# Exploring Power Advantage of Binary Search: An Experimental Study

Muhammad Al-Hashimi[1], Naif Aljabri[2]
Department of Computer Science
King Abdulaziz University, Jeddah, Saudi Arabia

***Abstract*—As exascale systems come online, more ways are needed to keep them within reasonable power budgets. This study aims to help uncover power advantages in algorithms likely ubiquitous in high-performance workloads such as searching. This study explored the power efficiency of binary search and its ternary variant, comparing consumption under different scenarios and workloads. Accurate modern on-chip integrated voltage regulators were used to get reliable power measurements. Results showed the binary version of the algorithm, which runs slower but relies on a barrel-shifter circuit, to be more power efficient in all studied scenarios offering an attractive time-power tradeoff. The cumulative savings were significant and will likely be valuable where the search may be a substantial fraction of workloads, especially massive ones.**

***Keywords*—Binary search; ternary search; time-power tradeoff; exascale computing; barrel shifter**

## I. Introduction

The fastest supercomputers in the world today are capable of solving problems at the petascale level, i.e., $10^{15}$ floating point operations/second (flops). Complex simulations of more realistic models are becoming less feasible on current petascale supercomputers. These workloads need exascale ($10^{18}$) machines. The main obstacle to exascale was the projected power consumption [1]–[4]. If petascale-era technology had been scaled, it would have consumed absurd amounts of power in the order of gigawatts. The US Defense Advanced Research Projects Agency (DARPA) specified that the peak power consumption should be below 20 megawatts (MW), which led [4] to conclude that substantial advances in hardware, software, and algorithms were needed to meet that requirement. Some experts [5], [6] had predicted that a complete exascale system would enter service by 2020. The first one that also meets the DARPA provision arrived in 2022 [7]. The Frontier supercomputer (Oak Ridge National Lab, USA) made its official debut on the 2022 Top500 list as the first supercomputer to exceed 1.0 exaflops [8], [9]. Engineers project reaching 1.5 exaflops peak at 29 MW (19.33 MW/exaflop) [10].

The previous studies on power and energy reduction focused on architecture, chip technology, algorithm optimization, and management. This study aims to explore the inherent power advantages in algorithms, i.e., those stemming from the method. It pays particular attention to the power (the time rate), which once was an obstacle in the path to exascale and likely will continue to be a significant concern. The work relies on direct measurement and observations at the microarchitectural level. It encourages rethinking time optimization, which has long been a classic interest in computing, as it may sometimes interfere with the power concerns of systems where saving power may be of primary interest. With those concerns in mind, [11] examined the power efficiency of mergesort against that of quicksort, the standard general-purpose sorting algorithm. Their results showed that the power consumption of the mergesort was significantly better than an optimized quicksort when it exploited the barrel shifter, a digital binary shift circuit, to do the partitioning phase. The barrel shifter is a simple, power-efficient component in modern processors utilized to perform powers (exponent) of two divisions and multiplications in one clock cycle. The quicksort ran faster on average, as expected. Hence switching sorting methods did involve a time-power tradeoff.

The preceding encouraged a look into more algorithms that rely on the shifter hardware to see if similar power trends hold elsewhere. Accordingly, binary search should perhaps have a power advantage against a ternary version since it divides search lists by two, which could utilize the power-efficient barrel shifter. This study attempts to investigate and quantify that advantage. Binary search is a log-efficient procedure for unordered querying an array of ordered keys based on an implicit optimal binary tree [12, see S. 6.2], which is very useful for multiple random queries. It is a significant part of various general workloads [13]. Conceptually, it looks for a key value $X$ in a sorted list as follows:

```
if ( X == middle element )
   X is found
else if ( X < middle element )
   search 1st half of list using the same method
else
   search 2nd half of list using the same method
```

A *ternary* search algorithm works as outlined next:

```
if ( X == middle1 element )
   X is found
if ( X == middle2 element )
   X is found
else if ( X < middle1 element )
   search 1st third of list using the same method
else if ( X > middle2 element )
   search 3rd third of list using the same method
else
   search 2nd third of list using the same method
```

The rest of the paper is organized as follows: Section II reviews related work in chronological order. Section III describes the experiments. In Section IV, findings are presented and discussed. Finally, concluding remarks and suggestions for further research are in Section V.

