Hardware-based Systems for Partial Sorting of Streaming Data

Artjom Rjabov
Department of Computer Engineering,
Tallinn University of Technology,
Tallinn, Estonia
artjom.rjabov@ttu.ee

Abstract—**Fast data sorting is an essential component of many high-performance computing systems. High-throughput and highly parallel sorting algorithms are very appropriate for devices which provide massive parallelism like FPGAs and APSoCs. One of the major drawbacks of these platforms is amount of available resources which is a serious obstacle for design of hardware sorters. However, for many practical applications complete sorting is not important and only partial sorting for extraction of maximum and minimum subsets of the data is required. In this paper we investigate the maximum and minimum extraction problem, present a hardware-based extracting circuit based on pipelined periodic sorting networks and compare it with known alternatives.**

Keywords—High-performance computing systems, Information processing; Sorting networks; Parallel sorting; Partial sorting; reconfigurable computing.

I. Introduction

Parallel algorithms for data sorting have been studied in computer science for decades. There are many different parallel sorting algorithms. The most notable of them are Parallel QuickSort, Parallel Radix Sort, Sample Sort, Histogram Sort [1] and a family of algorithmic methods known as sorting networks [2]. The latter presents a great interest for hardware acceleration. A sorting network is a set of vertical lines composed of comparators that can swap data to change their positions in the input multi-item vector. The data propagate through the lines from left to right to produce the sorted multi-item vector on the outputs of the rightmost vertical line. Sorting is a very computationally expensive and time consuming operation which requires a lot of hardware resources. There are different approaches to overcome these limitations. Utilizing iterative networks with reusable comparators permits to process significantly larger data sets. Another two possibilities to overcome these problems are utilization of a relatively small parallel sorter along with a merging circuit or implementation of partial sorting. In this paper we investigate the latter possibility.

Many applications do not require all inputs to be sorted. Some of them necessitate only maximal and minimal values to be selected. Many electronic, environmental, medical, and biological applications need to process data streams produced by sensors and measure external parameters within given upper and lower bounds (thresholds). Maximum and minimum subsets extraction is required in searching, statistical data manipulation and data mining (e.g. [3] [4]). To describe one of the problems from data mining informally let us consider an example [3] with analogy to a shopping card. A basket is the set of items purchased at one time. A frequent item is an item that often occurs in a database. A frequent set of items often occur together in the same basket. A researcher can request a particular support value and find the items which occur together in a basket either a maximum or a minimum number of times within the database [3]. Similar problems appear to determine frequent inquiries at the Internet, customer transactions, credit card purchases, etc. requiring processing very large volumes of data in the span of a day [3]. Fast extracting the most frequent or the less frequent items from large sets permits data mining algorithms to be simplified and accelerated. Sorting of subsets may be involved in many known methods from this area.

The paper suggests a method and high-performance hardware implementation of a partial sorter for maximum (or minimum) subset extraction and maximum (or minimum) unsorted subset selection based on parallel sorting network. The system is designed for working over streaming data. It utilizes AXI interfaces and is suggested as a PCI express peripheral.

The remainder of the paper contains 6 sections. Section II analyzes the related work. Section III describes highly parallel networks for sorting and explores hardware co-design. Section IV describes experimental setup and hardware accelerator architecture. Section V presents the results of experiments and comparisons with known alternatives. The conclusion is given in Section IV.

II. Related Work

The problem of finding subsets of minimum and maximum values is well known, but very small number of solutions exist. The majority of works in this area are focused on finding 1 or 2 maximal or minimal values in data sets ([5] [6]), but only few works are focused on subsets.

