High-Speed Median Filter Designs Using Shiftable Content-Addressable Memory

Chen-Yi Lee, Po-Wen Hsieh, and Jer-Min Tsai

Abstract— **This paper presents a very efficient VLSI architecture for real-time median filtering as requested in many image/video applications. The median is obtained by first sorting input sequences and then selecting identified order according to the number of inputs. To reach the goal of high-speed data sorting, an optimized delete-and-insert algorithm is derived and then mapped onto shiftable content-addressable memory architecture. The complete design can be decomposed into a set of processor elements, where each processor element consists of two basic cells— *sort-cell and compare-cell.* Thus the design becomes very regular. More specifically any specified order can be obtained within one cycle and a high-speed clock rate can be achieved. A proto-type chip for 64 samples based on this architecture has been implemented and tested. Results show that a clock rate up to 50 MHz can be achieved using a 1.2 μm CMOS double metal technology.**

I. Introduction

IMAGE median filters are well known for being able to remove impulse noise [1], to preserve image edges, and recently to improve coding efficiency [2] as well as to enhance color signal processing [3]. Many algorithmic approaches with hardware implementation can be found in the literature [1], [4], [5], [6] and they show that the applicability of this median filtering technique is very application-dependent. Moreover different windows and pixel locations may be required to achieve specified behavior for different target applications. Although analog implementation of median filters has been discussed recently [7], its output rate is not high enough to handle video-rate applications and hence applications are very limited. Thus only the digital approaches for median filtering are discussed in this paper.

A good survey paper of VLSI median filters can be found in [10], where the author discussed hardware complexity in terms of *number of samples 'N', word length 'l', and running size 'R'*. In principle, these digital algorithms and methods can be classified into two categories: word-level and bit-level as discussed in [6]. In this paper, only the word-level median filters are studied since they offer high throughput capability as required in many real-time image/video systems. However a very cost-effective hardware solution to meet this goal is often difficult to achieve and hence system performance becomes degraded to allow trade-off between hardware cost and achievable performance. For example, a fast median filter based on the bubble sorting algorithm can be found in [5]. By means of a set of processing elements or PEs, the required values can be obtained with a latency of N cycles, where N is the number of input samples. Though this approach is fast, the size of the hardware implementation complexity, is proportional to the square[1] of the number of input samples. Hence hardware overheads increase rapidly with the number of input samples. In addition to this sorting kernel, it is necessary to provide extra hardware in the form of a data buffer to rearrange input samples for the parallel processing and hence increase the memory bandwidth. Another solution is a message passing method [8] realized on a systolic array architecture [9]. Both deletion and insertion messages pass through the systolic arrays until certain conditions are encountered. Although the hardware complexity depends on the number of input samples (N), the latency remains the same as that needed in the parallel bubble sorter. This latency of N cycles may not be allowed when real-time performance is concerned.

In this paper, we present a more cost-effective hardware solution which can be integrated with other hardware without degrading overall system performance. This is achieved by reducing the latency of median search on a set of input samples so that the median can be obtained immediately without causing a stall on the data flow. For example, the median can be obtained right after the last sample is presented and then immediately passed to the next stage. In Section II, an algorithm suitable for such an implementation is first introduced and then a transform method is discussed to see how it can be applied to the selected algorithm for performance improvement. Section III presents a shiftable content-addressable memory (SCAM) VLSI architecture which is a processor element (PE) based structure. Each PE contains two different cells— one (sort-cell) stores sorted items and the other (compare-cell) compares current the input sample with the sorted items so that the input sample can be placed appropriately to reach high speed sorting. A proto-type chip based on this design approach is given in Section IV to evaluate design efficiency. Also some comparisons with available approaches are included to highlight the performance of the SCAM VLSI architecture in sorter designs.

Manuscript received January 8, 1993; revised July 24, 1993, and February 23, 1994. This paper was recommended by Peter Pirsch and the work supported by the National Science Council of Taiwan, ROC under grant NSC82-0404-E009-184.

