Sādhanā (2020)45:75
https://doi.org/10.1007/s12046-020-1292-9

VLSI implementation of high throughput parallel pipeline median finder for IoT applications

VASUDEVA BEVARA and PRADYUT KUMAR SANKI*

Department of ECE, SRM University-AP, Guntur, India
e-mail: pradyut.s@srmap.edu.in

MS received 16 December 2019; revised 30 December 2019; accepted 1 January 2020

Abstract. This paper proposes a high-throughput median finding architecture where the sorting of an incoming pixel is executed by a high-speed Compare and Select (CS) module. In this work, four clock pulses are required to populate the 4×4 window as four pixels are read at a time from the incoming grey image. This median finding process is carried out by parallel and pipeline median architecture. The proposed median finding process requires two read operations to take eight input pixels and generates four output pixels with a latency of seven clock cycles. The proposed architecture has been implemented on Xilinx Virtex–VII FPGA. The proposed architecture is synthesized using the SoC Encounter along with Faraday 90 nm standard cell library. The maximum operating frequency is 950.57 MHz, the total gate count is 4540, area is 0.40543 mm^2 and the dissipated power is 0.92617 mW. The high-throughput, high-speed and low-power-dissipation nature of the proposed architecture make it suitable for computationally extensive Internet of Things (IoT) applications.

Keywords. Low latency; compare and select module; parallel median filter; pipeline median filter; ASIC design.

1. Introduction

With the rapid growth of wireless sensor devices with high-speed communication and faster computing capability, smart visual Internet of Things (IoT) has been most commonly used in various industries [1]. The IoT platform provides support for various industries, such as healthcare, smart city, cloud computing, artificial intelligence and automotive industries [2, 3]. The IoT devices consist of various sensors, cameras, GPS *etc.* Information acquired by various sensor devices can satisfy the need for compression and assembly of multi-media information, such as images and videos in real-time applications [4]. IOT for capturing image and video needs high-throughput sensor devices. Captured images commonly sufferer from impulse noise during acquisition, processing, transmission and storing, which lead to loss of information and degraded performance of the devices.

In real-time applications, the digital images are transferred through a stage of image pre-processing in order to remove distorted and noisy information from the digital images, which means the images are mainly either distorted or corrupted due to process variations and environmental noise [5]. The existence of non-Gaussian noise is one of the most habitual problems in digital images.

Non-linear digital filters are used to eliminate non-Gaussian noise and retrieve the valuable information from the noisy images [6]. The median filter is one of the non-linear filters, and it is used to remove the non-Gaussian noise (especially impulse noise) from the noisy digital images [7]. The median filter is a more robust algorithm than the traditional linear and non-linear filters because it preserves the sharp edges [8, 9] with better efficiency.

The software implementation of the median filter algorithm does not support digital image processing in real time. The existing algorithm based on median filter does not support the high-throughput requirement of IoT-based system. Re-configurable computing architecture (i.e., FPGAs) [10, 11] is sufficiently flexible and cost-effective. Therefore, new operations provide sufficient opportunities for the implementation of application-specific architectures [12, 13] and they are quite fast for real-time applications.

The architecture reported by Smith [14] requires a latency of nine clock cycles with a single throughput. The algorithm introduced by Vega-Rodriguez *et al* [15] improves the throughput to four with a latency of nine clock cycles. The systolic array-based median architecture implemented by Vega-Rodriguez *et al* [15] is 32-bit word length architecture for data transfer in a digital image. In this architecture, the filtered output pixels are generated at a time from four parallel networks. This architecture needs 76 comparators for generating four output pixels as

*For correspondence
Published online: 19 March 2020

reported in the literature. The Cadenas [16] method gets the output after a latency of seven clock cycles.

The proposed fast median finder architecture uses only 32 comparators for generating four filtered output pixels. A pixel is read only twice from the digital input image, and it produces the output with a throughput of four. The architecture can be employed for automation industries, real-time digital controllers, CNN-based automated surveillance systems [17], image and video processing applications.

