Parallel and Pipelined 2-D Median Filter Architecture

Janarthanam Subramaniam [ORCID], Raju Jagadeesh Kannan, and David Ebenezer, *Senior Member, IEEE*

Abstract—The existing 2-D median filters in the literature are computationally intensive. It is proposed to optimally reduce the amount of data handled at the architecture level realization of the basic median filtering operation on images. The proposed architecture reads 4 pixels at a time in the input image, 4 pixels forming a word on a 32-bit hardware processing system; the subsequent processing is carried out by parallel and pipelined median filter architecture. Two read operations process eight input pixels which results in the generation of four output pixels with an initial latency. The proposed architecture offers reduced number of read operations and increased speed.

Index Terms—Median filter, parallel median filter, pipelined median filter, systolic arrays.

I. INTRODUCTION

IMAGE processing is integral to deciphering the intelligence associated with it. Generally, image processing involves huge data handling, and modeling human intelligence demands heavy amount of computations on automated processes. Handling images involves loss of original information at various levels through environment and process stages. One such level is the processing of image for feature extraction [1]. Prior to this, preprocessing an image for the retrieval of the original information from non-Gaussian noise corruption is carried out mostly by nonlinear digital filters. Dominant among these are median-based filters.

Median filter provides robustness to impulse noise; however, the development of median filtering algorithms does not include the requirements of real-time intelligence systems. Generic processing ICs do not provide cost effective solution for image processing because of predefined architectural

limitations [2]. In addition, there is always a tradeoff between the quality of information contained in images and the resources required to handle the images. FPGAs are sufficiently flexible and cost effective for prototyping and reconfiguring the applications [3] and, therefore, provide sufficient opportunity for the development of application specific architectures which cater to real-time requirements. Basic median filter can be implemented on specific architectures by performing median operation through sorting-based systolic arrays [4] and nonsorting-based techniques [5]–[8]. Systolic arrays are continuously optimized by researchers [9]–[11]. The architecture developed by Vega-Rodríguez *et al.* [4] exploits 32-bit data-width hardware for image data transfer; they have implemented the architecture as a systolic array which is reported in [10].

The architecture proposed in this letter handles image data effectively in such a way that a pixel in the input image is read only twice in comparison with architectures reported in the literature, wherein a pixel is read three or more times. Reduction in the number of read operations for filtering an image ensures reduction in the overall operating time. The proposed architecture also offers the advantage that it can be employed for different word length realizations, programmable digital signal processing, ASICs, and reconfigurable architectures. In addition, the proposed architecture does not impose any constraint on the time required for reading and processing pixels.

II. MEDIAN FILTER MEETING REAL-TIME REQUIREMENTS

Vega-Rodríguez *et al.* [4] have presented an architecture for basic median filter for systolic array implementation [10]. The architecture employs pipelining and parallelism. It is finally implemented with FPGA as target device. The FPGA is interfaced with computer through 32 bit PCI port for real-time interfacing and better human interactions. Every read instruction on a 32 bit system can read 4 pixels, each 8 bit wide. Multiple pixels on a single read cycle and parallelism on systolic array lead to simultaneous production of four filtered pixels. The parallel and pipelined median filter architecture is shown in Fig. 1. Smith's network [10] introduced parallelism and pipelining by splitting the nine level systolic arrays into two stages. The first stage in Fig. 1 is called elementary sorting stage also known as E-stage. The next stage is called network sorting stage also known as N-stage, with six levels of
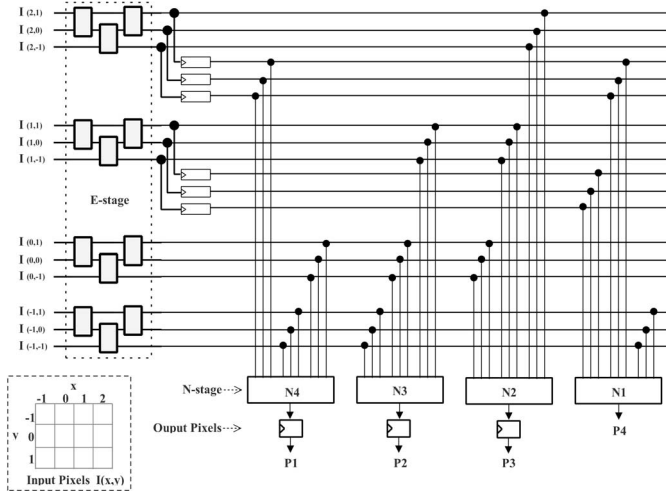
Fig. 1. Real-time median filter proposed by Vega-Rodríguez *et al.* [4].
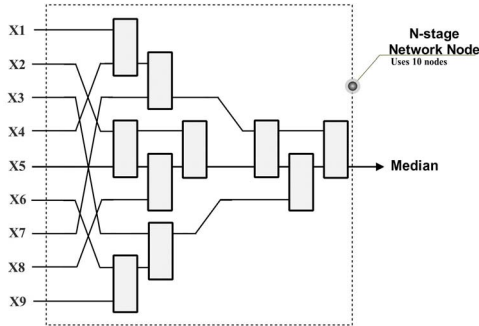
Fig. 2. Part of systolic array optimized by Smith [10].

