A Comprehensive Survey of Domain-Specific Hardware Acceleration in Scientific Computing

Soham Bhattacharya
Department of Electrical and Computer Engineering, Rowan University, USA.
*Email: bhatta22@rowan.edu

Abstract : Technology progress has changed the way research and development tasks are done. The capability to perform intricate system modeling and simulation is now crucial in engineering, physics, chemistry, biology, and other industries relying on scientific computing. As these scientific computing workloads require quick and efficient execution, there is an increasing need for disruptive technology. This is where hardware accelerators enter the picture. These accelerators, capable of managing intricate systems modeling and simulation, hold promise for improving precision and dependability in research and development results, thereby saving time and resources. In this survey, we define, summarize, and analyze the accelerators required in different scientific computing domains. We have also proposed a taxonomy based on these aspects: implementation methods; types of implementations; host coupling; cost factors; and applications, grouped into five macro-categories. Then we observed and categorized the intersection of micro-architectures with differential equations in three areas in the scientific computing domain, like acceleration for higher-order nonlinear systems, acceleration based on differential equations using off-the-shelf accelerators, and finally acceleration using customized co-processors. Lastly, we finish the survey by giving a brief summary. We think that these methods will be useful for researchers who are aiming to make new hardware accelerators in the scientific computing area.

Keywords: Hardware accelerators, Scientific Computing, Taxonomy, FPGA, GPU, Domain-specific architectures, Survey.

I. INTRODUCTION

In semiconductor electronics, Dennard scaling is the process of incorporating more transistors with the same power by reducing the supply voltage. Around fifteen years ago, hardware manufacturers faced setbacks on the path of Dennard scaling as they grappled with complexities based on power dissipation, leakage current, and wire delays [1]–[4]. During that time, stagnation arose in the improvement of the single-core microarchitecture. Hardware manufacturers shifted their attention to new design models, giving rise to “multi-core” micro-architectural design. This approach involves the incorporation of multiple cores onto a single die (functional electronic components fabricated on a thin piece of silicon) [5]. However, that approach was short-lived and insufficient for the worldwide demand for simultaneously diminishing the computing power and increasing the performance of the device. This realization made it apparent that more compact strategies are required to deal with high computing demands.

To overcome the opposing requirements of computing power and performance, the solution embraced by many computer architects is to customize the hardware for specific tasks. This approach has been successfully implemented in the graphics processing unit (GPU) domain, thereby marking significant growth over the past decade. This is referred to as Domain-Specific Architectures (DSAs) or accelerators [6]. By configuring hardware for special applications or workloads, these DSAs aim to deliver improved performance, efficiency, and low computing power. This fulfills the evolving demands of computational tasks in various domains of scientific computing [4], [8]–[11]. In Figure 1, an overview of a system of hardware accelerators architecture has been presented.

In this figure, the central processing unit (CPU) 'offloads' its task to any of the accelerators, which executes the assignment faster and delivers the result to the CPU [7]. Differential equations are often referred to as nature's language because they provide a way to describe and understand many natural phenomena in a precise and systematic way. Differential equations capture the dynamics of physical systems and are fundamental to progress in the computational sciences. It is known that systems of differential equations can be solved analytically to obtain exact solutions. However, this is often not possible due to the complexity of the systems being studied. This is where the power of scientific computing and DSAs can be leveraged. Numerical methods are instrumental in approximating solutions to differential equations and exploring their behavior effectively. The "Multigrid method" [12], [13] stands out as a very efficient technique for solving linear differential equations (specifically for partial differential equations (PDEs)) by utilizing a hierarchical structure of discretizations to obtain precise results. These methods typically involve dividing the domain of the differential equation into discrete points and then, approximating the solution at each point by employing techniques like finite difference methods or finite element methods [14].

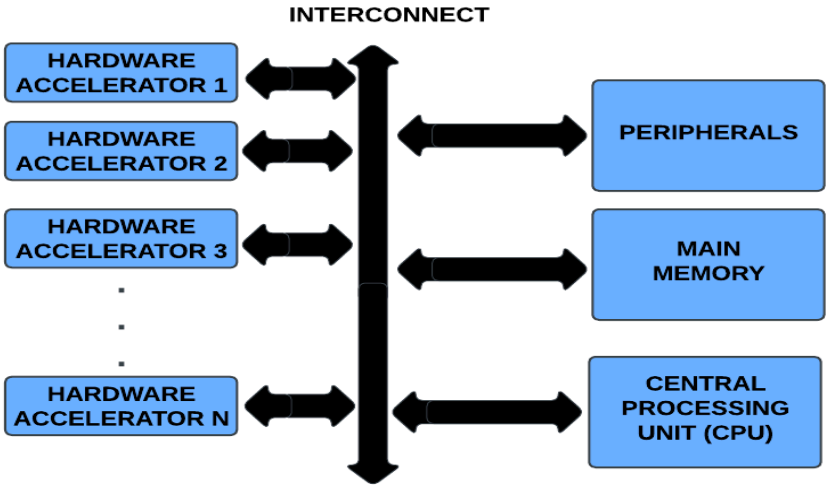

Figure 1. Overview of an architecture of a system of hardware accelerators.

Recent trends have popularized numerous techniques in machine learning for modeling, simulation, and the solution of differential equations. Machine learning models being used in this area are essentially powerful function approximators [15]–[17]. Although these algorithms utilize general-purpose processors, they are still computationally intensive. This is especially true when there is a requirement for solving large, complex systems. Also, complex machine learning models are not transparent in that the explainability aspect (showing how a decision was arrived at) is not clearly understood [18]. More efficient acceleration toward a solution can be achieved by DSAs, which leverage the parallel architecture inherent in GPUs [19], [20].

This comprehensive survey paper investigates the intricate relationship between differential equations, multigrid methods, and hardware acceleration in the realm of scientific computations. Specifically, the paper discusses:

• A variety of applications, including (1) aerodynamic simulations for aircraft design, (2) Monte Carlo simulations in statistical mechanics, and (3) computational fluid dynamics.

• The potential benefits of domain-specific accelerators designed specifically for advanced modeling and simulation tasks, thereby alleviating bottlenecks in the simulation process.

• The recent advancements in hardware acceleration for advanced modeling and simulation.

• The design of the specialized hardware to do certain scientific workloads, particularly multigrid methods, which can significantly enhance simulation efficiency.

The outline of this paper is as follows. In Section II, the taxonomy representation of this survey has been represented. The motivation is discussed by exploring the fundamental role of differential equations in modeling complex systems and emphasizing their pivotal role in the advancement of computational sciences in Section III. The focus of Section IV is to discuss the contributions of multigrid methods. Also discussed is their categorization into various modalities, such as structured, block-structured, and adaptively structured grids. In Section V, the paper discusses an overview of the current methodologies and implementations designed to accelerate several higher-order nonlinear systems. Additionally, in Section VI and VII, the paper aims to provide a comprehensive understanding of implementations of hardware accelerators in scientific computing using GPU and customized co-processors such as FPGA respectively. Finally, a summary has been provided based on the scientific computing in Section VIII before concluding the survey in Section IX.

II. TAXONOMY REPRESENTATION

To give a complete description of the hardware accelerator, we derive twenty one criteria and sorted them in five main categories: a. Implementation methods, b. Implementation types, c. Host Coupling, d. Cost factors, and e. Applications. Figure 2 gives a graphical representation of our proposed coherent taxonomy for this survey.
In this section, we discuss the descriptive classification of each area. It must be mentioned that certain possibilities may not be mentioned for two reasons: either they are not an accurate representation of the accelerators we considered, or we are restricted by time for more detailing. In the following, we acknowledge some exclusions for the sake of completeness.

A. IMPLEMENTATION METHODS

This macro-category holds certain numerical methods that apply to solve several ordinary differential equations in the field of scientific computing.

The **Euler method** [21] is one of the basic or simplest numerical methods for solving ordinary differential equations (ODEs). It was named after Leonhard Euler, the Swiss mathematician who formulated this approach during the 18th century. The Euler method comes from the concept of estimating the solution of an ODE by moving forward in time with small steps, using a derivative function for approximating how much the function value changes in each time step.

The **Modified Euler method,** which is known as “Heun's method” or “Improved Euler method”, offers a numerical way for solving ordinary differential equations (ODEs). This technique presents an enhancement on the basic Euler's way by providing more accurate estimations of slope over every time step size. This method also gives a superior precision in comparison to simple Euler methods because it employs a weighted mean value of slopes at start and middle points within each time interval. However, it's still quite simple and might not be as precise or efficient compared to more advanced methods like those from the Runge-Kutta family [21].

The **Runge Kutta method**, also known as the “RK” method, is one of the most commonly used numerical techniques for solving ordinary differential equations. It's named after the German mathematicians Carl David Tolmé Runge and Wilhelm Eduard Friedrich Kutta. The basic idea of this method is to approximate the solution of an ODE by advancing it through time steps that are discrete. Frequently used modification in this approach includes four stages; hence it's called fourth-order Runge-Kutta method (RK4). This particular variant has become quite popular because of its ability to deliver accurate results while being computationally efficient at the same time [22].

The **multigrid method** [23] is a type of iterative method that makes use of grid hierarchy having different levels of resolution to speed up the convergence process for PDEs solved by iterative solvers. The main concept behind this approach involves solving the problem initially on a coarse grid, where solution can be obtained relatively easily, and then progressively refining it on finer grids to reach more precise results. This approach is generally applied in solving partial differential equations (PDEs), specifically those that come up in computational fluid dynamics (CFD), electromagnetics, and other fields of applied mathematics along with engineering.

The **L-stable method** [24] is one that remains stable with a large step size even if changes occur in parameter values or step sizes. It's said to be L-stable because its stability holds up across a wide variety of situations, including when the numerical approximation's step size is quite big. This feature is crucial in making methods for solving stiff ordinary differential equations (ODEs) efficient and useful. It permits using bigger steps, leading to quicker convergence of results while maintaining stability. L-stability is a property that some numerical methods possess, allowing them to handle stiff ordinary differential equations (ODEs). Two such methods are the backward differentiation formulas (BDF) and Gear's method, which are implicit Runge-Kutta methods. These techniques find common application in tackling stiff ODEs encountered in various scientific and engineering domains.

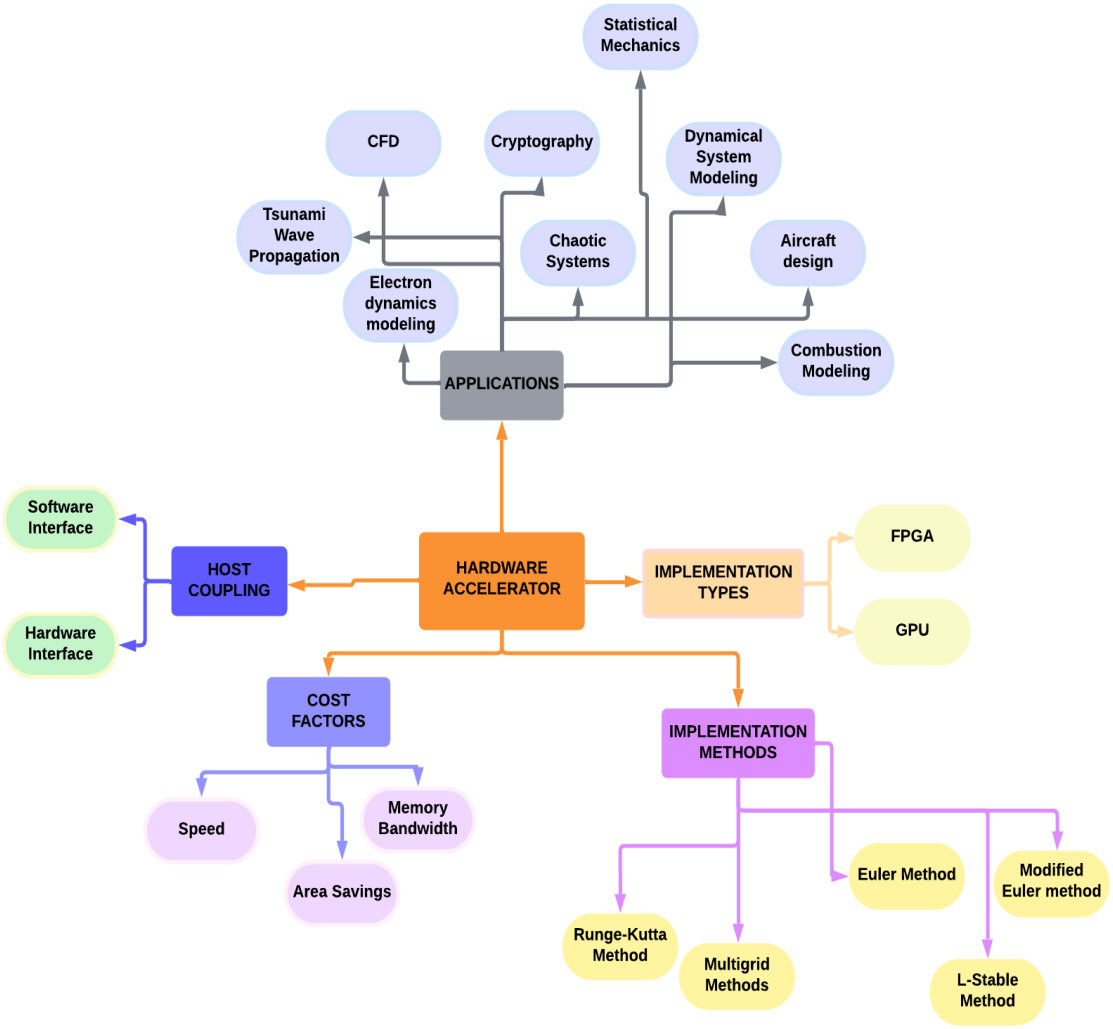

Figure 2. A graphical representation of our proposed taxonomy for the survey.

