VLSI Architecture for 9 Element Optimized Sorting Network Using 25 comparator for Image De-noising

E. Sindhu , K .Vasanth

Abstract*: The paper introduces a parallel VLSI architecture for Codish sorting. The Codish sorting used generate and prune method to optimize the sorting network for 9 elements using 25 comparators. The prune method optimize the location of each comparator thus makes it an optimized solution. The performance of the proposed architecture was compared with different sorting techniques targeted for XCV1000-5bg560. It was found that the proposed architecture for Codish sorting consumes 402 slices, 4200 gate count, operates at 66.46 ns delay and consumes 7mw of power. The proposed VLSI architecture surpasses all standard sorting algorithms in terms of area, speed and power.

***Index Terms*: Optimized Sorting network, 9-cell sorting, Rank Ordering, Parallel Architecture.**

I. INTRODUCTION

H Sorting networks have been broadly exploring from the past few decades. Sorting operation is used in many applications like signal processing, database management and image processing etc. The importance of sorting networks has been increasing rapidly. Hence the present technology has made it conceivable to straightforwardly actualize parallel algorithms in hardware equipment. Well-organized algorithm structured chips are turning into the essential building blocks in the new generation of high-performance computing systems. This Rank ordering technique is used in different applications of image processing such as de-noising, im-painting, classifications etc. Hence a sorting network is id desired to perform the specified task. The major huddle of sorting networks is to reduce both storage space and sorting time while arranging the data in monotonically increasing order. The basic component necessary for the arrangement of data in increasing order in this sorting network is comparator. Comparator is a very useful circuit for sorting the two elements, to do this each and every element should follow the basic operation that is comparing and swap the element by using an 8-bit comparator and placed it in respective order. Let's look at the existing methods on the proposed work. Codish et al [1] developed an optimized sorting network in terms of size (i.e. minimum no of comparators) for sorting the data of nine inputs by using twenty-five comparators. The proposed paper presents the SAT encoding and generates and prune approach for proving the problem of optimal size but the SAT encoding method can able to reproduce results up to n<=6. The disadvantage of SAT encoding is it fails to scale n=9 elements. So finally describes the combination of both generate and prune with SAT encoding for solving the case of sorting the nine inputs and also the ability to show the potential to scale towards n=9. The advantage of this generate and prune approach is to remove redundancy of comparator in parallel sorting. Bundala et al [2] introduced how to solve and prove the problem of optimal depth of upper bounds of previously defined sorting networks for inputs case of 11<=n<=16. The proposed presents the combination of two methods called symmetry breaking and Boolean satisfiability and explored the algorithm called SAT solver of exponential size in the number of channels. The main advantage of the proposed algorithm is it needs a few seconds for proving the sorting network optimality cases of n<=10 inputs. Campobello et al [3] explored the extension of previously known sorting network work based on the complexity of minimum-maximum circuits. The proposed paper gave the time complexity and spatial complexity based on design choice and also explored the design of high-speed serial and pipelined sorting networks with FPGA technology. Batcher [4] developed a high-speed sorting network it can be used as multiple inputs and multiple outputs switching purpose. The advantage of this sorting network is it requires less hardware compared to a normal crossbar switch, constant fan-in, and fan-out and lower cost. It is used to work large sets of data rapidly. The main applications of the sorting network are multi-access memory, multi-access content-addressable memory, and switching network with buffering and conflict resolution. Qian and Xu [5] introduced a parallel method that can be used for optimization of the sorting algorithms using multi-core and multi-thread. The main advantage of the proposed paper is it explored the design with improvement in the efficiency of sorting algorithms for sorting the large set of data. Savina and Kaur [6] gave the contrast of bubble sort and binary sort and finally combine them in order to achieve improvement in performance it mainly focuses on the time factor.

Revised Manuscript Received on April 08, 2019.

E.Sindhu, PG Scholar, Department of VLSI System Design, Vidya Jyothi Institute of Technology, Hyderabad, Telanagana, India.

K.Vasanth, Professor, Department of E.C.E, Vidya Jyothi Institute of Technology, Hyderabad, Telanagana, India.

