2nd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE)
8-10 December 2016, Rajshahi-6204, Bangladesh

Exploiting Parallelism for Faster Implementation of Bubble Sort Algorithm Using FPGA

Ashrak Rahman Lipu, Ruhul Amin, Md. Nazrul Islam Mondal, Md. Al Mamun
Dept. of Computer Science and Engineering, RUET, Rajshahi, Bangladesh
ashrakrahman@gmail.com, ruhul113056@gmail.com, nimbd109@gmail.com, cse_mamun@yahoo.com

***Abstract*—Sorting is a classic problem that has been studied for decades. From the beginning of computing, many Sorting algorithms have been investigated. Bubble sort is a very common and powerful sorting technique used in different applications. For high speed data processing, we need faster and efficient environment for any sorting algorithm. In this purpose, FPGA based hardware accelerators can show better performance for high speed data processing than the general purpose processors. In this paper, the sequential and parallel bubble sort algorithm is implemented using FPGA. We show that parallel implementation of Bubble sort algorithm is almost 10 times faster than that of sequential implementation for 20 different data inputs. However, this implementation is faster for more data inputs.**

Keywords—FPGA; Bubble Sort ; Clock Cycle; Odd-even transposition; Swappers; Schematic view

I. Introduction

Many applications in computing require things to be in order. Sorting is a procedure that is needed in numerous computing systems [1]. In computer science, sorting can be used to sort data either ascending or descending which is usually required in algorithm such as evolutionary algorithm. Sorting plays a major role in commercial data processing and in modern scientific computing. For example, transaction processing, combinatorial optimization, astrophysics, molecular dynamics, linguistics, genomics, weather prediction etc are the applications of Sorting.

Several algorithms to undertake the sorting process are Selection sort, Merge sort, Insertion sort, Heap sort, Radix sort, and Bubble sort. Heap sort, Radix sort, and Merge sort are powerful to sort large number of data [2]. Meanwhile the selection sort, Insertion sort and Bubble sort are powerful for few data. As we are concern for sorting a few data we took bubble sort for our implementation.

Reconfigurable gate arrays, also known as field programmable gate arrays (FPGA), are widely used in the industry to implement a variety of digital circuits. FPGA is an integrated circuit that takes advantage of the gate-level parallelism that multi-core GPPs cannot. The design implemented in FPGA has longer design cycle than design implemented for parallel processing in GPPs. A large amount of memory is available internal to the FPGA and is accessed with a small number of pins such that the reconfiguration time is, for example, four orders of magnitude faster than the traditional approaches and at notably low cost [3].

Sorting implementation on FPGA has been implemented by many researchers. Srivasta et al. [4] proposed hybrid design for large scaling sorting on FPGA. Other research on high-speed parallel scheme is found for data sorting on FPGA in the paper [5]. The authors in the paper [6] introduced parallel bubble sort to utilize the concept in parallel computing.

To gain a better performance, fast accelerators based on FPGAs [7], GPUs [8] and multi-core CPUs [8] have been investigated in depth with increased intensity during the last few years. For these reasons the goal of this paper is to implement the bubble sort algorithm using FPGA to achieve better performance. We exploit the parallelism implementing Bubble sort algorithm using FPGA and hence we show that parallel implementation of Bubble sort algorithm is almost 10 times faster than that of sequential implementation for 20 different data inputs. In this paper we implement the mentioned algorithm for 20 data, each of 8-bit only. However, this implementation is faster for more data inputs no doubt.

The key idea of this paper is to implement parallel swapping for independent data by exploiting the parallel reconfigurable architecture of FPGA. The main contribution of our work is as follows:

- We exploit the parallelism to implement Bubble sort algorithm using FPGA.
- We have shown that parallel implementation is almost 10 times faster than that of sequential implementation for 20 different data inputs.

II. Methodology

A. Sequential Bubble Sort

The basic principle of bubble sort is to compare two consecutive numbers from leftmost to rightmost and swaps them if certain condition is met.

In Fig. 1, a list L = {6, 2, 4, 1} is taken for ascending sort. In every pass, left to right scan is done and two consecutive numbers are compared and swapped. In the example, the largest number is moved to the rightmost position for every pass. As every pass is over, the no. of unsorted data is decreased by one it is go on until all the data are sorted. In every pass, compare is done sequentially which can be depicted in Fig. 2.

