**ARID ZONE JOURNAL OF ENGINEERING, TECHNOLOGY & ENVIRONMENT**
AZOJETE September 2022. Vol. 18(3):453-468
Published by the Faculty of Engineering, University of Maiduguri, Maiduguri, Nigeria.
**Print ISSN: 1596-2644, Electronic ISSN: 2545-5818**
**www.azojete.com.ng**

**ORIGINAL RESEARCH ARTICLE**

# IMPLEMENTATION OF NUMERICAL INTEGRATION USING FIELD PROGRAMMABLE GATE ARRAYS AND MICROCONTROLLERS: A REVIEW

**M. Adamu[1]*, A. I. Audu[2], C. U. Ngene[2]**
[1]*Department of Computer Science, College of Education and Legal Studies, Nguru, Nigeria*
[2]*Department of Computer Engineering, University of Maiduguri, Maiduguri, Nigeria*
**Corresponding author's email address: msmajemusa4@gmail.com*

**ARTICLE INFORMATION**

Submitted 31 January, 2022
Revised 3 April, 2022
Accepted 6 April, 2022

**Keywords:**
Numerical Integration
Field Programmable Gate Array
FPGA
Microprocessors

**ABSTRACT**

Majority of numerical integration algorithms are written in software and require a long time to initialize and generate results. There is a gradual shift from software-based designs to (Field Programmable Gate Arrays (FPGA)-based designs because they are versatile devices that are increasingly being used as real-time hardware targets in industries. FPGAs are utilized in real-time measurement and control, signal processing, and digital communication, among other applications, because they are inexpensive real-time reconfigurable devices. Because of the high increase in the volume of data used in designs and the large data dependencies in complex computations, researchers are motivated to highlight the need for a shift from the software based implementation of numerical integration to hardware-based FPGAs implementations that are much quicker, more efficient, and more reliable. In this paper, different numerical integration implementations are studied, their strategies discussed, findings assessed and recommendations drawn to enable the suggestion of a better way of improving them. A tabular summary of the review is provided to enhance the understanding of the findings of the review and recommendations for each of the implementations. Based on the review, it can be deduced that FPGAs implementations are faster than software based implementations in C++ and Matlab. It is also evident from the review that architectural designs like pipelining and parallelism when employed on FPGAs greatly improve the throughput of designs and models which significantly reduce data dependencies. It is recommended that FPGA implementation should be employed in the development of numerical integration algorithms due to their low cost, power consumption, small size, and high speed.



## 1.0 Introduction

The study of how to find the numeric values of an integral function at a given set of independent variables is known as numerical integration. A classic numerical analysis task is to find a square with the same area as the area under a curve. The purpose of numerical integration is to obtain an approximate solution to a definite integral (Khalil, 2015). Situations arise in which the analytical method developed so far cannot be used to evaluate some definite integrals (Abdel-Raouf et al., 2014). When analytical methods fail, we typically turn to numerical operations, which are normally carried out on a calculator or computer. These approaches do not give exact values for the definite integrals, but they do give plausible estimates (Khalil, 2015). Configurable computing, which adapts hardware resources to meet

specific hardware designs, has recently been shown to improve performance for a variety of computationally intensive applications (Wang and Ziavras, 2003).

The desire to design and implement definite integrals on hardware devices that are portable, fast, and accurate in computation, is an active area of research. Various attempts have been made in the literature to develop definite integral solution systems using approaches that are distinct from each other. The attempts made to implement definite integrals in hardware have largely focused on microcontrollers, which are devices that operate sequentially. Recent attention has been focused on the idea of implementing definite integral systems on field programmable gate arrays (FPGAs) (Razak et al., 2017). Because of the high increase in the volume of data used in designs and the large data dependencies in complex computations, researchers are motivated to highlight the need for a shift from the software-based implementation of numerical integration to hardware-based FPGAs implementations that are much quicker, more efficient reliable, and reconfigurable. They also have the additional feature of circuit design optimization by packages like VHDL. Software-based numerical integration implementations are dominant, and with the large increase in the volume of data for processing and the large dependencies on data during computations and processing, there is an urgent need for a shift to hardware-based designs (FPGA implementations) for better performance (Razak et al., 2017).

Studies on FPGA and microcontroller-based numerical integration are examined one by one in order to identify research gaps that will benefit and strengthen the numerical integration research trend. The goal of the review is to raise awareness among researchers about the need to enhance research in numerical integration by identifying gaps and encouraging the gradual shift from software-based numerical integration towards FPGA-based numerical integration so as to take advantage of the exceptional capabilities of the FPGAs, which include speed and precision, re-configurability, circuit optimization, parallelism, low power consumption, and, finally, the ability to reduce design latency by improving the overall processing time of a complex operation (Razak et al., 2017).

The paper is organized into five sections. Section one of the paper introduces the paper by presenting a background to the study, motivation, state of numerical integration and the goal of the study is to highlight the need for a shift from microcontroller-based to FPGA-based implementations. Section two presents a review of related literature. Section three examines and discusses different research on microcontroller and FPGA-based numerical integration implementations. Section four presents a tabular summary of the review of the different implementations of numerical integration. Conclusions are drawn in section five, and references are listed at the end.

