A Maskable Memory Architecture for Rank-Order Filtering

Lan-Rong Dung and Meng-Chun Lin

Abstract —*This paper presents a novel implementation of rank-order filtering using maskable memory. Based on a generic bit-sliced rank-order filtering algorithm the proposed design uses a special-defined memory, called parallel maskable memory (PMM) to realize major operations of rank-order filtering, threshold decomposition and polarization. In conventional designs, these operations are usually implemented as logic circuit and require complex computation. Using the memory-oriented architecture, the proposed rank-order filter can benefit from high flexibility, low cost and high speed. PMM has features of bit-sliced read, partial write, and pipelined processing. Bit-sliced read and partial write are driven by maskable registers. The maskable registers allows PMM to configure operating bits for parallel read/write operations. Combining the bit-sliced read with polarization selector allows PMM to perform polar determination while the partial write achieves polarization. Recursively combining the bit-sliced read and partial write, PMM can effectively realizes rank-order filtering in terms of cost and speed.*[1]

Index Terms —rank-order filter, image processing, VLSI architecture, maskable memory, median filter.

I. Introduction

Rank-order filtering, or order-statistical filtering, has been widely applied for various speech and image processing applications [1]-[6]. Given a sequence of input samples $\{x_{i-k}, x_{i-k+1}, \ldots, x_i, \ldots, x_{i+l}\}$, the basic operation of rank order filtering is to choose the *r*-th largest sample as the output y_i, where *r* is the rank-order of the filter. Unlike linear filtering, rank-order filtering can remove sharp discontinuities of small duration without blurring the original signal. Therefore, rank-order filtering becomes a key component for signal smoothing and impulsive noise elimination. To provide the key component for various signal processing applications, we intend to design a configurable rank-order filter that features low cost and high speed.

Articles [8,10-24] have presented hardware implementations of rank-order filtering in the past decades. Many of them are based on sorting algorithm [11, 22, 23, 25-28]. They consider the operation of rank-order of filtering as two steps: sorting and choosing. Papers [8, 10, 19] have proposed systolic architectures of rank-order filtering based on sorting algorithms, such as bubble-sort and bitonic sort. These architectures are fully pipelined to have high throughput rate at the expense of latency, but require a large number of compare-swap units and registers. To reduce the hardware complexity, papers [8, 12, 14, 15, 29-32] present linear-array approaches that maintain samples in sorted order. For a sliding window of size N, the linear-array architectures consist of N processing elements and require three steps in each iteration: finding the proper location for new coming sample, discarding the eldest one, and moving samples between the newest and oldest one position. The three-step procedure is called delete-and-insert (DI). Although the hardware complexity is reduced to O(*N*), they require a large latency for DI steps. Paper [20] presents a parallel architecture using two-phase design to improve the operating speed. In the paper, they first modify traditional content-addressable memory (CAM) to a shiftable CAM (SCAM) processor with shiftable memory cells and comparators and hence take advantages of CAM for parallelizing the DI procedure. Then, they use two-phase design to combine delete and insert operations. Therefore, the SCAM processor can quickly finish DI operations. Although the SCAM processor has significantly increased the speed of linear-array architecture, it only processes a new sample at a time and cannot efficiently process 2-D data. For a window of size *N*-by-*N*, the SCAM processor needs *N* DI procedures for each filtering computation. To have an efficient 2-D rank-order filter, papers [12, 27] present solutions for 2-D rank-order filtering at the expense of area.

Instead of sorting algorithm, papers [10, 13, 16-18, 21, 33, 34] address on bit-sliced majority algorithm for median filtering, the most popular type of rank-order filtering. The bit-sliced algorithm [35, 36] bitwisely selects the median candidates and generates the median result one bit at a time. The bit-sliced algorithm recursively executes two steps: majority calculation and polarization. The majority calculation dominates the execution time of median filtering. Papers [10] and [17] present digital circuits for implementation of majority calculation. However, the circuits are time-consuming and complex so that they cannot take full advantage of bit-sliced algorithm. Paper [21] uses inverter as a voter for majority calculation. It significantly improves both cost and processing speed, but the noise margin will become narrow as the number of inputs increases. The narrow noise margin makes the implementation impractical and limits the configurability of rank-order filtering. Therefore, this paper presents a novel memory architecture for rank-order filtering based on a generic rank-order filtering algorithm. The algorithm uses threshold decomposition to bitwisely determine the rank-order result from MSB to LSB and applies polarization in between to polarize impossible candidates. Using the algorithm, we can pick the result according to the

[1] This work was supported in part by Taiwan MOE Program for Promoting Academic Excellent of Universities under the grant number 91-E-FA06-4-4 and the National Science Council, R.O.C., under the grant number NSC 92-2220-E-009-003.

