Design and implementation of sorting algorithms based on FPGA

A. F. M. Fahad Alif
Dept. of Electronics and Communication Engineering
Khulna University of Engineering & Technology
Khulna 9203, Bangladesh
fahadalif98@gmail.com

Sheikh Md. Rabiul Islam
Department of Electronics and Communication Engineering
Khulna University of Engineering & Technology
Khulna 9203, Bangladesh
robi@ece.kuet.ac.bd

Polash Deb
Department of Electronics and Communication Engineering
Khulna University of Engineering & Technology
Khulna 9203, Bangladesh
debpolash1995@gmail.com

***Abstract*— Analysis of the efficiency of sorting algorithms most often bound up to software simulation. In practical field real time operation needs a system of faster sorting operation because software sorting is less effective. With the advancement of VLSI design sorting algorithm can be easily implemented as a block in any system which can be effectively used when necessary. In this paper, three most common sorting algorithms bubble sort, selection sort and insertion sort algorithm will be implemented using Verilog HDL language. For all algorithms RTL (Register Transfer Level) diagram will be examined and associated timing diagram will be analyzed for worst case scenario. Then the comparative analysis for the algorithms will be given form the analysis and synthesis report and from some operating parameters. The algorithm which has better hardware performance can be used as a block in any system with parallelism.**

***Keywords*— FPGA (Field Programmable Gate Array), RTL, Verilog HDL, sorter implementation, timing analysis.**

I. Introduction

Sorting is one of the fundamental problem in computer science that is studied for decades. Almost all field of technology requires sorting operation like image processing, multimedia, scientific computing, networking Over the years many sorting algorithms have proposed, each of them characterizable by a measure of how much time an algorithm takes to complete the sorting as the problem size gets bigger. While all algorithms take more time to sorting if number of elements increased, some are slower than others. It is not efficient to execute sort programs on general-purpose computer today due to its inherent algorithmic difference in computation and sorting in real time operation. Also, software algorithm is not feasible to achieve high speed. In every general-purpose computer there must be a block of arithmetic and logical operation which need sometimes sorted data for operation [1] [2] [3]. So, it will be natural to implement a dedicated hardware block inside the ALU (Arithmetic Logic Unit) for sorting operation.

With the rapid development of VLSI design field such a module for sorting can be implemented using FPGA (Field Programmable Gate Array) on a single chip [2]. It is a type of reconfigurable gate array where designer can easily change the design of implementation with gate level parallelism [4]. From a storage device large number of data proceed to an ALU and sorted result will be restored in storage device. This denotes that sorting of data set requires sequential transfer from and to a sorter. In FPGA large number of memory is available and can be accessed with a small number of pins. So, FPGA implementation gives a better speed performance with parallelism. Among all the sorting algorithm some are efficient to use for small amount of data and some are effective for large number of data. Bubble sort, selection sort and insertion sort are the most common and basic sorting algorithm which are faster than complex sorting algorithm like merge sort or quick sort etc. for small number of data. In this work bubble sort, selection sort and insertion sort algorithm will be implemented using Verilog HDL language. The RTL diagram for all the implementation will be given and associated timing diagram will be examined for worst-case situation (like worst case situation for bubble sort is the data are in reverse order). Some operation parameters will be studied. For comparative analysis and synthesis report and timing summary report will be tabled up and comparison result be will be given based on analysis of parameters.

The rest part of this paper is organizing in three sections. Section II describes the methodology of the implementation for all three algorithms. Section III describes the RTL diagram and comparative timing analysis. Section IV gives the concluding part.

II. Methodology

A. Bubble sort

In this sorting algorithm each pair of adjacent elements is compared, repeatedly steps through the end of the list to be sorted and swap them if they are in the wrong order (the latter one is smaller than the previous one). After each iteration, one less element (the last one) is required to be compared until there are no more elements left to be compared. If we have total n elements, then we need to repeat the swapping process for n-1times. Among all the sorting the implementation of bubble sort is simple compared to other algorithms, but the performance is impractical to implement. In worst-case scenario (numbers are in reverse order) bubble sort has complexity of $\underline{O}(n^2)$. Here n is the number of items being sorted [3]. The flowchart for hardware implementation of this sorting algorithm has given in Fig 1.

B. Selection sort

Selection sort algorithm sorts an array of number by repeatedly finding the smallest element from unsorted part and putting it in first position. It maintains two subarrays in the specific array: the subarray which is already sorted and remaining subarray which is unsorted. The procedure is: locate the smallest element in the array, interchange it with the element in the first position, locate the second smallest element and interchange it with the element in the second position and keep going on until the given array is sorted. For worst, average and best case scenario the complexity is same as bubble sort , $O(n^2)$ because run time depends only on the amount of order in the file. Figure 2 represents the implementation procedure of selection sort algorithm.

