SN Computer Science (2023) 4:129
https://doi.org/10.1007/s42979-022-01546-7

ORIGINAL RESEARCH

Sorter Design with Structured Low Power Techniques

P. Preethi[1] · K. G. Mohan[2] · Sudeendra Kumar K.[3] · K. K. Mahapatra[4]

Received: 19 April 2022 / Accepted: 3 December 2022 / Published online: 27 December 2022

Abstract

Sorting is a very important function which is widely used in several applications like signal processing and other data center acceleration. Sorting is generally implemented on CPU or GPU, which takes several cycles to finish the sorting process. Further improvement in performance in sorting is possible through hardware acceleration either in FPGA or ASIC. The performance improvement and reducing the power consumption are the primary goals for researchers to improve the hardware acceleration of sorting algorithms. The sorting techniques like Bubble sort, Bitonic sort and Odd–even sort are found suitable for hardware implementation and widely discussed in the research literature. It is evident from the literature survey 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 and Multi-Vth 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 and multi-Vth 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. Further performance improvement is achieved through adopting multi-Vth libraries without compromising the dynamic power reduction achieved through clock gating. 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 · Clock gating · Multi-Vth · Sorting architecture

Introduction

Most of the algorithms used in computing today needs sorting. Advancements in communications, cloud storage and high- performance computing are driving the innovation in the gig economy. Success in modern businesses depends upon reliable and fast services from the Cloud-based data centers. The major technological advancements like Bigdata, artificial intelligence (AI) and Blockchain are computationally intensive and it is necessary to have high-performance computing infrastructure for data acceleration. Sorting is an important operation which is frequently used in most of the applications like graph theory, AI and in crunching the data getting generated in applications of Internet of Things (IoT) like smart city and smart transportation etc. Sorting operations can be implemented in both software and hardware or in hybrid way. Better speed of operation for sorting can be achieved in hardware implementation than in software, as it consumes excessive number of CPU cycles. The well-structured and deterministic parallelism can be achieved in

This article is part of the topical collection "Smart and Connected Electronic Systems" guest edited by Amlan Ganguly, Selcuk Kose, Amit M. Joshi, and Vineet Sahula.

✉ P. Preethi
preethisrivathsa@gmail.com

K. G. Mohan
mkabadi@gitam.edu

Sudeendra Kumar K.
kumar.sudeendra@gmail.com

K. K. Mahapatra
kmaha2@gmail.com

[1] Presidency University, Bangalore, India

[2] GITAM University, Bangalore, India

[3] PES University, Bangalore, India

[4] NIT, Rourkela, India

hardware implementation than in software-based parallel implementation in multi-core CPUs. Hardware acceleration of sorting are implemented using either as a dedicated ASIC or FPGAs to meet the required performance [1, 2]. Hardware sorter architectures depend upon the target applications and number of inputs, width of the inputs and ordering of the data. The well-known categorization of the sorters is merge sorters and non-merge sorters. Merge sorters need the memory element along with sorting logic. Non-merge sorters like bitonic, bubble and odd–even sorters do not require memory element and general pre-processing and post-processing modules are enough [3]. Sorting in hardware is efficient than software implementation for short sequences whose length and width of the data are known a priori. A good amount of literature on sorting implementation in FPGA is available, because it does not require control flow instructions and it is not tedious to parallelize the sorting on FPGA devices. Modern data centers are built upon multi-core processors, GPUs and hardware accelerators for specific applications to serve the different application verticals, in which hardware sorters make place in most of the applications. Multi-core processors and GPUs are ASICs and hardware accelerators for custom applications like sorting are generally implemented in FPGAs currently. The existing data centers face a problem of huge thermal heating and high-power consumption and current computing configuration with FPGA-based hardware accelerators for custom applications like sorting does not solve the problem of high-power consumption and thermal heating. The problem of thermal heating and high-power consumption will further escalate with the full-fledged deployment of high-speed 5G networks, advanced searching and machine learning techniques. The implementation of data intensive operations like sorting which are currently implemented in FPGA must be implemented in ASIC to address the problem of power consumption. The FPGA -based accelerators in data centers must be replaced with dedicated low-power ASICs to achieve required low-power consumption and reduce overall thermal heating of the data centers. FPGA-based computing systems for sorting have got several advantages like reconfigurability, efficient parallelism for sorting, etc., and they have following disadvantages:

- Designing and developing low-power sorters without compromising the performance are limited in FPGA's as already device is fabricated.
- In ASIC designs, we can use standard low-power techniques like clock gating, Multi VDD, Multi-Vth, power gating and dynamic voltage frequency scaling (DVFS) to reduce power consumption which may not supported in all available FPGA devices.

