

The Role of Direct Memory Access (DMA) in Reducing CPU Load on High-Speed Data Transfer: A Literature Review

Rizky Parlika¹, Fathur Rahman², Yoga Ari Tofan³

^{1,2,3}Informatika, Fakultas Ilmu Komputer, Universitas Pembangunan Nasional Veteran Jawa Timur, Surabaya, Indonesia
¹rizkyparlika.if@upnjatim.ac.id, ²25081010057@student.upnjatim.ac.id, ³yoga.if@upnjatim.ac.id

Article Info

Article history:

Received March 31, 2026

Revised April 8, 2026

Accepted May 2, 2026

Keywords:

Direct Memory Access

DMA

CPU load

transfer data

computer architecture

ABSTRACT

Direct Memory Access (DMA) plays a crucial role in reducing CPU load in high-speed data transfer systems. This literature review examines how DMA enables direct data transfer between memory and I/O devices without CPU intervention, thereby improving overall system efficiency. The primary issue identified across the literature is the occurrence of CPU bottlenecks in conventional transfer protocols such as Programmed I/O and Interrupt-driven I/O, which cause performance degradation in applications such as High-Performance Computing (HPC) and the Internet of Things (IoT). Through analysis of literature from various scientific sources, including IEEE and ACM journals, this review finds that DMA implementation has been reported to reduce CPU load by up to 80% in gigabit-speed transfer scenarios and significantly decrease processor time compared to conventional methods. These findings underscore the importance of DMA integration in the design of future computing systems, with recommendations for further research on hybrid architectures and DMA implementation in heterogeneous computing.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Yoga Ari Tofan

1. INTRODUCTION

Data transfer efficiency is one of the determining factors of modern computer system performance. The ability of a system to perform processing quickly and efficiently is highly dependent on the data transfer method used between system components [1]. In conventional computer architecture, there are three primary methods of data transfer between the CPU and I/O devices: Programmed I/O, Interrupt-driven I/O, and Direct Memory Access (DMA) [2].

In the Programmed I/O method, the CPU must actively manage every data transfer cycle, rendering it unable to perform other tasks during the transfer process. The Interrupt-driven I/O method offers a slight improvement by allowing the CPU to perform other tasks until it receives an interrupt signal; however, the CPU must still be involved in every data movement. Both methods are identified as primary causes of CPU inefficiency, particularly in large-scale data transfers [2], [3].

Direct Memory Access (DMA) addresses these problems. DMA enables direct data transfer between memory and I/O devices without CPU intervention, allowing the CPU to be allocated to other computational tasks simultaneously. Various studies report that DMA usage can significantly reduce CPU load compared to conventional methods [3].

The application of DMA extends beyond general computing. In networking, Remote Direct Memory Access (RDMA) technology enables direct data transfer between computer memories without involving the operating system or CPU on either side, resulting in zero-copy networks with very low overhead [4], [5]. In SmartNIC and FPGA infrastructure, DMA cores facilitate the transfer of large data volumes directly to host memory, reducing CPU intervention and preventing congestion on PCIe communication paths [6]. In the field

of hardware acceleration for Machine Learning (ML), DMA engines in modern GPUs such as the AMD Instinct MI300X offload communication from processing units, minimizing disruptions to computation and memory when kernels run concurrently [7]. Similarly, in CUDA programming, streaming mechanisms are used to hide data transfer latency through PCIe by overlapping kernel execution and data transfer in parallel [8].

DMA efficiency has also proven important in low-power and wearable devices. DMA usage allows microcontrollers to transfer sensor data to storage media (e.g., SD cards) while the CPU remains in sleep mode during the transfer process, significantly reducing power consumption and extending device battery life [9].

Given the broad application of DMA across various computing domains, this literature review aims to analyze the working mechanisms and effectiveness of DMA in reducing CPU load across diverse high-speed data transfer scenarios. Specifically, this review addresses the application of DMA in: (1) general computing, (2) high-speed data networks, and (3) graphics accelerators and specialized hardware. Through synthesis of findings from various literature sources, this study is expected to provide a comprehensive overview of the role of DMA as a fundamental solution for overcoming I/O bottlenecks in modern computer architectures [10].

2. METHODOLOGY

2.1. Research Type

This study employs a literature review method with a descriptive-analytical approach. A literature review was selected because the research objective is to map and analyze published findings regarding the effectiveness of Direct Memory Access (DMA) in reducing CPU load on high-speed data transfer, rather than conducting direct experimental testing.

2.2. Data Sources

The literature used in this review was obtained from reputable scientific databases, including IEEE Xplore, ACM Digital Library, and Google Scholar. Searches were conducted using the following primary keywords: "Direct Memory Access", "DMA CPU overhead", "DMA high-speed data transfer", "RDMA network performance", and "DMA embedded systems". The selected literature encompasses journal articles, conference proceedings, and technical documentation relevant to the research topic. The reviewed literature covers studies on I/O system mechanisms in general [11] through to the characterization of parallel I/O on large-scale HPC systems [12].

