A Hardware-oriented Design for Weighted Median Filters

Chun-Te Chen

Depart. of Electrical Engineering
National Taiwan University
Taipei, Taiwan 10764
Tel: (886)2-363-5251 ext. 332
Fax: (886)2-363-8247
e-mail: d80049@video.ee.ntu.edu.tw

Liang-Gee Chen

Depart. of Electrical Engineering
National Taiwan University
Taipei, Taiwan 10764
Tel:(886)2-363-3174 ext. 443
Fax:(886)2-363-8247
e-mail: lgchen@cc.ee.ntu.edu.tw

Jue-Hsuan Hsiao

Depart. of Electrical Engineering
National Taiwan University
Taipei, Taiwan 10764
Tel:(886)2-363-5251 ext. 332
Fax:(886)2-363-8247
e-mail:d80049@video.ee.ntu.edu.tw

Abstract— In this paper, the design consideration and algorithm mapping for weighted median filters are presented. To achieve high throughput rate, a special coding technique and its dedicated architecture with block processing are constructed to handle multiple filtering inputs and multiple filtering outputs concurrently. The pipelined cycle in our design is merely as the delay time of 1-bit carry-save-adder (CSA). Due to this design strategy, the proposed architecture can support not only weighted median filters but also rank order-based filters in high-speed applications.

I. Introduction

Many nonlinear filtering techniques[1..6] have been successfully applied to the speech signal processing and digital image processing. Most of them utilize rank order signals of filtering window with the unit weight of the sample, e.g., median filters, rank order filters, and rank order-based filters. An alternative design is to include the temporal-order information by setting the weight not all equal to 1, such as weighted median filters[1,2,3], center weighted median filter[4], and the adaptive stack filter[5,6]. The Weighted Median Filter(WMF) is first introduced by Justusson[1] and further discussed by Brownrigg[2] and Wendt *et al*[3]. This filter gives more weight to some samples within the sliding window to allow a degree control of the smoothing behavior. The sample in the sliding window is duplicated as weight defined before sorting; then the center value of the sorted list is chosen as a median output. Recently, the adaptive stack(weighted) filter[5,6] techniques based on the estimation approach have been proposed to obtain the optimal weight set. These adaptive stack filters allow fractioned weights to get a better compromise between the detailed preservation and impulsive noise removal. The weight set can also be determined based on the structural approach for preservation of lines, corners, and edges.

Weighted median filter is a generalization of the median filter. With proper weights set, they have efficient impulsive noise suppression and excellent image detail-preserving capability. They have been successfully applied in various areas such as noise reduction, image restoration, field interpolation, and image DPCM coding. Recently, many theories about WM filters are under development and are focused on their deterministic and statistical properties. Since WM filters have a higher capability of noise reduction, the implementation of WM filters becomes very interesting. There are two main issues in the implementation of WM filters: an algorithm to find the proper weight, and an efficient algorithm to find its output. These two main algorithms are all required to find the median output of a given weight set efficiently. The first attempt to implement the WM filter was made by Yl. Harji *et. al* based on the positive boolean function. The complexity of the PBF will be $O(2^N)$, where N means the window size of the WM filters. Recently, Astola and Neuvo[7] proposed a novel method with $O(N^2 * S)$ complexity to compute the WM filters' output, where S is the total sum of the weights. Then, they have extended their algorithm to the real weights case. These algorithms are addressed on the permutation of the weight set. The number of the permutations becomes very huge even though the window size is only 9. In this paper, we propose an alternative method to save complexity through hardware consideration.

There are two approaches to implement the weighted median filter. The first method[6] is that the sorting is performed before the weight sum accumulation. The second is that the samples are duplicated as weight defined before sorting. In the former approach, the samples associated with weight inside the filtering window are sorted. Then, starting from the highest end of the sorted list, add up the corresponding weights till the sum of the weights is greater than the desired rank order output. The output of the weighted median filter is the sample corresponding to the last weight. However, the latency of this sequential realization is too long to meet the real time requirement. The latter approach requires a connection network to expand the samples as weight before applying to those of the existing architectures, which have been presented at[8,9] for rank filter realization. For the fractioned weight

set, all weights need to be normalized into integers before sorting. In addition, the total number of data to be sorted is larger than that of rank order filter with unit weight. The input size of the sorter varies depending on the associated weights defined, which will not only increase the complexity of the connection network but is also unsuited to hardware implementation. Therefore, it is still difficult to develop an efficient parallel algorithm and architecture to realize the weight median filter.

