VLSI ARCHITECTURES FOR WEIGHTED ORDER STATISTIC (WOS) FILTERS

Chaitali Chakrabarti and Lori Lucke

Department of Electrical Engineering
Arizona State University
Tempe, AZ 85287-5706

Minnetronix, Inc.
2610 University Ave.
St. Paul, MN 55114

ABSTRACT

The class of median filters has been extended to include *weighted order statistics* (WOS) filters, to improve the flexibility of the filtering operation. The WOS filter weights each input within a sample window, and thus retains the original temporal order information. In this paper, we present efficient VLSI architectures for non-recursive and recursive WOS filters based on (i) array (ii) stack filter and (iii) sorting network structures. All these architectures maintain a *weighted rank* for each sample in the sample window. As the window shifts for each new output, the weighted ranks are updated. We analyze the implementation complexity for each architecture and verify our results through physical implementations.

1. INTRODUCTION

Weighted Order Statistic (WOS) filters [1] are an extension of rank order based filters that incorporate both the rank and the temporal information of the input data to improve the usefulness of the filtering information. Specifically, a WOS filter chooses an output based on the *weighted* ranks of the inputs within a window of samples. A WOS filter is described by

$$y_i = R^{th} \textit{ smallest of } \{W_{-N1} \otimes x_{i-N1}, \ldots, W_0 \otimes x_i, \ldots, W_{N2} \otimes x_{i+N2}\}$$

where R is the rank of the filter, $K = 2N+1$ is the size of the sample window, $\otimes$ denotes duplication of the sample by the corresponding weight value, and

$$\vec{W} = \{W_{-N1}, \ldots, W_{N2}\}$$

is an integer weight vector. WOS filters are used as robust smoothers or predictors. In this paper, we focus on the development of VLSI architectures for WOS filters.

There exist many architectures for rank order filters [2-6]. These architectures can be classified as array, sorting network, or stack filter based architectures. In array architectures, each element in the window is operated on by maintaining its rank value. In sorting network architectures, the elements in the window are completely sorted for each output. A stack filter [7] translates the filtering operation to the binary domain through the use of threshold decomposition.

The existing rank order filter architectures can be extended to WOS filters architectures [5] in a straightforward manner by repeating each sample according to its weight and then operating on the repeated input set. Such extensions are inefficient since the window size increases to the sum of the weights, $S_W = \sum W_i$, and leads to architectures that require $O(S_W)$ processors. In this paper we present new efficient architectures for WOS filters which operate on the original window of size K. These architectures maintain a weighted rank value for each sample within the sample window and update it as the window shifts. Since each sample has a weight associated with it, as the window shifts, the weights of every sample within the window change. Thus, the procedures for updating weighted ranks are more complicated than updating the ranks in rank order filters. Formulating these procedures yields new array architectures, stack filter architectures and sorting network architectures for both nonrecursive and recursive WOS filters.

2. ARRAY ARCHITECTURES

In the array architectures for WOS filters, we maintain a *weighted* rank for each sample within the window. The weighted rank of a particular sample is updated by an amount which is a function of both the comparison results and the weights of the new and the discarded samples, plus the changes in the weights of all samples that are smaller than itself.

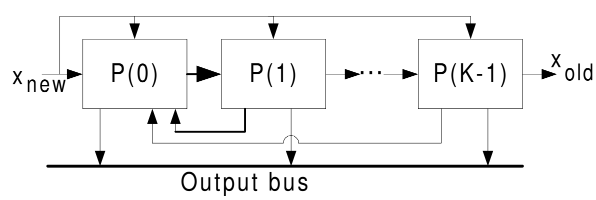

Figure 1: Block diagram of the linear array architectures for non-recursive WOS filter.

We propose an array architecture for non-recursive WOS filters which implements the weighted rank update procedure. The architecture for WOS filters with window size K consists of a semi-systolic linear array of K processors. The processors store the samples in the order of arrival; processor $P(0)$ stores the new sample, and $P(K-1)$ stores the oldest sample. Each processor (except $P(0)$) updates its weighted rank based on the comparisons and weights of the new and discarded samples, plus the change in the number of samples that are smaller than itself. Processor $P(0)$ computes the rank of the new sample by counting the number of samples that are smaller than itself. Figure 1 contains a block diagram of this architecture. Each processor consists of registers to hold the input and the weights, a comparator, a rank update unit and an output unit. Figure 2 describes the design of such a processor.

