International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 05 | May-2018 www.irjet.net p-ISSN: 2395-0072

Efficient comparison-free sorting algorithm and a hybrid approach for aerial image enhancement with applications in HPC and high end astronomy

Sreerag K M[1], Rajkumar P[2]

[1]M.Tech.VLSI Design, Dept. of Electronics and Communication Engineering, NCERC, Pampady, Kerala, India
[2]Professor, Dept. of Electronics and Communication Engineering, NCERC, Pampady, Kerala, India

---

Abstract - *This project proposes a hybrid approach for the enhancement of aerial images obtained from UAV/MAV cameras as well as an Application Specific Integrated Circuit (ASIC) design based sorting algorithm that excludes complex circuitry, but uses registers that hold the elements and their respective occurrences in the input set and employ matrix mapping to perform sorting. Aerial images are prone to many undesirable effects such as non-uniform lighting, turbidity due to haze and other atmospheric noise. The aim of the proposed scheme is to achieve the enhancement, where the existing techniques relying on Wavelet based Dynamic Range Compression (WDRC) algorithm followed by Local Contrast Enhancement (LCE) fails to achieve desired results. The processing involves both spatial and frequency domain operations which have been carried out through the implementation of post-processing techniques to those images which have still very poor contrast after the application of WDRC and LCE. The post processing techniques involve the use of histogram equalization, removal of noisy artifacts and Laplacian sharpening in a sequential order. The scheme was used for the enhancement of a series of aerial images obtained from MAV camera on flight. Images enhanced by the scheme are found to be of much better clarity and vividness, especially in the case of images acquired by MAV inbuilt cameras. Sorting plays a vital role in designing systems so as to achieve High Performance Computing (HPC) since once a set of items issorted, many other problems become easy.*

Key Words: *ASIC, Sorting, Astronomy, VLSI*

1.INTRODUCTION

Sorting algorithms have been widely researched for decades due to the ubiquitous need for sorting in many application domains. Sorting algorithms have been specialized for particular sorting requirements/situations, such as large computations for processing data], high-speed sorting, improving memory performance, sorting using a single CPU, exploiting the parallelism of multiple CPUs, parallel processing for grid-computing in order to leverage the CPUs powerful computing resources for big data processing Due to the ever-increasing computational power of parallel processing on many core CPU- and GPU-based processing systems, much research has focused on harnessing the computational power of these resources for efficient sorting. However, since not all computing domains and sorting applications can leverage the high throughput of these systems, there is still a great need for novel and transformative sorting methods. Additionally, there is no clear dominate sorting algorithm due to many factors, including the algorithms percentage utilization of the available CPU/GPU resources, the specific data type being sorted, amount of data being sorted.

To address these challenges, much research has focused on architecting customized hardware designs for sorting algorithms in order to fully utilize the hardware resources and provide custom, cost-effective hardware processing. However, due to the inherent complexity of the sorting algorithms, efficient hardware implementation is challenging. To realize fast and power-efficient hardware sorting, a significant amount of hardware resources are required, including, but not limited to, comparators, memory elements, large global memories, and complex pipelining, in addition to complicated local and global control units. Most prior work on hardware sorting designs are implemented based on some modification of traditional mathematical algorithms, or are based on some modified network of switching structures with partially parallel computing processing and pipelining stages. In these sorting architectures, comparison units are essential components that are characterized by high-power consumption and feedback control logic delays. These sorting methods iterativel move data between comparison units and local memories, requiring wide, high-speed data buses, involving numerous shift, swap, comparison, and store/fetch operations, and have complicated control logic, all of which do not scale well and may need specialization for certain data-type particulars. Due to the inherent mixture of data processing and control logic within the sorting structures processing elements, designing these structures can be cumbersome, imposing large design costs in terms of area, power, and processing time.