The authors are with Dept. of Electronics Eng. and Institute of Electronics, National Chiao Tung University, 1001, University Road, Hsinchu 300, Taiwan, ROC.

IEEE Log Number 9406761.

[1] More accurately, the number of PEs is $(N-1) \times N/2$ for odd numbers of input samples and $N^2/2$ for even numbers.

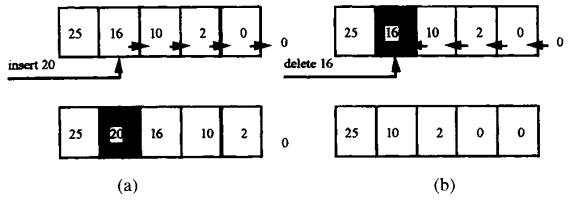


Fig. 1. Illustration of the Insert and Delete sorting operations.

II. Algorithm Description And Transformation

Since median search can be decomposed into two stages— (1) sorting input samples and (2) selecting specified order from the sorted sequence, and the former is much more complex than the latter, throughput can only be improved when complexity of the sorting stage is overcome. We select the delete-and-insert algorithm [8] due to its low memory bandwidth requirement.

The Delete-and-Insert Sort Algorithm: The delete-and-insert sort algorithm is one of the many available sort algorithms. Its operations can be described as follows. For data insertion as shown in Fig. 1(a), the current input sample is known to be 20 and should be placed in between 25 and 16 which are already sorted and stored in a memory. Each time a new sample is given, it can be placed according to this scheme. After all samples are processed, the sorted sequence can be obtained from the memory. For data deletion, it only needs to identify the sorted item whose value is equal to the input sample's value as shown in Fig. 1(b). Thus for any running order operation, this delete operation becomes useful since only those samples out of the mask region need to be removed and new input samples need to be sorted. In addition to these features, this algorithm can also provide an ascending sequence if smaller items are placed at left side instead of right side as used for a descending sequence (as illustrated in Fig. 1).

This algorithm description implies that a lot of comparisons are needed in order to allocate a position in which the input sample can be correctly placed or removed. Moreover a lot of move operations are needed before the input sample is inserted or deleted. For real-time image/video processing, either insert or delete operation has to be done in a very stringent timing constraint. Thus if this algorithm is implemented on a general-purpose computer, the result is obviously not suitable for real-time applications and hence the algorithm has to be modified. A modified version, called optimized delete-and-insert or ODI sort algorithm is thus developed to fully explore parallelism so that a very high throughput requirement can be obtained by exploiting VLSI advantages such as speed and density.

The ODI Sort Algorithm: As described above, the bottleneck of enhancing the delete-and-insert sorter's throughput lies in two factors— *(1) identification of insert/delete target and (2) data movement among sorted items*. The complexity of these two factors becomes higher as the number of input samples increases. However if we explore fully parallelism inherent in the algorithm, then this bottleneck can be overcome. To see how the parallelism can be explored, we summarize the algorithm description as below:

```
/* for insertion */
Sort[] = 0; /* initialize Sorted array Sort[] */
Sort[0] = int_max; /* set boundary condition */
DataIn = input sample;
SampleCount++;
for (i:= SampleCount; i > 0; i–) {
    if (DataIn >= Sort[i]) {
        Sort[i+1] := Sort[i]; /* shift right */
    }
    if (DataIn >= Sort[i]) & & (DataIn < Sort[i-1])
{
        Sort[i] := DataIn; /* insert data */
    }
}

/* for deletion */
DataIn = input sample; /* input sample to be
deleted */
for (i:=1; i < = SampleCount; i++) {
    if (DataIn >= Sort[i]) {
        Sort[i] := Sort[i+1]; /* shift left, i.e., delete
data */
    }
}
SampleCount–;
```

