Indian Journal of Science and Technology, *Vol 7(4), 452–462, April 2014*

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

A Novel 8 Bit Digital Comparator for 3x3 Fixed Kernel based Modified Shear Sorting

K. Vasanth[1], A. A. F. Kavirajan[2*], T. Ravi[3] and Nirmal Raj[4]

[1,4]Department of E.E.E, Sathyabama University, Chennai-119, Tamil Nadu, India; vasanthecek@gmail.com
[2,3]Department of E.C.E, Sathyabama University, Chennai-119, Tamil Nadu, India; april04.kavirajan@gmail.com

Abstract

The need for an optimized area, speed and power plays a vital role for any median filter is good at removing impulse noise without degrading the image details. The main operation of the median is Rank ordering. It is a computationally complex operation, so it is hard to implement it in real time. This paper introduces a new sorting technique called for Snake like sorting. The proposed Sorting technique is implemented as a parallel architecture. This algorithm is a Mesh based sorting that require less number of comparators for rank ordering. The proposed architecture is compared with other Rank Ordering algorithm on the basis of power, speed, and area and found to exhibit good results. The proposed architecture is implemented on parallel and pipelined schemes and is targeted for Spartan 3e Device with gate capacity 5000 using Xilinx 7.1i compiler version. The pipelined scheme has an operating frequency of 81 Mhz occupying 283 slices with a gate count of 5,640.

Keywords: Borrow Look Ahead Select Comparator, Median Filter, Modified Shear Sorting, Salt and Pepper Noise

1. Introduction

Median filtering is a popular method of noise removal, employed extensively in applications involving speech and signal and image processing. This non-linear technique has proven to be a good alternative to linear filtering as it can effectively suppress impulse noise, while preserve edge information[1–18]. These properties make it very popular filter in speech processing and image processing schemes. There are two types of linear filters. Non-recursive and recursive. In non-recursive median filtering, a window is moved along the sampled values of the image and the center value of each window is replaced by the median of the values in the window. For instance in 2D non-recursive median filtering, the $(i,j)^{th}$ window of size($k*k$), W_{ij}, is centered at (i,j) and the $(i,j)^{th}$ output $y_{ij} = \text{median}\{w_{ij}\}$. In recursive median filtering, the window consists of recent median values as well as input values. In 1D recursive median filtering, the i^{th} window of size($2N + 1$),W_{ij} consists of ($N+1$) input values x_i……..x_{i+N}, and N output values y_{i-N}………y_{i-1}; $wi=\{y_{i-N},\ldots\ldots y_{i-1},x_i,\ldots\ldots\ldots x_{i+1}\}$ and the i^{th} output $y_i = \text{median}\{w_i\}$. The existing architecture for median filter can be broadly classified in to two classes. The array based architecture[2–4], and sorting network based architecture[5,6] gives an excellent survey of the existing architecture. The array based architecture consists of K processors of the windows is of size K, but they have a large sample period compared to sorting network based architecture. A good survey paper of VLSI median filters is discussed the author[7] where the hardware complexity is expressed 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[8]; word-level and bit-level. In this paper, only 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

*Author for correspondence

on the bubble sorting algorithm[9]. By means of a set of processing elements or PE^s, 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 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[10] realized on a systolic array architecture[11] 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. Sorting is one of the most commonly used data processing applications as a fundamental operation on a computer system. Much effort has been devoted to find out faster sorting algorithms because of its practical importance as well as its theoretical interest, Batcher[12] proposed a parallel sorting algorithm based on a bitonic sequence and obtained an execution time of $O(\log^2 N)$ for N data using $O(\log^2 N)$ processors. Stone[13] Also proposed a bitonic sorting with the perfect shuffle network and achieved an execution time of $O(\log^2 N)$ using $O(N\log^2 N)$ processors. Preparata[14] proposed another algorithm with $O(\log^2 N)$ time for $O(N\log N)$ processors. Horiguchi and sheigei[15] proposed a parallel sorting algorithm with $O(N)$ time for $O(\log N)$ linearly connected arrays. Thompson and Kung[16] and Nassimi and Sahni[17] extended Bacher's algorithm to a mesh connected arrays with N^2 Processors. They obtained the execution time of $O(N)$ for N^2 data. The revolutionary VLSI device technology has made practical and production of special purpose computing system with highly parallel structure. The systolic array is generally a set of relatively a simple processing unit of the same type, which are connected by a simple interconnection scheme and are able to be operated in parallel.[19,21] The architecture serves a very high performance, because the primitive cells use data from neighbours without having to store and retrieve intermediate results. Section II deals with the implementation of proposed algorithm. Section III deals with simulation results Section IV deals with conclusions.

2. Modified Shear Sorting

In this work Modified Shear sorting was used to showcase the proposed Borrow Look Ahead select Comparator (BLAC). The Modified shear sorting is one of the simplest and fastest median finding algorithms in recent years. The methodology of the modified shear sorting is given in Figure 1.

The algorithm of the modified shear sorting algorithm is as follows. Consider a 2D processing window as shown in Figure 1.

Step 1: The rows of the window are arranged in ascending order as shown in Figure 1a.

Step 2: The columns are arranged in ascending order as shown in Figure 1b.

Step 3: The right diagonal of the window is now arranged in ascending order as shown in Figure 1c. After this operation it was found that the first element of window is the minimum value, the last element of window is the maximum value and the centre element of window is the median value.

3. Proposed BLAC as Two Cell Sorter

In an 8 bit image, an eight bit comparator acts as a basic operation in finding the median. This paper presents a novel 8 bit comparator that uses BLAC to find the greater of two numbers. This is now referred as two cell sorter (a comparator). The proposed architecture is developed for two input comparator. The borrow look ahead select logic network is implemented using compare-swap function. The select logic in each BLAC is modified version of borrow equation of A-B subtract function. The proposed BLAC network is shown in Figure 2.

