DESIGNING AN EFFICIENT HARDWARE ACCELERATOR FOR DATA SORTING INTEGRATED WITH A RISC-V

Preethi

Assistant Professor, Department of Computer Science and Engineering, School of Engineering, Presidency University, Rajankunte, Bengaluru, Karnataka, India, 560064
preethisrivathsa@gmail.com
https://presidencyuniversity.in/school-of-engineering/computer-science/ms-preethi/

Dr K G Mohan

Professor, Department of Computer Science and Engineering, GITAM University, NH 207, Nagadenehalli, Doddaballapura, Karnataka 561203,
mkabadi@gitam.edu
https://blrgstcse.gitam.edu/faculty/profile/700334

Dr. Jacob Augustine

Professor, Department of Computer Science and Engineering, School of Engineering, Presidency University, Rajankunte, Bengaluru, Karnataka, India, 560064,
jacob@presidencyuniversity.in
https://presidencyuniversity.in/school-of-engineering/computer-science/dr-jacob-augustine/

Dr. Sudeendra Kumar K

Associate Professor, Department of Electronics and Communication, PES University, Banashankari, Bengaluru, Karnataka 560085
kumar.sudeendra@gmail.com
https://staff.pes.edu/nm1283

Abstract

In microprocessor architecture, amid a few blocks outcome of the optimized sorting algorithm has proved its impact on the results. Sorters can be implemented in domains that includes data centers, cloud computing servers for IoT applications. Sorters can be implemented on hardware, by deploying the developed sorter on Field Programmable Gate Array (FPGA). By contrasting factors like power consumption, implementation time, and implementation space with those of the proposed algorithm, it is possible to show the shortcomings of existing sorters like Bubble sort, Bitonic sort, and Odd-Even sort. This approves that the sorter with higher capability will perform better for sorting involving large numbers, this helps in designing of large-scale sorting for aforementioned applications. Few sorters were compared based on the parameters and it was concluded that comparison-free odd-even was having upper hand. Hence, the optimized sorter was implemented on the MicroBlaze, which is based on RISC-V architecture.

***Keywords*: FPGA, MicroBlaze, RISC-V, Sorters**

1. Introduction

FPGAs are now widely used in embedded system and successfully deployed in the applications like edge devices in Internet of Things (IoT), cloud computing and data analytics. Even though FPGA based applications see a huge difference in performance, still there is a thirst both from industry and academia. Henceforth, RISC-V finds its demand along with FPGAs. Specialty of Reduced Instruction Set Computer (RISC) principles are the foundation of the open standard RISC-V processor, which was developed in 2010. Commercial processors (x86)

are upward binary compatible, which means that if present processors add something, all subsequent processors must incorporate it [1]. There are only 47 instructions in RISC-V ISA(RV32I) considered as base of RISC-V. We can use the RISC-V specification to enhance the ISA with either conventional extensions or customized instructions [2].

When opposed to software operating on a general-purpose CPU, hardware acceleration is the utilization of computer hardware designed to perform specialized functions more effectively. One of the major operation performed in data center is searching. In order to perform searching effectively, the data must be sorted. Sorting operation can be carried out by software and hardware. Software sorting operation consumes more number of CPU cycles. Through hardware implementation we can improve the performance of sorters.

Data level parallelism can be achieved by Single-Instruction-Multiple Data (SIMD) instructions. Most of the modern processors use SIMD as accelerator at instruction level. RISC-V provides the flexibility of customizing instruction according to the application and on required accelerator devices. Customized instruction and new template designed for odd-even merge sort in [3] and also improves performance by optimizing SIMD instructions, resulting reduction in length of the code.

Over decades, soft-core processors have increased its popularity. They are processor hardware description models that can be synthesized using a number of semiconductor technologies such as Application Specific Integrated Circuit (ASIC), Field Programmable Gate Array (FPGA) or Programmable Logic Device (PLD). Soft-core processors give designers the flexibility to tailor their structures to a specific application in order to obtain the optimum design trade-offs. In [3], the performance improved by optimizing bandwidth of soft core's cache memory hierarchy.

FPGA found its application in big data for processing data unparalleled [4] and few of the FPGA drawbacks are listed bandwidth bottleneck for big data analytics, higher end FPGAs are expensive, disrupts communication [5], cache memory access in FPGA is random and effects the design [6]. Customizing SIMD instructions in [3] included novel instruction types, Verilog templates, open source framework and high throughput achieved by concentrating on cache memory and communication. [3] combines RISCV32I and RISCV32M enhancing performance of customized instructions. It provides efficient implementation of FPGA using BRAM.

A conventional sorting system consists of the following steps: data collection and acquisition, processing, dynamic and long-term storage, sorting, and dispatch. Many hardware solutions rely on linear sorting to maintain a sorted list with in-order insertions, but they don't optimize throughput, or the rate at which data elements are processed. The weakest link in the system, which is often the sorting step, limits the system's sorting throughput. Although hardware-based sorting can be accelerated by parallelism, most sorted results are still retrieved one at a time. Therefore, a hardware-based linear sorter could only achieve a maximum throughput of one sorted output per clock cycle regardless of the system's available and exploitable parallelism. Furthermore, specialized hardware such as priority schedulers may have diverse data processing and arrival times that must be taken into account if the system is to work in a pipelined fashion. Because pipelined systems can't move on to the next stage until the previous one is finished, pausing on a stage can possibly halt the pipeline's progress until data is available [29].