## 2. Review of Related Literature

The review of related literature begins with an explanation of the concepts of FPGAs and microcontrollers, followed by research works on numerical integration.

*Corresponding author's e-mail address: msmajemusa4@gmail.com*

### 2.1 Field Programmable Gate Arrays (FPGAs)

An FPGA (Field Programmable Gate Array) is a semiconductor in which the circuitry can be customized. This allows you to describe exactly which wire (signal) leads where and what operation should be applied to that signal. The quantity of really concurrent instruction executions is not limited by the number of physical cores it has, whereas a traditional processor is limited by its physical capabilities (Cong et al., 2018). You can define your own instructions (sub-circuits) using an FPGA, which might potentially boost computational speed. Because of their speed and low energy consumption, FPGAs can outperform microcontroller CPUs in numerical workloads. FPGAs are also well-known for their high-throughput, low-latency signal processing capabilities in applications such as real-time audio, video, and data stream processing (Elizabeth, 2022).

### 2.2 Microcontroller

A microcontroller (MCU for microcontroller unit) is a miniature computer built on a single MOS integrated circuit (IC) chip. A microcontroller is a computer that has one or more CPUs (processor cores), memory, and programmable input and output peripherals. A microcontroller is related to, but less sophisticated than, a system on a chip in modern terms (SoC). A microcontroller may be one of the components of a SoC, but it is frequently combined with advanced peripherals such as a graphics processing unit (GPU), a Wi-Fi module, or one or more coprocessors (Raghunathan, 2021). Automobile engine control systems, implantable medical devices, remote controls, office equipment, appliances, power tools, toys, and other embedded systems all employ microcontrollers. They make it cost-effective to digitally control even more devices and processes by decreasing the size and cost compared to a system that uses a separate microprocessor (Papoutsidakis et al., 2017).

### 2.3 Literature on numerical integration

In the study of Jandja et al. (2018), the five columns rule in solving definite integration by parts through transformation of integral limits, offers an alternative method for solving definite integration by parts. A function f(x), which is continuous on the closed interval [a, b], serves as the starting point for the definition of the definite integral. The given interval is divided into "n" subintervals, which, while not required, can be assumed to have equal lengths (x). An arbitrary domain value, x i, is chosen in each subinterval, and its subsequent function value, $f(x_i)$, is determined. The product of each function value times the corresponding subinterval length is determined, and these "n" products are added to determine their sum (CliffsNotes, 2022). It solves integrals using a tabular integration by parts algorithm with integral limit transformations. The results showed that after transformation, the part-by-part integration formula takes on a new form. The final solutions obtained by this method and the standard technique in calculus textbooks were discovered to be identical. The authors devised the five-column rule to shorten the traditional methods' lengthy procedures (Jandja et al., 2018). The following is how each column works: The first column alternates the positive and negative signs, the second column is f(x), and the third column is g'(x), and then its integral lies below it, the fourth column is the lower limit (x = a) and upper limit (x = b), and the fifth column is the transformation of integral limits, i.e. the lower double limit u(a)v(a) and upper double limit u(b)v(b), and the sixth column is the transformation of integral limits (Jandja et al., 2018).

Corresponding author's e-mail address: msmajemusa4@gmail.com

Yu (2016) investigates two types of definite integrals in his paper "A Study of Definite Integrals Using Parseval's Identity." The fundamental conclusion regarding the summability of the Fourier series of a function is known as Parseval's identity, after Marc-Antoine Parseval. It is a generalized Pythagorean theorem for inner-product spaces from a geometric perspective (which can have an uncountable infinity of basis vectors) (Provenzi, 2021). The infinite series expressions of the two types of definite integrals were determined using Parseval's identity. For practical calculations, two instances were used. The study's research technique was to uncover solutions using hand computations and then validate them using Maple. The research method enables the detection of calculation errors as well as the modification of original thinking paths. When compared to the one obtained by Maple, Parseval's identity method produced good results. It further stated that Maple offers advice and guidance on problem-solving techniques (Yu, 2016).

Gonzalez et al. (2010) provided a new heuristic method for evaluating definite integrals in their paper "Definite integrals via the method of brackets." The method of brackets is a technique for evaluating definite integrals over the half-line that was created in the context of evaluating integrals derived from Feynman diagrams. A limited set of operating rules are used in this method to turn the integral into a bracket series. This bracket series is evaluated by a second, condensed set of rules, which yields the output as a regular series (Gonzalez et al., 2020). Despite the fact that the method is heuristic and lacks a rigorous explanation, the researchers believe it is a powerful method of integration. Working with and generating results of definite integrals is relatively simple. In methods developed for the evaluation of Feynman diagrams, evaluating a definite integral is reduced to solving a linear system of equations with method of brackets and origin. The study described the operating rules and provided various examples to demonstrate the method. The bracket technique simplifies the evaluation of a large class of definite integrals to the solution of a linear system of equations, with the main principle being that any parameter ‘a’ is assigned a bracket. The study presents a powerful approach of integration that is simple to use (Gonzalez et al., 2010).