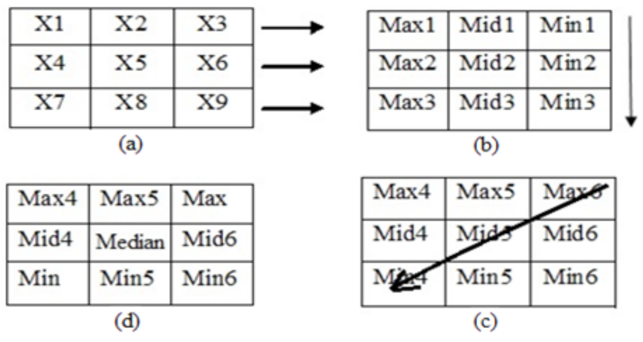


Figure 1. Illustration of the modified Shear sorting.

To understand the working of borrow look ahead select logic, consider the basic borrow equation of a full subtractor in equation 1. Let a and b are the two 8 bit inputs and cin refers to initial carry Cin (which is generally 0) and C refers to carry out.

$$C = ((\text{not } a) \text{ and } b) \text{ or } ((\text{not } a) \text{ and } Cin) \text{ or } (b \text{ and } Cin) \quad (1)$$

On assuming two functions to represent the basic borrow functions named as Generate (G), Propagate (P) as shown in equation 2 and 3. The subscript i represents the number of bits. In this case, the value of i varies between 0 and 7 where 0 and 7 refers to LSB and MSB of the given data respectively.

$$G_i = (\text{not } a_i) \text{ and } b_i \quad (2)$$
$$P_i = a_i \text{ xor } b_i \quad (3)$$

Substituting the equation 2 and 3 in 1 we get.

$$C_{i+1} = G_i + (\text{not } P_i)\, Cin \quad (4)$$

Now the equation 1 gets modified as equation 4. Vary the value of i from 0 to 7 resulting in a carry generation in C7 referred as carryout.

$$i = 0 \quad C0 = G0 \quad (5)$$
$$i = 1 \quad C1 = G1 + P1G0 \quad (6)$$
$$i = 2 \quad C2 = G2 + P2G1 + P2P1G0 \quad (7)$$
$$i = 3 \quad C3 = G3 + P3G2 + P3P2G1 + P3P2P1G0 \quad (8)$$
$$i = 4 \quad C4 = G4 + P4G3 + P4P3G2 + P4P3P2G1 + P4P3P2P1G0 \quad (9)$$
$$i = 5 \quad C5 = G5 + P5G4 + P5P4G3 + P5P4P3G2 + P5P4P3P2G1 + P5P4P3P2P1G0 \quad (10)$$
$$i = 6 \quad C6 = G6+P6G5+P6P5G4+P6P5P4G3+P6P5P4P3G2 + P6P5P4P3P2G1 + P5P4P3P2P1G0 \quad (11)$$
$$i = 7 \quad C7 = G7 + P7G6 + P7P6G5 + P7P6P5G4 + P7P6P5P4G3 + P7P6P5P4P3G2 + P7P6P5P4P3P2G1 + P7P6P5P4P3P2P1G0 \quad (12)$$

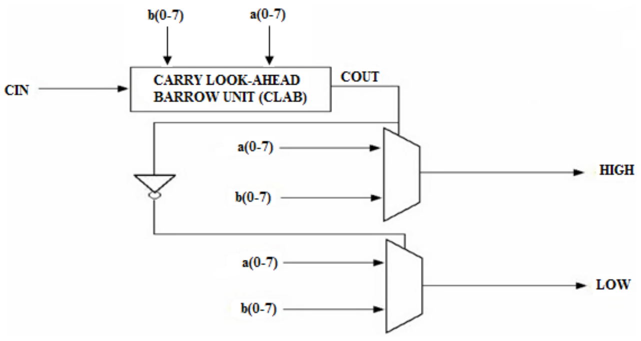


Figure 2. Illustration of Borrow look ahead select logic.

The generated carryout value act as a select line for the first 2:1 Multiplexer and the inverted Carryout value is fed to the another 2:1 Multiplexer which gives the HIGH pixel value and LOW pixel value is obtained after inverting the carryout. Equation 5 illustrates that the initial carry cin is 0. Hence the propagate function is also zero. So the first stage carry is equal to generate function.

4. Three Cell Sorter

The modified shear sorting algorithm extensively operates on three pixels at a time either in row, column or right diagonal. Hence a three pixel sorting is the basic operation for this algorithm as shown in Figure 3. The above is facilitated using three cell sorter. The output of the three cell sorter is maximum, middle and minimum of three pixels which are used for sorting. The internal architecture of three cell sorter requires three two cell sorter (used as BLAC) as shown in Figure 3.

The three cell sorter arranges three pixel elements in ascending order. The first two cells are compared in the first comparator. Here the comparator is BLAC, which results in a high and low. The low value is then compared with the third number in the second comparator resulting in a maximum and a minimum number. The minimum number of the second comparator is the minimum of the three numbers. The maximum value of the first and second comparator acts as an input to the third comparator. The output of the comparator results in maximum and minimum values. These values act as the maximum value of an array and median value of an array.

