Bit-sliced median filter design based on majority gate

C.L. Lee
C.-W. Jen

Indexing terms: Filters and filtering, Boolean algebra

Abstract: There are arithmetic problems for the hardware realisation of bit-level median filtering algorithms. A design of a majority gate which is composed of output-wired inverters is proposed. The area and time complexities are better than the digital and analogue designs now available. This circuit is applied to a median filter design which is based on majority selection, the computation problems are thus avoided. It is a bit-sliced architecture with constant cycle time. Window shapes can be arbitrarily changed through mask-and-set modules. A median filtering system for two-dimensional image processing is presented. A binary majority gate is also an essential element in decision-making circuitry which is applied in fault-tolerant computing systems, artificial neural networks or related applications.

1 Introduction

Signal smoothing using median filters has grown popular in recent years because of its simple operation and robust performance. A window with an odd number of elements is defined which slides across the digitised input sequence. The median filter simply takes the middle value of the elements lying in the window, as the window moves through the input sequence step by step. Such operation can remove impulsive types of noise while preserving sharp edges and the desired features of an image [1]. Median filtering techniques are widely applied in many fields, such as reducing 'sparkle' noises in digital TV images [1], improving digital speech quality in PCM channels [3], enhancing edge gradients of signals [4], edge detections [5], seismic data processing [6] and medical image applications [7].

Many algorithms and methods have been developed and used in median selection. They can be classified into two categories: word-level and bit-level. In word-level algorithms, the basic operation is applied to a word. The selection of the median value from a sorted sequence is the simplest method. Bubble sort, selection sort, quick sort, or odd–even transposition sort are common examples [8]. An updated histogram method [9] picks up the median value in a partially modified histogram. A moving border method [10] searches for the median element along a moving boundary on the sorted data matrix. Both methods have the advantage that only a minor part of the data come in and go out on each movement of the window. However, they are not suitable for hardware implementation because of their irregular data structure and operations. Only the odd–even transposition sort has been chosen to realise the median filtering in hardware [11–14].

In bit-level algorithms, the median result comes out with one bit at a time. Usually, there is a mask vector to define the effective subset in which the target result lies. The effective subset will shrink as the inspection proceeds from the most significant bits (MSBs) to the least significant bits (LSBs). The major difference is the counting schemes they perform. In Reference 15 the number of bits '0' among the effective subset at each bit position were counted. In Reference 16, bit '1' was counted in each inspection. Both bit '0' and '1' were counted separately in Reference 17 and either of the two numbers was used for the subsequent calculation. Hardware architecture designs for the last two algorithms were proposed in References 18 and 19, respectively. The algorithm in Reference 15 was formulated mathematically in Reference 20. Recently, two similar algorithms based on majority selection were developed independently in References 21 and 22, but with different concerns. The majority selection can be a special case of rank selection based on the positive Boolean function discussed in Reference 23.

In hardware implementation, algorithms based on binary radix are better for the following reasons:

(*a*) It is more intuitive and simple to derive combinational functions on binary variables.

(*b*) The basic modules are small, regular and highly repeatable.

(*c*) Their hardware complexity increases linearly with window size and word length of binary representation.

Most of the word-level algorithms are more suitable for software implementation. There is yet another bit-level method which selects the median from a sequence of threshold decomposed signals [24]. However, its complexity increases exponentially with word length and therefore it is not practical in hardware implementation.

These bit-level median filtering algorithms are basically performing a binary search among the unsorted data, while the masking functions are similar. The difficulty in hardware implementation is the counting circuit because it is either implemented by a large adder tree or by a large combinational Boolean function for the speed consideration. Now we consider whether there is a method that can reduce the problem while preserving the high speed of throughput. The majority gate is the target.

2 Design of a majority gate

A binary majority gate as shown in Fig. 1 is a circuit that can determine the majority of binary signals. W is the

Paper 8291G (E10), first received 29th May 1990 and in revised form 18th February 1991

The authors are with the Institute of Electronics, National Chiao Tung University, 75 Po-Ai Street, Hsinchu 30039, Taiwan, Republic of China

number of inputs which is usually odd. The output will be '1' if over half of its inputs are '1', otherwise it will be '0'. A number of designs can perform this job. They are briefly discussed in the following sections.

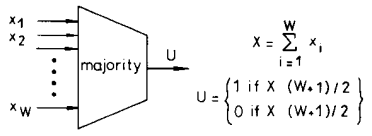

Fig. 1 *Function of the majority gate*

2.1 Some existing majority designs

2.1.1. Comparison after summation: This is a direct digital implementation of the equation shown in Fig. 1. An adder tree can be used to obtain the speed of summation. There is a comparison logic which flags the majority result. If a binary majority function is implemented by an adder tree followed by a comparator, its hardware will expand linearly with W while the delay time increases with log W.

2.1.2 Sorting network with position selection: This method sorts the binary input signals into an ordered sequence. The majority signal will be on the middle line of these sorted outputs. Some discussion about the binary sorting net is presented in Reference 25. The area complexity of this circuit grows at an order of $O(W^2)$ because the odd–even transposition sorting network was used [8].

2.1.3 Positive Boolean function [23, 26]: Instead of direct summation as in Fig. 1, one can implement the majority by a fully combinational Boolean function. For example, if there are three binary signals, a, b and c, the majority function will be $U = ab \vee bc \vee ca$, where $\vee$ denotes a logic OR function. Such a design needs $C^W_{(W+1)/2}$ combinations of product terms, so it is not practical as W becomes large.

2.1.4 Threshold logic gate: A majority function is a special case of a threshold logic gate when the threshold T is equal to $(W + 1)/2$. An early version of circuit implementation of a threshold-logic gate was a voltage divider by resistor–transistor logic (RTL) circuits [25]. A MOS transistor version of the same circuit was presented in Reference 28, which had a resistor network for weighting inputs and a voltage source in series with the drain to determine threshold level. Although this design is simple, the resistors are area-consuming.