B. IMPLEMENTATION TYPES

In this category, we have targeted two areas to construct the hardware acceleration in scientific computing. They are FPGA and GPU. Below is the detailed explanation of these two areas.

An **FPGA** or "Field-Programmable Gate Array" [25] is a kind of hardware accelerator used to enhance the computing performance for a certain workload. The FPGAs are usually reconfigurable and programmable hardware

devices that perform specific computations or tasks. Generally, they contain an array of configurable logic blocks (CLBs), interconnects, along with memory blocks which can be programmed in order to implement customized digital circuits. FPGAs are used for their flexibility and adaptability that allows the hardware architects or developers for their specific customized requirements. Some of the common applications where FPGAs are commonly used include signal processing, cryptography, networking, image processing etc.

A **GPU** or "Graphics Processing Unit", on the other hand, is another type of hardware accelerator designed primarily for rendering graphics in computer graphics applications. A common characteristic of GPUs includes many processing cores made for parallel computation along with dedicated memory and fast interconnects/buses. Operations such as matrix multiplication, vector computation and parallel algorithms are enhanced by the use of a GPU. In addition, GPUs excel at highly parallel computations with large amounts of data parallelism due to their parallel processing capabilities [26].

In situations where there is a requirement for high-performance computing, both FPGA and GPU can be combined in heterogeneous environments. This allows the use of each component's unique advantages to handle specific parts of a computation.

C. HOST COUPLING

Host Coupling is the process of incorporating external hardware or software protocols that assist in linking and exchanging communication between various hardware and software components within a computing system. Host coupling typically includes physically or virtually connecting customized hardware accelerators or peripheral devices with a host system (such as PC, server) for effectively working together. We have categorized this macro-category into two types: i. Hardware Interface and ii. Software Interface.

The **hardware interface** involves the actual connectors, buses and protocols that are used to connect host systems and accelerators. Based on the survey we conducted, it is seen that various interfaces like PCIe (Peripheral Component Interconnect Express), AXI (Advanced extensible Interface), Ethernet etc., have been utilized. The hardware interface gives a physical link for transferring data and might also involve power transfer, clock matching along with other signaling methods.

On the other hand, the **software interface** is the combination of APIs (Application Programming Interfaces), libraries, and protocols used to allow communication and exchange of data between the host system's software with external devices. Libraries along with APIs give a more understandable programming interface that helps developers access features and abilities of external devices in an efficient way. Protocols are responsible for setting communication standards and data formats to exchange information between the host system and outside devices which guarantees compatibility as well as interoperability.

D. COST FACTORS

We have derived this macro-category based on three sections, which involve speed, area saving and memory bandwidth. For hardware systems, the design and optimization depend on speed, area savings, and memory bandwidth. Every factor has a unique influence on the overall cost.

Speed is one of the crucial criteria for faster hardware systems. These systems often require more advanced and specific parts, resulting in increased costs. These systems bring better performance, less delay and faster results that are frequently crucial that is extremely crucial in the high performance computing domain. However, strategies for attaining more speed can be many, like utilizing faster clock speeds, optimizing important paths within the design, reducing propagation delays and lessening overheads in data processing. But, when we enhance speed, it generally includes compromises such as more energy usage, greater heat emission and possibly increased expenditures linked to specific parts or procedures for making them better or design intricacy. Sometimes, if we have a use that sets great importance on time-to-market or competitive edge, the advantages of quicker speed could make paying more acceptable.

Area savings refer to reducing the physical size or footprint of hardware components, which can result in cost reductions within semiconductor manufacturing and packaging. Methods frequently employed to optimize area are logic synthesis, layout optimization, resource sharing and architectural enhancements. When designers improve the design and arrangement of integrated circuits, they can reduce the amount of silicon area that must be used for a certain function. Saving area might result in less cost when making things, because it requires fewer materials to make silicon. Furthermore, the need for power and cooling is less in small components. This can also lessen the operational costs throughout the lifespan of a hardware system.

Memory Bandwidth indicates how fast data can move between the processor and memory subsystem. A bigger memory bandwidth lets us access and process data more quickly, which may enhance the total system performance. But, to increase memory bandwidth usually we need to use specific memory technologies like high-speed DDR or HBM (High Bandwidth Memory). These could be costlier than typical options for memory. So, when we speak about memory bandwidth optimization, it's like finding a balance between what performance needs and how much money one can spend to get that desired level of performance at an acceptable cost.

In general, making the most of speed, area savings and memory bandwidth is a balance that requires thinking about design aims, performance needs and cost limits. When these aspects are balanced well, it helps in creating hardware systems that meet the set performance goals while also reducing expenses and enhancing worth.

E. APPLICATIONS

According to our survey, we have focused on the hardware implementation for the applications specific to the scientific computing field. Those areas include Computational Fluid Dynamics (CFD), Cryptography, Chaotic systems, Statistical mechanics, Dynamical systems modeling, Electron dynamics modeling, Aircraft design, combustion modeling, Tsunami wave propagation.

Computational Fluid Dynamics or CFD is a section of fluid mechanics that employs numerical procedures and algorithms to replicate the movements of fluids. It involves the study and analysis of fluid flow situations in engineering and scientific applications. **Cryptography** refers to the study dealing with secure communication techniques. This involves the methods of encryption and decryption, which are used to maintain data's secrecy, accuracy as well as genuineness in information security systems. **Chaotic systems** are certain kinds of deterministic dynamical systems that have a built-in characteristic known as sensitive dependence on initial conditions (SDIC). They show a behavior that appears random or not foreseeable but is actually following hidden mathematical rules often studied under chaos theory. **Statistical mechanics** uses statistical methods to guess the properties of big systems made from many particles by looking at small-scale behavior on a single particle level. It has important use in thermodynamics and phase transitions. **Modeling of dynamical systems** is about learning how a system acts as it goes through changes over time. This consists of equations that explain how state variables change with respect to each other over time or in a sequence of events; this area is helpful to comprehend issues in physics, engineering and other fields like biology or economics among others. **Electron dynamics modeling** is about creating simulations on how electrons act inside materials, devices or systems - usually using quantum mechanical ways for studying features such as electronic transport, band structure and operation of semiconductor devices. **Aircraft design** is the process related to designing and building aircrafts, which involves things such as aerodynamics, structure analysis, propulsion systems and avionics. The goal is to model them in order to make these elements work at their best performance level efficiently and safely too. **Combustion Modeling** implies the formation of simulations related to chemical reactions and fluid movements occurring within combustion processes. Such processes can be found within engines, furnaces or any area where there's burning occurring like a combustion chamber. The key goal is comprehension of how these reactions occur, so we can enhance things like fuel burn efficiency by decreasing waste from it and also producing less harmful emissions; additionally optimizing energy production rates related with this activity (combustion). **Tsunami Wave Propagation** emphasizes the science behind the movement of tsunami waves in oceans worldwide. It includes comprehending different elements such as their creation or birth, travel through water bodies until they touch coastlines then what occurs when interacting with coastal areas. Here, the main aim is to assess risks related to tsunamis along shores all over the world so that actions can be taken for reducing risk which will help people living close by coastsides (tsunami hazards assessment).

III. MOTIVATION

The reason behind conducting this study is based on the changing landscape of the scientific environment, where the accuracy and efficiency of simulations are critical. The motivation for writing this survey is given in each subsection.

A. MULTIGRID METHODS

In this present era of high-performance computing, the need to solve complex systems of differential equations is at the heart of many scientific and engineering challenges [27], [28]. Nowadays, researchers face challenges in solving these sets of non-linear and complex equations and thereby heavily rely on numerical simulations to make informed decisions [29], [30]. Multigrid methods are important tools for solving these equations due to their robustness, scalability (ability to handle increasing numbers of computational tasks), and ability to be parallelizable for high-performance computing architectures [31]. Robustness implies the ability to solve a wide range of problems with several characteristics like boundary conditions, distinct grid resolutions (coarse to fine), and types of equations (linear and nonlinear). This survey aims to gain an understanding of these methods while considering their computational intensity and the increasing demand for faster and more accurate solutions.

B. HARDWARE ACCELERATION

Hardware acceleration is an essential instrument for a wide range of computational workloads across various domains due to the increased benefits in energy efficiency, real-time processing capabilities, parallelism, and better performance (high accuracy with less processing time). It is essential to investigate domain-specific hardware solutions in (1) expediting multigrid and other numerical methods, (2) comprehending the difficulties in implementing them, and (3) realizing their impact on simulation fidelity (accurate predictions in real-world phenomena) [32] and efficiency (measurement of the effective computational resources to meet the computational needs) [33]. This approach helps not only in assessing the current state of hardware-accelerated scientific computing but also in identifying future directions and potential breakthroughs in this rapidly evolving field.

C. PARALLEL PROCESSING

Parallel processing capabilities refer to the system's ability to perform multiple tasks simultaneously. The parallel processing competencies of hardware accelerators allow overall performance improvements in workloads that contain large and complicated numerical computations [34]. In Figure 3, the different types of parallel processing architecture have been represented. Generally, there are four types of parallel processing architecture. Figure 3(a) defines the "Single Instruction, Single Data" (SISD) architecture. This architecture executes a single instruction stream on a single data stream. This is typically seen on Von Neumann computers. Figure 3(b) is the "Multiple Instruction, Single Data" (MISD) architecture. This is typically executing multiple instructions on a single data stream. In Figure 3(c), we see how an instruction can work on many data elements at once, which is typically seen in vector processors and GPUs. This is termed as "Single Instruction, Multiple Data" (SIMD) architecture. Finally, in Figure 3(d), all the independent instructions are assigned to different areas of the chip, making it effective to perform various tasks autonomously with optimized resource utilization, which finally executes as "Multiple Instruction, Multiple Data" (MIMD) architecture.

Parallel processing also involves decomposing a larger computational complexity task into smaller concurrent problems with careful consideration of data sharing, communication, and dependencies between parallel tasks to ensure effective and accurate results. Another factor that is necessary for high-performance scientific simulations is the scalability of the system. Systems with parallel processing capabilities can branch out to satisfy growing computing needs or scale up to tackle bigger issue sizes.

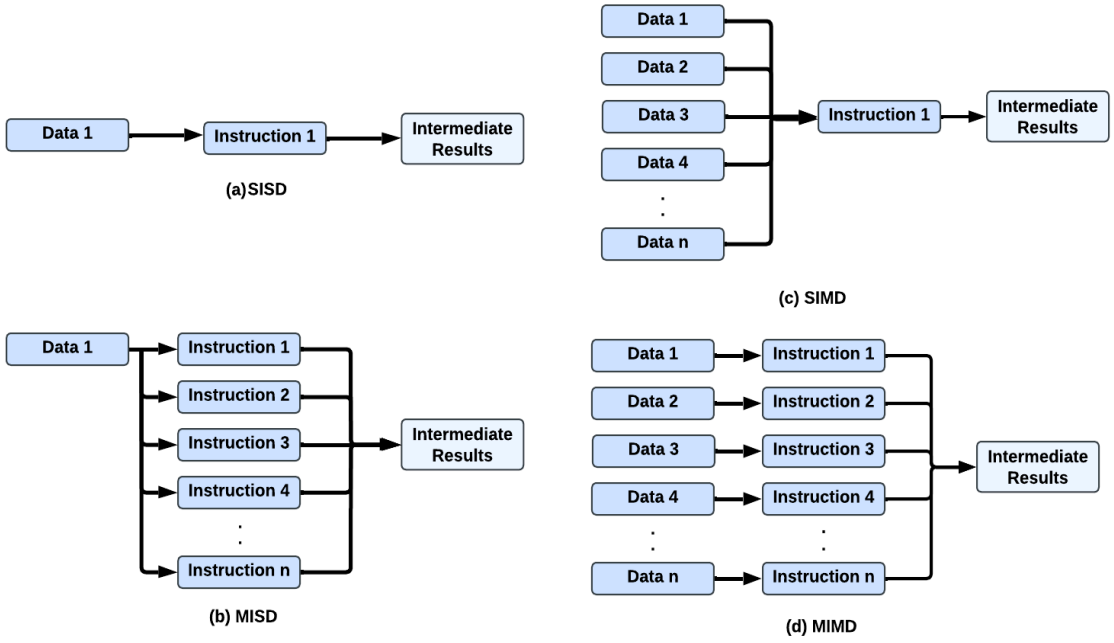

Figure 3. Types of Parallel Processing Capabilities of a system.