2.3. Literature Selection Criteria

The literature included in this review was selected based on the following criteria:

- Discusses DMA mechanisms in the context of data transfer and/or CPU load.
- Published in indexed journals or proceedings.
- Presents quantitative or qualitative data related to DMA performance (such as CPU utilization, throughput, or latency).
- Relevant to one of the DMA application domains under review, namely general computing, high-speed networking, hardware acceleration, or low-power embedded systems.

Literature that does not specifically address DMA performance aspects or only mentions DMA briefly in an irrelevant context was excluded from the review.

2.4. Analysis Method

The analysis was conducted in a descriptive-comparative manner on findings from the selected literature. Specifically, this review compares the performance data of the three data transfer methods most frequently discussed in the literature: Programmed I/O, Interrupt-driven I/O, and DMA Controller [13], [14]. The comparison parameters extracted from the literature include:

1. CPU Utilization: the percentage of CPU resources used during the data transfer process.
2. Transfer Throughput: the effective data transfer speed achieved by each method.
3. Transfer Latency: the time delay from the initiation to the completion of data transfer.

Quantitative data reported in the literature, such as test scenarios with 1 GB datasets [14], [15], were summarized and compared to identify patterns and consistency of findings across studies. The comparison results are presented in tables and figures in the following chapter to facilitate interpretation.

2.5. Analytical Framework

To provide structure to the discussion, findings from the literature were grouped into three categories of DMA application domains:

1. General computing: covering comparison of data transfer methods on conventional processor architectures.
2. High-speed networking: encompassing the application of RDMA, SmartNIC, and FPGA in the context of network data transfer.
3. Hardware acceleration and embedded systems: covering the application of DMA on GPUs, ML accelerators, and low-power IoT devices.

This framework enables cross-domain analysis to identify similarities and differences in DMA effectiveness across different application contexts. Figure 1 illustrates the analytical framework used in this review, encompassing the three DMA application domains along with the comparison parameters extracted from the literature.

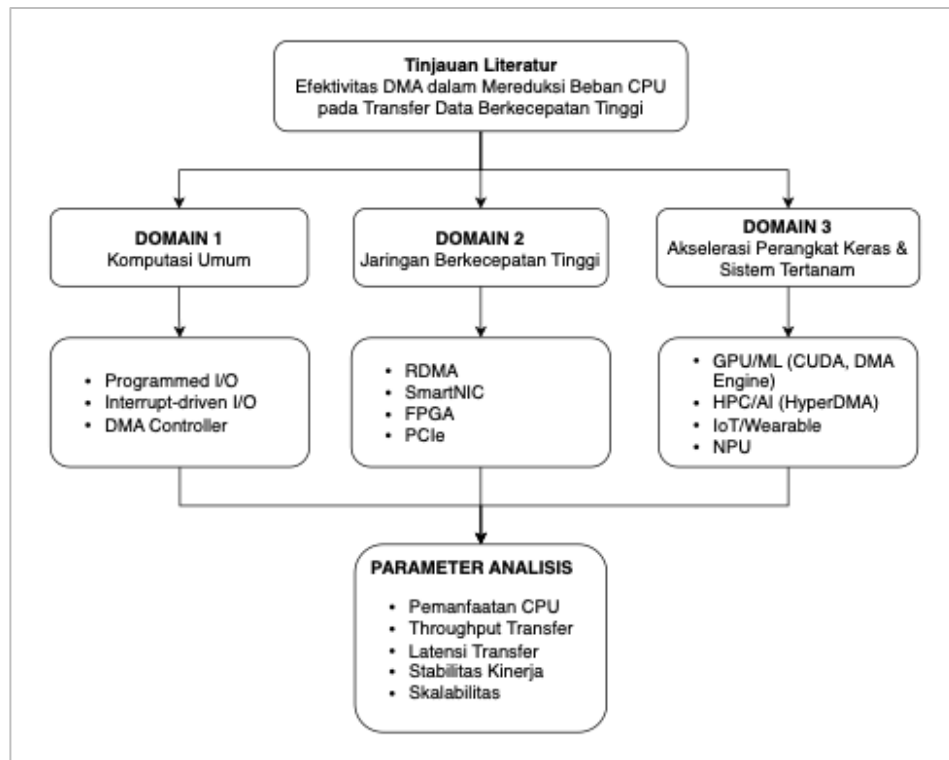


Figure 1. Analytical Framework Diagram

3. RESULTS AND DISCUSSION

3.1. Performance Comparison of Data Transfer Methods

Based on the synthesis of the reviewed literature, there are notable differences in CPU consumption among the three primary data transfer methods: Programmed I/O, Interrupt-driven I/O, and DMA Controller. Several studies report comparisons of these three methods using a 1 GB data transfer scenario with observed parameters including CPU utilization, transfer throughput, and data transfer latency [13], [14], [15]. Table 1 summarizes the performance comparison of the three data transfer methods based on parameters extracted from the literature.

