JOURNAL OF CRITICAL REVIEWS

ISSN- 2394-5125 VOL 7, ISSUE 15, 2020

IMPLEMENTATION OF SORTING OF ONE-DIMENSIONAL ARRAY USING RAM BASED SORTING ALGORITHM

A.S. Gowtham [1], Dr. D. Jaya Kumar,MTech., Ph.D.[2],

[1]Student, MTech, Department of ECE, Associate Professor, Kuppam Engineering College

[2]Department of ECE, Associate Professor, Kuppam Engineering College

Received: 14 March 2020 Revised and Accepted: 8 July 2020

ABSTRACT: As now a days most of the technology is dependent on the speed of operation and the speed of the Processor. Thecurrent stage in the development of information technology is characterized by the accumulation of large amounts of data. When processing such data arrays, it is necessary to use data sorting and searching operations. The purpose of data arrays sorting is to accelerate the search for the necessary information. The main ways of increasing the speed of sorting operations are the development of new sorting algorithms. This paper main logic involves the RAM based sorting which makes use of the multiple read and write characteristics of random-access memory. To sort one dimensional data and the numbers it performs Sequential search and swap operations simultaneously on numbers. Because of this reason cost of the implementation is also reduced and additional hardware logic to sort and store is not required and due to the same it uses the minimum memory and at most two registers are required. Basic sorting algorithms is having the orientation of hardware structures on VLSI implementation requires reducing the number of interface outputs and implementation of sorting algorithms based on the same type of processor elements with regular and local connections. The vertically-parallel method for sorting one dimensional arrays of numbers has been developed already involved in the graph-based algorithm proved to be not effective and this system with the adding of the RAM based sorting significantly improves the sorting speed of data arrays.

KEYWORDS: Bubble sort, FPGA, parallelism, sorting arrays

1. INTRODUCTION

The current stage in the development of information technology is represented by the accumulation of large amounts of data. When processing such data arrays, it is necessary to use data sorting and searching operations. The purpose of data arrays sorting is to stimulate the search for the requirement information. The main ways of increasing the speed of sorting operations are the advancement of new sorting algorithms, their transformation to the architecture of current mass parallel computer tools (software implementation) and their utilization in the form of a very large-scale integrated circuit (VLSI), whose hardware architecture emulates the network of the sorting algorithm. Mass parallel computers are in particular Graphics Processor Units (GPUs), which are SIMD class processors (Single Instruction Multiple Data). Their major feature is the use of one operation at a time to handle a bulk of independent data. CUDA is a cross platform compilation and execution system software for sorting large data arrays i.e., a part of it runs on the CPU, and remaining on the GPU.

Hardware implementation of data sorting algorithms requires the improvement of new algorithms and networks focused towards optimization for VLSI implementation by reducing the number of interface outputs based on the type of processing elements with regular and local connections. The Hardware oriented parallel sorting algorithms should be structured with deterministic data movement with x based on the same type of operations with regular and local

connections and x use conveyor and dimensional parallelism with a minimal number of interface outputs.

Previous data sorting methods were based on basic operations like pair wise comparison and rearrangement of numbers [1-3]. These sorting methods can be grouped into methods like insertion sort, exchange sort, merge sort and selection sort which are oriented toward sequential implementation.

In insertion sort algorithms, the process of pair wise comparison of numbers is associated with their transformation which usually defines places for numbers. The most VLSI implementations for these require a direct insertion algorithm due to its well-structured deterministic data movement. As the number of synchronously executed basic operations of pair wise comparison and transformation of numbers increase then the time of sorting the array of numbers reduces and complicates the hardware implementation.

The Algorithms for merge sort implementation are more structured, compatible, and adapted both to sorting one-dimensional and two-dimensional data arrays. The base is the basic operation of combining two or more ordered array i.e., while sorting data arrays, a two-way merger, is used. But this limits the speed as all of them are based on operations of pairwise comparison of data elements [8,15]. The solution provided in the literature is to improve its performance by multichannel consolidating and sending it to data groups. This would be helpful and improve the speed of data sorting tools may be different due to the similarity of sorting the numbers. The methods of sorting by counting and merging are mostly oriented on hardware implementation. still, the parallel algorithms for merge sorting of one-dimensional data arrays have the least speed when compared with the counting sorting algorithms, and their hardware implementation requires many interface outputs. It contains comparing each number in an array with all other numbers which is performed in two stages. In the first stage, a concurrent pairwise comparison of each number with all other numbers of the array, the amount of numbers greater, smaller and equal to such number is determined. In the second stage, based on the results of pairwise comparisons permutations of data is performed. The main advantage is high speed but it has disadvantages like heterogeneity, many interface outputs and significant hardware costs that are needed for its implementation. From literature, the sorting time and the number of interface outputs can be reduced and low hardware costs can be obtained by developing new parallel hardware oriented algorithms for data arrays sorting.