Fig. 3. Proposed two stage pipelining scheme.

comparators as shown in Fig. 2. It is also denoted as network node in the figure.

The output pixel value is calculated by considering eight neighbors with respect to a center pixel and 3×3 moving mask. In order to read 4 pixels of the input image in a machine cycle, the first three rows are considered at the start. With respect to these three rows, the first four columns capture three row vectors, each vector containing 4 pixels. Each row vector containing a set of four 8 bit pixels forms a 32 bit word. Before the start of the processing, three words are available for transfer to the architecture. Three consecutive read operations result in the transfer of a 3×4 matrix to the architecture. The last two columns of every 3×4 matrix processed previously are sorted and stored. These stored elements are combined with incoming 3×4 matrix for the formation of a 3×6 matrix. Consequently, this provides for placing four 3×3 masks simultaneously with respect to four center pixels.

In the architecture, the arrangement of four parallel network nodes results in four filtered pixels at a time. A set of three read operations, one cycle of filtering operation, and one cycle of write operation is the sequence of operations required for completing one cycle. The parallelism uses only 52 comparators for producing four filtered output pixels. This obviates the need for 76 comparator operations as employed in the approaches reported in the literature.
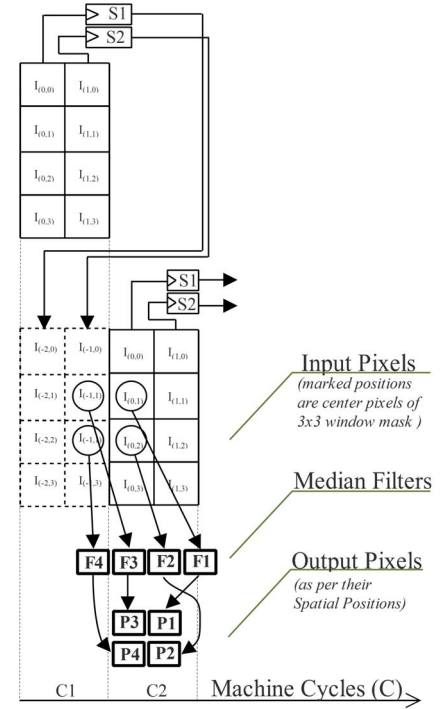
III. ARCHITECTURE WITH REDUCED DATA HANDLING

A fast median filtering scheme and a new architecture for hardware implementation on 32 bit system are proposed. The proposed scheme is common to both reconfigurable architectures and general purpose computing platforms.