Here it is found that for each input sample, N comparisons are needed in order to find the position where the input sample should be placed or removed. This parallel-comparison process can be realized on a content addressable memory (CAM) to identify the target to be inserted or deleted. Once the target is identified, the next step is to perform data movement on a part or all the sorted items. For example, Fig. 1(a) illustrates the insertion of data item whose value is 20. For those sorted items whose value is less than or equal to the input item, they have to move one position right; while in data deletion, those sorted items whose value is less than or equal to the input item have to move one position left as shown in Fig. 1(b). In other words, the sorted items are divided into two groups— one is the LE group (i.e. sorted items less than or equal to input sample) and the other is the GT group (i.e. sorted items greater than input sample). Thus the data movement can be replaced by shift operations working on the LE group instead of read/write operations on memory and in particular, these shift operations can be executed in parallel. Thus this modified ODI algorithm can operate on every sorted item simultaneously and generate any request order very fast if both concepts of content-addressable memory and shift registers are used.

III. Architecture Issues

In the previous section, we discussed how the ODI algorithm solves the bottleneck of data sorting and then achieves high throughput by means of the support of content-addressable memory and shift registers. In this section, we discuss in more detail how the ODI algorithm is mapped onto the shiftable content-addressable memory (SCAM) architecture which is very suitable for VLSI implementation.

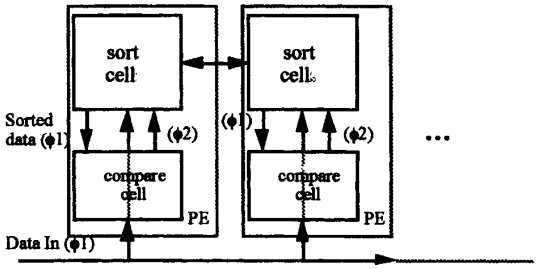


Fig. 2. The shiftable content-addressable memory is realized on a PE-based structure.

The PE-Based Structure: Suppose that there are N samples to be sorted, we would like to obtain any specified order right after the last sample is given. This is, in fact, the specification of our high-speed median filter design. We also assume that the sorted sequence is in a descending order. Thus N storage spaces are needed to store the sorted samples. Initially all contents of SCAM are reset to zero. For each input sample, the content of each storage space is read out and compared with the input sample in order to identify the position where the input sample should be placed. Once the position is identified, the content of each storage space has to be conditionally shifted according to the newly updated sequence. This implies that the architecture consists of N processor elements (PEs) and each of which contains two basic cells— (1) sort-cell and (2) compare-cell (as shown in Fig. 2). The former is a shift register containing the sorted data and can be shifted right or left while the latter intends to provide control signals orchestrating operations of the former.

Detailed Architecture and Circuit Design for the Sort Cell: From the previous discussions, it can be found that the sort-cell should have the following functionalities (see Fig. 3(a)):

1) data can be shifted right,
2) data can be shifted left,
3) data can be loaded,
4) data remains unchanged,
5) data can be reset.

Here items 1, 3, and 4 are required in insert operations, while items 2 and 4 are needed in delete operations. Item 5 is only used when a new input set is to be processed. With these pre-defined functionalities, the corresponding circuit for such a 1-bit cell can be easily derived as shown in Fig. 3(b). Only 3 inverters and 6 pass transistors are needed for each bit cell. The first inverter acts as an internal buffer for data pre-shifting to enhance the clock rate (see the last paragraph of this section for more details). Note that a two-phase non-overlapping clocking strategy is used here.

Detailed Architecture and Circuit Design for the Compare Cell: This cell intends to generate all required control signals for those used in the sort cell. The functional description of this cell is given in Fig. 4(a). Here the shc control signal is defined for insertion or deletion mode selection. The other control signals are generated respectively according to the following conditions:

- shift data right (shr): This occurs when data insertion is under execution. However only those sorted items whose value is less than or equal to that of the current input sample (or belongs to the LE group as defined above) will be shifted right and hence the carry (C_i in ith compare-cell) is demanded. Thus the shr is activated whenever shc is low (for insertion) and C_i is high.
- shift data left (shl): This shl is activated when data deletion is performed (i.e. shc is high) and C_i is high.
- load data ($load$): Since only one PE requires this signal at a time, i.e. the input sample can only be placed at one correct position, the generation of $load$ signal becomes more complex than the previous two control signals. Not only shc and C_i are considered, but also the carry from the previous stage (C_{i-1}) should be taken into account. This is obvious since the correct storage space for the current input sample is between those items less than or equal to the input sample (i.e. C_{i-1}=0, or LE-group members) and those items greater than the input sample (i.e. C_i=1, or GT-group members).