D. SYNERGIES AMONG DISCIPLINES

Finally, this study seeks to (1) highlight the synergies among different disciplines like pure science, theoretical mathematical concepts (particularly in the realm of differential equations), computer science, and engineering and (2) examine how their integration can lead to groundbreaking improvements in computational science. This holistic perspective is essential for fostering innovation and enhancing the capability to tackle complex scientific problems with greater precision and efficiency.

IV. CONTRIBUTION

The intersection of micro-architectures and differential equations is observed primarily in three different areas, as outlined below.

• Hardware implementation of higher-order non-linear systems, such as chaotic systems.

• Acceleration of scientific computing workloads, based on systems of differential equations, with commercial off-the-shelf accelerators such as graphics processing units.

• Acceleration of scientific computing workloads, based on systems of differential equations, with customized, application-specific hardware accelerators.

The overall contributions of the survey are based on these three aspects and are given in each subsection.

A. CATEGORIZATION OF MULTIGRID METHODS

Multigrid methods are generally utilized as preconditioners [35]–[37] and solvers [38]. For any generalized application multigrid methods can be broken down into various modalities. One of those modalities is the grid formation algorithm. Depending on this algorithm, multigrid methods can be structured, block-structured, or

adaptively structured. Another way to classify these methods is based on computational performance and convergence trajectory. Characterized by this path of dependency, multigrid methods are categorized into various cycle types such as V-cycle (a traversal pattern involving correction steps across different grid resolutions), F-cycle (Also known as 'Full Multigrid cycle' with an extra relaxation step at each grid level), and W-cycle (traversal pattern with additional correction steps at intermediate levels). In this survey, the attributes of multigrid methods will be examined at the intersection of various modalities (grid formation algorithm, computational performance, convergence trajectory, and scope of parallelization).

B. EVALUATION OF HARDWARE ACCELERATION USING COMMERCIAL HARDWARE

Software solvers often use multi-scale, multi-resolution multigrid algorithms to achieve the performance demanded by scientific computing tasks. Multigrid solvers are computationally demanding but they result in a bottleneck during the simulation process. In the current era of domain-specific computing, multigrid methods use commercially available general-purpose graphics processing units (GPGPUs) to expedite the solutions of the linearized system of equations generated from intricate systems of differential equations [39], [40]. Mostly, this survey focuses on exploring the role of GPGPUs in accelerating scientific computations. This survey also includes the challenges in implementing these methods on commercially accessible hardware.

C. ANALYSIS OF DOMAIN-SPECIFIC ARCHITECTURES

While there is an absence of a generalized framework for synthesizing co-processors to accelerate multigrid methods, the existing literature is focused on the instances of co-processor development customized for narrow applications in some scientific disciplines. This study illustrates and analyzes the usage of field-programmable gate arrays (FPGAs) and other domain-specific hardware accelerators in several scientific domains like computational fluid dynamics, multi-body physics, and computational biology. The review also scrutinizes these solutions by assessing their effectiveness, scope, and applicability across these domains.

V. HARDWARE IMPLEMENTATION OF HIGHER-ORDER NONLINEAR SYSTEMS

In modern times, higher-order nonlinear systems like chaotic systems (dynamical systems with unpredictable behavior) demand the use of hardware implementations for their simulation and analysis. Employing hardware provides a greater understanding of the behavior and dynamics of chaotic systems. In this case, the authors of [41] introduce a chaotic oscillator characterized by a single parameter and an exponential function. The oscillator exhibits an open curve of equilibrium points, underscoring the system's potential to facilitate hidden attractors. The system's dynamism is validated through the Lyapunov exponents spectrum (estimation of the rate of entropy production) [42] and bifurcation diagram computations, with implementation realized by a Field-Programmable Gate Array (FPGA). Also, the authors outline a step-by-step approach designed to solve Poisson's equation in two dimensions. The technique's design enables an easy programming style using vector and array operations, and it works comparably to the regular Successive Over-Relaxation (SOR) method. Moreover, the strategy exhibits notable performance improvements and fits well within a multigrid framework.

Another research in [43] presents FPGA implementations of two distinct chaotic systems (the single-switch Jerk system and the two-wing butterfly system) that employ a switching-type non-linearity. The authors utilize compact VHDL codes and develop two hardware architectures for parameter-independent and customized ones. According to the research, the parameter-independent architecture is more resource-intensive, while the customized architecture, using less FPGA resources, achieves higher throughput. The experimental findings underscore the substantial improvement in operating frequency and resource utilization when applying enhancement techniques with the customized architecture. The research also emphasizes the significance of these applications in chaotic systems for random bit generators and various digital signal processing applications. Figure 4 presents the top-level hardware block diagram of the oscillator system. The block takes initial state variables x0, y0, and z0 along with the coefficient parameters a, b, c, break-point B, and switching levels S1 and S2 as inputs. The block processes them according to the system's dynamics and generates three output buses for the state variables xn, yn, and zn. These buses are fed back as inputs to the block to generate updated outputs for the next iteration of the system.

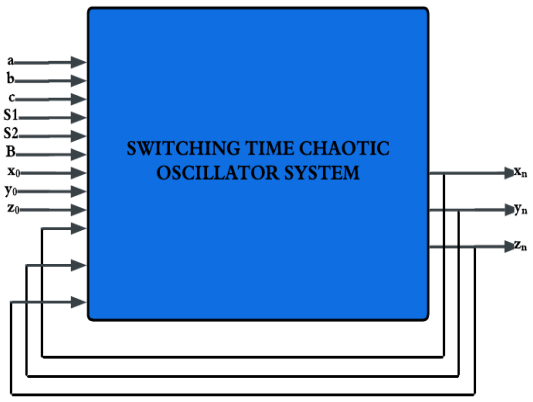

Figure 4. The top-level hardware block diagram of the oscillator system (adopted from [37]).

In [44], an FPGA implementation of an autonomous Lü-Chen (2002) chaotic system (a dynamical system with its ability to produce chaotic behavior in the absence of control inputs) using the Heun numerical algorithm in VHDL 32-bit IQ-Math fixed-point number format has been presented. The authors developed core units compatible with fixed-point number standards and synthesized the system for a Virtex-6 FPGA chip. The system is defined by Equation 1 and has the initial conditions set at (-10, 0, 37). The system is defined as the transition between the Lorenz and Chen systems. The system's real constants are 'a', 'b', and 'c'. The parameters 'a' and 'b' are fixed at 36 and 3, respectively, whereas 'c' is a variable parameter within the equations. In the paper, the system's attractor is homogeneous to the Lorenz attractor when the parameter 'c' lies within the range of 12.7 and 17, and similar to the Chen attractor when 'c' ranges between 23 and 28.5. According to the research, FPGA-based implementations of chaotic systems could surpass other conventional performance and resource utilization techniques. The research suggests that such hardware-based design of the Lü-Chen chaotic system can be applied in various chaos-based embedded system applications, including cryptography and secure communication.

$$\begin{aligned} x &= a(y - x) \\ y &= (xz) + (cy) \\ z &= (xy) - (bz) \end{aligned} \tag{1}$$

In [45], a time-delayed chameleon (TDC) like chaotic system has been implemented. The research is notable for exhibiting multiple chaotic flows depending on parameter settings. The paper explores the system's dynamics through bifurcation analysis and synchronization techniques. It introduces an adaptive modified functional projective lag synchronization (AMFPLS) algorithm for synchronizing systems with uncertain parameters. The TDC system and the synchronization algorithm are implemented on an FPGA platform using hardware-software co-simulation, demonstrating the system's practical application potential. The results confirm the feasibility of implementing complex chaotic systems on FPGA for applications like secure communications. The paper concludes with the validation of the proposed AMFPLS algorithm and FPGA implementation, suggesting the efficiency and effectiveness of these approaches.

The authors of [46] elucidate the feasibility of implementing two-dimensional rotation on a chaotic system, enabling the rotation of its attractor in space without altering the system's chaotic dynamics. This rotation operation preserves the system's eigenvalues at all equilibrium points. Furthermore, the largest Lyapunov exponent remains unchanged, as demonstrated through two chaotic systems, including the classical Lorenz system, with numerical simulations and FPGA implementation. The research has implications for chaos-based encryption, where dynamic rotation of attractors could enhance security without affecting the system's behavior. FPGA-based results show efficient resource use, suggesting the approach's feasibility for hardware applications. Another study in [47] modifies product integration (PI) rules modifies product integration (PI) rules for solving fractional-order systems (FOS) on FPGA, addressing high memory dependencies of traditional PI rules. The modified rules such as PI rectangular, PI

trapezoidal, and predict-evaluate-correct-evaluate (PECE), are validated with a benchmark system under varying memory window sizes. Also, a novel fractional-order chaotic system (FOCS) is simulated, and its bifurcation diagrams have been analyzed. Demonstrating higher throughput and lower resource usage, the research emphasizes that the FPGA implementation provides a flexible and efficient platform for implementing complex systems like FOCS, suitable for secure communications and other high-speed applications while comparing with existing literature.

In [48], a core design of the novel fractional-order multi-scroll chaotic system that offers complex chaotic behaviors has been introduced. The study investigates how system parameters affect behavior, providing bifurcation diagrams and discussing the maximum Lyapunov exponent (MLE) for both integer and fractional domains, indicating richer chaos in the fractional domain. The core block design, represented in Figure 6, offers complex behaviors due to added parameters. In the paper, the authors also present a novel and automated FPGA design tool to optimize system implementation and allow developers to design configurable modules for different applications easily. The tool is tested on Xilinx's Virtex-5 FPGA and validated using MATLAB, with the results demonstrating the design methodology's effectiveness. The paper provides a procedure to generalize the tool for other chaotic systems, contributing to the field of information security, encryption, and IoT applications by enabling the rapid design of complex, high-performance chaotic systems on FPGA platforms. In Figure 5, the authors demonstrated the FPGA design for the Grunwald-Letnikov (GL) block and illustrated its functionality with an example of three inputs (X_k) by initializing the parameters d2, d1, and d0 to zero. The proposed design needs n-cycles for n-inputs as shown in the figure. Finally, the proposed design in Figure 6(a) shows the GL of x, y, and z. The k-generator in Figure 6(b) is solely a clock divider with a specified threshold counter. Figure 6(c) shows how the selection mode block generates logic 0 and 1 outputs, respectively, corresponding to variance in the k parameter values for each shape (V, Hat, and X-shape).

The FPGA implementations for two nonlinear time-delay equations: the Johnson and Moon equation, used to model electromechanical system vibrations, and the Mackey-Glass equation, applicable to biological systems have been represented in [49]. It offers a detailed stability analysis for these models and employs a 32-bit floating-point precision Runge-Kutta method for the computations. The FPGA simulations are shown to align with numerical simulations, underlining the method's accuracy for time-delay systems. This research demonstrates the potential of FPGA-based modeling in scientific and engineering fields where precise time-delay dynamics are crucial. Another research in [50] also investigates the impact of numerical solution accuracy on the digital implementation of differential chaos generators. Implementing four chaotic systems on an FPGA using Euler, mid-point, and Runge-Kutta methods, the study compares these systems based on resource usage, throughput, and chaotic behavior indicators like Lyapunov exponents. It concludes that simpler numerical solution techniques, such as the Euler method, offer better chaotic response and higher throughput when compared to more complex methods like midpoint and Runge-Kutta fourth-order techniques.

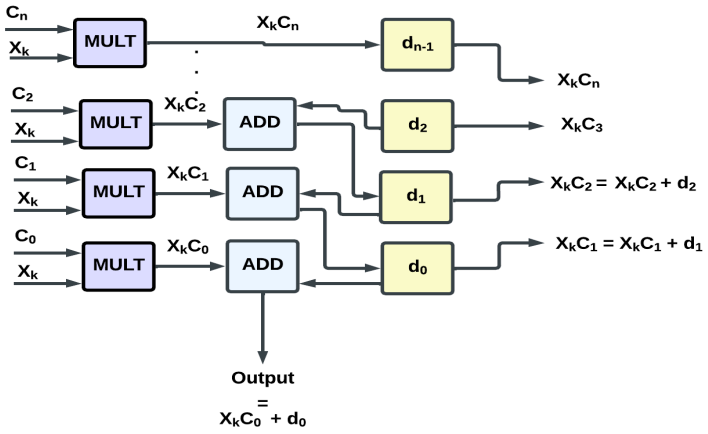

Figure 5. Proposed design of the Grunwald-Letnikov (GL) block with an example for three inputs (X_k), (adopted from [42]).