Frarmahini-Farahani et al. investigated the problem of partial sorting and max-set-selection in [7]. They proposed a modular design of a partial sorting system based on Batcher's odd-even and bitonic sorting networks. Their system is built on sorting blocks constructed from Batcher's odd-even merge (OEM) and bitonic sorting networks (BM), where bitonic sorters are reduced in order to get sorted maximal (or minimal) subset. They also proposed an approach to select unsorted maximal subset by replacing bitonic sorters with maximum selection units. In theory this technique is extendable to 2^n-to-2^m size (where m<n). Also they proposed an architecture for iterative max selection units that can potentially work with data streams. Another solution of this problem was developed by Biroli and Wang in [8]. Their approach is not based on sorting networks, but still uses parallel comparators. They applied fast circuit topologies for single max/min value search by Goren et al. [5] to find a subset of the largest or smallest values. In contradiction to Frarmahini-Farahani they didn't use Batcher's networks. Both works focused on finding relatively small subsets. The work by Frarmahini-Farahani is more suitable for work with large subsets, but its expansion will lead to large resource consumption.

In our previous works we also explored minimum and maximum extractors. In [9] and [10] we proposed different hardware/software methods for simultanious minimal and maximal subset and implemented them on Zynq-7000 APSoC devices. In [11] we 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.

III. Computing Sorted Subsets

As it was suggested in [9], some practical applications don't require maximal and minimal subsets simultaniously. For this purpose a reduced partial sorter that contains one main and one additional sorting network was discussed. In this paper we furher investigate this suggested approach because of its applicability and possible comparison with known alternatives. Additionally it is a pure hardware implementation which does not require embedded processing system like in our previous projects and it is designed as a hardware accelerator with PCI-express interface for ease on practical usage over streaming data.

Let set S containing N M-bit data items be given. The maximum subset contains L_{max} largest items in S. We mainly consider such tasks for which $L_{max} \ll N$ which are more common for practical applications (also applicable to a minimum subset). Since N may have very large value (millions of items) it cannot be processed completely in hardware due to the limited resources. It is shown in [12, 13] that even for relatively complex Field-Programmable Gate Arrays (FPGAs) the size N is limited. For example, for even-odd merge and bitonic merge networks [2] N cannot exceed a few hundreds of 32-bit items even for very advanced FPGAs (such as the largest devices from the Xilinx Virtex-7 family).

As a basis for our sorting circuit we use a periodic pipelined Odd-Even Transition Sorting Network (also known as Odd-Even Transposition Sorting Network or OETS). A periodic network is a type of network which consists of identical sequences of comparators. Traditional implementation of OETS is less efficient than Batcher's networks, but it is more reliable and its implementation is simpler. Salloum and Wang proved that OETS has good fault-tolerant properties [14].

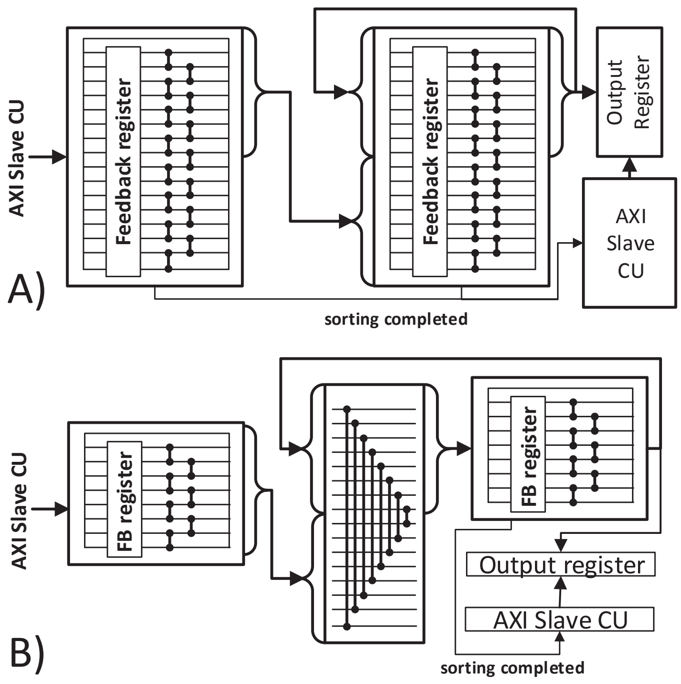

Fig. 1. Proposed methods for partial sorting circuits.