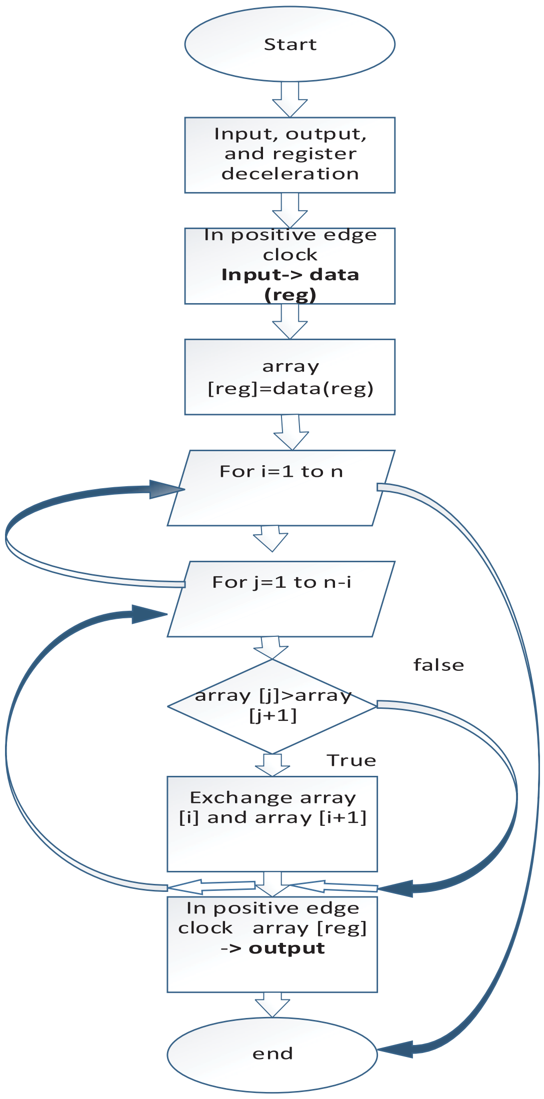
Start
Input, output, and register deceleration
In positive edge clock **Input-> data (reg)**
array [reg]=data(reg)
For i=1 to n
For j=1 to n-i
false
array [j]>array [j+1]
True
Exchange array [i] and array [i+1]
In positive edge clock array [reg] **-> output**
end

Fig 1. Flowchart for bubble sort implementation.

C. *Insertion sort*

In insertion sort algorithm at each iteration, one element is picked from the input data, the location is found where it should belong within the sorted list and inserted in its proper position. It is comparison-based (in-place) sorting algorithm where a sub-list is preserved to be always sorted (mainly the lower part of array is preserved to be sorted). A number which going to be inserted in this sorted sub-list, must find its proper place and then it will be inserted there. That's why it is called insertion sort. Worst case (numbers are in reverse order) performance needs $O(n^2)$ comparison or swap. But the worst-case space complexity is $O(n)$. The average case is also same as worst case which makes insertion sort impractical for sorting large arrays. But for small array it is much quicker than any other sorting algorithm. Figure 3 shows the flowchart for implementation of insertion sort algorithm.

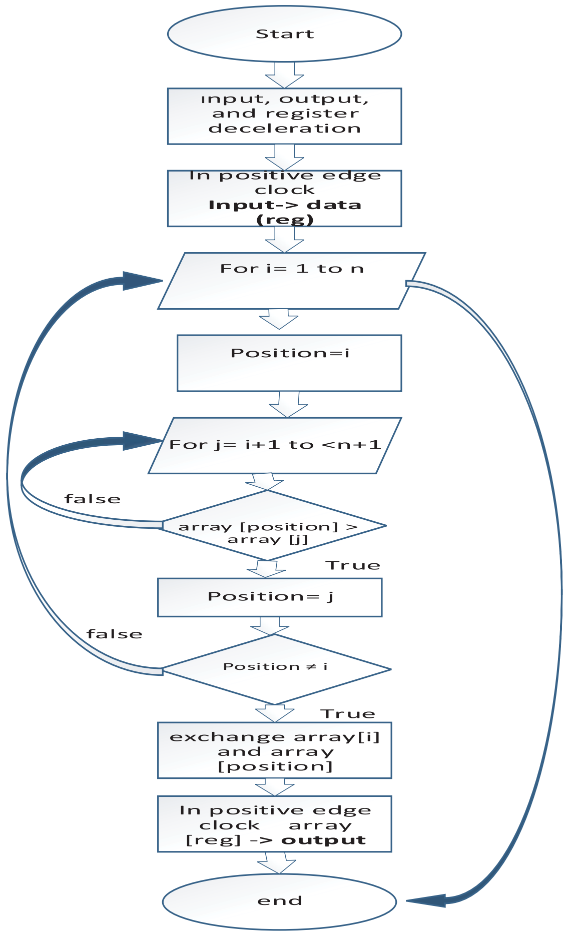
Start
Input, output, and register deceleration
In positive edge clock **Input-> data (reg)**
For i= 1 to n
Position=i
For j= i+1 to <n+1
false
array [position] > array [j]
True
Position= j
false
Position ≠ i
True
exchange array[i] and array [position]
In positive edge clock array [reg] **-> output**
end