This exploration is based on comparing and merges of an algorithm that can fit into every single record of arranging and inquires the information properly. Additionally, this work contrasts the past and proposed algorithms. Codish et al [7] developed a straightforward sorting network implementation which will have the advantage of the latest architecture of CPU. The work show various sorting networks have an effect on the efficiency of synthesizing C code. Finally, it evaluates a significant measure should take towards a small size, parallelism and block structure that can reduce register spilling. Najafi et al [8] presented an approach to sorting networks with area efficient and power using unary processing design which is accurate and deterministic unlike stochastic logic and also comparison had done with the conventional design. The proposed paper used a novel time encoding to improve the latency. The advantage of this approach is efficient energy and lower cost compared to conventional design implementation but with a small effect on accuracy. Valsalam and Miikkulainen [9] defined a two-pronged method referred as SENSO (symmetry and evolution based network sort optimization) which is used for the purpose of finding a solution on search space for sorting networks that can sort a large set of data inputs than previously known networks. The advantage is the reduction of comparators need for every step individually. Feng Shi et al [10] explored a new design which is the generalization of basic blocks of an odd-even sort called merge sorting algorithm depends on n-sorters which are used for parallel sorting networks. The proposed algorithm has an advantage of less number of gates and slighter latency. Codish et al [11] gave the new properties of sorting networks to the front end and back end and presents the use of these networks in the search for new bounds. The proposed paper describes a parallel sorting network for 17 to 20 inputs which are quicker than known previous networks. The main advantage is that no other existing sorting network requires fewer layers for 17 inputs. Codish et al [12] illustrates the study of optimal sorting network capable of generation in all two layer prefixes and shows improvement of symmetry breaking, the new method developed based on languages and graph isomorphism gave the representation of non-isometric. The proposed design overcome the generate and test approach problem i.e. applied do not scale by demonstrating new technique ease scales up to n=40. Verma and Chowdhary [13] introduced a sorting algorithm on smart phones and describes an approach for the minimizing the energy consumption through software with a proper selection of sorting method. This paper presents an approach to saving energy through software by choosing the appropriate sorting algorithm. Ramırez1et al [14] describes the implementation of sorting network for a small set of data as input for a microcontroller for the operation of ordering the data in sensors in the greenhouse. The advantage of this operation is to check whether actuators working each factor with iteratively or not. Faujdar and Ghrera [15] describes the analyzing and testing various Sorting Algorithms and to evaluate the total space complexity on a Standard Dataset and also presents the performance evaluation for each and every case of a network. The paper proposes Parallel VLSI architecture for optimized sorting networks for 9 elements developed by codish [1]. As specified a low area and high speed sorting technique is desired for many VLSI implementation of image processing application. Section 2 gives the proposed sorting network. Section 3 illustrates the parallel architecture of the proposed sorting. Section 4 gives discussions and simulation results and finally table comparing 25-cell sorting technique with different architectures of comparator. Section 5 gives the conclusion of the work.

II. PROPOSED SORTING ALGORITHM

The Sorting technique is used in numerous applications for rank ordering image data. One simple application is evaluation of Median to remove salt and pepper noise. The basic operation of median filtering is rank ordering. We use an optimized sorting network developed by Codish [1] using generate and prune approach. The Codish sorting is interpreted on a 3x3 neighborhood window to compute rank ordering on an 8 bit image as shown in image 1. With the advancement in VLSI technology, a parallel architecture for 9 elements is proposed for targeted FPGA. A faster sorting network with reduced area consumption is always desired. The proposed algorithm will be addressed Codish sorting [1] hereafter.

There are totally 9 elements present in a 3x3 window. The nomenclature of the 3x3 window is shown in figure 1. The center element is the element that is processed using the neighborhood values. The methodology of this Codish sorting is implemented on a 3x3 window using compare and swap operation to arrange the input data in increasing order. The rectangular box and circle indicates comparison of two elements in increasing order and replaced first value with minimum (min) and second with maximum (max) values respectively. The sorting requires 25 steps to arrange data in ascending order. The operation of codish sorting is shown in figure 2.

Step 1. Compare North-west (NW) element with the North (N) element and replace min value in the north-west (NW) position and max value in the north (N) position. **Step 2.** Compare North-east (NE) element with the West (W) element and replace min value in the north-east (NE) position and max value in the west (W) position. **Step 3.** Compare Center Element (CE) element with the East (E) position and replace min value in the Center Element (CE) position and max value in the east (E) position. **Step 4.** Compare South-west (SW) element with the South (S) element and replace min value in the south-west (SW) position and max value in the south (S) position. **Step 5.** Compare North (N) element with the West (W) element and replace min value in the north (N) position and max value in the west (W) position. **Step 6.** Compare East (E) element with the South (S) element and replace min value in the east (E) position and max value in the south (S) position.

Step 7. Compare North (N) element with the East (E) element and replace min value in the north (N) position and max value in the East (E)