The rest of this paper is organized as follows. Section 2 presents the median filter algorithm for parallel pipeline processing. In section 3 the proposed architecture for parallel pipeline median filter is described, while section 4 illustrates the hardware-implemented results and discussion; the conclusion is presented in section 5.

2. Median filter algorithm for parallel processing

The median filter algorithm for parallel and pipelined process is illustrated in figure 1. The objective of the proposed algorithm is to productively handle the input pixels so as to decrease the replicated read operations. The first four rows and columns of the input image pixels are selected as shown in figure 1(A), and each column vector (CV) has 32-bit data word length. The four CVs are formed as a 4×4 window; it is divided into four 3×3 windows and the median filter can be applied with respect to four central pixels as shown in figure 1(B).

In the proposed algorithm, only two words are passed, instead of more than three words for completing a single cycle process. The proposed algorithm requires only a single processing cycle for generating four output pixels. These four output pixels are generated using four median filters, namely MF1−MF4. In the first machine cycle, the first two CVs are transferred and stored in data elements. The size of each data element is 4×1. In the second machine cycle, the next two words (i.e. third and fourth CVs) of selected rows are transferred and form a 4×4 window with previously stored CVs. In this process, the pixels of the 4×4 window are divided into four 3×3 windows as shown in figure 1. Each encircled pixel should be the centre pixel of eight neighbourhood pixels as shown in figure 1. The process is used to generate four output pixels (P1−P4) using four median filters. Once the filtering process is executed, the last two CVs are stored in data elements. Now, these two CVs are ready for concatenation with the next incoming set of two CVs in the successive processing cycles. This process repeats until the last CVs, and the first two row vectors are made available in the output digital images. For the selection of next output row vectors, only 3−6 rows are considered for the next set of rows in the selection process. This procedure is repeated for finding the next two row vectors of the output digital image.

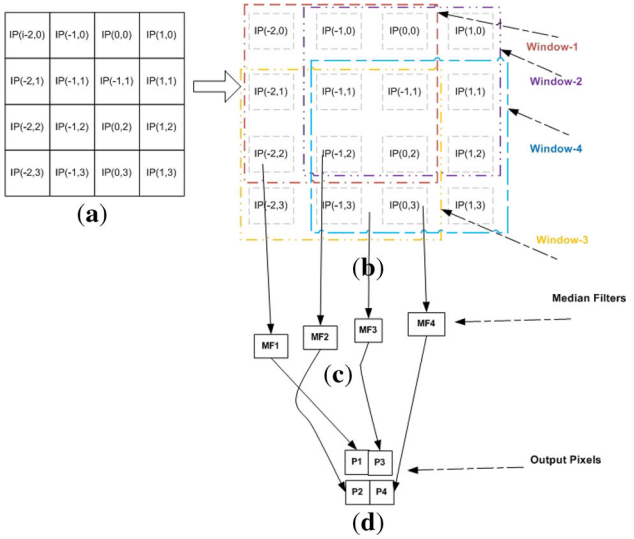

Figure 1. Algorithm of proposed architecture: (**A**) 4×4 window, (**B**) four 3×3 windows, (**C**) median filters and (**D**) output pixels.

3. Parallel pipeline median filter architecture

The proposed pipe-lined median filter architecture is illustrated in figure 2. The architecture requires eight input pixels, namely IP(0−1,0−3). These eight pixels are transferred as two CVs; each CV as 32-bit word size. IP(0,0−3) and IP(1,0−3) are considered as the first two columns of input pixels. These two CVs are arranged in ascending order and stored in data storage elements, namely, D1−D12. In the first stage, the sorting is carried out using Compare and Select (CS) modules. After the sorting, 24 values are available and they are divided into eight different sets.

