Information systems. Computer sciences. Issues of information security

Информационные системы. Информатика. Проблемы информационной безопасности

UDC 004.2
https://doi.org/10.32362/2500-316X-2026-14-2-29-41
EDN XHLRAX

RESEARCH ARTICLE

Heterogeneous computing systems with hardware acceleration of massively parallel stream processing design

Andrey S. Zuev [@], Peter N. Sovietov, Ilya E. Tarasov

MIREA – Russian Technological University, Moscow, 119454 Russia
[@] Corresponding author, e-mail: zuev_a@mirea.ru

• **Submitted:** 09.06.2025 • **Revised:** 23.09.2025 • **Accepted:** 05.02.2026

Abstract

Objectives. The growing demand for higher computational performance and energy efficiency has motivated the increasing adoption of specialized heterogeneous computing systems incorporating hardware accelerators with massive parallelism. This paper aims to develop a methodology for the analysis and evaluation of hardware accelerator implementation strategies for large-scale parallel stream data processing which systematically captures all major directions of performance improvement.

Methods. The study employs established techniques of digital system design and modeling.

Results. A comparative evaluation method is introduced to assess the efficiency of heterogeneous computing architectures based on massively parallel hardware accelerators composed of independently programmable nodes. A computational acceleration ratio is defined which consolidates three key dimensions of accelerator improvement: algorithmic support and microarchitecture; design automation tools; and fabrication technologies (lithography). Furthermore, the study proposes an optimization-based methodology for the systematic analysis and evaluation of the alternatives for hardware accelerator implementation.

Conclusions. The expressions derived herein for calculating the computational acceleration ratio and the aggregate throughput of hardware accelerators account for both multichannel and block-based massively parallel data stream processing. In contrast to conventional architectural exploration approaches, the evaluation method proposed herein enables hardware accelerator design alternatives to be assessed at the earliest stages of the design cycle. This incorporates variations in algorithmic versions and implementation strategies which influence hardware architecture optimization. The proposed methodology for analyzing and evaluating implementation options for hardware accelerators can be used to develop technical specifications for their manufacture, design them according to specified requirements, and justify configuration decisions. It can also support research and development assignments to achieve target characteristics for certain domain-specific tasks of massively parallel stream data processing and CAD capabilities.

Keywords: processor, hardware accelerator, coprocessor, special-purpose processor, architecture, compiler

For citation: Zuev A.S., Sovietov P.N., Tarasov I.E. Heterogeneous computing systems with hardware acceleration of massively parallel stream processing design. *Russian Technological Journal.* 2026;14(2):29–41. https://doi.org/10.32362/2500-316X-2026-14-2-29-41, https://www.elibrary.ru/XHLRAX

Financial disclosure: The authors have no financial or proprietary interest in any material or method mentioned.

The authors declare no conflicts of interest.

НАУЧНАЯ СТАТЬЯ

О проектировании гетерогенных вычислительных систем с аппаратным ускорением массово-параллельной потоковой обработки данных

А.С. Зуев [@], П.Н. Советов, И.Е. Тарасов

МИРЭА – Российский технологический университет, Москва, 119454 Россия
[@] Автор для переписки, e-mail: zuev_a@mirea.ru

• **Поступила:** 09.06.2025 • **Доработана:** 23.09.2025 • **Принята к опубликованию:** 05.02.2026

Резюме

Цели. Необходимость ускорения вычислений и достижения высоких показателей энергоэффективности приводит к расширению сфер использования специализированных гетерогенных вычислительных систем, имеющих в своем составе аппаратные ускорители с массовым параллелизмом вычислений. Целью настоящей работы является создание методики анализа и оценки вариантов реализации аппаратных ускорителей для задач массово-параллельной потоковой обработки данных, отражающей все направления совершенствования характеристик применяемых аппаратных ускорителей.

Методы. Использованы методы проектирования и моделирования цифровых систем.

Результаты. Предложен метод сравнительной оценки эффективности архитектуры гетерогенной вычислительной системы на основе аппаратных ускорителей с массовым параллелизмом вычислений, выполняемых независимо программируемыми узлами. Введен коэффициент ускорения вычислений, объединяющий три направления совершенствования характеристик применяемых аппаратных ускорителей – математическое обеспечение и микроархитектура, инструментарий проектирования, технологии изготовления (литография). Предложена основанная на решении оптимизационной задачи методика анализа и оценки вариантов реализации применяемых аппаратных ускорителей.

Выводы. Полученные авторами формулы расчета коэффициента ускорения вычислений и пропускной способности совокупности аппаратных ускорителей учитывают многоканальную и поблочную массово-параллельную обработку потоков данных. В отличие от известных подходов к поиску архитектурных решений, предлагаемая оценка вариантов реализации аппаратных ускорителей может быть проведена на самых ранних этапах проектирования с учетом версий алгоритма и альтернатив их реализации, влияющих на оптимизацию аппаратной архитектуры. Предложенная методика анализа и оценки вариантов реализации аппаратных ускорителей может применяться при разработке технических заданий на их изготовление, при их проектировании в соответствии с заданными требованиями, для обоснования решений относительно их конфигурации, а также при составлении заданий на научно-исследовательские и опытно-конструкторские работы с целью достижения целевых значений характеристик для конкретных задач массово-параллельной потоковой обработки данных и функциональных возможностей систем автоматизированного проектирования.

Ключевые слова: процессор, аппаратный ускоритель, сопроцессор, спецпроцессор, архитектура, компилятор

Для цитирования: Зуев А.С., Советов П.Н., Тарасов И.Е. О проектировании гетерогенных вычислительных систем с аппаратным ускорением массово-параллельной потоковой обработки данных. *Russian Technological Journal*. 2026;14(2):29–41. https://doi.org/10.32362/2500-316X-2026-14-2-29-41, https://www.elibrary.ru/XHLRAX

Прозрачность финансовой деятельности: Авторы не имеют финансовой заинтересованности в представленных материалах или методах.

Авторы заявляют об отсутствии конфликта интересов.