Fig 2. Flowchart for selection sort implementation.

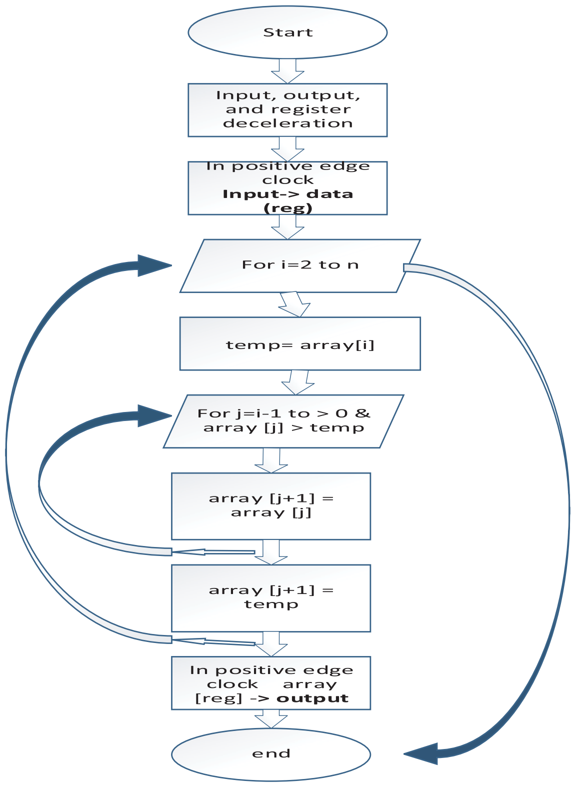
Start
Input, output, and register deceleration
In positive edge clock **Input-> data (reg)**
For i=2 to n
temp= array[i]
For j=i-1 to > 0 & array [j] > temp
array [j+1] = array [j]
array [j+1] = temp
In positive edge clock array [reg] **-> output**
end

Fig 3. Flowchart for insertion sort implementation.

III. RESULT

The methodology describes above sections are implemented using Quartus Prime software and in the Xilinx ISE 4.7 design suite for simulating chip performance. The main working procedures are implementing the processes, give the entire view of RTL diagram, timing diagram and analysis of operating parameters for determining the best algorithm.

A. RTL diagram

For all three sorting algorithms the initial implementation procedures are same as shown in flow diagram. At positive edge clocking input numbers that will be sorted will be stored in intermediate registers. Then the numbers are passed through to another register which will performed as one-dimensional array. Then the main algorithm will be performed using array. Lastly sorted data will be stored in register for displaying output. For all the algorithms 8-bit registers are taken for all operation. Fig 4,5,6 shows the RTL diagram for bubble, selection and insertion sort respectively. The rectangular box shown in figures are the inputs, outputs and clock pins.

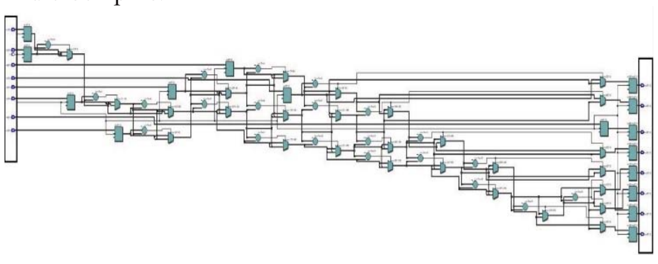

Fig 4. RTL diagram for bubble sort.

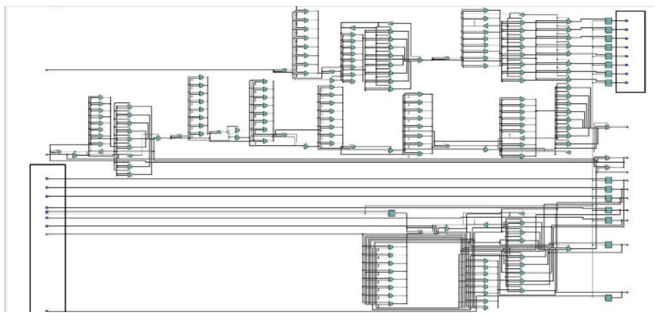

Fig 5. RTL diagram for selection sort.