5. VLSI Architecture for Modified Shear Sorting

The paper presents two different architectures for modified shear sorting. The former is a parallel structure and

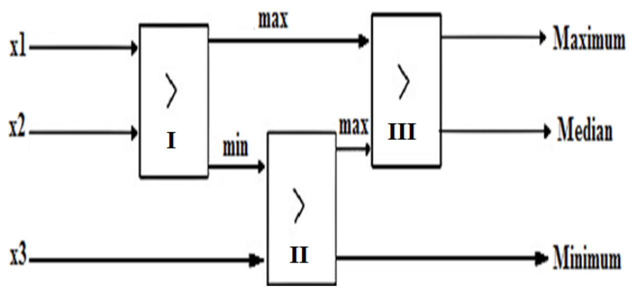


Figure 3. Internal architecture three cell sorter.

the later is a pipelined structure. The basic functional unit for these architectures are two cell sorter. Here this paper uses borrow look ahead select logic (BLAC) as two cell sorter. The higher processing element of the architecture is a three cell sorter. These three cell sorters were built using three two cell sorters (BLAC).

5.1 Parallel Architecture

In this proposed work, a minimum exchange network required to find the median value from nine input pixels values by performing a partial sort, is developed[20]. The Figure 4 shows the minimum exchange network for the parallel architecture. In this sorter seven three cell sorters are used, initially to perform Row sorting in three rows, three set of three cell sorters are used for comparison. Each three cell sorter exhibit outputs in the form of low, mid, high from its inputs. The high values of the first three cell sorter, the second three cell sorter and the third three cell sorter passes to the parallel first three cell sorter. The mid values of the first three cell sorter, the second three cell sorter and the third three cell sorter passes to the second parallel three cell sorter. The low values of the first three cell sorter, the second three cell sorter and the third three cell sorter passes to the third parallel three cell sorter. Then the low value of the first parallel three cell sorter, mid value of the second three cell sorter and the high value of the third parallel three cell sorter is compared using the final three cell sorter and the median value is finally produced at the output as shown in Figure 4.

5.2 Pipelined Parallel Architecture

Pipelining process leads to the reduction in the critical path, which reduces the speed. The circuit is implemented for pipelined using clock sequence. In this structure, at the first clock pulse the first set of three pixels values are passed into the three cell sorter and the outputs are passed into the three set of registers. In the second clock pulse, second set of three pixels values are passed into the same three cell sorter and stored in the next three set of registers. In the third clock pulse, last set of three pixels are passed into the same sorter and at the same instant the values stored in the two set of registers are compared using another three cell sorter and the result are obtained at the end of the third clock pulses. At the fifth clock pulse the results from the two sorter outputs are compared and the results are once again passed into the three cell sorter at the sixth clock pulse. At the seventh clock pulse the median output is obtained, the median is obtained at the seventh clock pulse as shown in Figure 5.

6. Simulation Results & Discussions

This paper compares conventional 8 bit comparator and carry select logic[23] with the proposed BLAC. The gate level complexities of the two cell sorter and the complexity when used in both parallel and pipelined architecture is discussed and tabulated. For a basic two cell sorter of existing Carry select logic we need 24 AND gate, 15 OR gate, 9 inverters and 35 EXOR gates. Totally we need 91 gates. In proposed BLAC we need 16 AND gates, 8 OR gate, 16 inverters and 40 EXOR gates. Totally we need 88 gates. For the parallel architecture of existing Carry select logic we need 1,911 gates. In proposed BLAC we need 1,848 gates only. For pipelined parallel architecture of existing Carry select logic we need 2,100 gates. In proposed BLAC

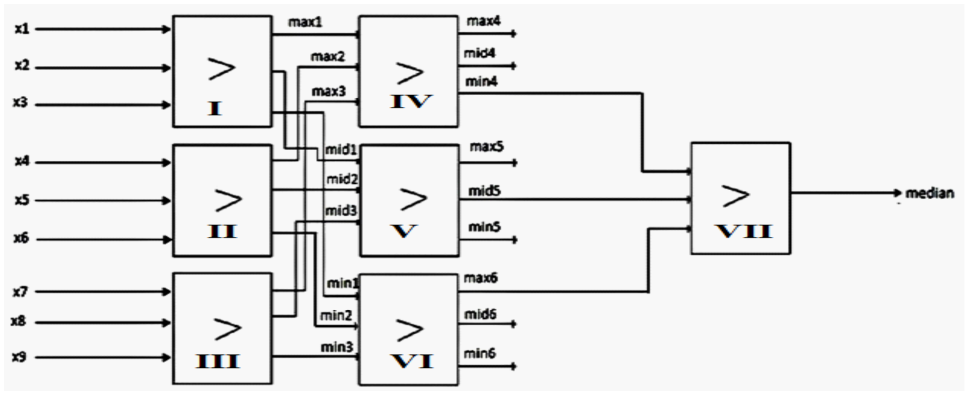


Figure 4. Illustration of the Parallel architecture for modified shear sorting.

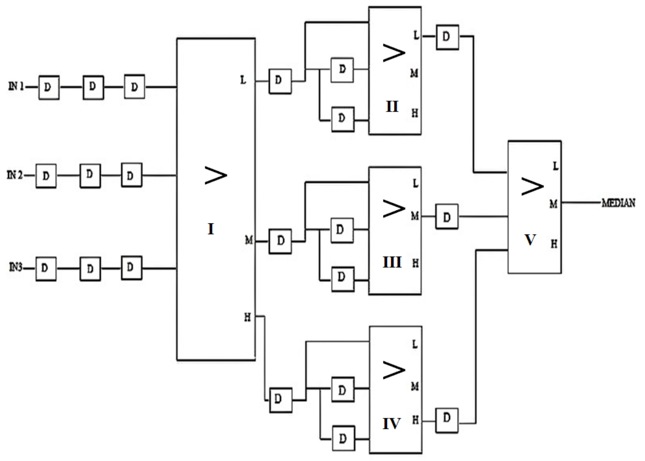


Figure 5. Illustration of the Pipelined Parallel architecture for modified shear sorting.