In this paper, we propose a hardware-oriented algorithm and architecture for the weighted median filter based on the radix search method. The algorithm is efficient enough to be performed in parallelism and pipelining. Therefore, it can be implemented on VLSI. Since there are many pipelined stages in the proposed architecture, the filtering operation can achieve high throughput rate by block processing. It will complete a block of K-bit output after K iterations. The pipelined cycle can be reduced to 1-bit carry-save-adder(CSA) delay. The concept for finding the median of the Weighted Median Filter(WMF) is introduced in Section 2. In Section 3, we will present a dedicated architecture of the proposed algorithm for the weighted median filter. The final conclusion is given in section 4.

II. A Hardware-oriented WMF Algorithm

In the weighted median filtering application, the weight should be adjustable for different specifications. This adjustment will change the size of the sorter. In addition, the sorting complexity increases when the input data increases. Its cycle time is too long to support the necessary operation speeds for today's high bandwidth system. So, we present a hardware-oriented algorithm, which has a fixed input and constant iterations independent on the weight value defined.

In order to illustrate the weighted median filter operation, consider the following example. Given a WM with W={1,2,3,2,1}, it median (threshold) is M=5, and the input signal is {4,7,5,11,9 }. In the traditional approach, we will duplicate the sample as weight defined before sorting. In this way, the sorter input data becomes to {4,7,7,5,5,5,11,11,9 } and its median is 7. An alternative method to find its output is to check the associated weight sum without sorting. It is evaluated in the bit-serial manner from the most significant bit to the least significant bit. For this example, at the first MSB iteration, the corresponding weighted sum for 9 and 11 component is less than 5. So, the MSB output is 0. And the final median output will be less than 9 and 11. At the second bit iteration, the corresponding weighted sum includes the weights of 4,7,5,11,and 9 components. The corresponding second bit output is 1. But we are not sure that the final output is larger than 4, 7, or 5. It will be determined at the following iterations. The procedure to find the output of the weighted median is summarized in

```
W[.]= [ 1 2 3 2 1 ]

X=[.]=[ 4 7 5 11 9]=[0100 0111 0101 1011 1001]

Tw=9, M=5,
k=4,
LS[.]=[. . . . .] ->0 +0 + 0 +2 +1 = 3 <5    Y(4)=0,
k=3,
LS[.]=[. . . 1 1] -> 1+2 +3 +2 +1 =9>=5   Y(3)=1,
k=2,
LS[.]=[. . . 1 1] -> 0 +2 + 0+ 2 +1=5>=5    Y(2)=1,
k=1,
LS[.]=[0 . 0 1 1] -> 0+ 2 +0 +2 +1 =5 >=5    Y(1)=1,

The weighted median filter output= Y(0111)=7,
```

Figure 1: A bit-serial example of finding the output of the weighted median filter

Fig. 1.

This hardware-oriented algorithm is executed in the bit-serial manner, which is executed from the most significant bit to the least significant bit. Let the total weight sum of the weighted median filter be T_w and its median be M $=(T_w+1)/2$. At the first iteration, the MSB of output is set to "1". The weight of sample with "1" at the MSB are added up. Then check this weight sum value, whether it is larger than or equal to the value M. If it is true, the setting is correct and the corresponding bit is "1"; otherwise it is "0". In the former case, the samples whose MSB bit is 0 are labeled "0". In the latter case, the samples whose MSB bit is 1 are labeled "1". All determined samples will keep their labels in the following iteration cycles. Similar to the first iteration, the second bit is also determined in the same manner, and so on. For 8-bit input data, the final median output can be obtained after 8 iterations. The computation complexity of the proposed realization is O(K), where K is the word length. The cycle time of this median finding depends on the cycle time of the weight sum operation, which will be reduced further at the next section by the hardware approach.

It is clearly that the main improvement of our proposed method, compared with other previous bit-serial algorithms[8,9] is that the proposed algorithm can find the median output with various definitions of weight. In Chen[8]and Kar[9], they can only support the median output with unit weight directly. In that situation, the input size of these filters can't remained fixed for the weight median filter realization, which will cause some realization problems. Fortunately, our algorithm can solve this problem easily. For fractioned weight set, we can either normalize or represent the weight values into the binary format without losing their generality before applying the proposed algorithm. The summary of the proposed algorithm is described as follows:

Table 1: An example of a weighted median filter, where the total weight is 9 with 5 input samples and output = 7(0111).