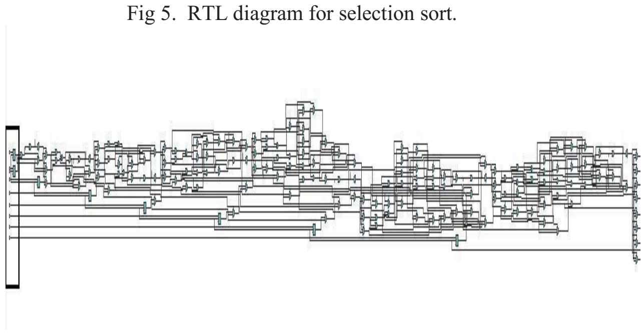

Fig 6. RTL diagram for insertion sort.

B. Timing diagram

To simulate the implemented chip ISim simulator is used. For bubble sort and insertion sort numbers are taken in reverse order as this is the worst-case situation. As selection sort doesn't depend on the initial orientation of numbers so a random input is given. The timing diagram for bubble, selection and insertion sort has given is Fig 7,8 and 9 respectively.

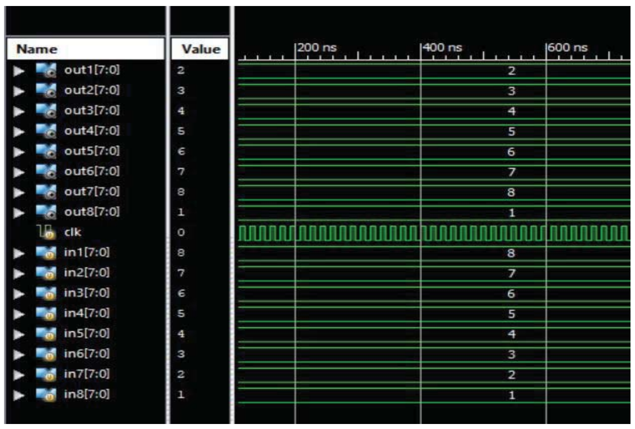

Fig 7. Timing diagram for bubble sort.

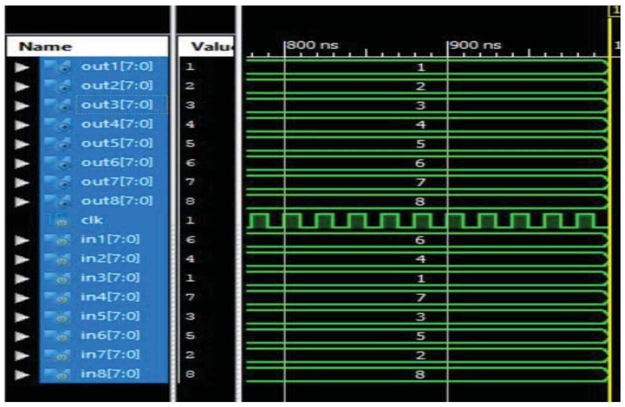

Fig 8. Timing diagram for selection sort.

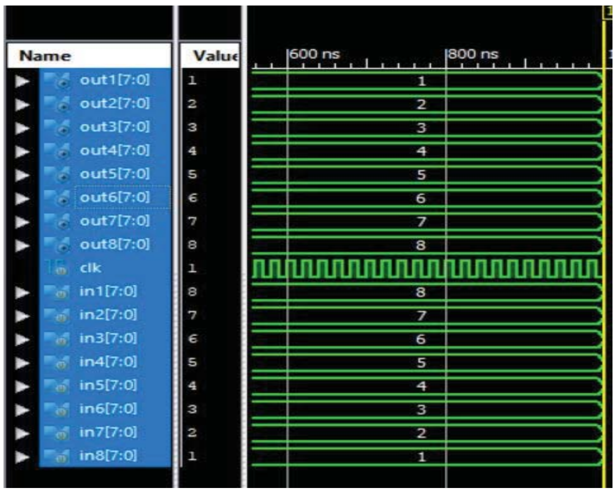

Fig 9. Timing diagram for insertion sort.