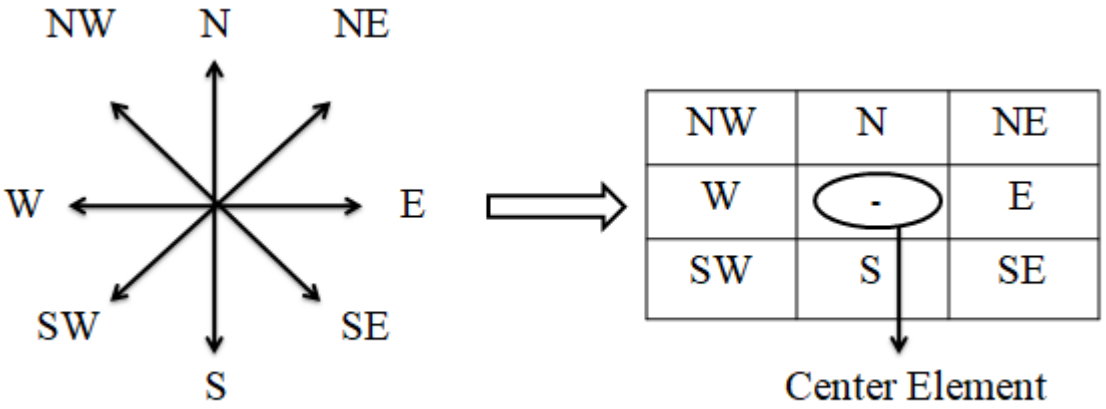

Figure1 Basic concept of 9-cell Sorting network

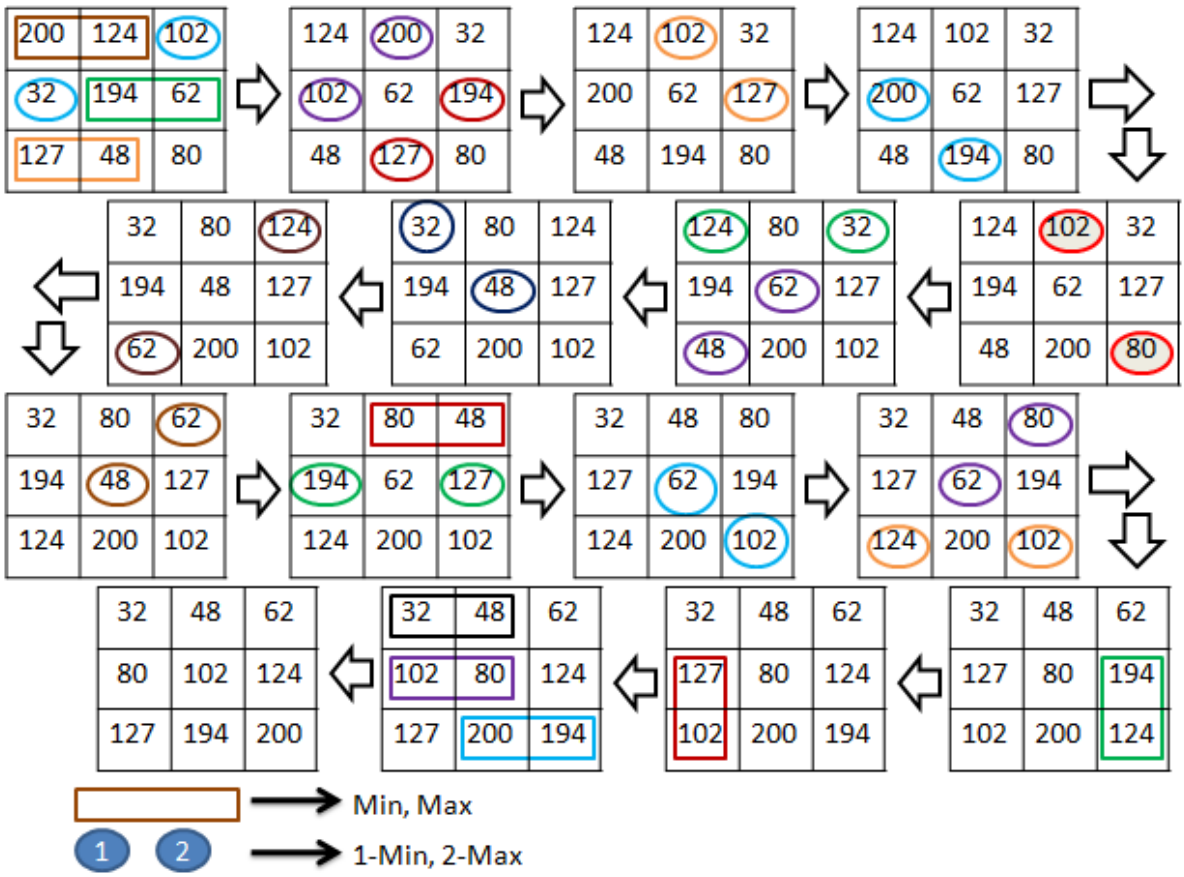

Figure2: Working of 25-cell Sorting network

position. **Step 8.** Compare West (W) element with the South (S) element and replace min value in the west (W) position and max value in the south (S) position. **Step 9.** Compare North (N) element with the South-East (SE) element and replace min value in the north (N) position and max value in the South-East (SE) position. **Step 10.** Compare North-west (NW) element with the North-East (NE) element and replace min value in the north-west (NW) position and max value in the North-East (NE) position. **Step 11.** Compare Center Element (CE) element with the South-west (SW) position and replace min value in the Center Element (CE) position and max value in the South-west (SW) position. **Step 12.** Compare North-west (NW) element with the Center Element (CE) element and replace min value in the north-west (NW) position and max value in the Center Element (CE) position. **Step 13.** Compare North-east (NE) element with the South-west (SW) element and replace min value in the north-east (NE) position and max value in the South-west (SW) position. **Step 14.** Compare North-east (NE) element with the Center Element (CE) element and replace min value in the north-east (NE) position and max value in the Center Element (CE) position.