Furthermore, these structures are not inherently scalable due to the complexity of integrating and combining the data path and control logic within the processing units, thus potentially requiring a full redesign for different data sizes, as well as complex connective wiring with high fan-out and fan-in in addition to coupling effects, thus circuit timing issues are challenging to address. Additionally, if multiple processors are used along with pipelining stages and global memories, the data must be globally merged from these stages to output the complete final sorted data set. To address these challenges, in this paper, we propose a new

sorting algorithm targeted for custom, IC-designed applications that sort small- to moderate-sized input sets, such as graphics accelerators, network routers, and video processing DSP chips. For example, graphics processing uses a painter unit that renders objects according to the objects depth value such that the object can be displayed in the correct order on the screen. In video processing, fast computation is required for small matrices in a frame in order to increase the resolution using digital filters that leverage sorting algorithms. Even though we present our design based on these scenarios, our design also supports processing large input sets by subsequently processing the data in multiple, smaller input sets (i.e., in sets of N ¡ 100000) using fast computations, and then merging these sets. However, since applications with larger input sets (on the order of millions) are usually embedded into systems with large computational resources, such as data mining and database visualization applications running on highperformance grid computing and GPU accelerators, these applications can harness those powerful resources for sorting.

Unmanned Air Vehicles like UAVS and MAVs have small inbuilt cameras to take photographs (aerial images) for a number of civil and defence applications. Images taken during the flight from heights above 100m are subjected to various undesirable effects due to camera movement, non-uniform lighting conditions, turbidity, haze, fog, clouds etc. This not only causes a significant reduction in the clarity of such images, but the important details may be hidden under the noise that contaminates the image. It is thus obvious that the utility of these captured images, unless processed, is very limited. Natural scenes have high dynamic range. Images captured by the cameras are often poor rendition of the natural scenes. Even though the human eyes have a lesser dynamic range, the biological system adjusts to it by dynamic range compression and adapting locally to each part of the scene. The lowdynamic range of cameras results in the loss of contrast. There are many algorithms for dealing with problems regarding limited dynamic range like global histogram modification, logarithmic compression etc. However, these methods alone contribute little to enhancement of the quality of images suffered from the above problems.

Considerable research work is under progress, on the processing of images obtained from Micro Air Vehicle (MAV), in vehicle detection, remote sensing, and correction of camera rolling. Manera et.al presents a computational system which automatically processes optical and multispectral MAV images. The system includes image acquisition, rectification and image mosaicing. Bharati et.al proposes a differential morphology closing profile for extracting vehicles automatically from traffic images, which includes image pre-processing, differential morphology profile, thresholding and filtering. A technique based on digital photogrammetric technique for large scaling mapping has been developed by Udin et al, which has advantages like low cost, faster and simplicity. A method for automatic generation of orthophoto mosaics using scale invariant feature transform (SIFT) for automatic key point detection and matching problems has been proposed by Marcus et.al in. Many advanced methods have been developed to improve the local contrast of the images. E. Lands theory has produced good results in dynamic range compression and color constancy while MSRCR (Multi-Scale Retinex with Color Restoration) is a retinex based algorithm that uses logarithmic compression and spatial convolution. The main drawback of MSRCR is that the color restoration changes chromatics of the image in an undesirable manner. Adaptive and Integrated Neighborhood Dependent Approach for Nonlinear Enhancement (AINDANE) is used for images taken under low illuminance condition, which includes adaptive luminance enhancement, adaptive contrast enhancement and color restoration. Locally Tuned Sine Non-Linear (LTSN) technique is used for extremely high contrast images, which includes adaptive intensity enhancement, contrast enhancement and color restoration. A novel approach for removing illuminance effect on aerial images taken under non-uniform lightning condition is proposed Ansari and Arigela, where histogram adjustment and color restoration are used along with WDRC algorithm to improve the visual quality. But this algorithm (Automatic WDRC) is not sufficient for images having very poor contrast. In thepresent work, we propose a postprocessing step which improves the visual quality of those images with very poor contrast and eliminates noise from the images.