INTRODUCTION

The design of HPC systems is currently based on two key areas: parallel architectures and specializations of individual computing nodes. One common example is the combination of general-purpose processors and graphics accelerators. Graphics accelerators are chips comprising a large number of specialized computing nodes which perform the transformations characteristic of 3D imaging. Similar architectural solutions combining universal and specialized computing subsystems can also be applied to tasks in artificial intelligence, network security, and big data analysis.

A set of interconnected problems must be resolved when developing computing systems with a heterogeneous architecture. These include:

- developing a program model and prototypes of specialized compilers for the selected domain-specific task;
- developing the architecture and basic components of computing devices at the register transfer level;
- integrating hardware components within the chosen technological framework. This includes both programmable logic devices (PLDs) and the more recently developed very-large-scale integration (VLSI) circuits.

A key objective is complex optimization based on established criteria. This determines the quality indicators of the computing system. Depending on the purpose of the system, optimization can be carried out in absolute terms (e.g., power consumption, cost, and performance) or in terms of derived indicators (e.g., specific power consumption or cost per performance unit). Due to the wide range of characteristics determined by the technological processes in the microelectronics industry, both the program model and the architectural solutions need to be optimized. Circuit engineering approaches also need to be aligned with the technological capabilities. Therefore, the hardware and software co-design approach can be extended to encompass joint optimization at the device component design technology level.

Currently, there are a number of general provisions which describe the situation and trends in microelectronics. Moore's law indicates the techno-economic side of the process of increasing computing performance. However, the law of scaling transistor sizes formulated by Dennard [1] in the early stages of microelectronics development explained this growth as an extensive process of increasing the density of components on a chip and reducing the signal propagation delay. Between 1980 and 2000, the factors led to a significant increase in processor clock speeds. During this period the transition from 4 MHz in the early 1980s to 4 GHz in the Intel Pentium 4 was observed for mainstream desktop processors.

At the same time, as new generations of technological processes were introduced in production, the effect of saturation was observed for many technical characteristics of digital chips. Table 1 shows a comparative analysis of some technological process generation parameters.

Table 1. Comparative characteristics of some technological process parameters

Year	2013	2015	2017	2019	2021
Notation	16/14	10	7	5	3
Half pitch, nm	40	32	25	20	16
FinFET[1] transistor width, nm	7.6	7.2	6.8	6.4	6.1
Number of 4I logic valves per square mm, million	4	6.4	10.1	16.1	25.5
Transistor gate length, nm	20	17	14	12	10

As can be seen from the above, transistor size reduction has stopped. Instead, an increase in the density of components per unit area of the semiconductor chip is the main factor in improving the characteristics of the chips produced. Consequently, transistor enhancement may progress beyond the constraints of Dennard's scaling law. This may be driven by the adoption of novel materials or modifications to the gate design,

[1] A FinFET (fin field-effect transistor) is a type of MOSFET (metal–oxide–semiconductor field-effect transistor) with a three-dimensional structure.

such as FinFET [2], GAAFET[2] [3], and MBCFET[3] [4]. In this respect, the development of computing devices has undergone a number of changes which reflect the characteristics of new generations of technologies for producing integrated circuits.

The study by Patterson and Hennessy [5] provides an overview of the main historical trends in processor design. Notably, the transition to highly integrated chips occurred during the period between 1980 and 1990. Pipeline depth increased from simple processor cores to pipelined architectures such as RISC[4] and up to technological standards of 90–65 nm. The end of Dennard scaling, coupled with an increase in the number of components on a single chip, enabled the shift towards multi-core processor devices. This allowed for an increase in overall processor performance, not through an increase in clock frequency, but through the parallel operation of multiple cores.

At the same time, Amdahl's law [6] limits the increase in computing performance in parallel architecture by taking into account the dependence on data in the executed algorithm. At the present time, the constraints imposed by Amdahl's law on numerous applications (especially desktops and servers) are not of great importance. This is because the tasks performed independently are not dependent on data, meaning they can make the most of the available processor cores.

The need to speed up calculations within a single task requires the use of appropriate architectural solutions, not only to increase the number of parallel computing nodes. A key element in enhancing the performance of computing nodes is the specialization of these nodes. This can be broken down into two main aspects: the specialization of their data path and memory subsystem; and the use of a system with a heterogeneous architecture. This system comprises a combination of general-purpose processors and specialized nodes which accelerate the calculation of specific algorithms or subclasses of algorithms for common, frequently solved domain-specific tasks.

The technique outlined in the paper enables the design of specialized computing devices for use as part of software and hardware complexes for a range of purposes. A key feature of this technique is its comprehensive consideration of design levels: from the system model to the topological implementation, using feedback from lower-level designs. This allows the characteristics of the computing device to be optimized according to established criteria. This paper also introduces mathematical criteria for accelerator efficiency based on a newly defined calculation acceleration ratio compared to a general-purpose processor. The problem of optimizing the architecture of a hardware accelerator is formulated as a research task within the set of architectural options.

The formulae for calculating the acceleration factor and throughput introduced in this paper differ from the analytical models commonly used, such as Roofline [7] and LogCA [8]. This is because they take into account the massively parallel processing of multichannel and block-by-block data streams.

Unlike existing methods of searching for architectural solutions [9], the proposed evaluation of options for implementing hardware accelerators can be performed at the initial design stage. This approach considers both the specifics of massively parallel data stream processing and the ability to select various options for the accelerated algorithm and its implementations, with the optimized hardware architecture.

Heterogeneous computing systems based on specialized hardware accelerators with massively parallel computing capabilities

Figure 1 shows the authors' interpretation of the study by Patterson and Hennessy [5] in terms of trends in processor system architecture.

This illustration offers an architectural solution for the domain-specific architecture direction proposed by Patterson and Hennessy. The specialization of a computing system can formally be achieved in a variety of ways. This paper proposes and considers a solution which preserves a general-purpose processor (including a multi-core version) and adds specialized accelerators working either independently or in parallel with the system.

The diagram in Fig. 2 illustrates the roles of general-purpose processors and accelerators. The axes represent the complexity of one iteration of calculations in a cyclic algorithm and the number of iterations required to solve the problem.