The authors are with the Department of Electrical and Control Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan, R.O.C. (email: jswcheng.ece89g@nctu.edu.tw; lennon@cn.nctu.edu.tw).

Contributed Paper
Manuscript received April 15, 2004

given order without sorting the numbers. Note that the sorting is the most complex part in conventional rank-order filtering implementation. To implement the algorithm efficiently, the target architecture requires three fast tasks: parallel read, fast threshold decomposition, and parallel polarization. This paper presents a maskable memory structure, motivated from CAM architecture, to realize these tasks efficiently. The maskable memory structure, called pipelined maskable memory (PMM), is a regular SRAM structure with maskable registers. The maskable registers allow the architecture read bit-slices of inputs in parallel and perform parallel polarization with partial writes. In our design, the architecture is highly programmable and scalable. As the result, the architecture can execute rank-order filtering at high speed while the complexity is way lower than conventional implementations.

II. Generic Bit-slicing Rank-Order Filtering Algorithm

Let $W_i=\{x_{i-k}, x_{i-k+1}, ..., x_i, ...,x_{i+l}\}$ be a window of input sample and the binary code of each input x_j is denoted as $u_j^{B-1}\cdots u_j^1 u_j^0$. The output y_i of the r-th order filter is the r-th largest sample in the input window W_i, denoted as $v_i^{B-1}\cdots v_i^1 v_i^0$. The algorithm sequentially determines the r-th order value bit-by-bit starting from the most significant bits (MSBs). To start with, we first count 1s from MSBs of input samples and use Z_{B-1} to denote the result. If Z_{B-1} is larger than or equal to r, v_i^{B-1} is 1; otherwise, v_i^{B-1} is 0. All input samples whose MSB has the same value as v_i^{B-1} are considered as candidates of the r-th order sample, and thus the others can be polarized to either the largest or smallest value. If the MSB of an input sample x_j is 1 and v_i^{B-1} is 0, the rest bits (or lower bits) of x_j are set to 1s. Contrarily, if the MSB of an input sample x_j is 0 and v_i^{B-1} is 1, the rest bits (or lower bits) of x_j are set to 0s. After the polarization, we then count 1s from the second MSBs and repeat the polarization procedure. Consequently, the r-th order value can be obtained by recursively iterating the steps bit-by-bit. The bit-sliced generic rank-order filtering algorithm is as follows:

Given the input samples, the window size $N=l+k+1$, the bitwidth B and the rank r, do:

Step 1: Set $b=B-1$.

Step 2: (Bit counting)

Calculate Z_b from $\{u_{i-k}^b, u_{i-k+1}^b, \cdots, u_i^b, \cdots, u_{i+l}^b\}$.

Step 3: (Threshold decomposition)

If $Z_b \geq r$, v_i^b=1; otherwise v_i^b=0.

Step 4: (Polarization)

If $u_j^b \neq v_i^b$, $u_j^m = u_j^b$ for $0 \leq m \leq b-1$, for $i-k \leq j \leq i+l$.

Step 5: $b=b-1$.

Step 6: If $b \geq 0$ go to Step 2.

Step 7: Output y_i.

III. Pipelined Maskable Memory Architecture for Rank-Order Filtering

From Section II, the generic rank-order filtering algorithm generates the rank-order value bit-by-bit without using complex sorting computation. The main advantage of the algorithm is that the calculation of rank-order filtering has low computation complexity and can be mapped to a highly parallel algorithm. In the algorithm, there are three tasks which can be parallelized: bit counting, threshold decomposition, and polarization. To have these tasks highly parallelized, this paper presents a rank-order filter based on novel maskable memory architecture, as shown in Fig.1. With the instruction decoder and maskable memory, the proposed architecture is programmable and scalable.

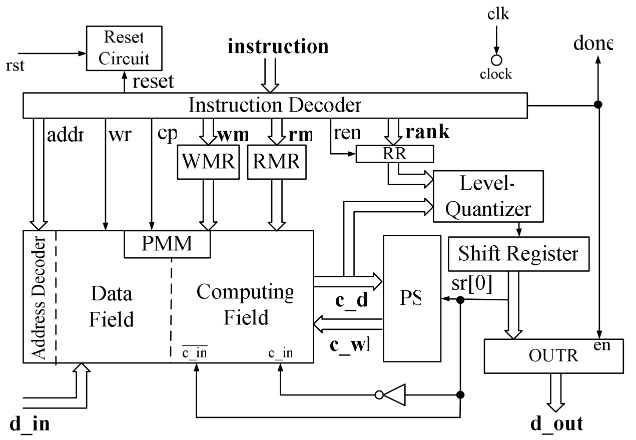

Fig. 1. The proposed rank-order filtering architecture.

