SOFTWARE AND VLSI ALGORITHMS FOR RECURSIVE MEDIAN FILTTERS

Sung-Jea Ko*, Yong Hoon Lee**, and Adly T. Fam**

* Dept. of Electrical and Computer Engineering
University of Michigan-Dearborn
Dearborn, MI 48128

** Dept. of Electrical and Computer Engineering
State University of New York at Buffalo
Buffalo, NY 14260

ABSTRACT

It is pointed out that in one-dimensional (1-D) recursive median (RM) filtering all of the previous outputs except the most recent output are redundant in determining the present output. Based on the observation, efficient software and VLSI algorithms for 1-D RM filters are presented. It is shown that the implementation of RM filtering based upon the proposed algorithms is simpler than that of standard median (SM) filtering.

I. INTRODUCTION

The SM filter is a simple nonlinear smoother that can suppress noise while retaining sharp sustained changes (edges) in signal values. It is particularly effective in reducing impulsive-type noise. The output of the SM filter at a point is the median value of the input data inside the window centered at the point. If we let $\{X(k) \mid 1 \le k \le L\}$ and $\{Y_s(k) \mid 1 \le k \le L\}$ be the input and the output of the 1-D SM filter of window size 2N+1, then

$$Y_s(k) = \text{med}\{X(k-N), \ldots, X(k), \ldots, X(k+N)\} \quad (1)$$

Here to account for start up and end effects X(1) and X(L), respectively, are repeated N times at the beginning and at the end of the input. The RM filter is a modification of the SM filter defined in (1); specifically, the output $Y_r(k)$ of the RM filter of size 2N+1 is given by

$$Y_r(k) = \text{med}\{Y_r(k-N), \ldots, Y_r(k-1), X(k), \ldots, X(k+N)\} \quad (2)$$

RM filtering can extract some signal features such as roots better than can SM filtering, and is a useful alternative to SM filtering in some applications [1]-[3].

While many software and VLSI algorithms for SM filters have been proposed [4]-[7], no fast algorithm has been introduced specifically for RM filters. In general, RM filters are implemented by modifying an SM filtering algorithm and, as a consequence, the implementation of RM filters is computationally and structurally more complex than that of SM filters. In this paper, we first point out that in RM filtering the previous outputs $\{Y_r(k-N), \ldots, Y_r(k-2)\}$ are not necessary to determine the present output $Y_r(k)$, but are redundant. Then, based on this observation, software and VLSI algorithms for 1-D RM filters are proposed. It is shown that RM filtering implemented by the proposed algorithms requires less computation and less hardware than SM filtering.

II. SOME PROPERTIES OF 1-D RM FILTERS AND THEIR APPLICATION TO IMPLEMENTATION

In this section, we derive some properties of RM filters on which the proposed algorithms are based. The property stated below shows that the previous outputs $\{Y_r(k-N), \ldots, Y_r(k-2)\}$ in (2) are redundant.

Property 1: In RM filtering, the output $Y_r(k)$ is represented by

$$Y_r(k) = \text{med}\{\overbrace{Y_r(k-1), \ldots, Y_r(k-1)}^{N \text{ times}}, X(k), \ldots, X(k+N)\} \quad (3)$$

Proof: In [2], it is shown that the output sequence $\{Y_r(k)\}$ is a locally monotonic sequence of length N+1, that is, $\{Y_r(k-N), \ldots, Y_r(k-1), Y_r(k)\}$ is either nondecreasing or nonincreasing for any k. Suppose that in (2) $Y_r(k-N) \le \ldots \le Y_r(k-1)$ and $Y_r(k-N) < Y_r(k-1)$. Then $Y_r(k) \ge Y_r(k-1)$, because otherwise the local monotonicity is violated. This implies that at least N samples among $\{X(k), \ldots, X(k+N)\}$ are greater than or equal to $Y_r(k-1)$. The output $Y_r(k)$ is rewritten as

$$Y_r(k) = \begin{cases} Y_r(k-1), & \text{if N samples in } S_k \ge Y_r(k-1) \\ \min\{X(k), \ldots, X(k+N)\}, & \text{if all samples in } S_k \ge Y_r(k-1). \end{cases}$$

CH2692-2/89/0000-0417 $1.00

$$= \text{med}\{Y_r(k-1), \ldots, Y_r(k-1), X(k), \ldots, X(k+N)\},$$

where $S_k = \{X(k), \ldots, X(k+N)\}$. Similarly, we can show the case where $Y_r(k-N) \geq \ldots \geq Y_r(k-1)$ and $Y_r(k-N) > Y_r(k-1)$. The proof for the case where $Y_r(k-N) = \ldots = Y_r(k-1)$ is trivial.

