Low Power Sorters Using Clock Gating

[1]Preethi, [2]K.G Mohan, [3]Sudeendra Kumar K, [4]K. K. Mahapatra
preethisrivathsa@gmail.com, mkabadi@gitam.edu, kumar.sudeendra@gmail.com, kmaha2@gmail.com
[1]Presidency University, Bangalore, [2]GITAM University, Bangalore, [3]PES University, Bangalore, [4]NIT, Rourkela

***Abstract*- Sorting is a very important task which is widely used in several applications like signal processing and database management. The importance of sorting has increased significantly in modern data center applications serving the applications of Cloud computing and Internet of Things. Sorting which is generally implemented in software on CPU or GPU, which takes several cycles to finish the sorting process. The further improvement in performance in sorting is possible through hardware implementation either in FPGA or ASIC. The performance improvement and reducing the power consumption are the dominant concerns. The conventional sorting techniques like Bubble sort, bitonic sort and odd-even sort are found suitable for hardware implementation in the research literature. There are several endeavors from researchers to make these sorting techniques more modular and low power, which is required to design large scale sorting for data center-based applications. In this paper, we investigate application of generic and structured low power technique like clock gating in designing the low power sorters. The bubble sort, bitonic sort and odd-even sorting techniques are redesigned to make them low power using clock gating technique. The implementation results show that, the clock gating reduces the dynamic power consumption on sorters by 47.5% without much impact on the performance. The power reduction results obtained are comparable with state-of-the-art low power sorters which are complex in design. The proposed sorters are implemented and results are presented for Saed90nm standard cell libraries.**

Keywords: **Low Power Design, Sorting architecture, Clock gating.**

I. INTRODUCTION

Successful implementation of Internet of Things (IoT) mainly depends upon the effective data crunching and deriving the meaningful conclusions from the collected data with minimum latency. The data crunching necessities has led to emergence of big data analytics. Big data analytics connected with modern businesses occupy a huge number of database tuples and the computer systems handling big data analytics in the IoT era require high performance computation ability which is generally achieved through multi-core processing systems, which has led to new research in heterogeneous computing. Sorting is an important operation in classical database systems and also in modern big data analytics [1]. Sorting operation is required in wide range of applications including data mining, digital signal processing, scientific computing, robotics, high energy physics etc [2] [3]. Sorting can be implemented in both software and hardware [4]. Most of the modern day IoT applications demand high performance big data analytics and to achieve required speed of operation, data intensive operations like sorting are implemented in hardware. Researchers in big data analytics found that sorting operations consume an excessive number of CPU cycles in software implementation [4]. Acceleration of Sorting operations to achieve high speed sorting are often implemented using either a dedicated ASIC or FPGAs to meet performance requirements [4] [5]. Hardware sorting units vary in their architecture based on target applications and number of inputs and width of inputs. Design of efficient sorting techniques for hardware implementation has attracted researchers in both industry and academia [5] [6]. Most of the sorting techniques proposed in recently primarily focus on improving the computational efficiency. Majority of sorting techniques implemented in ASIC or FPGA use odd-even sorter, bitonic sorter, bubble sort and merge sort [6]. Hardware implementation of sorting are efficient than software implementation for short sequences whose length is known a priori. We can find good amount of research literature on FPGA implementations of sorting techniques and researchers are attracted to work in novel FPGA implementation of sorting techniques, because it does not require control flow instructions and it is easy to parallelize the sorting techniques on reconfigurable hardware [7] [8] [9]. The servers or computer systems in data centres has changed drastically in last one decade. Several computing elements like multi-core processors, GPUs, hardware accelerators and signal processors are part of modern-day data centres catering to different application verticals. These specialized computing elements in data centres handling applications like sorting deliver the required performance and decrease the computational latency and response times [10] [11]. Another important open research problem in modern day data centres is reducing the power consumption in computing infrastructure and associated thermal cooling requirements [1] [11] [12]. Due to the growth in the use of internet, the demand placed on data-centre has increased and high-performance computing techniques has also increased the power consumption. The researchers are focusing on designing the new techniques which are low power, without compromising the performance in the data centre-oriented applications like sorting. It is very common today to use the FPGA based or heterogenous computing models like CPU-GPU and CPU-FPGA models in data centres. Even though FPGA based computing systems for sorting are attractive and has got several advantages like reconfigurability, easy implementation of parallelism for sorting etc, they have following disadvantages: -

- Designing low power computing elements without compromising the performance is limited in FPGA's as already device is fabricated in comparison with ASIC implementations.
- In ASIC designs, we can manage power-performance trade-off using low power management techniques like power gating and dynamic voltage frequency

978-1-7281-8753-2/21/$31.00 ©2021 IEEE
DOI 10.1109/iSES52644.2021.00015

scaling (DVFS) which may not supported in all available FPGA devices.

