Sādhanā (2020)45:192
https://doi.org/10.1007/s12046-020-01431-0

A new fast and efficient 2-D median filter architecture

VASUDEVA BEVARA and PRADYUT KUMAR SANKI*

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

MS received 4 January 2020; revised 3 February 2020; accepted 2 March 2020

Abstract. Existing architectures for the median filter are based on sorting algorithm where comparators are used in serial. This paper proposes a new high-speed architecture of two dimensional (2-D) median filter where compare and select modules are used in parallel to sort the incoming numbers. The hardware implementation results show that the proposed architecture (PA) operates at 26% and 34% higher frequency in Virtex 4 and Virtex 7 FPGA device, respectively, in comparison with the architectures reported. The PA is synthesized using the RTL Compiler of Cadence along with Faraday 180 nm standard cell library. The maximum operating frequency of the PA is 1.06 GHz with a total gate count of 917. The complete chip layout has been done using the SoC encounter tool. The area of the final chip is 0.13928 mm^2 with a power consumption of 0.168 mW analysed using prime-power.

Keywords. Median filter; latency; compare and select module.

1. Introduction

Digital images are mostly either corrupted or distorted due to many reasons such as process variations, sensor misalignment, faulty storage locations, environmental noise, etc. It is of utmost importance to retrieve the valuable information from the noisy images available [1] with the advent of big data, cloud computing and artificial intelligence. It is a very common practice to use nonlinear digital filters for non-Gaussian noise elimination. The median filter is one of the nonlinear filters, and it is the most popular algorithm to suppress the random noise (especially impulse noise) and periodic patterns from noisy images. The median filter algorithm is extensively used in image processing due to its effective noise elimination capability and better efficiency [2]. The limitation in the existing model is that the time complexity increases as the number of input elements is increased while sorting that is required to find the median value. Moreover, conventional sorting techniques like the bubble sort, selection sort, merge sort, radix sort, heap sort, bucket sort and quick sort are slow in process [3]. In digital image processing applications, most of the embedded systems require a high-speed median filter. Basically the predefined architectures of basic median filter perform the median operation in two ways: the first one is sorting-based systolic arrays [4, 5], where a number of basic nodes (i.e. two-element compare and swap circuit) are required for sorting, making it a slow process as well as requiring more space; the second one is non-sorting techniques [6–10], which require more storage and more latency.

The proposed architecture (PA) handles image data effectively in such a way that the data stream in the input image is read-only once in comparison with the existing algorithm [5]. The speed of the median filter is increased using the proposed compare and select (CS) module for finding the median value. The number of reading operations for filtering an image as well as overall operating time is reduced in the PA.

The PA for the two-dimensional (2-D) median filter algorithm has been implemented using an XC7VX330T-VIRTEX 7 FPGA device.

The proposed median filter architecture (MFA) increases the speed of the process and handles the digital image data effectively using the CS module in comparison with architectures reported in the literature, where triple number sorting architecture is used. The reduction in the overall operating time for filtering digital images ensures the reduction in latency of PA. In addition, the proposed MFA is synthesized using SoC encounter; it provides less gate count and the area of the final chip is also much less. The PA increases the speed of communication and reduces the total cost of the process. The proposed median filter can be used for real-time digital image and video processing applications.

2. ASIC implementation of 2-D median filter

2.1 *2-D median filter algorithm*

In the fast 2-D median filtering algorithm [11] the $m \times n$ window moves only one column, to get one output picture element to the next, where m and n represent the number of

*For correspondence
Published online: 01 August 2020

rows and columns of the window, respectively. The new window gets the numbers from the earlier window; thereafter n elements are removed and new n elements are buffered into the window. The remaining elements are all unchanged. Using this algorithm, 9 input data streams are considered. All the 9 input data streams are arranged in a 3×3 window form. This algorithm finds the median of 3×3 window in the following 3 steps.

Step 1: Sorting the three rows.
Step 2: Sorting the three columns.
Step 3: Sorting the diagonal numbers in the window.

After three steps of sorting, the middle value of 3×3 window is the median value of the given 9 input data streams [12]. The procedure of the median filter algorithm [13] is shown in figure 1.