we need 2,037 gates. In all the tables the proposed logic is addressed as look ahead logic. Table 1 gives the gate level complexities of the various architectures. Table 2 illustrates the comparison of various two cell sorters. The proposed architecture is implemented for XC3s5000-5fg900 using Xilinx 7.1 compiler tool for synthesis and model sim 5.8IIi for simulation using VHDL. All the Rank ordering algorithms have been implemented using VHDL for the above targeted device. Table 3 illustrates the comparison of other Rank ordering techniques with the proposed logic on the basis of area, speed, power. Figure 6 gives the utilization of number of slices for various algorithms. Figure 7 gives the utilization of 4 input look up table by various algorithms. Figure 8 illustrates the gate count of various algorithms. Figure 9 denotes the Maximum combinational delay for

Table 1. Gate level complexity

S.No	PARAMETERS	BASIC TWO CELL SORTER			PRALLEL ARCHITECTURE			PIPELINED ARCHITECTURE		
		CL	CSL	BLAC	CL	CSL	BLAC	CL	CSL	BLAC
1.	XOR	-	35	40	-	735	840	-	735	840
2.	AND	16	24	16	336	504	336	336	504	336
3.	OR	8	15	8	168	315	168	168	315	168
4.	INVERTER	24	9	16	504	189	336	504	189	336
5.	MUX	-	8	8	-	168	168	-	168	168
6.	REGISTERS	-	-	-	-	-	-	189	189	189

Table 2. Various two cell sorter for the targeted device XC3s400tq144-5

PARAMETERS	COVENTIONAL LOGIC	CARRY SELECT LOGIC	BLAC
AFTER SYNTHESIS			
Number of Slices	13	13	15
Number of 4 input LUTs	24	23	26
Number of bonded IOBs	32	32	34
Maximum combinational path delay	10.708 ns	13.638 ns	16.466 ns
AFTER MAPPING			
Slices occupied	12	12	13
Number of 4 input LUTs	24	23	26
Number of bonded IOBs	32	32	34
Gate count for design	168	141	165
AFTER PLACE AND ROUTE			
External Iob's	32	32	34
Number of Slices	12	12	13

Table 3. Various sorting algorithms for the targeted device XC3s5000-4fg900

S.no	PARAMETERS	BUBLE SORT	HEAP SORT	INSERTION SORT	MDF	SELECTION SORT	TDF	PA Parallel
	AFTER SYNTHESIS							
1	NO OF SLICES	4375	3810	4375	4021	4375	4132	252
2	NO OF 4 I/P LUT	6080	5312	6080	6854	6080	7066	439
3	BONDED IOB	328	321	321	82	321	82	80
4	MAX COMB DELAY PATHS (ns)	151.715	327.557	151.715	188.933	151.71	190.94	109.13
	AFTER MAPPING							
5	NO OF 4 I/P LUT	6080	5,312	6080	6,922	6,080	7,139	439
6	BONDED IOB	328	321	321	82	321	82	80
7	GATE COUNT	43075	37,699	43,075	42,055	43,075	43,927	2,634
	AFTER PLACE AND ROUTE							
11	EXTERNAL IOB	321	321	321	82	321	82	80
12	SLICES FLIP FLOP	3088	2783	3088	3689	3088	3800	227
13	EXTERNAL IOB	321	321	321	82	321	82	80
	POWER CONSUMED							
14	POWER CONSUMPTION	298	100	100	298	100	298	100

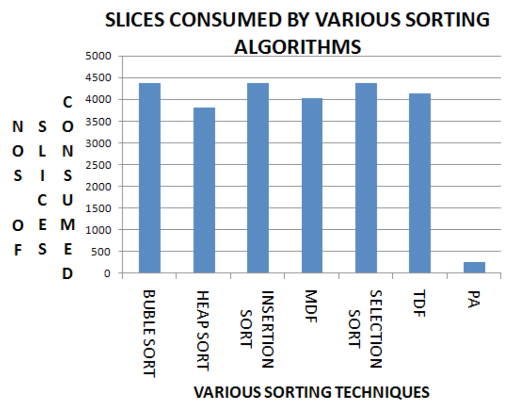


Figure 6. Various architecture vs number of slices.

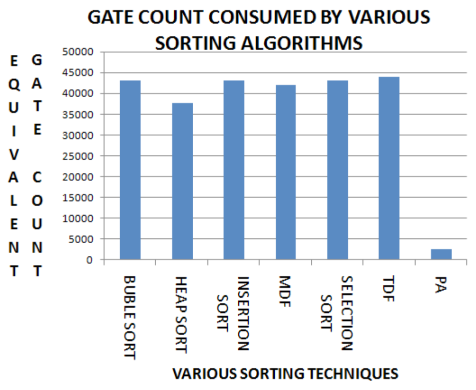


Figure 8. Various architecture vs equivalent gate count.

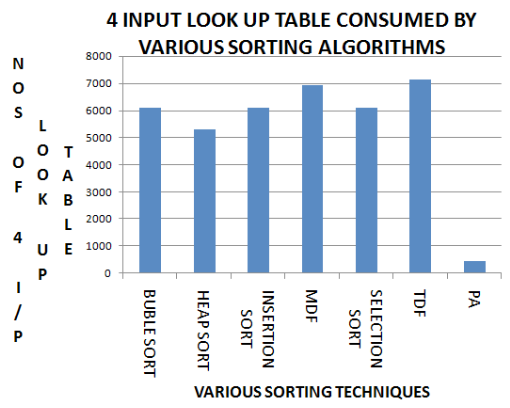


Figure 7. Various architecture VS number of 4i/p Lut’s.