An algorithmic way to find definite single and double integrals using Simpson's rule has been proposed (Selvakumar, 2017). One of the numerical techniques used to calculate the definite integral is Simpson's rule. Typically, we employ the calculus fundamental theorem to get the definite integral, which requires us to use antiderivative integration techniques. Finding the antiderivative of an integral, for example in scientific experiments where the function must be ascertained from the observed values, can occasionally be challenging. Therefore, in such circumstances, numerical approaches are utilized to approximate the integral (Byjus, 2022). Engineers and scientists will benefit from this study since it will make it easier to calculate definite integrals with less calculation time (run time) and storage space. The work highlighted a need to develop a Simpson approach that is distinct from the classic Simpson method used in literature to solve ordinary differential equations numerically. Numerical experiments were conducted after the invention of the new Simpson approach to demonstrate the algorithm's validity (Selvakumar, 2017).

Yu (2014) investigated three types of definite integrals in his research, "Evaluating Some Types of Definite Integrals" using Parseval's theorem, infinite series representations of these definite

*Corresponding author’s e-mail address: msmajemusa4@gmail.com* 

integrals were produced, and some examples were presented for practical computations. The work additionally used Maple to calculate estimated values of definite integrals in order to verify the results obtained through the application of Parseval's theorem to the research findings. According to the findings, the theorem has a wide range of applications and can be used to solve a variety of complex situations (Yu, 2014).

## 3. Research in Numerical Integration

Numerical integration algorithms were analyzed to establish the nature of the numerical integration method used, as well as the results and recommendations derived from the study. This will allow for the identification of areas where more work is required to strengthen and reposition the development of robust software models and reconfigurable hardware-based solvers with small size, high speed, high accuracy, power efficiency, and low resource utilization to benefit a wide range of complex applications, due to the increasing volume of process data and data dependencies in designs.

### 3.1 High Speed Numerical Integration Using FPGA

In their study, Razak et al. (2017) used FPGA to construct four numerical integration algorithms on hardware. The four hardware-based numerical integration methods are Left Riemann Sum (LRS), Middle Riemann Sum (MRS), Right Riemann Sum (RRS) and Trapezoidal Sum (TS). Using the target chip Altera cyclone IV FPGAs, system performance is assessed in terms of resource usage, clock latency, execution time, power consumption, and computational error. Cyclone IV devices are perfect for low cost, small-form-factor applications in the wireless, wireline, broadcast, industrial, consumer, and communications industries because they offer power and cost savings without losing performance, as well as a low cost integrated transceiver option (Altera Corporation, 2014). The customizable logic block (CLB) matrix at the center of field programmable gate arrays (FPGAs), a semiconductor device, is coupled by programmable interconnects. After production, FPGAs can be reprogrammed to meet specific application or feature needs. FPGAs are distinguished from Application Specific Integrated Circuits (ASICs), which are made especially for particular design tasks, by this property. Although one-time programmable (OTP) FPGAs are available, SRAM-based FPGAs, which can be updated as the design changes, are more common (Xilinx, 2022). The results of the study indicated that MRS and TS are more precise and error-free. When compared to other types of calculation, FPGA computations are the fastest, followed by C++ and MATLAB computations (Razak et al., 2017).

### 3.2 Pipelined FPGA Implementation of Numerical Integration of the Hodgkin-Huxley Model

In the study of Osorio (2016), the Hodgkin-Huxley model is utilized to describe the onset and propagation of action potentials in neurons' axons. The model consists of a set of nonlinear differential equations that may be solved numerically for a given set of parameters. Action potentials in neurons' axons start and spread according to the Hodgkin-Huxley model. Differential evolution techniques frequently work in tandem with numerical integration. Double-precision floating-point arithmetic can be used to implement complex functions in modern FPGAs with competitive performance and cost (Osorio, 2016). A mathematical model that describes the initiation and spread of action potentials in neurons is the Hodgkin-Huxley

Corresponding author's e-mail address: msmajemusa4@gmail.com

model, often known as the conductance-based model. An approximate representation of the electrical properties of excitable cells, such as neurons and muscle cells is provided by a set of nonlinear differential equations. It is a dynamical system in continuous time. In order to understand the ionic mechanisms behind the start and spread of action potentials in the squid giant axon, Alan Hodgkin and Andrew Huxley developed the model in 1952 (Stiles and Gray, 2022). The best set of empirical parameters is what optimization aims to achieve. Iterations or generations in differential evolution may number in the thousands or millions. For the following generation, new solutions that outperform the prior generation are kept. Those who fail the test are eliminated (Osorio, 2016). In this case, numerical integration on an FPGA was utilized to offer the best parameters at a specific point in order to build a model. According to the findings, FPGA-based pipelines improve model speed despite significant data dependencies, lowering latency and increasing throughput. The researchers proposed design efforts that will give a higher performance gap between a hardware implementation and a comparable software implementation due to large data dependencies. They also stated that complex architectures can be mapped on FPGAs since they are large enough. Large pipelines are required for complex computational models, which can be shared by multiple instances of the optimization algorithm running at the same time. Deeper pipelines and higher clock frequencies will necessitate the use of more segment adders and multipliers in future designs (Osorio, 2016).