Like in our previous works, we use pipelined approach with reusable comparators presented in [13]. K M-bit data items that have to be sorted are loaded (from block RAM) to the feedback register (FR). Sorting is executed in a segment of even-odd transition network composed of two linked lines with even and odd comparators. Sorting is completed in K/2 iterations (clock cycles) at most. Note, that almost always the number of iterations is less than K/2 because of the technique [13] according to which if there are no swaps of data on the right-most line of comparators then sorting is completed. Note that the network [13] possesses significantly smaller combinational delays than networks from [2]. Besides, in the proposed architecture iterations are done at the same time as subsequent data are being received from the PS. Such parallelism enables delays to be optimally adjusted allowing the total performance to be improved.

Two methods are depicted in Fig. 1. The first method utilizes two sorting networks of the same size. The first sorting network receives blocks of data and sorts them. After the sorting is completed, the maximal (or minimal) half loads into the second sorting network along with maximal (or minimal) half of outputs of the second network. For maximal set selection, in the initial step the second network is loaded with zeros. For minimal set selection, it is loaded with maximal possible value. After all the data is transmitted, the system waits for the completion of sorting in both sorting networks. The maximal (or minimal) half of the outputs of the second network is loaded in the output register and waits for read request.

The second method is also based on sorting networks, but these networks are two times smaller than in the first method,

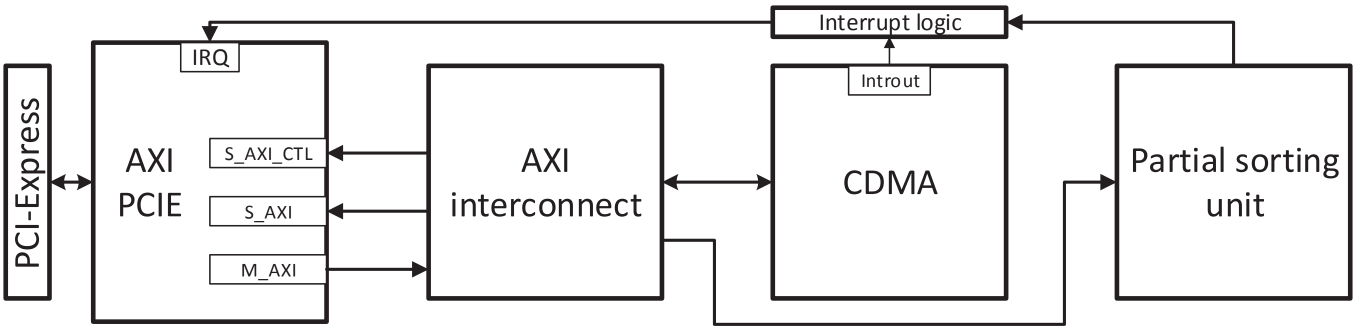

Fig. 2. Basic hardware architecture.

because we don't need here to sort results of the previous iteration with results of the current iteration. Both networks are connected here with a swapping network. All outputs of the first sorting network are connected to the swapping network along with all outputs of the second sorting network. On the outputs of the second network we receive unsorted maximal and minimum subsets of the input data, where all items of the upper half of the network are larger than all items of the lower half. In some practical applications receiving sorted maximal and minimal subsets is not required and only unsorted ones are needed. In that case we can turn the second sorting network off during the last iteration of the algorithm. It will reduce execution time which will be noticeable for relatively small amounts of data.

IV. Hardware Architecture and experimental setup

The system was designed as a hardware accelerator for a host PC which communicates through PCI-express interface in Direct Memory Access (DMA) mode. Fig. 2 depicts this architecture.

Software in the host PC runs the 32-bit Linux operating system (kernel 3.16) and executes programs (written in C language) that take results from PCI-express (from the FPGA) for further processing. We assume that the data collected in the FPGA are preprocessed in the programmable logic by applying various highly parallel networks (see Section III), and the results are transferred to the host PC through the PCI-express bus. To support data exchange through PCI-express, a dedicated driver was developed. The programmable logic uses the Intellectual Property (IP) core of the central direct memory access (Xilinx CDMA) module to copy data through AXI PCI express (Xilinx AXI-PCIE). Data transfer in the host PC is organized through direct memory access (DMA). To work with different devices, a driver (kernel module) was developed. The driver creates in the directory /dev a character device file that can be accessed through read and write functions, for example write(file, data array, data size). The PC BIOS assigns a number (an address) to the selected base address register (BAR) and a corresponding interrupt number that will be later used to indicate the completion of a data transfer. As soon as the driver is loaded, a special operation (probe) is activated and the availability of the device with the given identification number (ID) is verified (the ID is chosen during the customization of the AXI-PCIE). Then a sequence of additional steps is performed (see [15, pp. 302-326] for necessary details). A number of file operations are executed in addition to the probe function. In our particular case, access to the file is done through read/write operations.