2.1.5 Voltage level comparison: This is an analogue approach which detects the difference between a voltage divider output and a reference voltage. A design example is Reference 24 realised the voltage divider by nMOS circuits and compared the voltage levels by a differential amplifier.

2.2 Device programmable CMOS majority gate

Our majority circuit design is based on the voltage divider in Reference 24, but CMOS technology is used instead of nMOS design. The differential amplifier is replaced by an inverter to simplify the design. This can save the area of the differential comparator and half the number of input signals as in the nMOS voltage divider approach; both positive and negative inputs were required in the nMOS voltage divider.

The majority gate is shown in Fig. 2. It is made up of two parts: a nonlinear voltage divider built by output-wired inverters on the left-hand side and an inverting buffer which senses the majority transition and provides a positive output is on the right. In addition, this inverting buffer serves another two purposes: it isolates the divider output node from external circuitry to reduce

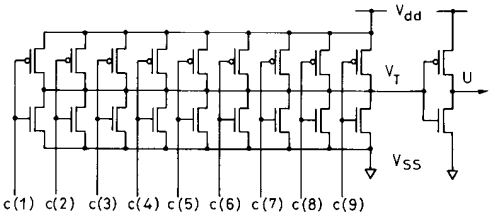

Fig. 2 *Circuit of the 9-majority gate*

noise effect and driving for the next stage and it reshapes the output waveform. The characteristics of this majority circuit obtained by SPICE simulations based on a 1.2 μm CMOS N-well technology is shown in Fig. 3A. Let T be

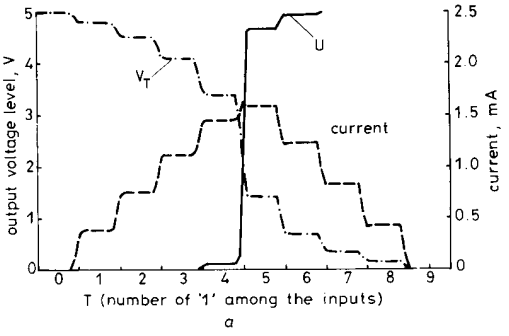

Fig. 3A *Transfer characteristics of the 9-majority gate*

the number of '1's appearing at the input side, so

$$T = \sum_{i=1}^{W} c(i)$$

As T increases, the output voltage V_T steps down a little at first, then the step size broadens at some middle values. Thereafter, the step size shrinks again and finally goes to zero as all input bits become '1'. The current through this divider is changed in the same way as the output voltage step size; it increases at the middle values and goes down to zero at both ends. This phenomenon tells us that this circuit is like a spatial inverter, as the output will be '1' if '0' is the majority among the input signals, and *vice versa*. The underlying mechanism is that the dynamic behaviour of a single inverter is spatially quantised by the W-input divider. Fig. 3B shows the I–V characterististics of pMOS and nMOS transistors when the number of parallel transistors (T_p or T_n) is increased. The intersections indicated (where $T_p + T_n = 9$) are just the nonlinearly quantised voltage levels and current values of the voltage divider shown in Fig. 3A.

A conventional inverter is the simplest threshold circuit in CMOS design. It is connected to the voltage divider to detect the maximum output transition gap. There are several ways to adjust the threshold voltage of an inverter [27], such as changing the effective surface state density (N_{SS}) by implantation, changing the oxide thickness (T_{ox}) or adjusting channel width ratio (W_p/W_n) of the p-channel and n-channel transistors when their

channel lengths are fixed. The last one of these three techniques may be the best choice because it is process-independent and the easiest to design. If we choose nine inputs, for example, the majority transition should appear when T goes from four to five. Note that nine

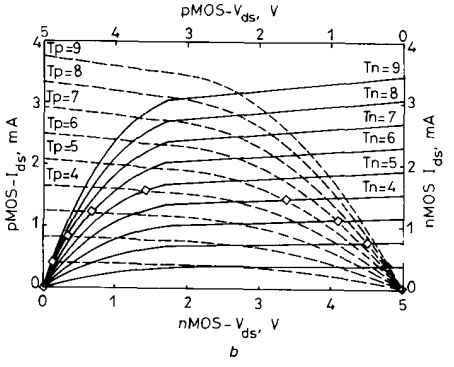

Fig. 3B *Dynamic behaviour of the nonlinear voltage divider*

invertors of the voltage divider should have identical geometrical structures. The results of tuning their W_p/W_n ratios to see their voltage transitions from $T = 4$ to $T = 5$ are plotted in Fig. 4. It is better to have the gap as

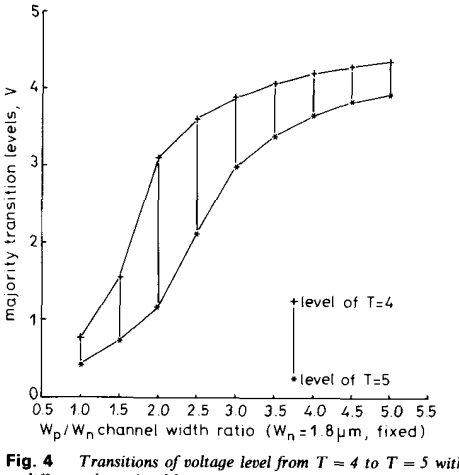

Fig. 4 *Transitions of voltage level from $T = 4$ to $T = 5$ with respect to different channel width ratios*

large as possible. $W_p/W_n = 4.0/1.8 = 2.22$ is chosen from Fig. 4 with $V_4 = 3.38$ V, $V_5 = 1.42$ V and the middle point = 2.40 V. From the plot of threshold voltage against channel width ratio shown in Fig. 5, $W_p/W_n = 3.6/1.8 = 2$ is picked up as the output inverting buffer for its $V_{th} = 2.39$ V, which satisfies the middle point requirement. The resulting majority circuit made up of these two choices is simulated and the output transfer curve has been shown in Fig. 3A.