Gradually, FPGA-based hardware accelerators in current servers will be replaced by programmable ASIC-based hardware accelerators, which is required to balance the power without compromising the performance requirements [4, 5].

In this work, we focus on designing the low-power sorting architectures and comparison of low-power implementation of widely used sorting techniques. The standard and structured low-power digital design techniques like clock gating, multi-VDD, multi-Vth, 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 analyses of power and performance is an unexplored area of research. In this paper, we primarily focus on clock gating and multi-Vth (Threshold Voltage) based techniques to achieve low power on different sorting engines and we present the power performance trade-off results for three widely used sorting engines.

This paper is organized as follows: the section "Background" presents the literature survey on existing ASIC and FPGA implementation of different sorting engines and brief literature review on standard sorting techniques used in the recent past. The section "Structured low-power techniques" discusses structured low-power techniques widely used in low-power design and their importance like clock gating, etc. The section "Implementation and Results" presents the implementation of low-power sorting and the section "Result analysis" discusses the results, comparison and analysis. Finally, in the section "Conclusion", conclusion of the paper is presented. This paper is an extension to our work presented in [6]. The improvements to the work done to the work presented in [6] are as follows:

- Modification of the Compare and Swap (CAS) to add the direction signal to the sorter to perform ascending and descending order. To accommodate the asynchronous direction signal, the complete CAS unit is redesigned to make direction signal synchronous.
- Standard sorting benchmarks are used to perform power analysis and to compare the different sorters.
- Power vs performance analysis of different sorters is performed for different multi-Vth libraries for modified CAS.

Background

Hardware implementation of sorting has attracted the researchers all along from last 4 decades [7]. Computational complexity of well-known sorting techniques is presented in Table 1. It is evident from the literature survey that merge sorting needs memory element along with sorting logic and other sorting techniques does not need memory element. Fastest sorting methods are odd–even and bitonic sorters for streaming data [2, 3, 8]. Sorters

Table 1 Algorithms and complexity

Sorting techniques	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	$O(n^2)/O(n^2)$	$O(n^2)$

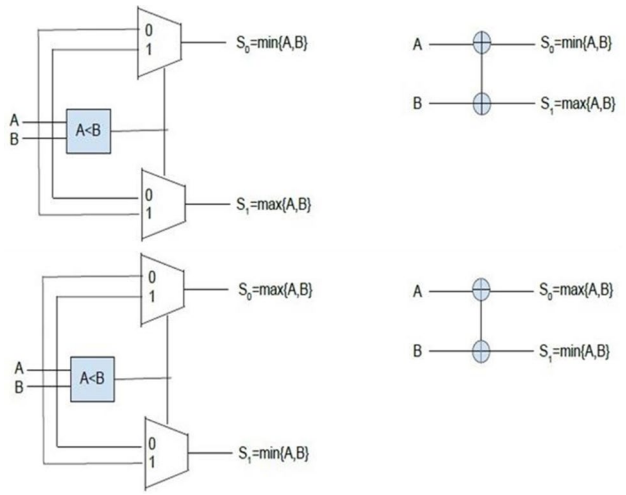

Fig. 1 Representation of compare and swap units used in sorters

can be implemented either using combinational circuits or can be sequential circuits with sorter datapath. Sequential Sorters are good candidates for pipelining and parallel processing.

Generic Compare and Swap Circuit (CAS) for Sorters

Hardware implementation of sorters primarily composed of series of compare and swap units. The circuit diagram of compare and swap unit is shown in Fig. 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.

To handle the large data sets, bubble, bitonic and odd–even sorts are employed [8, 9]. As the width of the data increase, the number of comparators required is enormous. It is practically not possible to implement such a large vertical array of comparators. There is a need to design the sorter in modular way, so that the design can be expanded easily to increasing data widths.

Taxonomy of Sorters

Sorting is a one of the primitive Data Base Management System (DBMS) operation and researchers have proposed several sorting algorithms which are generally implemented on general-purpose CPU for several decades. Software sorting algorithms can be classified into two categories: simple sorting algorithms, which work on limited amount of data where the unsorted data are completely fit in the available memory. Another category is Merge-sort algorithms, which work on large amount of data, in which data are split into several chunks and sorted. Finally, sorted data are merged to get the intended final sorting result. The common sorting algorithms which work on limited amount of known size of data are bubble sort, insertion sort, etc. The well-known merge sorters, which work on large chunks of data, are merge sorters and tag sorters.

In hardware implementation, sorters can be categorized in different ways. One such categorization is: Comparator based sorters and Comparator-less sorters. The well-known comparator-based sorters are insertion sort, bubble sort and quick sort. The comparator-less sorters are bucket sort and radix sort.