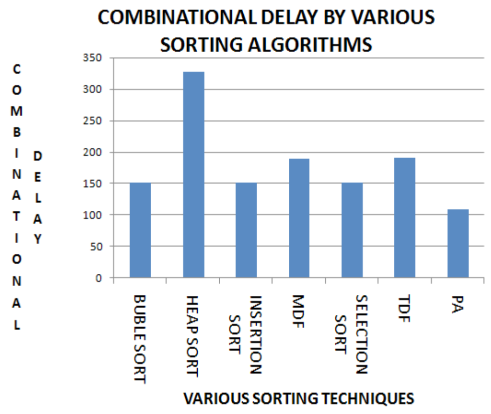


Figure 9. Various architecture vs combinational delay.

various algorithms after place and route. Figure 10 implies the power used in each algorithms. Figure 11–13 illustrates the simulation results, floor plan and routed FPGA for the parallel Modified Shear architecture. Figures 14 and 15 gives the simulation results, floor plan and routed FPGA for the pipelined Modified Shear Sorting. Table 2 gives the simulation result of proposed two cell, three cell algorithms. Table 5 gives the simulation results of pipelined architecture.

The Proposed Architectures are compared with the existing different architectures for modified shear sorting and its pipelined version, implemented using conventional 8 bit comparator, carry select comparator with the proposed BLAC respectively. The proposed architectures

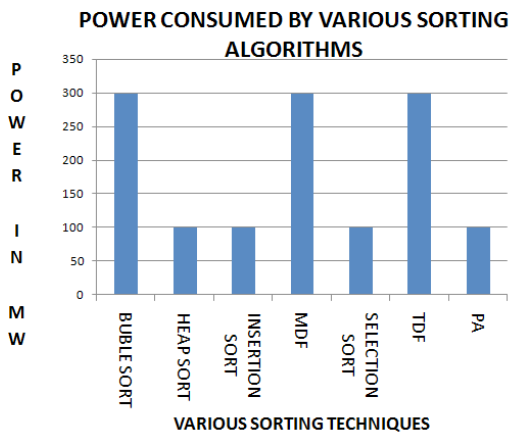
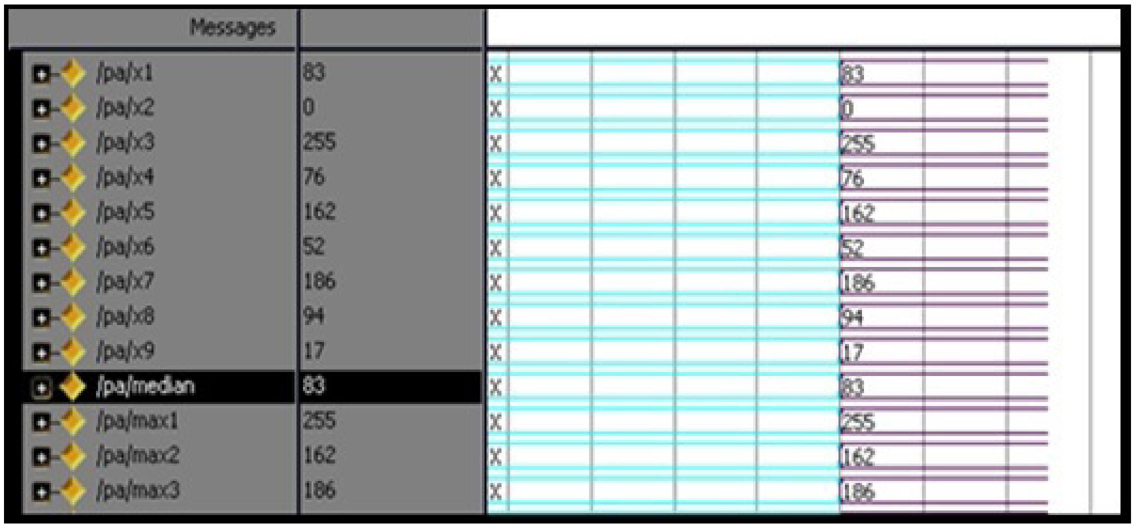


Figure 10. Various architecture vs power.



Messages		
/pa/x1	83	83
/pa/x2	0	0
/pa/x3	255	255
/pa/x4	76	76
/pa/x5	162	162
/pa/x6	52	52
/pa/x7	186	186
/pa/x8	94	94
/pa/x9	17	17
/pa/median	83	83
/pa/max1	255	255
/pa/max2	162	162
/pa/max3	186	186

Figure 11. Simulation results for the parallel architecture (using BLAC).

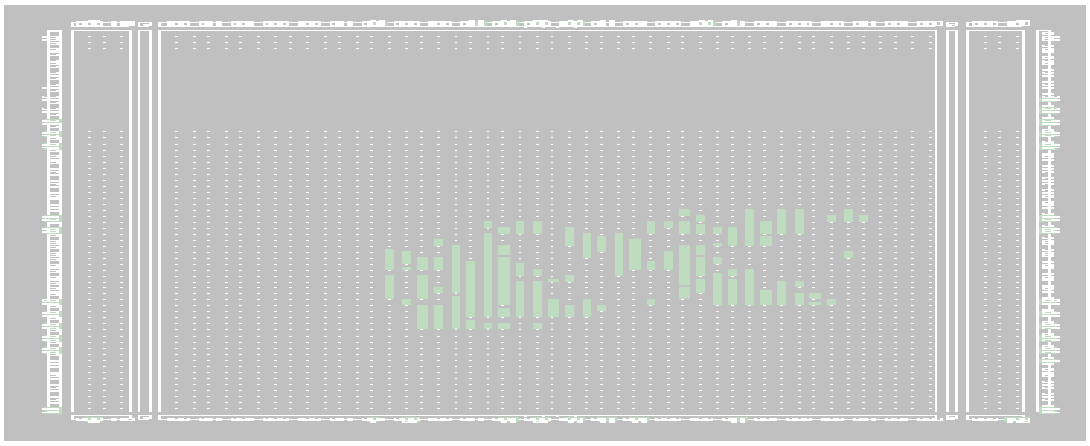


Figure 12. Floor plan of the parallel architecture (using BLAC).