For the same design process, a majority circuit with 25 inputs has also been designed and simulated. Its layout dimensions are listed in Table 1 with a 9-majority circuit. Note that the 1.2 ns time delay is equal to two invertor delays. The power consumption is calculated by assuming T is uniformly distributed. This design is low

Table 1: Specifications of two majority gates

		9-majority	25-majority
Divider	$\frac{(W/L)_p}{(W/L)_n}$	$\frac{(4.0/1.2)}{(1.8/1.2)}$	$\frac{(3.0/1.2)}{(1.5/1.2)}$
Buffer	$\frac{(W/L)_p}{(W/L)_n}$	$\frac{(3.6/1.2)}{(1.8/1.2)}$	$\frac{(3.0/1.2)}{(1.5/1.2)}$
Number of transistors		20	52
Delay time		1.2 ns	1.3 ns
Power consumption		3.88 mW	9.86 mW

power and high speed. For further considerations, the transfer characteristics of a majority circuit are affected by process variations. Its performance can be influenced by operating temperatures and supply voltage. All these variations were investigated. As the variation in supply

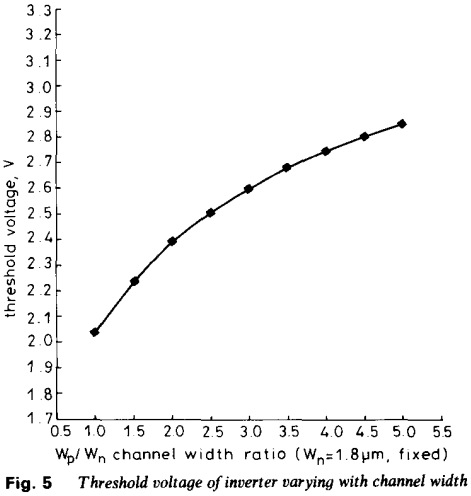

Fig. 5 *Threshold voltage of inverter varying with channel width ratios*

voltage has little effect on the overall performance, it is negligible in the following discussion.

For process variation, the worst case parameters provided by this 1.2 μm CMOS N-well process were used for simulation. We found that under a certain worst combination of parameters, the maximum transition may be shifted higher or lower in steps as shown in Fig. 6*a*. Similar effects can be seen under temperature variation. The transition will be shifted higher at very low temperatures and shifted lower at very high temperatures as shown in Fig. 6*b*.

To ensure the proper operation of the majority function, the misdetection of transition level by process variation should be avoided. As we see in the 9- and 25-majority circuit transfer curves, the number of nonlinearly quantised output voltage levels is $(W + 1)$. At the same time, the maximum transition gap is getting smaller as W increases. As the variations of process and operating conditions may cause more variations on the drain-to-source current of pMOS transistors, the intersections indicated in Fig. 3B will be moved. According to the simulation results, the voltage level variation of a majority transition gap may be as high as 1–2 V in 25-majority under variations of process or temperature. The same variation for 9-majority is about 1–1.5 V. The noise margin calculated for the V_T node of a 9-majority is 0.5 V. That means the 9-majority circuit can tolerate the

process variation if $(V_{IH} - V_{IL})$ of the buffering invertor is 1 V. However, the noise margin of 25-majority is zero in the best case, which is why the 25-majority is greatly influenced by process variation and external changes.

when W_p/W_n is between 2 and 2.5, it happens at $T = 4$–5 and when the ratio falls to within 3 and 4, it appears at $T = 5$–6. These phenomena are plotted in Fig. 7. As a matter of fact, one can have the desired maximum