Currently, data centres are already deployed large number of FPGAs to achieve the required performance and power consumption has become a serious bottleneck for further upgrades [8] [11]. Eventually, FPGA driven servers will be replaced by ASIC based servers eventually, which is necessary to balance the power and performance requirements [1] [11].

As we discussed above, sorting engines are one of primary computing elements in data centres and most of the researchers focused on designing the sorting techniques on FPGA devices. The amount of research work on ASIC implementation of sorting engines is relatively less in comparison with FPGAs. There is a need to design low power sorting engines which reduces the power consumption without compromising the performance which will be an attractive solution to data centres. In this work, we primarily focus on designing the low power sorting engines and compare the low power implementation of widely used sorting techniques. The standard and structured low power digital design techniques like clock gating, power gating and Dynamic Voltage Frequency Scaling (DVFS) can be applied on widely used sorting engines like bitonic sorting, odd-even sorting and bubble sorting and investigation of design trade-off between power and performance of different low power sorting engines is an open research problem. In this work, we primarily focus on clock gating to achieve low power in sorting engines and we present the power performance trade off results for three widely used sorting techniques.

This paper is structured as follows: next section presents the literature survey on existing ASIC and FPGA implementation of different sorting engines and brief literature review on standard sorting techniques used in this work. Section III discusses about standard and structured low power techniques widely used in low power design and their importance. Section IV discusses the implementation of low power sorting using the standard low power techniques and Section V presents the results, comparison and analysis. Finally, Section VI concludes the paper.

II. Background

Sorting has attracted the researchers all along from last four decades [13]. The computational complexity of well-known sorting techniques is presented in Table I. It is evident from the literature survey that fastest sorting methods are based on odd-even merge and bitonic merge networks [5] [6] [14]. Performance is critical and most of the sorting architectures proposed in last four decades primarily focused on reducing latency and increasing performance in terms of speed of operation. The sorting network which sorts 'N' data items will have a data dependent step from S0……. Sk….SN-1 will get executed sequential manner. Sorters can be designed and implemented either using pure combinational circuits or can be designed as sequential circuits with sorter datapath. In the pure combinational sorters, circuits operated at any step 'k' requires the result of previous step 'k-1'. The total delay of such combinational sorter will be 'N', for equal amount of propagation delay at all stages of the sorter. In clocked sorters, the delay of the circuit is determined by calculating the number of clock cycles consumed to produce the final result. Sequential Sorters are good candidates for pipelining and parallel processing.

TABLE I
Algorithms and Complexity

	Best case/Average case	Worst case
Bubble Sort	O(n)	$O(n^2)$
Heap Sort	O(nlogn)	O(nlogn)
Insertion Sort	O(n)/ $O(n^2/4)$	$O(n^2)$
Merge Sort	O(nlogn)/O(nlogn)	O(nlogn)
Quick Sort	O(n)/O(nlogn)	$O(n^2)$
Selection Sort	$O(n^2)$/ $O(n^2)$	$O(n^2)$