As per the authors Ivan Tsmots, Oleksa Skorokhoda, Volodymyr Antoniv[4] the requirements for progress of real time hardware are executed by using specialized tools that map the structural architecture of the sorting algorithm on hardware and is oriented on VLSI implementation. Implementation of highly efficient specialized sorting tools requires comprehensive use of modern element base, the development of new methods, algorithms, and VLSI structures. Real time and VLSI implementation of sorting algorithms with high productivity of equipment use are provided by parallelization and pipelining of sorting processes, hardware mapping of algorithms' structures on the architecture, which is adapted to the strength of the data streams flow. The orientation of sorting tools structures on VLSI implementation requires reduction of the interface pins number and implementation of algorithms based on the same type of processor elements (PE) with regular and local connections.

As per Ram´on Salvador Soto Rouco, the increasing computational power in modern computers in the form of several cores per processor and more processors, makes it necessary to reevaluate or to redesign sequential algorithms and data structures by using parallelism. A multi-threaded structure based on a shared memory model can be designed and implemented with low memory usage.

As per the authors Artjom Rjabov, Valery Sklyarov, an architecture for parallel data sorting with synchronous counting of every item frequency is designed for streaming data and consolidate data sorting in hardware, merging of primary sorted blocks with compressing of repeated items with calculating of repetitions in hardware, and merging large subsets received from the hardware in general purpose software. Hardware merges components of this architecture count

ISSN- 2394-5125 VOL 7, ISSUE 15, 2020

and compresses repeated items into sorted subsets in order to reduce merging time and prepare the data for frequent item computation.
The authors Marcelino et al. implemented a hardware/software hybrid sorter with a sorting unit based on insertion sorting algorithm and unequal merging unit by using Even-Odd sorting and quick sorting network for software implementation and Insertion sorting and unbalanced merge hardware along with pipelined sorting networks and balanced merging units. Chen and Prasanna [6] proposed a hardware/software hybrid solution for stimulating database operations using Field-Programmable Gate Array (FPGA) and Central Processing Unit (CPU) based on merge-sort algorithm where first few sorting stages are implemented in FPGA as folded bitonic sorting networks and the rest of the algorithm is implemented in CPU.

Teubner et al [7] analysed the acceleration of Hardware by frequent item computation to use for FPGAs by three different methods. The first method along with min-heap data structure in Block Random-Access Memory (BRAM) for data storage. The second method uses two search trees using lookup tables instead of BRAMs with min-heap structure. The third method uses a pipelined circuit that results in better performance and scalability.

In [9] they recommended a sorting network based hardware sorters with consecutive merge in software as well as different approaches of partial sorting for minimal and maximal subsets extraction which is used in frequent item computation [10, 11]. In [12] the authors proposed a multi level architecture for minimal/maximal subset extraction which utilizes a general purpose processor of a host PC and the programmable logic and processing system of Zynq device.

This paper is organized as Section II describes the Existing System and Section III discusses the Proposed Method and Section IV describes the Simulation results and Section V concludes the paper followed by references.

2. EXISTING SYSTEM

a) *The Bubble Sort Algorithm*

It requires linear sorting for n numbers given as a one dimensional array [13,14]. It uses the method of sorting one by one element stored in the array by comparing them against one another for their values as shown in figure 1.