Again, all the passes are done sequentially too. The algorithm of sequential bubble sort is given bellow:

```
Algorithm 1: Sequential Bubble Sort
Input :   L (unsorted data list)
Output: O (sorted data list)
    Step 1: compare and swap L sequentially
    Step 2: decrease L by 1 and repeat step1 until L is sorted
```

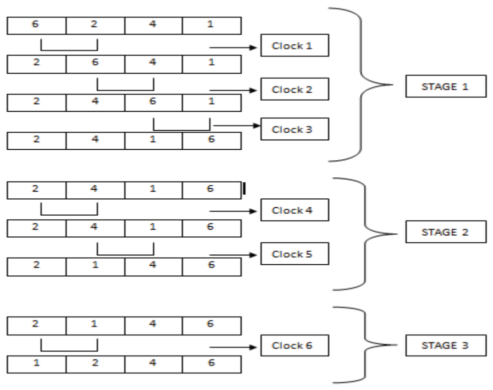


Fig. 1. Steps of sequential bubble sort algorithm for the example.

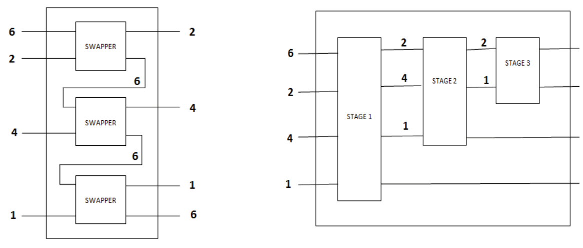


Fig. 2. Block Diagram of Stages for Sequential Sort

B. Parallel Bubble Sort

The motivated idea of parallel bubble sort is parallel comparison and swapping. Parallel swapping can be applied on independent data of a list. Odd-even transposition technique is a handy in this case. To explain it, let

$$L = < a_1, a_2, a_3, \ldots\ldots, a_p, a_q, \ldots.., a_n>$$

Be a list of n elements. Any pair of consecutive elements, say (a_p, a_q) is said to be odd or even pair if the position of first element p is odd or even respectively e.g. (a_1, a_2) is odd pair while (a_2, a_3) is even pair. Fig. 3 explains the technique with the above example.

From Fig. 3, every step/pass includes one odd and one even phase which work sequentially. But every phase works parallel swapping on consecutive data of the list. Fig. 4(a) depicts the odd-even transposition scheme. Again, steps/ passes are executed sequentially which is depicted in Fig. 4(b).On the basis of explanation, the algorithm of parallel bubble sort is given bellow.

```
Algorithm 2: Parallel Bubble Sort
Input : L (a unsorted list)
Output: O (a sorted list)
    Step 1: sort all the odd pair in parallel
    Step 2: sort all the even pair in parallel
    Step 3: Repeat steps 1-2 sequentially (n/2) times
```

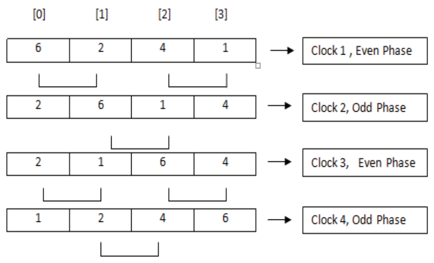


Fig. 3. Steps of parallel bubble Sort.

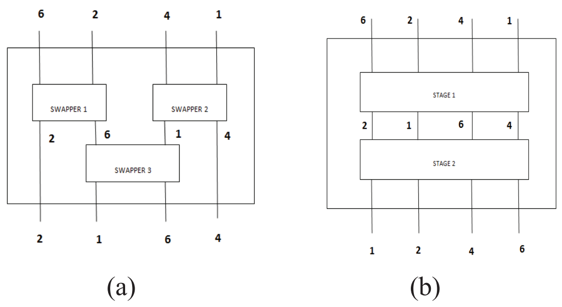


Fig. 4. Block Diagram of Stages for parallel bubble Sort.