## II. Related Work

The research in [14]–[16] introduced a new energy management component and used it to select the best energy-saving sorting algorithm on mobile platforms. The energy management component selects the best algorithm dynamically. The results show good energy saving and encourage more investigation.

The author in [17] presented a new methodology to estimate power consumption for parallel computation in a multicolor environment. They also prove that the total power consumption for the computation is not equal to the sum of each core individually. The results show that their measurement accuracy was 95%. In [18], the author enhanced the performance and energy consumption for a parallel program running on the Intel Core i7-2600 processor by increasing the data locality. They increased energy efficiency by more than six times.

The author in [19] presented a methodology to predict the energy consumption for algorithms using the algorithm atomic operations analysis. They use it to evaluate the energy consumption for the classical bubble sort algorithm. Their prediction methodology accuracy was more than 95%. The author in [20] used the scheduler control information to estimate the needed number of cores to execute the waiting program threads. Their results show some power savings. The author in [21] developed a power-efficient algorithm based on mathematical modeling for sorting on a mesh-connected network that utilized an optical pyramid layout (on chips equipped with optics capabilities). They showed a reduction in energy consumption due to data movement based on energy modeling vs. time. There was no direct assessment of power. The author in [22] presented a new methodology to save power in mobile platforms using Intel Core i5 processor by running different workloads on a limited number of cores. They reduce the power consumption by 40%.

In [23], the author presented a methodology to save power consumption in multicore environments by reducing the thread idle time. They applied it to the concurrent execution of ILUPACK (an *LU* decomposition software). The results showed that they achieved good savings in power. In [24], the author presented a new method to save energy by adopting an energy-efficient I/O management approach in supercomputers. They examined three radically different I/O schemes, including time partitioning, dedicated cores, and dedicated nodes. They also characterized how different configurations of the application and the system can impact performance and energy consumption. Reference [25] translated the CPU dissipated power into tokens to select the best power-saving technique. The dissipated power was predicted at cycle level and basic block level. They enhanced the energy consumption by 11%.

The authors in [26] and [27] observed power efficiency characteristics of the barrel shifter circuit in the context of digital signal processing where powers of two divisions and multiplications are dominant.

The author in [28] investigated power and energy consumption of mergesort against that of quicksort running on the Intel Xeon E5-2640 (Sandy Bridge). That study had two drawbacks. First, it eliminated some, but not enough, side activities that could contribute to the readings. Second, it relied on an older part that did not support fine power control or detailed internal instrumentation, including crucial cache activity information. Nevertheless, their results identified a statistically significant power advantage for basic mergesort. The author in [29], [30] developed a methodology for evaluating power and energy consumption on the NVIDIA Tesla K40c GPU and used it to compare GPU-optimized bitonic mergesort and quicksort. Their results showed that the mergesort outperformed quicksort in power consumption. The bitonic mergesort was further analyzed in [31] for power and energy consumption on the NVIDIA platform. They identified the factors that caused the mergesort to have a power advantage (in a follow-up study, [32] showed that varying the block size on the GPU further improved the power performance of the mergesort).

In [11], the author developed a rigorous experimental procedure to eliminate sources that could taint measurements based on a CPU with improved power instrumentation. They showed natural mergesort to have a clear power advantage against a highly optimized 3-way quicksort. A meta-analytical comparison of the energy consumption of mergesort and quicksort was conducted in [33] based on an extensive literature review that included [11]. The study concluded that there was no significant difference in energy needs. Results in [11] had earlier agreed on energy but found the mergesort to consume less power based on direct measurement, stressing a concern when two systems go through the same energy budget at different rates. It is the basis of time-power tradeoffs. In some applications, the time rate, i.e., power, is perhaps more consequential.

## III. Experimental Design and Procedures

Realistic measurement of the power consumption of a program running on a CPU is not easy. In a modern run environment, a program runs on a CPU with many other pieces of code that share the CPU and other resources. Programs run on many CPU cores under OS control. Therefore environmental factors, which confound measurement that may credibly be attributed to the experiment's code, had to be eliminated. Some challenges are not unlike those faced when profiling program run time or studying the time behavior of an algorithm. In addition, empirical assessment of power characteristics presents extra challenges. Ambient and CPU-generated heat act as thermal noise that can distort measurements. A modern CPU actively manages power, adjusting its cooling fan speeds, internal voltages, and frequencies to maintain health and keep power consumption in check, which changes power performance unpredictably. In this study, the authors strove to eliminate as many external factors and sources of power consumption as needed to ensure proper, consistent readings that serve the purpose of the study. The classic generic iterative versions of the binary and ternary search algorithms, hereafter referred to as BS and TS, respectively, were used. Finally, timing data was collected as an experimental control to verify programming and expected complexity behavior.