Table 1. Performance Comparison of Three Data Transfer Methods

Parameter	Programmed I/O	Interrupt-driven I/O	DMA Controller
CPU Utilization	Very high (~85–98%)	High (~45–65%)	Low (~5–18%)
Throughput	Low	Moderate	High
Latency	High (~220–260 ms)	Moderate (~140–180 ms)	Low (~40–75 ms)
Stability (Variance)	Low	Moderate	High
Scalability on large data	Poor (drastic degradation)	Poor	Good (stable)

As shown in Table 1, the differences in CPU utilization among the three methods are highly significant, with the DMA Controller consistently demonstrating the lowest values across all parameters examined. For the Programmed I/O method, the literature consistently reports very high CPU utilization because the CPU must actively manage every transfer cycle. The Interrupt-driven I/O method shows improvement by freeing the CPU between interrupt cycles; however, the CPU remains involved in every data movement. In contrast, the DMA Controller is reported to significantly reduce CPU utilization by up to 80% in gigabit-speed transfer scenarios, as the transfer process is handled entirely by the DMA controller without continuous CPU intervention [16], [17], [18]. These findings are consistent with studies on DMA-based transfer acceleration on I/O platforms [19] and high-performance DMA engine architectures designed for optimal energy efficiency [20].

From the perspective of performance stability, the literature also shows that DMA exhibits the lowest performance variance among the three methods, indicating high consistency in repeated transfers [16], [17]. This is a significant advantage, particularly for applications requiring continuous data transfer reliability, such as video buffering, network transfers, and real-time sensor readings.

3.2. DMA Offloading Mechanism for CPU

The primary finding from various literature sources indicates that DMA is capable of reducing CPU load through an offloading mechanism [18]. In this mechanism, the CPU only plays a role during the initialization and finalization stages of the transfer, while the entire data movement process is handled by the DMA controller [21]. This concept aligns with computer architecture theory, which states that DMA is designed to eliminate the need for the CPU to perform repetitive load/store operations on every transfer cycle [22].

This offloading mechanism has direct implications for overall system efficiency. By freeing the CPU from data transfer tasks, computational resources can be allocated to processing other tasks simultaneously. This is highly relevant in high-throughput applications where data transfers occur massively and continuously [23].

The implementation of DMA controllers in embedded systems has been demonstrated through various designs, including AMBA-based architectures for embedded applications [24] and the use of DMA cache to separate I/O data from CPU data in on-chip storage [25]. A more advanced approach proposes a decoupled DMA mechanism that architecturally isolates CPU and I/O traffic through dual-data-port DRAM, thereby eliminating interference between the two [23]. At the multi-accelerator SoC scale, distributed DMA architectures such as XDMA are designed to support flexible data movement between heterogeneous components [26].

Figure 2 illustrates the comparison of data transfer timelines between the conventional method (without DMA) and the DMA-based method. In the conventional method, the CPU is fully engaged during the transfer process and cannot perform other tasks. Conversely, with the DMA offloading mechanism, the CPU is only active during the initialization and finalization phases, while the entire transfer process is handled independently by the DMA controller [18], [21].

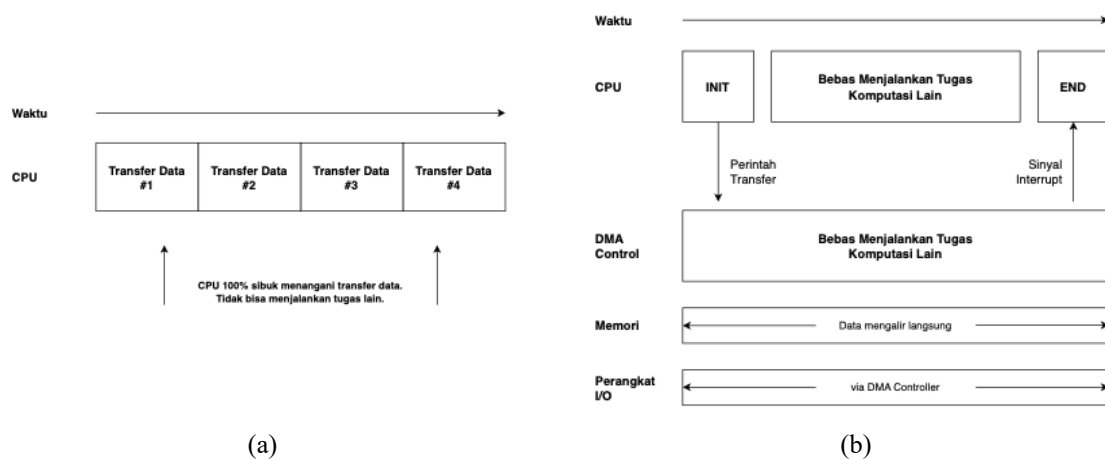


Figure 2. DMA Offloading Mechanism Diagram (a) Without DMA (b) With DMA

As illustrated in Figure 2, the parallelism between CPU activity and the DMA controller serves as the basis for significant CPU load reduction, particularly in large-scale data transfer scenarios discussed in the following subsection.