Other hardware sorting systems, such as Bitonic sorting networks [30, 31], are effectively rendered obsolete by this characteristic, as they can only obtain high throughput when acting on all data sets that are readily available. A unique sorter solution is required to enable low-latency pipelining and increased throughput through parallelism in a sorting system. There are a variety of sorting strategies available, including Bitonic sort, Odd-even sort, Bubble sort, Merge sort, and Insertion sort, each with its own set of benefits and drawbacks, which are detailed below [32].

Bubble sort is a simple and well-known method for sorting data [33] that offers a reasonable answer for a minimal number of inputs. When two numbers are compared, if they are not in the correct order, they will be swapped as needed. This procedure will be repeated many times, and this phase will be repeated throughout the entire sequence. It takes N comparisons in the first iteration for the input of size N, N-1 comparisons in the second iteration, and so on until all of the elements have been sorted. The complexity of this sorter is O(N(N-1)/2) [34]. This strategy has the advantage of using less memory and fewer lines of code to execute; however, the significant disadvantage is the longer time required, and it is apparent that the algorithm is unsuccessful for huge data sets.

For large-scale sorting, [35] suggested and successfully developed a tree-based merge sorter. The performance was found to be deteriorated due to the distorted data. A reliable strategy was required to deal with enormous data collections. It was accomplished through the use of improved sort-merge units, which increased overall throughput but were difficult to scale for big data sets. Casper et al. [36] compares various sorting techniques in great depth. The sorter that is comparator-free approach is the factor that is prioritized. Attempts to produce comparator-free sorting have been done in the recent past, demonstrating a focus on speed and resource

management. Optimization is also achieved in the suggested technique, in addition to not utilizing any comparator or complicated data path [36].

The Odd-Even approach is an alternative sorter algorithm that is quite similar to bubble sorting with a few variations. The process is repeated until the array elements are sorted, with Odd and Even Phases occurring in each iteration. We execute a bubble sort on odd indexed elements in the odd phase and an even phase bubble sort on even indexed elements [37]. This process is repeated for each alternating index (odd-even pair, even-odd pair) until no swapping operations are necessary and the array is sorted. In this sorting strategy, two modules are used: compare and swap. As a result, the time complexity can be calculated to be $O(N^2)$. By incorporating parallel computation with this sorting strategy, the time complexity can be reduced to O(N) [38]. This has recently been discovered to be the preeminent sorting method on hardware devices, as it provides stable throughput and consistent performance even when data distributions are skewed.

Bitonic sort is a typical parallel sorting method that is based on the Bitonic sequence. A Bitonic sequence consists of approximately half of the array arranged in ascending order and the other half arranged in descending order [39]. When compared to the odd-even sort, this sorting approach is extensively utilized since its structure makes implementation simple. This sorting algorithm has an $O(N \log_2(N))$ complexity. Insertion Sort is less efficient when sorting a large number of items, which necessitates a more advanced algorithm such as Quick Sort, Heap Sort, or Merge Sort because its average complexity is $O(N^2)$. Each iteration uses the Insertion Sort algorithm to integrate a new entry and compare the values of elements in the list [40]. If the value of an element is lower than its current value, a swap is made. The main disadvantage of this sorting approach is that it is not suitable for large data sets. The complexity is O(N) in the best case and $O(N^2)$ in the worst scenario [41].

The proposed article deals with design of comparator-free sorter in RISC-V processor and the obtained results are presented. The remainder section of the paper is structured as follows. Section 2 compares the benchmarks and provides an overview of relevant studies. Section 3 elaborates the proposed MicroBlaze architecture and its 3-stage pipeline. Section 4 discusses implementation of Odd-Even Comparison-free sorter in MicroBlaze softcore processor and its result analysis. Finally, Section V concludes the paper.

2. Related work

In this paper [7], an SRAM based reconfigurable architecture used to minimize the usage of units and low fragmentation. Also integrating low utilized LUTs for reconfigurability. Additional to above, to improve efficiency of reconfigurable architecture, configurable hard logics (CHLs) are used along with LUTs. This architecture is implemented on Berkeley RISC-V processor and MiBench Benchmarks. And as an alternative implemented on open source processor such as LEON and DSP core. Low power can be achieved by LEON processor and safety critical by RISC-V, leading to reconfigurability.

Some of the efficient soft core processors are SecretBlaze [8] and Chisel [9]. Conventional reconfigurable architectures have some drawbacks like application domain is not generic, power inefficient [10]- [13]. [7] proposes a reconfigurable architecture focusing on efficient area and power for soft core processors by making use of low utilization and fragmented functional units. To find the low utilization and fragmented functional units, benchmarks are used. Application Specific Integrated Circuit (ASIC) and Reconfigurable Unit (RU) module is best suitable for both soft and hard cores. But these methodology has certain drawbacks – access pattern not considered. The main concern is utilization of LUTs, even though in small input LUTs is underutilized in broad circuits [14] [15].