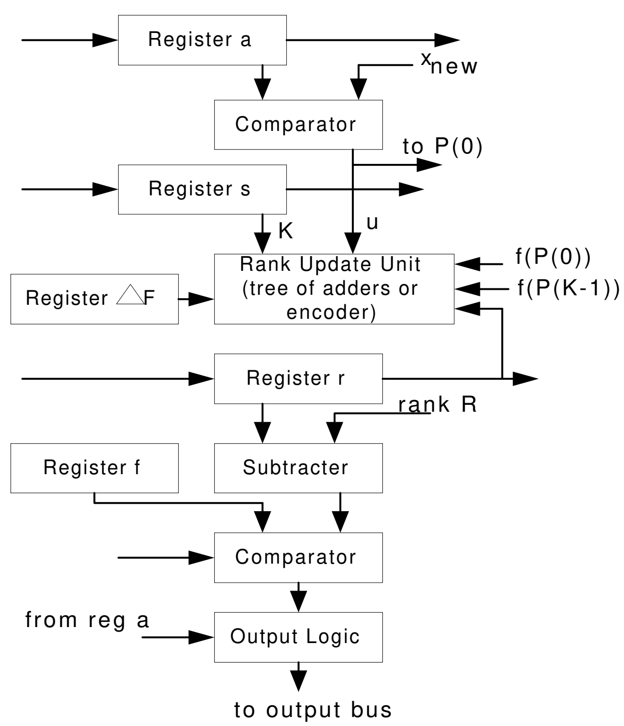

Figure 2: Design of a processor in the array architecture for non-recursive WOS filter.

The architecture for a recursive WOS filter consists of 2 semi-systolic arrays, one to hold the non-recursive (x) samples and the other to hold the recursive (y) samples. The rank update for the recursive filter is similar to the non recursive filter, except that it is now a function of two new samples and two discarded samples. Tables 2 and 4 summarize the hardware and timing requirements of the recursive architecture.

3. STACK FILTER BASED ARCHITECTURES

The bit-parallel stack filter realization of the WOS filter consists of one threshold decomposition unit for the new input sample, 2^b-1 bit-level logic (BLL) units, where b is the sample wordlength, and an output unit as shown in Figure 3. There is a shift register s at the input to the BLL unit to hold the 1-bit decomposed signals of the K samples. In every sample period, s shifts in a 1-bit signal corresponding to the new sample and shifts out a 1-bit signal corresponding to the discarded sample. The BLL unit counts the number of 0's within the sample window to determine whether the current output level should be a 0. The number of 0s, denoted by *NUM0*, is compared with R, the rank of the filter. Then if NUM0 $\geq R$, a `0' is sent to the output logic unit, otherwise a `1' is sent.

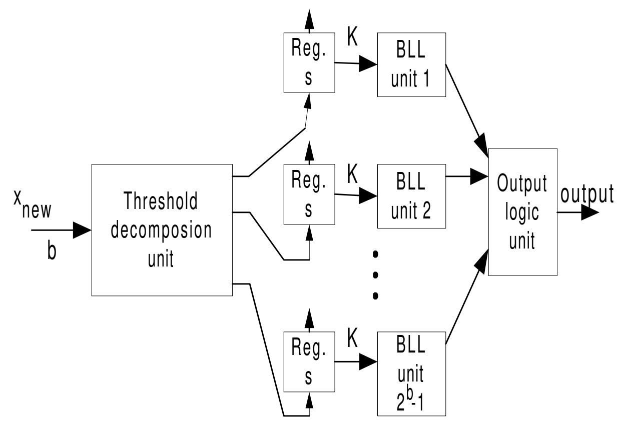

Figure 3: Block diagram of a bit-parallel stack filter based architecture for a non-recursive WOS filter when the window size is K.

The BLL unit updates the number of 0s for each window using one of two methods. The first method counts all the 0s for each new output. The second method maintains a count of the 0s and updates this count depending on comparisons with the oldest and the newest bits in a fashion similar to the rank update unit used in the array architectures.

The bit-parallel implementation based on counting the number of 0's every sample period can be mapped onto a bit-serial implementation. The computations are done MSB (Most Significant Bit) first, with the output of the p^{th} MSBit used to modify the inputs for the computation of the $(p+1)^{th}$ MSBit.

The b-bit stack filter realization of a recursive WOS filter is an extension of the architecture for the non-recursive WOS filter. Each BLL unit now has two shift registers at the input, one to hold the 1-bit signals of the x-samples and one to hold the 1-bit signals of the y-samples. The procedure for updating the number of 0s for each window is the same as in the non-recursive filter implementation.