2.2 *2-D MFA*

The proposed 2-D median filter has the advantage of lower hardware complexity. In order to find the median value of 9 input values, all the 9 values are arranged as a 3×3 window without considering their alignment. The proposed median filter mainly uses the CS modules, and a block-level diagram of MFA [14] for median value calculation is shown in figure 2.

In the first step, each of the three rows of 3×3 window is sorted using 3 CS modules (CS1, CS2 and CS3) and the sorted rows are placed in a new 3×3 window as shown in figure 2(a). In the second step, each of the three columns of the new window is sorted using 3 CS modules (CS4, CS5 and CS6) and the sorted columns are placed in the new 3×3 window as shown in figure 2(b). In the third step, the diagonal values are sorted using a single CS module (CS7) as shown in figure 2(c) and the middle value is considered as the median value of 3×3 window. Finally, it is evident that the MFA requires 7 CS modules.

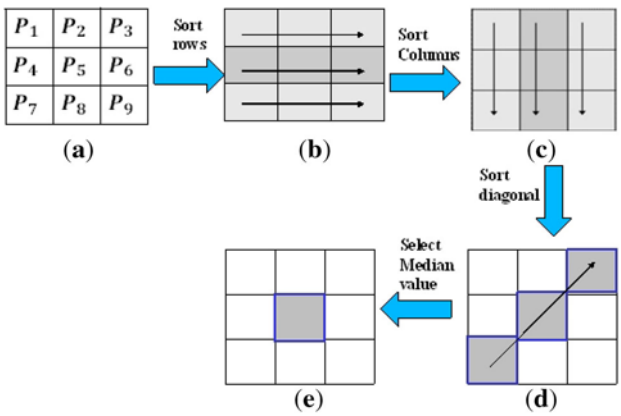

Figure 1. Algorithm of 3×3 window: (**a**) 3×3 window, (**b**) sorted rows, (**c**) sorted columns, (**d**) sorted diagonal and (**e**) median value.

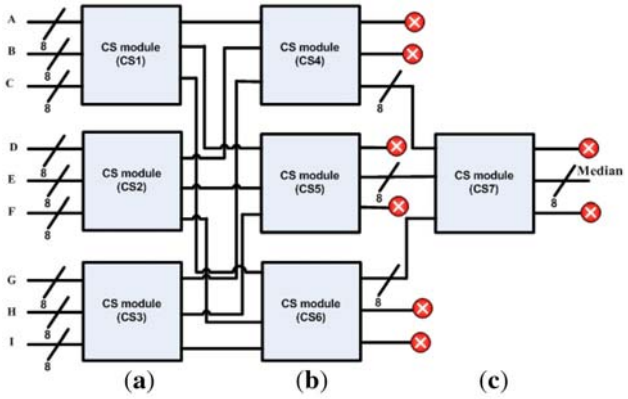

Figure 2. Block diagram of median filter architecture: (**a**) sorted rows, (**b**) sorted columns and (**c**) sorted diagonal.

2.3 *CS module*

A block-level diagram of the CS module [14] is shown in figure 3. The CS module is implemented using 3 comparators and 3 8:1 multiplexers, where all the comparators and multiplexers operate in parallel. The CS module is faster than the conventional 3-value sorter, which uses 3 comparators in series.

In this architecture, each comparator compares 2 input streams and it gives the single-bit output (CO) used as a selection line to select the sorted data steams. The first comparator compares the input data A and B, generating the output bit CO1. Similarly the second comparator compares A and C, and the third comparator compares B and C and generate output bits CO2 and CO3, respectively. All three CO values are taken as select lines of 8:1 multiplexers. The first multiplexer gives the output as the maximum (Max) number of 3 input data streams. Similarly, the