As shown in Fig. 2, there are four primary zones, each involving a blend of basic and advanced iterations. These iterations are characterized by data dependencies which complicate the parallelization of calculations. The figure also provides a visual representation of the numbers involved, highlighting their small and large values. It is not difficult to implement a simple case in the form of a simple algorithm with a small number of iterations (a special case is a linear algorithm with a single iteration) on a general-purpose processor. Therefore, creating a specialized accelerator for such a case is impractical. As the algorithm becomes more complex,

[2] A GAAFET (gate-all-around field-effect transistor) is a type of transistor where the channels are surrounded by gates on all four sides.

[3] MBCFET is a multi-bridge channel field-effect transistor.

[4] RISC (reduced instruction set computer) is a computer architecture that has a reduced instruction set.

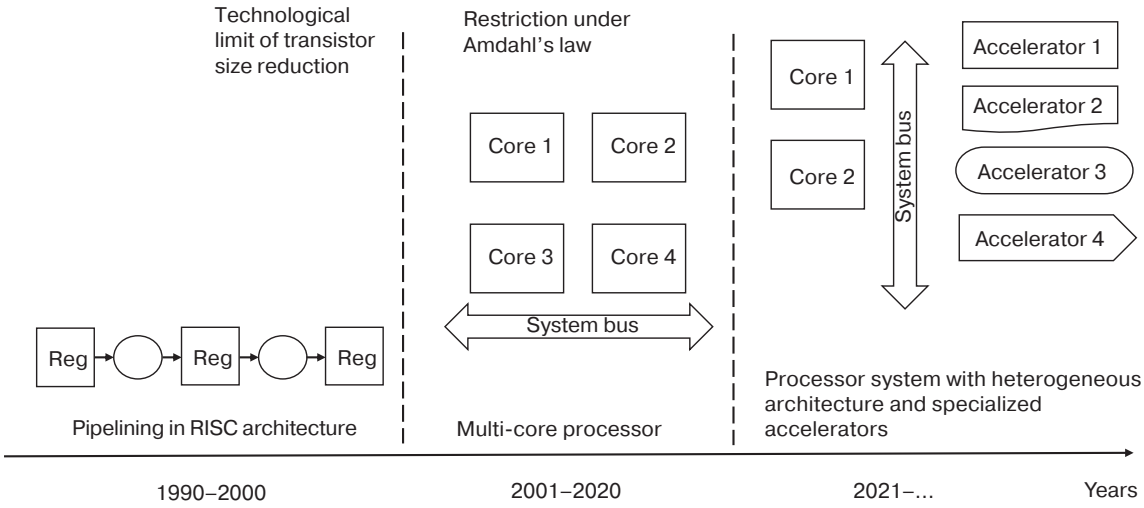

Fig. 1. Trends in processor architecture. Reg stands for register

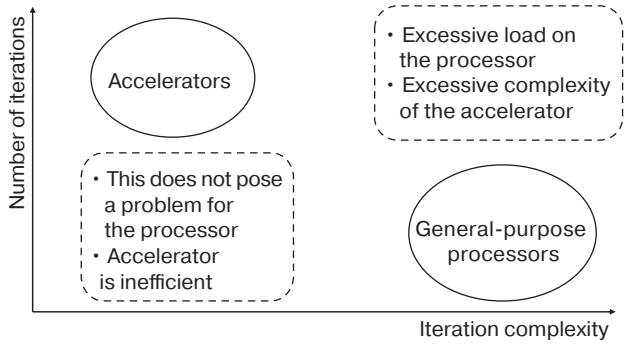

Fig. 2. Diagram showing the ratio of the complexity of iterations and their number, with the preferred computing device architectures

the device implementing it must have a well-developed set of functional nodes and enough memory to handle a long sequence of instructions. The implementation of a key computing subsystem such as an accelerator in this case can be a difficult task. Complex algorithms often require correction during system operation (e.g., updating the device's operating protocols). Conversely, a small number of iterations of such an algorithm do not create a significant computing load on the general-purpose processor.

An important issue is how to integrate the accelerator into the computing system. The increased integration of digital chips has led to the emergence of the interface wall effect [10], whereby there is an initial increase in computing performance compared to that of external interfaces (input/output (I/O) subsystems). In practice, placing in the accelerator numerous simple computing nodes which require intensive data exchange with a general-purpose processor will overload the interfaces used for data exchange. They will also reduce the accelerator's final computing performance due to an inability to provide the necessary input data stream. One example is the evolution of PLD characteristics with common architectures, such as FPGA[5], APSoC[6], and ACAP[7]. These are currently representative of highly integrated digital systems which combine subsystems for various purposes. Table 2 shows the comparative performance characteristics of the computing and interface subsystems of some PLD families manufactured by AMD/Xilinx (USA).

Table 2. Comparative performance characteristics of the computing and interface subsystems of certain AMD PLD families

PLD series	Series 7	UltraScale+	Versal
Process standard, nm	28	16	7
Logical cells, thou.	2000	8900	7350
Digital signal processing blocks	3600	12288	14352
Total computation performance, trln ops/s with int8 data	15	38.3	100
Total data transfer performance, Tb/s	2	4.1	17.6
Ratio of computation to data transfer performance	60	74	45

[5] A field-programmable gate array is a user-programmable gate array.

[6] All programmable system-on-chip is a type of programmable integrated circuit.

[7] Adaptive compute acceleration platform.

As shown in Table 2, the ratio of computation to data transfer performance generally remains between 50 and 70. This means that, for an effective computing load, at least 50–70 processing operations are performed on the data transmitted. This range of values is conditional and does not take into account the practical aspects of implementing operations. However, it should be noted that an excessively simple accelerator will reduce system efficiency, since each operation will require intensive data exchange with other system components. Figure 3 illustrates a negative use case for a compute accelerator with limited interface throughput.

Thus, in terms of architecture, it is important to ensure an appropriate level of computing performance. This guarantees the efficient use of accelerator resources at a given throughput level for its interfaces. The options for such scenarios are shown in Fig. 4.