### 3.3 A Hardware Intensive Approach for Efficient Implementation of Numerical Integration for FPGA Platforms

In their research, Khursid and Mir (2014) employed a one-to-one link between Integration algorithms and universal FIR filters. The trapezoidal rule for numerical integration was implemented in this study by translating the integrating algorithm onto FIR structures. The Integration algorithm is implemented using this relationship to produce a structure. However, critical route delays in such systems are common, limiting sampling/throughput rates. Concurrency was used at several points throughout the procedure to tackle the problem. Independently, the effects of pipelined and parallel structures on speed and power metrics were investigated (Khursid and Mir, 2014). FPGAs enable rapid prototyping of Application Specific Integrated Circuit (ASIC) at low to medium production volumes. Because the design-implement-test-debug cycle may take only hours rather than months as with ASIC, design modifications will be easier. FPGAs also have intrinsic advantages such as low power consumption, low cost, ease of modification, real-time processing, and large capacity. Furthermore, FPGA is distinguished by its flexibility as a result of the predesigned Configurable Logic Blocks (CLB) that is used in industrial applications that match the aimed requirements, and the parallel structure that makes it suitable for high-speed applications; it also outperforms microprocessors. Furthermore, FPGA clock rates are in the hundreds of MegaHertz range (MHz) (Abdelaty et al., 2020). Performance can be greatly improved by architectural modifications like pipelining and parallel processing in models, according to a study. The dynamic power dissipation is drastically decreased when all of the structures are run at the same nominal frequency. System tweaks can be used to create low-power designs (Khursid and Mir, 2014).

Corresponding author's e-mail address: msmajemusa4@gmail.com

### 3.4 Implementation of Fractional Order Integrator/differentiator on Field Programmable Gate Array

Using the Grunwald–Letnikov concept, Rana et al. (2016) proposed a systematic technique for hardware implementation of the basic operators of fractional calculus (fractional integrator and derivative) using a field programmable gate array (FPGA) in the LabVIEW environment. The fractional order integrator and derivative of sinusoid and square waveform signals were simulated, and hardware implementation results were provided. In system modeling, fractional order calculus has been discovered to be useful. When compared to integer order modeling, fractional calculus modeling is substantially more accurate. The strong match between the simulated and experimental results demonstrated the FPGA device's suitability for fractional order control and signal processing applications (Rana et al., 2016). A fractional-order integrator, also known as a fractional integrator, is a type of integrator that determines the fractional-order integral or derivative of an input, also known as a differintegral. The parameter of differentiation or integration is a real or complex one. The fractional integrator is helpful in fractional-order control when the output of the control system depends on the history of the system being controlled (Krishna, 2011). A toolkit developed transforms LabVIEW code into VHDL (very high-speed integrated circuits hardware description language) for FPGA processing. The paper recommends future studies to integrate fractional operators with the hardware processes for fractional order control or signal processing applications to enhanced performance (Rana et al., 2016).

### 3.5 Solution of Definite Integrals Using Functional Link Artificial Neural Networks

Changdar and Bhattacharjee (2019) described a method for solving definite integrals using feedforward artificial neural networks in their paper "Solution of Definite Integrals Using Functional Link Artificial Neural Networks." The major goal was to develop a new method based on neural networks that could be used to solve definite integrals in a more efficient and unique way than existing approaches. The integrand was approximated by training the neural network weights using a neural network model based on FLANN architecture. When compared to other current approaches, the solved examples show that the procedure is effective and practicable in all circumstances. By reducing a well-constructed error function, the definite integrals were solved (Changdar and Bhattacharjee, 2019). The results showed that the procedure is effective and precise when compared to existing traditional numerical approaches, and that it is well-suited for applications that need the integration of higher order polynomials. Although it takes time to set up and acquire results, the research effort provided a fast and reliable technique to integrate higher order polynomials using a robust algorithm. The process needs to be sped up, and the portability of the method and solver also needs to be addressed. This will enable quick and on-the-spot answers to definite integral problems.

### 3.6 High Precision Computing of Definite Integrals with .NET Framework C# and X-MPIR

High Precision Computing of Definite Integrals with.NET Framework C# and X-MPIR is a study by Dzhambov (2014) that is part of a series of publications chronicling the progress of building tools for high precision computations in this environment. An application for realizing arbitrary precision functions and numerical algorithms in a given context, created with the

*Corresponding author's e-mail address: msmajemusa4@gmail.com*

most up-to-date state-of-the-art library. The main goal of the research project is to create powerful, mutually supportive computational instruments for dealing with non-trivial problems in a given environment. The program solves infinite integrals with an arbitrary real integer a, or (a,∞). It employs a text form function expression that can include algebraic operations and functions. The application was written in C# with the goal of being as portable as possible (Dzhambov, 2014). There is need of hastening the results generation procedure. This can be achieved by implementing it on hardware (FPGA) using Hardware Description Language such as VHDL (Abdelaty et al., 2020). The researchers also proposed opportunities for further advancement in a variety of approaches, including the use of contemporary processors' multi-core architecture for parallel computing. Because similar software products are generally built using Unix-like systems with specialized hardware, and therefore, wider audience would be interested.