3.3. DMA Scalability in Large-Scale Data Transfer

DMA effectiveness is reported to increase as the size of the transferred data grows. In a 1 GB data transfer scenario, the advantages of DMA over conventional methods become very clear [27]. This is due to the relatively small DMA initialization overhead compared to the volume of data being transferred. In other words, the larger the data being sent, the less significant the overhead cost compared to the parallelism advantages offered by DMA [21].

This characteristic explains why Programmed I/O and Interrupt-driven I/O experience drastic performance degradation as data volume increases, while DMA continues to demonstrate stable performance [28]. These findings confirm that DMA is the most suitable transfer method for applications involving large-scale data movement.

In modern storage systems, DMA mechanisms have also been optimized for large-scale data transfer on NVMe SSD devices through a user-level DMA approach that minimizes software overhead [29]. A comprehensive study by Li et al. demonstrates that on-chip DMA optimization significantly accelerates data movement in non-volatile memory (NVM)-based storage systems [30].

In the context of persistent memory, a study by Li et al. shows that the DMA-assisted I/O mechanism, which adaptively combines the strengths of CPU and DMA, can achieve performance improvements of up to 10.4 times compared to CPU-only methods under heavy load conditions, while substantially reducing CPU consumption [31].

3.4. Impact of Bus and Memory Architecture on DMA Performance

Although DMA demonstrates consistent advantages, the literature also identifies that DMA performance is highly dependent on the system's bus and memory architecture [32]. In systems with low bus bandwidth, DMA may compete with the CPU for bus access (bus contention), which can reduce its effectiveness. However, in modern systems adopting high-bandwidth architectures, this issue becomes insignificant [32], [33].

The evolution of computer architecture using the latest generation PCIe and integrated memory controllers further strengthens DMA effectiveness in parallel transfers [33]. In the networking context, RDMA technology leverages this architecture to achieve direct data transfer between computer memories with very low latency [4], [5]. Similarly, in SmartNIC and FPGA infrastructure, DMA cores utilize high-speed PCIe paths to facilitate the transfer of large data volumes directly to host memory [6].

At the inter-core interconnect level, various approaches have been proposed to optimize DMA transfers, including PCIe-based multicore communication systems [34] and low-latency inter-core DMA mechanisms using hybrid packet-circuit switching networks [35]. Evaluation of interconnect architectures on streaming accelerators also demonstrates that interconnect design has a significant impact on transfer latency [36].

Looking ahead, next-generation interconnect technologies such as Compute Express Link (CXL) offer significant potential to extend DMA effectiveness. A comprehensive survey on CXL shows that this protocol enables more flexible memory management across devices [37], including the transformation of PCIe storage access through the CXL memory protocol [38]. Hybrid approaches combining CXL and Ethernet have also been explored to support the scalability of data-parallel applications [39].

3.5. DMA Application in Hardware Acceleration and Low-Power Systems

In the field of hardware acceleration for Machine Learning (ML), the literature reports that DMA engines in modern GPUs such as the AMD Instinct MI300X play a role in offloading communication burden from the main processing units. Consequently, disruptions to computation and memory access can be minimized when kernels run concurrently, thereby narrowing the gap between theoretical and actual performance for complex ML applications [7]. In CUDA programming, DMA-based streaming mechanisms are used to hide data transfer latency through PCIe by overlapping kernel execution and data transfer in parallel [8].

Further studies indicate that data transfer is a critical factor in overall GPU computing performance [40]. In the more specific context of ML computing, GPU DMA engine optimization enables computation and communication to run concurrently, significantly increasing training and inference throughput [41]. In heterogeneous computing architectures integrating CPUs, GPUs, and FPGAs, the role of DMA becomes increasingly central as the data transfer bridge between different processing units [42].

The Role of Direct Memory Access (DMA) in Reducing CPU Load on High-Speed Data Transfer (Parlika et al.)

More advanced DMA architectures such as HyperDMA have also been proposed to support multi-dimensional tensor transfers for HPC and AI workloads, with a minimum 39% improvement in matrix computation performance compared to systems without DMA [43].

On a different spectrum, DMA efficiency has also proven crucial in low-power devices. The use of DMA in microcontrollers allows sensor data to be transferred to storage media while the CPU remains in sleep mode, significantly reducing power consumption and extending device battery life [9]. These findings demonstrate that the benefits of DMA are not limited to performance enhancement alone but also extend to energy efficiency, an aspect of growing importance with the expanding IoT and wearable device ecosystem.