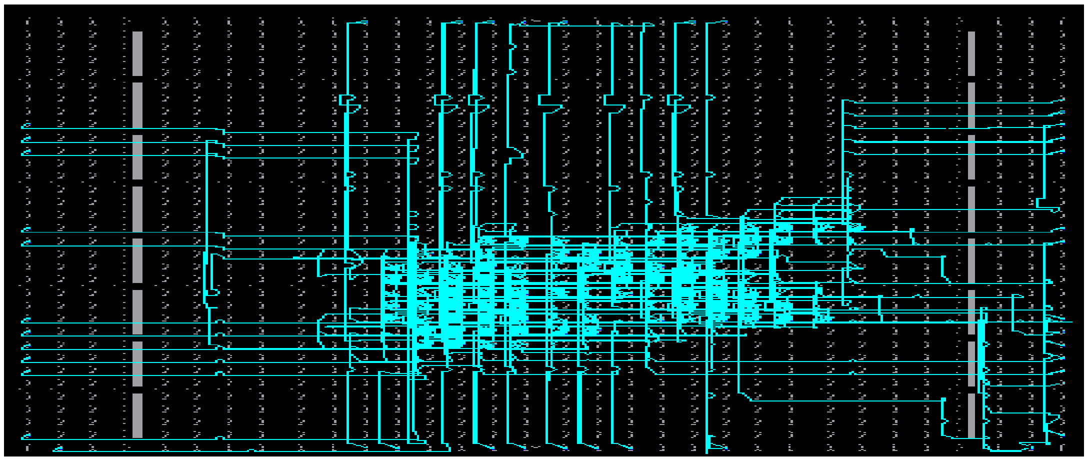
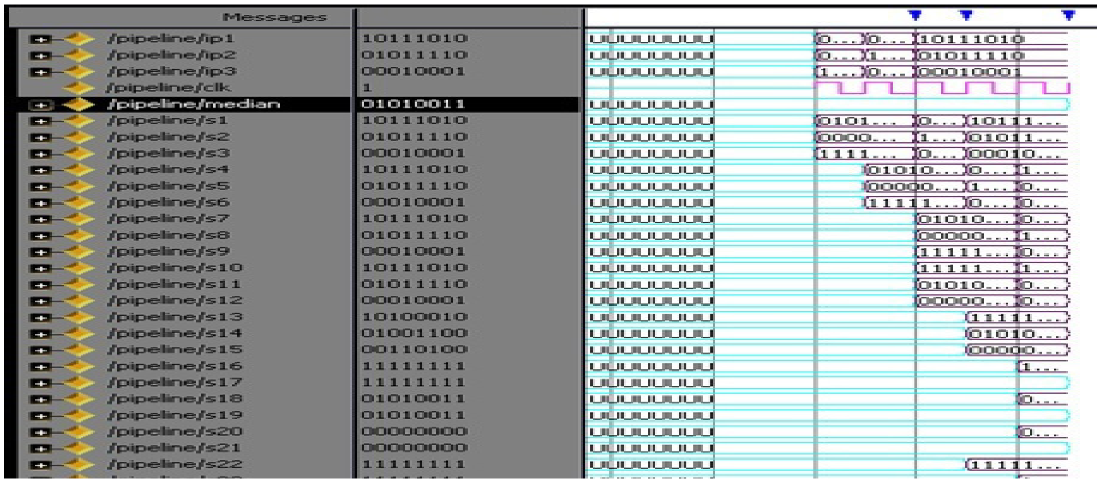


Figure 13. Routed FPGA of the parallel architecture (using BLAC).



```
Messages
/pipeline/ip1      10111010
/pipeline/ip2      01011110
/pipeline/ip3      00010001
/pipeline/clk      1
/pipeline/median   01010011
/pipeline/s1       10111010
/pipeline/s2       01011110
/pipeline/s3       00010001
/pipeline/s4       10111010
/pipeline/s5       01011110
/pipeline/s6       00010001
/pipeline/s7       10111010
/pipeline/s8       01011110
/pipeline/s9       00010001
/pipeline/s10      10111010
/pipeline/s11      01011110
/pipeline/s12      00010001
/pipeline/s13      10100010
/pipeline/s14      01001100
/pipeline/s15      00110100
/pipeline/s16      11111111
/pipeline/s17      11111111
/pipeline/s18      01010011
/pipeline/s19      01010011
/pipeline/s20      00000000
/pipeline/s21      00000000
/pipeline/s22      11111111
```

Figure 14. Simulation results for the pipelined parallel architecture (using BLAC).