Weight value W[i]	1	2	3	2	1	M=5	T_w=9
Input data $X[i]$	4	7	5	11	9	W_{sum}	Output Y
BCD code $X[i]$	0100	0111	0101	1011	1001	W_{sum}	
Iterations	S_1, L_1	S_2, L_2	S_3, L_3	S_4, L_4	S_5, L_5	*	
K=4,$X[i]_4$	0,0	0,0	0,0	1,1	1,1	3	Y_4= 0
K=3,$X[i]_3$	0,0	0,0	0,0	1,1	1,1	9	Y_3= 1
K=2,$X[i]_2$	0,0	0,0	0,0	1,1	1,1	5	Y_2= 1
K=1,$X[i]_1$	1,0	0,0	1,0	1,1	1,1	5	Y_1= 1

Hardware-oriented algorithm for weighted median filters

Let $X[i]$, W[i], S[i], L[i], Y_k, and $X[i]_k$ represent the input sample value, weight, set or pass, larger or smaller than M, kth bit of Y output, and kth bit of $X[i]$, respectively

for i=1, ... n do in parallel
load W[i], $X[i]$, S[i]=0, L[i]=0
for k=MSB, ...,LSB do in series
for i=1,.... ,n do in parallel
begin
W_{sum}=0;
case (S[i], L[i])
0,0 : { XW[i] = $X[i]_k$* W[i];}
0,1 : { XW[i] = un-used;}
1,0 : { XW[i] = 0;}
1,1 : { XW[i] = W[i];}
endcase
for j=1,....,n do W_{sum}= W_{sum} + XW[j]
if ($W_{sum} \geq M$) then Y_k = 1; else Y_k= 0;
if ($W_{sum} \geq M$) then {if ((S[i]= 0) and ($X[i]_k$=0)) then L[i]= 0;}
else {if ((S[i]=0)and ($X[i]_k$=1)) then L[i]= 1;}
end

An example of the weighted median filter is shown in Table 1. The weighted median filter output is achieved after K iterations, where K=4.

III. Systolic Array architecture of WMF

The proposed algorithm can be efficiently implemented by using systolic array. We mapped the proposed algorithm into three main parts: Word-to-Bit serial conversion (WTBC), weighted sum calculation and comparison, and Bit-serial-to-Word conversion (BTWC). The basic structure of the proposed architecture is shown in Fig. 2. An adder binary-tree is used to accumulate the corresponding weights. Hence, it completes a filtering operation after 8 * (log n +1) cycles to generate 8 bit output. With this cycle time, the delay of every adder must be very short to support the real-time requirement. We adopted a bit-level realization and task interleaving processing to reduce the cycle time and power consumption.

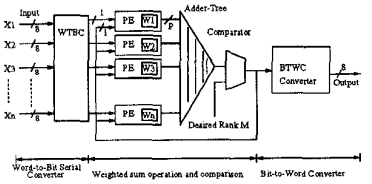

Figure 2: A basic structure of proposed architecture.

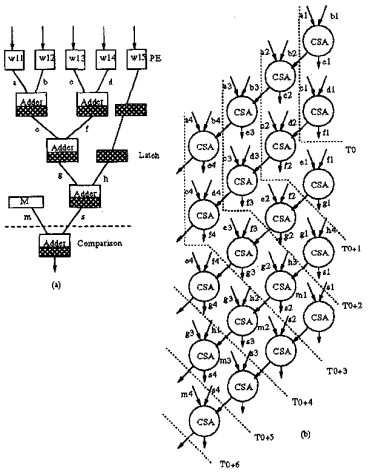

Figure 3: An adder-tree architecture of weighted median filter

At first, the comparative result may be obtained at the MSB output of an adder when the weighted sum value and the 2's complement of M are driven as inputs. With this transformation, all operations may be executed from the LSB to the MSB. Hence, they can be regularly pipelined down to the bit-level to achieve high throughput rate. Fig. 3 illustrates an adder-tree and its corresponding bit-level to accumulate 5 weights.

Since there are many idle or waiting cycles in the adder-tree, many independent tasks may be executed concurrently with task interleaving processing. In this case, the filtering operation is performed using multiple filter input with multiple filter output as block processing. Therefore, it can concurrently complete 14 tasks in the proposed architecture, where "task" means the process to to find a single output. The detailed consideration of PE function design, data flow, and related technology to support this architecture are given in the subsection 3.1 and 3.2. The evaluation of the proposed architecture is also summarized in the subsection 3.3.