The algorithm employs a hybrid approach, as it uses both spatial and frequency domain operations. Initially, the histogram of the image is adjusted to avoid the illuminance effect so as to deal with the strong spectral characteristics of the aerial images. Local contrast enhancement is used in the algorithm in order to restore and improve the contrast lost during dynamic range compression. A non-linear color restoration is then used to provide color constancy. In the post processing step, use of histogram equalization enhances the contrast of very low contrast images for which the approach of Ansari and Arigela is inadequate. It may also be noted that the application of Gaussian and median filtering eliminates the noise in the images which occur while performing color restoration. Finally, Laplacian sharpening is employed to make the edges more clearly visible.

2. DESIGN METHODOLOGY

The data path contains several circuit components: a one-hot decoder, register arrays, a serial shifter, a parallel counter (PC), tri-state buffers and multiplexors, a one-detector, and an incrementor/ decrementor circuit. In order to meet the setup-hold delay time bewteen the clock and data stabilization for the elements storage registers, the delay elements components are a cascade of an even number of inverters. These circuit components are standard CMOS circuit components, which are commonly used components for advanced CMOS technologies beyond 90 nm, making our design scalable for further advanced lowcost CMOS technologies.

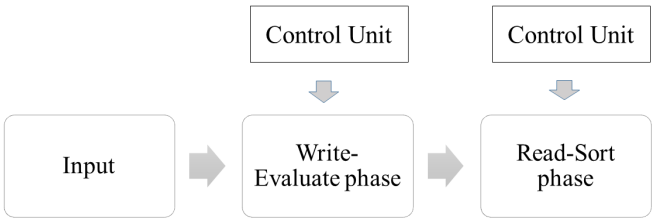


Fig 1. Overall Sorting System

Write-Evaluate Phase

During the write-evaluate phase, each binary input element is converted to the elements one-hot weight representation by the one-hot decoder. The decoders output enables an associated register in a register array to record the binary input elements occurrence. We refer to this register as an order register (ORi) array, where the i th register stores the i th input element. Each register is a simple DFF register of size k-bit. This operation is equivalent to the recording of the element in the transposed matrix in our algorithm. Simultaneously, the one-hot decoder enables an associated register in another register array the flag register (FRi) arraywhich records the number of occurrences of this element in the input set. For each occurrence of a duplicated element, the associated flag register is triggered, and the occurrence is recorded by incrementing the registers stored value using a 10 bit incrementor. This operation is equivalent to having multiple 1s for repeated elements in a row in the transpose matrix.

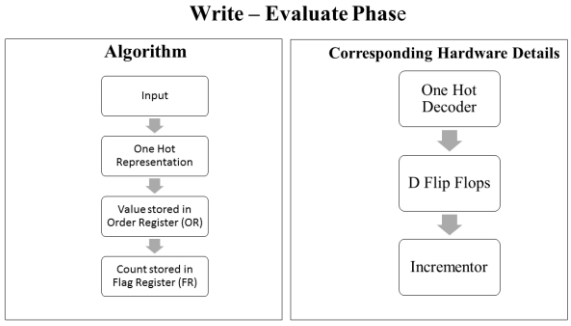
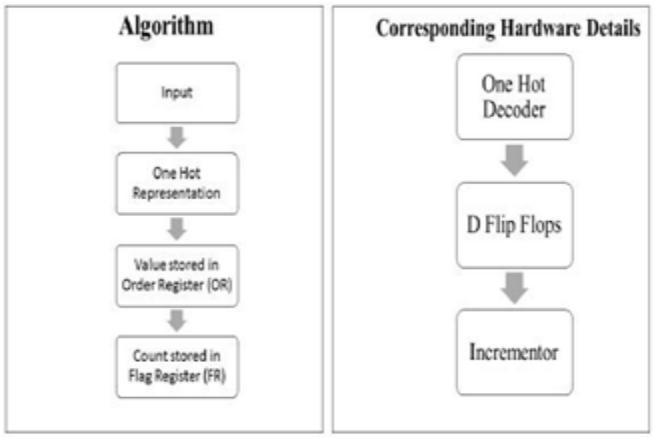