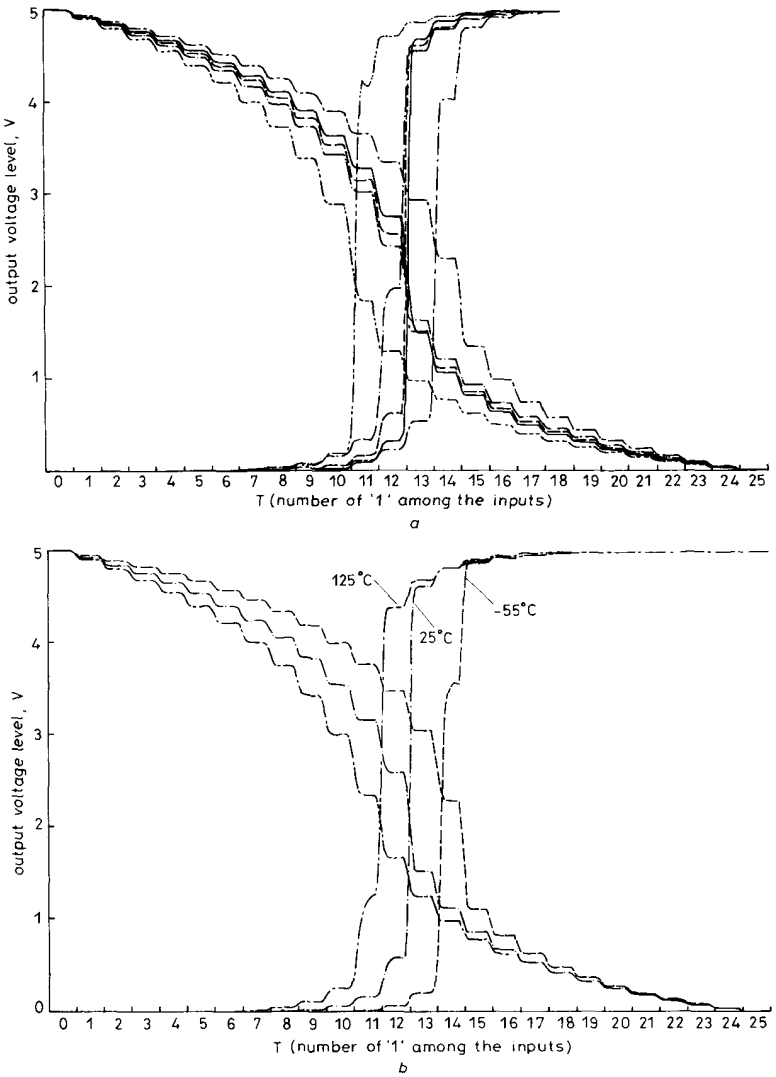

Fig. 6 *Transfer curves of the 25-majority*
a under process variations
b under temperature variations

We think that the majority circuit below nine inputs can be designed and can work properly under the variation of the process we considered. However, the yield will be lower as more inputs of the majority circuit are required. If this is desired, a more carefully controlled process is required. For an even higher number of inputs, some extra circuitry should be involved to compensate for the process variation and detect the smaller transition gap.

During the simulations with varying channel width ratios, we found that the maximum transition was shifted as W_p/W_n increased. For example, when $W_p/W_n = 1.8/1.8 = 1$, the maximum transition step is at $T = 2$–3, voltage transition gap by adjusting only the channel width ratio. Together with a proper choice of W_p/W_n ratio of the inverting buffer, the same circuit structure of the majority circuit will become a threshold logic gate with equal weighting on each input. The threshold level is programmed by the channel width ratio of invertors. The majority gate thus becomes a special case.

In summary, there are three steps in building a majority circuit: First, the majority transition gap characteristics with respect to different channel width ratios should be obtained; secondly, the threshold voltage variations of invertor with respect to different channel width ratios should be available; thirdly, a proper W_p/W_n ratio with a

maximum transition gap must be selected for the nonlinear voltage divider and an invertor in which the threshold voltage equals the middle point of this gap. The cascading of the two devices makes up the majority gate.

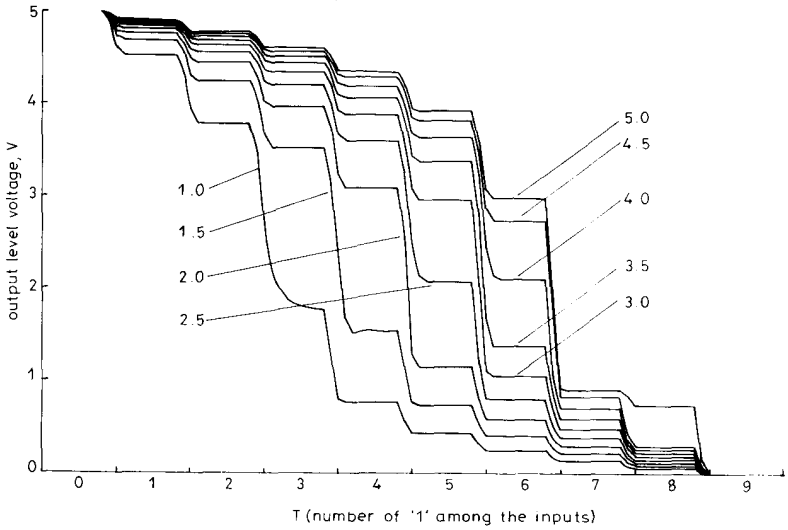

Fig. 7 *Transfer curves of nonlinear voltage divider with respect to different channel width ratios*

2.3 Comparison

Table 2 summarises some features of the new majority gate design in comparison with those designs presented in Section 2.1. The two columns furthest to the right are obviously better than the others because of the linear area complexity and constant cycle time. However, our new design is much better in several ways. First, the analogue divider approach needs two voltage dividers for both the input signal and the reference, while ours needs only one; so the transistor number is halved. Secondly, they need both complement and noncomplement signals for nMOS design while CMOS needs only noncomplement signals. Thirdly, a digital circuit is much easier to manipulate than an analogue circuit.

The simple and regular CMOS design of this majority gate is very attractive, however the programming of device geometries for different rank orders in the designing stage decreases the run-time flexibility. This is because the rank-order is fixed after the majority gate is designed.

3 Bit-level median filtering algorithm based on majority selection

In a one- or two-dimensional situation, the total number of elements lying in the sliding window can be denoted by W (or $W = W_1 * W_2$), which is called the window size. The data within the window are written as $\{a(i) \mid i = 1, \ldots, W\}$. These data are taken to be non-negative integers, as, in most of the applications, gray level images have no negative pixel and digital signals can also be offset to be positive. If each element in the window has an N-bit binary representation, then the data value $a(i)$ within the window can be defined as

$$a(i) = \sum_{j=0}^{N-1} b_j(i) \cdot 2^j$$

where $b_j(i) \in \{1, 0\}$, for $i = 1, \ldots, W$.

For the kth most significant bits (the $(N - k + 1)$th least significant bits) of all data, a binary row vector $\boldsymbol{B}_k$ can be formed which is defined as

$$\boldsymbol{B}_k = [b_{N-k+1}(1), b_{N-k+1}(2), \ldots, b_{N-k+1}(W)] \quad \text{for } k = 1, \ldots, N$$

Corresponding to binary row vector $\boldsymbol{B}_k$, two additional binary vectors $\boldsymbol{M}_k$ and $\boldsymbol{S}_k$ are defined for later use. They are

$$\boldsymbol{M}_k = [m_k(1), m_k(2), \ldots, m_k(W)]$$

and

$$\boldsymbol{S}_k = [s_k(1), s_k(2), \ldots, s_k(W)] \quad \text{for } k = 1, \ldots, N$$

where $m_k(i)$ and $s_k(i) \in \{0, 1\}$ for $i = 1, \ldots, W$.