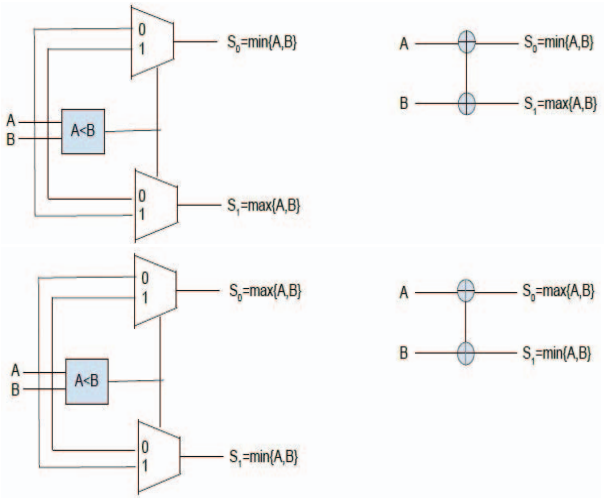

Figure.1: Representation of compare and swap units used in sorters

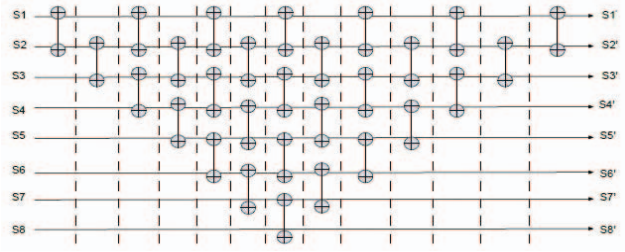

Figure.2: Bubble Sorter

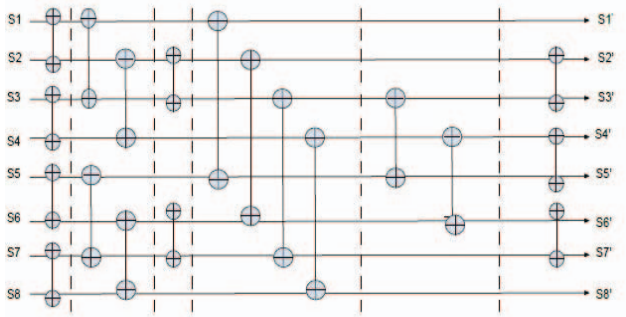

Figure.3: Odd-Even Sorter

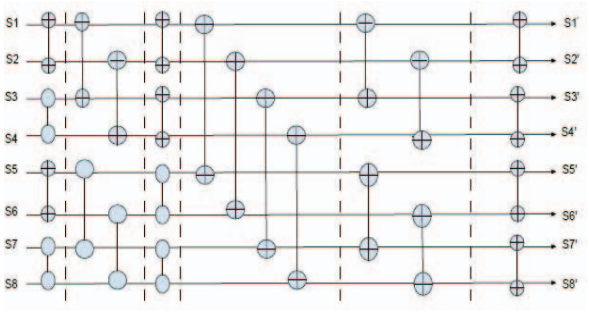

Figure.4: Bitonic Sorter

Combinational and Sequential Sorters: -

To handle the large random data sets, bitonic and odd-even merge sorts are widely employed [14] [15] and as the size of the data increase, the number of comparators and other hardware resources required are enormous for combinational implementations. It is practically impossible to implement such a large vertical array of comparators either in ASIC or FPGA. Sequential implementations of odd-even sorter/bitonic sorter reduces the hardware resources considerably, but increases the number of transactions/clock cycles. To further decrease the latency/clock cycles consumed, pipelining and parallel processing are employed. It is well known and it is established in research that, bitonic and odd-ever sorter are suitable to design large scale high speed pipelined sorting networks [6].

Sorting Techniques: - In this section, we briefly present the conventional sorting techniques like bubble sort, bitonic sort and odd even sort. Hardware Implementation of sorting modules primarily composed of series of compare and swap units as shown in figure 1. The two values are compared using a comparator and the result is applied to the two multiplexers that select the values to generate the output in an ascending or descending order. The two symbols + and – used to represent the ascending and descending units respectively.

Bubble sort: - For a small set of inputs, it is feasible to implement the bubble sort in hardware. The smallest or largest number is selected in each round through a series of comparisons. The bubble sort algorithm requires M comparisons and switching events in first stage with an input size of M. Similarly in second stage, it requires M-1 comparisons and switching events and so on until the completion of the sorting process. The run time complexity is $O(M^2)$, in which 'M' is the input size. The figure 2 shows the parallel bubble sorting network in which compare operations can be executed sequentially by applying pipelining.

Odd-even sort: - Batcher proposed the odd-even (OE) merging sort by merging the two sorted sequences into a complete sorted result. An M – input odd even merging unit is represented by OE-M, where 'M' should be the power of two. Sorted output can be generated by constructing the series of parallel merging units from OE-2s, OE-4s…to OE-M. When the input dataset consists of 2P samples, there will be 2P-1 OE-2s in the first stage, 2P-2 OE-4s in the second stage and so on. The figure 3 illustrates the eight-input odd even merge sorting network.