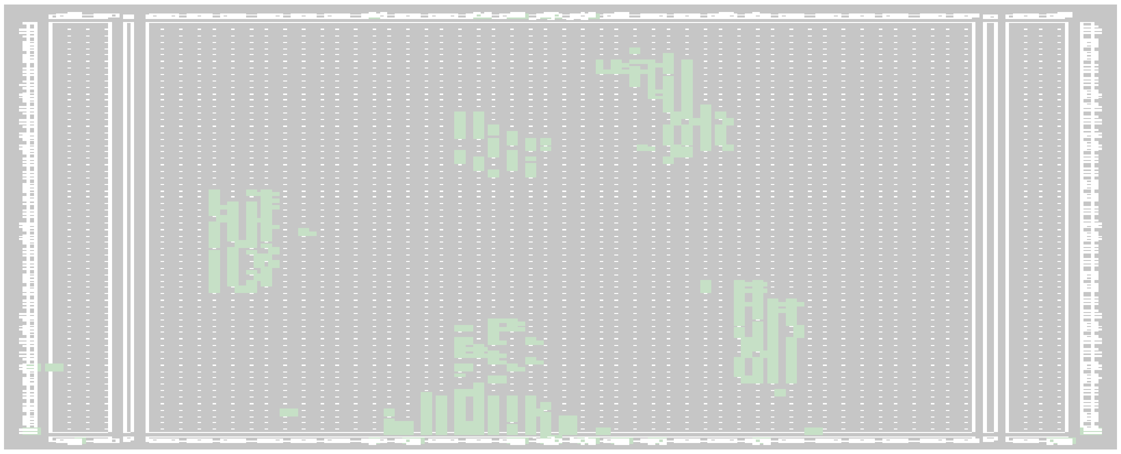


Figure 15. Floor Plan of the pipelined parallel architecture (using BLAC).

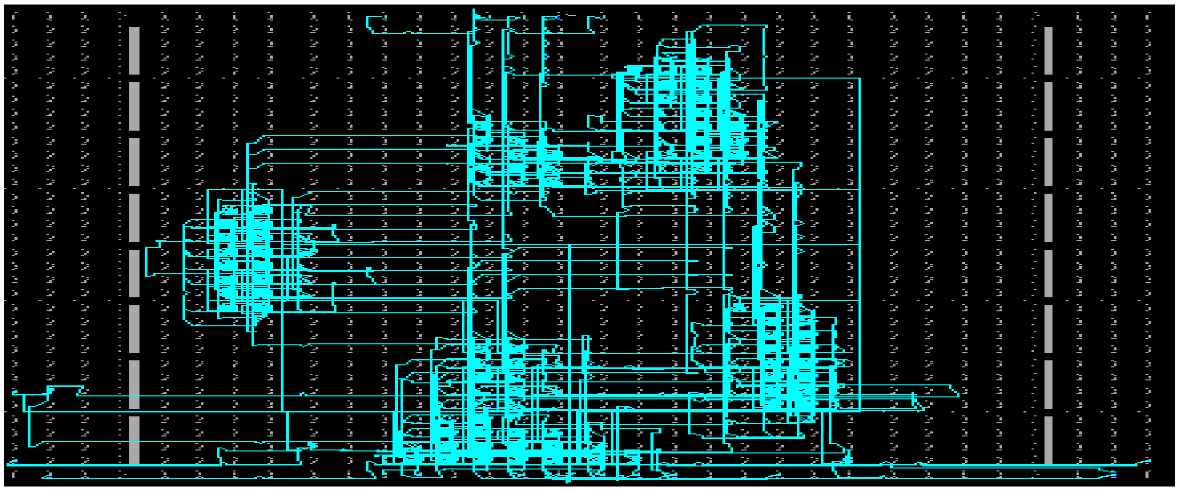


Figure 16. Routed FPGA of the pipelined parallel architecture (using BLAC).

are developed using VHDL in Xilinx Project manager environment for the targeted devices XC3s5000-4fg900 and XC3s400tq144-5. The simulations are carried out using Model sim 5.8IIe and the synthesis tool is XST which is available in Xilinx 7.1i FPGA design suite. The architectures are targeted for Spartan 3e family of device having 5000 gate capacity and 400 gate capacity (Devices XC3s5000–4fg900 and XC3s400tq144-5). The Simulation and synthesis are carried out on an i3 processor with 2.2 GHz with 3 GB RAM.

Table 1 illustrates the gate level complexities of the BLAC, and its position as a processing element in parallel and pipelined architecture of modified shear sorting architectures. From Table 2 we infer that the two cell sorter has a on par area usage and low speed when compared to conventional 8 bit comparator (Conventional logic) and comparator that uses carry select logic (Carry select logic). It is vivid from Table 3 that the proposed parallel architecture has a low area, high speed and low power architecture when compared to conventional 1D sorting algorithms. It was also noted that the proposed BLAC logic on modified parallel shear sorting requires one tenth of the gate count when compared with other 1D algorithms. It is also evident from Figures 6–10 that the parallel architecture occupies less number of slices, 4 input look up table, very meagre gate count, very less combinational delay and low power. This indicates that the proposed parallel architecture consumes less area, works at high speed and consumes low power for the targeted device XC3s5000-4fg900. From the Table 4 the performance of the proposed parallel architecture is compared with Conventional logic and look ahead logic (BLAC). It is observed that the parallel architecture has a par area, speed and power performance when compared with existing logic. The Table 5 tabulates the performance of parallel pipelined architecture using BLAC and other algorithms. When compared to carry select logic, the proposed pipelined parallel architecture has a low combination delay, lower gate count and occupies less number of slices. Figures 11–13 gives simulation result, floor plan and routed FPGA of the parallel architecture. Figures 14–16 gives simulation result, floor plan and routed FPGA of the parallel pipelined architecture. It is visually vivid that the proposed algorithms occupy less area and routed in FPGA.