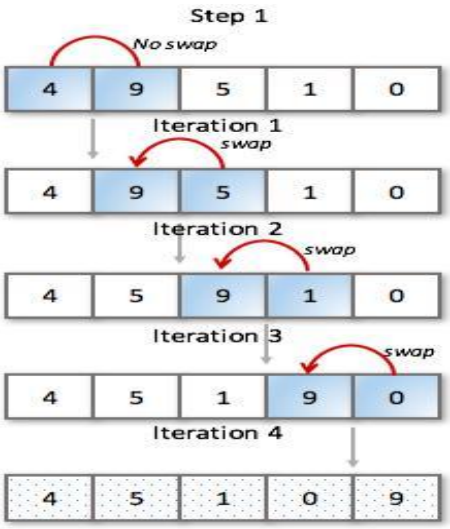

Figure 1 Bubble sort algorithm

ISSN- 2394-5125 VOL 7, ISSUE 15, 2020

This algorithm has certain drawbacks like Longer Execution time or low speed and requires large memory and variable.

b) Vertical parallel Method

This method of sorting requires the parallel receipt of N numbers by bit cuts with higher bits forward and the parallel formation of bit cuts of sorted numbers. sorting of a one dimensional array of numbers which executes Nxn basic operations. When performing each of the i-th (i=1,..,n) stage of sorting for its implementation.

The flow graph of the algorithm for vertically parallel sorting of a one-dimensional array of numbers is shown in Fig. 2, where F1 and Fc are respectively functional and control operators, and PU – processor units.

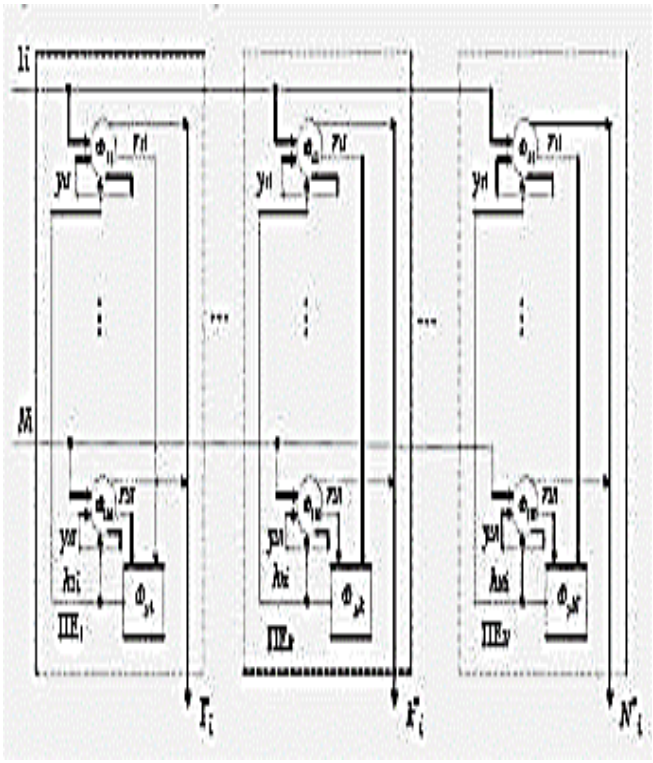

Figure 2 Flow graph of the algorithm for vertically-parallel sorting of a one dimensional array of numbers

3. PROPOSED METHOD

RAM based Sorting Method

The RAM based sorting uses the multiple read and write characteristics of Random Access Memory to sort one dimensional data stored for numbers to be sorted. It performs sequential search and swap operations simultaneously to sort the numbers. External sorting is a class of sorting algorithms that can handle massive amounts of data. External sorting is required when the data being sorted do not fit into the main memory of a computing device (usually RAM) and instead they must reside in the slower external memory, usually a hard disk drive.

The major advantages of the RAM based Sorting method are no additional hardware is required, reduction in cost of implementation and usage of the minimum memory i.e., utmost two registers for sorting the given numbers.