### 3.7 Chaotic Firefly Algorithm for Solving Definite Integral

In their paper "Chaotic Firefly Algorithm for Solving Definite Integral," Abdel-Raouf et al. (2014) created a Firefly Algorithm with Chaos (IFCH) for calculating definite integrals. The Improved Firefly Algorithm with Chaos (IFCH) solves the problem of concurrent numerical integration calculations in sciences and engineering while also giving adaptive segmentation points. The Firefly algorithm is the most recent addition to the swarm intelligence metaheuristics for optimization problems family. The results of the IFCH algorithm were compared to those of the Monte Carlo method, trapezoidal rule, Simpson's rule, and midpoint rule. When compared to exact values, the researchers discovered that the IFCH algorithm outputs were quite close to the exact values of the specified functions under consideration. This approach has numerous applications and can easily solve a variety of difficult problems (Abdel-Raouf et al., 2014).

### 3.8 A Chaotic Bat Algorithm for Solving Definite Integral

By combining the Bat Algorithm with chaos, Khalil (2015) improved the Bat Algorithm for estimating the numerical value of definite integrals. The algorithm is capable of overcoming the problem that older algorithms have with non-uniform segmentation spots. Several simulation examples show that the technique can converge to the best solution with excellent accuracy and a high convergence rate. IBACH solves the problem of concurrent numerical integration and adaptive segmentation. The bat algorithm's superior performance can be attributed to its search capacity (Khalil, 2015). When implemented on hardware (FPGA) using Hardware Description Languages such as VHDL, the algorithm is improved due to the parallelism and pipelining inherent in FPGA (Abdelaty et al., 2020).

### 3.9 The Efficient Implementation of Numerical Integration for FPGA Platforms

The research of Hemavathi and Ravichanda (2015) is focused on numerical strategies for computing approximation solutions to infinite integrals. These are software-oriented strategies. The researchers believe that a hardware-oriented solution that assesses performance in terms of speed and area is required. Integral implementations have long critical path delays, which limit throughput rates. This has been mitigated by architectural changes

Corresponding author's e-mail address: msmajemusa4@gmail.com

that allowed the structure to run at higher throughput rates while using less space. The trapezoidal rule is used to implement pipelining structures, fine grain pipelined structures, and parallel structures. The method used in this study is mapping the trapezoidal rule onto the FIR structure. At the system level, pipelining and parallel processing have been utilized to manage concurrencies (Hemavathi and Ravichanda, 2015). According to the study, the greater the number of parallel structures, the better the throughput and error rates of the models. Further research into architectural changes such as pipelining and parallel processing is recommended to significantly improve performance.

### 3.10 Quick Implementation of Definite Integrals Computation with Very High Precision

Dzhambov (2017) discussed how to use parallel calculations with two perspective quadrature algorithms in a specific context, namely the.NET Framework. Comparative experiments show that using specific tools from the environment, it is possible to perform effective computations with extremely high precision. This paper describes a portion of the research done to create a tool library for arbitrary precision calculations. The main goal is to demonstrate that tools for solving non-trivial problems with high precision computations can be implemented in a real-world setting, such as desktop computers. It is successful because of the following factors: 1) appropriate quadrature schemes 2) optimization of basic program instruments for high precision computations, and 3) effective implementation of fundamental mathematical functions (Dzhambov, 2017).

### 3.11 Numerical simulations and FPGA implementations of fractional-order systems based on product integration rules.

Product integration (PI) rules are well-known numerical algorithms for solving integer and, more recently, fractional order differential equations. The PI rules used to solve fractional-order systems (FOS) have high memory reliance, making their hardware implementation complex and resource-intensive. Abdelaty et al. (2020) modified utilized versions of the PI rules to enable digital implementations in their study "Numerical Simulations and FPGA Implementations of Fractional-Order Systems Based on Product Integration Rules." The predict-evaluate-correct-evaluate (PECE) and PI rectangle rules were investigated. FPGA implementations of the three PI rules were created using Xilinx ISE 14.7 on Artix 7 kit. The improved throughput for PI rectangular was 128.8 Mbits/sec, 129.12 Mbits/sec for PI trapezoidal, and 1280 Mbits/sec for fractional-order PECE (Abdelaty et al., 2020).

### 3.12 Strategies for choosing an appropriate numerical method for FPGA-based HIL

Yushkova (2021) did research on "Strategies for choosing an acceptable numerical method for FPGA-based Hardware-In-the-Loop (HIL)" to improve the results of HIL techniques by combining Field-Programmable Gate Arrays (FPGAs), which allow for faster and more accurate real-time simulations. The selection of right numerical approach for implementation is one of the implementation issues. Four widely used numerical methods were evaluated for HIL performance: first order Forward Euler, second order Adams-Bashforth, second order Runge-Kutta, and fourth order Runge-Kutta. The simplest methods achieved the best overall

performance (1st order Forward Euler and 2nd order Adams-Bashforth), even though Runge-Kutta methods achieved the highest accuracy when accuracy limiting constraints in real-world environments are considered (Yushkova, 2021).