The Chaotic systems also have tremendous potential for modeling critical real-world systems. The work done in [51] presents efficient FPGA implementation of chaotic models for epidemic spread and disease progression on FPGA platforms. The research focuses on Ebola, Influenza, and COVID-19 virus spreading models, plus a cancer disease progress model. Despite their complexity due to a large number of parameters and arithmetic operations, these models are numerically analyzed for parameter sensitivity and successfully executed on two different FPGA platforms, paving the way for real-time control of chaotic dynamics in disease propagation. This work proves the feasibility of real-time simulation, control, and prediction of complex biological systems on FPGA, a step forward for applications in predictive health management.

Another such example can be found in [52], which focuses on a high-level synthesis (HLS) approach to algorithm development for evaluating tsunami wave dangers using FPGA technology. The paper discusses a specialized calculator designed to quickly simulate tsunami wave propagation by applying the MacCormack finite-difference scheme. The developed software is tested for precision and shows strong agreement with exact solutions, highlighting its capability to evaluate tsunami risks in a matter of minutes following seismic events. This rapid assessment tool could be crucial for early warning systems in coastal regions. The research demonstrates the application of HLS in creating efficient computational pipelines for numerical solutions of complex equations on FPGAs, which are critical for time-sensitive simulations like tsunami wave propagation.

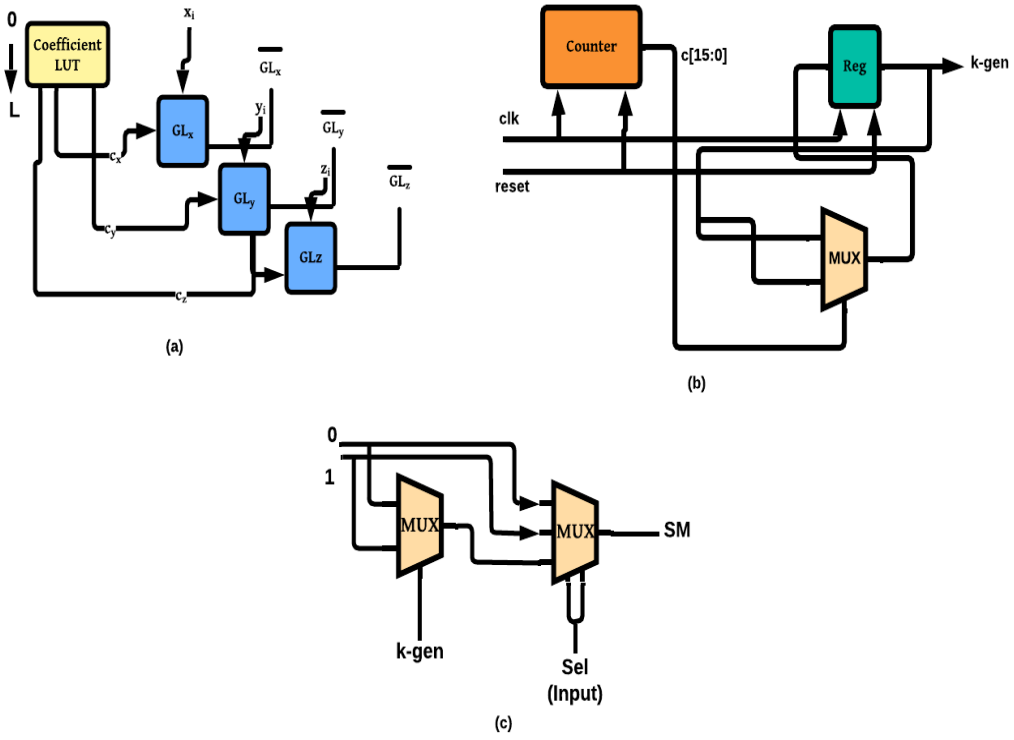

Figure 6. Core block design (adopted from [42]), is divided into three parts: (a) the Grunwald-Letnikov (GL) block LUT, (b) the k-generator, and (c) selection mode.

Table I represents a summary that is based on the types of accelerator, their characteristics and the results of experiments. The table relies on the acceleration of higher order nonlinear systems. It offers details about different kinds of accelerators, their traits or characteristics and also shows results from experiments performed with these methods.

TABLE I. A summary based on the types of accelerator, their characteristics and experimental results that relies on the hardware implementation of higher-order nonlinear systems.

HARDWARE ACCELERATOR	IMPLEMENTATION TYPES	CHARACTERISTICS
Chaotic Oscillator System [41]	FPGA	The system's dynamism is validated through the Lyapunov exponents spectrum and bifurcation diagram computations, Multigrid frameworks.
Chaotic Systems [43]	FPGA	Resource-intensive, Customized architecture, Random bit generators, digital signal processing.
Lü-Chen Chaotic System [44]	FPGA	Compatibility with fixed-point standards, Heun numerical algorithm, Chaos-based embedded systems, cryptography, secure communication.
Time-Delayed Chameleon System [45]	FPGA	Multiple chaotic flows, Adaptive modified functional projective lag synchronization algorithm, Secure communications, synchronization techniques.
Two-Dimensional Rotation System [46]	FPGA	Preservation of chaotic dynamics,Chaos-based encryption, security enhancements.
Fractional-Order Chaotic Systems [47]	FPGA	Modified product integration rules, Fractional-order chaotic systems, high throughput, lower resource usage,Secure communications, high-speed applications, fractional-order systems.
Fractional-Order Multi-Scroll Chaotic System [48]	FPGA	Information security, encryption,automated FPGA design tool, Complex chaotic behaviors, IoT applications.
Nonlinear Time-Delay Equations [49]	FPGA	Scientific modeling, 32-bit floating-point precision Runge-Kutta method,Stability analysis, engineering applications, time-delay systems. .
Differential Chaos Generators [50]	FPGA	Euler, midpoint, and Runge-Kutta methods, resource usage, throughput, Lyapunov exponents, Chaos generation, resource optimization, high-speed applications.
Epidemic Spread and Disease Progression Model [51]	FPGA	Epidemic spread models, disease progression models,Predictive health management, disease propagation, real-time simulation.
Tsunami Wave Propagation Simulator [52]	FPGA	MacCormack finite-difference scheme,Early warning systems, coastal region risk assessment, tsunami simulations.

VI. ACCELERATION OF SCIENTIFIC COMPUTING WITH GPU

General-purpose graphics processing units (GPUGPUs) have been instrumental in scientific computation due to their capabilities for parallel computing. The researchers in [53] investigate the potential of graphics processing units (GPUs) for speeding up scientific calculations, specially for Monte Carlo simulations of classical spin models in statistical mechanics. The authors present a significant performance gain, up to a factor of 1000, over serial CPU code when simulations are tailored to the GPU architecture. Experimental findings indicate that careful algorithm optimization for specific GPU architectures can yield significant speed-ups, often exceeding two orders of magnitude, emphasizing the importance of GPUs in the future of computational science. The paper also details examples such as simulations of the Ising and Heisenberg models and parallel tempering, along with an argument that the efficiency of GPU simulations depends on algorithmic adjustments to match the architecture.

Advanced computational strategies for fluid dynamics simulations have been outlined in [54].The study suggests that optimizing the DLR TAU code by profiling the most time-consuming processes, testing new partitioning algorithms, and exploring hardware acceleration, particularly using FPGAs and GPUs is a crucial component for performance acceleration. Simulations for computational fluid dynamics (CFDs) have been readily ported to the GPGPU paradigm. Methods for improving an unstructured grid solver's efficiency on modern graphics hardware are proposed in [55], focusing on the three-dimensional Euler equations for inviscid, compressible flow. The performance of the solver has been demonstrated based on two benchmark cases, such as a National Advisory Committee for Aeronautics wing (NACA0012) and a missile. Implementing these techniques yielded an average speed-up factor of roughly 9.5×over parallelized

OpenMP code running on a quad-core CPU and about 33× over the equivalent serial code, suggesting potential benefits for computational fluid dynamics problem-solving. The paper concludes that GPUs, with careful memory and computation management, can deliver substantial performance improvements for unstructured grid-based CFD solvers, although double-precision performance gains are less pronounced. Future GPUs are anticipated to enhance double-precision performance further, and the techniques presented are expected to be relevant for OpenCL codes. The study sets the stage for more advanced solvers with features like fourth-order damping and flux limiters.

In general, software tools for fluid dynamics modeling, analysis, and simulation are largely available to the community [56]. The highly parallel structure of GPUs has been leveraged in [57] for enhancing combustion modeling, demonstrating substantial simulation acceleration and predictive capability improvements. The authors demonstrate substantial acceleration of combustion simulations without sacrificing the accuracy of detailed chemistry. This is achieved by implementing GPU-enhanced algorithms for reaction rate evaluation and the integration of ordinary differential equations (ODEs), specifically matrix factorizations, yielding more than tenfold speed increases over CPU-only simulations. The computational time is shown to scale less than quadratically with the number of species involved when GPUs are utilized, compared to the super-quadratic scaling seen with CPUs. The paper concludes that GPUs will play a crucial role in the future of combustion modeling, allowing for the practical application of increasingly large reaction mechanisms. The authors also provide practical guidance for transitioning from CPU-only to GPU-enhanced combustion modeling, underscoring the potential for GPUs to significantly advance the field of combustion science. The GPU-enhanced algorithms significantly outperform CPU-only simulations, with computational time scaling less favorably on CPUs, and GPUs are expected to play an increasingly important role in future combustion modeling. Although not explored at length, there are some prior works focused on porting generalized multigrid solvers to GPUs [58]. A multigrid solver implemented on GPU hardware to efficiently solve boundary value problems across various applications such as simulation and image processing. Leveraging the GPU's shaders, the authors achieve significant computational speed-ups by conducting all operations on the graphics card, avoiding the bottleneck of data transfer to and from the CPU has been detailed in the paper. While highlighting the GPU's potential as a computational tool, the paper also acknowledges current limitations in memory bandwidth and precision, suggesting future advancements in graphics hardware could further enhance its applicability in scientific computing.

Table II represents a summary that is based on the types of accelerator, their characteristics and the results of experiments. The table relies on the acceleration of scientific computing with GPU and GPGPU. In this table, "Type" signifies different types of accelerators that can be used in scientific computing. The "Characteristics" column explains their unique features and "Experimental Results" tells about how effective they are when applied to various tasks. The information in Table III comes from a source that discusses the use of GPU and GPGPU for speeding up scientific calculations. It offers details about different kinds of accelerators, their traits or characteristics and also shows results from experiments performed with these methods.

VII. ACCELERATION OF SCIENTIFIC COMPUTING WITH CUSTOMIZED CO-PROCESSORS

The work done in this area cuts across multiple problem domains. The work in [59] employs an FPGA-based system to simulate two-dimensional Ising lattices, achieving considerable speed improvements over single CPU, single GPU, and prior FPGA systems. A significant speedup in Monte Carlo simulations of the Ising model using FPGA technology has been represented. The authors developed a system capable of updating multiple spins per FPGA clock cycle, resulting in a linear increase in simulation time with lattice size. They designed a hybrid random number generator that combines pseudorandom and true random generators, which demonstrated superior statistical

TABLE II. A summary based on the types of accelerator, their characteristics and experimental results that relies on speeding up scientific computing with GPU and GPGPU.

HARDWARE ACCELERATOR	IMPLEMENTATION TYPES	CHARACTERISTICS	EXPERIMENTAL RESULTS
Monte-Carlo simulations [53]	GPU	Up to 1000x speed-up over serial CPU code, algorithm optimization for GPU architecture.	Scientific calculations, Ising and Heisenberg models, Parallel tempering.
Optimization of DLR TAU Code [54]	GPU	Computational fluid dynamics (CFD) simulations.	Profiling, new partitioning algorithms.
Unstructured Grid Solver [55]	GPU	Speed-up factor of 9.5x over OpenMP on CPU, 33x over serial code, potential for CFD problem-solving.	Unstructured grid-based CFD solvers.
Combustion Modeling [57]	GPU	Combustion modeling, reaction rate evaluation, ODE integration.	More than tenfold speed increases over CPU-only simulations, practical guidance for GPU-enhanced modeling
Multigrid Solvers [58]	GPU	Efficient solution of boundary value problems, significant computational speed-ups.	Simulation, image processing, multigrid solvers.

quality and helped overcome correlation issues affecting Monte Carlo simulations. Their FPGA-based system provided considerable speedups over CPU and GPU systems and showed potential for real-time simulations, especially near critical phase transitions in the Ising model. The paper concludes with an affirmation of FPGAs as valuable additions to scientific computing platforms, particularly for Monte Carlo simulations. The authors of [60] propose a methodology for optimizing FPGA implementations of digital signal processing (DSP) circuits. The authors propose a methodology for optimizing FPGA implementations of digital signal processing circuits. It introduces a design exploration process that combines module selection and resource sharing within pipeline scheduling to meet throughput constraints while minimizing hardware area. The authors present two algorithms for exploring this design space, demonstrating that integrated application of these techniques can achieve significant area savings over using either technique independently. This approach aids in automating the generation of high-performance FPGA circuits from high-level descriptions, addressing the growing gap between design productivity and chip complexity. The paper underscores the potential for using these methods to efficiently map computations onto FPGAs, improving performance and reducing design complexity.