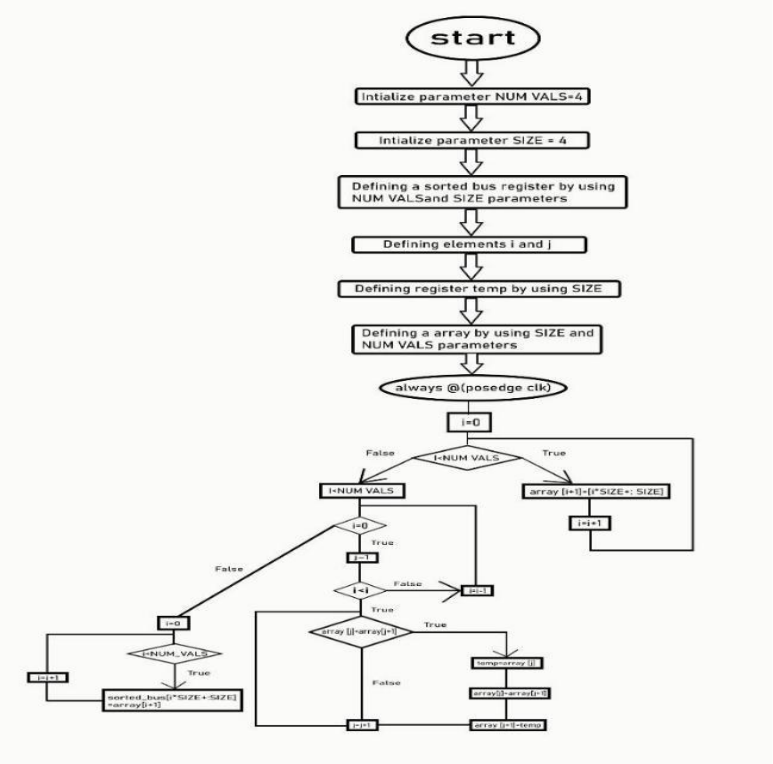

Figure 5 Flow Graph of the Algorithm for Ram Based Sorting of a One Dimensional Array of Numbers

4. SIMULATION RESULTS

The designs are simulated by using ISIM Simulator for functional verification. The designs are modelled in Verilog HDL and are synthesized by using Xilinx ISE 14.5 Tool. The FPGA Device opted is XC3S500E with a speed grade of -5 and package of FG320.

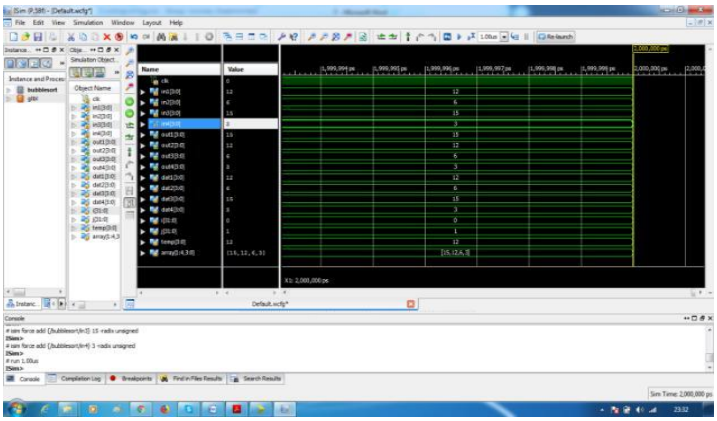

Figure 6 Simulation Waveform of Bubble Sort Algorithm

ISSN- 2394-5125 VOL 7, ISSUE 15, 2020

The figure 6 shows the simulation waveform for bubble sorting algorithm. The figure 7 shows the Register Transfer Logic which shows the conversion of algorithm to gate level implementation of bubble sorting algorithm. figure 8 shows the technology equivalent schematic developed for bubble sorting algorithm in 90nm cmos technology. The figure 9 shows the FPGA implementation of the bubble sort algorithm for optimal delay area trade off.