Further, sorters can be categorized as modular and non-modular sorters. The Compare and Swap (CAS) is used in both modular and non-modular sorters. Bubble sort, Odd even sort and bitonic sort are the examples of modular sorters, which are scalable. By adding and arranging the CAS structures, we can scale these sorters for different bit-widths.

There are few types of sorters in which scaling is difficult and such sorters can be categorized as non-modular sorters. Example of such sorters are panning sorter and weaving sorters [10]. The modular sorters like Bubble sort, Odd–even sort and Bitonic sorters are briefly explained.

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 the first stage with an input size of M. Similarly, in the 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. 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 merging sort by merging the two sorted sequences into a complete sorted result. An

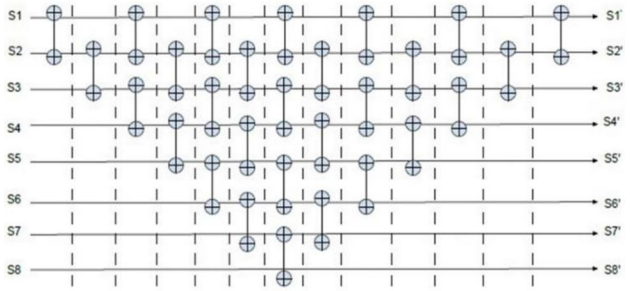

Fig. 2 Bubble sorter

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-2 s, OE-4 s…to OE-M. When the input dataset consists of 2P samples, there will be 2P-1 OE-2 s in the first stage, 2P-2 OE-4 s in the second stage and so on. Figure 3 illustrates the eight-input odd–even merge sorting network.

Bitonic Sort

Bitonic sorting is also proposed by Batcher in [8]. Bitonic sorter is constructed 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 2P. To build the complete 2P input bitonic sorting circuit, a series of bitonic merging units is applied recursively to generate the sub-sequence.

The 2P input bitonic sorting network comprises 2P-1 parallel BM-2 s in first stage and 2P-2 BM-4 s in the second level and so on. Figure 4 shows the Bitonic sorting network for eight inputs.

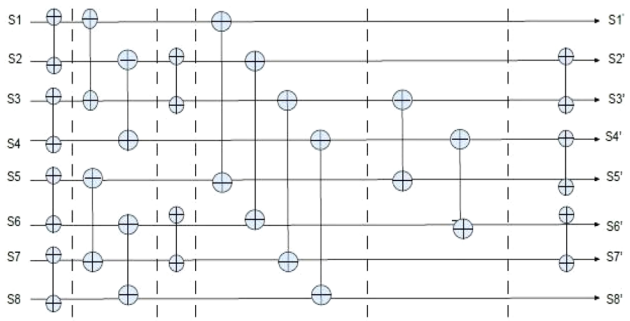

Fig. 3 Odd–even sorter

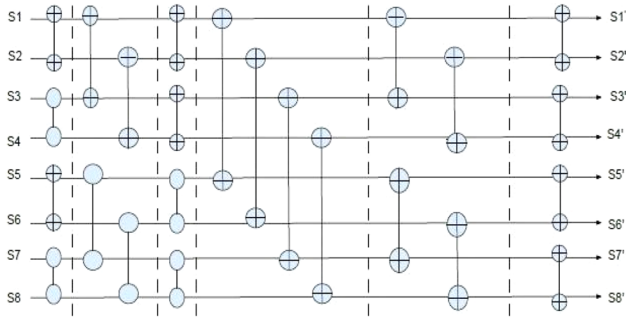

Fig. 4 Bitonic sorter

It is well known from the literature and it is established in research that bitonic and odd–even sorter is suitable to design large-scale high-speed pipelined sorting networks.

ASIC Implementation of Modular Sorting Techniques

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 [3]. The modular approach proposed in [8] is extended in [9]. The paper [9] propose the low-power sorting network, which uses the modular sorter design proposed in [8].

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, which increases the switching activity enormously, which increases the dynamic power consumption. To reduce the dynamic power, an immobile comparing unit which move the indexes of samples instead of moving the actual input data is proposed. This technique [9] reduces the power significantly. Different kind of modular approach is presented in [11], 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 [12]. The authors of [12] present the results that unary processing-based bitonic sorting network reduces the hardware cost and power consumption in comparison with conventional bitonic sorting network.