A novel framework for the real-time evaluation of one-dimensional computational fluid dynamics (1D-CFD) models is presented in [61], deployed on a field-programmable gate array (FPGA) to enhance fuel pressure estimation in diesel engines. The work is particularly applicable in designing fuel systems for diesel engines. The real-time model improves fuel pressure estimation and contributes to more efficient engine operation. The paper describes how each segment of the interconnected pipes is partitioned into sub-volumes that compute pressure and flow rates, with calculations performed every time step. Precision Timed (PRET) processor cores are used for synchronization to meet real-time requirements. The paper demonstrates the feasibility and scalability of this approach, which allows for real-time 1D CFD modeling on FPGAs, with potential benefits in automotive engineering and other domains requiring real-time fluid dynamics analysis. In [62], an FPGA-based system designed to accelerate iterative linear-equation solvers with low memory bandwidth has been evaluated. The system uses a pipelining approach over successive iterations, showing nearly linear speedup and outperforming microprocessor-based software execution. The study addresses the performance limitations faced by general-purpose microprocessors due to memory bandwidth constraints in high-performance computing (HPC) applications. By designing a custom FPGA-based

accelerator, the research demonstrates scalable and efficient performance for HPC tasks, such as CFD simulations, which typically suffer from low memory bandwidth on traditional computing systems.

The Johnson and Moon equation and the Mackey-Glass equation, both nonlinear time-delay systems, have been implemented on an FPGA [49]. Using the fourth-order Runge-Kutta method and 32-bit IEEE-754-1985 floating-point standards, the systems are implemented on a Xilinx Virtex-6 FPGA, with maximum frequencies of around 432 MHz and results that align with numerical simulations. Alongside the implementation, the numerical algorithms themselves are also major contributors to performance. A novel L-stable method for numerically solving ordinary differential equations (ODEs) has been outlined in [63]. It introduces a stabilized fourth-order L-stable method that addresses the accuracy and stability in the numerical integration of ODEs, particularly stiff equations. The method proposed extends upon the A-stable two-step methods and is designed to achieve higher-order convergence compared to existing methods. Numerical results are presented for three test problems, demonstrating that the proposed method provides a more accurate solution when compared to some existing A-stable and L-stable methods, including the Runge-Kutta fourth-order method. The paper suggests that the new method is well-suited for practical applications due to its enhanced stability and convergence properties. The findings from FPGA-based deployment in [54] show significant computational gains from optimization, superior scalability with the graph partitioner algorithm, and promising results from a mixed hardware-software computation platform for simulations. Computational fluid mechanics problems are a prime candidate for customized co-processor-based acceleration, especially in the aerospace domain.

The authors of [64] introduce a novel hardware solver based on the open-source RISC-V instruction set architecture for solving ordinary differential equations (ODEs). The study designs reconfigurable co-processors that implement Euler and Runge-Kutta numerical methods for ODE solving. The coprocessor achieves a remarkable 4.8x performance improvement in the best-case scenario, with only a marginal increase of 13.3 percent in hardware resources and 8.1 percent in additional power dissipation. Another study in [65] discusses a hardware accelerator based on second-order, third-order, and fourth-order Runge-Kutta ODE solvers. The experimental results demonstrate the accelerator has been deployed on the Zynq ZC702 FPGA evaluation board. A comprehensive comparative analysis of different Runge-Kutta hardware accelerators, highlighting factors including timing performance, hardware resource usage, and the total on-chip power consumption has been given. According to the findings, the fourth-order Runge-Kutta ODE solver has a significant power consumption of an estimated 51.37 percent and 12.445 percent when compared with second-order and third-order Runge-Kutta solvers at a clock rate of 100 MHz, respectively. In [66], a design-space exploration for the half, single, and double floating-point precision formats at a clock frequency of up to 80 Mhzs has been presented intended for the Runge-Kutta ODE solvers. The accelerator for the fourth-order Runge-Kutta solver consumed the most power when compared with the second-order and third-order Runge Kutta ODE solvers. In [67], an FPGA-based hardware accelerator for the Euler and Modified Euler method for solving an ODE has been demonstrated. The Xilinx Vivado Power Analyzer tool has been used to determine the total on-chip power consumption. at a clock frequency of 2.85Mhz. According to the experimental results, the paper reveals that there's a slight increase of 0.001W in the modified Euler method after deploying on the ZYNQ - ZC702 FPGA Evaluation Board (xc7z020clg484-1). The paper also focuses on an in-detailed FPGA hardware resource summary for both the methods along with a timing analysis.

The researchers in [68] present the Chipyard framework, a comprehensive environment for SoC design, simulation, and implementation designed for specialized computing systems. Chipyard is a collection of generator-based, open-source, configurable IP (intellectual property) components that are meant to be used at different phases of the hardware development process, all the while maintaining consistency in design intent and integration. Chipyard enables fast ASIC (Application-Specific Integrated Circuit) development and cloud-hosted FPGA-accelerated simulation to enable ongoing validation of customized systems that are both physically and conceptually feasible. The application of Field Programmable Gate Arrays (FPGAs) in Computational Fluid Dynamics (CFD) to shorten computation times in aeronautics design has been explored in [69]. The authors present a methodology to develop an FPGA-based solution for the quasi-1D Euler equations. The authors address the challenges of long simulation times and propose a methodology using FPGA for hardware acceleration. The approach includes four steps: performance analysis of current CFD software, design and development of a CFD-specific hardware system architecture, development of specific hardware for time-consuming subroutines, and implementation of software for synchronization and communication between PC and hardware modules. The paper demonstrates the methodology on the Sod's Shock Tube problem, revealing that FPGA implementations can yield considerable speedups—up to two orders of magnitude—compared to CPU-based solutions. The study concludes that integrating FPGA into CFD

computations could be a significant advancement for the aeronautics industry, offering a new level of parallelism and optimization in simulation platforms. The challenge of adapting Computational Fluid Dynamics (CFD) software to efficiently utilize emerging multi/many-core, GPU, and FPGA architectures.

The problem of adapting large CFD software to new microprocessor designs is tackled in [70], focusing on accelerating computational kernels in Air Force CFD applications across various architectures. It demonstrates the effectiveness of hybrid architectures, allowing better mapping of algorithm requirements and hardware architecture and achieving significant speed-ups. It outlines the computational bottlenecks common to CFD tasks and suggests acceleration techniques, including the use of hybrid systems combining different architectures' strengths. The paper presents specific examples where such approaches significantly improve performance, indicating a promising future for heterogeneous computing in accelerating key CFD computations. An FPGA-based systolic architecture designed for computational fluid dynamics (CFD) simulations, showcasing enhanced computational speed and efficiency has been represented in [71]. The systolic array structure maximizes parallelism and optimizes resource utilization on the FPGA, offering a significant performance leap over traditional CPU-based systems. This approach is particularly adept at handling the regularity and locality demands of CFD computations. Benchmarked against standard CPUs, the FPGA implementation reveals a marked improvement in processing times for a 2D cavity flow problem, establishing a promising avenue for advanced CFD computing solutions.

In [72], the paradigm shift required for algorithm developers when using Field Programmable Gate Arrays (FPGAs) for efficient numerical solutions has been discussed. The authors focus on the advantages of FPGAs for solving partial differential equations (PDEs), highlighting three levels of parallelism: fine-grained, coarse-grained, and algorithm-level. The authors demonstrate these concepts through the solution of the 2D Laplace equation using a novel "Laplace Atom" which optimizes the algorithm's computational units. Their FPGA-based approach achieves significant speedups over traditional CPU solutions, showcasing the potential of FPGAs in high-performance computing for scientific applications. The paper concludes that mastering FPGA-based algorithm design can lead to breakthroughs in computational speed, offering hundreds to thousands of times faster performance than conventional methods.In [73], the authors examine the utilization of hybrid parallel programming models combining MPI and OpenMP for computational fluid dynamics (CFD) simulations using high-order methods. The authors highlight the benefits of such models, particularly the dual-level parallelism that exploits the hierarchical nature of modern supercomputing architectures. Through a coarse-grain approach, they aim to minimize OpenMP synchronization overhead and improve scalability for high-order methods like the spectral/hp element method.

The authors of [74] examine the use of Field Programmable Gate Arrays (FPGAs) to accelerate the numerical solutions of fluid dynamics equations, providing a solution speed up to 20 times faster than a conventional CPU in some cases without sacrificing data precision. The researchers acknowledge that further speed improvements could be achieved with more powerful reconfigurable hardware, lower precision data formats, and parallelizing several FPGAs to perform computations as a single unit. The authors compare the performance of FPGAs to conventional CPUs, showcasing instances where FPGA computations were up to 20 times faster without sacrificing precision. The research highlights the potential of FPGA's parallel processing capabilities and its system-level and hardware-level configurations. Numerical solutions for typical CFD problems such as the Laplace and 1-D Euler equations are implemented and evaluated, with the FPGA-based solutions showing significant time savings over CPU calculations. The study not only demonstrates the computational advantages of using FPGAs but also suggests a promising future for FPGA utilization in computational fluid dynamics. A mixed-precision linear solver for FPGAs, utilizing lower precision data formats for performance benefits while ensuring high computational accuracy, is demonstrated in [75]. It details the implementation and performance comparison with CPUs, marking it as the first work of its kind to implement a hardware architecture for the pivoting algorithm in a mixed-precision linear solver. Throughout existing literature, we note that innovation must be adopted both in software and hardware design; algorithm developers must undertake a paradigm shift to utilize FPGAs and reconfigurable computing, emphasizing three levels of parallelism: fine-grained, coarse-grained, and algorithm-level. The authors assert that once a developer overcomes the Von Neumann programming approach and embraces multi-level parallelism, reconfigurable computing can result in significantly faster and more efficient numerical solutions.

The authors of [76] discuss FPGA-based accelerators designed for solving ordinary differential equations (ODEs), demonstrating substantial speed improvements over CPU solutions. By focusing on the Lotka-Volterra model, the study presents configurable hardware accelerators that offer up to 14 times faster performance. It also considers the balance between computational efficiency and hardware resource utilization, setting the stage for further

optimization in high-performance computing applications. In [77], the authors explore the implementation of the V-cycle Multigrid algorithm for solving 2D Poisson's equation on FPGAs. The authors use Handel-C, a high-level hardware design language, and demonstrate that their hardware implementation outperforms both Jacobi and successive over-relaxation (SOR) iterative solvers in terms of execution speed when compared to a C++ software version running on general-purpose processors. The paper presents a detailed analysis of the performance using FPGA vendor tools, revealing substantial improvements in the speed of convergence and accuracy of the solution. They discuss the potential of FPGAs for solving partial differential equations efficiently due to their ability to exploit parallelism inherent in the Multigrid method. The authors conclude with the suggestion of future work including a pipelined version of the algorithm, transitioning to lower-level hardware description languages, and extending the application of Multigrid methods.

In [78], the authors provide an overview of explainable machine learning (ML) in the natural sciences, underscoring the importance of models that are not only accurate but also transparent, interpretable, and explainable. The paper emphasizes that explainable ML should incorporate domain knowledge for scientific consistency and highlights the role of interpretability in scientific discoveries. The paper reviews recent applications of ML in scientific domains and suggests that explainability is crucial for advancing scientific understanding. Two FPGA-based techniques for implementing fractional-order integrators and differentiators, aiming to better model dynamical systems have been described in [79]. The techniques focus on accuracy and efficient resource usage, with applications demonstrated in various systems. Experimental results show these methods offer significant improvements over previous approaches.

The benefits of SIMD many-core architectures for data-intensive applications, acknowledging challenges in fabrication cost and complexity has been outlined in [80]. It proposes leveraging advanced silicon integration, IP cores, and FPGAs to enhance design efficiency and performance. The work presents an IP-based design methodology for customizable SIMD processing on FPGA platforms, demonstrating the methodology's effectiveness and the SoC's performance through experimental results. A novel chaotic system characterized by a single parameter and an open curve of equilibrium points, implemented on an FPGA (Field-Programmable Gate Array) has been represented in [81]. The study explores the system's dynamics through bifurcation and Lyapunov exponent analysis. The FPGA implementation is detailed, showcasing the approach of the exponential function with a power series to simplify the hardware design. The system's application to secure communications is demonstrated by synchronizing two chaotic oscillators, with experimental results underscoring the effectiveness of the chaotic system in transmitting an image securely. The work concludes by affirming the utility of FPGAs in rapidly prototyping and testing dynamic systems for secure communication applications.