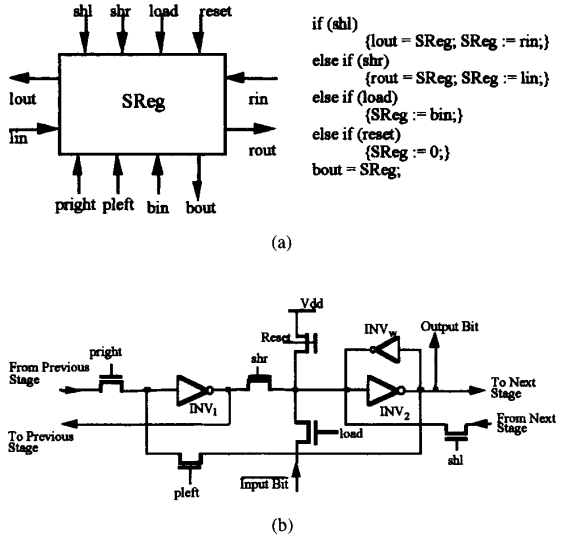


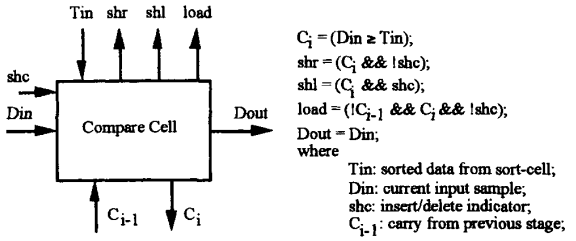
Fig. 3. (a) Behavioral description and (b) circuit diagram of each 1-bt sort-cell. Note that a weak inverter is placed at INV_2 to overcome leakage paths.

To speed up the carry generation, a carry-look-ahead technique [11] is exploited. The overall circuit diagram for this compare-cell is shown in Fig. 4(b).

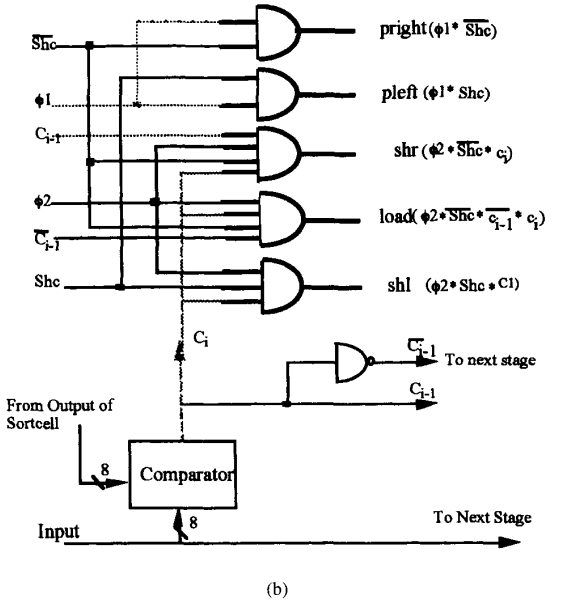
Operation and Clocking Strategy: To speed up the clock rate, a 2-phase non-overlapping clock is exploited here. All the control signals are generated at $\phi 1$ and data shift operations are performed at $\phi 2$. In addition, a *pre-shift* strategy is also exploited to improve clock speed. This strategy shifts sorted items to the buffer (the first inverter INV_1 of the sort-cell) of next PE during $\phi 1$ and then are conditionally stored at $\phi 2$. The detailed operation for data insertion on the sort-cell is illustrated in Fig. 5. Note that the first inverter of each sort cell acts as a buffer during pre-shifting as needed for both shift right and left operations.