The pipelined maskable memory (PMM) plays a key role in the rank-order filtering architecture. The PMM has two fields for reusing the input data and pipelining the filtering process. For one-dimensional (1-D) applications, the architecture receives an input sample and renews the input window at a time; for N-by-N two-dimensional (2-D) applications, the architecture receives N input samples and renews the input window in a filtering cycle. To speed up the process of rank-order filtering and pipeline the data loading and filtering calculation, the data field loads the input data while the computing field is performing bit-sliced operations. Hence, the execution of the architecture has two phases: data fetching and rank-order calculation. The two phases are executed in parallel for two consecutive iterations. In each iteration, the data fetching phase first loads the input sample(s) into data field and then makes copies from data field to computing field. Then, the rank-order calculation phase bitwisely access the computing field driven by three filtering tasks mentioned above.

The maskable part of PMM is the computing field where requires parallel reads for bit counting and parallel writes for polarization. The read-mask register (RMR) is configured to mask unwanted bits of the computing field during read operation, and the value of RMR is one-hot encoded so that the bit-sliced values can be read from the memory in parallel. The bit-sliced values will then go to Level-Quantizer for threshold decomposition. When the rank-order filter performs polarization, the write-mask register (WMR) is configured to mask untouched bits and allow polarization selector (PS) to polar lower bits of noncandidate samples. Since memory circuits have regular structure and the maskable scheme provides fast logic operations, the maskable memory structure features low cost and high speed, and, obviously, outperforms logic circuits on implementation of bit counting and polarization.

To start with the algorithm, the RMR is one-hot masked according to the value b in Step 1 and then the PMM outputs a bit-sliced value $\{u_{i-k}^b, u_{i-k+1}^b, \cdots, u_i^b, \cdots, u_{i+l}^b\}$ on "**c_d**". The bit-sliced value will send to both Level-Quantizer and PS. The Level-Quantizer performs the Step 2 and Step 3 by summing up bits of the bit-sliced value and comparing with the rank value stored in rank register (RR), respectively. The Fig.2 illustrates the block diagram of Level-Quantizer where FA denotes full-adder and HA denotes half-adder. "S" and "C" in FA or HA represent sum and carry respectively. The circuit in dash-lined part is a comparator implemented by carry generator, since the result of comparing Z_b and r is the carry output of adding r with the two's complement of Z_b, denoted as $(-Z_b)$. If the addition of r and $(-Z_b)$ is negative, $Z_b \geq r$ is true. Instead of using an adder, the carry generator is implemented by using majority (Maj) circuits. The Boolean equation of Maj is $Maj(A, B, C) = AB+BC+CA$.

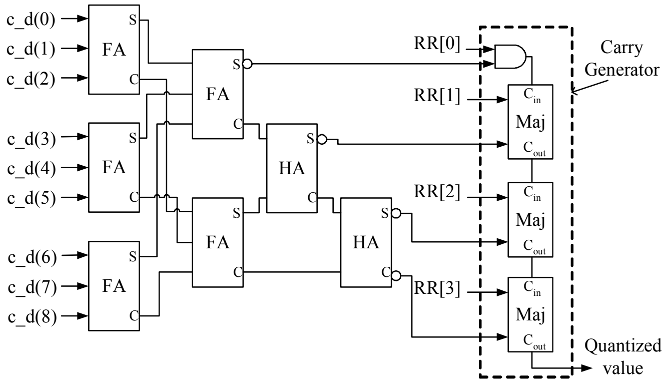

Fig.2. The block diagram of Level-Quantizer.

Each time the Level-Quantizer finish the threshold decomposition the shift register shifts a bit left and stores the result v_i^b in "sr[0]". Once the result v_i^b is stable at "sr[0]", one cycle after the PMM outputs the bit-sliced value, as shown in Fig.3, PS uses exclusive ORs (XORs) to determine polarization words and drive the wordlines "**c_wl**" of computing field accordingly. Thus, the noncandidate samples can be chosen by "**c_wl**" and their low bits can be polarized by configuring WMR.

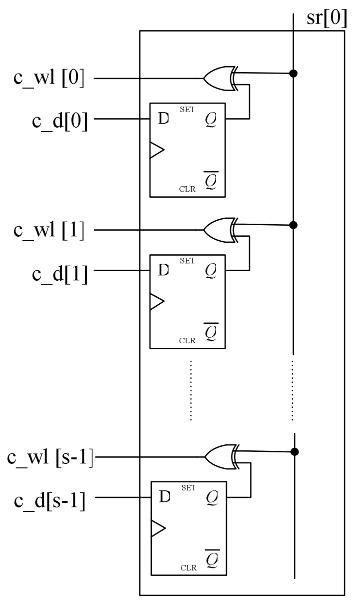

Fig.3: The polarization selector (PS).