The options provided ensure that the accelerator performs multiple operations on the transmitted data. Option 1 is used when implementing a relatively complex stand-alone algorithm which iterates calculations without the need to transmit a constant stream of data. Option 2 is used in parallel data processing where numerous operations are implemented in series-connected stages of the pipeline computer. The paper focuses on Option 2,

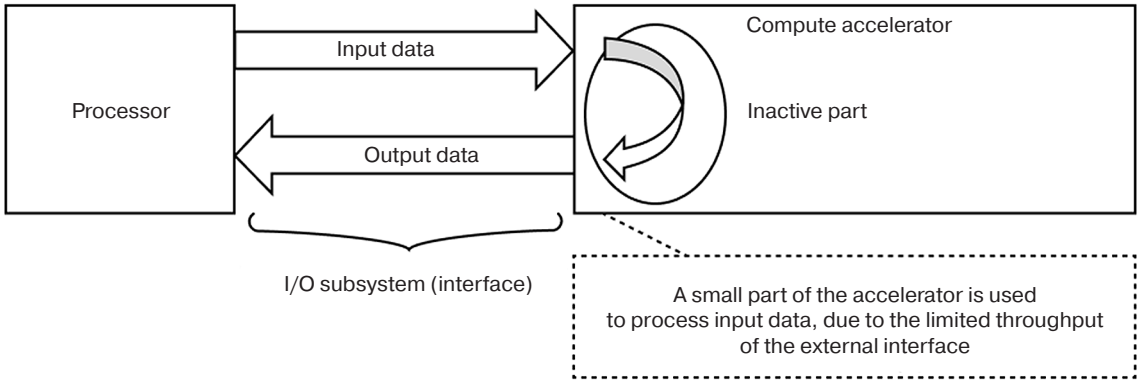

Fig. 3. Negative use case for a compute accelerator with limited I/O subsystem (interface) throughput

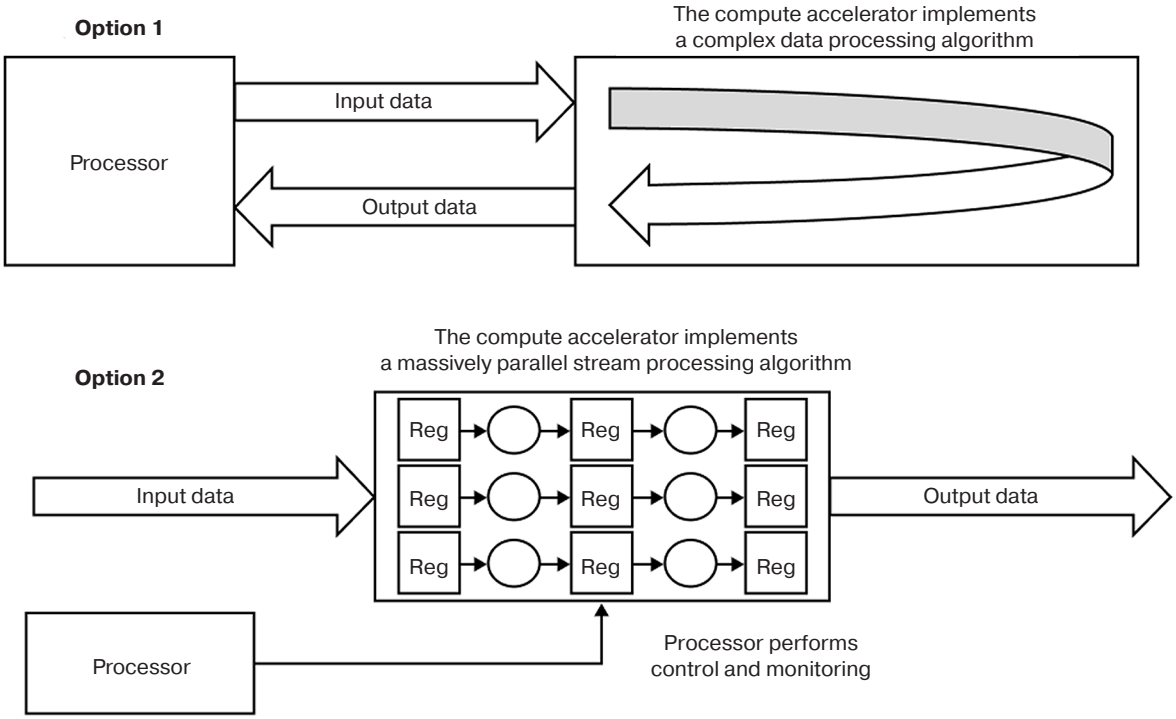

Fig. 4. Use cases (options) for a compute accelerator that uses limited interface throughput effectively (the compute pipeline is used)

shown in Fig. 4, which is a heterogeneous computing system with hardware acceleration of massively parallel data streaming.

Tasks and algorithms for massively parallel streaming data processing

Streaming tasks arise in a variety of fields, including digital signal processing, data compression, cryptography, real-time analytics, and network traffic analysis.

A streaming process involves processing each element of a potentially endless stream of data, denoted as x_1, x_2, …, x_t, …, where x_t represents the element available at time t for processing. The elements are all of the same data type and may be individual bits, bytes, or more complex structures.

Streaming algorithms are characterized by their single-pass execution. They cannot revisit elements which have already been viewed or processed. Processing takes place within a limited local memory environment with a space complexity of $O(1)$, regardless of the number of streaming elements.

Key features of streaming processing systems include a data throughput of $R = N/t_N$, where N represents the number of elements and t_N represents the total processing time. Here, $R \le R_{max}$, where R_{max} is the maximum permissible throughput which may be limited by the capabilities of the I/O subsystem. During real-time operations, the delay per element should also be limited to $L \le L_{max}$.

Hardware acceleration enables the use of parallelism at different levels in data processing. Specifically, instruction-level parallelism depends primarily on the computational intensity of an element in a stream, i.e., the dominance of computational operations over memory operations. Task parallelism, also known as channel parallelism, involves the simultaneous processing of multiple channels.

Often, the streaming algorithm must be adapted to accommodate the specific hardware architecture. For example, if the tables used by the algorithm exceed the accelerator's local memory capacity, additional calculations may be required.

The literature describes hardware accelerators for processing stream-based data in various fields and for specific tasks within domains. Specifically, accelerators have been developed which can efficiently determine whether an item belongs to a set, such as Bloom filters [11], as well as accelerators which can estimate the number of distinct items in a multiset, such as HyperLogLog [12]. Hardware accelerators have also been developed to optimize the performance of various string-processing algorithms, such as calculating the edit distance [13], searching for regular expressions [14], and JSON[8] parsing [15]. Topics of data compression acceleration and decompression processes are discussed in [16], while [17] describes a hardware accelerator that combines data decompression and JSON parsing.