Bitonic sort: - Bitonic sorting is also proposed by Batcher in [14]. Bitonic sorter is constructed by using the one sequence in increasing order and another one in decreasing order. The bitonic merging unit merge the two sequences with equal length into sorted output. The input size of the bitonic merging unit must be in a power of two. The 'M' sorter unit receives an ascending and descending sequence, and both the sequences contain 'M/2' samples. It is called BM-M, where M is represented as 2^P. To build the complete 2^P input bitonic sorting circuit, a series of bitonic merging units is applied recursively to generate the sub-sequence. The 2^P input bitonic sorting network comprises of 2^{P-1} parallel BM-2s in first stage and 2^{P-2} BM-4s in the second level and so on. The figure 4 shows the Bitonic sorting network for eight inputs.

Earlier ASIC implementation of Sorting Algorithms: -

One of the earliest ASIC implementations of odd-even sorting can be found in [13]. In [13], Odd-even sorter circuit is implemented at transistor level with both serial comparator and parallel comparator. The paper [6] discusses the implementation of different sorting algorithms in ASIC and analyze the area, timing and memory requirements holistically. In [6], fully pipelined bit-serial architecture for sorting is proposed and it consumes (n + 1) k cycles to sort a sequence of n, k-bit numbers. Another important ASIC implementation of binary sorting engine implemented in full custom method is presented in [18]. It is shown that, the computational complexity of the sorter is (n + k -1), where 'n' is the number of elements with k-bit size. Further improvements to [18] are proposed in [13], which describes a hardware implementation which consumes less area for large number of datasets by taxing the latency reasonably. Design is implemented in full custom method in 65nm process and this work is an extension to the earlier techniques.

Modular Techniques for ASIC Implementation of Sorting Algorithms: -

It is difficult to design large scale sorting networks from transistor level using full custom methods. Large scale sorting networks are designed using modular design techniques that hierarchically design large sorting units from smaller building blocks. The sorting units are composed of small and regular design blocks connected with modular fashion which simplifies the designing of large-scale sorting circuits. The modular approach of designing sorting networks will make adopting pipelining and low power schemes easy [14]. The modular approach proposed in [14] is extended in [15] to achieve low power. Dynamic power consumption is the major part of power consumption in CMOS designs. Reducing the switching activity will reduce the dynamic power consumption. Switching activity increases when the input dataset increases or bit width of the input is large. With the increase in bit-width, the compare and swap action in sorting network increases the switching activity enormously, which increases the dynamic power consumption [19]. To reduce the dynamic power, an immobile comparing unit which move the indexes of samples instead of moving the actual input data is proposed in [15] and this technique reduces the power significantly. Different kind of modular approach is presented in [16], which sorts the data on the fly without any comparison operation and number of clock cycles consumed is linearly proportional to the number of inputs, with a speed complexity of order of O(N). Unary processing based bitonic

sorting network is proposed in [17]. The authors of [17] present the results that, unary processing based bitonic sorting network reduces the hardware cost and power consumption in comparison with conventional bitonic sorting network. The detailed comparison of the above-mentioned techniques is presented in Table II.

Challenges in designing large scale sorting networks: - It is evident from literature survey that, the major challenges in designing large sorting networks are: -

- The most important challenge is the long critical paths which increase with the input bit-width. The long critical paths impact the operating frequency and decrease the overall performance. Pipelined implementation of sorters will reduce the critical paths up to an extent [14].
- The pipelined implementation increases the switching activity significantly and make the design power hungry. There is a need to decrease the power consumption by adopting the suitable low power techniques.
- In the existing literature, application of structured well established low power techniques like clock gating are not applied on hardware implementation of sorting techniques and trade-off between power and performance is not investigated.