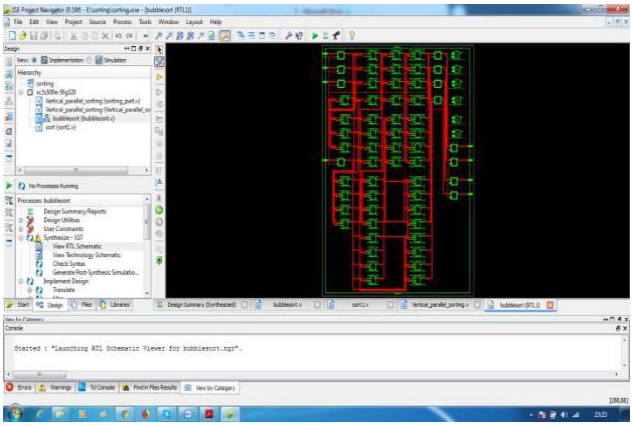

Figure 7 RTL Schematic of bubble Sort Algorithm

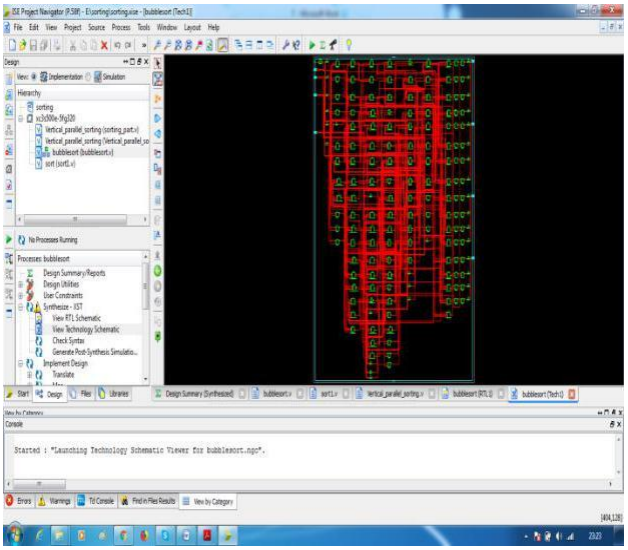

Figure 8 Technology Schematic for Bubble Sort Algorithm

ISSN- 2394-5125 VOL 7, ISSUE 15, 2020

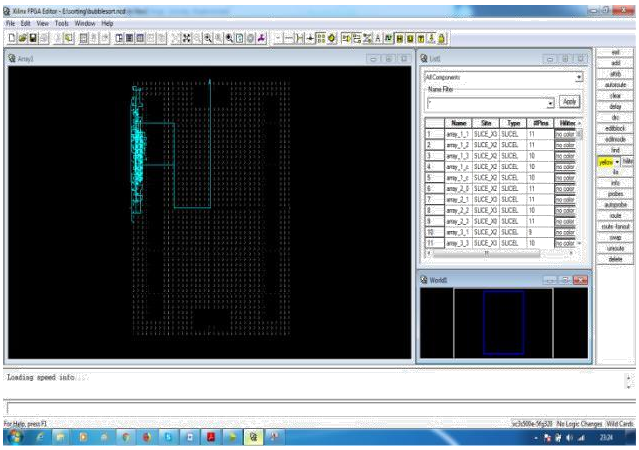

Figure 9 FPGA Implementation of the bubble sort algorithm

The figure 10 shows the simulation waveform for vertical parallel sorting algorithm. The figure 11 shows the Register Transfer Logic which shows the conversion of algorithm to gate level implementation of the vertical parallel sorting algorithm. figure 12 shows the technology equivalent schematic developed for vertical parallel sorting algorithm in 90nm cmos technology. The figure 13 shows the FPGA implementation of the vertical parallel sorting algorithm for optimal delay area trade off.