IV. IMPLEMENTATION

Fig.4 illustrates a basic element of PMM. Each element has two cells for data field and computing field, respectively. The data cell is an SRAM with a pair of memory outputs where "INV1" and "INV2" store a bit of input sample addressed by the wordline "d_wl[i]". The computing cell performs three tasks: copy input sample from data cell, update data by "wm[j]" and "c_wl[i]", and read data by "rm[j]". When the copy line "cp" is high, the pair of "INV3" and "INV4" will have a copy of the 1-bit datum in data cell. The pair of "INV5" and "INV6" guarantees the unidirectional copy operation. When the one-bit value stored in computing cell is being polarized, the "wm[j]" and "c_wl[i]" will be asserted and thus the computing cell will update the value according to the pair of bitlines "*c_bl[j]*" and "$\overline{c\_bl[j]}$". When reading the bit-sliced value, the computing cell uses an NMOS gated by "rm[j]" to output the complement value to the dataline "$\overline{c\_d[i]}$". The datalines of computing cells in each word will be then merged as a single net, since the RMR is one-hot configured and only a bit is activated in a word. As shown in Fig.5, the dataline "$\overline{c\_d[i]}$" finally goes to an inverter and each bit-sliced bit is on "*c_d[i]*". Besides, the bitline pairs of computing cells are merged as a single pair of "*c_in*" and "$\overline{c\_in}$" because the low bits are set to either all 1s or all 0s when polarization.

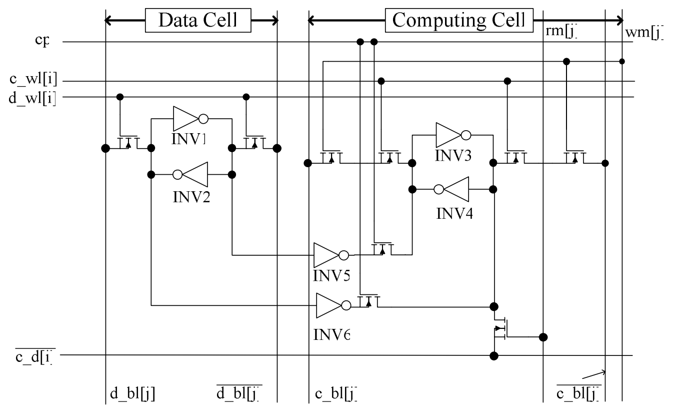

Fig.4. A basic element of PMM.

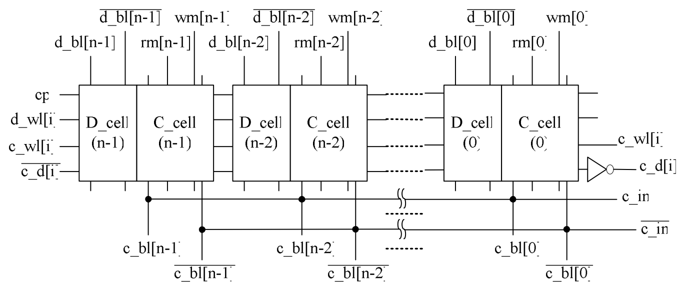

Fig.5. A PMM word mixing data field and computing field.

The PMM is implemented as shown in Fig.6. Each D_i-C_i pair is a maskable memory cell where D_i denotes D_cell(i) and C_i denotes C_cell(i). Each word is split into high and low parts for speeding up the memory access and reducing power dissipation. The control block is an interface between control signals and address decoder. It controls wordlines and bitlines of PMM. When the write signal "wr" is not asserted, the control block will have address decoder disassert all wordlines.

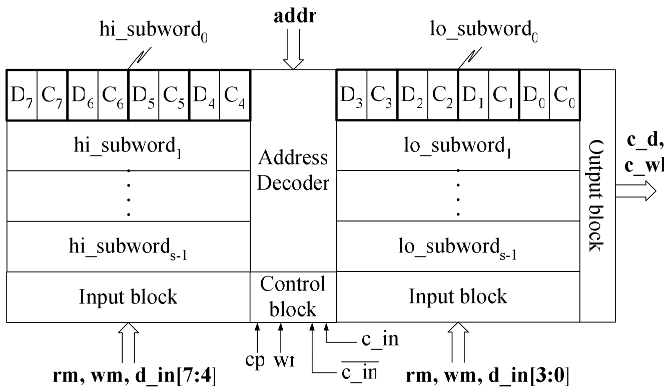

Fig.6. The PMM configuration.