C. Comparative analysis

TABLE I. BASIC OPERATING PARAMETER

Parameter	Value
Vcc	1.1 V
I/O standard voltage	2.5 V
Junction temperature	27.7◦C
Total thermal power estimation	359.5 mW

In TABLE I some basic operating parameter has shown. These parameters mainly depend on the vendor family i.e. which chip is selected to implement and simulate the design. In all three implementation these Vcc and I/O voltage are same. Only a slight change has seen in junction temperature and total thermal power estimation, but it is negligible. So, the comparison hasn't shown in TABLE I.

TABLE II. : ANALYSIS AND SYNTHESIS REPORT

	Bubble sort	Selection sort	Insertion sort
ALM needed	406	692	438
Average fanout	3.34	3.94	3.44
Number of slices LUTs	528	1675	983
Average LUT depth	23.74	77.11	19.88
Total current drawn	51.58 mA	51.56mA	51.55mA
Clock to setup on destination clock	19.795	47.056	17.364

TABLE II shows the comparative analysis and synthesis report for the implemented design. Firstly, Adaptive Logic Module (ALM) is the basic building block of supported devices families and is designed to maximize performance and resource usage. An implementation needs efficient use of ALM. Selection sort uses so much ALM compared to others, so it is inefficient. Bubble sort has some less amount of ALM than insertion sort, so it seems to be a little bit efficient. But an implementation should require less amount of time to process input. So, efficiency of insertion sort lies within its timing performance. LUTs (Look Up Tables) are usually read-only and their content can only be changed during FPGA configuration. Adding more LUTs was either too expensive or considered not very useful. So, when considering number of slices LUT insertion sort has medium number of LUT and bubble sort has less. But when Average LUT depth is considered insertion sort has a smaller number of LUTs compared to other. So, insertion sort is more and fast compared to other. Current drawn is same for all implementation. When we look at the clock to set up on destination clock insertion sort required a smaller number of clocks.

TABLE III. TIMING SUMMARY

	Bubble sort	Selection sort	Insertion sort
Clock period	16.729ns	41.499ns	14.931ns
Clock Frequency	59.775MHz	24.097MHz	66.975MHz
Minimum input arrival time before clock	0.302ns	0.307ns	0.288ns
Maximum output required time after clock	0.640ns	0.640ns	0.640ns

TABLE III shows the estimated timing summery for the synthesis of all program. Here Minimum period after synthesis is an estimation of the clock period for signals inside the implementation. It calculates the worst-case path timing from clock edge to clock edge for flip-flops within the design. For insertion sort it is less than other two. And, insertion operates in higher clock than other two. But will they work in higher clock depend on the rest of the two summaries in table 2. Minimum input arrival time before clock means the required setup time from worst case top-level design input to the clock. And maximum output required time after clock defines the delay of the top-level design from the clock to external outputs for the worst-case. The main significance of these two is if they are less than minimum period then the implementation may perform in highest clock frequency. Insertion sort will perform better then bubble and selection sort. And selection sort has the worst performance.

IV. CONCLUSION

In this paper three common sorting algorithm is implemented using FPGA. The RTL diagram, timing diagram and comparative analysis is discussed. From our point of view insertion sort algorithm shows much faster operation when implemented in hardware. Though these algorithms are less effective when large data is encountered but if the data set can be divided into several parts and proceed into several blocks (designing several modules of sorter in a large module) for sorting then it can be a faster operation and it will hold the parallelism [3] [5] of the system. We will develop new optimized sorting algorithm for computer processing.

REFERENCES

[1] Magesh.V, Megavarnan.S, Pragadish.A, Saravanan.S , " FPGA IMPLEMENTATION OF SORTING ALGORITHMS " in International Journal For Technological Research In Engineering Volume 5, Issue 8, April-2018.

[2] Gayathri K,HarshiniV S, Dr Senthil Kumar K K, " Hardware implementation of sorting algorithm using FPGA" in IJARIIE-ISSN(O)-2395-4396, Vol-4 Issue-2 2018.

[3] Ashrak Rahman Lipu, Ruhul Amin, Md. Nazrul Islam Mondal, Md. Al Mamun, "Exploiting Parallelism for Faster Implementation of Bubble Sort Algorithm Using FPGA" in 2nd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE) 8-10 December 2016, Rajshahi-6204, Bangladesh.

[4] Dmitri Mihhailov, Valery Sklyarov, Iouliia Skliarova, Alexander Sudnitson, "Hardware implementation of recursive sorting algorithms" in 2011 International Conference on Electronic Devices, System and Applications (ICEDSA).

[5] Stephan Olariu, M. Cristina Pinotti, Si Qing Zheng, "An Optimal Hardware-Algorithm for Sorting Using a Fixed-Size Parallel Sorting Device" in IEEE TRANSACTIONS ON COMPUTERS, VOL. 49, NO. 12, DECEMBER 2000.