It is interesting to note that the RM filter can be thought of as a recursive weighted median filter that gives more weight only to the last output $Y_r(k-1)$. Obviously, the implementation of RM filtering will become easier by using this property. The following corollary is particulary useful in implementating 1-D RM filters on a general purpose computer.

Corollary 1: The output $Y_r(k)$ of the RM filter is given by

$$Y_r(k) = \text{med}\{X^k_{min}, X^k_{max}, Y_r(k-1)\} \quad (4)$$

where X^k_{min} and X^k_{max}, respectively, are the minimum and the maximum of $\{X(k), .., X(k+N)\}$.

Next we show that 1-D RM filtering implemented by using this corollary requires less computation than the fast algorithm for 1-D SM filtering described in [8]. Fast algorithms for SM filters usually make use of the results obtained in computing the median of the previous window to process the current window. In 1-D SM filtering, given the ordered data in the previous window, the data in the current window can be sorted through at most 4N+2 compare/swap (C/S) operations [8]. Therefore, after initial ordering of the data in the first window, evaluating each $Y_s(k)$, $k \geq 2$, requires at most 4N+2 C/S operations. Corollary 1 indicates that the output $Y_r(k)$ of a RM filter can be obtained through ordering of N+1 input values $\{X(k), \ldots, X(k+N)\}$ followed by the median operation of size three. If an ordered set of $\{X(k-1), \ldots, X(k+N-1)\}$ is given, then $\{X(k), \ldots, X(k+N)\}$ can be sorted after at most 2N+2 C/S operations following the procedure described in [8]. Since the median operation of size three requires two comparisons, then $Y_r(k)$, $k \geq 2$, can be obtained through at most 2N+4 C/S operations. Thus, due to Corollary 1, the implementation of 1-D RM filtering can be simpler than that of 1-D SM filtering.

The corollary stated below shows an interesting relationship between RM and last output reference (LOR) filters [9], and leads to an efficient hardware structure of 1-D RM filters. Here the output at time k of the LOR filter of size N+1 is one of $\{X(k), \ldots, X(k+N)\}$ which is the closest in value to the most recent output.

Corollary 2: If the input is a binary sequence, then $Y_r(k)$

$$Y_r(k) = \begin{cases} X(k), & \text{if } Y_r(k-1) \neq X(k) = \ldots = X(k+N) \\ Y_r(k-1), & \text{otherwise.} \end{cases} \quad (5)$$

It is significant to note from this corollary that the output $Y_r(k)$ of the RM filter of size 2N+1 can be thought of as one of $\{X(K), \ldots, X(k+N)\}$ which is closest in value to the last output $Y_r(k-1)$ when the input is binary. This observation indicates that if the input is binary, then the RM filter of size 2N+1 is identical to the LOR filter of size N+1. This equivalence was observed in [9], but was proved in a different manner. It should be pointed out that the equivalence between RM and LOR filtering does not hold unless the input is binary; for a multi-level input sequence, the RM filter having the threshold decomposition property [10] and the LOR filter, which does not have such property, usually produce different results.

The rule in Corollary 2, which determines the output of the RM filter, can be represented as a Boolean expression. For example, under the assumption that the input to this filter is restricted to binary values,

$$Y_r(k) = Y_r(k-1)[X(k)+X(k+1)+ \ldots +X(k+N)] + [X(k)X(k+1) \ldots X(k+N)] \quad (6)$$

where we represented the OR operation by addition, and the AND operation by multiplication. Fig. 1 illustrates the logic network realizing the Boolean function in (6). It is important to note that this Boolean function is realized by using two OR gates and two AND gates irrespective of the window size 2N+1. In Fig. 2, we present the logic network that can produce the outputs of any binary RM filter whose window size is less than or equal to 2M+1.

The SM filtering operation can also be represented as a Boolean expression [11]. It is straightforward to see that the realization of a Boolean function of SM filtering with size 2N+1 requires $(2N+1)!/[(N+1)!N!]$ AND/OR gates. Therefore, by using Corollary 2 the implementation of RM filtering for binary signals becomes remarkably simpler than that of SM filtering[1].

The RM filter for multi-level input signals can be implemented by using the binary RM filters. In the following section, we present efficient VLSI structures for multi-level RM filters consisting of binary RM filters.