4. SORTING NETWORK ARCHITECTURES

The proposed sorting network architecture for the non-recursive WOS filter with window size K consists of (i) Batcher's odd-even merge sorting network [8] to sort K samples, (ii) a rank computation unit to compute the ranks of the K samples, and (iii) an output unit which determines the element with rank R. Such a network is described in Figure 4.

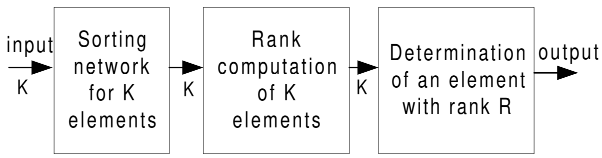

Figure 4: Block diagram of the sorting network for non-recursive WOS filters when the window size is K.

Sorting network: The sorting networks based on Batcher's odd-even merge sort can be used to sort the input samples. The compare-swap unit in the odd-even merge sorting network is modified to pass the weights along with the inputs. However, if Batcher's odd-even merge sorting network is used, then all K samples of a window are sorted for every output computation. Since the samples of consecutive windows overlap by K-1 samples, it is not necessary to completely sort these samples for each output. Instead, the samples could be stored in order as a sorted list. In every sample period, the sorted list is updated by removing the oldest element and inserting the newest element.

Rank computation unit: A tree of adders can compute the weighed ranks of the sorted list. The delay is then $(log_2K)t_{\text{add}}$.

Output unit: The output unit compares the weighted ranks of the sorted list with R, the desired rank, and gates the appropriate sample to the output.

The proposed sorting network architecture for the recursive WOS filter consists of two units: a sorting network to sort M x-samples followed by computation of the ranks of each sample, and N insertion stages to insert the N y-samples one at a time into the sorted list while updating the ranks.

5. IMPLEMENTATION COMPARISONS

The array architecture, the parallel stack filter architecture, and the sorting network architecture utilize physical and time resources as shown in Tables 1, 2, 3 and 4. Tables 1 and 2 compare the number of computational units required by each architecture. Tables 3 and 4 compare the input-output latency (the time from the introduction of a new input to its corresponding output) for each architecture. The nonrecursive versions of these are shown in Tables 1 and 3 while the recursive versions are shown in Tables 2 and 4.

In these tables, the following filter parameters are used: b is the filter sample wordlength, c is the wordlength of the weights, K is the non-recursive window length, and $M+N$ represents the recursive window length. The timing parameters include: $T_{\text{comp}}(n)$ is the time to compare two n-bit numbers, $T_{\text{add}}(n)$ is the time to add two n-bit numbers, $add_{\text{tree}}(m,n)$ is an adder tree that adds m numbers each n bits wide and $T_{\text{addtr}}(m,n)$ is its delay. In the tables, each filter is assumed to be a complete bit-parallel implementation with no pipelining. Pipelined or bit-serial designs can be developed as needed.

A comparison of these architectures is dependent on the choice of filter parameters. The stack filter is clearly larger than the other two architectures for large b since it depends exponentially on b. For large K and large b, the sorting network architecture is the smallest of the three architectures. The sorting network has the smallest latency when K (or $M+N$) is large. Furthermore, the stack filter is faster than the array architecture since it has no comparators.

To verify the complexity analysis of the WOS filters we physically implemented the array and the stack filter architectures for recursive WOS filters using Mentor Graphics Tools and the 1.2um CMOSN standard cell library for c=8, M=4 and N=2. For the array filter b=3 and for the stack filter b=8. The results of the area and time measurements are shown in Table 5 and agree well with the theoretical analysis. Note that with b=8, the stack filter would require 1.3 million transistors. Thus the array architecture is most suitable for large window sizes while the stack filter is more suitable for applications requiring small latencies.

6. CONCLUSION

In this paper we have presented new array architectures, new stack filter architectures, and new sorting network architectures for WOS filters. The array architecture is more suitable for large window sizes, since the size grows linearly with the window size. The sorting network architectures have the smallest latency for large window sizes. The sorting network and the stack filter architectures are the easiest to pipeline and can be used to develop bit-serial architectures. The recursive WOS architectures can be pipelined using approximate lookahead techniques [9].