Table 2 Comparison of low-power sorting techniques

Paper	Technique	Technology	Node and implementation methodology	Remarks
[8] 2013	The sorting units are designed for instances when only the M largest numbers from a set of N inputs need to be sorted. To meet the requirement, bitonic and odd even sorting techniques are modified/changed	65 nm standard cell library used for synthesis	For various input configuration, an area and timing analysis report is being presented (semi-custom design)	The suggested sorting network does not have a power analysis
[9] 2017 (LP)	The extension work of this [9] reduces the power consumption. Dynamic power reduction is possible in modular sorting network by adopting a pointer-like design in which sample indexes are moved instead of input data	90 nm standard cell library used for synthesis	For different input configurations, dynamic power analysis reports are made and compared with existing technique [8] For various input configuration, an area and timing analysis report is being presented (semi-custom design)	Power analysis reported that, with increase in bit-size of the input, the power analysis improved minimally and there is an improvement in power reduction reported for lower frequencies at higher bit sizes
[11] 2017	A novel sorting algorithm proposed which can sort the input data integer elements on the fly without any comparison The proposed technique results in O(N) computational complexity and it's a significant contribution	90-nm technology used for synthesis	The results of transistor count, timing and power are presented (full custom design)	The design is made up of digital components and simply transferred to FPGA and semi-custom ASIC. Scalability cannot be achieved for larger input bit sizes due to transistor level implementation The dynamic power and latencies may be high for large input bit sizes in the proposed circuit Hence an optimized circuit required
[12] 2018 (unary)	[12] proposes a low-cost, low-power sorting algorithm based on unary processing This processing reduces the wiring between compare and swap units in the large sorting networks	45-nm standard cell library used for synthesis	Area, power and timing results are reported for 8-, 16- and 32-bit data length (semi-custom design)	Proposed technique reduces the power notably for the higher bit- width at the cost of increased latency. The processing of digital unary streams takes a lengthy time
[16] RTHS	[16] propose and implement the multi-dimensional sorting algorithm (MDSA) and its architecture for hardware implementation called RTHS (real-time hardware sorter)	Implemented on FPGA	Performance of the proposed algorithm is validated by modifying the Bitonic sorter	High-performance sorter, but increases the dynamic power consumption

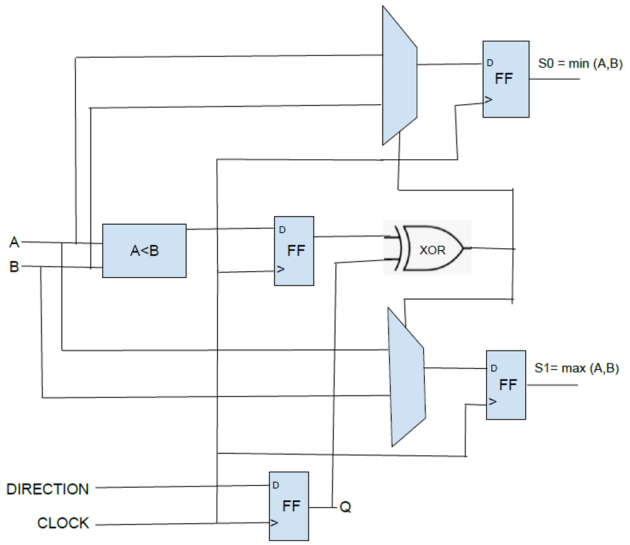

Fig. 5 Modified CAS

Table 3 Features of multi-Vth CELLS

Vth	Leakage power	Dynamic power	Speed
LVth	Highest	Highest	Highest
SVth	Normal	Normal	Normal
HVth	Lowest	Lowest	Lowest

The detailed comparison of the above-mentioned techniques is presented in Table 2.

Challenges in Designing Large-Scale Modular Sorting Networks

It is evident from the literature survey that the major challenges in designing large sorting networks are as follows:

- The most important challenge is the long critical paths and increase with the input bit width. The long critical paths impact the operating frequency decrease the overall performance. Pipelined implementation of sorters will reduce the critical paths to the larger extent [8].
- The pipelined implementation increases the switching activity significantly and makes the design power hungry. There is a need to decrease the power consumption by adopting the suitable low-power techniques.

The pipelining/parallel processing increases the dynamic power which has become the one of the primary concerns in modern data centers.

In this work, we primarily focus on three important sorting algorithms: odd–even sorting bubble sorting and bitonic sorting. We will apply structured low-power technique like clock gating on pipelined sequential implementations of odd– even sorting and bitonic sorting and analyze the dynamic power for ASIC design. To the best of our knowledge, a generic well-established clock gating and multi-Vth (threshold) technique is employed on sorting networks to reduce the power consumption and optimize the critical paths. We will present the comparative results for different sorters for power, performance and area.

Modified Compare and Swap Circuit

The modified Compare and Swap (CAS) circuit diagram is shown in Fig. 5.