A. Median Filtering Scheme for Effective Data Handling and Parallel Filtering

The proposed scheme is shown in Fig. 3. The objective is to effectively handle pixels for reducing repeated read operations. Beginning with the input image, the first four rows of the pixels are selected; the pixels in the selected rows are grouped as column vectors of 4 pixels each. Each column vector is read as one 32 bit data word. Four column vectors are presented as a single 4×4 matrix, where 3×3 masks can be applied with respect to four center pixels. Only two words are transferred instead of three or more words for completing one processing cycle. Only a single processing cycle is required for generating four filtered outputs; these four outputs are generated by four filters, namely, F1–F4. The two words can be transferred in one or two machine cycles depending on the capability of system. For simplicity, reading two words per machine cycle is considered. The first two column vectors are transferred in cycle 1 and the words are stored in elements S1 and S2, each of size 4×1. In the next machine cycle, the third and fourth column vectors of selected rows are transferred and combined with the previously stored columns to form a 4×4 matrix. In this process, the elements of the 4×4 matrix are grouped such way that four 3×3 sets are available with respect to the pixels encircled in Fig. 3. Each encircled pixel happens to be the center pixel of an eight neighborhood. By median
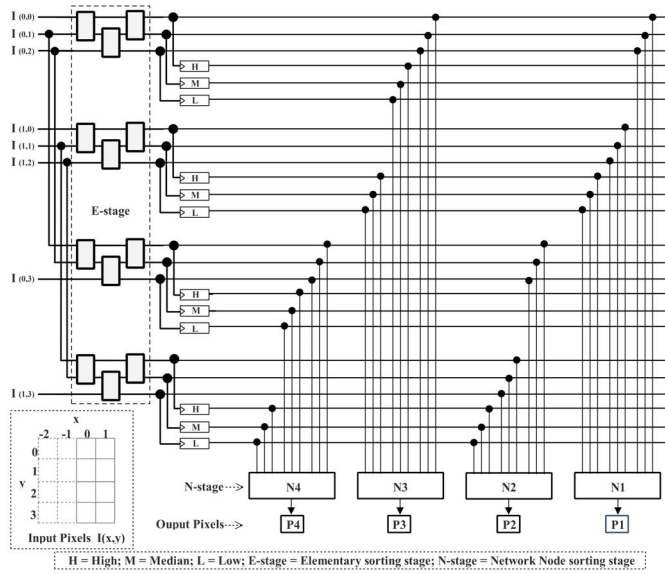
Fig. 4. Proposed CPMA.

filtering each set, four output pixels, namely, P1–P4 are generated. Once the execution of filtering process is over, the latest two column vectors are stored in S1 and S2; the latest two column vectors are now ready for concatenation with the new incoming set of vectors in the subsequent processing cycles. This process repeats until the last column vector is processed and the first sets of two rows are made available in the output image. For the selection of the next set of rows, only rows 3 to 6 need to be considered; the above process is repeated for finding the next two rows of the output image.

B. Column-Vectors Processing Median Filter Architecture

The column-vectors processing median filtering architecture (CPMA) is based on the proposed scheme for effective data handling and parallel filtering. CPMA requires 8 pixels, $I_{(0-1,0-3)}$, at the input. These 8 pixels are transferred as two 32 bit words. These are the column vectors of input pixels, $I_{(0,0-3)}$ and $I_{(1,0-3)}$. Input pixels representing the first and second columns are sorted in the ascending order and stored; the sorting is carried out by the E-stage. After E-stage, 24 values in eight different sorted sets are available. These sets provide inputs to four network nodes operating in parallel in the N-stage. N1 receives values from $I_{(-1,0-2)}$, $I_{(0,0-2)}$, and $I_{(1,0-2)}$ similar to F1 in Fig. 3. N2, N3, and N4 also receive values from input pixel matrix as shown in Fig. 4 similar to filters F2, F3, and F4. E-stage and N-stage are allocated each one machine cycle. Though E-stage has only three stages of compare-and-swap operations in comparison with six stages of compare-and-swap operations at the N-stage; it is a tradeoff between storage elements and speed. However, the speed has a dependence on the user's decision to split or not the N-stage. Pipelining between E-stage and N-stage requires 12 storage elements for holding the sorted set of values produced by the E-stage. All four

filtered output pixels are organized as a word; the word is subsequently loaded to the memory for write operation. CPMA, therefore, needs four machine cycles to complete read, E-stage, N-stage, and write operations. It is to be noted that the machine cycles of this section are different from that of the scheme for effective data handling described in the previous section.