Step 15. Compare North (N) element with the North-East (NE) element and replace min value in the north (N) position and max value in the North-East (NE) position. **Step 16.** Compare West (W) element with the East (E) element and replace min value in the west (W) position and max value in the East (E) position. **Step 17.** Compare Center Element (CE) element with the South-East (SE) position and replace min value in the Center Element (CE) position and max value in the South-East (SE) position. **Step 18.** Compare North-east (NE) element with the Center Element (CE) element and replace min value in the north-east (NE) position and max value in the Center Element (CE) position. **Step 19.** Compare South-west (SW) element with the South-East (SE) element and replace min value in the south-west (SW) position and max value in the South-East (SE) position. **Step 20.** Compare East (E) element with the South-East (SE) element and replace min value in the east (E) position and max value in the South-East (SE) position. **Step 21.** Compare West (W) element with the South-west (SW) element and replace min value in the west (W) position and max value in the South-west (SW) position. **Step 22.** Compare North-west (NW) element with the North (N) element and replace min value in the north-west (NW) position and max value in the north (N) position. **Step 23.** Compare West (W) element with the Center Element (CE) element and replace min value in the west (W) position and max value in the Center Element (CE) position. **Step 24.** Compare East (E) element with the South-west (SW) element and replace min value in the east (E) position and max value in the South-west (SW) position. **Step 25.** Finally compare South (S) element with the South-East (SE) element and replace min value in the South (S) position and max value in the South-East (SE) position. Hence step1 to 25 gives a 3*3 sorted window as shown in figure2.

III PARALLEL ARCHITECTURE FOR 9 CELL CODISH SORTING

The proposed Codish sorting is implemented using a Parallel VLSI architecture using compare and swap approach. The proposed parallel VLSI architecture is shown in figure 3. The figure 3 illustrates a nine input sorting networks consisting of twenty-five comparators and describes sorting networks on 9 channels, each comprising of individual comparator for arranging the data in increasing order. The even lines i.e. horizontal lines are indicated as channels, whereas vertical lines i.e. rectangular boxes are demonstrated as comparators connecting a pair of channels and input values are implicit to transmit from left to right as shown in the figure. The grouping of comparators related with a pictorial representation is acquired by a left-to-right, top-down respectively. These are the steps while processing a 9 cell Codish sorting using two-cell sorter or comparator is explained below. **Step1:** All the input values are fed to nine channels then followed by comparator networks i.e. sorting network is based on compare and probably exchange pairs of inputs finally gives outputs as maximum(High) and minimum(Low) values respectively. **Step2:** Comparator1 compares x1 and x2 which gives high and low values. Similarly, for Comparator2, Comparator3 and Comparator4 compare x3 and x4, x5 and x6, x7 and x8 which gives high and low values and finally x9 is directly fed to the input of comparator9. **Step3:** The low values of comparator1 and comparator2 is given as input to comparator5 which on comparing and exchange of inputs gives high and low values. **Step4:** The low values of comparator3 and comparator4 is given as input to comparator6 which on comparing and exchange of inputs gives high and low values. **Step5:** The high values of comparator5 and comparator6 is given as input to comparator7 which on comparing and exchange of inputs gives high and low values. **Step6:** The low values of comparator5 and comparator6 is given as input to comparator8 which on comparing and exchange of inputs gives high and low values. **Step7:** The high value of comparator7 and x9 value are directly given as input to comparator9 which on comparing and exchange of inputs gives high and low values. **Step8:** The high values of comparator1 and compartor2 is given as input to comparator10 which on comparing and exchange of inputs gives high and low values. **Step9:** The high values of comparator3 and compartor4 is given as input to comparator11 which on comparing and exchange of inputs gives high and low values. **Step10:** The high values of comparator10 and compartor11 is given as input to comparator12 which on comparing and exchange of inputs gives high and low values. **Step11:** The low values of comparator10 and compartor11 is given as input to comparator13 which on comparing and exchange of inputs gives high and low values. **Step12:** The high value of comparator13 and low value of compartor12 is given as input to comparator14 which on comparing and exchange of inputs gives high and low values. **Step13:** The high values of comparator9 and compartor14 is given as input to comparator15 which on comparing and exchange of inputs gives high and low values. **Step14:** The high value of a comparator8 and low value of compartor7 is given as input to comparator16 which on comparing and exchange of inputs gives high and low values. **Step15:** The low values of comparator14 and compartor9 is given as input to comparator17 which on comparing and exchange of inputs gives high and low values. **Step16:** The low value of comparator15 and high value of compartor17 is given as input to comparator18 which on comparing and exchange of inputs gives high and low values. **Step17:** The low values of comparator13 and compartor17 is given as input to comparator19 which on comparing and exchange of inputs gives high and low values. **Step18:** The low values of comparator16 and compartor19 is given as input to comparator20 which on comparing and exchange of inputs gives high and low values. **Step19:** The high values of comparator16 and compartor19 is given as input to comparator21 which on comparing and exchange of inputs gives high and low values.