In [82], the sensitivity of chaotic systems to implementation details, which can lead to mismatches affecting chaos-based applications like cryptography has been examined. It identifies significant discrepancies between different software and hardware (FPGA) implementations of chaotic systems due to variances in precision and operation order. The study showcases how mismatches in software implementations using different precision levels and in hardware due to the order of multiplications can be exploited in encryption applications, proposing a new approach for pseudo-random number generation (PRNG) and image encryption. The authors demonstrate that despite these mismatches, chaotic systems still exhibit their fundamental properties, and the mismatch signals can provide a novel source of randomness for cryptographic applications, passing various performance metrics and statistical tests. The authors of [83] present an open-source high-level synthesis tool called LegUp. It enables the conversion of standard C programs into hybrid architectures, incorporating both an FPGA-based MIPS soft processor and custom hardware accelerators. The tool aims to harness the performance and energy benefits of hardware design while maintaining the ease of software development. The paper demonstrates that LegUp can generate hardware solutions that are comparable in quality to those from commercial tools and can significantly improve performance and reduce energy consumption in comparison to purely software-based implementations. This tool is positioned to make FPGA technology more accessible to software engineers and expand the FPGA user base, thus promoting the broader application of FPGA in various fields. Key themes include the need for efficient partitioning algorithms for parallel computations and the potential of heterogeneous computing architectures in enhancing CFD simulation speeds. The paper also details the challenges and benefits of implementing these computational strategies in aeronautics engineering, where accurate and fast simulations are critical for energy savings, cost reduction, and safety improvements.

In [84], the authors present an exploration of FPGA-based accelerators for computational fluid dynamics (CFD) codes, with a specific focus on the spectral element method (SEM) used in high-fidelity CFD systems. The authors

design a custom FPGA-based accelerator targeting both single and double precision for the SEM-based Pressure-Poisson solver, utilizing an analytical performance model based on the I/O cost of the algorithm. They evaluate the accelerator's performance in terms of speed and power consumption, comparing it to traditional CPU solutions. Moreover, they introduce a data movement-reducing technique by computing geometric factors on the fly, leading to a substantial reduction in runtime for local evaluations. The research reveals that FPGAs can achieve significant single-precision performance improvements, suggesting that mastering FPGA-based algorithm design can lead to breakthroughs in computational speed for CFD applications. The paper also discusses the challenges and opportunities of using reconfigurable architecture for future CFD applications, especially with the advent of emerging technologies. The study's findings open up possibilities for future research to further enhance FPGA-based CFD solutions. The simulation results solve the 2D Laplace equation, indicating that the hybrid model performs better in terms of scalability and computational efficiency on certain systems compared to pure MPI or OpenMP models. The study shows that the hybrid model's effectiveness hinges on several factors including the MPI library, the shared-memory system, and hardware specifics. Key findings include the hybrid model's superior performance with two threads per process on SGI Origin and IA64 clusters and its ability to counter-balance cost increases from p-type refinement by employing multiple threads. However, on IBM SP and Compaq Alpha clusters, pure MPI outperforms the hybrid model. Overall, the paper demonstrates the potential of hybrid programming models to improve the performance of high-order CFD methods on parallel computing platforms. In [85], enhancing iterative methods for solving equations in Banach spaces has been focused. Polyak introduces multi step iterative methods, providing convergence proofs and conditions for their efficiency over traditional methods. The paper meticulously lays out the theoretical foundations for these methods, including lemmas and theorems that guide the optimization of iterative processes. Polyak's work is a significant theoretical contribution to numerical analysis, offering practical solutions for achieving faster convergence in computational algorithms.

In [86], the authors outline the advancements and applications of the MEGAFLOW software system in aerodynamic simulations for aircraft design. This system incorporates the block-structured Navier-Stokes solver FLOWer and the unstructured/hybrid solver TAU, offering solutions to the three-dimensional Reynolds-averaged Navier-Stokes equations. The paper discusses the development, validation, and industrial application of these software components, emphasizing their integration into the design process for complex configurations in the aerospace industry. The document also covers the software's ability to predict viscous flows around intricate geometries, its use in shape optimization, and the potential for applications beyond aeronautics, such as ground transport vehicle aerodynamics. Overall, the MEGAFLOW system has become a cornerstone in German aerospace for designing new aircraft and optimizing aerodynamic performance. A high-performance FPGA-based solver for the 3D heat equation, aimed at improving non-destructive evaluation (NDE) techniques such as the detection of antipersonnel mines has been discussed in [87]. The solver uses an explicit finite-difference (FD) method, achieving a speedup factor of 34 over purely software solutions. The proposed system integrates software developed in C++ and hardware described in Handel-C and VHDL, enabling efficient simulation and processing times suitable for field use. The work is significant because it addresses the problem of detecting plastic antipersonnel mines which are challenging to find with traditional methods. The FPGA implementation not only demonstrates considerable computational speed improvements but also presents an integrated approach combining software flexibility with hardware acceleration. This study provides a valuable contribution to the field of NDE and showcases the capabilities of FPGA platforms in accelerating complex computational tasks.

In [88], the design and implementation of a hardware-based simulator for electron dynamics in semiconductors, focusing on the Poisson equation's parallel solution as part of a Monte Carlo method has been discussed. The authors propose a parallel approach using the finite differences method to solve the Poisson equation, which is traditionally the most time-consuming part of semiconductor device simulators. They compare this approach to the classical Gauss-Seidel iteration, highlighting the potential for substantial computational speed increases due to the parallel nature of the new method. This part of the study presents a detailed structure for the hardware simulator and explains the implementation of the parallel Poisson equation solver across a mesh of computational nodes. The results indicate that using a parallel processor matrix can significantly enhance the efficiency of the Gauss-Seidel algorithm for harmonic equations. The paper concludes with the authors' intentions to develop an FPGA solution, which could further improve the speed and efficiency of semiconductor device simulations, and identifies synchronization among parallel particle processors as an area for future research. A summary based on the types of the accelerator, their characteristics, and experimental results has been represented in Table III, depending on the acceleration of scientific computing with customized co-processors.

Table III. A summary based on the types of accelerator, their characteristics and experimental results that relies on speeding up scientific computing with customized co-processors.

HARDWARE ACCELERATOR	IMPLEMENTATION TYPES	CHARACTERISTICS	EXPERIMENTAL RESULTS
Ising Model Simulation [59]	FPGA	Considerable speed improvements over CPU/GPU, real-time potential.	Monte Carlo simulations of Ising model, real-time simulations near critical phase transitions.
Optimization of DSP Circuits [60]	FPGA	Area savings in FPGA implementations, optimization.	Digital signal processing (DSP) circuits optimization methodologies.
Real-time 1D-CFD Model [61]	FPGA	Improved fuel pressure estimation, real-time 1D CFD modeling.	Fuel system design for diesel engines, real-time fluid dynamics analysis.
Linear Equation Solver [62]	FPGA	Nearly linear speedup, outperforms microprocessor-based execution.	Acceleration of iterative linear-equation solvers with low memory bandwidth.
Johnson-Moon & Mackey-Glass Equations [49]	FPGA	High-frequency operation, results aligned with numerical simulations.	Nonlinear time-delay systems on FPGA, numerical algorithm optimization.
Stabilized L-stable Method for ODEs [63]	FPGA	Higher-order convergence, accuracy, and stability.	Numerical integration of stiff ordinary differential equations (ODEs).
Euler and Runge-Kutta Solver [64]	FPGA	Solving ordinary differential equations (ODEs).	The coprocessor achieves a remarkable 4.8x performance improvement in the best-case scenario, with only a marginal increase of 13.3 percent in hardware resources and 8.1 percent in additional power dissipation.
Runge-Kutta Solver [65, 66]	FPGA	Solving ordinary differential equations (ODEs).	The fourth-order Runge-Kutta ODE solver has a significant power consumption of an estimated 51.37 percent and 12.445 percent when compared with second-order and third-order Runge-Kutta solvers at a clock rate of 100 MHz. Also, a design-space exploration for the half, single, and double floating-point precision formats at a clock frequency of up to 80 Mhzs.

Euler and Modified Euler ODE solver [67]	FPGA	Solving ordinary differential equations (ODEs).	A slight increase of 0.001W in the modified Euler method after deploying on the ZYNQ - ZC702 FPGA Evaluation Board (xc7z020clg484-1) at a clock frequency of 2.85 Mhz. An in-detailed FPGA hardware resource summary for both the methods along with a timing analysis.
Quasi-1D Euler Equations [69]	FPGA	Acceleration of computational fluid dynamics (CFD) simulations.	Speedups up to two orders of magnitude over CPU-based solutions.
Hybrid Architectures for Accelerating CFD Kernels [70]	FPGA	Acceleration of computational fluid dynamics (CFD) tasks.	Significant speed-ups, acceleration techniques.
Systolic Architecture for CFD Simulations [71]	FPGA	Acceleration of computational fluid dynamics (CFD) simulations.	Enhanced computational speed and efficiency.
Algorithm Design for Efficient PDE Solutions [72]	FPGA	Efficient numerical solutions for partial differential equations (PDEs).	Significant speedups over traditional CPU solutions.
Hybrid Parallel Programming Models for CFD Simulations [73]	FPGA	Computational fluid dynamics (CFD) simulations using high-order methods.	Dual-level parallelism, improved scalability.
Numerical Solutions of Fluid Dynamics Equations [74]	FPGA	Acceleration of fluid dynamics equations solved using FPGAs.	20x faster speed than conventional CPU.
Mixed-Precision Linear Solver [75]	FPGA	Acceleration of linear solver for FPGAs.	High computational accuracy, significant speed improvements.

Lotka-Volterra Solver [76]	FPGA	Acceleration of ordinary differential equations (ODEs) solvers using FPGAs.	Configurable hardware accelerators that offer up to 14 times faster performance. Also considers the balance between computational efficiency and hardware resource utilization, setting the stage for further optimization in high-performance computing applications.
V-Cycle Multigrid Solver [77]	FPGA	Implementation of the V-cycle Multigrid algorithm for solving 2D Poisson's equation on FPGAs. Uses Handel-C, a high-level hardware design language.	Outperforms both Jacobi and successive over-relaxation (SOR) iterative solvers in terms of execution speed when compared to a C++ software version running on general-purpose processors.
Explainable Machine Learning in Natural Sciences [78]	FPGA	Applications in natural sciences, interpretability in scientific discoveries.	Transparent and interpretable models, domain knowledge integration.
Fractional-Order Integrators [79]	FPGA	Dynamical system modeling, fractional calculus.	Accuracy and efficient resource usage.
Customizable SIMD Processing [80]	FPGA	Enhancing design efficiency and performance in data-intensive applications.	IP-based design methodology, customizable SIMD processing.
Chaotic System Implementation [81]	FPGA	Chaotic systems for secure communication, FPGA-based dynamic system modeling.	Synchronization for secure communication, hardware simplification.
Sensitivity Analysis of Chaotic Systems [82]	FPGA	Cryptography, exploitation of chaos for secure communication.	Exploitation for pseudo-random number generation, image encryption.
LegUp: High-Level Synthesis Tool for Hybrid Architectures [83]	FPGA	Making FPGA technology more accessible, improving performance and energy efficiency.	Converting C programs to hybrid architectures, performance improvement.
Spectral Element Method [84]	FPGA	Acceleration of spectral element method (SEM) for computational fluid dynamics (CFD).	Single-precision performance improvements, data movement reduction.
Polyak's Multi-Step Iterative Methods [85]	FPGA	Enhancing iterative methods for solving equations in Banach spaces.	Faster convergence in iterative algorithms.

MEGAFLOW Software System for Aerodynamic Simulations [86]	FPGA	Aerodynamic simulations for aircraft design.	Predicting viscous flows, shape optimization, industrial applications.
3D Heat Equation [87]	FPGA	Accelerating non-destructive evaluation (NDE) techniques using FPGA-based solvers.	Speedup factor of 34 over software solutions, NDE applications.
Parallel Poisson Equation Solver [88]	FPGA	Semiconductor device simulations, electron dynamics modeling.	Parallel solution, speed improvement over classical algorithms.