A. PE Processor

A PE processor constructed for each pixel will output "0"value or "weight" value to the input of the adder-tree depending on the the compared result, (Y) and corresponding bits, (X[i], L and S). The corresponding state

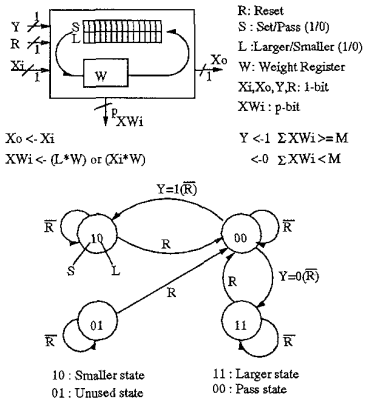

Figure 4: A state diagram of PE function.

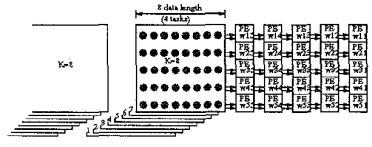

Figure 5: The 2-D array of the weighted median filter with 5*5 window size (4 tasks)

diagram of PE is shown in Fig. 4. When the label can be determined, it will be set to S=1 and stored as the correct label at register L. Although the PE state may not be determined immediately until the delay cycle of weight sum operation, it is available for the evaluation of the next task. Hence, we correct these S and L registers constructed as a register-ring in PE for corresponding task. And this ring will shift one position per clock cycle corresponding to the proper interleaved task.

B. Data Flow

In the practical implementation, the input bandwidth must be reduced or shared with other input for saving VLSI fabrication and packaging cost. And the data flow of the proposed architecture must be regular to minimize the number of skewing registers and latch registers. Therefore, we arrange the PE processor into a 2-D array as shown in Fig. 5. Each PE processor will receive the bit-streams from WTBC converter or from its left side and shift them to its right.

Since the feedback loop has more than one delay cycle, it will generate idle cycles for task switching or data waiting, and degrade the performance of the throughput. We use a pair of bit-streams to carry input in sequence as shown in Fig. 6 to improve the efficiency to 100%. In the old iteration cycle, the data of the corresponding filtering window is cut and obtained from the bit-stream A. In the even iteration cycle, it will switch to the bit-stream B. In this way, the idle cycle is shrunk. There is only one bit width for each bit-stream, which can serve on the VLSI implementation. We used a WTBC converter with a circle-buffer to generate these bit streams as shown in Fig. 7. Since the filtering operation is performed in block processing, the MSB bit plane of the block is generated first, then the second plane, and so on. Since the conservative blocks have overlapped data, they may be reused and left at the circle-buffer. These bit planes are shifted into the 2-D array of PEs to perform the state evaluation for the corresponding tasks. The weighted median output comes out of the MSB bit plane to the LSB bit plane, which are stored in shifted registers before being converted to the word level output. The output rate is 14 tasks per 112 cycle time instances.

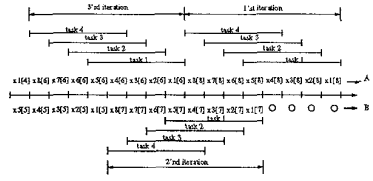

Figure 6: Data flow for task interleaving (4 tasks).

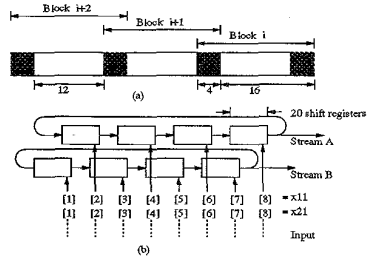

Figure 7: (a)The overlapped data of the conservative block. (b) The WTBC converter for one row input.

C. Hardware Evaluation

The proposed architecture is realized to support weighted median filters with n window size and the corresponding total weight, N. For comparison, we chose the PBF implementation[8] and the H-tree algorithm[9] constructed for realization of the rank order filter with N inputs as references. The connection network for that of the PBF implementation and the H-tree algorithm is not included. The evaluation criteria of Table 2 summarize the complexity, the latency, the pipelined cycle time, and the throughput rate. The throughput rate is enhanced in the proposed algorithm and architecture.

Since the proposed architecture had been pipelined down to the bit-level, the clock rate may be lower than that of the word-level design. Hence, the power consumption is reduced.

D. Applications of the Weighted Median Filter

Before executing the weighted median filter, the weight parameters of the filter must be obtained in advance by

Algorithm	Our Algorithm	PBF Algorithm	H-Tree Algorithm
Filter Design	WMF, and ROF	ROF	ROF
Complexity	O(K·Log n)	O(K·2^N)	O(K·LogN)
Sum Operation	n adders	1 PBF	N adders
Latency	K·[LogN+Logn+1] ·T_{CSA}	K·T_{pbf}	K·LogN· T_{adder}
Pipelined Cycle	T_{CSA}	T_{pbf}	Log N · T_{adder}
Throughput Rate	K·T_{CSA}/task	K·T_{pbf}/task	K·[LogN]·T_{adder} /task

Table 2: Evaluation of proposed algorithm, PBF algorithm, and H-tree Algorithm, where n means the window size and N($\geq n$) the total weight sum.