$\boldsymbol{M}_k$ is referred to as a masking vector and $\boldsymbol{S}_k$ is referred to as a setting vector for $\boldsymbol{B}_k$.

3.1 The algorithm

Generally speaking, a bit-level median filtering algorithm determines the kth most significant output bit by inspecting the kth most significant bits of all elements in the

Table 2: Comparison for various designs of majority gate

	Tree adder + comparison	Sorting network	PBF	Analogue divider	Majority
Area complexity	$O(W)$	$O(W * W)$	$O(W!)$	$O(W)$	$O(W)$
Delay time	$O(\log W)$	$O(W)$	$O(1)$	$O(1)$	$O(1)$
Technology	MOS	MOS	CMOS	nMOS	CMOS
Circuit	digital	digital	digital	analogue	digital
Rank order selection	by circuit function	by output line selection	by Boolean function	by input assignment	by device geometries
References	18, 19	25, 8	23, 26	24	

window. Starting from the first MSB, one checks whether '1' or '0' is the majority and the median is in the subset of which the MSB is the majority bit. Thus we set the masking-flags $m_k(i)$ to be '1' to indicate the desired subset where the median value stays, and force the elements which are not in the desired subset to a local extreme value by putting the corresponding setting-flags $s_k(i)$ to be the opposite value of the present output bit. Once the $m_k(i)$ becomes '0' the ith element in the window is no longer in the desired subset and the setting value will take over for the rest of the calculation, i.e. $m_l(i) = 0$ and $s_l(i) = s_k(i)$ for $l = k + 1, \ldots, N$. For binary signals, the median and majority value are the same. The setting-flags help to preserve the rank order of the expected median result, so that the majority selection is happening in each bit position from MSB to LSB.

Let the kth MSB of median output result be denoted as u_k and let $c_k(i)$ be a temporary signal corresponding to $b_{N-k+1}(i)$ and $s_k(i)$. We use '$\wedge$', '$\vee$' and '$\sim$' to denote the logic AND, OR and NOT operations, respectively. The new median filtering algorithm can be formally written as follows:

(i) Initially, all elements in the window are in the desired subset $m_1(i) = 1$, for $i = 1, \ldots, W$

(ii) The following statements (steps (iii)–(vi)) are repeated from MSB to LSB: $k = 1$, the first MSB

(iii) Generate the kth MSB of median output u_k by finding the majority of intermediate signals $c_k(i)$.

$$c_k(i) = (m_k(i) \wedge b_{N-k+1}(i)) \vee (\sim m_k(i) \wedge s_k(i)) \quad \text{for } i = 1, \ldots, W$$

$$u_k = \text{maj}\,\{c_k(i) \mid i = 1, \ldots, W\}$$

(iv) If $k = N$, then stop

(v) For the masking operation, define the desired subset for the next inspection:

$$m_{k+1}(i) = m_k(i) \wedge ((u_k \wedge b_{N-k+1}(i)) \vee (\sim u_k \wedge \sim b_{N-k+1}(i))) \quad \text{for } i = 1, \ldots, W$$

(vi) For the setting operation, assign values to the entries outside of desired subset:

$$s_{k+1}(i) = (\sim m_k(i) \wedge s_k(i)) \vee (m_k(i) \wedge \sim u_k) \quad \text{for } i = 1, \ldots, W$$

(vii) $k = k + 1$, goto step (iii).

The main idea of this new algorithm is to get the median output without changing its rank order. In other methods [15–17], the rank order of median value had been changed during the calculation. Here the point is to compress the values which are not in the desired subset to be a local extreme value. This process guarantees that rank order of median value is unchanged. The advantage of preserving its rank order is that majority selection alone is enough and there is no need for arithmetic operations such as addition or subtraction. An example which demonstrate the new algorithm is shown in Fig. 8 with $N = 4$ and $W = 9$. Those elements which are not in the desired subset are marked by a circle and the neighboring S values are then taken instead to make a majority decision.

3.2 Discussion

There are two special properties of this algorithm: First, the majority is the median in a set of binary signals with an odd number of elements. Secondly, the mask-and-set operations presented in steps (v) and (vi) of the algorithm will preserve the rank order of median value through all N cycles of inspections. The former statement is clearly valid and the latter statement is true because they are specially designed to do so.

For $k = 1$, the first cycle, we are finding the $(W + 1)/2$th large (also the $(W + 1)/2$th small) data value in the set of W elements, where W is odd. Let T be the number of '1's in $b_N(i)$ of the set defined by $m_1(i)$ for $i = 1, \ldots, W$, i.e.

$$T = \sum_{i=1}^{W} m_1(i) \wedge b_1(i)$$

	j=3			j=2			j=1			j=0		
	b	s	m	b	s	m	b	s	m	b	s	m
a(9)=2	0		1	0		1	(1)	0	1	(0)	0	1
a(8)=14	1		1	(1)	1	0	(1)	1	0	(0)	1	0
a(7)=7	0		1	1		1	1		1	1		1
a(6)=3	0		1	0		1	(1)	0	0	(1)	0	0
a(5)=5	0		1	1		1	0		1	(1)	0	0
a(4)=8	1		1	(0)	1	0	(0)	1	0	(0)	1	0
a(3)=13	1		1	(1)	1	0	(0)	1	0	(1)	1	0
a(2)=11	1		1	(0)	1	0	(1)	1	0	(1)	1	0
a(1)=6	0		1	1		1	1		1	0		1
majority	u(3)=0			u(2)=1			u(1)=1			u(0)=1		

Fig. 8 *Example for the median selection algorithm*

3.2.1 Case 1

If $T \leqslant (W - 1)/2$, $u_1 = 0$, because 0 is the majority. Those data values which have $b_N(i) = 1 \neq u_1$ should be masked out of the desired subset by letting $m_l(i) = 0$ for $l = 2, \ldots, N$. After this masking operation, T elements have been removed. Thus, for $k = 2$, we will find the $[(W + 1)/2 - T]$th large data value in $(W - T)$ of elements. To preserve the rank order of median value, we have to set those elements which have $m_2(i) = 0$ to $s_l(i) = \sim u_1 = 1$ for $l = 2, \ldots, N$. The median value will then go back to the $(W + 1)/2$th large data value in W elements. (It is still the $(W + 1)/2$th small data value in W elements.)