The following describes the experimental environment, the tools used to profile power, the executables that code the algorithms, and the datasets and related test procedures.

TABLE I. SPECIFICATIONS OF THE TEST BED WORKSTATION

| | |
|---|---|
| **Processor** | Intel Xeon E5-2680 v3 (Haswell-EP)<br>12 Cores (24 Logical)<br>TDP Limit = 120 W |
| **Cache Memory** | L1/core: 32 KB Instruction, 32 KB Data<br>L2/core = 256 KB<br>L3 Shared = 30 MB |
| **OS** | Linux Ubuntu 16.04 LTS 64-bit |
| **Physical Memory (RAM)** | 8.00 GB |

### A. The Environment

The test bed machine was a FUJITSU HPC workstation kept under consistent server-room conditions to control ambient thermal effects. Specifications are listed in TABLE I. The experiments ran with the power supply fan ON. The processor environment is an HPC-grade Intel Xeon CPU that appears in machines such as the SGI ICE X supercluster, which ranked 40 in the 2015 Top500 list.

The following settings were applied to the CPU and the OS to eliminate thermal noise and other factors that could contaminate the power readings:

- Disabled hyper-threading (architectural optimization).
- Disabled power management options (architectural optimization).
- Disabled the CPU fan to eliminate effects of cooling (an external fan was used between runs to bring the CPU back to a consistent initial thermal state).
- Closed unnecessary OS services and processes to run the OS with minimum resources.
- Moved the necessary OS processes and services to cores 2 and above to dedicate core 0 for the experiments.
- Left core 1 idle to reduce thermal interference from the other cores.

### B. The Profiler

The Linux *perf* tool was used to profile the CPU. The profiler, part of the Linux kernel, provides a wealth of information about the CPU, including power and energy consumption. It largely depends on the RAPL (running average power limiting) interface, which, in turn, depends on the quality of CPU state control and probe components embedded in the processor. The HPC-grade 2600v3 Haswell-series, including the part used in this work, featured [34]: 1) a shift from an indirect model-based reporting to actual measurement based on fully integrated voltage regulators (FIVR), and 2) the addition of per core voltage regulators that control the frequency and power states of cores individually. Previous generations relied on one mainboard regulator to infer detailed information. A 2018 study found that RAPL closely matched plug power readings in the Haswell architecture [35]. As a result, the profiler can provide reliable, higher-resolution instrumentation on the Xeon 2680v3. The tool was configured to obtain an average of 300 runs and used the following command for each one.

```
perf stat -e power/energy-pkg ./algorithm
```

```
        Disassembly of section .text:

        00000000004005b6 <binarySearch(int*, int, int)>:
        _Z12binarySearchPiii():
          push   %rbp
 0.31     mov    %rsp,%rbp
          mov    %rdi,-0x18(%rbp)
          mov    %esi,-0x1c(%rbp)
          mov    %edx,-0x20(%rbp)
 0.31     movl   $0x0,-0xc(%rbp)
          mov    -0x1c(%rbp),%eax
          sub    $0x1,%eax
          mov    %eax,-0x8(%rbp)
 2.83 1e: mov    -0xc(%rbp),%eax
 1.26     cmp    -0x8(%rbp),%eax
        ↓ jg     8b
          mov    -0xc(%rbp),%edx
 1.89     mov    -0x8(%rbp),%eax
 1.26     add    %edx,%eax
          mov    %eax,%edx
 0.94     shr    $0x1f,%edx
 1.57     add    %edx,%eax
 1.57     sar    %eax
 1.26     mov    %eax,-0x4(%rbp)
13.84     mov    -0x4(%rbp),%eax
 3.14     cltq
 3.14     lea    0x0(,%rax,4),%rdx
          mov    -0x18(%rbp),%rax
 4.40     add    %rdx,%rax
24.84     mov    (%rax),%eax
          cmp    -0x20(%rbp),%eax
10.69   ↓ jne    5a
 3.46     mov    -0x4(%rbp),%eax
        ↓ jmp    90
 0.31 5a: mov    -0x4(%rbp),%eax
          cltq
          lea    0x0(,%rax,4),%rdx
 4.09     mov    -0x18(%rbp),%rax
```

Fig. 1. Barrel Shifter Instructions SAR and SHR in Disassembled BS Code.

