* vol. 41, pp. 1201–1222, Mar. 1993.
[29] J.-H. Lin, T. M. Sellke, and E. J. Coyle, "Adaptive Stack filtering under the mean absolute error criterion," *IEEE Trans. Acoust. Speech Signal Processing,* vol. 38, pp. 938–954, June 1990.
[30] K. D. Lee and Y. H. Lee, "Optimization of threshold Boolean filters," in *1993 IEEE Winter Workshop Nonlinear Digital Signal Processing* (Tampere, Finland), Jan. 1993.
[31] B. Jeong and Y. H. Lee, "Design of weighted order statistic filters using the perceptron algorithm," in*1993 IEEE Winter Workshop Nonlinear Digital Signal Processing,* (Tampere, Finland), Jan. 1993.
[32] K. Chen, "Bit-serial realization of a class of nonlinear filters based on positive Boolean functions," *IEEE Trans. Circuits Syst.,* vol. 36, pp. 785–794, June 1989.



Ki Dong Lee was born in Gangwon-do, Korea, on April 6, 1964. He received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1987 and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology, Taejon, Korea, in 1989 and 1994, respectively.

From August 1989 to July 1990, he was a visiting researcher at the Department of Radiology, Columbia University, New York, NY. Since 1991, he has been a research engineer at the Image and Media Laboratory, GoldStar Co., Ltd., Seoul, Korea. His major research interests are in nonlinear and linear digital signal processing, image coding, and digital communication systems.

Yong Hoon Lee was born in Seoul, Korea, on July 12, 1955. He received the B.S. and M.S. degrees in electrical engineering from Seoul National University, Seoul, Korea, in 1978 and 1980, respectively, and the Ph.D. degree in systems engineering from the University of Pennsylvania, Philadelphia, in 1984.

From 1984 to 1988, he was an Assistant Professor at the Department of Electrical and Computer Engineering, State University of New York at Buffalo. Since 1989, he has been with the Department of Electrical Engineering at the Korea Advanced Institute of Science and Technology, where he is currently an Associate Professor. His research activities are in the areas of one- and two-dimensional digital signal processing, VLSI signal processing, and digital communication systems.