The proposed rank-order filter is programmable and the instruction set is listed in Fig.7. All registers in the architecture are updated a cycle after instruction issued. The instruction SET resets all registers and set rank register RR for a given rank value. The instruction LOAD loads data from "**d_in**" by asserting "wr" and setting "addr". The instruction COPY/DONE has two modes: "copy only" and "copy and done". In the "copy only" mode, the PMM copies a window of input samples from data field to computing field. In the "copy and done" mode, the PMM not only copies data but also wrap up an iteration by asserting 'en' for OUTR; thus, the filter outputs the rank-order value after every iteration cycle. The instruction format concurrently issues two field instructions for pipelining data input and filter execution. Given a 1-D application with N=9 and rank=3, for instance, the instruction sequence is as follows: (The lines between "loop" and "loop end" are repeatedly executed until the end of filtering.)

```
SET 3;
LOAD 0000;
COPY ONLY, P_READ 10000000;
-- loop
P_WRITE 01111111;
P_READ 01000000;
P_WRITE 00111111;
P_READ 00100000;
P_WRITE 00011111;
P_READ 00010000;
P_WRITE 00001111;
P_READ 00001000;
P_WRITE 00000111;
P_READ 00000100;
P_WRITE 00000011;
P_READ 00000010;
P_WRITE 00000001;
P_READ 00000001;
LOAD xxxx, CF_NULL;
-- xxxx is the value of iteration count.
COPY/DONE, P_READ 10000000;
-- loop end
```

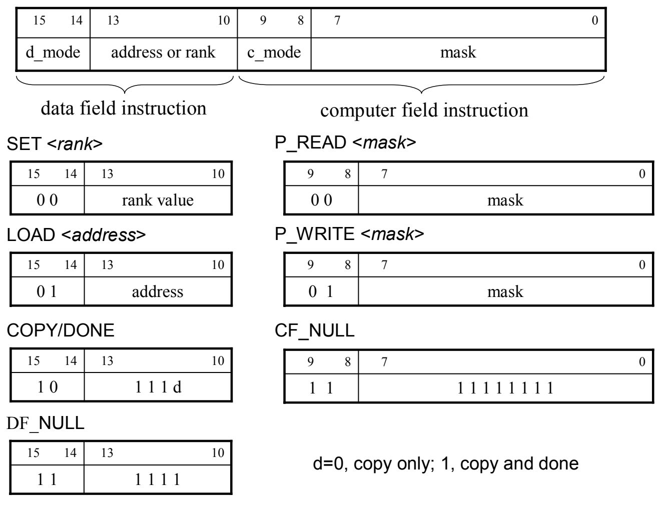

Fig.7. Instruction set format of proposed architecture.

V. Results

In this paper, we use TSMC 0.35um 1P4M process to implement the rank-order filter with N=9. The chip layout is shown in Fig.8 and its core size is 1405.4×1449.6 μm^2. After simulating a single cell with HSPICE, as shown in Fig.9, the architecture can safely run at 300 MHz. For 1-D applications, the sample rate can be 46.6 Mbytes/s at the maximum. Note that the sample rate is independent of the window size and the architecture is highly scalable with the change of window size.

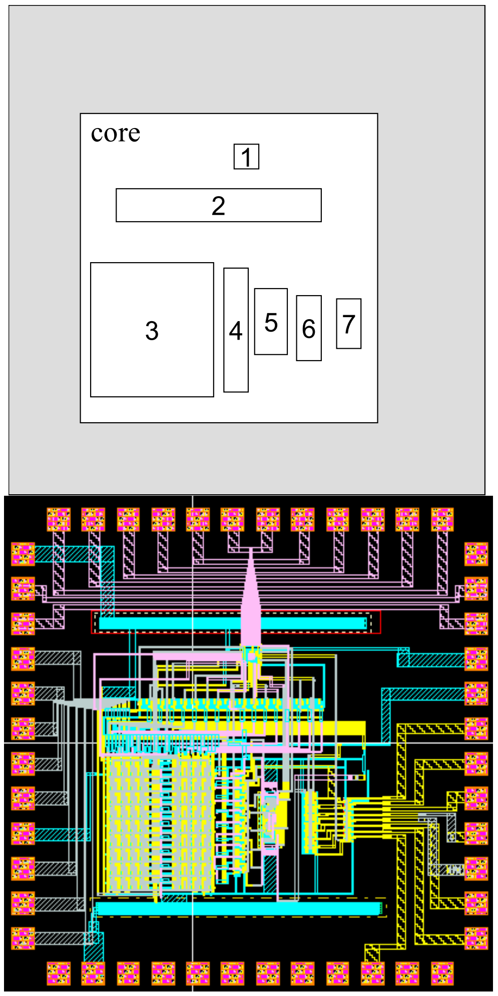

Fig.8. Chip layout of proposed rank-order filter. (1: Instruction decoder; 2: RMR, WMR and RR; 3: PMM; 4: PS; 5: Level Quantizer; 6: Shift Register; 7: OUTR.)