A variety of database algorithms represent a significant area for the implementation of hardware accelerators. The acceleration of relational database operations through hardware is discussed in [18]. These operations include sampling, mapping, joining, collating, and various aggregate functions. The implementation of hardware acceleration for graph database pattern matching is proposed in [19]. As outlined in [20], streaming analytics is a promising area for hardware acceleration. The study describes the main algorithms in this field which can be executed on hardware. Furthermore, approaches to streaming processing have been developed for various machine learning algorithms, leading to the creation of hardware accelerators for gradient boosting [21] and clustering tasks [22].

Comparative assessment of the efficiency of data streaming processing computing systems using a general-purpose processor or hardware acceleration

The effectiveness of the accelerator can be quantitatively assessed by comparing T_{soft} (task execution time using only a general-purpose processor) and T_{hard} (task execution time using the accelerator), i.e.,

$$K = \frac{T_{soft}}{T_{hard}}.$$

When estimating program execution time, the following formula, detailed in [23], may be useful:

$$\text{time} = \frac{\text{instructions}}{\text{program}} \cdot \frac{\text{cycles}}{\text{instruction}} \cdot \frac{T_{period}}{\text{cycle}}.$$

In this equation, the terms represented as fractions correspond to the characteristics of the compiler, microarchitecture, and workflow:

1. The number of instructions executed per iteration of the program is dependent on the ability of the compiler to generate machine code that is optimized for improved performance.
2. The number of cycles required to execute an instruction is determined by the processor's microarchitecture.
3. The duration of one clock period, T_{period}, is primarily affected by the method used to manufacture the processor.

[8] JavaScript object notation is a text-based data exchange format based on JavaScript syntax.

Thus, the overall program execution time can be improved through various methods which can be combined to design specialized accelerator systems.

When designing an accelerator connected to a processor using one of several interfaces, the configuration time, T_{conf}, for sending input problem parameters and receiving solution results must also be considered.

Assuming that the same task requires b iterations, the ratio of the execution time of a general-purpose processor to that of an accelerator can be expressed as follows:

$$K = \frac{bT_{soft}}{T_{conf} + bT_{hard}}.$$

In this form, the equation does not account for several computing nodes in the accelerator working in parallel. The number of instances of the data processing algorithm performed simultaneously in the accelerator will be referred to as "stream processing nodes." For example, when processing a data stream to search for regular expressions, the number of data stream processing nodes corresponds to the number of regular expressions for which analysis of each data block is performed simultaneously and in parallel.

Let t_{hard} be the execution time of the algorithm using a single hardware accelerator node. If p denotes the number of parallel nodes involved in solving this problem, then the formula takes the following form:

$$K = \frac{bT_{soft}}{T_{conf} + bt_{hard}/p}.$$

Due to stream processing, the accelerator is configured for long periods of operation during which a large (potentially infinite) number of data blocks of the same type are processed, so the T_{conf} value can be disregarded. The calculation acceleration tends towards:

$$K = \frac{pT_{soft}}{t_{hard}} = \frac{pn_{soft}f_{hard}}{n_{hard}f_{soft}}, \quad (1)$$

wherein n_{hard} and n_{soft} are the number of clock cycles required by the accelerator and processor, respectively, to solve the problem, and f_{hard} and f_{soft} are their corresponding clock frequencies.

Equation (1) shows that the hardware acceleration of the entire computing system is based on the following two factors:

- an increase in the number of parallel nodes;
- improving the efficiency of a single node by increasing its clock frequency and reducing the number of cycles required to perform a task.

The methodology for developing specialized processor cores has been previously discussed in [24]. When developing specialized electronic component bases (ECBs), a lack of access to advanced technological processes and the need to rapidly adapt newly developed designs to existing technology can make achieving a high operating frequency for the accelerator challenging. Therefore, it may be difficult to achieve a high ratio between the operating frequency of the hardware accelerator and that of commercially available general-purpose processors (in practice, this ratio may be less than 1). It is thus essential to focus on reducing the number of clock cycles required for the hardware accelerator to process a single data block. This is equivalent to reducing the n_{hard} parameter while maintaining the operating frequency f_{hard} rather high (in general, without considering power consumption factors). This can be accomplished using specialized parallel algorithms and stages of the optimizing compiler while taking into account the hardware specifications of the accelerator.

The proposed calculation acceleration ratio (1) combines three areas for improving the characteristics of the hardware accelerators of heterogeneous, massively parallel data processing systems: mathematical software and accelerator microarchitecture (p and n_{hard}); design tools, such as computer-aided design systems (CAD); and ECB manufacturing technologies (lithography) (p and f_{hard}).

Taking into account (1), for an accelerator with p parallel stream processing nodes, the processing time $T_{hard}(n)$ of n data blocks can be defined as follows:

$$T_{hard}(n) = T_{conf} + n \cdot \max\left(T_{block}, \frac{T_{soft}}{K}\right), \quad (2)$$

wherein T_{block} is the round-trip time of the next data block and is determined by the capabilities of the I/O subsystem, so cannot be reduced. Then p is the number of algorithm instances processing the next data block in parallel.

Integrating the parallel accelerators discussed above into a single chip would enable C streams of data blocks to be processed in parallel. This combination could be considered the final version of the accelerator. In this scenario, the throughput R_{hard} of the combination of C parallel accelerators (with the $T_{hard}(n)$ characteristic of formula (2)) would be determined by the following equation:

$$R_{hard} = \frac{Cn}{T_{hard}(n)} = \frac{C}{\frac{T_{conf}}{n} + \max\left(T_{block}, \frac{T_{soft}}{K}\right)} \approx \frac{C}{\max\left(T_{block}, \frac{T_{soft}}{K}\right)}.$$

The C value is limited by the number of physically available interfaces in the implemented I/O subsystem

and the application task specification. Thus, when designing a hardware accelerator, decisions should be made not only about the technological processes of its manufacture and technical characteristics but also about the simultaneous development of tools. In particular this requires a specialized compiler to be used to optimize both the software and the hardware.