IV. Evaluation And Discussions

To evaluate this SCAM architecture, a proto-type chip for a maximum of 64 input samples has been designed. The figure



(a)



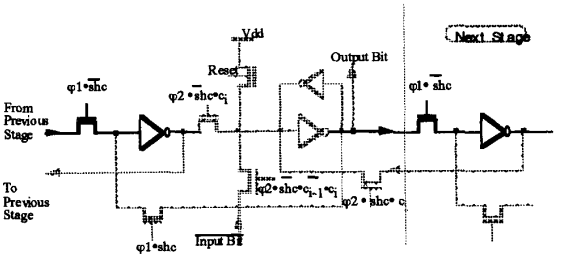
(b)

Fig. 4. (a) Behavioral description and (b) circuit diagram of the compare-cell. The carry is generated by the carry-look-ahead technique to enhance clock speed.

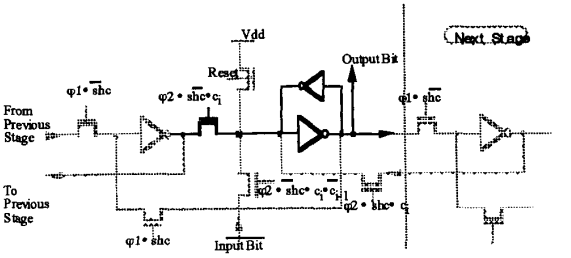
of 64 is considered here because we would like to make a chip for median filtering of variant sizes as well as cascadable data sorting. In this section, we first present some experimental results about this ODI chip and then provide some comparisons with the two approaches described in the Introduction. Also some applications where this ODI chip can be exploited are presented.

Design of the ODI Chip: Block diagram of this chip is given in Fig. 6. This chip can be used either as a high-speed data sorter or as an image median filter. Input samples are first sorted through the shiftable content-addressable memory. Then the specified order, such as median, can be identified from the sorted items by means of a dynamic selection circuit. For raster scan images, this chip provides an optimal solution since the required order can be obtained right after the last sample is given. It should be noted that using the pre-shift strategy, both phases ($\phi 1$ and $\phi 2$) are quite balanced. Experimental results show that a clock rate up to 50 MHz can be achieved. The micro-photo of this chip design is shown in Fig. 7. Some key features of this ODI chip are given below:

- Each chip can handle 64 samples and can be cascaded for more samples when data sorting is considered.



(a)



(b)

Fig. 5. Insert data flow on the 1-bit sort-cell. (a) pre-shift stage at $\phi 1$ and (b) data shift at $\phi 2$. Note the dark lines indicate how the circuit works at different phase.

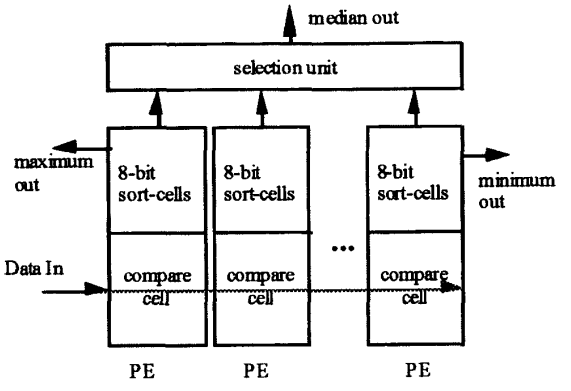


Fig. 6. Block diagram of the ODI chip.

- Any specified order can be obtained right after the last sample is given, i.e. without latency. This is very useful for pipelined architecture design.
- Running order can be handled by exploiting both the delete and insert operations.
- Design is very regular and hardware complexity is linearly proportional to the number of input samples.