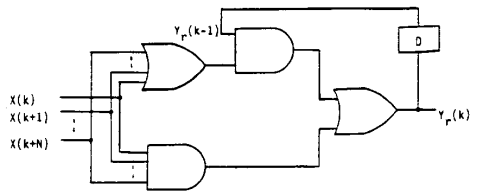

Fig. 1. The logic network for binary RM filtering (Window size 2N+1).

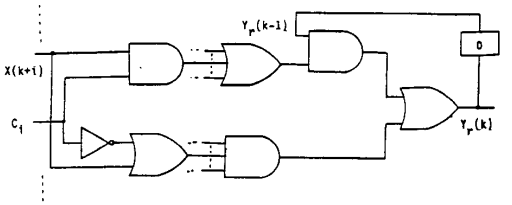

Fig. 2. The logic network producing the outputs of any binary RM filter whose window size is less than or equal to 2M+1 where $0 \neq i \leq M$, and $C_i=1$ if $0 \leq i \leq N$, and $C_i=0$ if $N+1 \leq i \leq M$. Here 2N+1 is the size of the RM filter and $M \geq N$.

III. VLSI REALIZATION

The output of a SM filter with the multi-level input sequence can be obtained from the outputs of binary SM filters. This has been done through either the threshold decomposition [11],[12] or the bit-serial approach [5],[13]. In what follows, we show that both the threshold decomposition and the bit-serial approach can be incorporated with the binary RM filtering structure in Fig. 2.

Since RM filters have the threshold decomposition property, they can be implemented based on the property as shown in Fig. 3. The block diagram of Fig. 3 shows an implementation where the binary RM filtering realization in Fig. 2 is directly applied (the input is an M-level signal).

The RM filtering algorithm based on the bit-serial approach is summarized as follows: Let the input level $M = 2^P$ (P-bit signal). Given $\{Y_r(k-1), X(k), \ldots, X(k+N)\}$, we assume that $(X^j(k+i) \mid 1 \leq j \leq P)$ and $(Y_r^j(k-1) \mid 1 \leq j \leq P)$, respectively, are the radix-2 binary representations of $X(k+i)$ and $Y_r(k-1)$, where $X^1(k+i)$ and $Y_r^1(k-1)$ are the most significant bits. The output $Y_r(k)$ of the RM filter for the P-bit signal is represented by

$$Y_r(k) = \sum_{j=1}^{P} Y_r^j(k)\, 2^{P-j}$$

where

$$Y_r^j(k) = \text{med}\{N \text{ copies of } \hat{Y}_r^j(k-1), \hat{X}^j(k), \ldots, \hat{X}^j(k+N)\} \quad (7)$$

where $\hat{Y}_r^1(k-1)=Y_r^1(k-1)$ and $\hat{X}^1(k+i)=X^1(k+i)$ for all i, $0 \leq i \leq N$, and for each j, $2 \leq j \leq P$,

$$\hat{Y}_r^j(k-1) = \begin{cases} Y_r^j(k-1), & \text{if } Y_r^m(k-1)=Y_r^m(k) \text{ for } m=1,2,\ldots,j-1 \\ Y_r^n(k-1), & \text{if } Y_r^m(k-1)=Y_r^m(k),\ m=1,2,\ldots,n-1 \text{ and, } Y_r^n(k-1) \neq Y_r^n(k),\ n \leq j-1, \end{cases} \quad (8a)$$

$$\hat{X}^j(k+i) = \begin{cases} X^j(k+i), & \text{if } X^m(k+i)=Y_r^m(k) \text{ for } m=1,2,\ldots,j-1 \\ X^n(k+i), & \text{if } X^m(k+i)=Y_r^m(k),\ m=1,2,\ldots,n-1, \text{ and } X^n(k+i) \neq Y_r^n(k),\ n \leq j-1. \end{cases} \quad (8b)$$

In essence, this algorithm replaces $Y_r(k-1)$ with a certain value which is greater (smaller) than $Y_r(k-1)$, once it becomes evident that $Y_r(k-1)$ is greater (smaller) than $Y_r(k)$; similarly, each $X(k+i)$ is replaced. Note that $Y_r^j(k)$ in (7) can be obtained by using the RM filtering structure in Fig. 2. An implementation of the algorithm for the j-th bit, which is a slight modification of the one in [13], is shown in Fig. 4. The logic units in this implementation produce the modified values in (8) depending on the outputs $\{Y_r^1(k), \ldots, Y_r^{j-1}(k)\}$; see [13] for the details.