### 3.13 Accelerating Interference-based QoS Analysis of Vehicular Ad Hoc Networks for BSM Safety Applications: Parallel Numerical Solutions and Simulations

Zhao et al. (2021) considered the context of more practical vehicular communication environment: BSM broadcast, asynchronous timing between hidden terminals, Nakagami channel fading, and Non-Homogeneous Poisson Process vehicle distribution in their study "Accelerating Interference-based Quantity of Service (QoS) Analysis of Vehicular Ad Hoc Networks for (Broadcasting Basic Safety Message) BSM Safety Applications: Parallel Numerical Solutions and Simulations." The computational complexity of numerical solutions for Quantity of Service (QoS) and capacity metrics is so high that the computation time is unacceptably long. The metrics were evaluated using a combination of numerical integration and a parallel technique. For speeding up the computational process, the Monte Carlo integration and MPI (Message Passing Interface) methods were used, and the analysis of QoS measures was validated (Zhao et al., 2021).

## 4. Comparative Analysis of different Implementations of Numerical Integral Solver

Numerical integration algorithms are studied and their strengths, weaknesses, and application strategies are deduced and presented in Tables 1-3. Appropriate recommendations are drawn from each of the algorithms studied to suggest areas that can be improved in order to enhance research in numerical integration.

*Corresponding author's e-mail address: msmajemusa4@gmail.com*

**Table 1:** FPGA-Based Numerical Integral Solver Implementation Summary

| S/N | TITLE | STREGNTH | WEAKNESSES | APPLICATION STRATEGY | RECOMMENDATION |
|---|---|---|---|---|---|
| 1 | High Speed Numerical Integration Using FPGA (Razak et al., 2017) | Able to successfully established that FPGA perform better than software based models | Multiple Algorithms: FPGAs and Software based-models. Advantages and disadvantages discussed and the choice left open. | FPGAs and Software based designs. MRS, RRS, LRS, TP, C++ and MATLAB. | The choice of an algorithm for a specific usage is determined by type of work to be done. Indicated that FPGAs implementations are faster than software and hence recommended improvement by using floating points. |
| 2 | Pipelined FPGA Implementation of Numerical Integration of the Hodgkin-Huxley Model (Osorio, 2016). | Use of FPGA to reduce data dependencies and increase model performance | Peculiar application of FPGA and Numerical Integration to a specific model | FPGA and Hodgkin-Huxley model Numerical Integration Method | The design needs to be generalized or extended to capture more numerical integration methods not just that of Hodgkin-Huxley model. |
| 3 | A Hardware Intensive Approach for Efficient Implementation of Numerical Integration for FPGA Platforms (Khursid and Mir, 2014). | Eliminate critical route delays in models using numerical integration | Not a generalized model design. Meant to improve performance of a specific design. | FPGA implementation using One-to-one relation with FIR filter. | Concurrency can be achieved through pipelined and parallel structures in models. |
| 4 | Implementation of Fractional Order Integrator/differentiator on Field Programmable Gate Array (Rena et al. 2016). | Accurate performance achieved through FPGA implementation | Application of FPGA and numerical integration for fractional order control and signal processing applications. | FPGA in LabVIEW environment for fractional order control and signal processing. | The researchers recommend future studies integrating the developed fractional operators with the hardware processes. |
| 5 | The Efficient Implementation of Numerical Integration for FPGA Platforms (Hemarathi and Ravichanda, 2015). | The design is able to discover that the bigger the number of parallel structures, the better the models' throughput. | Restricted to the calculation of numerical integration of infinite integrals. | FPGA implementation, Mapping trapezoidal rule onto the FIR structure to implement pipelining structures, fine grain pipelined structures, and parallel structures. | Recommended architectural adjustments such as pipelining and parallel processing to significantly enhance performance. |

*Corresponding author's e-mail address: msmajemusa4@gmail.com*

| | | | | | |
|---|---|---|---|---|---|
| 6 | Numerical simulations and FPGA implementations of fractional-order systems based on product integration rules (Abdelaty et al., 2020). | Improved throughput achieved with the implementations in the model and a reduced memory reliance. | Specific Application of FPGA to PI rules used to solve fractional-order systems (FOS) | Used of FPGA Implementations of Fractional-Order Systems Based on Product Integration Rules | Recommend modified use of versions of the PI rules to enable digital implementations on FPGAs to improve throughput. |
| 7 | Strategies for choosing an appropriate numerical method for FPGA-based HIL (Yushkova, 2021). | Achieved Performance improvement in Runge–Kutta,1st order Forward Euler and 2nd order Adams–Bashforth numerical methods. | Restricted to research on strategies for choosing an acceptable numerical method for FPGA-based Hardware-In-the-Loop (HIL). | Used FPGA-based Hardware-In-the-Loop (HIL) using four numerical integration methods (first order Forward Euler, second order Adams–Bashforth, second order Runge–Kutta, and fourth order Runge–Kutta.) | Runge–Kutta methods get the highest accuracy among the numerical methods while the 1st order Forward Euler and 2nd order Adams–Bashforth have high Cost Performance Indexes. |