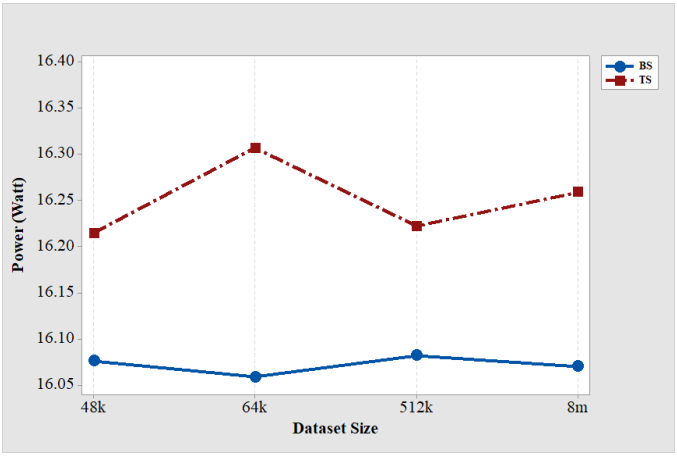
TABLE II. DATASETS AND RUN SCENARIOS

| Dataset Size | | Number Datasets | Number Runs | Total Runs |
|---|---|---|---|---|
| Set 1 | 48,000 | 20 | 300 | 20×300 = 6000 |
| Set 2 | 64,000 | 20 | 300 | 20×300 = 6000 |
| Set 3 | 512,000 | 20 | 300 | 20×300 = 6000 |
| Set 4 | 8,000,000 | 20 | 300 | 20×300 = 6000 |

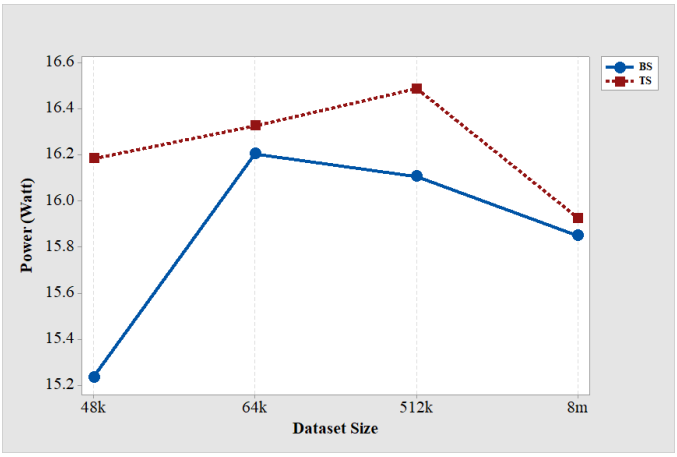
### C. The Executable Files

The search code was generated using the GCC 64-bit compiler in release mode. All optimization options in the compiler were disabled to avoid those that replace bare code with parts optimized for specific architectural features. Optimizations do not serve the purpose of the experiments. They may result in readings that reflect the machine or the compiler more than the algorithms in principle. The authors consider bare code a fair rendition of the algorithm on machines from the same instruction set architectural (ISA) style. Turning off power features in an otherwise power-efficient CPU also fits that concern. Finally, disassembled code from the BS executable was examined to confirm that it utilized the shifter circuit instructions to halve the sizes of search lists (Fig. 1).

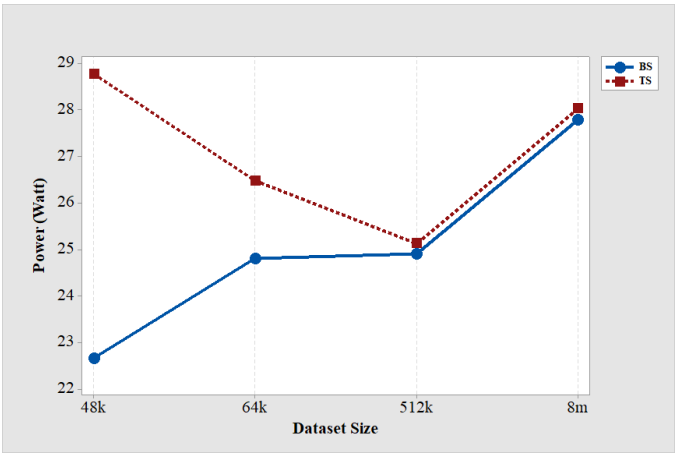
TABLE IV shows the average power consumption for various dataset sizes under the different run scenarios for each search algorithm. The averages were calculated from 290 runs/list (after discarding the first ten of 300) for 20 different lists. So each power value in the table represents 5800 runs. Fig. 2 show these findings comparatively.