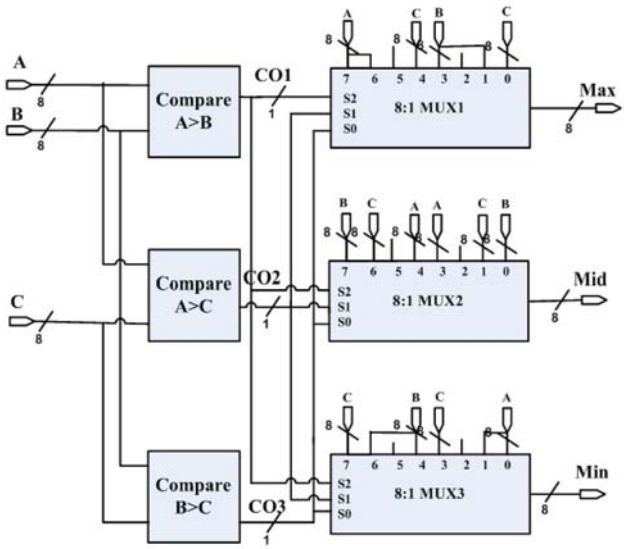

Figure 3. Block diagram of compare and select module.

second multiplexer gives the output as the middle (Mid) and the third multiplexer gives the output as minimum (Min) values of 3 input data streams. The inputs (A, B and C) and the selective lines (S0, S1 and S2) of each multiplexer are taken as shown in table 1.

2.4 *Comparator architecture*

Figure 4 illustrates the architecture of the comparator [14], and the comparator made up of basic logic gates and 2:1 multiplexers. The comparator module compares two 8-bit numbers and generates a single output bit, i.e. CO. If CO = 1, the input 1 is greater than the input 2; else CO = 0 and input 2 or input A is greater than input 1 or input B.

3. FPGA and ASIC implementation result

3.1 *FPGA implementation result*

The PA has been implemented in an XC4VSX25-SF363 VIRTEX 4 device. The maximum operating frequency and device utilization after post-place and route simulation using Xilinx ISE 14.7 have been reported in table 2. The total dynamic power consumption of MFA is 0.21 mW at an operating frequency of 574 MHz as per the Xilinx XPower Analyser.

3.2 *Comparison with some other architectures*

The MFA has been implemented and compared with some other recently implemented architectures [4–6, 9, 15, 16] as shown in table 3. The PA is compared with existing architectures, and the synthesis results are compared with those of various architectures as shown in table 3. The implementation based on odd merge sort [15] requires 349 slices and bitonic sorter [5] requires 782 slices but the PA requires 164 slices, which is much lower than those of the reported architectures.

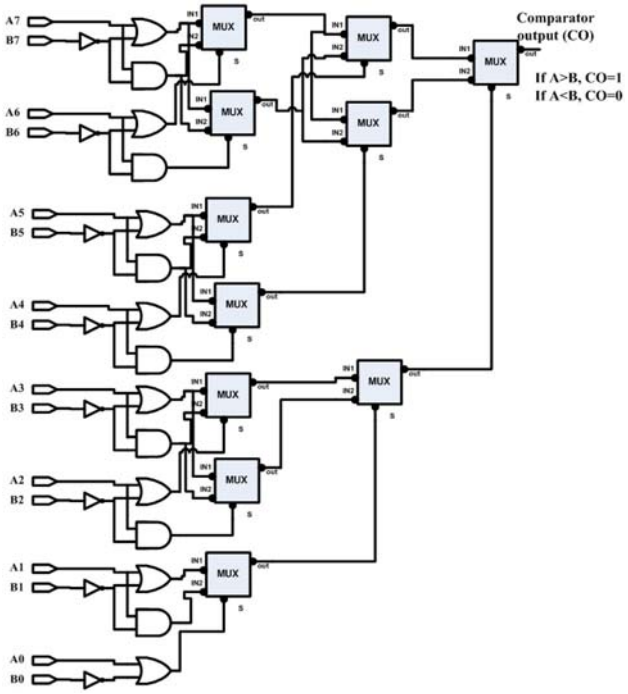

Figure 4. Architecture of 8-bit comparator.

Table 2. Synthesis report of the proposed architecture (device selected: XC4VSX25 - SF363 VIRTEX 4).

Proposed architecture: 2-D median filter			
Max. operating frequency: 574 MHz			
Resource	Available	Utilized	Utilization
No. of slice registers	408,000	121	<1%
No. of slice LUTs	204,000	76	<1%
No. of LUT–FF pairs	123	74	60%

The median calculation process requires 9 machine cycles as per the architecture reported by Smith [16] and 7 machine cycles as per the Cadenas model [6], whereas the

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

Table 3. Comparison of median filter architecture.