Extension of RISC-V by adding multiplication and division instructions which executed on hardware to open standard set provided in [16]. This architecture targeted for 32-bit soft processor RISC-V ISA by assuming in order processing of instructions and compared with various Benchmarks.

Xuantie-910 [17] RISC-V processor is a 64-bit out of order execution 12-stage pipeline releases high efficient vector processing, supports multi core processor, delivers high performance compared to its predecessor belonging to RISC-V family. The design [18] has been emphasized on instructions encoding, instruction functionalities, instruction types, decoder logic complexity, data hazard detection, register file organization and access, pipeline operation, impact of branch instructions, control flow, data memory access, operating modes, and hardware resources for the execution unit.

Bit flips in digital systems causes threats, well known as soft errors. This leads to corruption of data which remains undetected. So, [19] provides a study on handling various soft errors by BOOM and Rocket processor. [20] implemented on Xilinx Kintex-7 (xc7k410t-3fbg676) FPGA 3-stage pipeline with low complexity control circuits. Additionally, unique instructions that extend the functionality of the existing ISA are proposed to aid with energy metering. Mainly synthesized and deployed instrumented code in RISC-V and FPGA during execution resulted in eliminating execution time overhead [21].

In proposed work [22], RISC-V allows us to add new instructions as it supports open source and its taped out on 180nm TSMC process. In this paper, proposes embedded FPGA (eFPGA) is directly located inside main CPU. LowRISC Ibex RISC-V is a light weight core and uses two pipeline stage consuming less area. [23] discussed regarding resource optimization in paper. RISC-V with FPGA optimized resource utilization compared to only soft processor including out of order execution.

RISC-V supports ISA with no costs [24] and has characteristics like adaptability, flexibility, and features that are not yet present in the current solutions. The main goal of this work is the development of a RISC-V softcore processor to be implemented in an FPGA, using a non-RISC-V core as the base of this architecture. SecretBlaze [25] is an open source 32-bit RISC soft core processors with five stage pipeline and easily configurable suitable for system on chip (SOC) design. SecretBlaze uses a MicroBlaze instruction set. This approach is modular provides efficient implementation and increases abstraction level of hardware descriptions.

3. Proposed MicroBlaze architecture and optimization of sorter

FPGA processor cores are intellectual property (IP). As a soft cores involve design elements that can be implemented within the FPGA fabrics. The architecture of the proposed processor is presented in this section. The main focus is on the changes made to the MicroBlaze architecture and the implementation of the RISC-V ISA, as well as the new features and decisions made regarding the FPGA implementation. In this paper, discussion begins with a CPU architecture overview, which highlights the major structural changes. Each component is discussed in further depth in the subsections that follow.

A. Optimization of Sorter

The various sorters, such as Bitonic, Insertion, selection, and bubble sort were compared with the proposed comparison free odd-even sort. All types of sorters were designed to sort 4-, 8-, 16- and 32-bit numbers and the parameters chosen for the comparison are Number of slices, Number of LUT's, Number of used logics, Number of LUT Flipflops, IO utilization and lastly Delay. The result of the proposed sorter is highlighted in Table I by keeping the font in bold.

B. Architecture Overview

MicroBlaze is a soft core processor designed for Xilinx FPGAs and mainly implemented in the primary memory and logic block of FPGAs. A new instruction is issued along with data at every clock cycle. Assistance for multi-cycle functional units, caches, and pre-built peripherals including UART, timers, and counters stand apart. Based on this finding, the paper solution concentrated not only on the implementation of the RISC-V ISA but also on maintaining the MicroBlaze characteristics, such as the approaches adopted and their robustness. Apart from compatibility for the RISC-V ISA, the architecture's new features include sorter unit which is considered to be the important operation for real time application.

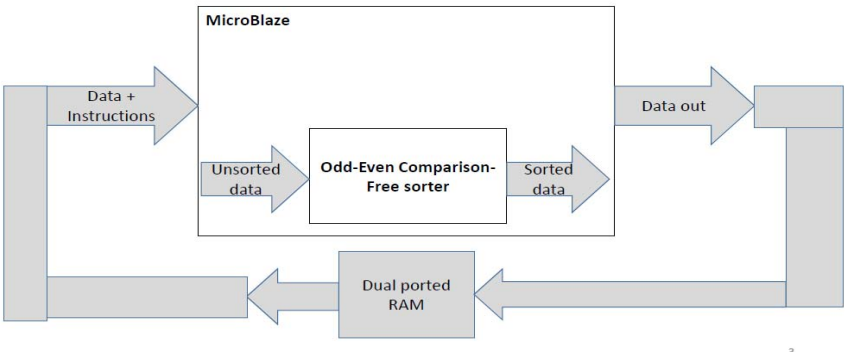


Figure 1: Overview of MicroBlaze Architecture

Figure 1 represents processor architecture of MicroBlaze by incorporating Odd-Even comparison-free sorter. MicroBlaze fetches data and instructions from local RAM memory. The Odd-Even comparison-free sorter accepts unsorted data from RAM as input and effectively sorts them using the sorter unit. Figure 2 shows the MicroBlaze architecture, which is based on MIPS processors and has a 3-stage pipeline with a modular implementation design, as well as two distinct instruction and data memory. Each phase was systematically constructed and defined in its own component, using the same hardware description methodology and signal

naming conventions. Because of its connectivity to the Register File, the Write-Back stage was implemented in the same component as the Instruction Decode.