The basic CAS circuit presented in Fig. 1 is improved by adding an XOR gate. 2-Input XOR gate is fed with direction signal which is an external input and other input to the XOR gate is the output of the comparator. The direction signal is added to change the direction of sorter to perform ascending or descending order sorting [13]. Most of the times, the CAS circuits used in sorters are combinational circuits as shown in Fig. 5, which will have the following issues:

- The output of CAS circuits is not synchronized and mismatches in the output of multiplexer during gate-level simulation and post-layout simulation. To ensure the mismatches between the outputs of the multiplexers, the outputs must be clocked.
- In the combinational CAS circuit, direction signal is fed into XOR gate is fully asynchronous and it needs to be synchronous to avoid mismatches and proper functioning of the overall circuit.

The proposed modified CAS circuit is fully synchronous and both asynchronous inputs and outputs are clocked. This improves the working of CAS significantly. The un-clocked CAS is large combinational circuit and converting the large combinational CAS into sequential CAS reduces the synthesis and simulation mismatches. The combinational CAS is replaced by a sequential CAS shown in Fig. 5. In this work, we have used modified sequential CAS in the implementation of Bubble, Bitonic and Odd–even sorter.

Structured Low Power Techniques

The design for low power is a vast subject and low-power requirements can be achieved at circuit level, gate level, RTL level and system level. The circuit level changes are difficult to adopt in semi-custom design. As semi-custom designs use already available standard cell libraries from foundry, the

improvements at logic gates level are also difficult. After a several years of research, industry has adopted matured low-power techniques which can be adopted in semi-custom design flow easily and require no modification in design or architecture. Such techniques are: Clock gating, Power gating, using Multi VDD and Multi-Vth (Threshold Voltage) libraries. In this work, we apply only two structured low-power techniques on sorters namely, clock gating and multi-threshold.

Clock Gating

Generally, data are loaded into the registers infrequently in most of the designs, and the 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. The conceptual diagram of clock gating is shown in Fig. 5.

Multi-Threshold Libraries

Standard cell libraries contain cells with different threshold voltages for MOS devices in each standard cell.

Most of the standard cell libraries support different power and speed characteristics. Along with other several parameters like sizing of the transistor, the Vth (threshold voltage) of the logic gate determines the power and speed of the cell. Generally, libraries support High Vth (HVt), Standard Vth (SVt) and Low Vth (LVt) classes of cells. The features of the three different multi-Vth cells are presented in the Table 3. The HVth cells have lower leakage and dynamic power and less speed. LVth cells will have highest leakage and dynamic power and smaller delays. SVth cells will have normal power consumption and normal speed (Table 3).

Implementation and Results

Clock Gating Implementation

To further speed up the operation, we implemented 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 that 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 Xilinx Vivado Simulator and 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.

Multi-Vth Implementation

Multi-threshold voltage technique uses Standard Vth, Low Vth and High Vth cells. In this technique, lower threshold gates are used in critical path and high threshold are used in the non-critical path. This methodology improves the speed of operation without an increase in the power consumption. The areas of low Vth and high Vth cells are same as that of standard Vth cells. Therefore, in the critical paths or in any other paths of the design, Standard Vth cells can be replaced with either low Vth or High Vth cells as per the requirement. This technique improves the performance without compromising on area and increasing power consumption.

Multi-Vth implementation can be performed in various ways. It can be done during RTL synthesis, and gate-level netlist or during layout through engineering change order. Logic synthesis is a process of converting the RTL Verilog code into functionally equivalent gate-level netlist.

The standard cell libraries with multiple Vth can used in synthesis to achieve different Vth cells for different timing paths in the design. Synthesis is an ideal place to perform multi-Vth implementation. Synthesis is performed using the Standard Vth (SVth) library.

In this work, three sorters are implemented using both clock gating and multi-Vth. The process followed to incorporate clock gating is explained above. Multi-Vth implementation for sorters is performed as follows:

- Clock gating is performed after the RTL synthesis. The clock gated gate-level circuit obtained from synthesis is simulated for functional verification. Synthesis is performed using Standard Vth (SVt) library with normal design constraints.
- Static timing analysis (STA) is performed to find out the critical path/group of critical paths in the design. With

Table 4 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 nanoseconds)		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	36,456.23	35,345.87	6.545	6.776	1.3	0.585	46.59
	16	74,234.45	75,456.78	6.900	6.723	1.47	0.71	48.90
	32	145,667.3	144,589.8	7.423	7.733	1.88	0.912	51.83
Odd–even sort	8	32,762.34	30,231.89	5.845	5.812	1.39	0.641	52.55
	16	63,298.75	61,234.98	5.987	5.634	1.62	0.71	46.90
	32	125,987.6	122,889.9	8.412	8.854	2.1	0.912	47.02
Bubble sort	8	40,014.59	38,234.88	6.598	6.734	1.37	0.651	49.53
	16	81,298.56	79,126.38	7.378	7.564	1.68	0.845	52.48
	32	178,238.3	174,398.5	8.231	8.432	2.08	1.101	52.65

the help of STA, we can analyze the delay of all the paths in the design.