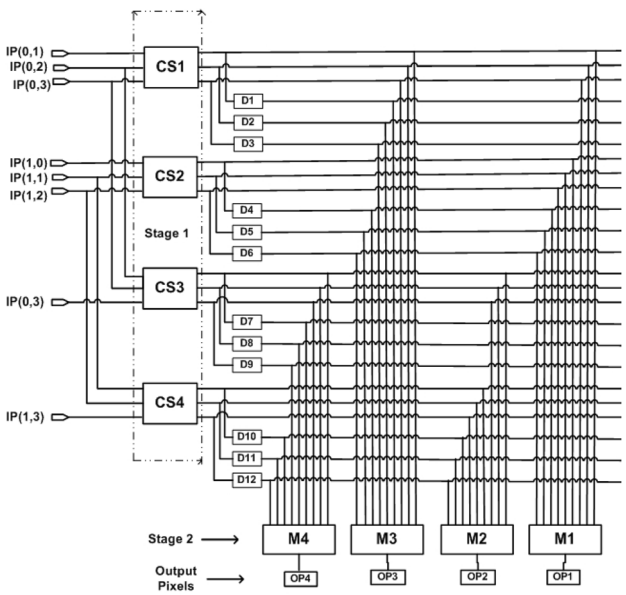

Figure 2. Architecture of proposed parallel pipelined median filter.

The eight different sets are provided as inputs to four median filters, which are operated in parallel in stage 2, namely M1−M4. M1 receives pixels from IP(−1,0−2), IP(0,0−2) and IP(1,0−2). M2–M4 also receive values from input pixels, as shown in figure 2. The four median filters, which do a similar operation, are compared to the filters used in figure 1. Stages 1 and 2 are allocated one machine cycle each. Stage 1 has four CS modules and stage 2 has four median filters. The pipeline between stage 1 and stage 2 requires 12 data storage elements (D1−D12). These storage elements hold the sorted set of values produced by the first stage. All the output pixels (OP1−OP4) of four median filters are organized as a word and it is subsequently loaded to the memory for a write operation. The proposed process requires four machine cycles (*viz.* read, stage 1, stage 2 and write operations) to complete the process.

3.1 *Median filter architecture*

The median filter has a spatial filtering operation, and it uses a two-dimensional (2D) mask. It is applied to each pixel in the digital input image. The input pixels are arranged in ascending order and the central pixel is selected. This selected pixel is considered as a median pixel. The obtained median value will be the pixel value in the output digital image.

The architecture of 2D median filter, in order to find the median value of nine input pixels, is illustrated in figure 3. All nine input pixels are arranged in a 3 × 3 window and it needs 3 stages to find median value. In the first stage, three rows are sorted using CS1−CS3 modules. In the next stage, three columns are sorted using CS4−CS6. The diagonal pixels in the 3 × 3 window are sorted in the final stage. The middle value of the last stage (CS7) is considered as a median value of 3 × 3 window.

3.2 *CS module*

The proposed architecture of CS module is illustrated in figure 4. It is implemented using three 8 × 1 multiplexers

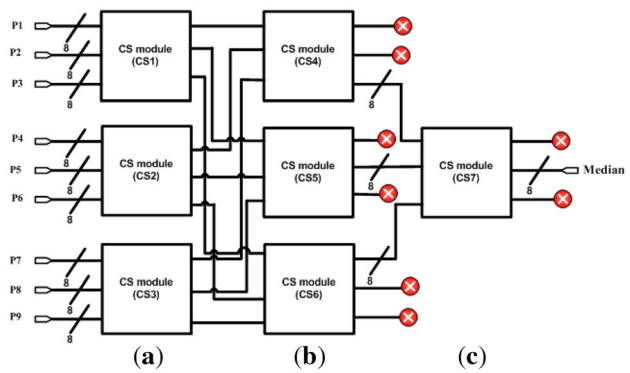

Figure 3. Block diagram of proposed median filter.