V. Experimental results and comparison

All hardware solutions were implemented, evaluated and tested in Xilinx Virtex-7 XC7VX485T FPGA. The implementation in this paper was designed with an aim to compare it with known alternatives. We compared it with software sorting and a hardware solution from [7] (OEM/BM). Software solution is the most obvious and the most widely used quicksort implementation from C++ language (sort function). With this approach a whole data set is being sorted with subsequent extraction of the maximal (or minimal) subset. For comparison in hardware area, the system from [7] was implemented. After some experiments we found the most optimal configuration for implementation for Virtex-7 device which extracts 128-item data sets. Any implementation for extracting 256-item data sets utilizes more than 100% resources of the device. We implemented suggested in the paper concept of iterative max-set-selection units. The basis of this system is constructed from the two following blocks: 256-to-128 odd-even merge max-selection units and reduced bitonic 256-to-128 unit which starts with core max-selection unit. Inputs for core max selection units are outputs of OEM 256-to-128 and outputs of BM sorter (which contains results from the previous iteration).

For our methods we implemented two different systems. One for finding 128-item data subset in order to compare with OEM/BM method, and another for finding 1024-item data sets which is the maximal possible circuit that fits in Virtex-7 device. Post-implementation resource usage is shown in Table 1.

TABLE I. RESOURCE UTILIZATION

Method	Resources	
	FF	*LUT*
Method A 128	9%	22%
Method B 128	8%	19%
Method A 1024 (max)	38%	94%
Method B 1024 (max)	22%	70%
OEM/BM 128 (max)	52%	78%

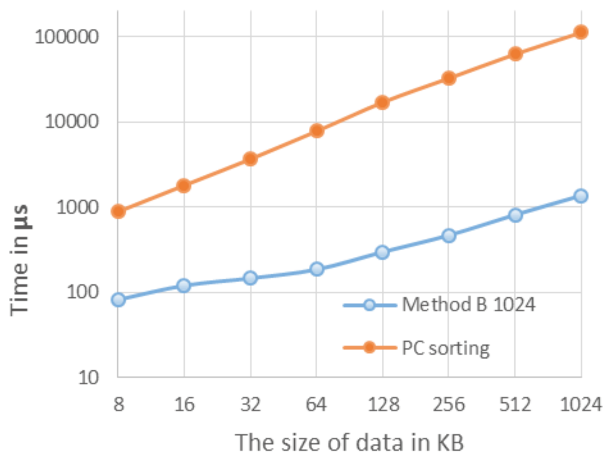

Fig. 3. Experimental results.

Lookup table (LUT) usage for the method A is 3,5 times smaller and for the method B is 4 times smaller than OEM/BM based solution. The method A requires 5,7 times fewer amount of flip-flop (FF) than OEM/BM and the method B requires 6,5 times fewer FFs. Also it is necessary to mention that all modules required for PCIe DMA system utilize about 15% of LUTs. By subtracting these resources we see that pure min/max system for the method A requires 9 times fewer LUTs and the method B requires 15,7 times fewer LUTs.

Available resources of Virtex-7 device allow us to expand our circuits for extracting larger maximum or minimum subsets. Both proposed architectures were expanded to extract subsets of 1024 items which is 10 times more than with OEM/BM approach. Although for simultaneous extracting of maximum and minimum subsets both proposed methods are identical in terms of resource usage and performance, the method B is better for extraction of maximum or minimum subset alone.