C. FPGA Implementation of Sequential Bubble Sort

A hierarchical or layered approach is followed to implement the sequential and parallel bubble sort. The system is designed bottom-to-top approach in three layers. Fig. 5 shows the required layers to implement the sort in FPGA and again shows the schematic of swapper module.

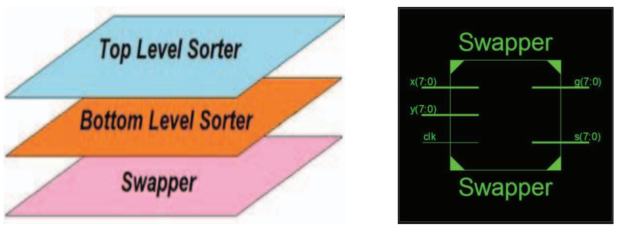
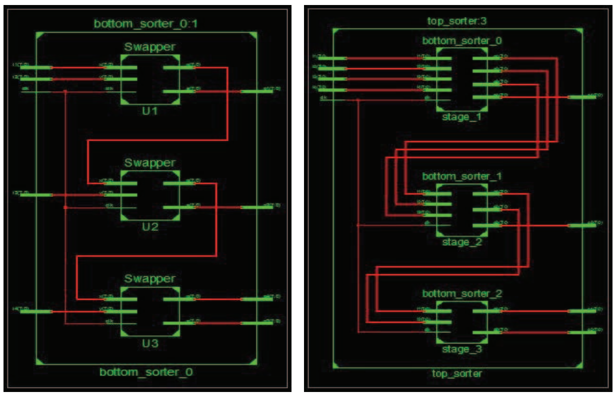


Fig. 5. Hierarchical layer and Schematic view of a swapper.



(a) (b)

Fig. 6. Wiring of bottom level and top level sorter for sequential bubble sort.

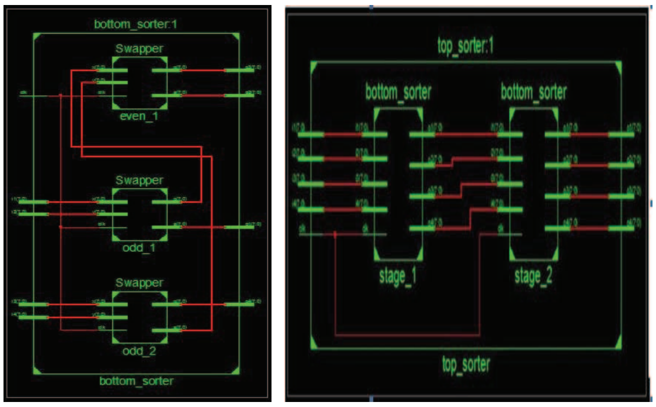
Swapper takes two input data, compare it and swaps them if swapping condition is met. Fig. 6(a), depicts the second upper layer, bottom_level_sorter of the system which is formed of swappers and their essential wiring. It is the implementation of Fig. 2 of sequential bubble sort scheme. It indicates the single step of Fig. 2(a), in which swappers run sequentially. The last layer, layer three, top_level_sorter's schematic is shown in Fig. 6(b). It is the implementation of Fig. 2(b), of sequential bubble sort. Here, bottom_level_sorters are wiring sequentially likely passes are done sequentially to form top_level_sorter or the whole system.

D. *FPGA Implementation of Parallel Bubble Sort*

Parallel bubble sort is also implemented as three layer system like the sequential one. It is most likely the sequential bubble sort but the orientation or wiring or connection is different. The basic building block of parallel bubble sort is the first layer swapper whose schematic is shown in Fig. 5 of sequential bubble sort. The second layer of the hierarchy, bottem_level_sorter is shown in Fig. 7(a).

In Fig. 7(a), swappers are connected in parallel exactly as described in Fig. 4(a). This type of connection forms the odd-even transposition here. FPGA implementation of last layer, layer three of parallel bubble sort is top_level_sorter which is shown in Fig. 7(b). It shows the sequential connection of bottom_level_sorter or odd-even transposition unit which implement the structure of Fig. 4(b), of parallel bubble sort. In case of FPGA implementation of the whole system, number is represented as 8 bit signed integer number.