(a) Similar-Iterations




(b) Single-Element




(c) All-Elements

Fig. 2. Average Power Consumption in Watts for Search Scenarios.

TABLE III. NUMBER OF ITERATIONS FOR SIMILAR-ITERATIONS AND SINGLE-ELEMENT SEARCH SCENARIOS

| Dataset Size | Similar-Iterations | | Single-Element | |
|---|---|---|---|---|
| | BS | TS | BS | TS |
| 48k | 9 | 9 | 14 | 9 |
| 64k | 9 | 9 | 16 | 9 |
| 512k | 11 | 11 | 19 | 11 |
| 8M | 15 | 15 | 23 | 15 |

### D. The Datasets

Lists of unique random integers, sorted in ascending order, were generated in four size sets, each comprising 20 different lists. Each dataset ran 300 times for a total number of 24,000 runs. The first ten runs/dataset were discarded to avoid effects from the initial thermal state of the CPU. See TABLE II for a summary of the datasets and run scenarios.

The authors chose 300 after running test trials and found that averages tended to start converging after 230 runs. Dataset sizes were chosen based on cache miss data from [11], where they used the same CPU. Some experimentation with different sizes confirmed that power readings for datasets located in the same cache level were reasonably similar. So sizes were based on crossing L1, L2, and L3 cache boundaries. Sizes of sets 1 and 2 represent lists in L1 and L2, respectively, and those of sets 3 and 4 represent small and large lists in the L3 cache, taking care not to cross over to DRAM. The DRAM was excluded for two reasons. First, to maintain comparable readings since the on-chip SRAM-based cache levels likely share the same physical characteristics, hence the power profile. Second, to better discern effects from algorithmic operations that the off-chip DRAM access may further drown. Data movement along communication links is a known prominent contributor to power consumption, so the decision works for the authors' approach to emphasizing the algorithm.

## IV. RESULTS AND DISCUSSION

To avoid unpredictable effects from the best and worst cases on averages and to gain better insights, the experiments ran in three controlled search scenarios:

1) **Similar-iterations search**: The scenario picks an element near the middle of the randomly-generated list, then calculates the iterations needed to find this element. Each algorithm needs a different number of iterations to find it. The smallest is selected to set both to run iterations equal to that number. For example, if BS needed nine iterations and TS needed eleven, both are set to run nine iterations. It represents the minimum run cost to find the key. It also quantifies the power cost of searches/finds that take the same amount of iterating, which may be a fair basis to compare the algorithms. TABLE III shows the number of iterations used for this scenario.
2) **Single-element search**: Again, set the two algorithms to search for an element near the middle of the list. Here each algorithm runs all the needed iterations to find it. This scenario quantifies the full cost of a find that can compare with the first scenario. The number of iterations for this scenario is in TABLE III.

TABLE IV. AVERAGE POWER IN WATTS (COLUMN Δ% SHOWS PERCENT INCREASE FROM BS TO TS)

| Dataset Size | Similar Iterations | | | Single Element | | | All Elements | | |
|---|---|---|---|---|---|---|---|---|---|
| | BS | TS | Δ% | BS | TS | Δ% | BS | TS | Δ% |
| 48k | 16.08 | 16.21 | **0.81** | 15.23 | 16.18 | **6.24** | 22.65 | 28.77 | **27.02** |
| 64k | 16.06 | 16.31 | **1.56** | 16.20 | 16.33 | **0.80** | 24.80 | 26.48 | **6.77** |
| 512k | 16.08 | 16.22 | **0.87** | 16.11 | 16.49 | **2.36** | 24.90 | 25.13 | **0.92** |
| 8M | 16.07 | 16.26 | **1.18** | 15.85 | 15.92 | **0.44** | 27.78 | 28.03 | **0.90** |

3) **All-element search**: This scenario sets each algorithm to search for all the elements in the dataset in each run. For example, if the dataset size is 100, the algorithm searches for element 1, then 2, until 100. Some element locations are the worst for the algorithms, while others are the best. The power consumption of all searches is measured to avoid the impact of the element's location on the power consumption and averaged to find the average power consumption for all the positions for each algorithm.