- Categorize the paths in the design as high slack, medium slack and low slack paths. The categorization of paths is performed based on the slack value of each path. The path with lowest slack (which is also a critical path) is grouped in low slack category, and similarly, all paths are placed in either high slack or medium slack or low slack group depending upon their slack value. Design is synthesized using SVth library.
- Further, in the non-critical paths (paths with high slack), the standard Vth cells are replaced with higher Vth (High Vt) cells. In paths with lowest slack (critical paths), the standard Vth cells are replaced with LVth cells.
- The standard cells which are common to two or three timing paths are not replaced with either HVth or LVth cells. The normal standard Vth cells are retained.
- PERL scripting is developed to replace the cells in synthesized gate-level netlist.
- Cadence Genus and Tempus tools are used to perform synthesis and static timing analysis, respectively. Synopsys Saed90nm is used for synthesis.

Result Analysis

Benchmarks: Sorting accounts for more than 20% of computing in most of the enterprise applications. And there are several sorting algorithms which are intended to get deployed on general-purpose processors, GPGPUs and FPGAs. A sorting technique which will work efficiently on multi-core processors may not work on the FPGAs or dedicated ASICs. To measure and evaluate the performance of the sorting techniques, sorting benchmark programs are devised and explained in [15]. The existing benchmarks [15] primarily target the general-purpose CPUs and GPGPUs and not developed to evaluate the sorting algorithms implemented on FPGAs or dedicated ASICs. The authors in [15] present the case for comprehensive framework for sorting benchmarks, which suggests that sorting benchmarks must be designed targeting the intended target like FPGA, GPGPU or CPU and irrespective of hardware target, and the programs/benchmarks used to evaluate the sorting algorithms must have some statistical characteristics like Gaussian distribution and collectible system statistics like I/O utilization, etc. The existing benchmarks in [15] are designed for CPUs and GPUs and those are not useful for FPGA or ASIC evaluation.

The benchmarks presented in [15] like Gray sort, Tencent sort, etc. are designed to verify the performance based on fast memory access and SDRAM burst sizes. These benchmarks are good for CPU and GPU and there is a need for benchmarks for FPGA or ASIC implementations for comparison of sorting architectures independent of other parameters like memory access etc. In this work, to evaluate the implementation of sorting techniques (Bubble, Bitonic and Odd–even) for power consumption, we define the microbenchmark with the following statistical characteristics:

- The number of test vectors in microbenchmark is 512 and width of each vector can be 8-bit, 16-bit or 32-bit. The input vectors are fed into sorters sequentially and each vector will have a randomness of 50%. It means that each vector will have equal number of ones and zeros. The uniformity in each vector is chosen as 50% to ensure the maximum switching activity in the circuit, so that we can measure maximum dynamic power more accurately.

Table 5 Comparison of power and timing of proposed (with clock gating and multi-Vth) and conventional sorters

Sorter	Bit width	Critical path @ 50 MHz frequency (in nano seconds)				Dynamic power consumption (mW) (@50 MHz)				
		Conventional pipelined sorter	Proposed clock gated sorter	Proposed clock gated with multi-Vth sorter	Reduction using clock gated with multi-Vth (%)	Conventional pipelined sorter	Proposed clock gated sorter	Proposed clock gated with multi-Vth sorter	Reduction using clock gated (%)	Reduction using clock gated with multi-Vth (%)
Bitonic sort	8	6.545	6.776	4.234	64.69	1.3	0.585	0.589	46.59	45.31
	16	6.9	6.723	4.789	69.41	1.47	0.71	0.672	48.9	45.71
	32	7.423	7.733	5.981	80.57	1.88	0.912	1.011	51.83	53.77
Odd–even sort	8	5.845	5.812	3.912	66.93	1.39	0.641	0.653	52.55	46.97
	16	5.987	5.634	4.232	70.69	1.62	0.71	0.689	46.9	42.53
	32	8.412	8.854	5.821	69.20	2.1	0.912	1.101	47.02	52.42
Bubble sort	8	6.598	6.734	4.432	67.17	1.37	0.651	0.662	49.53	48.32
	16	7.378	7.564	4.782	64.81	1.68	0.845	0.849	52.48	50.53
	32	8.231	8.432	5.921	71.94	2.08	1.101	1.210	52.65	58.17

- Each vector is fed sequentially into the sorters and verified for correct functionality and switching activity file (VCD) is generated.