The device utilized is Spartan-6 of 400 MHz clock speed. Xilinx ISE-14.7 software is used to design different modules, simulate to verify and to implement them FPGA. The main performance criterion in this study is execution time. We focus only to execute the sort in parallel to reduce clock cycle.



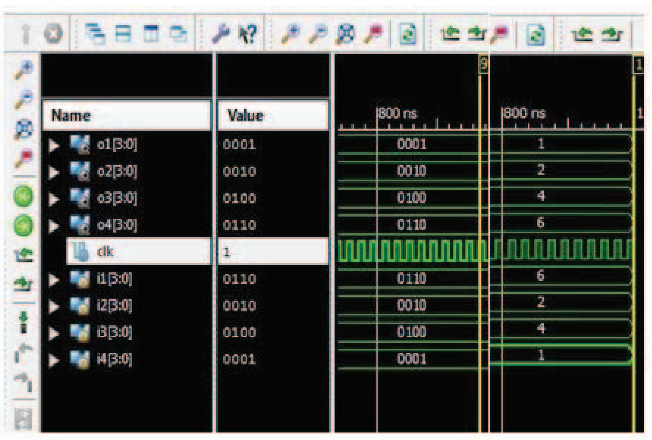
(a) (b)

Fig. 7. Wiring of bottom level and top level sorter for parallel bubble sort.

III. Result and Discussion

A. *Sequential Bubble Sort*

From the experiments, it is seen that sequential bubble sort can be applied to sort a list. Sequential bubble sort successfully sorted all the variation of inputs from 4 to 20. The example of 4 inputs is shown in Fig. 8.



Name	Value	800 ns	800 ns
o1[3:0]	0001	0001	1
o2[3:0]	0010	0010	2
o3[3:0]	0100	0100	4
o4[3:0]	0110	0110	6
clk	1		
i1[3:0]	0110	0110	6
i2[3:0]	0010	0010	2
i3[3:0]	0100	0100	4
i4[3:0]	0001	0001	1

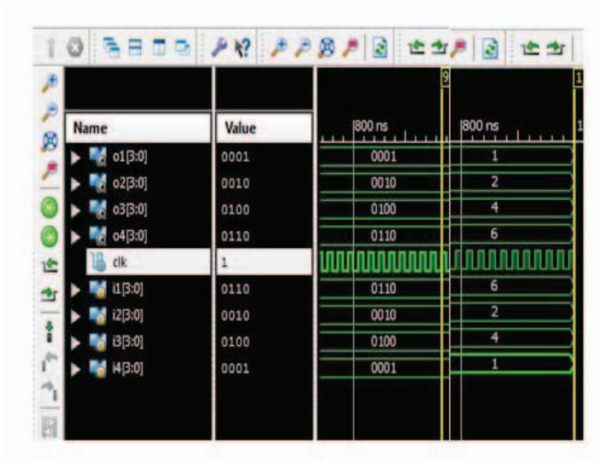
Fig. 8. Visualization of the result for four input sequential bubble sort.

From Fig. 1, it is seen that 6 clock cycle is required to completely sort the list. Its clock cycle depends on input number. If input number is N then required clock cycle is N*(N-1)/2.To improve the system i.e. reducing execution time, optimization procedure can be empowered.

B. *Parallel Bubble Sort*

From the experiments, it is evident that parallel bubble sort can be utilized to sort the numbers. Parallel bubble sort successfully sorts all the input from 4 to 20 numbers. Fig. 9 shows the example of 4 inputs above.

From Fig. 4, it is seen that only 4 clock cycles are needed to completely sort the list. Its clock cycle depends on input number. If input number is N then required clock cycle is N. To improve the system i.e. reducing execution time, optimization procedure can be empowered.



Name	Value
o1[3:0]	0001
o2[3:0]	0010
o3[3:0]	0100
o4[3:0]	0110
clk	1
i1[3:0]	0110
i2[3:0]	0010
i3[3:0]	0100
i4[3:0]	0001

Fig. 9. Visualization of the result for four input parallel bubble sort

C. *Comparison of Sequential and Parallel Bubble Sort*

In this paper, sequential and parallel bubble sort are compared. The comparison is focused on the clock cycle only. The clock cycle of sequential and parallel bubble sort is listed at Table 1. Finally, Fig. 10 depicts the clock cycle comparison.