3 × 3 median filter in XILINX FPGA Virtex4 XC4VSX25							
(number of input data streams (N) = 9, width (W) = 8 bits)							
Resource	[9]	[6]	[16]	[15]	[4]	[5]	PA
CLB	459	254/284	1552	1552	3790	3770	160
DFF	516	507/478	368	344	200	224	147
LUT	632	336/567	152	152	832	832	112
fmax (MHz)	327	332/335	454	454	454	454	574
Latency	8	7	9	8	9	9	6
Throughput	1	1	1	1	4	4	1
No. of times a pixel in i/p image to be read	9	9	9	9	3	2	3

Table 4. Comparison of median filter.

3 × 3 median filter in XILINX FPGA					
Virtex 7 XC7VX330T (N = 9, width = 8 bits)					
Performance metrics	[16]	FMWCA [15]	Vega's [4]	CPMA [5]	PA
Slices	360	349	802	782	164
DFF	96	80	200	244	147
LUT	326	330	865	865	196
fmax (MHz)	631	631	631	631	839

PA requires only 6 machine cycles. The pipelined implementation of pre-defined architecture [5] employs 3770 CLBs and requires a latency of 9 clock cycles, while the PA employs 160 CLBs with 6 machine cycles only. The comparison in table 3 shows that the usage of CLBs in the pre-defined architecture [4, 5, 15, 16] is high because of the maximum utilization of the resources available. The PA uses only fewer 147 DFFs and 112 LUTs.

The latest median-finding word comparator array is one of the fast median-finding word comparator arrays [5]. The proposed algorithm reduces around half of the CLBs, 27% of LUTs and 80% of used DFFs; at the same time the maximum operating frequency increases by 30%. All these resources are mentioned in table 2.

Table 4 presents the implemented and synthesized sorting-based algorithms for evaluating the utilization and the speed on the state of the art prototyping platform. The performance of the PA is further confirmed by XILINX FPGA Virtex 7 and the performance comparison metrics are shown in table 4.

3.3 *ASIC implementation result*

The complete 2-D MFA has been synthesized using RTL Compiler of Cadence along with Faraday 180 nm standard cell library. The maximum operating frequency of the 2-D median filter module is 1.06 GHz and the total gate count is 917. The complete chip layout has been prepared using the SoC encounter tool. The reported area of the final chip is 0.13928 mm^2. The power has been analysed using prime-power and the reported power consumption is 0.168 mW. The layouts of the PA and the CS module have been shown in figures 5 and 6, respectively.

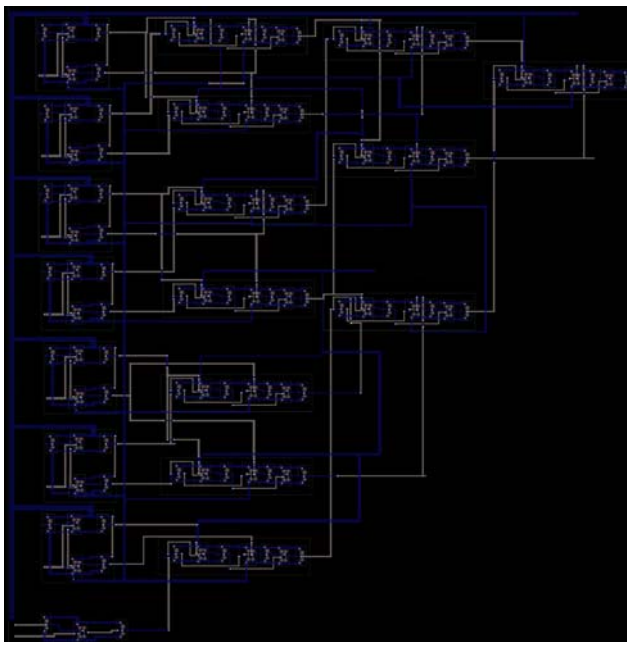

Figure 5. Layout of CS module.