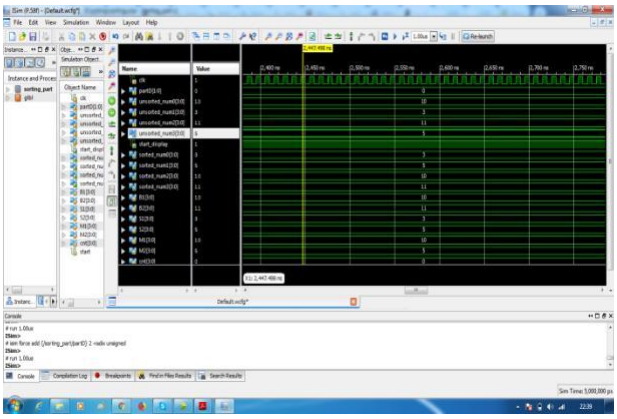

Figure 10 Simulation result of vertical parallel sorting algorithm

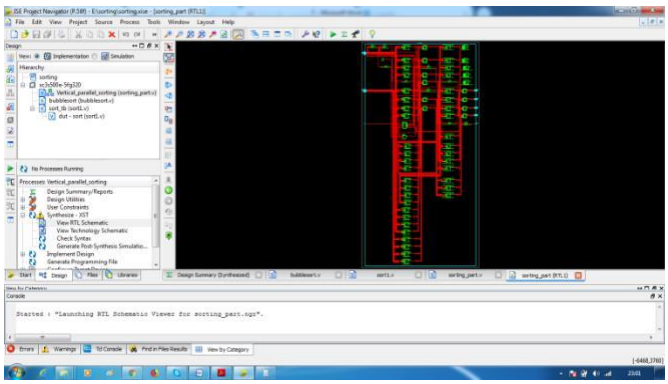

Figure 11 RTL Schematic for vertical parallel algorithm

ISSN- 2394-5125 VOL 7, ISSUE 15, 2020

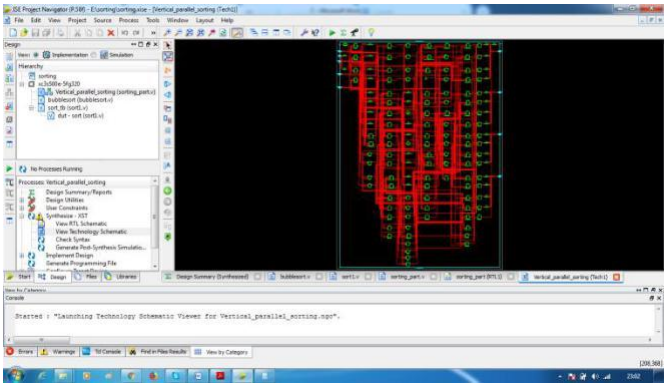

Figure 12 Technology Schematic of the vertical parallel algorithm

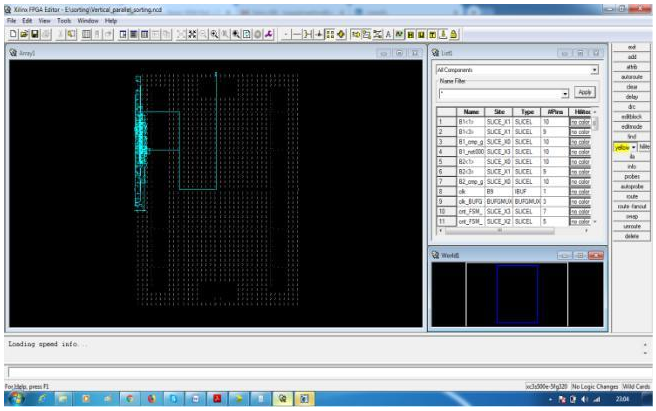

Figure 13 FPGA Implementation of vertical parallel algorithm

The figure 14 shows the simulation waveform for RAM based sorting algorithm. The figure 15 shows the Register Transfer Logic which shows the conversion of algorithm to gate level implementation of the RAM based sorting algorithm. figure 16 shows the technology equivalent schematic developed for RAM based sorting algorithm in 90nm cmos technology. The figure 17 shows the FPGA implementation of the RAM based sorting algorithm for optimal delay area trade off.