VIII. SUMMARY

Scientific computing is now a fundamental instrument in current research across many different areas. The world we live in today has experienced a massive increase in data creation from multiple sources, like scientific experiments, simulations, and observational studies. Luckily, progress made with computer hardware and software helps satisfy the requirements of scientific computing in today's world. The design of specialized systems to speed up the simulation of dynamical systems isn't a novel issue; it has been examined and solutions have been created, at least since the early 1990s. However, these design processes are super-specialized, costly, and extremely hard to integrate with common computing frameworks or programming environments. In addition, targeted systems do not possess strong enough characteristics to speed up the simulation of dynamical systems that they were not designed for. This creates a very high development cost and limits the use of high-performance computing infrastructure. In this survey, we have discussed accelerators in depth, adopting a holistic perspective. We presented many aspects by proposing a taxonomy organized into five macro-categories: implementation methods, implementation types, host coupling, cost factors, and applications. This literature survey provides a comprehensive analysis of the convergence between these areas in implementing the hardware accelerator in the scientific computing domain. This review will also make it possible for the researchers to choose domain-specific computing systems that are modular, fast, efficient, and optimally targeted towards any given dynamical system.

One of the primary methods for handling scientific computing requirements is high-performance computing (HPC). HPC systems have been developed to carry out complicated calculations and simulations in a rapid and effective manner. The main feature is that they consist of many interconnected processors that work together to solve large-scale problems. Simulations in areas like physics, chemistry, and biology regularly employ these systems. This is how we meet the needs for scientific computing. These systems are often used for simulations in fields such as physics, chemistry, biology, astronomy, genetics, and climate science. Cloud computing is a method to gain entry over the internet into computing resources. Researchers who do not have their own HPC system or find it difficult to maintain can use cloud computing. Cloud computing providers offer a range of services, including virtual machines, storage, and software tools, that can be used for scientific computing. Scientific computing needs are being met in today's world through a combination of high-performance computing systems, cloud computing, specialized software tools, and data management tools [90]. These tools and resources provide researchers with the ability to analyze large datasets, perform complex simulations, and collaborate with other researchers worldwide. As technology continues to advance, scientific computing will continue to play a crucial role in modern research.

In this survey, we have explored various applications of hardware accelerators within the domain of scientific computing. Specifically, we concentrated on Field-Programmable Gate Arrays (FPGAs) and Graphics Processing Units (GPUs) and these accelerators promise notable speed improvements compared to traditional CPU methods. They hold potential for performing simulations instantly and making substantial progress in many computational duties [53, 55]. Moreover, FPGA-based solutions have shown the effectiveness in real-time computational fluid dynamics (CFD) modeling [61]. Also, implementing FPGAs for solving linear equations and ordinary differential equations (ODEs) has shown nearly linear speedups as well as significant performance enhancements when

compared with executions on microprocessors. This shows their usefulness in speeding up iterative solvers and numerical integration tasks. Also, modeling of complex dynamical systems like Johnson-Moon and Mackey-Glass equations [49] have been done using FPGA architectures. They show high-frequency operation that matches well with numerical simulations. These systems include stabilization methods for stiff ODEs as well as efficient implementations for Euler and Runge-Kutta solvers which give rise to higher-order convergence along with accuracy and stability advantages while providing noteworthy performance enhancements. Likewise, just like how FPGA implementations have progressed, GPU-based accelerators are also showing notable development in scientific computing. For example, Monte Carlo simulations and computational fluid dynamics optimizations on GPUs are delivering impressive speed-ups compared to CPU-only implementations. These advancements open up fresh areas for algorithm optimization as well as parallel tempering techniques. In addition, GPU-based solvers for unstructured grids along with multigrid methods show significant improvements in computational speed. This makes solving boundary value problems and handling image processing jobs faster. These enhancements are setting up the ground for practical guidance in modeling and simulation that uses GPUs across different areas like combustion modeling, reaction rate evaluation as well as multigrid solvers. Also, the study of chaotic systems on FPGA architectures highlights their resource-intensive nature and the requirement of customized architectures.

IX. CONCLUSION

This systematic survey has classified three different areas of hardware acceleration in scientific computing in Sections V, VI, and VII. The classifications are as follows: a. hardware acceleration in terms of higher-order nonlinear equations; b. commercial off-the-shelf hardware acceleration in scientific computing; and c. customized co-processors for scientific computing. Using this concept, for scientists, researchers, or any professionals working within this domain, this survey emphasizes the effectiveness of using ensemble approaches to predict the suitable accelerator for their research. The parts of the survey contain many elements on how to construct hardware acceleration for a particular area of scientific computing. This survey also represents three different tables based on the analysis, that will shorten the topics. To conclude, this survey provides a brief overview of hardware accelerators in various scientific computing areas. However, there are some limitations to consider for this work. It was not feasible to examine every paper published on this subject within the chosen time frame due to timing constraints. Conversely, the predictions for the five distinct macro-categories of the taxonomy of hardware accelerators could possess more classified assumptions. The future ways that the research community will suggest make it all more interesting as we move towards a world where hardware acceleration is the key to success for better performance in the scientific computing domain.

Funding

Not Applicable

Institutional Review Board Statement

Not Applicable.

Informed Consent Statement

Not Applicable.

Data Availability Statement

Not Applicable.

Conflicts of Interest

The authors declare no conflict of interest.

REFERENCES

[1] W. Haensch, E.J. Nowak, R.H. Dennard, P.M. Solomon, A. Bryant, O.H.Dokumaci, A. Kumar, X. Wang, J.B. Johnson, M.V. Fischetti, Silicon CMOS devices beyond scaling, IBM J. Res. Dev. 50 (4.5) (2006) 339–361.

[2] M. Bohr, A 30-year retrospective on Dennard's MOSFET scaling paper, IEEE Solid-State Circuits Soc. Newslett. 12 (1) (2007) 11–13.

[3] H. Esmaeilzadeh et al., "Dark Silicon and the End of Multicore Scaling," in 2011 38th Annual International Symposium on Computer Architecture, pp.365–376.

[4] J. Cong, V. Sarkar, G. Reinman and A. Bui, "Customizable Domain-Specific Computing," in IEEE Design & Test of Computers, vol. 28, no. 2, pp. 6-15, March-April 2011, doi: 10.1109/MDT.2010.141.

[5] D.A. Patterson, J.L. Hennessy, Computer Organization and Design RISC-V Edition: The Hardware Software Interface, Elsevier Science, 2017.

[6] M. Zahran, "Heterogeneous computing: Here to stay," Queue, vol. 14, no. 6, pp. 31–42, 2016.

[7] M. Ravi, A. Sewa, S. T.G. and S. S. S. Sanagapati, "FPGA as a Hardware Accelerator for Computation Intensive Maximum Likelihood Expectation Maximization Medical Image Reconstruction Algorithm," in IEEE Access, vol. 7, pp. 111727-111735, 2019.

[8] A. Krishnakumar, U. Ogras, R. Marculescu, M. Kishinevsky, and T. Mudge, "Domain-Specific Architectures: Research Problems and Promising Approaches," ACM Transactions on Embedded Computing Systems, vol. 22, no. 2, Article 28, March 2023, pp. 1-26. DOI: 10.1145/3563946.

[9] J. Cong et al., "CHARM: A Composable Heterogeneous Accelerator-Rich Microprocessor," in Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design, pp. 379–384.

[10] W.J. Dally, Y. Turakhia, and S. Han, "Domain-Specific Hardware Accelerators," Communications of the ACM, vol. 63, no. 7, pp. 48–57, 2020.

[11] B. Reagen, Y.S. Shao, G-Y. Wei, and D. Brooks, "Quantifying Acceleration: Power/Performance Tradeoffs of Application Kernels in Hardware," in International Symposium on Low Power Electronics and Design (2013), pp.395–400.

[12] T. M. Shah, "Analysis of the Multigrid Method," Ph.D. thesis, Oxford University, 1989.

[13] M. T. Heath, "Section 11.5.7 Multigrid Methods," in "Scientific Computing: An Introductory Survey," McGraw-Hill Higher Education, 2002, pp. 478, ISBN: 978-0-07-112229-0.

[14] Y. Zhu and A. C. Cangellaris, "Multigrid finite element methods for electromagnetic field modeling," Wiley, 2006, pp. 132, ISBN: 978-0-471-74110-7.

[15] P. Sharma and V. Jadhao, "Molecular Dynamics Simulations on Cloud Computing and Machine Learning Platforms," 2021 IEEE 14th International Conference on Cloud Computing (CLOUD), Chicago, IL, USA, 2021, pp.751-753, doi: 10.1109/CLOUD53861.2021.00101.

[16] S. Farrell et al., "MLPerf™ HPC: A Holistic Benchmark Suite for Scientific Machine Learning on HPC Systems," 2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC), St. Louis, MO, USA, 2021, pp. 33-45, doi:10.1109/MLHPC54614.2021.00009.

[17] M. Book, M. Riedel, H. Neukirchen and E. Erlingsson, "Facilitating Collaboration in Machine Learning and High-Performance Computing Projects with an Interaction Room," 2022 IEEE 18th International Conference on e-Science (e-Science), Salt Lake City, UT, USA, 2022, pp. 529-538, doi: 10.1109/eScience55777.2022.00093.

[18] Hassija, V., Chamola, V., Mahapatra, A. *et al.* Interpreting Black-Box Models: A Review on Explainable Artificial Intelligence. *Cogn Comput* 16, 45–74 (2024).

[19] M. Harris, "Fast Fluid Dynamics Simulation on the GPU," in GPU Gems 1, pp. 637–665, 2004.

[20] J. P. Hayes, "Introduction to Stochastic Computing and Its Challenges," in DAC, 2015.

[21] Ascher, Uri M.; Petzold, Linda R. (1998). *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Philadelphia: Society for Industrial and Applied Mathematics. ISBN 978-0-89871-412-8.

[22] DEVRIES, Paul L.; HASBUN, Javier E. A first course in computational physics. Second edition. Jones and Bartlett Publishers: 2011. p. 215.

[23] W. H.; Teukolsky, S. A.; Vetterling, W. T.; Flannery, B. P. (2007). "Section 20.6. Multigrid Methods for Boundary Value Problems". *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.

[24] R. Piché, "An L-stable Rosenbrock Method for Step-by-Step Time Integration in Structural Dynamics," *IEEE Access*, vol. 126, no. 3–4, pp. 343-354, 1995, ISSN 0045-7825, https://doi.org/10.1016/0045-7825(95)00823-J.

[25] Leber, Christian; Geib, Benjamin; Litz, Heiner (September 2011). *High Frequency Trading Acceleration Using FPGAs*. International Conference on Field Programmable Logic and Applications. IEEE. doi:10.1109/FPL.2011.64.

[26] Peddie, Jon (1 January 2023). *The History of the GPU – New Developments*. Springer Nature. ISBN 978-3-03-114047-1. OCLC 1356877844.

[27] J. W. Brown and R. V. Churchill, "Complex variables and applications,"McGraw-Hill, 2009.

[28] E. Kreyszig, "Advanced Engineering Mathematics," John Wiley Sons, 2010.

[29] Ming Li, "A numerical simulation algorithm for non-Markovian dynamics in open quantum systems," Proceeding of the 11th World Congress on Intelligent Control and Automation, Shenyang, 2014, pp. 1252-1256.

[30] L. Scialaqua, L. J. Foged and C. J. Reddy, "Measurements as Enhancement of Numerical Simulation for Antenna Placement on a Rotocraft," 2021 International Applied Computational Electromagnetics Society Symposium (ACES), Hamilton, ON, Canada, 2021, pp. 1-3.

[31] W. L. Briggs, V. E. Henson, and S. F. McCormick, "A Multigrid Tutorial, Second Edition,"IEEE Access, vol. 8, pp. 100228-100229, 2020, doi: 10.1109/ACCESS.2020.3003091.

[32] D. Mira, E. J. Pérez-Sánchez, R. Borrell, and G. Houzeaux, "HPC-enabling technologies for high-fidelity combustion simulations," Proceedings of the Combustion Institute, vol. 39, no. 4, pp. 5091-5125, 2023.

[33] A. Auweter, A. Bode, M. Brehm, H. Huber, and D. Kranzlmüller, "Principles of Energy Efficiency in High Performance Computing," in Information and Communication on Technology for the Fight against Global Warming, D. Kranzlmüller and A. Min Toja, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 18-25.

[34] B. Jamroz and P. Mullowney, "Performance of Parallel Sparse Matrix-Vector Multiplications in Linear Solves on Multiple GPUs," 2012 Symposium on Application Accelerators in High Performance Computing, Argonne, IL, USA, 2012, pp. 149-152.

[35] A. V. Knyazev and K. Neymeyr, "Efficient solution of symmetric eigen-value problems using multigrid preconditioners in the locally optimal block conjugate gradient method," Electronic Transactions on Numerical Analysis, vol. 15, pp. 38–55, 2003.

[36] J. Xu, "Theory of multilevel methods," vol. 8924558, Ithaca, NY, Cornell University, 1989.