Step20: The high values of comparator12 and compartor15 is given as input to comparator22 which on comparing and exchange of inputs gives max and max1 values. **Step21:** The high value of comparator18 which directly gives max2 value. **Step22:** The high value of comparator21 and low value of compartor18 is given as input to comparator23 which on comparing and exchange of

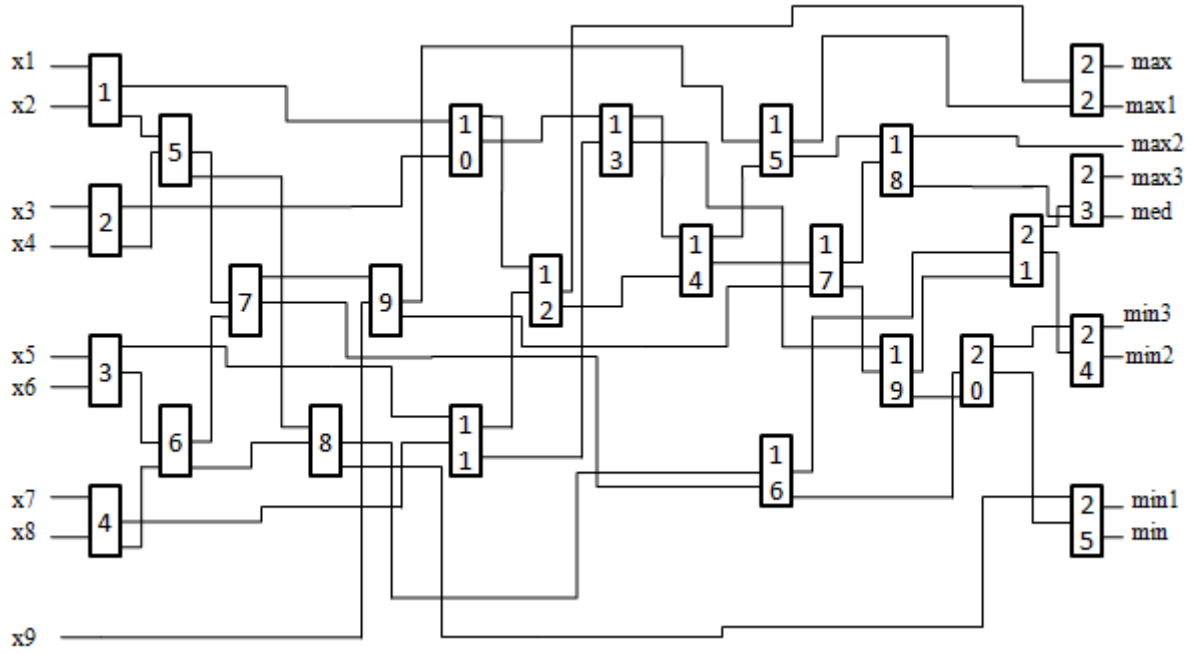

Figure 3 Parallel Architecture for 9-Cell Codish Sorting

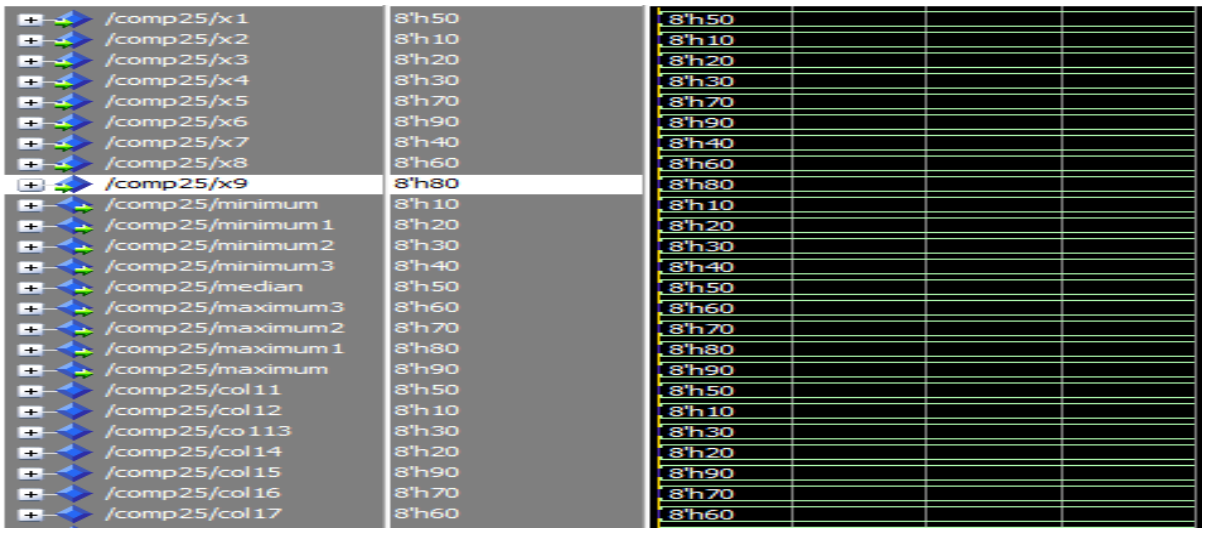

Figure 4 Simulation of 9-cell Codish sorting

inputs gives max3 and med values. **Step23:** The high value of comparator20 and low value of compartor21 is given as input to comparator24 which on comparing and exchange of inputs gives min3 and min2 values. **Step24:** The low values of comparator8 and compartor20 is given as input to comparator25 which on comparing and exchange of inputs gives min1 and min values.