	Hardware Requirements			
	Comps.	Adders	Registers	Misc.
Array	w=b:K-1 w=c: K	Add$_{tree}$(K,c):K w=c: K	w=b:K w=c: K^2+$3K$-2 w=K:K	Out Logic (mux): K
Stack	None	Add$_{tree}$(K+1,c): 2^b-1 Add$_{tree}$(2^b-1,1): c	w=K:2^b-1 w=c: (2^b-1)(K+2)	Thresh Decom Unit
Sort	w=b:K w=c: K	w=c: $K/2logK$	w=b:K w=$\lceil logK \rceil$: K	Exor, Mux:K

Table 1: Hardware requirements of the three candidate architectures for nonrecursive WOS filters (w=wordlength).

	Hardware Requirements			
	Comps.	Adders	Regsisters	Misc.
Array	w=b: M+N-1 w=c: M+N	Add$_{tree}$(M+N+3,c): M+N w=c: M+N	w=b:M+N w=c:M+N^2 +4(M+N)-3 w=M+N: M+N	Out Logic (mux): M+N
Stac	None	Add$_{tree}$(M+N+3,c): 2^b-1 Add$_{tree}$(2^b-1,1): c	w=M+N: 2^b-1 w=c: (2^b-1) * (M+N +3)	Thres Dec Unit
Sort	w=b: MN+N^2 –0.5N w=c: $2M$+N	w=c: $M(0.5logM+N)$+ $N(N_1)/2$	w=b: M(N+1) +N(N-1)/2 w=$\lceil logM \rceil$: M	Mux: M logic: MN+ N^2-N

Table 2: Hardware requirements of the three candidate architectures for recursive WOS filters (w=wordlength).

Architecture	Input-Output Latency
Array	$T_{comp}(b)+T_{addtr}(K,c)+T_{add}(c)+T_{comp}(c)$
Stack (parallel)	$T_{thresh}(b)+T_{addtr}(K+1,c)+T_{addtr}(2^b-1,1)$
Sorting Network	$T_{comp}(b)+(\log K)T_{add}(c)+T_{comp}(c)$

Table 3: Input/Output Latency of the three candidate architectures for nonrecursive WOS filters.

Architecture	Input-Output Latency
Array	$T_{comp}(b)+T_{addtr}(M+N+3,c)$ $+T_{add}(c)+T_{comp}(c)$
Stack (parallel)	$T_{thresh}(b)+T_{addtr}(M+N+3,c)+T_{addtr}(2^b-1,1)$
Sorting Network	$T_{comp}(b)+(\log M)T_{add}(c)+NT_{comp}(b)$ $)+T_{comp}(c)$

Table 4: Input/Output latency of the three candidate architectures for recursive WOS filters.

Architecture	Transistor Count	Input-Output Latency
Array	43,540	120 nsec
Stack	36,800	52 nsec

Table 5: Actual measured area and latency for the recursive WOS filters using the 1.2 um CMOSN standard cell library.

7. REFERENCES

[1] Lin Y., Astola J., and Neuvo Y. "Optimal weighted order statistic filters under the mean absolute error criterion", *Proc. of the IEEE Int'l Conference on Acoustics, Speech, and Signal Processing,* pages 2529-2532, 1991.

[2] Chakrabarti C. "Efficient stack filter implementations of rank-order filters", *Proc. of the Int'l Symposium on Circuits and Systems*, May 1993.

[3] Chakrabarti C. "Sorting network based architectures for median filters". *IEEE Trans. on Signal Processing*, March 1994.

[4] Chakrabarti C., and Lucke L. "Highly concurrent architectures for recursive median filters". *VLSI Signal Processing VI*, 1993.

[5] Lucke L., and Parhi K. "VLSI architectures for weighted order statistics filters". *Proc. of the IEEE Winter Workshop on Nonlinear Digital Signal Processing*, January 1993.

[6] Egiazarian K., Vainio O., and Astola J. "Implementation of cascaded and recursive stack filters". *Circuits, Systems, and Signal Processing*. vol. 15, no. 1, pages 93-110, January 1996.

[7] Wendt P., Coyle E., and Gallagher N. "Stack Filters". *IEEE Trans. On Acoustics, Speech, and Signal Processing*. vol. ASSP-34, no. 4, August 1986.

[8] Batcher K. "Sorting networks and their applications". *AFIPS Spring Joint Computer Conference*, vol. 32, pages 307-314, 1968.

[9] L.E. Lucke, R. Kroenke, "Training recursive structures for weighted order statistic filtering", *Proc. of the IEEE Int'l Symp. on Circuits and Systems*, London, June 1994.