Analysis and evaluation of the implementation of hardware accelerators in heterogeneous computing systems for massively parallel stream processing

The previous section sets out a method for assessing the effectiveness of a general-purpose processor for massively parallel stream processing, compared to a specialized accelerator with the following characteristics:

- T_{block} is the time taken to receive and transmit a data block, as determined by the I/O subsystem (i.e., the implementation options for interfacing the accelerator with the computing system);
- f_{hard} is the clock frequency which depends on the purchasing or manufacturing options available for an accelerator. It is also determined by the capabilities of its design tools (CAD) and manufacturing technologies (PLD or custom chip);
- n_{hard} is the number of cycles required by the accelerator (stream processing node) to resolve a problem, depending on the composition of specialized parallel algorithms used and the phases of optimizing the compiler. This number is determined by the options available for providing a mathematical solution to a problem using an accelerator and its microarchitecture, considering the hardware features of the accelerator;
- p is the number of parallel data processing nodes within the accelerator involved in solving a problem.

The available options for f_{hard} values are determined by the characteristics of the PLDs and VLSI chips available for use when purchasing a ready-made ECB, as well as by their production options i.e., those available for the production of ECBs through technological processes and standard cell libraries.

When designing an accelerator, it is advisable to add the following parameters to the above-listed characteristics:

- T_{task} is the processing time of one data block by the accelerator, determined by the technical assignment requirements;
- C_{max} is the maximum number of available I/O subsystem channels;
- C ($C \le C_{max}$) is the number of accelerator instances placed on the chip;
- S ($S \le S_{max}$) is the area of the chip limited in its particular embodiment.

Some of the specific values of the above parameters depend on the implementation of the problem solution on the accelerator and the variant of the corresponding data processing algorithm used. Let a_i, $i = \overline{1,n}$ be a set of alternative algorithms, i.e., implementation options for resolving the problem on an accelerator. Each algorithm is characterized by the following set of parameters:

- n^i_{hard} is the number of clock cycles required by one computing node of the accelerator to resolve the problem;
- P_i is the maximum possible number of parallel instances of the algorithm running as part of the accelerator;
- local_i is the area of the chip occupied by the amount of local memory m_i required by the accelerator to execute the a_i algorithm;
- alu_i is the chip area occupied by one data streaming node;
- glob_i is the portion of the chip dedicated to storing data utilized by all instances of the a_i algorithm. This allocation may remain the same for regular expressions, for instance, or it may change depending on the throughput of the I/O subsystem, the workload requirements of the general-purpose processor and other factors.

Then $S_i = (\text{local}_i + \text{alu}_i)$ is the chip area allocated for one accelerator stream processing node, i.e., for the functioning of one instance of the a_i algorithm. For an accelerator with p_i stream processing nodes, each implementing an instance of the a_i algorithm, the required chip area can be calculated using the following equation:

$$U_i = S_i p_i = (\text{local}_i + \text{alu}_i)p_i,\ p_i \le P_i.$$

When placing accelerator instances on the C_i chip, each of which implements p_i instances of the a_i algorithm, the required chip area S_i can be determined in the following way:

$$S_i = \text{glob}_i + U_i C_i = \text{glob}_i + S_i p_i C_i = \\ = \text{glob}_i + (\text{local}_i + \text{alu}_i)p_i C_i,\ p_i \le P_i,\ S_i \le S.$$

Equation (2) allows not only the time $T_{hard}(n)$ required for processing n data blocks by an accelerator with p nodes to be defined, but also its implementation options to be analyzed and evaluated on the basis of existing (available) technological capabilities. We define a limit for the available values of the performance characteristic of the accelerator, i.e., the time T_{fix} it takes to process one data block, as follows:

$$T_{fix} = \min(T_{block}, T_{task}).$$

The T_{fix} time, determined either by the capabilities of the I/O subsystem or by the terms of reference for the accelerator development, can be considered to be

unambiguously deterministic. This imposes natural limitations on the combinations of characteristics that can be reasonably considered:

$$\frac{T_{\text{soft}}}{K} \approx T_{\text{fix}}. \quad (3)$$

Implementations of the accelerator which meet condition (3) using the '<' operator would be impractical, since their excessive efficiency would be limited by the capabilities of the I/O subsystem. Implementations which satisfy condition (3) with the '>' operator would not achieve maximum performance.

As a criterion for the optimal implementation of the accelerator, considering the specific implementation of the I/O subsystem and based on available technological capabilities, we can express the following:

$$\left| \frac{T_{\text{soft}}}{K} - T_{\text{fix}} \right| \to \min.$$

Taking into account (1) and the potential for implementing multiple alternative algorithms $a_i, i = \overline{1,n}$ on the accelerator, this expression can be written as follows:

$$\left| \frac{n^i_{\text{hard}}}{p_i f_{\text{hard}}} - T_{\text{fix}} \right| \to \min. \quad (4)$$

Given that the f_{hard} parameter can take different values (described by a vector of values), the following optimization accelerator design problem can be formulated: for each version of the problem-solving algorithm a_i, $i = \overline{1,n}$, the combinations of p_i, C_i, and f^i_{hard} values should be determined so that:

$$S_i = \text{glob}_i + (\text{local}_i + \text{alu}_i) p_i C_i,$$

$$p_i \le P_i,$$

$$S_i \le S,$$

$$\frac{n^i_{\text{hard}}}{p_i f^i_{\text{hard}}} \le T_{\text{fix}},$$

$$T_{\text{fix}} - \frac{n^i_{\text{hard}}}{p_i f^i_{\text{hard}}} \to \min. \quad (5)$$

As a result, corresponding sets $|\text{A}_i| = r_i$ of combinations of the values for the considered parameters $(p^k_i, C^k_i, f^{i,k}_{\text{hard}})$, $k = \overline{1,r_i}$, which are alternative versions of accelerator manufacture, can be determined for the algorithms a_i, $i = \overline{1,n}$. The accelerator design option(s) can then be selected based on the simulation results obtained using CAD tools, taking into account additional criteria such as power consumption, cost, and probability of failure.