**Table 2:** Neural Network-Based Artificial Intelligence (AI) Numerical Integral Solver Implementation Summary

| S/N | TITLE | STREGNTH | WEAKNESSES | APPLICATION STRATEGY | RECOMMENDATION |
|---|---|---|---|---|---|
| 1 | Chaotic Firefly Algorithm for Solving Definite Integral (Abdel-Raouf et al., 2014) | Has a good convergence rate, precision, robustness and accuracy. | Initialization delays and takes time to generate results. | Solves definite integral using Improved Firefly Algorithm with Chaos (IFCH) | Can be greatly improved if implemented on FPGA using appropriate toolkits. |
| 2 | A Chaotic Bat Algorithm for Solving Definite Integral (Khalid, 2015) | Overcome problem of non-uniform segmentation spots in other models. Able to converge to the best solution with excellent accuracy and high convergence rate. | Software based design and takes time to initialize and generate results. | Solves definite integral using Improved Bat Algorithm with chaos. | Can be greatly improved if implemented on FPGA using appropriate toolkits. |
| 3 | Solution of Definite Integrals Using Functional Link Artificial Neural Networks (Changdar and Bhattacharjee, 2019) | Efficient, effective and precise, with a well-constructed error function | takes time to initialized and generate results | Used Functional Link Artificial Neural Networks to perform numerical integration | Can be greatly improved if implemented on FPGA using appropriate toolkits. |

*Corresponding author's e-mail address: msmajemusa4@gmail.com*

**Table 3:** Software-Based Numerical Integral Solver Implementation Summary

| S/N | TITLE | STREGNTH | WEAKNESSES | APPLICATION STRATEGY | RECOMMENDATION |
|---|---|---|---|---|---|
| 1 | High Precision Computing of Definite Integrals with .NET Framework C# and X-MPIR (Dzhambov, 2014) | powerful, mutually supportive computational instruments for double exponential transformation | The approach is for a specific problem. | Definite integration with .NET Framework C# and X-MPIR. | A software based design. Initialization delays can be reduced or eliminated by FPGA implementation using appropriate toolkits. It can be extended to include more class of definite integrals. |
| 2 | Quick Implementation of Definite Integrals Computation with Very High Precision (Dhambov, 2017) | Perform effective computations with extremely high accuracy using specific tools of an environment. | Design for a specific environment. Initialization delays and takes time to generate results. | Used parallel calculations with two perspective quadrature algorithms in a specific context, namely the.NET Framework | Can be greatly improved if implemented on FPGA using appropriate toolkits |
| 3 | Accelerating Interference-based QoS Analysis of Vehicular Ad Hoc Networks for BSM Safety Applications: Parallel Numerical Solutions and Simulations (Zhoa et al., 2021) | Numerical integration and a parallel techniques were used to reduce computation time delay. | Restricted to computational complexity of numerical solutions for Quantity of Service. | Used Monte Carlo integration and MPI (Message Passing Interface) | Numerical integration and parallel techniques can reduce latency in deigns if employed. |

## 5. Conclusion

This study was able to review different types of numerical integration techniques implemented in software and Field Programmable Gate Array (FPGA). It is evident from the review that pipelining and parallel processing were the major techniques employed to achieve speed and accuracy, both in software-based and FPGA-based numerical integral solvers. Numerical integration on FPGA-based hardware is becoming more common as researchers devote more time and resources to maximizing the capabilities of FPGAs. The ability of FPGAs to operate at higher speeds while incorporating numerous pipelining and parallel processing with a high degree of accuracy, low power, and minimum resource utilization has made them the choice of most industries and researchers. The software-based algorithms are the conventional numerical integration approaches that operate sequentially (programmes that follow sequence of instructions). Different researchers found out that such algorithms require improvement in terms of power consumption, speed, latency, throughput, accuracy, size reduction, and resource utilization. Numerical integrators developed on FPGAs are shown to employ pipelining and parallel processing to greatly speed up complex processes. The research trend in numerical integrals needs to tap into these functionalities coupled with the re-configurability of the FPGAs. Hence, FPGA implementation should be employed in the development of numerical integration algorithms due to their low cost, power consumption, small size, and high speed.

*Corresponding author's e-mail address: msmajemusa4@gmail.com*

## REFERENCES

Abdel-Raouf, O., Abdel-Baset, M. and El-henawy, I. 2014. Chaotic Firefly Algorithm for Solving Definite Integral. International Journal of Information Technology and Computer Science, 6(6): 19–24.

Altera Corporation. 2013. Cyclone IV FPGA Device Family Overview. Cyclone IV Device Handbook, (1). Retrieved from https://www.digikey.com/en/datasheets/intel-fpgasaltera/intel-fpgasaltera-cyclone4-handbook Accessed on 2/9/2022.

Abdelaty, AM., Roshdy, M., Said, LA. and Radwan, AG., 2020. Numerical simulations and FPGA implementations of fractional-order systems based on product integration rules. IEEE Access, 8:102093-102105.

Byjus. 2022. Simpson Rule. Retrieved from https://byjus.com/maths/simpsons-rule/ Accessed on 2/9/2022

Changdar, S. and Bhattacharjee, S. 2019. Solution of Definite Integrals using Functional Link Artificial Neural Networks. Retrieved from http://arxiv.org/abs/1904.09656, 2019 Accessed on 12/12/2021.

CliffsNotes. 2022. Definite Integral. Retrieved from https://www.cliffsnotes.com/study-guides/calculus/calculus/integration/definite-integrals Accessed on 2/9/2022.