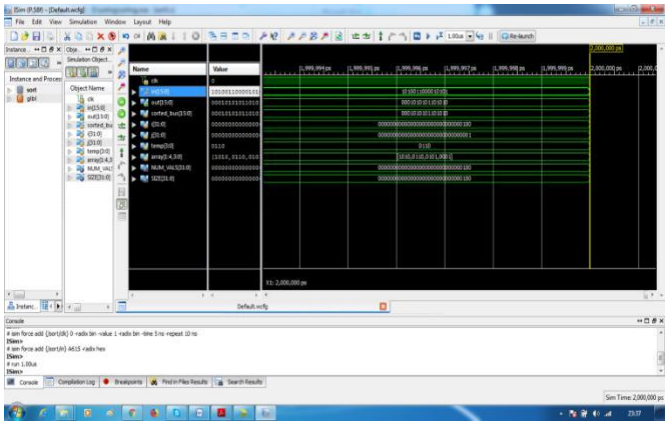

Figure 14 simulation Result of the RAM based sorting algorithm

ISSN- 2394-5125 VOL 7, ISSUE 15, 2020

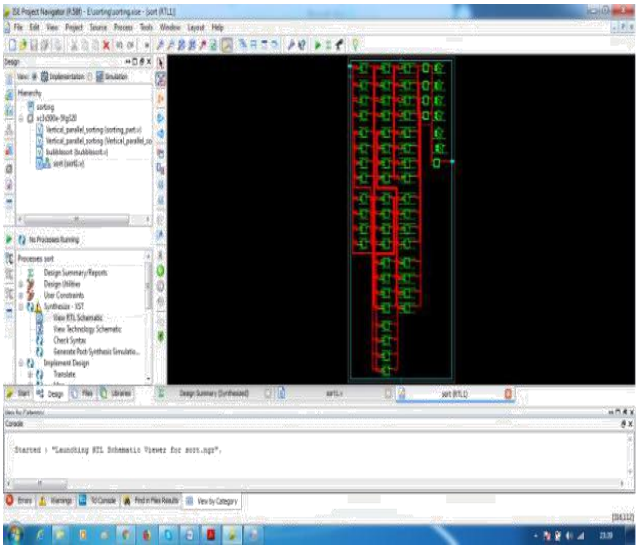

Figure 15 RTL Schematic of RAM based sorting algorithm

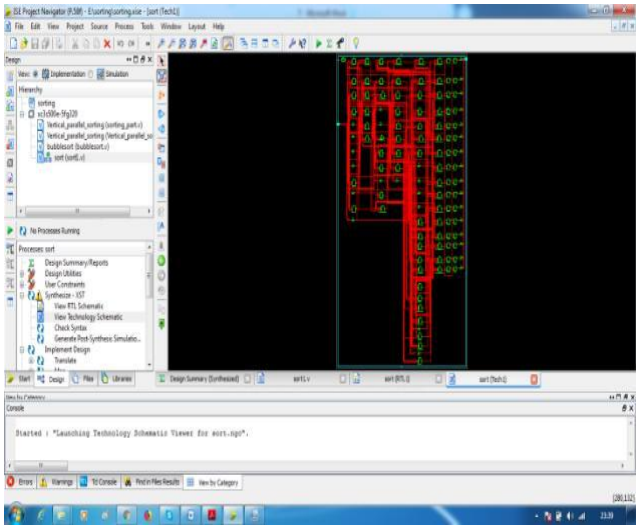

Figure 16 Technology Schematic for RAM based sorting algorithm

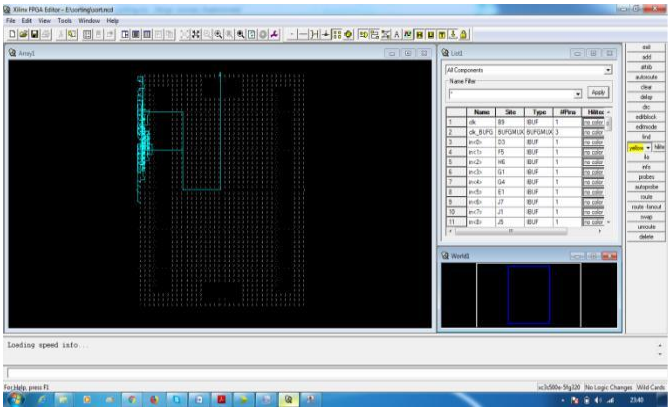

Figure 17 FPGA Implementation for RAM based Sorting algorithm

ISSN- 2394-5125 VOL 7, ISSUE 15, 2020

Table. 1: Comparison Table

Parameter	Bubble sorting algorithm	Vertical parallel sorting algorithm	RAM based sorting algorithm
Number of Slices	58 out of 4656	38 out of 4656	35 out of 4656
Number of 4-input LUTs	90 out of 9312	64 out of 9312	62 out of 9312
Average fan-out of Non-clock Nets	2.83	3.18	2.81
Logic Power	0.00025	0.00023	0.00004
Signal Data Power	0.00001	0.00002	0.00004
I/O Power	0.00004	0.00108	0.00006

The table 1 shows that the proposed algorithm which is based on RAM occupies less area both in terms of number of slices and 4-input Flip-Flops i.e., 57.39% when compared with bubble sort algorithm 37.07% when compared with vertical parallel sorting algorithm. The Average Fan-out of non clock nets is best for vertical parallel method. Also the logic power dissipated is reduced by 84% when compared with bubble sort algorithm and 22.82% when compared with vertical parallel sorting algorithm but the I/O power reduces only optimally and signal data power is slightly reduced.