This chip has been fabricated through the MPC services supported by Chip Implementation Center (CIC). Test results show that a maximum clock rate can be up to 50 MHz. Some characteristics of this chip are given below:

- Die size: 5053 μm × 4774 μm;
- Pin-count: 67;
- Transistor-count: N-type: 13060, P-type: 10141, Totally: 23201;
- Clock rate: 50 MHz;

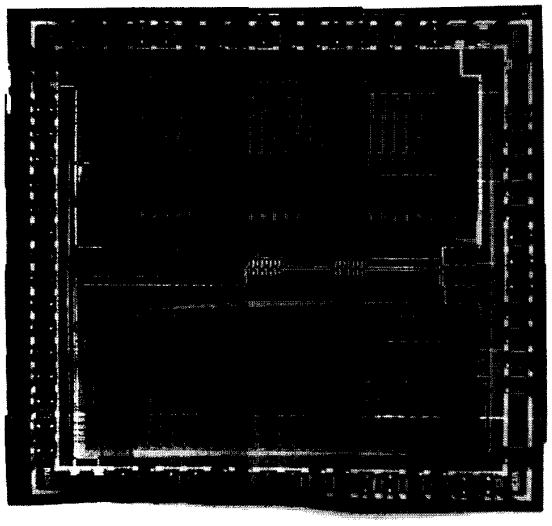


Fig. 7. Microphoto of the ODI chip.

- Power consumption: 0.45W@50 MHz;
- Technology: 1.2 μm CMOS double-metal technology.

Comparisons with Other Approaches: We only compare the results based on the bit-parallel approaches as given in the Introduction since they are more practical for real-time image/video applications.

The parallel bubble sorter as described in [5] requires a lot of hardware as the number of input samples increases. In addition, its input format has to be adjusted so that a set of input samples can be fed simultaneously to the sorter arrays. This implies that a large memory bandwidth is needed. In addition, given a set of N input samples, the required order can only be obtained right after N cycles. Obviously, our ODI chip does outperform this bubble sorter in terms of area and latency.

The ROS sorter from [8] based on a systolic architecture offers similar hardware complexity compared to our design. However our design offers higher throughput and expansibility.

Table I shows some comparison data from these three approaches. In addition to the specific features of less area and no data latency, our design also provides a testable strategy since test patterns can easily be applied to the SCAM and detected from the selection circuit or from both shift output ports.

Some Typical Applications: With a clock speed of 50 MHz, this ODI chip can handle median filtering for current video rates in real-time. It can also be exploited in advanced television receivers for noise reduction and scan rate up-conversion [3]. In addition, we can use this chip for recursive median filtering to improve the efficiency of noise-reduction, where several mask sizes are demanded. For those running window applications, line-delays can be added on-chip or off-chip. However, it should be noted that only one sample can be handled by this chip at a time.

Based on the SCAM architecture and the experience from the ODI chip design, we have also found a lot of applications

TABLE I
SOME COMPARISONS OF OUR ODI CHIP WITH OTHER BIT-PARALLEL APPROACHES

item \ sorter	parallel bubble sorter [8]	ROS sorter[5]	ODI sorter
hardweare complexity	αN^2	αN	αN
latency (L)	αN	αN	$0 \le L \le 1$
clock rate	high	high	high
expansibility degree	difficult	easy	easy

Note: (1) α stands for *proportional to*
(2) N: number of input samples to be sorted.

where this SCAM architecture can be exploited. In such cases, the ODI chip can be regarded as a block fitting into the overall system organization. For example, in adaptive entropy coding, the SCAM architecture provides a very cost-effective solution in sorting the occurrences of input samples within the specified history to enhance compression ratio. Depending on the number of input samples to be handled, the SCAM hardware can easily be constructed based on the PE-based regular structure, and hence the design cycle can be reduced.