The results show that BS was consistently more power efficient, with slim margins in the similar-iterations and single-element search scenarios (datapoint 64k-single-element appears anomalous). Power savings, however, were substantial in the all-elements case. The savings seem to depend on the cache level. They seem to diminish as the computation spills to the next cache level and then stabilize at around 1% in L3.

Before discussing the results, readers are minded that both algorithms are incredibly fast, with TS holding a slight runtime edge. They go quickly through keys to reach either a found or not found outcome. The run times are comparable since both belong to the logarithmic efficiency class (TS in base 3, hence its edge). Therefore, it is perhaps unsurprising to see close power readings in the similar-iterations and single-element search scenarios. They both depict an isolated query. Notably, BS still manages to hold a consistent edge in power consumption. In the similar-iterations case, had BS taken as few iterations as TS, it would have consumed less power, i.e., it had a lower power cost of iterating. The all-elements search scenario paints a more realistic picture of what to expect as a cumulative effect likely to be experienced during an extended period. It is where those seemingly minor differences observed in the other two scenarios come into play.

Furthermore, cache memory interferes with the run time performance of the binary search (likewise the TLB, if DRAM is involved) [13]. With large subarrays, binary search is prone to generating indices to elements in distant memory. The effects of weaker locality increase as datasets grow in size. Power consumption should follow a similar trend. Power overheads due to cache miss rise as more physical circuits are engaged to satisfy the misses. Interactions with cache memory generally may explain the decline in power savings as the effects from misses increase. Therefore, limiting list sizes was conducive to the aims of the experiments. The L1 cache is expected to yield the most realistic view of power from the algorithmic operations since it suffers the least from the power overheads of misses.

Little differences in power consumption may not look promising when viewed from a single application standpoint. In a large-scale system, however, the cumulative effect should be significant, especially for a computation likely a good fraction of the general workload for extended times like searching. The savings may be substantial, which may be more valuable for a power-sensitive system like an exascale supercomputer. Moreover, those savings were really due to the algorithm. BS calls for an inherently inexpensive operation on a binary computer, which reflects in simple power-efficient hardware, not the other way around. If not mindful, it may be tempting to pick the ternary version to speed up runs, especially since the cost of implementation is negligible.

The findings seem to support the idea that some algorithms have a power (not energy necessarily) advantage over those that need to work harder, i.e., expend energy at a higher rate, to achieve the same results. The inherent power efficiency of an operation reflects in the hardware, as is the case with the division by two on a binary device. One may expect an algorithm that relies on that operation to be more power efficient. This observation should be interesting where power savings are more significant than some runtime gains, such as those offered by TS. Eventually, interactions with microarchitectural features, such as those underlying the memory, may erode those savings in the real world. However, the savings can add up system-wide in pervasive computations. Thus a massive exascale system may be in a better position to benefit from the cumulative savings. Mobile systems should also benefit from power-focused optimizations since they relate to the critical concern of how quickly a battery drains. A power-efficient *workload* helps a smaller (lighter) battery go a long way.

## V. CONCLUSIONS

This work experimentally explored the power efficiency of two variants of binary search: the classic binary decision (BS) and the faster-running ternary decision version (TS), both in the same time efficiency class. The authors tuned the experimental environment to focus on the search code while eliminating other run environment factors. Under the experimental conditions, the results showed that BS outperformed TS in power consumption under different search scenarios. The BS could take advantage of the more efficient shift hardware. The findings seem to corroborate the authors' initial suspicions about BS, which stemmed from work reported in [11]. The savings in BS were likely inconsequential for individual searches, but the cumulative effects may be significant.