3.2.2 Case 2

If $T \geqslant (W + 1)/2$, $u_1 = 1$, because 1 is the majority. Those data values who have $b_N(i) = 0 \neq u_1$ should be masked out of the desired subset by letting $m_l(i) = 0$ for $l = 2, \ldots, N$. Thus, $(W - T)$ elements have been removed. For $k = 2$, we will then find the $(W + 1)/2$th large data value among the rest T elements. That is the $[T - (W + 1)/2]$th small value in the T elements. To preserve the rank order of median value, we have to set those elements which have $m_2(i) = 0$ to $s_l(i) = \sim u_1 = 0$ for $l = 2, \ldots, N$. The median value will then go back to the $(W + 1)/2$th small data value in W elements. (It is still the $(W + 1)/2$th large data value in W elements.)

For $k \geqslant 2$, the setting values are taken into account to preserve the median rank. T should be calculated as

$$T = \sum_{i=1}^{W} (m_k(i) \wedge b_{N-k+1}(i)) \vee (\sim m_k(i) \wedge s_k(i))$$

The majority decision for $k \geqslant 2$ is the same as in cases 1 and 2 but the modification is held only for those data values which have $m_k(i) = 1$. Those which have $m_k(i) = 0$ are permanently set by $s_k(i)$ throughout the rest of the inspections.

Combining the calculation of cases 1 and 2 from $k = 1$ to N, the masking operation should be steps (v) and (vi) of the algorithm.

4 Flexible median filtering system

The advantage of this algorithm is evident from its hardware design. The mask-and-set operations are simple combinational logic and the majority is implemented by a novel design without arithmetic drawbacks.

4.1 Word-parallel and bit-pilelined design

In addition to the majority circuit, all functions are included in a mask-and-set (M/S) module. As the algorithm has been written in a single assignment form, every variable is assigned only once. The Boolean functions in an M/S module can be obtained by direct transformation from the software statements of steps (iii)–(vi). Let M, S, C and B be the masking, setting, intermediate and binary data bit, respectively. If we carefully check the Boolean function of signals C and S, they are the same, which was verified in Reference 23. An M/S module is presented in Fig. 9. Every M/S module receives M and S signals from

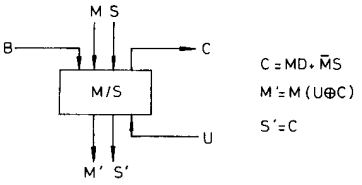

Fig. 9 *M/S module and its functions*

the previous stage with the current input data B to generate an intermediate signal C. The maj gate collects these C signals from each M/S module to calculate the median value U and feeds back to each M/S module to modify M' and S' values for the next stage. A single stage of this median selection unit is shown in Fig. 10. This structure

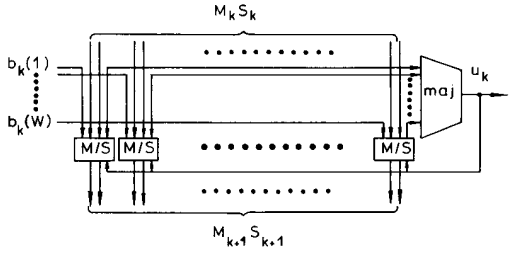

Fig. 10 *One bit stage of the median selection unit*

is rather simple and regular. It can be cascaded into a word-parallel and bit-pipelined design. If M' and S' signals are fed back to the same stage, a bit-serial and word parallel design is formed.

As the cascaded N stages work in a pipeline fashion, the output may not be correct without properly scheduling the I/O data bits, therefore skewing delays are needed to delay one more time unit for each successive stage. The input data bits can then be fired at a correct timing slot. Throughout the filter the output result should be skewed back to its original data word. Therefore, 'deskewing' delays are placed at the output side. Note that the shift register column butted to each stage is called a window buffer. Only one additional row of skewing delays is required for each stage, including both input side and output side, rather than W rows of delays for each stage as presented in similar designs [11, 13, 18]. A bit-sliced architecture with skewing delays is shown in Fig. 11.

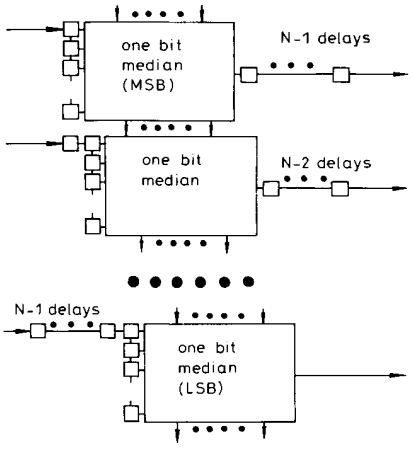

Fig. 11 *Bit-sliced median filter structure with skewing delays*

In the hardware realisation of a bit-level median filtering algorithm, the area complexity is dominated by the basic boxes which are for defining the effective subset and the time delay of a single stage is dominated by the counting circuits. These features are compared in Table 3. The majority design is better because of its linear complexity with constant cycle time.