It would also be advisable to automate the solution to the above problem of determining the accelerator characteristics in accordance with the implemented algorithm would be advisable. The authors will conduct further research in this area. However, a general approach to solving the problem is presented below.

Universal sets of various P, C and F values can be composed for the above parameters p_i, C_i, and f^i_{hard} of each of the a_i, $i = \overline{1,n}$ alternative algorithms. Together with the set A ($a_i \in \text{A}$, $i = \overline{1,n}$), these form a four-dimensional array W of the original data for the problem under consideration. If a specific variant of the four parameters or the value of the expression (5) is incompatible, then the elements of the corresponding array contain the "0" value. However, if their joint implementation is acceptable, the value of the expression (5) is used. The n^i_{hard}, local_i, alu_i, and glob_i parameters of the corresponding algorithm a_i, $i = \overline{1,n}$. are used to calculate it.

The P, C, and F sets are generally deterministic and there are usually only a few algorithmic variants available for solving a specific problem. Therefore, in the context of sets P, C, and F, array W is universal and, as a rule, does not have a large dimension when applied to a specific data processing task. Consequently, it does do not require a high level of computational complexity to calculate and enumerate the values of its elements, or to analyze the implementation of heterogeneous computing systems with hardware acceleration of massively parallel data streaming.

The problem statement and procedure for determining the initial data enable the development of a methodology for analyzing and evaluating the implementation of hardware accelerators in heterogeneous computer systems for massively parallel stream data processing. If further developed, the proposed methodology has the potential to become universal and could be used to develop technical specifications for hardware accelerators, when designed according to specified requirements. It can also be used to justify the decisions made regarding the configuration of a hardware accelerator. Furthermore, it can be used to set research and development goals aimed at achieving the target values of f_{hard}, n_{hard}, $C_{\max}$, and $S_{\max}$ for domain-specific massively parallel data processing tasks and CAD functionality.

In future works, the authors will describe the development of the methodology, including its application to working with the W array using online analytical processing technologies.

Considering heterogeneous systems of massively parallel data processing as a separate promising direction could advance the development of computer technology. The methodology proposed here is based on requirements for resolving popular object-oriented problems. It can be used as a tool for scientific and technical policy, enabling the determination and justification of the need to achieve target values. These include:

- characteristics of system buses and interfaces of accelerators with computing systems (T_{block});
- circuit engineering solutions and lithography technologies (f_{hard});
- integrated circuit packaging technologies (p);
- algorithms for solving problems and the mathematical support of compilers (n_{hard});
- high-level design systems and software modeling.

CONCLUSIONS

In order to improve the efficiency with which massively parallel accelerators resolve tasks and expand their range, an architecture and methodology for designing heterogeneous computing systems is proposed. These systems use specialized hardware accelerators with massive parallelism of calculations and independently programmable specialized processor cores. This solution complements general-purpose processor-based computing systems. Due to its ability to program individual nodes independently, its architecture differs from that of graphics processing unit devices. Increasing the number of parallel nodes and their specialization is shown to increase productivity. This is established by analysis of the computation graph formed by a specially developed compiler focused on a subclass of domain-specific tasks. The proposed calculation acceleration ratio demonstrates the feasibility of complex optimization during the accelerator development. This takes into account three levels of project presentation: a program model, a schematic description, and a topological representation based on the selected technology. The methodology proposed for analyzing and evaluating the implementation options for hardware accelerators of heterogeneous computing systems for massively parallel stream processing allows the technical characteristics of the required accelerator to be determined. The resulting methodology for designing computing systems of this class provides an adjustable process for optimizing performance according to the required criteria.

Authors' contributions

The research and results presented in the article were obtained jointly by the authors as an outcome of the scientific school "Specialized Heterogeneous Computing Systems" organized by the Institute of Information Technologies at the MIREA – Russian Technological University.

The main content of the section "Analysis and evaluation of implementation options for hardware accelerators in heterogeneous computing systems for massively parallel stream data processing," including the task of optimizing the hardware accelerator architecture formulation as a search problem in the space of architectural options, as well as the methodology for analyzing and evaluating implementation options of hardware accelerators, was developed by A.S. Zuev.

REFERENCES

1. Dennard R.H., Gaensslen F.H., Yu H.-N., Rideout V.L., Bassous E., LeBlanc A.R. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*. 1974;9(5):256–268. https://doi.org/10.1109/JSSC.1974.1050511
2. Jain P.U., Tomar V.K. FinFET Technology: As A Promising Alternatives for Conventional MOSFET Technology. In: *2020 International Conference on Emerging Smart Computing and Informatics* (*ESCI*). 2020. P. 43–47. https://doi.org/10.1109/ESCI48226.2020.9167646
3. Yakimets D., Eneman G., Schuddinck P., et al. Vertical GAAFETs for the Ultimate CMOS Scaling. *IEEE Transactions on Electron Devices*. 2015;62(5):1433–1439. https://doi.org/10.1109/TED.2015.2414924
4. Lee S.-Y., Kim S.-M., Yoon E.-J., et al. A Novel Multibridge-Channel MOSFET (MBCFET): Fabrication Technologies and Characteristics. *IEEE Transactions on Nanotechnology*. 2003;2(4):253–257. https://doi.org/10.1109/TNANO.2003.820777
5. Hennessy J.L., Patterson D.A. *Computer Architecture: A Quantitative Approach* (*The Morgan Kaufmann Series in Computer Architecture and Design*). 6th ed. 2017, 936 p.
6. Annaratone M. MPPs, Amdahl's Law, and Comparing Computers. In: *Proceedings of The Fourth Symposium on the Frontiers of Massively Parallel Computation.* 1992. P. 465–470. https://doi.org/10.1109/FMPC.1992.234879
7. Verhelst M., Benini L., Verma N. How to keep pushing ML accelerator performance? Know your rooflines! *IEEE Journal of Solid-State Circuits*. 2025;6(60):1888–1905. https://doi.org/10.1109/JSSC.2025.3553765
8. Altaf M.S.B., Wood D.A. LogCA: A high-level performance model for hardware accelerators. *ACM SIGARCH Computer Architecture News*. 2017;45(2):375–388. https://doi.org/10.1145/3079856.3080216
9. Molina R.S., Gil-Costa V., Crespo M.L., et al. High-level synthesis hardware design for FPGA-based accelerators: Models, methodologies, and frameworks. *IEEE Access*. 2022;10:90429–90455. https://doi.org/10.1109/ACCESS.2022.3201107
10. A New Golden Age for Computer Architecture: Domain-Specific Hardware/Software Co-Design, Enhanced Security, Open Instruction Sets, and Agile Chip Development. In: *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture* (*ISCA*). 2018. P. 27–29. https://doi.org/10.1109/ISCA.2018.00011
11. Liu K., Lu A., Fang Z. BitBlender: Scalable Bloom Filter Acceleration on FPGAs with Dynamic Scheduling. In: *34th International Conference on Field-Programmable Logic and Applications* (*FPL*). 2024. P. 325–331. https://doi.org/10.1109/FPL64840.2024.00052