TABLE I. COMPARISON OF CLOCK CYCLE

SL. No	Number of Data	Clock for Parallel sort	Clock for Sequential sort
1	4	4	6
2	6	6	15
3	8	8	28
4	10	10	45
5	12	12	66
6	14	14	91
7	16	16	120
8	18	18	153
9	20	20	190

In execution clock cycle, parallel bubble sort performs better than sequential bubble sort. Parallel bubble sort sorts inputs faster than the sequential bubble sort. This is because several processes execute parallel in parallel bubble sort. Therefore, it does not have to wait for previous one completion. On the contrary, in sequential bubble sort, the next process executes after the previous process have finished. In all cases: best, average and worst, parallel bubble sort requires less clock cycles than sequential bubble sort. That's why; parallel one is faster than sequential one.

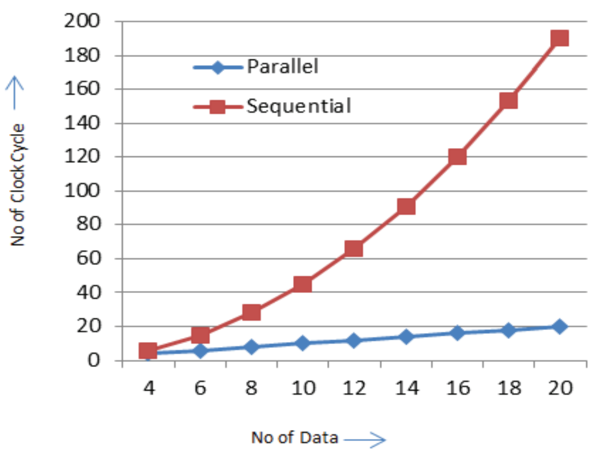


Fig. 10. Clock cycle comparison between serial and parallel bubble sort.

IV. CONCLUSION

In this paper, we have implemented Bubble sort algorithm in parallel using FPGA in order make it faster than that of sequential implementation. In fact, we exploit the parallelism for implementing the mentioned algorithm in FPGA. We have shown that parallel implementation of Bubble sort algorithm is almost 10 times faster than that of sequential implementation for 20 different data inputs. It is highly expected that this implementation is faster for more data input and hence we have a plan to implement it in future.

REFERENCES

[1] D.E. Knuth, The Art of Computer Programming. Sorting and Searching, vol. III, Addison-Wesley, 2011

[2] R. Abirami, "VHDL Implementation of Merge Sort Algorithm", *International Journal of Computer Science and Communication Engineering*, Vol. 3, No. 2, pp. 15-18, 2014.

[3] Chatter, Mukesh. "High performance self modifying on-the-fly alterable logic FPGA, architecture and method." *U.S. Patent No*. 5,838,165. 17 Nov. 1998.

[4] A. Srivastava, R. Chen, V. K. Prasanna, and C. Chelmis, "A Hybrid Design for High Performance Large-scale Sorting on FPGA" *In Re ConFigurable Computing and FPGAs*, pp. 1-6, 2015.

[5] S. Dong, X. Wang, and X. Wang, "A Novel High-Speed Parallel Scheme for Data Sorting Algorithm Based on FPGA" *2nd International Congress on Image and Signal Processing*, pp. 1-4, 2009.

[6] R. Rashidy, S. Yousefpour, and M. Koohi, "Parallel Bubble Sort Using Stream Programming Paradigm" *5th International Conference on Application of Information and Communication Technologies*, pp. 1-5, 2011.

[7] Asano, Shuichi, Tsutomu Maruyama, and Yoshiki Yamaguchi. "Performance comparison of FPGA, GPU and CPU in image processing." *2009 international conference on field programmable logic and applications*. IEEE, 2009.

[8] Che, Shuai, et al. "Accelerating compute-intensive applications with GPUs and FPGAs." *Application Specific Processors, 2008. SASP 2008. Symposium on*. IEEE, 2008

[9] white paper : www.xilinx.com