Fig. 3. shows experimental results. With Virtex-7 and the proposed PCI express transfer system all hardware implementations showed approximately identical results. With architectures that allow faster data transfer OEM/BM approach may show better results, because for the proposed methods A and B worst case performance is K/2 clock cycles for K inputs and OEM/BM performance is dependent on the number of pipeline stages. But because of significant economy of resources with the proposed methods (especially the method B) it is possible to speed up sorting by placing two or more instances of the sorting circuit that will sort parts of the whole data simultaneously.

VI. Conclusion

The paper suggests hardware-based methods of partial sorting for computing the maximum and minimum subsets of large sets of streaming data. The proposed solutions are highly parallel permitting capabilities of programmable logic to be used very efficiently. All the proposed methods were implemented in commercial microchips, tested, evaluated, and compared with alternatives. The results of experiments have shown significant advantages over other software and hardware solutions.

Acknowledgment

This research was supported by the institutional research funding IUT 19-1 of the Estonian Ministry of Education and Research, the Study IT in Estonia Programme.

References

[1] E.V. Kalé, E. Solomonik, "Sorting," in Encyclopedia of Parallel Computing, Springer Science+Business Media, 2011, pp. 1855-1862.

[2] S.W. Aj-Haj Baddar, K.E. Batcher, Designing Sorting Networks. A New Paradigm., Springer, 2011.

[3] Z.K. Baker, V.K. Prasanna, "An Architecture for Efficient Hardware Data Mining using Reconfigurable Computing Systems," in Annual IEEE Symposium on Field-Programmable Custom Computing Machines, Napa, USA, 2006.

[4] X. Wu, V. Kumar, J.R. Quinlan, et al., "Top 10 algorithms in data mining," Knowledge and Information Systems, vol. 14, no. 1, pp. 1-37, 2014.

[5] S. Goren, G. Dundar, B. Yuce, H.F. Ugurdag, "A fast circuit topology for finding the maximum of N k-bit numbers," in Symp. on Computer Arithmetic, 2013.

[6] C. Wey, M. Shieh, S. Lin, "Algorithms of finding the first two minimum values and their hardware implementation," IEEE Trans. Circuits and Systems I, vol. 55, no. 11, p. 3430–3437, 2008.

[7] A. Farmahini-Farahani, H.J. Duwe, M.J. Schulte, K. Compton, "Modular design of high-throughput, low-latency sorting units.," IEEE Transactions on Computers, vol. 62, no. 7, pp. 1389-1402, 2013.

[8] A.D.G Biroli, J.C. Wang, "A fast architecture for finding maximum (or minimum) values in a set. In Acoustics,," in 2014 IEEE International Conference on Speech and Signal Processing (ICASSP).

[9] V. Sklyarov, I. Skliarova, A. Rjabov, A. Sudnitson, (2015). Zynq-based System for Extracting Sorted Subsets from Large Data Sets. Informacije MIDEM, 45(2), 142-152.

[10] V. Sklyarov, A. Rjabov, I. Skliarova, A. Sudnitson, (2016). High-performance Information Processing in Distributed Computing Systems. International Journal of Innovative Computing, Information and Control, 12 (1), 139−160.

[11] A. Rjabov, V. Sklyarov, I. Skliarova, A. Sudnitson, (2015). Processing Sorted Subsets in a Multi-level Reconfigurable Computing System. Elektronika ir Elektrotechnika, 21(2), 30-33.

[12] R. Mueller, J. Teubner, G. Alonso, (2012); Sorting networks on FPGAs, The VLDB Journal—The International Journal on Very Large Data Bases, 21(1), 1-23.

[13] V. Sklyarov, I. Skliarova,. (2014); High-performance implementation of regular and easily scalable sorting networks on an FPGA, Microprocessors and Microsystems, 38(5): 470-484.

[14] S.N. Salloum, D.H. Wang,, "Fault tolerance analysis of odd-even transposition sorting networks with single pass and multiple passes," in IEEE Pacific Rim Conference on Communications, Computers and signal Processing, 2003.

[15] J. Corbet, A. Rubini, G. Kroah-Hartman, Linux Device Drivers, http://lwn.net/Kernel/