12. Kulkarni A., Chiosa M., Preuber T.B., et al. HyperLogLog Sketch Acceleration on FPGA. In: *30th International Conference on Field-Programmable Logic and Applications* (*FPL*). 2020. P. 47–56. https://doi.org/10.1109/FPL50879.2020.00019
13. Marchisio A., Teodonio F., Rizzi A., et al. ISMatch: A Real-Time Hardware Accelerator for Inexact String Matching of DNA Sequences on FPGA. *Microprocess. Microsyst*. 2023;97:104763. https://doi.org/10.1016/j.micpro.2023.104763
14. Zhang C., Tang X., Peng Y. Enhancing Regular Expression Processing through Field-Programmable Gate Array-Based Multi-Character Non-Deterministic Finite Automata. *Electronics.* 2024;13(9):1635. https://doi.org/10.3390/electronics13091635
15. Dann J., Wagner R., Ritter D., et al. PipeJSON: Parsing JSON at Line Speed on FPGAs. In: *Proceedings of the 18th International Workshop on Data Management on New Hardware*. 2022;Article 3:1–7. https://doi.org/10.1145/3533737.3535094
16. Karandikar S., Udipi A.N., Choi J., et al. CDPU: Co-Designing Compression and Decompression Processing Units for Hyperscale Systems. In: *Proceedings of the 50th Annual International Symposium on Computer Architecture*. 2023;Article 39:1–17. https://doi.org/10.1145/3579371.3589074
17. Hahn T., Wildermann S., Teich J. JSON-CooP: A JSON Decompression/Parsing Co-Design for FPGAs. In: *34th International Conference on Field-Programmable Logic and Applications* (*FPL*). 2024. P. 11–18. https://doi.org/10.1109/FPL64840.2024.00012
18. Fang J., Mulder Y., Hidders J., et al. In-Memory Database Acceleration on FPGAs: A Survey. *The VLDB Journal.* 2020;29(10):33–59. https://doi.org/10.1007/s00778-019-00581-w
19. Dann J., Götz T., Ritter D., et al. GraphMatch: Subgraph Query Processing on FPGAs. *arXiv.* arXiv:2402.17559. 2024.
20. Kejariwal A., Kulkarni S., Ramasamy K. Real Time Analytics: Algorithms and Systems. *arXiv.* arXiv:1708.02621. 2017. https://doi.org/10.48550/arXiv.1708.02621
21. Alcolea A., Resano J. FPGA Accelerator for Gradient Boosting Decision Trees. *Electronics*. 2021;10(3):314. https://doi.org/10.3390/electronics10030314
22. Graf J.R., Perera D.G. Optimizing Density-Based Ant Colony Stream Clustering Using FPGA-Based Hardware Accelerator. In: *2023 IEEE International Symposium on Circuits and Systems* (*ISCAS*). 2023. https://doi.org/10.1109/ISCAS46773.2023.10181665
23. Shen J.P., Lipasti M.H. *Modern Processor Design: Fundamentals of Superscalar Processors.* Waveland Press; 2013, 658 p.
24. Tarasov I.E., Sovietov P.N., Lulyava D.V., Mirzoyan D.I. Method for designing specialized computing systems based on hardware and software co-optimization. *Russian Technological Journal*. 2024;12(3):37−45. https://doi.org/10.32362/2500-316X-2024-12-3-37-45

About the Authors

Andrey S. Zuev, Cand. Sci. (Eng.), Associate Professor, Head of the Department of Quantum Information Technologies, Practical and Applied Informatics, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: zuev_a@mirea.ru. Scopus Author ID 23977152300, RSCI SPIN-code 6737-5778, https://orcid.org/0000-0002-1797-7585

Peter N. Sovietov, Cand. Sci. (Eng.), Associated Professor, Department of Corporate Information Systems, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: peter.sovietov@gmail.com. Scopus Author ID 57221375427, RSCI SPIN-code 9999-1460, http://orcid.org/0000-0002-1039-2429

Ilya E. Tarasov, Dr. Sci. (Eng.), Associated Professor, Professor, Department of Corporate Information Systems, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: tarasov_i@mirea.ru. Scopus Author ID 57213354150, RSCI SPIN-code 4628-7514, http://orcid.org/0000-0001-6456-4794

Об авторах

Зуев Андрей Сергеевич, к.т.н., доцент, заведующий кафедрой квантовых информационных технологий, практической и прикладной информатики, Институт информационных технологий, ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: zuev_a@mirea.ru. Scopus Author ID 23977152300, SPIN-код РИНЦ 6737-5778, https://orcid.org/0000-0002-1797-7585

Советов Петр Николаевич, к.т.н., доцент, кафедра корпоративных информационных систем, Институт информационных технологий, ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: peter.sovietov@gmail.com. Scopus Author ID 57221375427, SPIN-код РИНЦ 9999-1460, http://orcid.org/0000-0002-1039-2429

Тарасов Илья Евгеньевич, д.т.н., доцент, профессор кафедры корпоративных информационных систем, Институт информационных технологий, ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: tarasov_i@mirea.ru. Scopus Author ID 57213354150, SPIN-код РИНЦ 4628-7514, http://orcid.org/0000-0001-6456-4794

Translated from Russian into English by Kirill V. Nazarov
Edited for English language and spelling by Dr. David Mossop