In the context of energy efficiency, the DMA double-buffering technique has been demonstrated to support continuous energy consumption measurement on IoT devices without burdening the CPU [44]. Furthermore, the role of DMA extends to newer domains such as Neural Processing Unit (NPU) accelerators, where DMA-based NPU virtualization enables multi-sensor AI systems to operate with low latency [45]. A comparison of inference performance between NPUs and GPUs shows that data transfer efficiency through DMA is one of the determining factors of NPU architecture superiority [46]. Table 2 presents a summary of DMA applications across various domains along with their associated technologies and primary benefits based on the reviewed literature.

Table 2. Summary of DMA Applications Across Various Domains

Application Domain	DMA Technology	Primary Benefit	References
General Computing	DMA Controller	CPU reduction up to 90%	[16], [18], [21]
HPC Networking	RDMA	Zero-copy, low latency	[4], [5]
SmartNIC/FPGA	PCIe DMA Core	Preventing PCIe bottleneck	[6], [16], [17]
GPU/ML	GPU DMA Engine	Kernel & transfer overlap	[7], [8], [41]
HPC/AI	HyperDMA	Multi-dimensional tensor transfer	[43]
IoT/Wearable	Microcontroller DMA	Power saving, CPU sleep mode	[9], [44]
NPU/AI	DMA Virtualization	Low latency, multi-sensor	[45], [46]
NVM Storage	On-chip DMA	Data movement acceleration	[30], [31]

4. CONCLUSION

Based on the literature review conducted, several conclusions can be drawn. First, DMA significantly reduces CPU load in high-speed data transfer. Through the offloading mechanism, DMA allows the CPU to be involved only during the initialization and finalization stages of the transfer, while the entire data movement process is handled by the DMA controller. The literature reports that this mechanism can reduce CPU utilization by up to 80% in gigabit-speed transfer scenarios [18], [21].

Second, DMA demonstrates increasingly greater advantages at higher data volumes. In a 1 GB data transfer scenario, DMA consistently outperforms Programmed I/O and Interrupt-driven I/O in terms of throughput, latency, and performance stability [27], [28]. The literature also reports increased data transfer speeds and reduced processor time compared to conventional methods [3], [47].

Third, DMA applications span diverse computing domains, including high-speed networking through RDMA and SmartNIC [4], [6], hardware acceleration for Machine Learning on modern GPUs [7], [8], and low-power embedded devices in the IoT ecosystem [9]. DMA performance depends on bus and memory architecture, with optimal effectiveness on modern systems adopting the latest generation PCIe and integrated memory controllers [32], [33].

This review also identifies several potential research directions for future development, including: (1) analysis of DMA impact on artificial intelligence accelerators such as TPUs and Neural Engines [45], [48]; (2) DMA optimization on chiplet-based processor architectures [37], [48]; (3) implementation of AI-based adaptive DMA for memory access pattern prediction [46], including programmable DMA controller architectures supporting multi-dimensional access patterns [49]; (4) development of secure DMA methods to address potential security vulnerabilities in direct memory access [50], [51]; and (5) integration of DMA with next-generation interconnect technologies such as PCIe Gen6, CXL, and NVLink [37], [38].

REFERENCES

- [1] A. Aljumah and M. A. Ahmed, "AMBA Based Advanced DMA Controller for SoC," *Int. J. Adv. Comput. Sci. Appl. IJACSA*, vol. 7, no. 3, Mar. 2016, doi: 10.14569/IJACSA.2016.070326.
- [2] S. K. Bhadrappa and V. B. Ravishankar, "Central Processing Unit Load Reduction Through Application Code Optimization and Memory Management," *Int. J. Reconfigurable Embed. Syst. IJRES*, vol. 14, no. 1, pp. 79–88, Mar. 2025, doi: 10.11591/ijres.v14.i1.pp79-88.