Figure 2 : Write-Evaluate Phase



A parallel counter in the control unit (Section V-B) controls the end of the write-evaluate phase when the counters value reaches the maximum number possible inputted elements (i.e., N = 2k). Even though the input set may contain less than the maximum number of elements, assuming that the input set is full realizes the simplisity of the read-sort phases operation. The control unit asserts the READ-ENA signal and deasserts the WRITEENA signal when the writeevaluate phase completes, which enables the read-sort phase on the next clock edge. The write-evaluate phase requires a fixed N clock cycles since the phase always iterates for the maximum number of potential input elements.

Read-Sort Phase

Read-sort phases data path, which comprises of a k-bit sorted shift register (SRi) array of size N that stores the elements in their final sorted order, and a k-bit PC that indexes into the order register array to process each element in turn. The element ordering, ascending or descending, is userspecified, and can be controlled by either left- or right-shifting in the elements. A one-detector circuit detects if the flag register value is 1 or not, and a decrementor circuit subtracts a 1 from the flag register, the result of which is stored back in to the flag register, when processing replicated elements. In this figure, the write-evaluate phases data path components that are used in the read-sort phase are encompassed in the dashed lines.

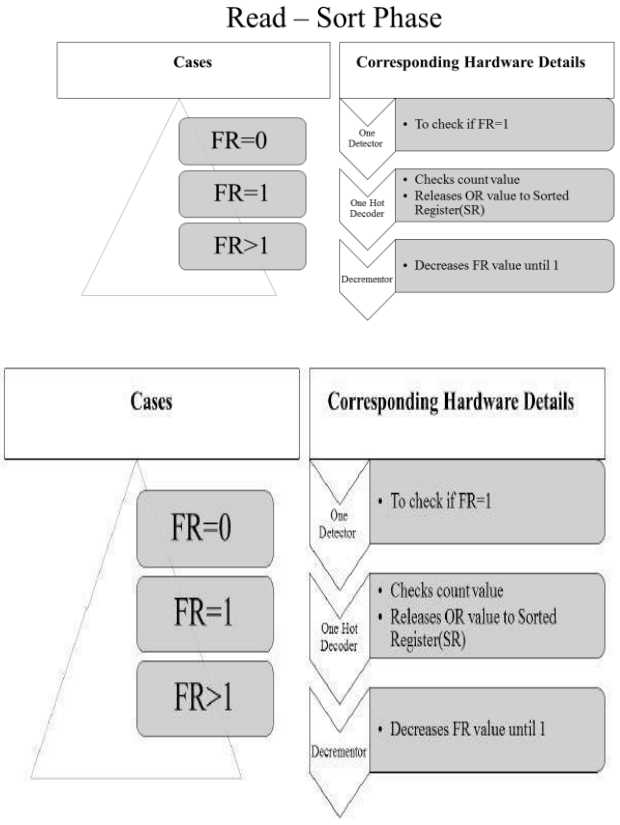


Figure 3 Read Sort Phase

The read-sort phase begins after the WRITE-ENA signal is deasserted and the READENA signal is asserted, which sends the PCs value to the one-hot decoder at each new readsort clock cycle. The one-hot decoder converts this counter value to the values one-hot representation, which enables the associated order and flag registers to read/release the registers values, and the order registers value is stored into the sorted register array if-and-only-if that elements flag register value is greater than 0, meaning there was at least one occurrence of that input element. The one-detector evaluates the flag register value to control whether or not the element is stored in the sorted register array. If the flag register records a value equal to or greater than 1, the associated element should be stored in the sorted register array a number of times equal to the flag registers value. The case is simple when the flag register value is 1, which is detected by the one-detector. To avoid complex comparison units (i.e.,equal to or greater than 1), detecting values greater than 1 can be easily determined using the decrementors carry out single. Thus, if the one-detectors evaluation is false (i.e., 0 is the onedetectors decision output), but when decrementing the flag registers value, the resulting carry out flag is 0, this means that the flag registers value was greater than 1. In both cases, the input element should be stored into the sorted register array. Indexing to the next input element is inhibited by disabling the PCs increment, which allows the replicated element to be stored in the sorted register array until the flag register value reaches 0. Otherwise, the flag registers value is 0, the element is not in the input set, and thus is not stored into the sorted register array, and the PC is incremented.