The study has been an early attempt to establish the binary decision version of binary search as a preferred alternative for exascale applications. It quantified a power advantage and exposed a potentially interesting tradeoff with time, especially from a workload-wide view. Further investigation with different scenarios and environments may be needed to argue that the reported findings stem from the method (algorithmic) rather than the run environment, i.e., a binary decision is inherently efficient in terms of power consumption on any binary computer. Future investigations in other processor environments

would be interesting, for example, later core processors or GPUs whose importance in HPC continues to grow. Higher resolution instrumentation and better reporting tools should provide deeper insights. Such studies may uncover trends independent of architecture. Eliminating other microarchitectural interferences help further amplify behaviors due to the algorithm. For example, basic binary search is quite branchy, so it may be helpful to turn off branch prediction. Other algorithms that rely on the binary shift hardware may be worth exploring. Finally, this work focused on average power. It may be interesting to observe peak power, which should be of interest from a system-wide workload in a massive supercomputer viewpoint.

## ACKNOWLEDGMENT

This work was funded by King Abdulaziz University, under grant No. (1-611/1433 HiCi). The authors, therefore, acknowledge technical and financial support of KAU.

## REFERENCES

[1] K. Bergman, S. Borkar, D. Campbell, W. Carlson *et al.*, "Exascale computing study: Technology challenges in achieving exascale systems Peter Kogge, editor & study lead," DARPA Information Processing Techniques Office, Exascale Working Group Report, September 2008.

[2] S. Ashby, P. Beckman, J. Chen, P. Colella *et al.*, "The opportunities and challenges of exascale computing," US Department of Energy Office of Science, Summary Report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee, Fall 2010.

[3] O. Villa, D. R. Johnson, M. O'Connor, E. Bolotin, D. Nellans, J. Luitjens, N. Sakharnykh, P. Wang, P. Micikevicius, A. Scudiero *et al.*, "Scaling the power wall: a path to exascale," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 2014, pp. 830–841.

[4] D. A. Reed and J. Dongarra, "Exascale computing and big data," *Communications of the ACM*, vol. 58, no. 7, pp. 56–68, 2015.

[5] P. Beckman, "On the road to exascale," *Scientific Computing World*, no. 116, pp. 26–28, 2011.

[6] J. Hsu, "When will we have an exascale supercomputer?" *IEEE spectrum*, vol. 52, no. 1, pp. 13–16, 2015.

[7] D. Schneider, "The exascale era is upon us," *IEEE spectrum*, vol. 110, no. 1, pp. 34–35, 2022.

[8] T. Trader, "Top500: Exascale is officially here with debut of frontier," *HPCwire*, 2022. [Online]. Available: https://www.hpcwire.com/?p=142590

[9] C. Q. Choi, "Beneath the hood of the first exascale computer," *IEEE spectrum*, vol. 59, no. 8, pp. 8–9, 2022.

[10] T. Trader, "US closes in on exascale: Frontier installation is underway," *HPCwire*, 2021. [Online]. Available: https://www.hpcwire.com/?p=125875

[11] N. Aljabri, M. Al-Hashimi, M. Saleh, and O. Abulnaja, "Investigating power efficiency of mergesort," *The Journal of Supercomputing*, vol. 75, pp. 6277–6302, 2019.

[12] D. E. Knuth, *The art of computer programming*, 2nd ed. Addison-Wesley, April 2018 (36th printing), vol. 3 (Sorting and Searching).

[13] P.-V. Khuong and P. Morin, "Array layouts for comparison-based searching," *ACM Journal of Experimental Algorithmics*, vol. 22, no. 1, pp. 1.3:1–1.3:39, may 2017.

[14] C. Bunse, H. Höpfner, S. Roychoudhury, and E. Mansour, "Energy efficient data sorting using standard sorting algorithms," in *International Conference on Software and Data Technologies*. Springer, 2009, pp. 247–260.

[15] C. Bunse, H. Höpfner, E. Mansour, and S. Roychoudhury, "Exploring the energy consumption of data sorting algorithms in embedded and mobile environments," in *Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on*. IEEE, 2009, pp. 600–607.

[16] C. Bunse, H. Höpfner, S. Roychoudhury, and E. Mansour, "Choosing the "best" sorting algorithm for optimal energy consumption," in *ICSOFT (2)*, 2009, pp. 199–206.

[17] R. Basmadjian and H. de Meer, "Evaluating and modeling power consumption of multi-core processors," in *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*. ACM, 2012, p. 12.

[18] A. Al Hasib, P. G. Kjeldsberg, and L. Natvig, "Performance and energy efficiency analysis of data reuse transformation methodology on multicore processor," in *European Conference on Parallel Processing*. Springer, 2012, pp. 337–346.