- [3] A. C. Fauzan, U. S. A. Baqi, A. A. Auladi, and A. Z. M. Sph, "Direct Memory Access untuk Menghitung Waktu Prosesor Intel Celeron N2840 dan AMD A8-7410 dalam Menangani Transfer Data," *ILKOMNIKA*, vol. 1, no. 1, pp. 1–6, Aug. 2019, doi: 10.28926/ilkomnika.v1i1.4.
- [4] A. Cilaro, "Evaluation of HPC Acceleration and Interconnect Technologies for High-Throughput Data Acquisition," *Sensors*, vol. 21, no. 22, p. 7759, Jan. 2021, doi: 10.3390/s21227759.
- [5] K. Taranov, F. Fischer, and T. Hoefler, "Efficient RDMA Communication Protocols," Dec. 20, 2022, *arXiv*: arXiv:2212.09134. doi: 10.48550/arXiv.2212.09134.
- [6] X. Wei, R. Cheng, Y. Yang, R. Chen, and H. Chen, "Characterizing Off-path SmartNIC for Accelerating Distributed Systems," presented at the 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23), 2023, pp. 987–1004. Accessed: Mar. 28, 2026. [Online]. Available: <https://www.usenix.org/conference/osdi23/presentation/wei-smartnic>
- [7] Y. Yuan *et al.*, "ORCA: A Network and Architecture Co-design for Offloading us-scale Datacenter Applications," Oct. 17, 2022, *arXiv*: arXiv:2203.08906. doi: 10.48550/arXiv.2203.08906.
- [8] P. Czarnul, "Investigation of Parallel Data Processing Using Hybrid High Performance CPU + GPU Systems and CUDA Streams," *Comput. Inform.*, vol. 39, no. 3, pp. 510–536, Dec. 2020, doi: 10.31577/cai_2020_3_510.
- [9] C. C. Dobrescu, I. González, D. Carneros-Prado, J. Fontecha, and C. Nugent, "Direct Memory Access-Based Data Storage for Long-Term Acquisition Using Wearables in an Energy-Efficient Manner," *Sensors*, vol. 24, no. 15, p. 4982, Jan. 2024, doi: 10.3390/s24154982.
- [10] A. (ITEC) Engelhart, "ITEC-OS Staff - An Analysis of DMA Interference Using Synthetic Load from an NVMe Device." Accessed: Mar. 28, 2026. [Online]. Available: https://os.itec.kit.edu/21_3192.php
- [11] M. R. Hidayatulloh, F. A.-H. S. Bahri, A. Muqtashida, R. Gunawan, and S. Azahra, "Studi Sistem Input/Output: Perangkat, Interface, dan Optimalisasi Kinerja Komputer," *J. Ris. Multidisiplin Edukasi*, vol. 2, no. 6, pp. 363–369, Jun. 2025, doi: 10.71282/jurmie.v2i6.451.
- [12] H. Ather, J. L. Bez, C. Wang, H. Childs, A. D. Malony, and S. Byna, "Parallel I/O Characterization and Optimization on Large-Scale HPC Systems: A 360-Degree Survey," Dec. 31, 2024, *arXiv*: arXiv:2501.00203. doi: 10.48550/arXiv.2501.00203.
- [13] A. Ruzhanskaia, P. Xu, D. Cock, and T. Roscoe, "Rethinking Programmed I/O for Fast Devices, Cheap Cores, and Coherent Interconnects," Apr. 24, 2025, *arXiv*: arXiv:2409.08141. doi: 10.48550/arXiv.2409.08141.
- [14] K. Salah, K. El-Badawi, and F. Haidari, "Performance Analysis and Comparison of Interrupt-Handling Schemes in Gigabit Networks," *Comput. Commun.*, vol. 30, no. 17, pp. 3425–3441, Nov. 2007, doi: 10.1016/j.comcom.2007.06.013.
- [15] E. Joelianto, F. Ramdhani, and E. M. Budi, "Analisis Pengaruh Waktu Latensi Terhadap Akurasi Sistem SCADA Bacaan Metering Listrik Waktu Nyata Melalui Jaringan Internet," *J. Rekayasa Elektr.*, vol. 16, no. 3, Dec. 2020, doi: 10.17529/jre.v16i3.16465.
- [16] W. M. Zabolotny, "Versatile DMA Engine for High-Energy Physics Data Acquisition Implemented with High-Level Synthesis," *Electronics*, vol. 12, no. 4, p. 883, Jan. 2023, doi: 10.3390/electronics12040883.
- [17] K. Cheng, W. Liu, Q. Shen, and S. Liao, "Design and Implementation of High-throughput PCIe with DMA Architecture between FPGA and PowerPC," Sep. 17, 2018, *arXiv*: arXiv:1809.07702. doi: 10.48550/arXiv.1809.07702.
- [18] C. K. Shaila, M. J. Arthur, and G. Manoj, "Optimizing IoT Applications with RTL-Based DMA Controller for Data Transfers," presented at the The 2025 International Conference on Advanced Research in Electronics and Communication Systems (ICARECS-2025), Atlantis Press, Jun. 2025, pp. 756–764. doi: 10.2991/978-94-6463-754-0_66.