III. STRUCTURED LOW POWER TECHNIQUES

Clock gating: - Generally data is loaded into the registers infrequently in most of the designs and clock signal is fed continuously. Clock signal will continuously toggle at every pre-defined cycle. Clock signal drives the large capacitive load making clock as a major source of switching power dissipation. Gating a group of flip-flops in the design, which are enabled by same control signal reduces the unnecessary toggles on clock. Power is not dissipated during the idle period when the register is switched off by gating function. Dynamic power consumption is saved in the gated clock network. The clock to the register files of an unused module is turned off using an extra AND gate, which is controlled by an enable signal. Extra AND gate and enable control signal can be introduced either in RTL or in the gate level netlist using scripts. Clocking gating can be implemented without disturbing the functionality of the circuit through automation [19].

TABLE II
Comparison of Low Power Sorting Techniques

Paper	Technique	Technology Node and Implementation Methodology	Remarks
Modular Sorter, 2013 [14].	The sorting units are optimized for situations in which only the M largest numbers from N inputs sorting are required. Traditional bitonic and odd even sorting techniques are modified to achieve the requirement.	The proposed design is synthesized using 65nm standard cell library. Area and Timing analysis report is presented for different input configurations. (Semi-custom design)	Power Analysis is not reported for the proposed sorting network.
Low Power Sorter, 2017, [15]	The work presented in [15] is extended in this work to reduce the power consumption. Dynamic power is reduced by redesigning the modular sorting network by introducing the pointer like design in which indexes of samples are moved instead of input data.	The proposed design is synthesized using 90nm standard cell library. Dynamic power analysis reports are presented for different input configurations and compared with earlier technique [14]. Area and Timing analysis reports are also presented for different input configurations. (Semi-custom design)	Power analysis reports shows that, with increase in bit-size of the input, the improvements in power analysis is minimal and improvements in power reduction presented for lower frequencies at higher bit-sizes.
Comparison Free Sorter, 2017 [16]	A new sorting algorithm which can sort the input data integer elements on the fly without any comparison is proposed and implemented. The computational complexity of the proposed technique is of the order O(N), which is a significant contribution.	The proposed design is implemented from transistor level using 90-nm technology. The transistor count, timing and power results are presented. (Full custom design)	As the implementation is done at transistor level, it is difficult to scale it for large input bit sizes. For large input bit-sizes, the dynamic power and latencies may be high for the presented circuit, unless it is properly optimized during circuit design.
Unary Processing based Low Power Sorter, 2018 [17]	Low cost and low power sorting technique is proposed in [17] is based on unary processing. Unary processing reduces the wiring between compare and swap units in the large sorting networks	The proposed design is synthesized using 45-nm standard cell library and area, power and timing results are presented for 8,16 and 32-bit data widths. (Semi-custom design)	Unary processing-based sorting networks reduces the power significantly for the higher bit-width at the cost of increased latency. Processing digital unary streams takes relatively long running time.

TABLE III
Comparison of Area, Power and Timing of Proposed and Conventional Sorters

Sorter	Bit Width	Area in square microns (Synthesized @ 50 MHz)		Critical Path @ 50 MHz Frequency (in nano seconds)		Dynamic Power Consumption(mW) (@50 MHz)		
		Conventional Pipelined Sorter	Proposed Clock gated Sorter	Conventional Pipelined Sorter	Proposed Clock gated Sorter	Conventional Pipelined Sorter	Proposed Clock gated Sorter	Reduction (%)
Bitonic Sort	8	36456.23	35345.87	6.545	6.776	1.3	0.585	46.59
	16	74234.45	75456.78	6.900	6.723	1.47	0.71	48.90
	32	145667.3	144589.8	7.423	7.733	1.88	0.912	51.83
Odd-Even Sort	8	32762.34	30231.89	5.845	5.812	1.39	0.641	52.55
	16	63298.75	61234.98	5.987	5.634	1.62	0.71	46.90
	32	125987.6	122889.9	8.412	8.854	2.1	0.912	47.02
Bubble Sort	8	40014.59	38234.88	6.598	6.734	1.37	0.651	49.53
	16	81298.56	79126.38	7.378	7.564	1.68	0.845	52.48
	32	178238.3	174398.5	8.231	8.432	2.08	1.101	52.65