The implementation based on the threshold decomposition property is parallel and modular, but its hardware complexity grows exponentially with the number of bits in the input. On the other hand, the hardware complexity of the implementation based on the bit-serial approach grows linearly with the number of bits. However, its speed is rather slow because the output at each bit is obained in series.

CONCLUSION

It is evident that Property 1 is useful in analyzing the deterministic and statistical properties of RM filters. The analysis of RM filters using Property 1 is under investigation.

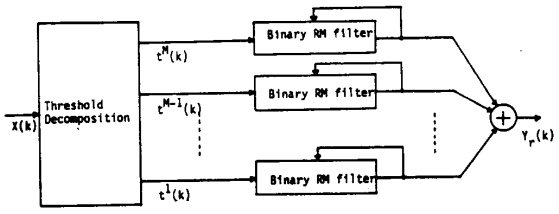

Fig. 3. The implementation of a RM filter based on the threshold decomposition where $t^j(k) = \begin{cases} 1, & \text{if } X(k) \geq j \\ 0, & \text{if } X(k) < j. \end{cases}$

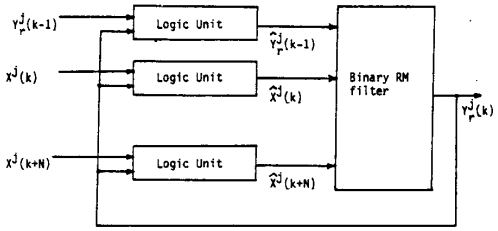

Fig. 4. Bit-serial realization of the RM filter of size 2N+1.

REFERENCES

[1] T. A. Nodes and N. C. Gallagher, "Median filters: Some modifications and their properties," IEEE Trans. Acoust., Speech, and Signal Proc., vol. ASSP-30, pp. 739-746, Dec. 1982.

[2] M. P. McLoughlin and G. R. Arce, "Deterministic properties of the recursive separable median filter," IEEE Trans. Acoust., Speech, and Signal Proc., vol. ASSP-35, pp. 98-106, Jan. 1987.

[3] G. R. Arce and N. C. Gallagher, "Stochastic analysis for the recusive median filter process," IEEE Trans. Infor. Theory, vol. IT-34, pp. 669-679, Jul. 1988.

[4] T. S. Huang, G. T. Yang, and G. Y. Tang, "A fast two-dimensional median filtering algorithm," IEEE Trans. Acoust., Speech, and Signal Proc., vol. ASSP-27, pp. 13-18, Feb. 1979.

[5] E. Ataman, V. K. Aatre, and K. M. Wong, "A fast method for real-time median filtering," IEEE Trans. Acoust., Speech, and Signal Proc., vol. ASSP-28, pp. 415-420, Aug. 1980.

[6] K. Oflazer, "Design and Implementation of a single chip 1-D median filter," IEEE Trans. Acoust., Speech, and Signal Proc., vol. ASSP-31, pp. 1164-1168, Oct. 1983.

[7] V. V. B. Rao and K. S. Rao, "A new algorithm for real-time median filtering," IEEE Trans. Acoust., Speech, and Signal Proc., vol. ASSP-34, pp. 1674-1675, Dec. 1986.

[8] J. B. Bednar and T. L. Watt, "Alpha-trimmed means and their relationship to median filters," IEEE Trans. Acoust., Speech, and Signal Proc., vol. ASSP-32, pp. 145-153, Feb. 1984.

[9] Y. H. Lee, S. -J. Ko, and A. T. Fam, "Efficient impulsive noise suppression via nonlinear recursive filtering," IEEE Trans. Acoust., Speech, and Signal Proc., vol. ASSP-37, Feb. 1989.

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

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

[12] R. G. Harber, S. C. Bass, and G. L. Neudeck, "VLSI implmentation of a fast rank-order algorithm," Proc. ICASSP'85, Tampa, FL, Mar. 1985.

[13] K. Chen, "Realizations of a class of non-linear filters using a bit-serial approach," Proc. ISCAS'88, Finland, pp. 1749-1752, June 1988.

[1] The SM filter for binary signals may be implemented by using a counter followed by a comparator [11], which is also more complicated than the proposed RM filtering algorithm. In [12], in order to avoid the difficulty in implementing SM filters with the digital logic, an analog circuit structure was developed. The analog circuit for SM filtering, however, should be more vulherable to process parameter variations than digital logic circuits.