IV. HARDWARE IMPLEMENTATION AND PERFORMANCE ANALYSIS

For the purpose of verifying the functionality of CPMA, the design is implemented in RTL Verilog HDL using XILINX ISE Design Suite 14.7 and synthesized for implementation in Xilinx FPGA Virtex 4 XC4VSX25. This FPGA is chosen for straight forward comparison with the state-of-the-art nonsorting methods [5], [8], sorting methods [10], [11], and sorting-based parallel architecture [4] available in the literature. The comparison is in terms of resource utilization and speed. The results are presented in Table I for the purpose of comparison of the performance metrics. The schemes in Table I employ 3×3 masks. Input sample width is 8 bits. The focus of the methods presented in [5], [8], [10], and [11] is on the design of new median finding methods with improved resource and time metrics and their pipelined architectures. For each cycle, these methods require nine input values and use the resources optimally. Although the proposed CPMA is based on the existing sorting method in [4], parallelism and pipelining reduce repeated comparator operations of the input pixels. The proposed CPMA also reduces the number of times a pixel is read and transferred for processing.

Architecture in [4] needs 12 pixel values as input. CPMA needs only 8 pixels as input. Reading of two words per cycle requires only one machine cycle for reading input pixels from memory, i.e., 8 pixels are transferred in two 32 bit words. In comparison, scheme in [4] needs two reading cycles before an execution cycle. The extra read operation results in the addition of one more machine cycle which is 25% extra burden on overall processing time in comparison with the proposed architecture.

Consider an image with $m \times n$ number of pixels. In the scheme proposed in [4], every pixel is read and transferred three times to the median finding architecture, i.e., $3 \times m \times n$ number of pixel read operations are executed for producing the filtered output image. In CPMA, an input image pixel is read and transferred to the architecture only twice, which means only $2 \times m \times n$ number of pixel read operations need to be executed for producing the filtered image. Proposed CPMA handles only two-third of the data in comparison with the best schemes available in the literature. Implementations presented in [5], [8], [10], and [11] require that pixel of the input image be handled nine times.

Table I lists the latency cycles in the pipelined designs. Throughput is displayed next. Throughput is defined as the number of median outputs generated per clock cycle. Fast median-finding word comparator array (FM-WCA) [11] is the latest median finding word comparator array and reported to be the fastest in the pipelined architectures, for example,

TABLE I
3×3 MEDIAN FILTER IN XILINX FPGA VIRTEX 4 XC4VSX25 ($N = 9$, INPUT SAMPLE WIDTH = 8 BITS)

Performance Metrics	LCBP[5]	Cadenas's [8]	Smith's[10]	FM-WCA[11]	Vega's (*)	CPMA (*)
CLB	459	254 / 284	1552	1552	3790 (947.5)	3770 (942.5)
DFF	516	507 / 478	368	344	200 (50)	224 (56)
LUT	632	336 / 567	152	152	832 (208)	832 (208)
f_{max} (MHz)	327	332 / 286 / 335	454	454	454 (1816)	454 (1816)
Latency (clock cycles)	8	7	9	8	9	9
Throughput (median outputs per clock cycle)	1	1	1	1	4 (1)	4 (1)
No. of times a pixel in input image to be read	9	9	9	9	3	2
Pipelined (P) / Pipelined & Parallel (P&P)	P	P	P	P	P&P	P&P

* Resources required per output pixel

TABLE II
3×3 MEDIAN FILTER IN XILINX FPGA VIRTEX 7 XC7VX330T

Performance Metrics	Smith's	FM-WCA	Vega's (*)	CPMA (*)
Slices	360	349	802 (200.5)	782 (195.5)
DFF	96	80	200 (50)	224 (56)
LUT	326	330	865 (216.25)	851(212.75)
f_{max} (MHz)	631	631	631 (2524)	631 (2524)