TABLE IV
Dynamic Power Consumption of Proposed and Conventional Sorters

Sorter (32-bit)	Operating Frequency (MHz)	Dynamic Power Consumption (mW)		
		Conventional Pipelined Sorter	Proposed Clock gated Sorter	Reduction (%)
Bitonic Sort	20	1.76	0.82	46.59
	40	1.82	0.89	48.90
	60	1.91	0.99	51.83
	80	1.96	1.03	52.55
Odd-Even Sort	20	1.94	0.91	46.90
	40	2.02	0.95	47.02
	60	2.14	1.06	49.53
	80	2.21	1.16	52.48
Bubble Sort	20	1.88	0.99	52.65
	40	2.06	1.05	50.97
	60	2.18	1.12	51.37
	80	2.23	1.24	53.91

TABLE V
Comparison of Dynamic Power consumption of proposed and existing schemes

Bitonic Sorter	Percentage of dynamic power reduction in comparison with conventional Bitonic Sorter (Unary processing) [17]	Percentage of reduction in comparison with conventional Bitonic Sorter (Low Power) [15]	Percentage of reduction in comparison with conventional Bitonic Sorter with Proposed Clock gated Bitonic Sorter
32-bit	89%	37.01%	48.51%

IV. IMPLEMENTATION AND RESULTS

Sorting networks are generally combinational circuits and slow in operation. To speed up the operation, we implemented basic pipelining of sorting networks as described in [14]. Sorting circuits will have large number of comparators between inputs and outputs. Pipelining is implemented in sorting network by ensuring the clocked register/pipeline stage has only one comparator between its input and output. Pipeline stages will increase with size of the input and 8-element sorting network needs a 6-stage pipelining, 16-element sorting network needs 10 pipelining stages and 32-bit sorting network needs 15 stage pipelining and so on. In this work, we implemented

pipelined versions of bitonic sorting, odd-even sorting and bubble sorting in Verilog and simulated in Cadence NCSim. The designs are synthesized in Cadence Genus Synthesis tools using saed90nm library and synthesized gate level netlist is verified in Cadence NCSim. Clock gating is implemented using PERL scripting and clock gated netlist is verified for functionality in Cadence NCSim.

Result Analysis: -

The experiments are performed to evaluate the performance of the different clock gated sorter architectures. The area, timing and power results for 8-, 16- and 32-bit widths are presented in Table III. The proposed clock gated sorting architectures show the power reduction of 47.5% at an operating frequency of 50 MHz. The advantage of clock gating technique is, it does not add much penalty on area or timing. The proposed clock gated sorting architectures are analysed for different clock frequencies and result is presented in Table IV. The average reduction of dynamic power is 50.2% between 20 MHz and 80 MHz for a Saed90nm standard cell library.

Comparison: - Comparative results for reduction in dynamic power for relevant technique (bitonic sorter) is presented in Table V. Both [15] and [17] are based on bitonic sorting. The technique proposed in [17] reduce the dynamic power significantly up to 89% for 32-bit sorter. The proposed Clock gated 32-bit sorter reduces the dynamic power by 48.51%, which is better than the low power sorter presented in [15]. The technique presented in [17] is based on unary processing, which needs a pre – processing and post processing modules. The dynamic power calculation does not include the pre and post processing modules in dynamic power calculation.

V. Conclusion

Software implementation of sorting on CPUs and GPUs consume more machine cycles and further performance improvement in sorting applications require the hardware implementation in FPGA or ASIC. The suitable sorting techniques for hardware implementation are bitonic sorting, bubble sorting and odd-even sorting. The improvements in performance and reduction in power consumption are primary concerns in designing sorting techniques for hardware implementation. In this paper, we have used well established clock gating technique to achieve low power consumption on three different hardware sorting techniques and trade-off between area, timing and power consumption is analysed. The clock gated sorting architectures are compared with state-of-the-art recent low power sorting schemes and comparative results are presented. The proposed clock gated sorters reduce the dynamic power consumption by 47.5% in comparison with conventional (non-clock gated) sorting circuits. The clock gating technique on sorters is also analysed for range of frequencies and it is found that, the dynamic power consumption decreases in proposed clock gated sorters incrementally with increase in frequency of operation. The proposed clock gated technique reduces the dynamic power consumption significantly in three widely used sorting techniques in comparison with earlier techniques.