C. RISC-V ISA

The required adjustments to implement the RISC-V ISA were mostly focused on the instruction decoder because different formats necessitate rewriting the parsing of each instruction. There are only two types of instructions in the MicroBlaze ISA: Type A (for register-register instructions) and Type B (used for register-immediate operations). The RISC-V ISA, on the other hand, has six different types of instructions. To parse the new sorts of instructions, new entries were added to the decoder. Because instructions of the same kind share the same opcode, each instruction type is assigned a unique opcode. The specific instruction is then discovered and the operands are obtained when the type has been determined.

Table I. Comparison various parameters of existing sorters with the proposed sorter for different bit length.

	Sorter	Number of Slice Registers: (Out of 515200)	Number of Slice LUTs: (Out of 257600)	Number used as Logic (Out of 257600)	Number of LUT Flip Flop pairs used (%)	IO Utilization (350)	Delay: (ns)
4 bits	Bitonic	72	219	221	31	68	2.874
	Proposed Odd Even	**70**	**168**	**168**	**30**	**68**	**2.147**
	Insertion	70	168	168	41	68	2.497
	selection	79	214	216	33	68	3.338
	bubble	76	235	236	31	68	3.433
8 bits	Bitonic	140	430	436	33	134	3.431
	Proposed Odd Even	**134**	**284**	**284**	**28**	**132**	**3.104**
	Insertion	138	637	640	32	132	3.409
	selection	145	378	382	38	132	3.743
	bubble	145	395	400	37	132	3.475
16 Bit	Bitonic	264	650	652	41	260	4.697
	Proposed Odd Even	**262**	**491**	**491**	**37**	**260**	**3.791**
	Insertion	274	1087	1097	45	260	5.607
	selection	270	645	646	42	260	4.865
	bubble	273	678	679	41	260	5.942
32 bits	Bitonic	518	1536	1536	44	516	5.786
	Proposed Odd Even	**521**	**1052**	**1052**	**42**	**518**	**4.445**
	Insertion	620	2777	2867	46	516	5.873
	selection	530	1150	1155	46	56	4.899
	bubble	533	1331	1338	48	516	6.125

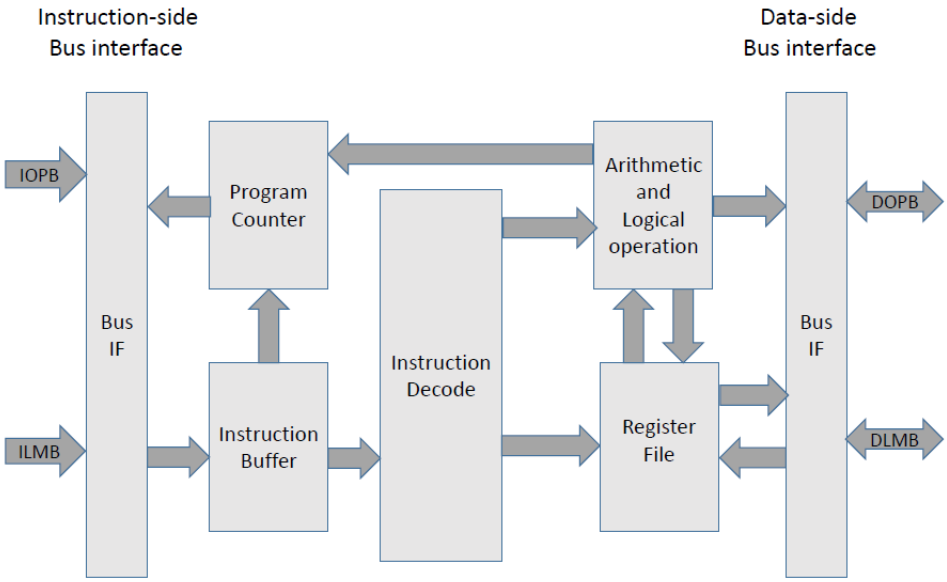


Figure 2: 3-stage Pipeline MicroBlaze Architecture

D. Memory Organization

In a Harvard architecture, instructions and data are stored in different memories. This framework offers simultaneous access to data and instructions, but it is not possible to combine the program and data addressing areas, such as loading a new program as data before execution. As a consequence, this processor is intended to be used as a slave, as each program must be transmitted before being executed. The CPU supports peripherals that are part of the address space in addition to instructions and data storage. They can be accessed by memory instructions, which are processed by the address decoder and routed according to the address map.