* Resources required per output pixel

low hardware complexity pipelined rank filter (LCBP) [5] and Cadenas's method [8]. However, Cadenas's method has fewer latency cycles. FM-WCA uses 7% fewer DFFs and smaller latency than Smith's. LCBP, Cadenas's, Smith's, and FM-WCA are implemented as pipelined architectures. Throughput for all these methods is same and not modified for improvements at the same speed. Parallelism with pipelining is introduced in Vega's architecture and CPMA for the improvement of throughput at 454 MHz. The introduction of parallelism has reduced the amount of image data to be handled by these architectures. Vega's architecture handles only 33.3% of image data handled by pipelined methods. But CPMA handles only 66.6% image data handled by Vega's architecture. It is evident from Table I that the number of times a pixel in input image read is only twice in the complete filtering process of the input image. Per-throughput resources of CPMA are comparatively fewer than Vega's architecture, and speed of 1816 MHz is significantly higher than FM-WCA and other methods. Table II presents the implementation of sorting-based methods for evaluating resource utilization and the speed on the state-of-the-art prototyping platform. Performance of CPMA is further confirmed by Xilinx FPGA Virtex 7 implementation. CPMA uses at least 30% less hardware resources than FM-WCA and offers increased throughput four times that of FM-WCA. Similarly, CPMA uses 2.5% less logical resources and handles 33.3% less image data for the same throughput than architecture in [4].

V. CONCLUSION

Computationally intensive median filtering algorithms are a challenge in the context of real-time processing. For the basic median filter, optimization of the amount of data handled at the architecture level with pipelining and parallelism of existing systolic array is considered. While the number of filter processing cycles is fixed by the size of the image to be filtered, median filtering by the proposed architecture requires reduced data handling. The proposed architecture saves significant amount of resources and the time required to handle median filtering of images. Further optimization on pipelining and parallelism may lead to further improvement in the context of real-time processing.

REFERENCES

[1] G. A. Baxes, *Digital Image Processing: Principles and Applications*. New York, NY, USA: Wiley, 1994.
[2] M. J. S. Smith, *Application-Specific Integrated Circuits*. Reading, MA, USA: Addison-Wesley, 2008.
[3] S. Hauck, "The roles of FPGAs in reprogrammable systems," *Proc. IEEE*, vol. 86, no. 4, pp. 615–638, Apr. 1998.
[4] M. A. Vega-Rodríguez, J. M. Sánchez-Pérez, and J. A. Gómez-Pulido, "An FPGA-based implementation for median filter meeting the real-time requirements of automated visual inspection systems," in *Proc. 10th Mediterr. Conf. Control Autom.*, 2002.
[5] D. Prokin and M. Prokin, "Low hardware complexity pipelined rank filter," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 57, no. 6, pp. 446–450, Jun. 2010.
[6] J. Cadenas, G. M. Megson, R. S. Sherratt, and P. Huerta, "Fast median calculation method," *Electron. Lett.*, vol. 48, no. 10, pp. 558–560, May 2012.
[7] J. Cadenas, "Pipelined median architecture," *Electron. Lett.*, vol. 51, no. 24, pp. 1999–2001, Nov. 2015.
[8] J. O. Cadenas, G. M. Megson, and R. S. Sherratt, "Median filter architecture by accumulative parallel counters," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 7, pp. 661–665, Jul. 2015.
[9] B. Morcego, J. Frau, and A. Català, "Suavizado de imágenes en tiempo real mediante filtrado por mediana utilizando arrays sistólicos," in *Proc. VII DCIS*, Toledo, Spain, 1992, pp. 545–546.
[10] J. L. Smith, "Implementing median filters in xc4000e FPGAs," *Xilinx Xcell*, vol. 23, no. 1, p. 16, 1996.
[11] J. Subramaniam, J. K. Raju, and D. Ebenezer, "Fast median-finding word comparator array," *Electron. Lett.*, vol. 53, no. 21, pp. 1402–1404, Dec. 2017.