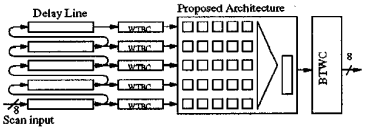

Figure 8: A system block of the proposed architecture for the Weighted Median Filter.

the training algorithms, structural algorithms, or adaptive stack algorithms. Then these weights must be down loaded into the weight registers to perform the filtering operation. In the training algorithms, this dedicated architecture may speedup the cycle time of filtering to get optimal weights. That is because in each cycle, it needs to perform the filtering operation of full image data with given weight. An example of a training algorithm based on this hardware-oriented algorithm has been proposed in [9]. The system block for this filter's design is shown in Fig 8. The output rate is 14 tasks per 8*14 cycle time of the 1-bit CSA adder. At the left side, 5 rows of line buffers are used to convert the scanned input to block output. Each WTBC loads 18 units of the data from the corresponding line buffer and generates bit planes. At the 8-th iteration of the filtering operation, the WTBC not only generates bit plane to PEs but also pre-loads the 14 data units concurrently for the next block operation, and so on. The bit serial output of the boundary value (comparator's output) is stored and converted to a word as final output at BTWC converter.

IV. Conclusion

The arithmetic method for finding the median of the weighted median filter is proposed. It requires K iterations of the weight sum operation to complete the full operation of the median finding. The adder tree architecture is adopted to reduce each cycle time. With task interleaving, the adjacent tasks can be mutual-exclusively interleaved in order to fill the pipeline 100%. The throughput is increased correspondingly to the number of the inserted tasks. The proposed design is regular to pipeline these tasks in bit-levels for high-speed applications. This hardware-oriented algorithm and dedicated architecture for the weighted median filter has several advantages: 1) easy expandability, 2) the very short pipeline cycle, and 3) the output property can be controlled using the programmable weight and rank selection.

References

[1] B.I.Justusson, " Median filtering: Statistic properties," in Topic in Applied Physical, Two-Dimensional Digital Signal Processing II, T. S. Huang, Ed. Berlin: Springer, 1981.

[2] D. R. K. Brownrigg, " The weighted median filter," *Commun. Assoc. Comput. Mach.* , vol. 27, pp. 807-818, Aug. 1984.

[3] P.D.Wendt, E.J.Coyle, and N. C. Gallagher, "Stack Filters," *IEEE Trans. on Acoust.,Speech, and Signal Processing,* vol. 34, pp. 898-911, 1986.

[4] S. J. Ko and Y. H. Lee, "Center Weighted Median Filters and Their Application to Image Enhancement,"*IEEE Trans. on Circuits and Systems*, vol. 38 pp. 984-993,1991

[5] J. H. Lin, T. M. Sellke, and E. J. Coyle, "Adaptive Stack Filtering Under the Mean Absolute Error Criterion," *IEEE Trans. on Acoust. ,Speech, and Signal Processing,* vol. 38, No. 6, pp. 938-954, 1990

[6] L. Yin J. T. Astola, and Y. A.Neuvo, "Adaptive Stack Filtering with Application to Image Processing,", *IEEE Trans. on Signal Processing,* vol. 41, pp. 162-184, 1993.

[7] R. Yang, M. Gabbouj, and Y. Neuvo, "Fast algorithms for analyzing and designing weighted median filters,",*Signal Processing* , vol 41, pp. 135-152, 1995.

[8] K. Chen, "Bit-serial Realizations of a class on Nonlinear filter based on Positive Boolean Function," *IEEE Trans. on Circuits and Systems*, vol. 36, pp. 785-794,1989.

[9] B. K. Kar and D. K. Pradhan, " A New Algorithm for Order statistics and Sorting," *IEEE Trans. on Signal Processing,* vol. 41. No. 8 Aug. 1993, pp. 2688-2694.

[10] C. T. Chen, L. G. Chen, T. D. Chiueh, and J. H. Hsiao, " Design and VLSI Implementation of Real-Time Weighted Median Filter," *1994 IEEE Asia-Pacific Conference on Circuits and Systems,*" Taipei, pp. 91-97,1994.