E. Instruction memory

The program instructions are stored inside BRAMs because the instruction cache was not implemented. The memory should have two ports: one for receiving and sending data through the Peripheral Component Interconnect Express (PCIe) interface, and another for fetching instructions from the CPU. Through the Vivado's IP Catalog, Xilinx offers a BRAM generator [26], which supports many types of RAMs with customizable sizes. The Dual-Port BRAM is the best option for the architecture since it supports two read and write ports for distinct addresses at the same time, as well as two independent clocks. It is feasible to connect one of the ports to receive data supplied over PCIe in this manner, but this interface must be compatible with AXI4 owing to implementation restrictions. Because BRAMs are not compatible with AXI, we must utilize Xilinx's AXI Block RAM Controller IP to convert between the two interfaces. The MicroBlaze's connection technology is flexible enough to accommodate a wide range of embedded applications. The Advanced eXtensible Interface (AXI) connection, the main I/O bus of MicroBlaze is a master-slave transaction bus that is system memory mapped. MicroBlaze is a 32-bit Core Processor tightly coupled with local memory, debug interface, controller UART interconnect, timer, and sorter unit. Figure 3 depicts the connections between the AXI Block RAM Controller and the instruction memory, which is implemented using a Dual-Port BRAM. Two distinct clock signals, one from PCIe logic and the other from the specialized core clock, are used to represent it.

AXI UARTlite provides asynchronous serial data transfer. One transmission and one receive channel are used in the AXI interface, which is based on the AXI4-Lite specification (full duplex). MicroBlaze soft processor is connected to Odd-Even sorter unit via AXI interface.

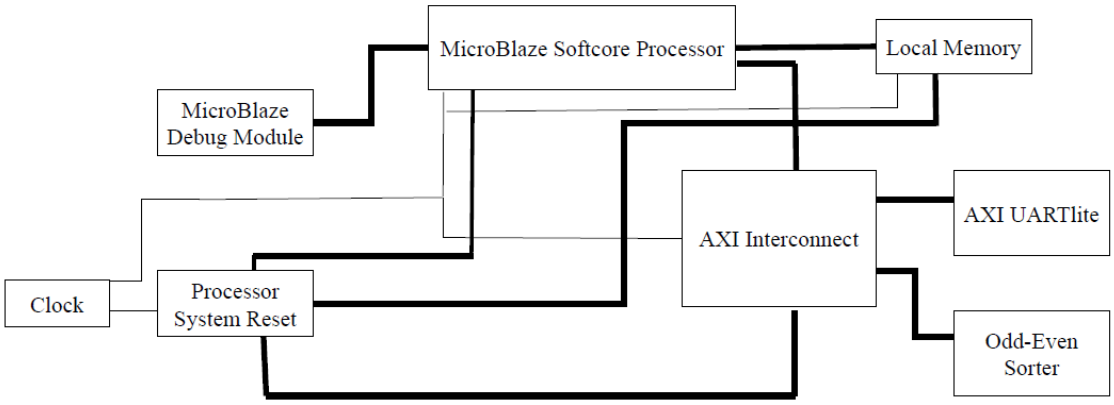


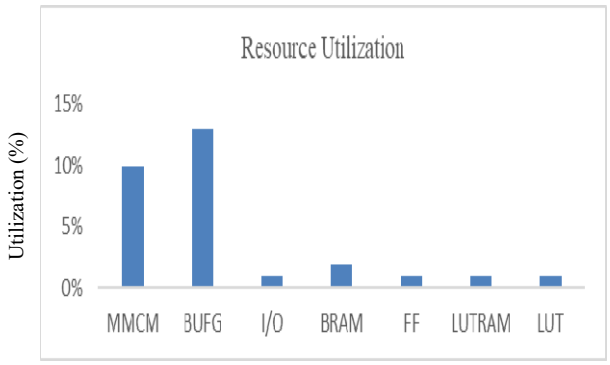
Figure 3: Connections between the Core, the Local Memory (implemented with a Dual Ported RAM IP), and the AXI Block RAM Controller and interconnect

4. Performance Analysis

The proposed Odd-Even Comparison-Free sorter was primarily simulated and synthesized on Xilinx Artix7 "XC7A200T-2FBG6762" MicroBlaze softcore FPGA at 100MHz. Table II represents the devices utilization of proposed sorting on softcore FPGA, open RISC and Rocket core. Utilizing the Cadence Encounter RTL compiler and UMC 90nm, a processor has been synthesized.

Table II Device utilization of MicroBlaze Softcore Processor

Parameters	Proposed softcore	Open RISC [27]	Rocket Core[28]
Slice Registers	1609	1287	1342
Slice LUTs	1327	1784	1376