The read-sort cycle time can be divided into three cases based on the flag registers value. In case one, the flag registers value is 0 (i.e., the element is not in the binary matrix), and thus, this element is not stored in the sorted register array, and the PC is incremented (i.e., proceed to the next row in the transpose matrix). The timing of the readsort cycle (Treadcycle) in case one is the sum of the PCs increment (TPC), the one-hot decoders (TOH), and the onedetectors (TOD) delays,

Treadcycle = TPC + TOH + TOD.

We can see that the one-detector and decrementor both operate concurrently with the flag register values evaluation.

In case two, the flag registers value is 1, meaning that the element is in the input set once, and thus this element is read from the order register using the one-hot decoder and a tri-state buffer at the registers output, the element is stored in the sorted register array, and the PC is incremented. As with case one, a flag register value of 0 and 1 both require one clock cycle. The timing of the read-sort cycle (Treadcycle) in this case is the sum of the PCs increment (TPC), the onehot decoders (TOH), the one-detectors (TOD), and the sorted register arrays (TSR) delays,

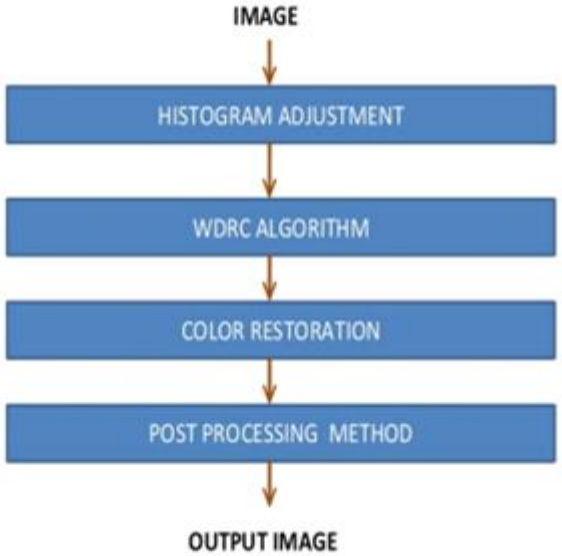
Treadcycle = TPC + TOH + TOD + TSR.

In case three, the flag registers value is greater than 1 (i.e., the elements corresponding row in the transpose matrix contains more than one 1). Similar to case two, this element is stored into the sorted register array, but in this case, the flag register is also decremented. The PCs increment is disabled until the elements flag register reaches 1, signaling that all occurrences of the element have been stored into the sorted output array. The timing of the read-sort cycle (Treadcycle) in this case is the sum of the PCs increment (TPC), the one-hot decoders (TOH), the decrementors (TDA), and the flag register arrays (TFR) delay,

Tread-cycle = TPC + TOH + TDA + TFR.

Aerial Image Enhancement

An aerial image is a projected image which is ”floating in air”, and cannot be viewed normally. It can only be seen from one position in space, often focused by another lens. Aerial image technology was used in optical printers and movie special effects photography before the advent of computer graphics in movie production, and also for combining animation and live action footage onto one piece of film. Aerial photography is the taking of photographs from an aircraft or other flying object. Platforms for aerial photography include fixed-wing aircraft, helicopters, unmanned aerial vehicles (UAVs or ”drones”), balloons, blimps and dirigibles, rockets, pigeons, kites, parachutes, stand-alone telescoping and vehicle-mounted poles. Mounted cameras may be triggered remotely or automatically; handheld photographs may be taken by a photographer. Aerial photography should not be confused with air-to-air photography, where one or more aircraft are used as chase planes that ”chase” and photograph other aircraft in flight. The methods used in this project includes three basic steps namely preprocessing steps which include spatial domain operations, (histogram adjustment) followed by WDRC algorithm. Then the resultant image is subjected to post processing techniques like histogram equalization, Laplacian sharpening, Gaussian filtering and median filtering.