Experimental Analysis with Clock Gating: The experiments are conducted to evaluate the performance of the different hardware sorters with clock gating (multi-Vth not included). The microbenchmark described in the above section is used to generate the switching activity and dynamic power is calculated for the proposed benchmark. The area, timing and power results are presented in Table 4 for an operating frequency of 50 MHz. The synthesis is performed for 50 MHz and gate-level simulations are performed to generate the switching activity at different frequencies and power consumption values are recorded and presented in Table 4.

Experimental Analysis with Multi-Vth: The clock gating and multi-Vth are two important structured low-power techniques which we have incorporated in three different sorters and we have presented the results. The primary objective of deploying the multi-Vth is to improve the performance without increasing the dynamic power. It is evident from the table x that, there is significant reduction in the critical path, so that we can operate the design at higher frequencies across the PVT (Process, Voltage, Temperature) corners. In Table 6, we also present the dynamic power consumption for both Clock gating and Multi-Vth with Clock gating for different frequencies. It is evident from Table 5 that the improvement achieved in improving the critical path without compromising the reduction of power consumption achieved through clock gating is stable across frequencies.

As we have discussed above, there are no standard sorting benchmarks available for architectural evaluation of hardware sorting architectures, and we developed a novel validation benchmark which fits the objective of measurement and analysis of dynamic power consumption across hardware sorting architectures. In this work, we created the randomized testbench using system Verilog and generated the randomized stimulus to match the statistical requirements of the power analysis benchmark.

In this work, we are working with three basic sorter architectures, the number of CAE blocks and the order in which the data are fed influences the switching activity, which in turn will have an effect on dynamic power consumption. The power consumption for conventional sorters, clock gated and clock gated with multi-Vt is presented for the described benchmark.

Comparison

Comparative results for reduction in dynamic power for relevant technique (bitonic sorter) are presented in Table 6.

Table 6 Dynamic power consumption of proposed (with clock gating and multi-Vth) and conventional sorters

Sorter (32-bit)	Operating Frequency (MHz)	Dynamic power consumption (mW)				
		Conventional pipelined sorter	Proposed clock gated sorter	Proposed clock gated with multi-Vth sorter	Reduction (clock gated) (%)	Reduction (clock gated with multi-Vth) (%)
Bitonic sort	20	1.76	0.82	0.89	46.59	50.57
	40	1.82	0.89	0.94	48.9	51.64
	60	1.91	0.99	1.05	51.83	54.97
	80	1.96	1.03	1.11	52.55	56.68
Odd–even sort	20	1.94	0.91	0.98	46.9	50.51
	40	2.02	0.95	1.06	47.02	52.47
	60	2.14	1.06	1.19	49.53	55.61
	80	2.21	1.16	1.22	52.48	55.21
Bubble sort	20	1.88	0.99	1.1	52.65	58.51
	40	2.06	1.05	1.18	50.97	57.28
	60	2.18	1.12	1.21	51.37	55.51
	80	2.23	1.24	1.29	53.91	57.84

Both [9] and [12] are based on bitonic sorting. The technique proposed in [12] reduces the dynamic power significantly up to 89% for 32-bit sorter. The presented values in the table for comparison in Table 7 are normalized values. The formulas used to derive the normalized values are shown below

$$\text{Normalised power}_{\text{proposed}} = \frac{\text{power} \times \text{Excution time} \times 10^2}{V_{DD}^2 \times M \times N},$$

where '*M*' is number of elements sorted and 'N' is width of the element. The calculation for normalized power is presented in [17]. The proposed clock gated 32-bit sorter reduces the dynamic power by 48.51%, which is better than the low-power sorter presented in [9]. The technique presented in [12] 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. The critical paths in the three different sorters are replaced with LVt cells in the place of SVt cells using multi-Vth cells. The proposed low-power bitonic sorter with clock gating and multi-Vth significantly achieves significant reduction of power by 48.51%. Multi-Vth also improves the speed of the circuit by reducing the critical path. Table 7 represents the comparisons between previous sorters and proposed sorters. The performance has been improved without compromising dynamic power consumption achieved through clock gating using multi-Vth cells.

Comparative results for reduction in dynamic power for relevant technique (bitonic sorter) are presented in Table 6. Both [9] and [12] are based on bitonic sorting. The technique proposed in [12] reduces 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 [9]. The technique presented in [12] 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. The critical paths in the three different sorters are replaced with LVt cells in the place of SVt cells using multi-Vth cells. The proposed low-power bitonic sorter with clock gating and multi-Vth significantly achieves significant reduction of power by 48.51%. Multi-Vth also improves the speed of the circuit by reducing the critical path. Table 7 represents the comparisons between previous sorters and proposed sorters. The performance has been improved without compromising dynamic power consumption achieved through clock gating using Multi-Vth cells.