Types of Resources
Figure 4: Resource Utilization

Table III Resource Utilization of Logic Blocks

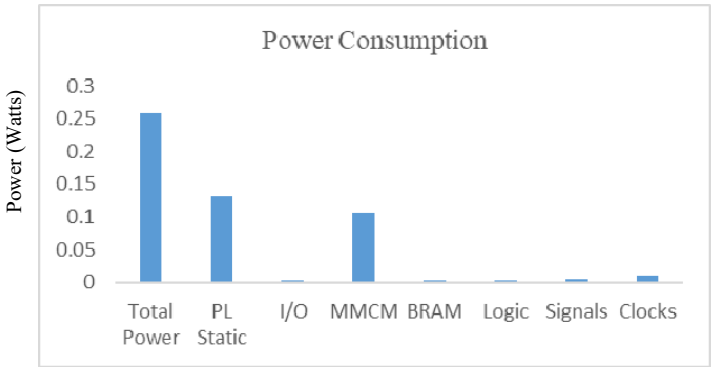
MMCM	BUFG	I/O	BRAM	FF	LUTRAM	LUT
10%	13%	1%	2%	1%	1%	1%

The results of the proposed Odd-Even Comparison-Free sorter considering the parameters such as IOB, Number of slices, Number of LUT's, Number of logic, Number of LUT Flipflops, Number of Registers, and Muxes are tabulated in Table IV. The frequency at which tests performed is 100MHz. The number of elements considered for sorting was seven. Hence, N=7 in all conditions. Therefore, for 32-bit numbers the total length is 224-bits.

Table IV Resource Utilization of Odd-Even Comparison-Free Sorter

Sorting Technique	IOB	Number of slices	Number of LUT's	Number of Registers	Number of LUT Flipflops	Muxes
Odd-even comparison free merge sort(proposed)	164	84	320	130	520	80

Figure 5 depicts the power consumption by various on-chip components like I/O, BRAM, static power, Muxes, Registers, and other logic components. As noted in graph, the total power consumption is 25.9% for overall sorting elements.



Parameters

Figure 5: Power Consumption

Table V Power Consumption by various blocks

Total Power	PL Static	I/O	MMCM	BRAM	Logic	Signals	Clocks
0.259	0.131	0.003	0.107	0.002	0.003	0.004	0.01

Table V represent details of power consumption by various blocks in watt unit and Table VI shows the delays consumed by data and instruction path.

RISC-V Benchmarks: Not only simple tests, more complicated programs are executed on RISC-V benchmarks.

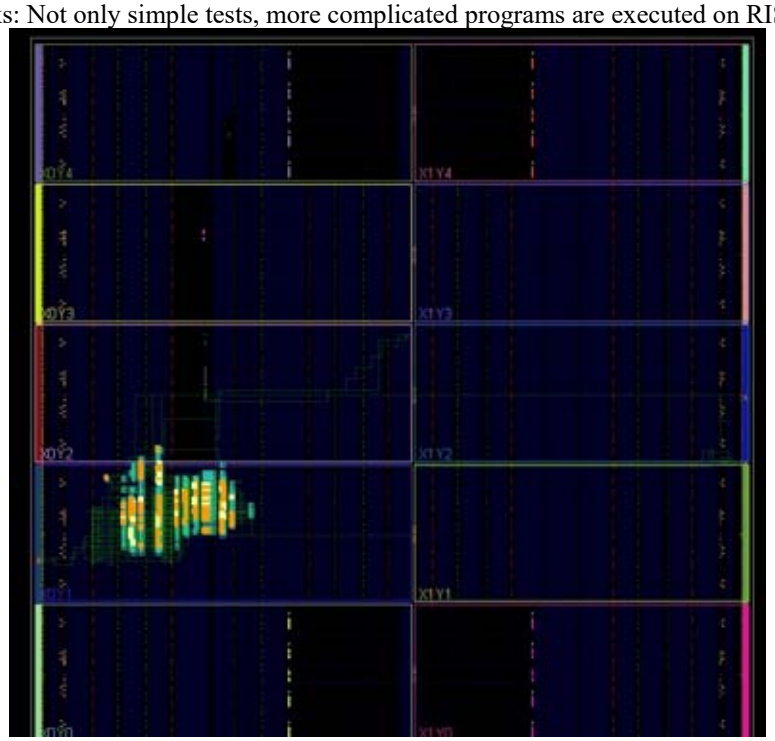


Figure 6: RV32I Sorting implementation for N=7 elements

Table VI Maximum path delays

Data Path Delay	Destination Clock Delay	Source Clock Delay	Slack (MET)
2.580ns	3.709ns	4.162ns	14.037ns

5. Conclusion

Here, optimized comparison free even-odd sorter was proposed for the RISC-V architecture. It was proved that it had capacity for better performance after comparing with several parameters such as memory, power consumption and resource utilization. Therefore, comparison-free Odd-Even sorter was chosen and it was to be implemented on the MicroBlaze architecture. Implementation of the proposed method had scope for reducing the active need of memory range as well reducing the area required to be implemented. The proposed sorter validates that it had potential to sort any kind asymmetrical data and gives out completely sorted output. The sorter was successfully implemented on Xilinx Artix7 "XC7A200T-2FBG6762" MicroBlaze softcore FPGA at 100MHz.

Acknowledgments

On behalf of all authors, the corresponding author states that there is no conflict of interest and no funding received.