(E

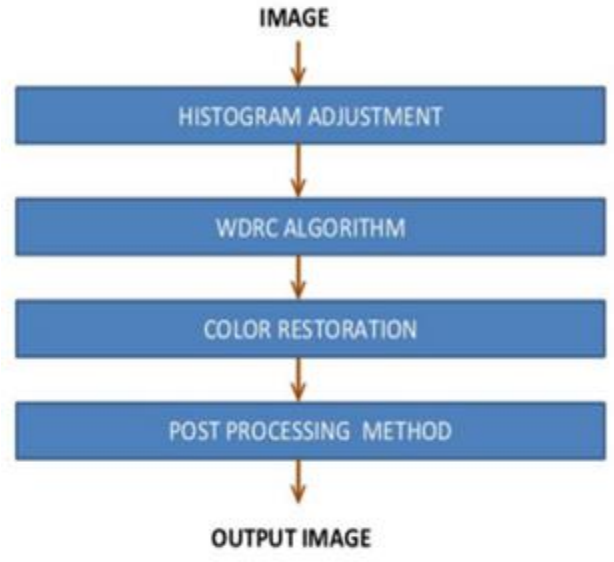
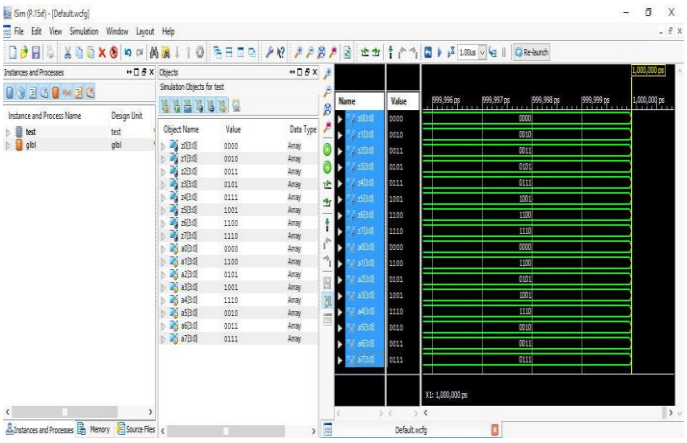
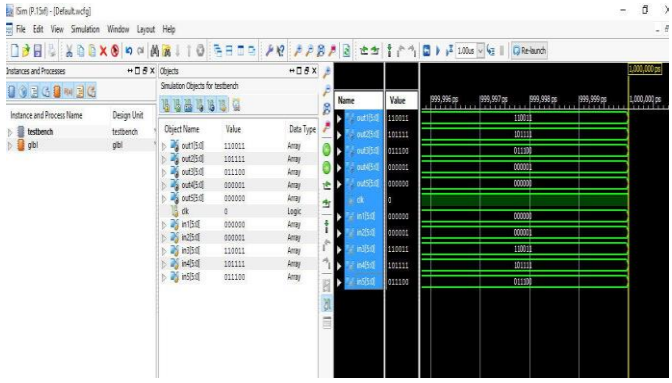


Figure 4 Aerial Image Enhancement Steps

3. RESULTS

The proposed algorithm was tested and found to be efficient than existing sorting algorithm which are based on comparison. Xilinx ISE platform was used for testing the results. Timing analysis as well as overall efficiency was tested. Some existing algorithms as well as the proposed method was simulated for 3 bit sorting. The proposed method was found to be more effective.