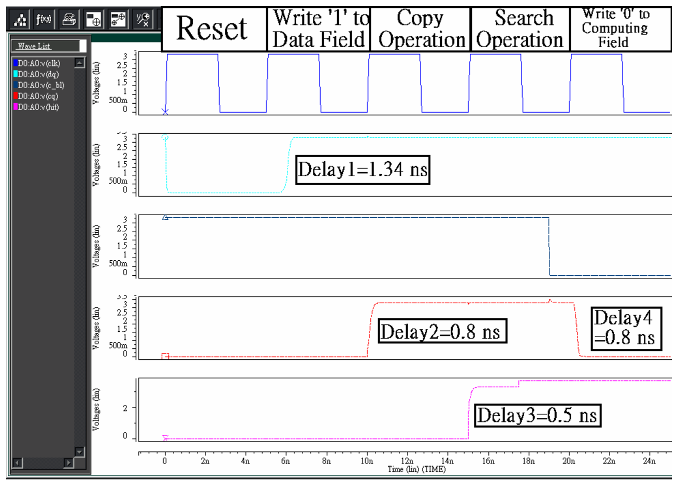

Fig.9. Simulation result of a single PMM cell.

To simulate with the 2-D application, where window size is 3-by-3 and rank order is 5, we construct the system as shown in Fig.10. Since each iteration updates three pixels, the data field loads three inputs every pixel cycle. Figure 11 is the simulation results of 2-D application. The follows are the program codes of the rank-order filter:

```
SET 5;
LOAD 0000;
LOAD 0011;
LOAD 0110;
COPY ONLY, P_READ 10000000;
-- iteration_cnt=0
-- loop
-- iteration_cnt=(iteration_cnt+1) mod 3
P_WRITE 01111111;
P_READ 01000000;
P_WRITE 00111111;
P_READ 00100000;
P_WRITE 00011111;
P_READ 00010000;
P_WRITE 00001111;
P_READ 00001000;
P_WRITE 00000111;
P_READ 00000100;
LOAD xxxx, P_WRITE 00000011;
-- xxxx is (the iteration count)
P_READ 00000010;
LOAD yyyy, P_WRITE 00000001;
-- yyyy is (the iteration count)+3
P_READ 00000001;
LOAD zzzz, CF_NULL;
-- yyyy is (the iteration count)+6
COPY/DONE, P_READ 10000000;
-- loop end
```

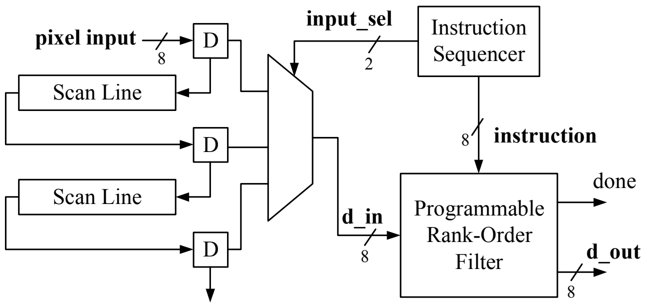

Fig.10. Block diagram of 2-D application with 3-by-3 window.

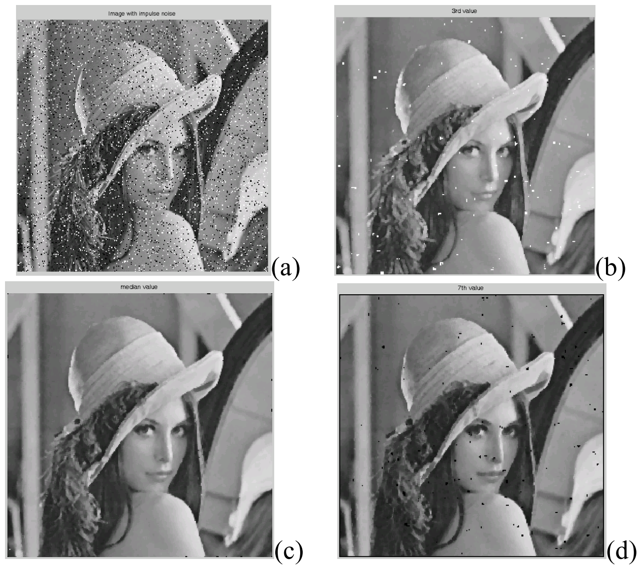

Fig.11. Simulation results of 2-D application: (a) image with impulsive noise (b) image with 3rd-order filtering and N=3 (c) image with 5th-order filtering and N=3 (d) image with 7th-order filtering.