References

[1] Andrew Waterman, "Design of the RISC-V Instruction Set Architecture", Technical Report No. UCB/EECS-2016-1 http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-1.html
[2] Andrew Waterman, Krste Asanoví c, "The RISC-V Instruction Set Manual", Volume I: User-Level ISA, Document Version 20191213, Editors RISC-V Foundation, December 2019.
[3] Philippos Papaphilippou, Paul H. J. Kelly, Wayne Luk, "Extending the RISC-V ISA for exploring advanced reconfigurable SIMD instructions", CARRV 2021, June 17, Virtual Workshop, Co-located with ISCA 2021.
[4] Philippos Papaphilippou and Wayne Luk. 2018. Accelerating database systems using FPGAs: A survey. In 2018 28th International Conference on Field Programmable Logic and Applications (FPL). IEEE, 125–130.
[5] Jeffrey Stuecheli, William J Starke, John D Irish, L Baba Arimilli, D Dreps, Bart Blaner, Curt Wollbrink, and Brian Allison. 2018. IBM POWER9 opens up a new era of acceleration enablement: OpenCAPI. IBM Journal of Research and Development 62, 4/5 (2018), 8–1.
[6] Robert J Halstead, Ildar Absalyamov, Walid A Najjar, and Vassilis J Tsotras. 2015. "FPGA-based Multithreading for In-Memory Hash Joins".. In CIDR.
[7] Sajjad Tamimi, Zahra Ebrahimi, "An Efficient SRAM-based Reconfigurable Architecture for Embedded Processors", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, March 2018.

[8] L. Barthe, L. V. Cargnini, P. Benoit, and L. Torres, “The SecretBlaze: A configurable and cost-effective open-source soft-core processor,” in Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), International Symposium on. IEEE, 2011, pp. 310–313.
[9] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avi˘zienis,J. Wawrzynek, and K. Asanovi´c, “Chisel: constructing hardware in a scala embedded language,” in Proceedings of the 49th
[10] A. R. Ashammagari, H. Mahmoodi, T. Mohsenin, and H. Homayoun, “Reconfigurable STT-NV LUT-based functional units to improve performance in general-purpose processors,” in Proceedings of the 24th edition of the great lakes symposium on VLSI. ACM, 2014, pp. 249–254.
[11] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh, and M. C. Eliseu Filho, “Morphosys: an integrated reconfigurable system for data-parallel and computation-intensive applications,” Computers, IEEE Transactions on, vol. 49, no. 5, pp. 465–481, 2000.
[12] S. C. Goldstein, H. Schmit, M. Budiu, S. Cadambi, M. Moe, and R. R. Taylor, “Piperench: A reconfigurable architecture and compiler,” Computer, vol. 33, no. 4, pp. 70–77, 2000.
[13] Z. A. Ye, A. Moshovos, S. Hauck, and P. Banerjee, CHIMAERA: a high-performance architecture with a tightly-coupled reconfigurable functional unit. ACM, 2000, vol. 28, no. 2.
[14] I. Ahmadpour, B. Khaleghi, and H. Asadi, “An efficient reconfigurable architecture by characterizing most frequent logic functions,” in Field Programmable Logic and Applications (FPL), 2015 25th International Conference on. IEEE, 2015, pp. 1–6.
[15] A. Ahari, B. Khaleghi, Z. Ebrahimi, H. Asadi, and M. B. Tahoori, “Towards dark silicon era in fpgas using complementary hard logic design,” in Field Programmable Logic and Applications (FPL), 2014 24th International Conference on. IEEE, 2014, pp. 1–6.
[16] Md Ashraful Islam, “RVCoreP-32IM: An effective architecture to implement mul/div instructions for five stage RISC-V soft processors”, in arXiv:2010.16171v1.
[17] Chen Chen, “Xuantie-910: A Commercial Multi-Core 12-Stage Pipeline Out-of-Order 64-bit High Performance RISC-V Processor with Vector Extension”, IEEE 47th Annual International Symposium on Computer Architecture (ISCA), 2020.
[18] Aneesh Raveendran, Vinayak Baramu Patil, “A RISC-V Instruction Set Processor-Microarchitecture Design and Analysis”, International Conference on VLSI Systems, Architectures, Technology and Applications, 2016.
[19] Hyungmin Cho, “Impact of Microarchitectural Differences of RISC-V Processor Cores on Soft Error Effects”, Digital Object Identifier 10.1109/ACCESS.2018.2858773, 2018.
[20] Yajie Wang, “An Application-Specific Microprocessor for Energy Metering Based on RISC-V”,
[21] Iván Gamino del Río, “A RISC-V Processor Design for Transparent Tracing”, MDPI electronics journal, 2020.
[22] Nguyen Dao, “FlexBex: A RISC-V with a Reconfigurable Instruction Extension”, International Conference on Field-Programmable Technology (ICFPT), 2020.
[23] Susumu Mashimo, “An Open Source FPGA-Optimized Out-of-Order RISC-V Soft Processor”,
[24] Joao Filipe Monteiro Rodrigues, “Configurable RISC-V softcore processor for FPGA implementation”, 2019.
[25] Lyonel Barthe, Lu´ıs Vit´orio Cargnini, “The SecretBlaze: A Configurable and Cost-Effective Open-Source Soft-Core Processor”, IEEE International Parallel & Distributed Processing Symposium, 2011.
[26] “Block Memory Generator,” Available: https://www.xilinx.com/support/documentation/ip documentation/blk mem gen/v8 3/pg058-blk-mem-gen.pdf, 2017, [Online].
[27] Damjan Lampret et al., "OpenRISC 1000 Architecture Manual", Architecture Version 1.0, Document Revision 0, December 5, 2012
[28] RISC-V processor core “Rocket Core” EECS Department, UCB, May 2011.
[29] Jorge Ortiz, “A Streaming High-Throughput Linear Sorter System with Contention Buffering”, International Journal of Reconfigurable Computing, 2011.
[30] K. Batcher, “Sorting networks and their applications,” in Proceedings of the AFIPS Spring Joint Computer Conference, vol. 32, pp. 307–314, ACM, 1968.
[31] E. W. Dijkstra, “A heuristic explanation of Batcher's Baffler,” Science of Computer Programming, vol. 9, no. 3, pp. 213–220, 1987.
[32] A. F. M. Fahad Alif, S. Md. R. Islam, and P. Deb, “Design and implementation of sorting algorithms based on FPGA,” in 2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2), 2019, pp. 1–4. doi: 10.1109/IC4ME247184.2019.9036675.
[33] O. Astrachan and B. Sort, “An Archaeological Algorithmic Analysis”, SIGCSE, 2003.
[34] G. K. Harshini, S. Kumar, and K. K. Gayathri, “Hardware implementation of sorting algorithm using FPGA,” 2018. [Online]. Available: www.ijariie.com
[35] T. Usui, T. van Chu, and K. Kise, “A Cost-Effective and Scalable Merge Sorter Tree on FPGAs,” in Fourth International Symposium on Computing and Networking (CANDAR), 2016, pp. 47–56. doi: 10.1109/CANDAR.2016.0023.
[36] S. Ghosh, S. Dasgupta, and S. Saha Ray, “A Comparison-free Hardware Sorting Engine,” in Proceedings of IEEE Computer Society Annual Symposium on VLSI, ISVLSI, Jul. 2019, pp. 586–591. doi: 10.1109/ISVLSI.2019.00110.
[37] A. Hematian, Suriayati Chuprat, Azizah Abdul Manaf, and Nadia Parsazadeh, “Zero-Delay FPGA-Based Odd-Even Sorting Network,” in IEEE Symposium on Computers & Informatics, 2013, pp. 128–131.
[38] R. Ayoubi, S. Istambouli, A.-W. Abbas, and G. Akkad, “Hardware Architecture for A Shift-Based Parallel Odd-Even Transposition Sorting Network,” in Fourth International Conference on Advances in Computational Tools for Engineering Applications (ACTEA), 2019, pp. 1–6. doi: 10.1109/ACTEA.2019.8851099.
[39] R. Chen and V. K. Prasanna, “Computer Generation of High Throughput and Memory Efficient Sorting Designs on FPGA,” IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 11, pp. 3100–3113, 2017, doi: 10.1109/TPDS.2017.270512.
[40] Sodhi, T., Kaur, S., & Kaur, “Enhanced insertion sort algorithm”, International journal of Computer applications, 2013.
[41] Yomna Ben Jmaa, Rabie Ben Atitallah, “A Comparative Study of Sorting Algorithms with FPGA Acceleration by High Level Synthesis”, Computación y Sistemas, Vol. 23, No. 1, 2019, pp. 213–230.