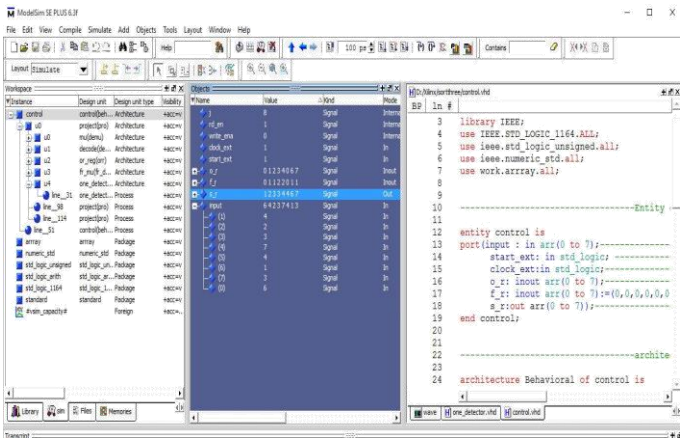


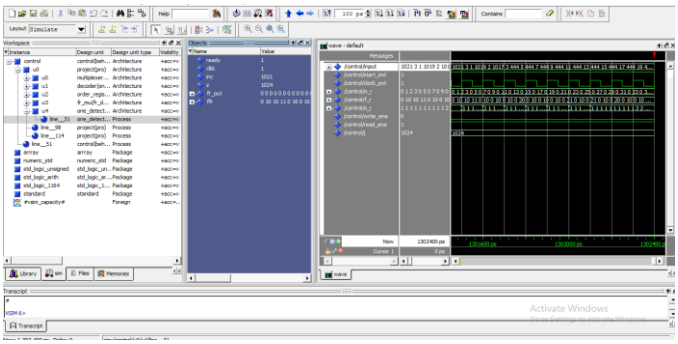


Bitonic Sort

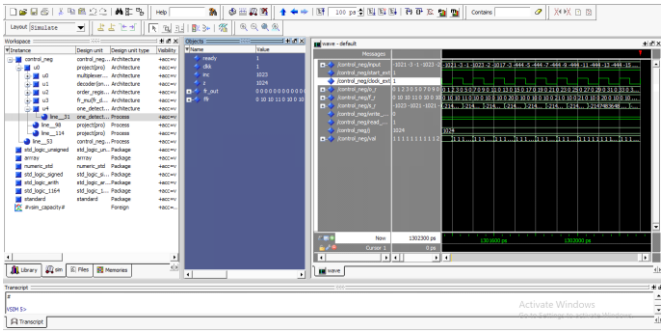
```
ISim P.15xf (signature 0xc3576ebc)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.

Selection Sort

                TIME START RESET  |          ARRAY
Finished circuit initialization process.
                   2     1     0  |   xxxxx xxxxx xxxxx xxxxx xxxxx xxxxx xxxxx xxxxx xxxxx xxxxx
                   3     1     0  |   0 0 0 0 0 0 0 0 0 0
                   4     1     1  |   0 0 0 0 0 0 0 0 0 0 Reset Pressed
                   7     1     0  |   0 0 0 0 0 0 0 0 0 0
                   7     1     0  |   3 5 0 1 6 9 8 7 2 4
                  31     1     0  |   0 5 3 1 6 9 8 7 2 4
                  53     1     0  |   0 1 3 5 6 9 8 7 2 4
                  73     1     0  |   0 1 2 5 6 9 8 7 3 4
                  91     1     0  |   0 1 2 3 6 9 8 7 5 4
                 107     1     0  |   0 1 2 3 4 9 8 7 5 6
                 121     1     0  |   0 1 2 3 4 5 8 7 9 6
                 133     1     0  |   0 1 2 3 4 5 6 7 9 8
                 151     1     0  |   0 1 2 3 4 5 6 7 8 9
                 200     1     0  |   0 1 2 3 4 5 6 7 8 9
                ------------------------------------------
                 201     1     0  |   b 9 3 c 5 e 7 f 1 0
                 225     1     0  |   0 9 3 c 5 e 7 f 1 b
                 247     1     0  |   0 1 3 c 5 e 7 f 9 b
                 285     1     0  |   0 1 3 5 c e 7 f 9 b
                 301     1     0  |   0 1 3 5 7 e c f 9 b
                 315     1     0  |   0 1 3 5 7 9 c f e b
                 327     1     0  |   0 1 3 5 7 9 b f e c
                 337     1     0  |   0 1 3 5 7 9 b c e f
                 338     1     1  |   0 1 3 5 7 9 b c e f Reset Pressed
```