4.2 System architecture design

Owing to the advance of VLSI technology, many software algorithms are embedded on a single VLSI chip for cost and performance considerations. Modular, regular and repeatable structures are preferred in such a VLSI system. The bit-sliced approach of median filtering provides a bit-level scalable hardware structure which is adaptive to changeable word length. The repeatable nature makes it very attractive to a VLSI design.

In real-time two-dimensional image smoothing, using a median filter with window size 3 × 3, we need buffers to store data in the former two scan lines as the window raster-scans over the image. The structure is as shown in Fig. 12. Scan line buffers stand for temporary storage and a window buffer expands the pixels lying in the window into a column. A shaper butted to the window buffer provides four types of window shapes: square, cross, 'X' and dot, to meet different background noise conditions. As illustrated in Fig. 13, useful pixel bits are indicated by '*', while unused pixel bits are masked by '1' or '0' equally, so that the rank order of desired output result may not be changed. The median selection unit is just the structure shown in Fig. 10.

Table 3: Time and area complexity for hardware realisation of bit-level median filtering algorithms

	Counting '1'	Counting '0'	Counting '1' and '0'	Threshold decomposition	Majority
Area complexity	$O(NW)$	$O(NW)$	$O(NW)$	$O(2^N W)$	$O(NW)$
Cycle time	$O(\log W)$	$O(\log W)$	$O(\log W)$	$O(1)$	$O(1)$
References	16, 18	15	17, 19	24	

In the above description, one may notice that the function of the shaper is somehow similar to the mask-and-set operation. Hence it should be possible to perform these shaping functions by properly setting the values of

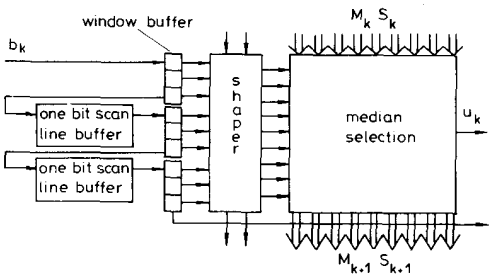

Fig. 12 *One bit stage of median filter architecture for image smoothing with 3 × 3 window*

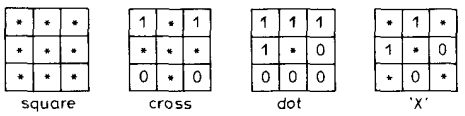

Fig. 13 *Four types of window shape*

$m_k(1)$ and $s_k(1)$ signals. For example, the four shaping functions can be implemented by the following initial assignments:

(*a*) square-window:

$$M_1 = [m_1(1), \ldots, m_1(9)] = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$$
$$S_1 = [s_1(1), \ldots, s_1(9)] = [-\ -\ -\ -\ -\ -\ -\ -\ -]$$

(*b*) cross-window:

$$M_1 = [m_1(1), \ldots, m_1(9)] = [0\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0]$$
$$S_1 = [s_1(1), \ldots, s_1(9)] = [1\ -\ 1\ -\ -\ -\ 0\ -\ 0]$$

(*c*) 'X' window:

$$M_1 = [m_1(1), \ldots, m_1(9)] = [1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1]$$
$$S_1 = [s_1(1), \ldots, s_1(9)] = [-\ 1\ -\ 1\ -\ 0\ -\ 0\ -]$$

(*d*) dot-window:

$$M_1 = [m_1(1), \ldots, m_1(9)] = [0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]$$
$$S_1 = [s_1(1), \ldots, s_1(9)] = [1\ 1\ 1\ 1\ -\ 0\ 0\ 0\ 0]$$

where '–' denotes the 'don't-care' conditions. Thus, through initial assignments of $m_1(i)$ and $s_1(i)$ signals, the window shape can be arbitrarily chosen. Each position in the window can now be assigned separately. This feature allows a very high flexibility in using this structure for a variety of image or signal characteristics.

Testing is a very important issue in today's VLSI systems. In *ad hoc* testing, it is better to partition a large system into several independent submodules so that they can be tested separately. Let us consider the testable design of a single stage; as the other bit-slices can be tested in the same way. Scan lines and window buffers are shift registers or memories in nature and there are standard procedures to test them. In the mean time, they can serve as the scan path buffer for the testing of median selection unit. The problem in testing a median selection unit is the poor observability on signals *C* from the M/S box to the majority gate. Scan path registers can be inserted here to improve the testability of the majority gate. The M/S modules can be tested independently in the same time by 16 patterns for an exhaustive functional test. The majority circuit can be tested in the same way.

5 Concluding remarks

A simple design of a majority gate was proposed, which consists of output-wired invertors. It consumes fewer transistors and has a constant delay time. The programming of majority selection is through the choosing of channel width ratios of *p*- and *n*-channel transistors in CMOS circuits. This majority gate was applied to implement a median filtering algorithm based on majority bit selection. A VLSI system architecture design for two-dimensional median filtering was also demonstrated. It is window-shape changeable, bit-level scalable and easy to implement. It is a flexible system for high speed signal smoothing.

The mask-and-set operation is a basic function in cellular logic array used in signal processing. The architecture proposed in the last section is in fact a special purpose design for two-dimensional image processing. These simple cells should be applicable to many other signal processing applications such as speech/image smoothing, stack filters and morphological filtering.

By adjusting channel width ratios, the majority circuit may becomes a threshold logic gate with equal input weighting . This approach may play an important role in majority-decision applications, such as threshold decoding circuits, fault tolerant systems, binary artificial neural networks and many other threshold decision-involved designs.

6 Acknowledgment

The authors wish to thank the anonymous reviewers for their helpful and constructive comments. This work was supported by the National Science Council, Taiwan, Republic of China, under grant NSC77-0201-E009-01.