Authors Profile

Preethi received B.E and masters from VTU. Total 14 years of Teaching experience. Currently working as Assistant professor at Presidency University, Bangalore. Area of interest: Computer Architecture and Internet of Things.

Dr. Mohan K G received B.Tech and M.Tech from VTU**,** Doctoral Degrees from NIT Suratkal. With vast total 34 years of Teaching and Administrative experience working in different cadre like Assistant Professor, Associative Professor, Professor,Head of the Department, Dean-R&D, Dean-Academics and Principal. Currently working as Professor at Gitam University, Banagalore. His area of interest: Digital Design and Computer Architecture.

Dr. Jacob Augustine received M.Tech from IIT, Madras and Doctoral Degree from IISc, Bangalore, Research Post-Doctoral Fellow (Concordia University, Montreal, Canada). Teaching experience around 12 years and Worked as Principal Engineer, Multimedia Group at Sasken Communication Technologies, Hewlett-Packard R&D Bangalore in various groups (HP Labs, Servers, Networking) as tech lead and R&D Manager for 17 years. Currently working as Associate Dean at Presidency University, Bangalore. His area of interest: Automata Theory and Programming.

Dr. Sudeendra kumar K received B.E. from VTU and holds Masters and Doctoral Degrees from NIT Rourkela. His professional career includes a stint as Application Engineer for Freescale Semiconductors and Product Test Engineer for Qualcomm. He has got significant experience in Special Manpower Development Project -II (SMDP-II) and C2SD (SMDP-Chip to Systems Development) project under the aegis of Ministry of Communication and Information Technology, Govt. of India. Currently, he is Domain Head of VLSI and Embedded Systems in the Department of Electronics and Communication Engineering, PES University.