Cong, J., Liu, B., Neuendorffer, S., Noguera, J., Vissers, K. and Zhang, Z. 2011. High-level synthesis for FPGAs: From prototyping to deployment. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 30(4): 473-491.

Dzhambov, V. 2014. High precision computing of definite integrals with .NET framework C# and X-MPIR. Cybernetics and Information Technologies, 14(1): 172–182.

Dzhambov, V. 2017. Quick Implementation of Definite Integrals Computation with very High Precision. Journal of Information Technologies and Control, 3: 20-24., DOI: 10.1515/itc-2017-0030.

Gonzalez, I. and Moll, VH. 2010. Definite Integrals by the Method of Brackets. Part 1. Advances in Applied Mathematics, 45(1): 50–73.

Gonzalez, I., Jiu, L. and Moll, VH. 2020. An extension of the method of brackets, Part 2, Open Mathematics, 18(1): 983-995. Retrieved from https://doi.org/10/1515/math-2020-0062 Accessed on 2/9/22022.

Elizabeth, M. 2022. What is FPGA and How Does it Work? Retrieved from https://history-computer.com/fpga-field-programmable-gate-array/ Accessed on 10/02/2022.

Jandja, M. and Mohammad, L. 2018. The Five Columns Rule in Solving Definite Integration by Parts Through Transformation of Integral Limits. Journal of Physics: Conference Series 1028,

2nd International Conference on Statistics, Mathematics, Teaching and Research, 9-10 October, Makassar, Indonesia, 012109: 1-7.

Hemavathi, H. and Ravichandra, V. 2015. The Efficient Implementation of Numerical Integration for FPGA Platforms. International Journal of Ethics in Engineering & Management Education, 2(7): 34-38.

Khalil, AW. 2015. Chaotic Bat Algorithm for Solving Definite Integral. International Journal of Computers and Technology, 14(4): 5592-5598.

Khurshid, B., and Mir, RN. 2014. A hardware intensive approach for efficient implementation of numerical integration for FPGA platforms. 27th International Conference on VLSI Design and 13th International Conference on Embedded Systems, 312-317.

Krishna, BT. 2011. Studies on fractional order differentiators and integrators: A Survey of Signal Processing, 91(3): 386-426.

Millenium Circuits Limited. 2022. Field Programmable Gate Arrays (FPGA) VS. Microcontroller – What's the difference? Retrieved from https://www.mclpcb.com/blog/fpga-vs-microcontroller/ Accessed on 2/9/ 2022.

Osorio, RR. 2016. Pipelined FPGA implementation of numerical integration of the Hodgkin-Huxley model. IEEE 27th International Conference on Application-specific Systems, Architectures and Processors (ASAP), 202-206.

Papoutsidakis, M., Tanwar, R., Chatzopoulos, A. and Tseles, D. 2017. Custom made embedded automation systems for smart homes-part 2: the implementation. International Journal of Engineering, Applied Science and Technology, 2(5): 16-19.

Provenzi, E. 2021. From Euclidean to Hilbert Spaces: Introduction to Functional Analysis and Its Applications. New York, John Wiley and Sons.

Razak, FNA., Talip, MSA., Yakub, MFM., Khairudin, ASM., Izam, TFTMN. and Zaman, FHK. 2017. High speed numerical integration algorithm using FPGA. Journal of Fundamental and Applied Sciences, 9(4): 131-144.

Raghunathan, KR. 2021. History of Microcontrollers: First 50 Years. IEEE Microcontrollers, 41(6): 97-104.

Rana, KPS., Kumar, V., Mittra, N. and Pramanik, N. 2016. Implementation of fractional order integrator/differentiator on field programmable gate array. Alexandria Engineering Journal, 55 (2): 1765-1773.

Stiles, PJ. and Gray, CG. 2021. Improved Hodgkin–Huxley type model for neural action potentials. European Biophysics Journal, 50(6): 819-828.

*Corresponding author's e-mail address: msmajemusa4@gmail.com* 

Wang, X. and Ziavras, SG. 2003. Parallel LU factorization of sparse matrices on FPGA-based configurable computing engines. Concurrency and Computation: Practice and Experience, 16(4): 319-343.

Xilinx 2022. Field Programmable Gate Array (FPGA), https://www.xilinx.com/products/ silicon-devices/fpga/ what-is-an-fpga.html accessed 2nd September, 2022.

Yu, C. 2014. Evaluating Some Types of Definite Integrals. American Journal of Software Engineering, 2(1): 13-15., DOI: 10.12691/ajse-2-1-3.

Yu, C. 2016. A Study of Definite Integrals Using Parseval ' s Identity. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 1(2): 28–35.

Yushkova, M., Sanchez, A. and de Castro, A. 2021. Strategies for choosing an appropriate numerical method for FPGA-based HIL. International Journal of Electrical Power and Energy Systems, 132: 107186.

Zhao, J., Zhou, H., Wang, Y., Lu, H., Li, Z. and Ma, X. 2021. Accelerating Interference-based QoS Analysis of Vehicular Ad Hoc Networks for BSM Safety Applications: Parallel Numerical Solutions and Simulations. VEHITS, 600-610

*Corresponding author's e-mail address: msmajemusa4@gmail.com* 