7. Conclusions

The Paper proposes two architectures which are on par with the standard 8 bit comparator and carry select comparator. The performance of parallel architecture for modified shear sorting median architecture occupies one tenth of the area occupied by other 1D algorithms. The delay offered by the parallel architecture is found to be less and consumes less power. When targeted for XC3s5000-4fg900 for FPGA, the architecture consumed 252 slices of the logic, having a combinational delay of 109.13ns and consuming a power of 100mw. When targeted for XC$_3$s400tq144-5 FPGA, Parallel pipelined version of the parallel architecture the architecture

Table 4. Comparision of different parallel architecture for modified shear sorting for the targeted device XC3s400tq144–5

PARAMETERS	COVENTIONAL LOGIC	CARRY SELECT LOGIC	BLAC
AFTER SYNTHESIS			
Number of Slices	226	216	250
Number of 4 input LUTs	411	378	442
Number of bonded IOBs	80	48	80
Maximum combinational path delay	47.352 ns	82.496 ns	94.664 ns
AFTER MAPPING			
Slices occupied	211	194	231
Number of 4 input LUTs	411	378	442
Number of bonded IOBs	80	80	80
Gate count for design	2,964	2,301	2,739
AFTER PLACE AND ROUTE			
External Iob's	80	80	80
Number of Slices	211	194	231

Table 5. Various pipelined architecture for modified shear sorting for the targeted device XC3s400tq144-5

PARAMETERS	COVENTIONAL LOGIC	CARRY SELECT LOGIC	BLAC
AFTER SYNTHESIS			
Number of Slices	172	371	283
Number of Slice Flip Flops	120	120	120
Number of 4 input LUTs	318	696	530
Number of bonded IOBs	33	33	33
Number of GCLKs	1	1	1
Minimum period	14.727 ns	10.656 ns	12.346 ns
Maximum Frequency	67.902 MHz	93.845 MHz	81.000 MHz
AFTER MAPPING			
Slices occupied	120	120	120
Number of 4 input LUTs	672	294	560
Number of occupied Slices	377	187	298
Number of bonded IOBs	33	33	33
Gate count for design	6,579	4,629	5,640
AFTER PLACE AND ROUTE			
External Iob's	33	33	33
Number of Slices	377	187	298

consumes 282 slices of area and operates at 81 MHz frequency and consumes less power. The pipelined architecture is found to operate faster when compared to conventional 8 bit pipelined comparator. The parallel pipelined comparator also occupies less area when compared to existing carry select logic. Hence a low area, high speed parallel pipelined architecture is proposed.

8. References

1. Oflazer SK. Design and implementation of a simple chip ID median filter. IEEE Trans on Acoustics, Speech and Signal Processing, ASSP-30. 1983; 37(12): 1164–68.
2. Fisher A. Systolic Algorithm for running order statistics in signal and image. Digital System. 1982; 6(14):251–64.
3. Hwang JN. Systolic architecture for 2-D rank order filtering. Proceedings of the International Conference on Application Specific Array Processors. 1990 Sep 5–7; Princeton, NJ. p. 90–99.
4. Kung SY. VLSI Array Processor. Conf. on Systolic Arrays; 1989 May; San Diego. Prentice Hall.
5. Karaman M, Onural L, Atalar A. Design and implementation of general purpose median filter in VLSI. International Conference on Signal Processing and Communications. 1988 Mar; p. 111–19.
6. Lucke L, Parli K. Parallel Structures for rank order & stack filters. 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP-92; 1992 Mar 23–26; San Francisco, CA.
7. Richards DS.VLSI Median Filters. IEEE Trans. on Acoustics, Speech; 1982. p. 251–264.
8. Lee CL, Jen CW. Bit-sliced median filter design based on a majority gate. IEE Proceedings G Circuits, Devices and Systems. 1992 Feb; 139(1):63–71.
9. Offen J, Raymond R.VLSI Image Processing. McGraw-hill; 1985.
10. Fisher AL. Systolic algorithms for running order statistics. Signal and image processing, Dept. of Computer Science, Paper 2396; Carnegie Mellon University; 1981.
11. Kung HT. Why Systolic Architectures. IEEE Computer. 1982 Jan; 15(1):37–46.
12. Batcher KE. Sorting Network and their applications. AFIPS '68 (Spring) Proceedings of the April 30–May 2, 1968, spring joint computer conference. 1968; p. 307–314.
13. Stone HS. Parallel processing with the perfect shuffle. IEEE Trans Comput. 1971 Feb; C-20(2):153–61.
14. Preparata FP. New parallel sorting schemes. IEEE Trans Comput. 1978; C-27(7):669–73.
15. Horiguchi S, Sheigei Y. Parallel sorting algorithm for a linearly connected multiprocessor system. Proc. Int'l Conf. Distributed Computing Systems ICDCS. 1986; IEEE Computer Society. p. 111–18.
16. Thompson CD, Kung HT. Sorting on a mesh-connected parallel computer. Comm. ACM. 1981; C-27(1):151–61.
17. Nassimi D, Sahni S. Bitonics sort on a mesh-connected parallel computer. IEEE Trans. Comput. 1979 Jan; 28(1):2–7.
18. Vasanth K, Karthik S. FPGA implementation of modified decomposition filters. International Conference on Signal and Image Processing. 2010 Dec 15–17; Chennai, India. p. 526–30.
19. Vasanth K, Karthik S. Unsymmetrical trimmed median as detectors for salt and pepper noise removal. National Conference on signal and image processing- NCSIP2012, Gandhi gram rural university. 2012. p. 31-35.
20. Srinivasan KS, Ebenezer D. A new fast and efficient decision- based algorithm for removal of high-density impulse noises. IEEE Signal Processing Letter. 2007; 14(3):189–92.
21. Srinivasan KS, Ebenezer D. VLSI implementation of decision based median filter using new efficient shear sorting. ICACS. 2007; 115–20.