References

[1] M. Dayarathna, Y. Wen and R. Fan, "Data Center Energy Consumption Modeling: A Survey," in *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 732-794, First quarter 2016

[2] D. E. Knuth, The Art of Computer Programming, vol. 3, Sorting and Searching. Reading, MA, USA: Addison-Wesley, 1998.

[3] K.E. Batcher, "Sorting Networks and Their Applications," Proc. AFIPS Proc. Spring Joint Computer Conf., pp. 307-314, 1968.

[4] R. Marcelino, H.C. Neto, and J.M.P. Cardoso, "Sorting Units for FPGA-Based Embedded Systems," Proc. IFIP Cong. Distributed Embedded Systems: Design, Middleware and Resources, pp. 11-22, Sept. 2008.

[5] Valery Sklyarov, Iouliia Skliarova, High-performance implementation of regular and easily scalable sorting networks on an FPGA, Elsevier Journal of Microprocessors and Microsystems Volume 38, Issue 5, Pages 470-484, 2014.

[6] S. Alaparthi, K. Gulati and S. P. Khatri, "Sorting binary numbers in hardware - A novel algorithm and its implementation," *2009 IEEE International Symposium on Circuits and Systems*, 2009, pp. 2225-2228.

[7] D. Koch and J. Torresen, "FPGA Sort: a high-performance sorting architecture exploiting run-time reconfiguration on FPGAs for large problem sorting," in Proc. of International Symposium on Field Programmable Gate Arrays, pp. 45–54, February 2011.

[8] A. Farmahini-Farahani, A. Gregerson,M. Schulte, and K. Compton, "Modular High-Throughput and Low-Latency Sorting Units for FPGAs in the Large Hadron Collider," Proc. IEEE Int'l Symp. Application Specific Processors, pp. 38-45, June 2011.

[9] M. Zuluaga, P. Milder, and M. Püschel, "Streaming sorting networks," ACM Trans. Design Autom. Electron. Syst., vol. 21, no. 4, Art. no. 55, May 2016,

[10] R. Chen, S. Siriyal, and V. Prasanna, "Energy and memory efficient mapping of bitonic sorting on FPGA," in Proc. ACM/SIGDA Int. Symp. Field Program. Gate (FPGA), Monterey, CA, USA, pp. 240–249, Feb. 2015.

[11] B. Falsafi, B. Dally, D. Singh, D. Chiou, J. J. Yi and R. Sendag, "FPGAs versus GPUs in Data centers," in *IEEE Micro*, vol. 37, no. 1, pp. 60-72, Jan.-Feb. 2017.

[12] Christian Bunse, Hagen Höpfner, Suman Roychoudhury, Essam Mansour: Choosing the "Best" Sorting Algorithm for Optimal Energy Consumption. ICSOFT (2) 2009: 199-206.

[13] C D Thompson, "The VLSI Complexity of Sorting", IEEE Transactions on Computers, Volume 32, Issue 12, pp 1171–1184, December 1983.

[14] A. Farmahini-Farahani, H. J. Duwe III, M. J. Schulte and K. Compton, "Modular Design of High-Throughput, Low-Latency Sorting Units," in *IEEE Transactions on Computers*, vol. 62, no. 7, pp. 1389-1402, July 2013.

[15] S. Lin, P. Chen and Y. Lin, "Hardware Design of Low-Power High-Throughput Sorting Unit," in *IEEE Transactions on Computers*, vol. 66, no. 8, pp. 1383-1395, 1 Aug. 2017.

[16] S. Abdel-Hafeez and A. Gordon-Ross, "An Efficient O(N) Comparison-Free Sorting Algorithm," in *IEEE Transactions on Very Large-Scale Integration (VLSI) Systems*, vol. 25, no. 6, pp. 1930-1942, June 2017.

[17] M. H. Najafi, D. J. Lilja, M. D. Riedel and K. Bazargan, "Low-Cost Sorting Network Circuits Using Unary Processing," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 8, pp. 1471-1480, Aug. 2018.

[18] Nozar Tabrizia, Nader Bagherzadeh, "An ASIC design and formal analysis of a novel pipelined and parallel sorting accelerator", Elsevier VLSI Integration Journal, Volume 41, Issue 1, Pages 65-75, January 2008.

[19] John M. Rabaey, "Low Power Design Essentials", Springer, 2009. ISBN: 978-0-387-71712-8.