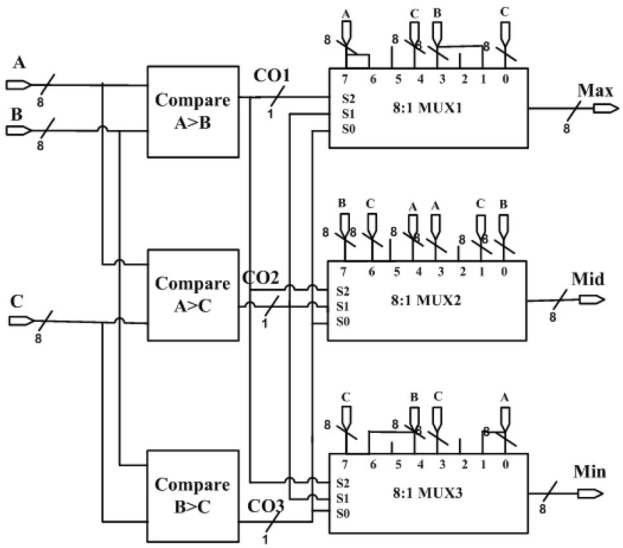

Figure 4. Architecture of compare and select module.

and three comparators (*AgtB*, *AgtC*, *BgtC*). All three comparators and multiplexers are operated in parallel. Hence, it requires a single cycle for processing. Each comparator compares two 8-bit input streams and it gives a single-bit output.

The first comparator compares two input values *A* and *B* and generates CO1 output bit. Similarly, the second and third comparators compare *A* and *C*, *B* and *C* and generate CO2, CO3, respectively. The three comparator output (CO) values are considered as selective lines (S0, S1 and S2) for 3 multiplexers. The output of three multiplexers (Mux1 producing Maximum (Max), Mux2 producing Middle (Mid), and Mux3 producing Minimum (Min) values) produces necessary values. The inputs and selective lines of each multiplexer are shown in table 1.

3.3 *Comparator architecture*

The proposed comparator architecture is shown in figure 5. The comparator is designed using basic logic gates and 2:1 multiplexers. The comparator compares two 8-bit numbers, and it generates a 1-bit CO. If CO = 0, the first input is less than the second input, else the first input is greater than the second input.

4. Hardware-implemented results and discussion

4.1 *FPGA implementation result*

The proposed architecture needs only one read cycle before execution, in contrast with two read cycles requirement of the architecture proposed in [15]. Due to its parallel

Table 1. Table for selecting inputs and outputs of MUXes.

Comparators outputs			Multiplexers outputs		
$A > B$ CO1 (S2)	$A > C$ CO2 (S1)	$B > C$ CO3 (S0)	MUX1 (Max)	MUX2 (Mid)	MUX3 (Min)
1	1	1	*A*	*B*	*C*
1	1	0	*A*	*C*	*B*
1	0	0	*C*	*A*	*B*
0	1	1	*B*	*A*	*C*
0	0	1	*B*	*C*	*A*
0	0	0	*C*	*B*	*A*

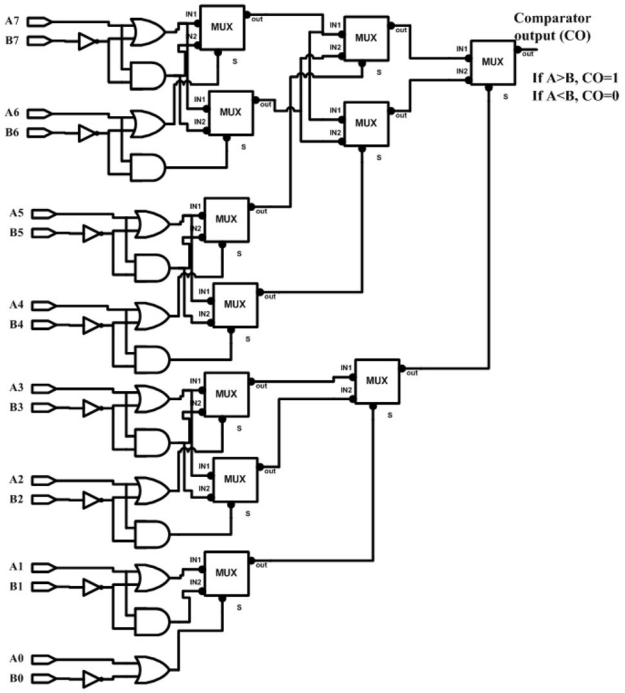

Figure 5. Architecture of eight-bit comparator.