[19] Y. Yuechuan, Z. Guosun, D. Chunling, and W. Wei, "Analysis method of energy for c source program and its application," in *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*. IEEE, 2013, pp. 1397–1402.

[20] T. Kodaka, A. Takeda, S. Sasaki, A. Yokosawa, T. Kizu, T. Tokuyoshi, H. Xu, T. Sano, H. Usui, J. Tanabe *et al.*, "A near-future prediction method for low power consumption on a many-core processor," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 1058–1059.

[21] P. Poon and Q. F. Stout, "Time-power tradeoffs for sorting on a mesh-connected computer with optical connections," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*. IEEE, 2013, pp. 611–619.

[22] F. Hamady, A. Kayssi, A. Chehab, and M. Mansour, "Evaluation of low-power computing when operating on subsets of multicore processors," *Journal of Signal Processing Systems*, vol. 70, no. 2, pp. 193–208, 2013.

[23] J. I. Aliaga, M. Barreda, M. F. Dolz, A. F. Martín, R. Mayo, and E. S. Quintana-Ortí, "Assessing the impact of the cpu power-saving modes on the task-parallel solution of sparse linear systems," *Cluster computing*, vol. 17, no. 4, pp. 1335–1348, 2014.

[24] O. Yildiz, M. Dorier, S. Ibrahim, and G. Antoniu, "A performance and energy analysis of i/o management approaches for exascale systems," in *Proceedings of the sixth international workshop on Data intensive distributed computing*. ACM, 2014, pp. 35–40.

[25] J. M. Cebrián, D. Sánchez, J. L. Aragón, and S. Kaxiras, "Managing power constraints in a single-core scenario through power tokens," *The Journal of Supercomputing*, vol. 68, no. 1, pp. 414–442, 2014.

[26] S. K. Mitra and A. R. Chowdhury, "Optimized logarithmic barrel shifter in reversible logic synthesis," in *VLSI Design (VLSID), 2015 28th International Conference on*. IEEE, 2015, pp. 441–446.

[27] M. M. Kamble and S. P. Ugale, "FPGA implementation and analysis of different multiplication algorithm," *International Journal of Computer Applications*, vol. 149, no. 2, pp. 33–36, 2016.

[28] M. Al-Hashimi, M. Saleh, O. Abulnaja, and N. Aljabri, "On the power characteristics of mergesort: an empirical study," in *Advanced Control Circuits Systems (ACCS) Systems & 2017 Intl Conf on New Paradigms in Electronics & Information Technology (PEIT), 2017 Intl Conf on*. IEEE, 2017, pp. 172–178.

[29] M. A. Al-Hashimi, O. A. Abulnaja, M. E. Saleh, and M. J. Ikram, "Evaluating power and energy efficiency of bitonic mergesort on graphics processing unit," *IEEE Access*, vol. 5, pp. 16 429–16 440, 2017.

[30] M. J. Ikram, O. A. Abulnaja, M. E. Saleh, and M. A. Al-Hashimi, "Measuring power and energy consumption of programs running on kepler GPUs," in *Advanced Control Circuits Systems (ACCS) Systems & 2017 Intl Conf on New Paradigms in Electronics & Information Technology (PEIT), 2017 Intl Conf on*. IEEE, 2017, pp. 18–25.

[31] O. A. Abulnaja, M. J. Ikram, M. A. Al-Hashimi, and M. E. Saleh, "Analyzing power and energy efficiency of bitonic mergesort based on performance evaluation," *IEEE Access*, vol. 6, pp. 42 757–42 774, 2018.

[32] M. J. Ikram, M. E. Saleh, M. A. Al-Hashimi, and O. A. Abulnaja, "Investigating the effect of varying block size on power and energy consumption of GPU kernels," *The Journal of Supercomputing*, vol. 78, pp. 14 919–14 939, 2022.

[33] G. Dlamini, F. Jolha, Z. Kholmatova, and G. Succi, "Meta-analytical comparison of energy consumed by two sorting algorithms," *Information Sciences*, vol. 582, pp. 767–777, 2022.

[34] D. Hackenberg, R. Schöne, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer, "An energy efficiency feature survey of the intel haswell processor," in *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, 2015, pp. 896–904.

[35] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen, and Z. Ou, "Rapl in action: Experiences in using rapl for power measurements," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 3, no. 2, pp. 9:1–9:26, 2018.