VI. Conclusion

We proposed a novel architecture for rank-order filtering using a maskable memory circuit. The architecture has features of low cost, high degree of flexibility, and high speed. From Section V, the circuit can run at as fast as 300 MHz while the core size is small comparing with published VLSI rank-order filters.

References

[1] D.H. Kang, J.H. Choi, Y.H. Lee, and C. Lee, "Applications of a DPCM system with median predictors for image coding," IEEE Trans. Consumer Electronics, vol.38, no.3, pp.429-435, Aug. 1992.

[2] H. Rantanen, M. Karlsson, P. Pohjala, and S. Kalli, "Color video signal processing with median filters," IEEE Trans. Consumer Electron., vol.38, no.3, pp.157-161, Aug. 1992.

[3] T. Viero, K. Oistamo, and Y. Neuvo, "Three-dimensional median-related filters for color image sequence _ltering," IEEE Trans. Circuits Syst. Video Technol., vol.4, no.2, pp.129-142, Apr. 1994.

[4] X. Song, L. Yin, and Y. Neuvo, "Image sequence coding using adaptive weighted median prediction," Signal Processing VI, EUSIPCO-92, Brussels, pp.1307-1310, Aug. 1992.

[5] K. Oistamo and Y. Neuvo, "A motion insensitive method for scan rate conversion and cross error cancellation," IEEE Trans. Consumer Electron., vol.37, pp.296-302, Aug. 1991.

[6] P. Zamperoni, "Variation on the rank-order filtering theme for grey-tone and binary image enhancement," IEEE Int. Conf. Acoust., Speech, Signal Processing, pp.1401-1404, 1989.

[7] C.T. Chen and L.G. Chen, "A self-adjusting weighted median filter for removing impulse noise in images," Int. Conf. Image Processing, pp.16-19, Sept. 1996.

[8] D. Yang and C. Chen, "Data dependence analysis and bit-level systolic arrays of the median filter," IEEE Trans. Circuits and Systems for Video Technology, vol.8, no.8, pp.1015-1024, Dec. 1998.

[9] T. Ikenaga and T. Ogura, "CAM2: A highly-parallel two-dimensional cellular automation architecture," IEEE Trans. Computers, vol.47, no.7, pp.788-801, July 1998.

[10] L. Breveglieri and V. Piuri, "Digital median filter," Journal of VLSI signal Processing, vol.31, pp.191-206, 2002.

[11] C. Chakrabarti, "Sorting network based architectures for median filters," IEEE Trans. Circuites and Systems II: Analog and Digital Signal Processing, vol.40, pp.723-727, Nov. 1993.

[12] C. Chakrabarti, "High sample rate architectures for median _lters," IEEE Trans. Signal Processing, vol.42, no.3, pp.707-712, March 1994.

[13] L. Chang and J. Lin, "Bit-level systolic array for median filter," IEEE Trans. Signal Processing, vol.40, no.8, pp.2079-2083, Aug. 1992.

[14] C. Chen, L. Chen, and J. Hsiao, "VLSI implementation of a selective median filter," IEEE Trans. Consumer Electronics, vol.42, no.1, pp.33-42, Feb. 1996.

[15] M.R. Hakami, P.J.Warter, and C.C. Boncelet, Jr., "A new VLSI architecture suitable for multidimensional order statistic filtering," IEEE Trans. Signal Processing, vol.42, pp.991-993, April 1994.

[16] Hatirnaz, F.K. Gurkaynak, and Y. Leblebici, "A compact modular architecture for the realization of high-speed binary sorting engines based on rank ordering," IEEE Inter. Symp. Circuits and Syst., Geneva, Switzerland, pp.685-688, May 2000.

[17] A.A. Hiasat, M.M. Al-lbrahim, and K.M. Gharailbeh, "Design and implementation of a new efficient median _ltering algorithm," IEE Proc. Image Signal Processing, vol.146, no.5, pp.273-278, Oct. 1999.

[18] R.T. Hoctor and S.A. Kassam, "An algorithm and a pipelined architecture for order-statistic determination and L-filtering," IEEE Trans. Circuits and Systems, vol.36, no.3, pp.344-352, March 1989.

[19] M. Karaman, L. Onural, and A. Atalar, "Design and implementation of a general-purpose median filter unit in CMOS VLSI," IEEE Journal of Solid State Circuits, vol.25, no.2, pp.505-513, April 1990.