IV SIMULATION RESULTS & DISCUSSIONS

The proposed Parallel VLSI architecture is implemented for XCV1000-5bg560 using Xilinx 7.1 compiler tool for synthesis and Modelsim 10.4a for simulation as a third-party tool using VHDL. Table1 illustrates the comparison of 9-cell Codish sorting algorithm with different sorting techniques based on area, speed, and power. Figure 4 illustrates the simulation results of 9 cell codish sorting using Modelsim 10.4a tool. Table 1 shows the Device utilization summary of different sorting techniques for the targeted FPGA. Figure 5 Illustrates the different parameters I) Number of slices occupied II) Maximum Combinational Delay path III) Gate count IV) Power consumption of different sorting techniques. Figure 6 shows the floor plan of different sorting algorithms on targeted FPGA.

Figure 7 shows the routed FPGA of different sorting algorithms. It is vivid from table 1 the proposed Codish sorting consumes less number of slices, operates with lesser delay and consumes less power when compared to other sorting algorithms. The Proposed Codish sorting consumes 406 slices in comparison with 7208 slices by selection sorting. This indicates that the proposed Codish sorting requires 17% of reduced area when compared to other algorithms. Also, the Proposed VLSI architecture has a gate count of 4200. This reduction in area is mainly due to parallel architecture implementation of the proposed Codish sorting. The Proposed VLSI architecture operates at a lesser combinational delay path delay when compared

Table 1 Performance of 9-cell Codish sorting with standard and existing sorting for the target device XCV1000-5bg560

PARAMETERS		BITONIC	BUBBLE	HEAP	INSERTION	ODD-EVEN TRANSPOSITION	SELECTION	SNAKE1	26-CELL SNAKE	PROPOSED CODISH
SYNTHESIS REPORT	8-BIT COMPARATORS	80	-	-	-	-	-	42	33	25
	NO OF SLICES	1346	6713	6770	6423	946	7208	675	528	406
	SLICES FLIPFLOPS	-	9	9	9	-	9	-	-	-
	NO OF 4 I/P LUT	1920	9711	9620	9765	1344	10481	1008	792	600
	BONDED IOB	256	328	328	328	257	328	144	144	144
	MAX COMBINATIONAL DELAY in ns	81.362	337.765	613.266	483.533	80.231	742.394	113.469	104.894	66.46
MAP REPORT	NO OF 4 I/P LUT	1920	9730	9641	9819	1344	10504	1008	792	600
	NO OF BONDED IOB	256	328	328	328	256	328	144	144	144
	GATE COUNT	13440	72156	71526	70002	9408	77568	7056	5544	4200
	AVG FANOUT OF LUT	2.73	3.06	3.00	2.97	2.62	2.97	2.69	2.60	2.52
	MAX FANOUT OF LUT	6	93	96	98	6	95	6	6	6
	AVG FANIN FOR LUT	3.26	3.29	3.38	3.33	3.23	3.34	3.28	3.27	3.22
PLACE AND ROUTE REPORT	EXTERNAL IOB	256	328	328	328	256	328	144	144	144
	SLICES	1227	6150	6191	5836	857	6584	617	481	373
	POWER CONSUMPTION in mW	7	32	32	32	32	32	32	32	7

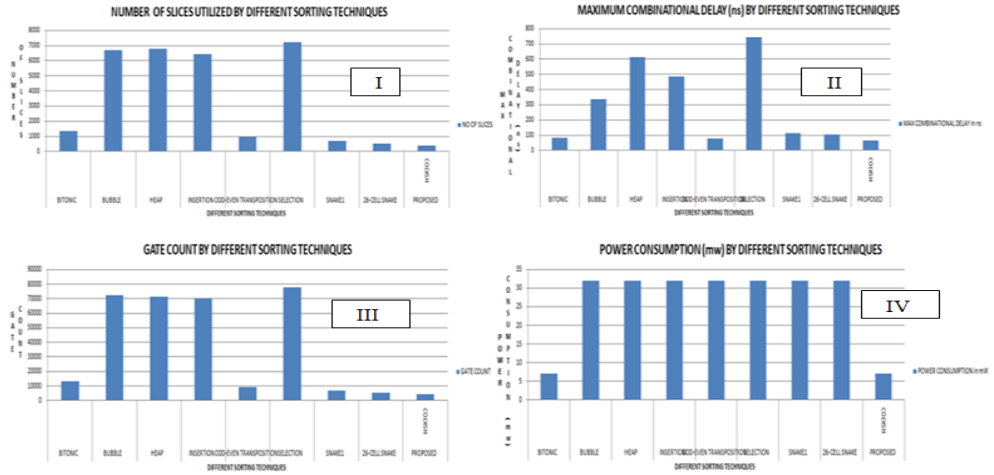

Figure 5 Illustrates I) Number of slices occupied II) Maximum Combinational Delay path III) Gate count IV) Power consumption of different sorting techniques