V. CONCLUSION

In this paper, we have presented a high-throughput median filter design based on the ODI algorithm and the SCAM architecture. This chip is obtained by first exploring the inherent parallelism and then exploiting shift operations to achieve high speed sorting so that any order of input samples can be obtained. Using the SCAM architecture, we have developed a very cost-effective hardware solution for median search as well as for data sorting. Test results from prototype chips show that the ODI chip does outperform available approaches for median search in terms of both area and throughput. Also this area-efficient solution can be integrated with other hardware when median search is demanded in other system development. We are currently looking into the applicability of exploiting this high-speed sorting architecture for adaptive coding which is often used in entropy coding to enhance compression ratio.

ACKNOWLEDGMENT

The authors would like to thank their colleagues within the VLSI/CAD group of NCTU for many fruitful discussions. Also the MPC support from Chip Implementation Center (CIC) of NSC is acknowledged. In particular, the authors express their gratitude to the anonymous reviewers for their help in making this paper more readable.

REFERENCES

[1] A. K. Jain, *"Fundamentals of Digital Image Processing"*, Prentice-Hall, 1989.
[2] 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. on Consumer Electronics*, vol. 38, no. 3, pp. 429–435, Aug. 1992.

[3] H. Rantanen, M. Karlsson, P. Pohjala, and S. Kalli, *"Color Video Signal Processing with Median Filters"*, *IEEE Trans. on Consumer Electronics*, vol. 38, no. 3, pp. 157–161, Aug. 1992.
[4] H. M. Lin and A. N. Jr. Willson, *"Median Filters with Adaptive Length"*, *IEEE Trans. on CAS*, vol. 35, no. 6, pp. 675–690, Jun. 1988.
[5] J. Offen and R. Raymond, *"VLSI Image Processing"*, McGraw-Hill, 1985.
[6] C. L. Lee and C.W. Jen, *"Bit-sliced Median Filter Design Based on a Majority Gate"*, IEE Proc-G V139, no. 1, pp. 63–71, Feb. 1992.
[7] P. H. Dietz and L. R. Carley, *"An Analog Circuit Technique for Finding the Median"*, in Proc. of Custom Integrated Circuits Conference, San Diego, May 9–12, 1993.
[8] A. L. Fisher, *"Systolic Algorithms for Running Order Statistics"*, in Signal and Image Processing, Dept. of Computer Science, Carnegie Mellon University, Pittsburgh, Jul. 1981.
[9] H. T. Kung, *"Why Systolic architectures"*, IEEE Computer, vol. 15, no. 1, Jan. 1982.
[10] D. S. Richards, *"VLSI Median Filters"*, *IEEE Trans. on Acoustics, Speech, and Signal Proc.*, vol. 38, no. 1, Jan. 1990.
[11] N. Weste and K. Eshraghian, *"Principles of CMOS VLSI Design-- A Systems Perspective"*, Addison-Wesley, 1985, pp. 320–335.

Chen-Yi Lee received the B.S. degree in electronics engineernig from National Chiao Tung University in 1982 and the M.S.and Ph.D. degrees in electrical engineering from Katholieke University Leuven (KUL), Belgium in 1986 and 1990.

From 1986 to 1990, he was with IMEC/VSDM division, working in the area of architecture synthesis for DSP. He joined the faculty at National Chiao Tung University in February, 1991, where he is currently an Associate Professor of Electronics Engineering. His research interests mainly include VLSI architectures, visual communications and related VLSI designs, and CAD for VLSI.

Po-Wen Hsieh was born in Kaoshiung City, Taiwan, on November 10, 1967. He received the B.S. and M.S. degrees from National Chiao University, Hsinchu, Taiwan in 1991 and 1993, both in electronics engineering. His research interests include image processing, visual communications, VLSI architectures, and high-speed circuit design.

Jer-Min Tsai received the B.S. degree from National Cheng Kung University, Tainan, Taiwan, in 1989, and the M.S. degree from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 1993, both in electrical engineering. He is currently working toward the Ph.D. degree at NCTU. His research interests include VLSI architectures, high-speed networking, and related VLSI designs.