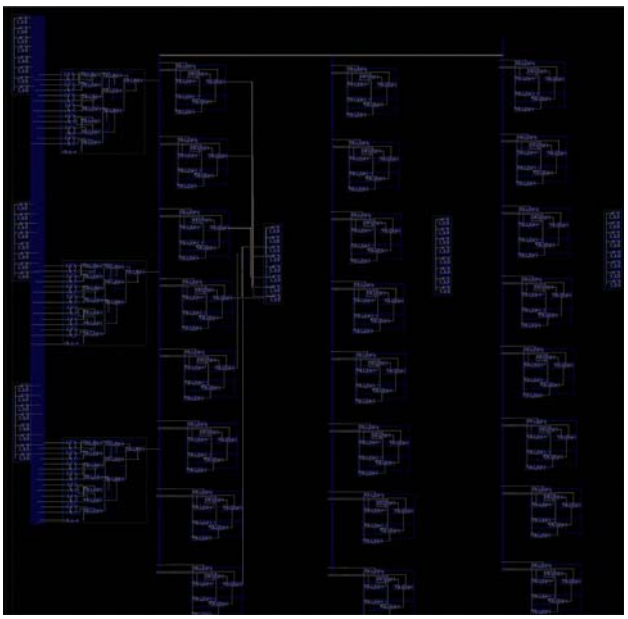

Figure 6. Layout of 2-D median filter architecture.

4. Conclusion

This paper presents a low latency hardware architecture of the 2-D median filter for image and video processing applications. The CS module is used in parallel to design the proposed 2-D median filter, which gives a throughput of 1 pixel per clock cycle for the median calculation of the 3×3 window of incoming digital images. The designed CS module is faster than the conventional 3-value sorter architecture. The proposed 2-D median filter can be used in real-time image processing applications as its processing time is 1.192 ns. The PA can be extended to design a 3-D median filter.

Acknowledgements

The authors wish to thank SRM University, Guntur, Andhra Pradesh, India, for their continuous support and encouragement.

References

[1] Karaman M, Onural L and Atalar A 1990 Design and implementation of a general-purpose median filter unit in cmos vlsi. *IEEE J. Solid-State Circuit.* 25(2): 505–513

[2] Tukey J W 1974 Nonlinear (nonsuperposable) methods for smoothing data. In: *Congress Record (EASCO)*

[3] Milutinovic V, Salom J, Veljovic D, Korolija N, Markovic D and Petrovic L 2017 *DataFlow supercomputing essentials: research, development and education*. Springer, Berlin

[4] 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*

[5] Subramaniam J, Kannan R J and Ebenezer D 2018 Parallel and pipelined 2-D median filter architecture. *IEEE Embedd. Syst. Lett.* 10(3): 69–72

[6] Cadenas J 2015 Pipelined median architecture. *Electron. Lett.* 51(24): 1999–2001

[7] Cadenas J O, Megson G M and Sherratt R S 2015 Median filter architecture by accumulative parallel counters. *IEEE Trans. Circuit. Syst. II Exp. Br.* 62(7): 661–665

[8] Cadenas J, Megson G M, Sherratt R S and Huerta P 2012 Fast median calculation method. *Electron. Lett.* 48(10): 558–560

[9] Prokin D and Prokin M 2010 Low hardware complexity pipelined rank filter. *IEEE Trans. Circuit. Syst. II Exp. Br.* 57(6): 446–450

[10] Astola J T and Campbell T G 1989 On computation of the running median. *IEEE Trans. Acoust Speech Signal Process.* 37(4): 572–574

[11] Huang T, Yang G J T G Y and Tang G 1979 A fast two-dimensional median filtering algorithm. *IEEE Trans. Acoust. Speech Signal Process.* 27(1): 13–18

[12] Bates G L and Nooshabadi S 1997 FPGA implementation of a median filter. In: *Proceedings of IEEE TENCON'97: IEEE Region 10 Annual Conference on Speech and Image Technologies for Computing and Telecommunications*, IEEE, Cat. No. 97CH36162, vol. 2, pp. 437–440

[13] Kumar V, Asati A and Gupta A 2017 Low-latency median filter core for hardware implementation of 5×5 median filtering. *IET Image Process.* 11(October): 927–934

[14] Bevara V and Sanki P K 2020 VLSI implementation of high throughput parallel pipeline median finder for IoT applications. *Sadhana* 45(75)

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

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