[37] J. H. Bramble, J. E. Pasciak, and J. Xu, "Parallel multilevel preconditioners," Mathematics of Computation, vol. 55, no. 191, pp. 1–22, 1990.

[38] J. Bolz, I. Farmer, E. Grinspun, and P. Schroeder. Sparse matrix solvers on the GPU: conjugate gradients and multigrid. In ACM Transactions on Graphics (TOG), volume 22. ACM, 2003, 917–924.

[39] Grossauer, H., & Thoman, P. (2008). GPU-based multigrid: Real-time performance in high-resolution nonlinear image processing. In Computer Vision Systems: 6th International Conference, ICVS 2008 Santorini, Greece, May 12-15, 2008 Proceedings 6 (pp. 141-150). Springer Berlin Heidelberg.

[40] P. Thoman, "GPGPU-based multigrid methods," na, 2007.

[41] Trottenberg, U., Oosterlee, C. W., & Schuller, A. (2000). Multigrid. Elsevier.

[42] N. Kuznetsov and V. Reitmann, Attractor Dimension Estimates for Dynamical Systems: Theory and Computation. Cham: Springer, 2020.

[43] Bonny, T., & Elwakil, A. S. (2018). FPGA realizations of high-speed switching-type chaotic oscillators using compact VHDL codes. Nonlinear Dynamics, 93, 819-833.

[44] Tuna, M., Alçın, M., Koyuncu, ˙I., Fidan, C. B., & Pehlivan, ˙I. (2019). High speed FPGA-based chaotic oscillator design. Microprocessors and Microsystems, 66, 72-80.

[45] Rajagopal, K., Jafari, S., & Laarem, G. (2017). Time-delayed chameleon: Analysis, synchronization and FPGA implementation. Pramana, 89(6), 92.

[46] Sayed, W. S., Radwan, A. G., Elnawawy, M., Orabi, H., Sagahyroon, A.,Aloul, F., ... & El-Sedeek, A. (2019). Two-dimensional rotation of chaotic attractors: Demonstrative examples and FPGA realization. Circuits, Systems, and Signal Processing, 38, 4890-4903.

[47] Abdelaty, A. M., Roshdy, M., Said, L. A., & Radwan, A. G. (2020). Numerical simulations and FPGA implementations of fractional-order systems based on product integration rules. IEEE Access, 8, 102093-102105.

[48] Soliman, N. S., Tolba, M. F., Said, L. A., Madian, A. H., & Radwan, A. G.(2019). Fractional X-shape controllable multi-scroll attractor with parameter effect and FPGA automatic design tool software. Chaos, Solitons & Fractals, 126, 292-307.

[49] Ngouabo, U. G., & Tchamdjeu, F. X. N. (2022). FPGA implementation of nonlinear equations with delay. Alexandria Engineering Journal, 61(8), 6237-6246.

[50] Zidan, M. A., Radwan, A. G., & Salama, K. N. (2011, December). The effect of numerical techniques on differential equation based chaotic generators. In ICM 2011 Proceeding (pp. 1-4). IEEE.

[51] Elnawawy, M., Aloul, F., Sagahyroon, A., Elwakil, A. S., Sayed, W. S.Said, L. A., ... & Radwan, A. G. (2021). FPGA realizations of chaotic epidemic and disease models including Covid-19. Ieee Access, 9, 21085-21093.

[52] Lavrentiev, M., Lysakov, K., Marchuk, A., Oblaukhov, K., & Shadrin, M.(2021). Algorithmic design of an FPGA-based calculator for fast evaluation of tsunami wave danger. Algorithms, 14(12), 343.

[53] Weigel, M. (2012). Performance potential for simulating spin models on GPU. Journal of Computational Physics, 231(8), 3064-3082.

[54] Andres, E., Widhalm, M., & Caloto, A. (2009, January). Achieving high-speed CFD simulations: Optimization, parallelization, and FPGA acceleration for the unstructured DLR TAU code. In 47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition (p.759).

[55] Corrigan, A., Camelli, F. F., Löhner, R., & Wallin, J. (2011). Running unstructured grid-based CFD solvers on modern graphics hardware. International Journal for Numerical Methods in Fluids, 66(2), 221-229.

[56] Kroll, N., Rossow, C. C., Schwamborn, D., Becker, K., & Heller, G. M. (2002, September). A numerical flow simulation tool for transport aircraft design. In ICAS 2002 Congress.

[57] Shi, Y., Green Jr, W. H., Wong, H. W., & Oluwole, O. O. (2011). Redesigning combustion modeling algorithms for the Graphics Processing Unit (GPU): Chemical kinetic rate evaluation and ordinary differential equation integration. Combustion and Flame, 158(5), 836-847.

[58] Goodnight, N., Woolley, C., Lewin, G., Luebke, D., & Humphreys, G.(2005). A multigrid solver for boundary value problems using programmable graphics hardware. In ACM SIGGRAPH 2005 Courses (pp. 193-es).

[59] Lin, Y., Wang, F., Zheng, X., Gao, H., & Zhang, L. (2013). Monte Carlo simulation of the Ising model on FPGA. Journal of computational Physics, 237, 224-234.

[60] Sun, W., Wirthlin, M. J., & Neuendorffer, S. (2007). FPGA pipeline synthesis design exploration using module selection and resource sharing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 26(2), 254-265.

[61] Liu, I., Lee, E. A., Viele, M., Wang, G., & Andrade, H. (2012, April). A heterogeneous architecture for evaluating real-time one-dimensional computational fluid dynamics on FPGAs. In 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines (pp. 125-132). IEEE.

[62] Sano, K., Hatsuda, Y., & Yamamoto, S. (2011, June). Performance evaluation of FPGA-based custom accelerators for iterative linear-equation solvers. In 20th AIAA Computational Fluid Dynamics Conference (p. 3223).

[63] Inc, M. (2007). New L-stable method for numerical solutions of ordinary differential equations. Applied mathematics and computation, 188(1), 779-785.

[64] Hollabough, A., & Chakraborty, D. (2022, May). An open-source co-processor for solving Lotka-Volterra equations. In 2022 IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 1690-1694). IEEE.

[65] S. Bhattacharya and D. Chakraborty, "Design and Analysis of a Hardware Accelerator with FPU-Based Runge-Kutta Solvers," *2023 International Conference on Electrical, Communication and Computer Engineering (ICECCE)*, Dubai, United Arab Emirates, 2023, pp. 1-6, doi: 10.1109/ICECCE61019.2023.10442325.

[66] S. Bhattacharya and D. Chakraborty, "Design-Space Exploration of the Runge-Kutta Hardware Accelerator for Solving Ordinary Differential Equation," *2023 IEEE International Conference on Electrical, Automation and Computer Engineering (ICEACE)*, Changchun, China, 2023, pp. 260-264, doi:10.1109/ICECCE61019.2023.10442 325.

[67] S. Bhattacharya and D. Chakraborty, "Implementation of a Hardware Accelerator with FPU-Based Euler and Modified Euler Solver for an Ordinary Differential Equation," in *10th Annual Conf. on Computational Science & Computational Intelligence (CSCI),* Las Vegas, NV, USA, 2023, (Forthcoming).

[68] Amid, A., Biancolin, D., Gonzalez, A., Grubb, D., Karandikar, S., Liew, H., ... & Nikoli´c, B. (2020). Chipyard: Integrated design, simulation, and implementation framework for custom socs. IEEE Micro, 40(4), 10-21.

[69] Andrés, E., Carreras, C., Caffarena, G., Molina, M. D. C., Nieto-Taladriz, O., & Palacios, F. (2008, January). A methodology for CFD acceleration through reconfigurable hardware. In the 46th AIAA Aerospace Sciences Meeting and Exhibit (p. 481).

[70] Dongarra, J., Peterson, G., Tomov, S., Allred, J., Natoli, V., & Richie, D. (2008, July). Exploring new architectures in accelerating CFD for Air Force applications. In 2008 DoD HPCMP Users Group Conference (pp. 472-478). IEEE.

[71] Sano, K., Iizuka, T., & Yamamoto, S. (2007, April). Systolic architecture for computational fluid dynamics on FPGAs. In the 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2007) (pp. 107-116). IEEE.

[72] Nunez, R., Gonzalez, J., & Camberos, J. (2007, June). Large-scale numerical solution of partial differential equations with reconfigurable computing. In the 18th AIAA Computational Fluid Dynamics Conference (p. 4085).

[73] Dong, S., & Karniadakis, G. E. (2004). Dual-level parallelism for high-order CFD methods. Parallel Computing, 30(1), 1-20.

[74] Ebrahimi, A., & Zandsalimy, M. (2017). Evaluation of FPGA hardware as a new approach for accelerating the numerical solution of CFD problems. IEEE Access, 5, 9717-9727.

[75] Sun, J., Peterson, G. D., & Storaasli, O. O. (2008). High-performance mixed-precision linear solver for FPGAs. IEEE Transactions on Computers, 57(12), 1614-1623.

[76] Stamoulias, I., Möller, M., Miedema, R., Strydis, C., Kachris, C., & Soudris, D. (2017, June). High-performance hardware accelerators for solving ordinary differential equations. In Proceedings of the 8th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies (pp. 1-6).

[77] Kasbah, S. J., Damaj, I. W., & Haraty, R. A. (2008). Multigrid solvers in reconfigurable hardware. Journal of Computational and Applied Mathematics, 213(1), 79-94.

[78] R. Roscher, B. Bohn, M. F. Duarte and J. Garcke, "Explainable Machine Learning for Scientific Insights and Discoveries," in IEEE Access, vol. 8, pp. 42200-42216, 2020, doi: 10.1109/ACCESS.2020.2976199.

[79] M. F. Tolba, L. A. Said, A. H. Madian and A. G. Radwan, "FPGA Implementation of the Fractional Order Integrator/Differentiator: Two Approaches and Applications," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 4, pp. 1484-1495, April 2019, doi: 10.1109/TCSI.2018.2885013.

[80] M. Baklouti, Ph. Marquet, J.L. Dekeyser, and M. Abid, "FPGA-based many-core System-on-Chip design," Microprocessors and Microsystems, vol. 39, no. 4-5, pp. 302-312, 2015, ISSN 0141-9331, https://doi.org/10.1016/j.micpro.2015.03.007.

[81] Tlelo-Cuautle, E., de la Fraga, L. G., Pham, V. T., Volos, C., Jafari, S., Quintas-Valles, A. D. J. (2017). Dynamics, FPGA realization and application of a chaotic system with an infinite number of equilibrium points. Nonlinear Dynamics, 89, 1129-1139.

[82] Sayed, W. S., Radwan, A. G., Fahmy, H. A., & El-Sedeek, A. (2020). Software and hardware implementation sensitivity of chaotic systems and impact on encryption applications. Circuits, Systems, and Signal Processing, 39, 5638-5655.

[83] Canis, A., Choi, J., Aldham, M., Zhang, V., Kammoona, A., Anderson, J. H., ... & Czajkowski, T. (2011, February). LegUp: high-level synthesis for FPGA-based processor/accelerator systems. In Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays (pp. 33-36).

[84] Karp, M., Podobas, A., Kenter, T., Jansson, N., Plessl, C., Schlatter, P., & Markidis, S. (2022, January). A high-fidelity flow solver for unstructured meshes on field-programmable gate arrays: Design, evaluation, and future challenges. In the International Conference on High Performance Computing in Asia-Pacific Region (pp. 125-136).

[85] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," USSR Computational Mathematics and Mathematical Physics, vol. 4, no. 5, pp. 1-17, 1964, ISSN 0041-5553, doi: 10.1016/0041-5553(64)90137-5

[86] C.-C. Rossow, N. Kroll, and D. Schwamborn, "The MEGAFLOW Project—Numerical Flow Simulation for Aircraft," in Progress in Industrial Mathematics at ECMI 2004, Berlin, Heidelberg: Springer, 2006, pp. 3-33.

[87] Pardo, F., López, P., Cabello, D., & Balsi, M. (2009). Efficient software–hardware 3D heat equation solver with applications on the non-destructive evaluation of minefields. Computers & geosciences, 35(11), 2239-2249.

[88] A. Negoi, ¸S. Bara, J. Zimmermann, et al., "A Dedicated Circuit for Charged Particles Simulation Using the Monte Carlo Method," The Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology, vol. 21, pp. 103–116, 1999, https://doi.org/10.1023/A:1008096121417.

[89] L. Du and Y. Du, 'Hardware Accelerator Design for Machine Learning', Machine Learning - Advanced Techniques and Emerging Applications. InTech, Sep. 19, 2018. doi: 10.5772/intechopen.72845.

[90] E. Roloff, M. Diener, A. Carissimi and P. O. A. Navaux, "High Performance Computing in the cloud: Deployment, performance and cost efficiency," *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, Taipei, Taiwan, 2012, pp. 371-378, doi: 10.1109/CloudCom.2012.6427549.