architecture, the proposed median filtering technique reads eight input pixels (*i.e.* two words (32 bits)) per cycle. Consider the input digital image having *CR* number of pixels where *C* and *R* represent the number of columns and rows, respectively. Vega-Rodriguez *et al* [15] proposed an architecture where every input pixel is transferred three times to the median filter architecture for reading operation, which means 3*CR* number of reading operations are executed for generating the output image. In the proposed architecture, the input pixel is transferred only two times to the median filter architecture for the read operation, i.e. 2*CR* number of reading operations are executed for generating the output image. The proposed architecture is designed with the help of Verilog HDL using Xilinx ISE design suite 14.7 and the design is synthesized in Xilinx FPGA virtex7-XC7VX330T. The device utilization and maximum operating frequency after post-place and route simulation have been reported in table 2.

Table 2. Synthesis report of the proposed architecture (device selected: XC7VX330T-FFG1157 VIRTEX 7).

Proposed architecture: median filter architecture			
Max. operating frequency: 950 MHz			
Resource	Available	Utilized	Utilization (%)
No. of slice registers	408000	1358	<1
No. of slice LUTs	204000	1027	<1
No. of LUT–FF pairs	1876	509	27

Table 3. Comparison of median filter architecture in XILINX FPGA Virtex 7 XC7VX330T.

Resources	[13]	[14]	[15]	[16]	[18]	Proposed
Slices	349	360	802	326	451	1027
LUT	330	326	865	336	766	849
fmax (MHz)	631	631	631	332	326	950

Table 3 reflects the proposed architecture; it has high throughput with high speed, but at the cost of a critical growth in area as compared with the papers reported in [13–16, 18]. The architecture proposed in Vega-Rodriguez *et al* [15] requires eight machine cycles and FMWCA [13] architecture needs nine machine cycles for the median calculation process, but the proposed parallel and pipeline median filter architecture requires only seven machine cycles.

4.2 *ASIC implementation result*

The proposed parallel and pipeline median filter architecture has been synthesized using RTL compiler of Cadance along with 90-nm CMOS Technology. In this proposed architecture, a total of 4540 gates occupy a chip area of 0.40543 mm^2. The proposed circuit chip layout is done using the SoC encounter tool. The power consumption of the chip is 0.92617 mW, and it is analysed using prime power. Figure 6 depicts a complete parallel and pipelined median filter architecture bounded chip. The complete circuit of the proposed parallel and pipelined median filter

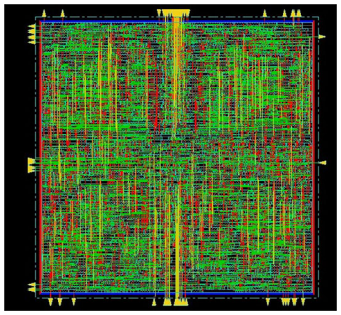

Figure 6. Parallel and pipelined median filter architecture chip design.

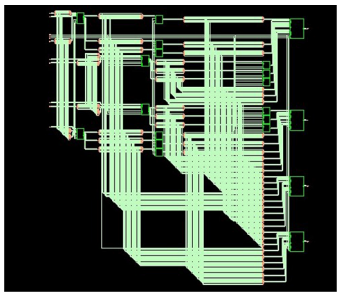

Figure 7. Complete circuit of parallel and pipelined median filter architecture using 90-nm CMOS technology.

architecture using 90-nm CMOS technology is illustrated in figure 7.