Table 7 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) [12]	Percentage of reduction in comparison with conventional bitonic sorter (low power) [9]	Percentage of reduction in comparison with conventional bitonic sorter with proposed clock gated bitonic sorter	Percentage of reduction in comparison with conventional bitonic sorter with proposed clock gated with multi-Vth bitonic sorter
32-bit	89%	37.01%	48.51%	48.36%

Conclusion

It is evident from the literature survey that the suitable non-merge sorting techniques for hardware implementation are bitonic sorting, bubble sorting and odd–even sorting. The improvements in performance and reduction in power consumption are concerns in designing sorting techniques for hardware implementation. In this paper, we have used well-established structured low-power techniques like clock gating and multi-Vth to achieve low-power consumption on three different hardware sorting techniques and trade-off between area, timing and power consumption is analyzed. Multi-Vth libraries are used to further reduce the critical path or increase the frequency.

The clock gated sorting architectures are compared with state-of-the-art recent low-power sorting schemes and comparative results are presented and multi-Vth implementation will reduce the critical path without compromising the dynamic power consumption reduction achieved through clock gating. 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 analyzed 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.

Declarations

Conflict of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Marcelino R, Neto HC, Cardoso JMP. Sorting Units for FPGA-Based Embedded Systems. In: Proc. IFIP Cong. Distributed Embedded Systems: Design, Middleware and Resources, 2008; 11–22.
2. Sklyarov V, Skliarova I. High-performance implementation of regular and easily scalable sorting networks on an FPGA. Elsevier J Microprocess Microsyst. 2014;38(5):470–84.
3. Alaparthi S, Gulati K, Khatri SP. Sorting binary numbers in hardware—a novel algorithm and its implementation. In: 2009 IEEE International Symposium on Circuits and Systems, 2009, pp. 2225–2228.
4. Dayarathna M, Wen Y, Fan R. Data center energy consumption modeling: a survey. IEEE Commun Surv Tutor. 2016;18(1):732–94.
5. Falsafi B, Dally B, Singh D, Chiou D, Yi JJ, Sendag R. FPGAs versus GPUs in data centers. IEEE Micro. 2017;37(1):60–72.
6. Preethi P, Mohan KG, Sudeendra Kumar K, Kamala Kanta Mahapatra. Low power sorters using clock gating. In: 7th IEEE International Symposium on Smart Electronic Systems (iSES-2021) 2021.
7. Thompson CD. The VLSI complexity of sorting. IEEE Trans Comput. 1983;32(12):1171–84.
8. Farmahini-Farahani A, Duwe HJ III, Schulte MJ, Compton K. Modular design of high-throughput, low-latency sorting units. IEEE Trans Comput. 2013;62(7):1389–402.
9. Lin S, Chen P, Lin Y. Hardware design of low-power high-throughput sorting unit. IEEE Trans Comput. 2017;66(8):1383–95.
10. Pedroni V, Jasinski RP, Pedroni RU. Panning sorter: an approach to the design of minimal-hardware parallel-input data sorters. Electron Lett. 2010;46(18):1262–3.
11. Abdel-Hafeez S, Gordon-Ross A. An efficient O(N) comparison–free sorting algorithm. IEEE Trans Very Large-Scale Integr VLSI Syst. 2017;25(6):1930–42.
12. Najafi MH, Lilja DJ, Riedel MD, Bazargan K. Low- cost sorting network circuits using unary processing. IEEE Trans Very Large-Scale Integr VLSI Syst. 2018;26(8):1471–80.
13. Norollah A, Kazemi Z, Beitollahi H, Hely D. Hardware support for efficient and low-power data sorting in massive data application: the 3D sorting method. IEEE Consum Electron Mag. 2021;11:87–94.
14. Rabaey JM. Low power design essentials. Springer; 2009. (**ISBN: 978-0-387-71712-8**).
15. Madaminov S. Comprehensive framework for sorting benchmarks. In: Proceedings of the VLDB 2019 PhD Workshop, California, August 26th, 2019.
16. Norollah A. RTHS: a low-cost high-performance real-time hardware sorter, using a multidimensional sorting algorithm. IEEE Trans Very Large Scale Integr VLSI Syst. 2019;99:1–13.
17. Locharla GR, Kallur SK. Implementation of MIMO data reordering and scheduling methodologies for eight-parallel variable length multi-path delay commutator FFT/IFFT. In: IET computers and design techniques, VDAT, 2016.