5. CONCLUSION

This proposed paper has fast sorting algorithm for one dimensional array using RAM based sorting method. Since existing methods use longer execution time or it has low speed it requires large memory and variables and also the cost of equipment required for sorting numbers is very high. This proposed paper does not require a additional hardware to store the data maximum of two registers are used and RAM itself will be act as a memory to store data which reduces sorting time so that speed of operation will be increased and also it reduces cost of implementation. Therefore RAM based sorting method is best suitable for faster sorting algorithm. The designs are synthesized by using Xilinx ISE 14.5 Tools and are functionally verified by using ISIM Simulation tool. The proposed sorting algorithm that is based on RAM occupies less area both in terms of number of slices and 4-input Flip-Flops i.e., 57.39% when compared with bubble sort algorithm 37.07% when compared with vertical parallel sorting algorithm. Also the logic power dissipated is reduced by 84% when compared with bubble sort algorithm and 22.82% when compared with vertical parallel sorting algorithm but the I/O power reduces only optimally and signal data power is slightly reduced.

ISSN- 2394-5125 VOL 7, ISSUE 15, 2020

6. REFERENCES

[1] D. Knuth. The art of computer programming: Sorting and searching. M., 1978. - 844p.

[2] Tsmots I.G., Rahman M.L. Parallel Algorithms and Devices for Number Sorting. Collection of Scientific Papers of the Institute for Modeling Problems in Energy / Issue 11, Kyiv 2001. - P.83-91.

[3] Kukharev G.A. and others. The technique of parallel processing of binary data on VLSI. - Mn .: Vyssh. Shk., 1991. - 226 p.

[4] Grushitsky R.I., Mursaev A.H., Ugryumov E.P. Designing systems on the chips of programmable logic. - St. Petersburg: BHV-Petersburg, 2002. - 608 p.

[5] Parallel-vertical sorting of one-dimensional data arrays by the merge method using the calculation // I.G. Tsmots, V.O. Parubchak, V.Ya. Antoniv // Collection of scientific works of the Institute of Modeling Problems in the Energy. G. E. Puff Issue 68. - Kyiv: 2013. p. 92-100.

[6] Methods of parallel sorting of one-dimensional data arrays // V.O. Parubchak, V.Ya. Antoniv // Materials of the international conference "Intelligent decision-making and computing intelligence systems" ISDMCI 2014, May 28-31. - The Iron Port, 2014. - P.23-24.

[7] Method of development of hardware of parallel-coordinated sorting of data arrays in real time // I.G. Tsmots, V.Ya. Antoniv // Materials of the international conference "Intelligent decision-making and computing intelligence systems" ISDMCI 2015, May 25-28. - The Iron Port, 2015. - P.221-222

[8] Hardware for data sorting by real-time merge method // I.G. Tsmots, V.Ya. Anthoniv // Bulletin of the National University "Lviv Polytechnic" Collection of scientific works, № 814, ISM, Lviv, 2015. - P. 171-18.

[9] Application of a graphic processor for increasing the speed of the process of sorting large data arrays // I.G. Tsmots, Ya.P. Kiss, V.Ya. Antoniv // SCIENTIFIC WRITER of NLTU of Ukraine: Collection of scientific and technical works. - Lviv: RVB NLTU of Ukraine. - 2015. - Issue. 25.6. - P. 328 – 334.

[10] Parallel algorithms and structures for implementation of merge sort. // Ivan Tsmots, Oleksa Skorokhoda, Volodymyr Antoniv // International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 5 Issue 3, March 2016. Pp. 798-807.

[11] Algorithms and parallel structures for sorting data by insertion method // Tsmots I.G., Antoniv V.Ya. // SCIENTIFIC BULLETIN NLTU of Ukraine: Collection of scientific and technical works. - Lviv: RVB NLTU of Ukraine. - 2016. - Issue 26.1. - P. 340-350.

[12] Device for determining the maximum number from a group of numbers. // Tsmots I.G., Skorokhoda O.V., Medykovskyy M.O., Antoniv V.Ya. // Patent of Ukraine for invention No. 110187, Nov 25, 2015, Bul. No. 22

[13] FPGA Implementation of Vertically Parallel Minimum and Maximum Values Dtermination in Array of Numbers / I. Tsmots, V. Rabyk, O. Skorokhoda, V. Antoniv. // 2017 14th International Conference, The Experience of Desingning and Application of CAD Systems inMicroelectronics (CADSM). PROCEEDINGS. Polyana. February 2125, 2017. Pp. 234-236.

[14] Vertically-parallel method of sorting arrays of numbers. / I.G. Tsmots, V.Ya. Anthoniv // Collection of scientific works "Modeling and Information Technologies". Institute of Modeling Problems in Power Engineering. Issue 77, 2016. pp. 186 - 192.