[20] C. Lee, P. Hsieh, and J. Tsai, "High-speed median filter designs using shiftable content-addressable memory," IEEE Trans. Circuits and Systems for Video Technology, vol.4, pp.544-549, Dec. 1994.

[21] C.L. Lee and C. Jen, "Bit-sliced median filter design based on majority gate," IEE Proc.-G Circuits, Devices and Systems, vol.139, no.1, pp.63-71, Feb. 1992.

[22] L.E. Lucke and K.K. Parchi, "Parallel processing architecture for rank order and stack filter," IEEE Trans. Signal Processing, vol.42, no.5, pp.1178-1189, May 1994.

[23] K. Oazer, "Design and implementation of a single-chip 1-D median filter," IEEE Trans. Acoust., Speech, Signal Processing, vol.ASSP-31, no.4, pp.1164-1168, Oct. 1983.

[24] D.S. Richards, "VLSI median filters," IEEE Trans. Acoust., speech, and Signal Processing, vol.38, pp.145-153, January 1990.

[25] C.G. Boncelet, Jr., "Recursive algorithms and VLSI implementations for median filtering," IEEE Int. Sym. on Circuits and Systems, pp.1745-1747, June 1988.

[26] C. Henning and T.G. Noll, "Architecture and implementation of a bitserial sorter for weighted median filtering," IEEE Custom Integrated Circuits Conference, pp.189-192, May 1998.

[27] C.C. Lin and C.J. Kuo, "Fast response 2-D rank order filter by using max-min sorting network," Int. Conf. Image Processing, pp.403-406, Sept. 1996.

[28] M. Karaman, L. Onural, and A. Atalar, "Design and implementation of a general purpose VLSI median filter unit and its application," IEEE Int. Conf. Acoustics, Speech, and Signal Processing, pp.2548-2551, May 1989.

[29] J. Hwang and J. Jong, "Systolic architecture for 2-D rank order filtering," Int. Conf. Application-Specific Array Processors, pp.90-99, Sept. 1990.

[30] I. Pitas, "Fast algorithms for running ordering and max/min calculation," IEEE Trans. Circuits and Systems, vol.36, no.6, pp.795-804, June 1989.

[31] O.Vainio, Y. Neuvo, and S.E. Butner, "A signal processor for median-based algorithms," IEEE Trans. Acoustics, Speech, and Signal Processing, vol.37, no.9, pp.1406-1414, Sept. 1989.

[32] H. Yu, J. Lee, and J. Cho, "A fast VLSI implementation of sorting algorithm for standard median filters," IEEE Int. ASIC/SOC Conference, pp.387-390, September 1999.

[33] J.P. Fitch, "Software and VLSI algorithms for generalized ranked order filtering," IEEE Trans. Circuits and Systems, vol.CAS-34, no.5, pp.553-559, May 1987.

[34] M. Karaman and L. Onural, "New radix-2-based algorithm for fast median filtering," Electron. Lett., vol.25, pp.723-724, May 1989.

[35] J.P. Fitch, E.J. Coyle, and N.C. Gallagher, Jr., "Threshold decomposition of multidimensional ranked order operations," IEEE Trans. Circuits and Syst., vol.CAS-32, no.5, pp.445-450, May 1985.

[36] B.K. Kar and D.K. Pradhan, "A new algorithm for order statistic and sorting," IEEE Trans. Signal Processing, vol.41, no.8, pp.2688-2694, Aug. 1993.

Lan-Rong Dung received a BSEE and the Best Student Award from Feng Chia University, Taiwan, in 1988, an MS in electronics engineering from National Chiao Tung University, Taiwan, in 1990, and Ph.D. in electrical and computer engineering from Georgia Institute of Technology, in 1997.

From 1997 to 1999 he was with Rockwell Science Center, Thousand Oaks, CA, as a Member of the Technical Staff. He joined the faculty of National Chiao Tung University, Taiwan in 1999 where he is currently an assistant professor in the Department of Electrical and Control Engineering. He received the VHDL International Outstanding Dissertation Award celebrating in Washington DC in October, 1997. His current research interests include VLSI design, digital signal processing, hardware-software codesign, and System-on-Chip architecture. He is a member of Computer and Signal Processing societies of the IEEE.

Meng-Chun Lin received the B.S. degree in Electronic Engineering from Fu Jen Catholic University, Taipei, Taiwan, R.O.C. in 2001, and the M.S. degree in the Electrical and Control Engineering, National Chiao Tung University Hsinchu, Taiwan, R.O.C. in 2003. He is currently working toward the Ph.D degree in the Electrical and Control Engineering, National Chiao Tung University. His research interests are image processing, memory circuits design, VLSI architecture and digital signal processing.