5. Conclusion

This paper proposes a new ASIC and FPGA implementation of parallel and pipelined median finding process for eliminating impulse noise for image and video processing. The proposed parallel and pipelined median filter is better than the existing median filters in terms of frequency and throughput. The high-speed and high-throughput performances of the proposed architecture make it suitable for computationally extensive IoT applications such as IoT-based 3D digital image and video processing.

Acknowledgements

The authors wish to thank Department of ECE, SRM University-AP, Guntur, India, for their continuous support and encouragement.

References

[1] Sehrawat D and Gill NS 2019 Smart sensors: analysis of different types of IoT sensors. In: *3rd International Conference on Trends in Electronics and Informatics (ICOEI)*

[2] Atzori L, Iera A and Morabito G 2010 The internet of things: a survey. *Computer Networks* 54(15): 2787–2805

[3] Li Y, Sun X, Wang H, Sun H and Li X 2012 Automatic target detection in high-resolution remote sensing images using a contour-based spatial model. *IEEE Geoscience and Remote Sensing Letters* 9(5): 886–890

[4] H. Li, S. Liu, Q. Duan and W. Li 2018 Application of multi-sensor image fusion of internet of things in image processing. *IEEE Access* 6: 50776–50787

[5] Burian A, Takala J and Topa M D 2003 Parallel iterations for recursive median filter. In: *Proceedings of the 2003 International Symposium on Circuits and Systems (ISCAS '03)*, Bangkok, pp. IV–IV

[6] Itai A, Funase A, Cichocki A and Yasukawa H 2012 Non-linear filter based outer product expansion with reference signal for EEG analysis. In: *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, Seoul, pp. 345–348

[7] Buell D A and Pocek K L 1995 Custom computing machines: an introduction. *The Journal of Supercomputing* 9(3): 219–229

[8] Fan Q, Yang J, Hua G, Chen B and Wipf D 2012 A generic deep architecture for single image reflection removal and image smoothing. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Seoul, pp. 3258–3267

[9] Biradar R L and Kohir V V 2013 A novel image inpainting technique based on median diffusion. *Sadhana* 38(4): 621–644

[10] Aras E, Delbruel S, Yang F, Joosen W and Hughes D 2019 A low-power hardware platform for smart environment as a call for more flexibility and re-usability. In: *Proceedings of the 2019 International Conference on Embedded Wireless Systems and Networks (EWSN 19)*. USA: Junction Publishing, pp. 194–205

[11] J. Villasenor and W. H. Mangione-Smith 1997 Configurable computing. *Scientific American* 276(6): 66–71

[12] Benkrid K, Crookes D and Benkrid A 2002 Design and implementation of a novel algorithm for general purpose median filtering on FPGAs. In: *Proceedings of the IEEE International Symposium on Circuits and Systems*. Cat. No. 02CH37353, vol. 4, pp. IV–IV

[13] J. Subramaniam, J. K. Raju, and D. Ebenezer 2017 Fast median-finding word comparator array. *Electronics Letters* 53(21): 1402–1404

[14] J. L. Smith 1996 Implementing median filters in xc4000e FPGA's. *Xilinx Xcell* 23(1): 16

[15] Vega-Rodriguez M A, Sanchez-Perez J M and Gomez-Pulido J A 2002 An FPGA-based implementation for median filter meeting the real-time requirements of automated visual inspection systems. In: *Proceedings of the 10th Mediterranean Conference on Control and Automation*

[16] J. Cadenas 2015 Pipelined median architecture. *Electronics Letters* 51(24): 1999–2001

[17] Cameron J, Savoie P, Kaye M E and Scheme E 2019 Design considerations for the processing system of a CNN-based automated surveillance system. *Expert Systems with Applications* 136: 105–114. https://doi.org/10.1016/j.eswa.2019.06.037

[18] D. Prokin and M. Prokin 2010 Low hardware complexity pipelined rank filter. *IEEE Transactions on Circuits and Systems II: Express Briefs* 57(6): 446–450