7 References

1 ARCE, G.R., GALLAGHER, N.C., and NODES, T.A.: 'Median filters: theory for one- and two-dimensional filters', *in* HUANG, T.S. (Ed.): 'Advance in computer vision and image processing' (JAI Press Inc., London, 1986), Chap. 3
2 NAQVI, S.S.H., GALLAGHER, N.C., and COYLE, E.J.: 'An application of median filters to digital television'. Int. Conf. Acoust., Speech and Signal Processing, Tokyo, 1986, pp. 2451–2454
3 JAYANT, N.S.: 'Average- and median-based smoothing techniques for improving digital speech quality in presence of transmission errors', *IEEE Trans.*, 1976, **CON-24**, (9), pp. 1043–1045
4 FRIEDEN, B.R.: 'A new restoring algorithm for the preferential enhancement of edge gradients', *J. Opt. Aoc. Am.*, 1976, **66**, (3), pp. 280–283
5 STEIN, R.A., and FOWLOW, T.J.: 'The use of median filters for edge detection in noise signals'. Proc. Int. Symp. Circuits and Systems, Kyoto, 1985, pp. 1331–1334
6 BEDNAR, J.B.: 'Applications of median filtering to deconvolution, pulse estimation, and statistical editing of seismic data', *Geophysics*, 1983, **48**, (12), pp. 1598–1610
7 RITENOUR, E.R., TNELSON, T.R., and RAFF, U.: 'Applications of the median filter to digital radiographic image'. Proc. Int. Conf. ASSP, 1984, pp. 23.1.1–23.1.4
8 KNUTH, D.E.: 'The art of computer programming. Vol. 3: sorting and searching' (Addison-Wesley, New York, 1973)
9 HUANG, T.S., YANG, G.J., and TANG, G.Y.: 'A fast two-dimensional median filtering algorithm', *IEEE Trans.*, 1979, **ASSP-27**, (1), pp. 13–18
10 AHMAD, M.O., and SUNDARARAJAN, D.: 'A fast algorithm for two-dimensional median filtering', *IEEE Trans.*, 1987, **CAS-34**, (11), pp. 1364–1374
11 OFLAZER, K.: 'Design and implementation of a single-chip 1-D median filter', *IEEE Trans.*, 1983, **ASSP-31**, (5), pp. 1164–1168
12 DEMASSIEUX, N., JUSTAND, F., DANA, M.: 'A low cost custom IC for real time image median filtering'. IEEE Custom Integrated Circuit Conf., 1986, pp. 166–169
13 KARAMAN, M., ONURAL, L., and ATALAR, A.: 'Design and implementation of a general purpose median filtering VLSI', *in* VLSI signal processing III (IEEE Press, 1988), pp. 111–119

14 CHRISTOPHER, L.A., MAYWEATHER, W.T., and PERLMAN, S.S.: 'A VLSI median filter impulse noise elimination in composite or component TV signals', *IEEE Trans.*, 1988, **CE-34,** (1), pp. 262–267
15 KARAMAN, M., and ONURAL, L.: 'New radix-2-based algorithm for fast median filtering', *Electron. Lett.*, 1989, **25,** (11), pp. 723–724
16 ATAMAN, E., AATRE, V.K., and WONG, K.M.: 'A fast method for real-time median filtering', *IEEE Trans.*, 1980, **ASSP-28,** (4), pp. 415–420
17 DANIELSSON, P.-E.: 'Getting the median faster'. Computer graphic and image processing, 1981, **17,** pp. 71–78
18 ROSKIND, J.A.: 'A fast sort-selection filter chip with efficiently linear hardware complexity'. Int. Conf. Acoust., Speech, Signal Processing, Tampa, USA, 1985, pp. 1519–1522
19 ARAMBEPOLA, B.: 'VLSI architecture for high-speed rank and median filtering', *Electron. Lett.*, 1988, **24,** (18), pp. 1179–1180
20 HOCTOR, R.T., and KASSAM, S.A.: 'An algorithm and a pipelined architecture for order-statistic determination and L-filtering', *IEEE Trans.*, 1989, **CAS-36,** (3), pp. 344–352
21 CHANG, L.W., and LIN, J.H.: 'Bit-level systolic arrays for median filters'. Int. Conf. Acoust., Speech, Signal Process., New Mexico, USA, 1990, 54.D13.10
22 LEE, C.L., and JEN, C.-W.: 'A novel design of binary majority and its application to median filtering'. Int. Symp. Circuits and Systems, New Orleans, 1990, pp. 570–573
23 CHEN, K.: 'Bit-serial realizations of a class of nonlinear filters based on positive boolean functions', *IEEE Trans.*, 1989, **CAS-36,** (6), pp. 785–794
24 HARBER, R.G., BASS, S.C., and NEUDECK, G.W.: 'VLSI implementation of a fast rank order filtering algorithm'. Int. Conf. Acoust., Speech, Signal Processing, Tampa, USA, 1985, pp. 1396–1399
25 HURST, S.L.: 'The logical processing of digital signals' (Grane, Russak & Company, New York, 1978)
26 FITCH, J.P.: 'Software and VLSI algorithms for generalized ranked order filtering', *IEEE Trans.*, 1987, **CAS-34,** (5), pp. 553–559
27 TRONT, I.G., and THAKAR, A.V.: 'An analysis of FET based multiple-valued logic circuits'. Int. Symp. Multiple Valued Logic, Paris, France, 1982, pp. 69–76
28 GLASER, A.B., and SUBAK-SHARPE, G.E.: 'Integrated circuit engineering' (Addison-Wesley, Massachusetts, 1977)
29 HWANG, K.: 'Computer arithmetic, principles, architecture, and design' (John Wiley & Sons, 1979)