- [19] S. Larsen, S. Larsen, and B. Lee, "Platform IO DMA Transaction Acceleration," in *CACHES. ACM*, 2011. Accessed: Mar. 28, 2026. [Online]. Available: <https://www.semanticscholar.org/paper/Platform-IO-DMA-Transaction-Acceleration-Larsen-Larsen/847955ec5e2777f771e9ad757638b28671fc5f25>
- [20] T. Benz *et al.*, "A High-Performance, Energy-Efficient Modular DMA Engine Architecture," *IEEE Trans. Comput.*, vol. 73, no. 1, pp. 263–277, Jan. 2024, doi: 10.1109/TC.2023.3329930.
- [21] C. Lu, H. Yang, and Q. Wu, "Design and Implementation of a Direct Memory Access ControllerBased on Microcontroller Unit," *J. Phys. Conf. Ser.*, vol. 2221, no. 1, p. 012016, May 2022, doi: 10.1088/1742-6596/2221/1/012016.
- [22] D. Tang, Y. Bao, Y. Chen, W. Hu, and M. Chen, "Exploiting the Produce-Consume Relationship in DMA to Improve I / O Performance," 2009. Accessed: Mar. 28, 2026. [Online]. Available: <https://www.semanticscholar.org/paper/Exploiting-the-Produce-Consume-Relationship-in-DMA-Tang-Bao/62ea2cab84e5099edbc5dfa283fa1aa230ce6d8b>
- [23] D. Lee, L. Subramanian, R. Ausavarungnirun, J. Choi, and O. Mutlu, "Decoupled Direct Memory Access: Isolating CPU and IO Traffic by Leveraging a Dual-Data-Port DRAM," in *2015 International Conference on Parallel Architecture and Compilation (PACT)*, Oct. 2015, pp. 174–187. doi: 10.1109/PACT.2015.51.
- [24] M. A. A. Ahmad Abdullah Aljumah, M. Gulam, "Design and Implementation of a Direct Memory Access Controller for Embedded Applications," *IJTech - Int. J. Technol.*, vol. 10, no. 2, pp. 309–319, 2019, doi: 10.14716/ijtech.v10i2.795.
- [25] D. Tang, Y. Bao, W. Hu, and M. Chen, "DMA cache: Using On-Chip Storage to Architecturally Separate I/O Data From CPU Data for Improving I/O Performance," in *HPCA - 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, Jan. 2010, pp. 1–12. doi: 10.1109/HPCA.2010.5416638.
- [26] F. Kong, Y. Deng, X. Yi, R. Antonio, and M. Verhelst, "XDMA: A Distributed, Extensible DMA Architecture for Layout-Flexible Data Movements in Heterogeneous Multi-Accelerator SoCs," presented at the 2025 IEEE 43rd International Conference on Computer Design (ICCD), IEEE Computer Society, Nov. 2025, pp. 690–693. doi: 10.1109/ICCD65941.2025.00104.
- [27] S. Saidi, P. Tendulkar, T. Lepley, and O. Maler, "Optimizing Explicit Data Transfers for Data Parallel Applications on The Cell Architecture," *ACM Trans Arch. Code Optim.*, vol. 8, no. 4, p. 37:1–37:20, Jan. 2012, doi: 10.1145/2086696.2086716.
- [28] A. Mera, Y. H. Chen, R. Sun, E. Kirda, and L. Lu, "D-Box: DMA-enabled Compartmentalization for Embedded Applications," in *Proceedings 2022 Network and Distributed System Security Symposium*, San Diego, CA, USA: Internet Society, 2022. doi: 10.14722/ndss.2022.24053.
- [29] J. Zhu, L. Wang, L. Xiao, and G. Qin, "uDMA: An Efficient User-Level DMA for NVMe SSDs," *Appl. Sci.*, vol. 13, no. 2, p. 960, Jan. 2023, doi: 10.3390/app13020960.
- [30] J. Li *et al.*, "Fastmove: A Comprehensive Study of On-Chip DMA and its Demonstration for Accelerating Data Movement in NVM-based Storage Systems," *ACM Trans Storage*, vol. 20, no. 3, p. 19:1–19:30, Jun. 2024, doi: 10.1145/3656477.
- [31] D. Li, W. Zhang, M. Dong, and K. Ota, "DMA-Assisted I/O for Persistent Memory," *IEEE Trans. Parallel Distrib. Syst.*, vol. 35, no. 5, pp. 829–843, May 2024, doi: 10.1109/TPDS.2024.3373003.
- [32] D. Syrivelis, A. Reale, K. Katrinis, and C. Pinto, "A Software-defined SoC Memory Bus Bridge Architecture for Disaggregated Computing," in *Proceedings of the 3rd International Workshop on Advanced Interconnect Solutions and Technologies for Emerging*