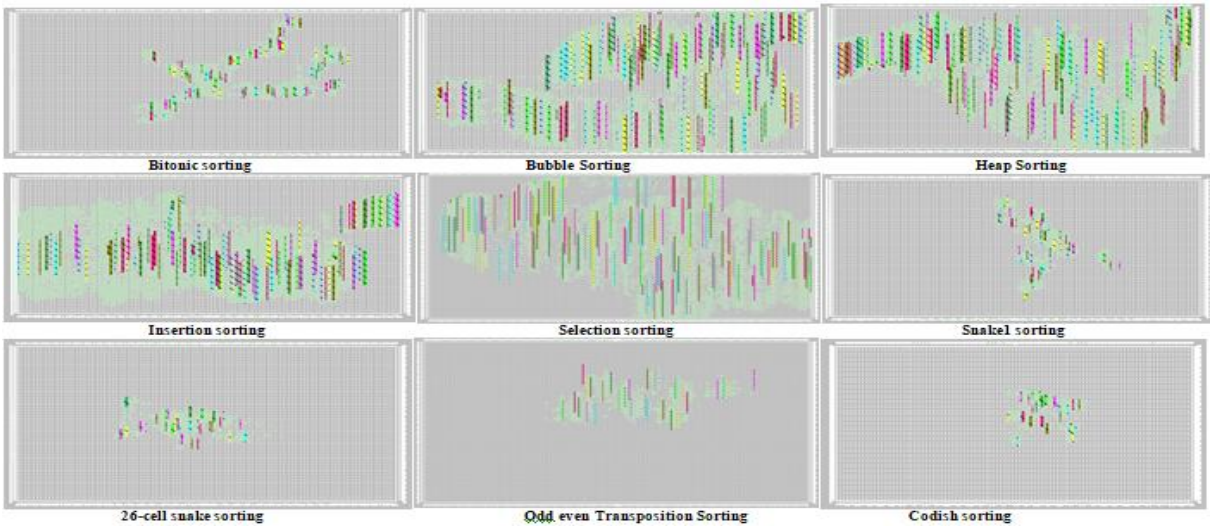

Figure 6 Floor plan of the different sorting Technique on FPGA

to other sorting techniques. The architecture requires a maximum combinational delay of 66.46ns. The proposed Codish sorting is 5 times faster when compared to other conventional algorithms.

The Codish sorting operates at high speed due to its parallel architecture and the ordering of comparators using generate and prune approach.

The Generate and prune approach optimize the location of each comparator in Codish sorting. This makes it an optimized sorting network for 9 element sorting network. The proposed architecture is parallel in nature and hence consumes very less power of 7mw when targeted on FPGA. It is visually evident from figure 6 and 7 that the proposed algorithm requires very less area when compared to other sorting algorithms

V. CONCLUSOION

In this paper, a new parallel VLSI architecture has been proposed for 9-cell Codish sorting for employing the rank ordering of data for image de-noising application. Codish sorting used generate and prune approach to optimize the sorting network for 9 inputs. The proposed VLSI architecture was targeted for the device XCV1000-5bg560. It was found that the proposed algorithm requires 406 slices, 4200 gate count, operates at 66.46ns and consumes 7 mw power. These values were found to be less when compared with other sorting technique. Hence an VLSI architecture for Codish sorting is proposed which requires a less area, operates at high-speed architecture and consumes low power.