Selection Sort





Proposed System – 3 Bit



Proposed System – 10 Bit

Method	Time Taken
Bitonic Sort	12.025ns
Bubble Sort	8.852ns
Odd Even Sort	1.826ns
Comparison Free Sorting	1.705ns

Proposed System – 10 Bit

Timing Reports

Aerial Image Enhancement Results

5. CONCLUSION

Our sorting design exhibits linear complexity O(N) with respect to the sorting speed, transistor count, and power consumption. This linear growth is with respect to the number of elements N for N = 2K where K is the bit width of the input data. The slope of the linear growth rate is small, with a growth rate of approximately 6 for the transistor count and power consumption, and 1.5 for the sorting speed. The order complexity and growth rates are due to simple basic circuit components that alleviate the need for SRAM-based memory and pipelining complexity. Our mathematically-simple algorithm streamlines the sorting operation in one forward flowing direction rather than using compare operations and frequent data movement between the storage and computational units, as with other sorting algorithms. Our design uses simple standard library components including registers, a onehot decoder, a one detector, an incrementer/ decrementer, and a PC, combined with a simple control unit that contains a small amount of delay logic.

In this project application of the WDRC algorithm in aerial imagery is presented. The results obtained from large variety of aerial images show strong robustness, high image quality, and improved visibility indicating promise for aerial imagery during poor visibility flight conditions. The proposed algorithm is expected to improve the visual quality of the digital images by minimizing the effects of haze, motion blur etc. on the aerial images. A wavelet based dynamic range compression method is optimized for the overall picture enhancment along with a motion blur detection and correction method. The future scope of this project include processing the video that are captured from a specific altitude, employing accurate street views of urban areas in the field of cartography etc. Camera roll problem can also be corrected.

Combined with efficient image processing algorithms, this sorting algorithm can be employed for further ASIC development specifically for astronomical indexing and imaging.

REFERENCES:

[1] Saleh Abdel-Hafeez, Member, IEEE, and Ann Gordon-Ross, Member, IEEE, 'An Efficient O(N) Comparison-Free Sorting Algorithm', IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 25, NO. 6, JUNE 2017

[2]Željko Ivezić, Andrew J. Connolly, Jacob T VanderPlas, Alexander Gra, 'Statistics, Data Mining, and Machine Learning in Astronomy: A Practical Python Guide for the Analysis of Survey Data', Princeton Univeristy Press, Princeton and Oxford

[3] PRABHAKAR GUPTA, VINEET AGARWAL, MANISH VARSHNEY, 'DESIGN AND ANALYSIS OF ALGORITHMS', PHI Learning private LTD.

[4] Y. Bang and S. Q. Zheng, "A simple and efficient VLSI sorting Architecture," in Proc. 37th Midwest Symp. Circuits Syst., vol. 1. 1994,pp. 70–73.

[5] Y. Han, "Deterministic sorting in O(n log log n) time and linear space," J. Algorithms, vol. 50, no. 1, pp. 96–105, 2004.

[6]F.-C. Leu, Y.-T. Tsai, and C. Y. Tang, "An efficient external sorting algorithm," Inf. Process. Lett., vol. 75, pp. 159–163, Sep. 2000.

[7] E. Mumolo, G. Capello, and M. Nolich, "VHDL design of a scalableVLSI sorting device based on pipelined computation," J. Comput. Inf.Technol., vol. 12, no. 1, pp. 1–14, 2004.