- Computing Systems*, in AISTECS '18. New York, NY, USA: Association for Computing Machinery, Jan. 2018, pp. 1–4. doi: 10.1145/3186608.3186611.
- [33] J.-H. Chae, “High-Bandwidth and Energy-Efficient Memory Interfaces for the Data-Centric Era: Recent Advances, Design Challenges, and Future Prospects,” *IEEE Open J. Solid-State Circuits Soc.*, vol. 4, pp. 252–264, 2024, doi: 10.1109/OJSSCS.2024.3458900.
 - [34] S. Otani, H. Kondo, I. Nonomura, T. Hanawa, S. Miura, and T. Boku, “Peach: A Multicore Communication System on Chip with PCI Express,” *IEEE Micro*, vol. 31, no. 6, pp. 39–50, Nov. 2011, doi: 10.1109/MM.2011.93.
 - [35] A. K. Abousamra, R. G. Melhem, and A. K. Jones, “Déjà Vu Switching for Multiplane NoCs,” in *Proceedings of the 2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, in NOCS '12. USA: IEEE Computer Society, May 2012, pp. 11–18. doi: 10.1109/NOCS.2012.9.
 - [36] C.-T. Axinte, A. Stan, and V.-I. Manta, “Embedded Streaming Hardware Accelerators Interconnect Architectures and Latency Evaluation,” *Electronics*, vol. 14, no. 8, p. 1513, Jan. 2025, doi: 10.3390/electronics14081513.
 - [37] C. Chen, X. Zhao, G. Cheng, Y. Xu, S. Deng, and J. Yin, “Next-Gen Computing Systems with Compute Express Link: a Comprehensive Survey,” Feb. 20, 2025, *arXiv: arXiv:2412.20249*. doi: 10.48550/arXiv.2412.20249.
 - [38] M. Kwon *et al.*, “From Block to Byte: Transforming PCIe Solid-State Devices With Compute Express Link Memory Protocol and Instruction Annotation,” *IEEE Micro*, vol. 45, no. 06, pp. 46–55, Nov. 2025, doi: 10.1109/MM.2025.3581448.
 - [39] X. Zhang, K. Liu, Y. Chang, K. Zhang, and M. Chen, “DFabric: Scaling Out Data Parallel Applications with CXL-Ethernet Hybrid Interconnects,” Oct. 30, 2024. doi: 10.5555/3768039.3768113.
 - [40] Y. Fujii, T. Azumi, N. Nishio, S. Kato, and M. Edahiro, “Data Transfer Matters for GPU Computing,” in *2013 International Conference on Parallel and Distributed Systems*, Dec. 2013, pp. 275–282. doi: 10.1109/ICPADS.2013.47.
 - [41] A. Agrawal, S. Aga, S. Pati, and M. Islam, “ConCCL: Optimizing ML Concurrent Computation and Communication with GPU DMA Engines,” presented at the 2025 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), IEEE Computer Society, May 2025, pp. 1–11. doi: 10.1109/ISPASS64960.2025.00018.
 - [42] Elavarasan, “The Future of Heterogeneous Computing: Integrating CPUs, GPUs, and FPGAs for High-Performance Applications,” *Int. J. Artif. Intell. Data Sci. Mach. Learn.*, vol. 5, no. 1, pp. 20–31, Mar. 2024, doi: 10.63282/3050-9262.IJAIDSML-V5I1P103.
 - [43] M. Peng, H. Chen, Y. Zhang, and S. Liu, “HyperDMA: Enhancing High-Performance Computing and AI Workflows with Advanced Data Transfer Capabilities,” in *2024 9th International Conference on Integrated Circuits and Microsystems (ICICM)*, Oct. 2024, pp. 636–644. doi: 10.1109/ICICM63644.2024.10814280.
 - [44] D. Vaquerizo-Hdez, P. Muñoz, D. F. Barrero, and M. D. R-Moreno, “Continuous Energy Consumption Measure Approach Using a DMA Double-Buffering Technique,” *EURASIP J. Wirel. Commun. Netw.*, vol. 2021, no. 1, p. 172, Aug. 2021, doi: 10.1186/s13638-021-02043-w.
 - [45] J.-H. Jean and D.-S. Kim, “Hardware-Assisted Low-Latency NPU Virtualization Method for Multi-Sensor AI Systems,” *Sensors*, vol. 24, no. 24, p. 8012, Jan. 2024, doi: 10.3390/s24248012.
 - [46] Y. Hong and D. Kim, “Performance and Efficiency Gains of NPU-Based Servers over GPUs for AI Model Inference,” *Systems*, vol. 13, no. 9, p. 797, Sep. 2025, doi: 10.3390/systems13090797.
 - [47] J. Wang, G. Lv, Z. Liu, and X. Yang, “Programmable Deterministic Zero-Copy DMA Mechanism for FPGA Accelerator,” *Appl. Sci.*, vol. 12, no. 19, p. 9581, Jan. 2022, doi: 10.3390/app12199581.
 - [48] S. Gener *et al.*, “RIMMS: Runtime Integrated Memory Management System for Heterogeneous Computing,” Jul. 28, 2025, *arXiv: arXiv:2507.20514*. doi: 10.48550/arXiv.2507.20514.
 - [49] L. Zulberti, A. Monorchio, M. Monopoli, G. Mystkowska, P. Nannipieri, and L. Fanucci, “SmartDMA: Adaptable Memory Access Controller for CGRA-based Processing Systems,” in *2024 27th Euromicro Conference on Digital System Design (DSD)*, Aug. 2024, pp. 306–313. doi: 10.1109/DSD64264.2024.00048.
 - [50] S. Pati, M. Islam, S. Aga, and M. A. Ibrahim, “DMA Collectives for Efficient ML Communication Offloads,” Nov. 10, 2025, *arXiv: arXiv:2511.06605*. doi: 10.48550/arXiv.2511.06605.
 - [51] B. Shan, M. Araya-Polo, and B. Chapman, “DiOMP-Offloading: Toward Portable Distributed Heterogeneous OpenMP,” in *Proceedings of the SC '25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, in SC Workshops '25. New York, NY, USA: Association for Computing Machinery, Nov. 2025, pp. 1289–1301. doi: 10.1145/3731599.3767505.