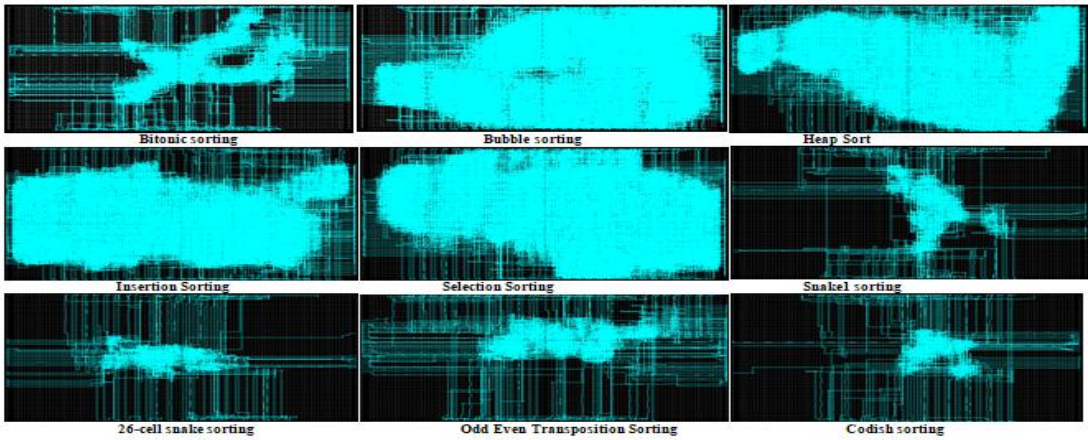

Figure 7 Routed Design of different sorting Technique on FPGA

REFERENCES

1. Michael Codish, Luis Cruz-Filipe, Michael Frank, and Peter Schneider-Kamp, "Twenty-Five Comparators is Optimal when Sorting Nine Inputs (and Twenty-Nine for Ten)", Proceedings of International Conference on Tools with Artificial Intelligence, pp 186-193, May 2014.
2. Daniel Bundala, Michael Codish, Luis Cruz-Filipe, and Peter Schneider-Kamp, "Optimal-depth Sorting Networks", Journal of computer and system sciences, Vol.84, pp 185-204, Sep 2016.
3. Giuseppe Campobello, Giuseppe Patane and Macro Russo, "On the complexity of min-max Sorting Networks", Journal of information sciences, Vol.190, pp 178-191, Dec 2011.
4. K.E. Batcher, "Sorting networks and their applications", Proceedings of Spring Joint Computing Conference of AFIPS, Thomson Book Company, Vol.32, pp 307–314, 1968.
5. Xiao-jie Qian and Jin-bang XU, "Optimization and implementation of sorting algorithm based on multi-core and multi-thread", Proceedings of IEEE International Conference on Communication Software and Networks, pp 29-32, 2011.
6. Savina and Surmeet Kaur, "Study of Sorting Algorithm to Optimize Search Results", International Journal of Emerging Trends and Technology in computer science (IJETTCS), Vol.2, Issue.1, pp 204-207, Jan-Feb 2013.
7. Michael Codish, Luis Cruz-Filipe, Markus Nebel and Peter Schneider-Kamp, "Study, Optimizing sorting algorithms by using sorting networks", Formal Aspects of Computing, Vol.29, Issue.3, pp 559-579, May 2017.
8. M.Hassan Najafi, David J. Lilja, Marc Riedel and Kia Bazargan,"Power and Area Efficient Sorting Networks using Unary Processing", Proceedings of IEEE International Conference on Computer Design, pp 125-128, 2017.
9. Vinod K. Valsalam and Risto Miikkulainen, "Utilizing Symmetry and Evolutionary Search to Minimize Sorting Networks", Journal of Machine Learning Research, Vol.14, pp 303-331, 2013.
10. Feng Shi, Zhiyuan Yan, and Meghanad Wagh, "An Enhanced Multiway Sorting Network Based on n-Sorters", Proceedings of IEEE Global conference on Signal and Information Processing (GlobalSIP), pp 60-64, July 2014.
11. Michael Codish, Luis Cruz-Filipe, Thorsten Ehlers, Mike Muller and Peter Schneider-Kamp, "Sorting Networks: to the End and Back Again", Journal of computer and system sciences, pp 1-20, July 2015.
12. Michael Codish, Luis Cruz-Filip and Peter Schneider-Kamp, "The Quest for Optimal Sorting Networks: Efficient Generation of Two-Layer Prefixes", International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pp 1-8, Sep 2014.
13. Murlidhar Verma and K.R.Chowdhary, "Analysis of Energy Consumption of Sorting Algorithms on Smartphones", Proceedings of International Conference on Advances in Internet of Things and Connected Technologies (ICIOTCT-2018), pp 472-475, 2018.
14. Blanca C Lopez Ramirez, Giovanni Guzman, Wadee Alhalabi, Nareli Cruz-Cortes, Miguel Torres-Ruiz and Marco Moreno-Ibarra, "on the usage of sorting network to control green house climatic factors", International Journal of Distributed Sensor Networks", Vol.14, Issue.2, pp 1-10, 2018.
15. Neetu Faujdar and Satya Prakash Ghrera," Analysis and testing of Sorting algorithm on a standard data set" ,International Conference on Communication Systems and Network Technologies", pp 962-967, 2015.

AUTHORS PROFILE

E Sindhu is a PG scholar in department of VLSI system Design, Vidya Jyothi Institute of Technology, Hyderabad, Telanagana. She did her UG in Electronics and communication. Her research interest include VLSI signal Processing. She had published 2 papers in scopus journal.

K.Vasanth is working as professor in Department of E.C.E, Vidya Jyothi Institute of Technology, Hyderabad. He had completed B.E in ECE from Arulmigu Kalasalingam college of Engineering and M.E and Ph.D from Sathyabama University. He was the technical project lead behind the Design, launch and tracking of "Sathyabamasat – Nano Satellite". He has more than 65 research publications from peer reviewed journals and conference. His research interests include system engineering and non linear signal processing.