AIMS Bioengineering

AIMS Bioengineering, 12(3): 412–434.
DOI: 10.3934/bioeng.2025020
Received: 24 June 2025
Revised: 20 August 2025
Accepted: 01 September 2025
Published: 05 September 2025

https://www.aimspress.com/journal/Bioengineering

Research article

Electronic modeling of biochemical pathways on reconfigurable computing platform-a repressive circuit of soil bacteria

Syeda Ramish Fatima* and Maria Waqas

Department of Computer and Information Systems Engineering, NED University of Engineering and Technology, Karachi, Pakistan

* **Correspondence:** Email: syedaramish@neduet.edu.pk; Tel: +923342037313.

Abstract: Cytomorphic hardware, in which bio-cellular processes are mapped onto an electronic substrate, has gained quite a prominence in the past few decades due to advancements in the fields of systems biology, synthetic biology, and other related disciplines. Conventionally, the analog substrate is the preferred choice for nonlinear biochemical reactions in the biological circuits; however, many researchers have exploited the flexibility, reconfigurability, and ease of design of digital platforms such as reconfigurable field-programmable gate arrays (FPGAs). In this work, we briefly examined work on such reconfigurable platforms and proposed a novel and straightforward technique to implement the systems of ordinary differential equations describing such biochemical reactions in a modular fashion using single-precision floating-point intellectual property cores available on a representative reconfigurable platform. A simplified biochemical system of a repressive three-way network of soil bacteria was taken into account, and the system of ordinary differential equations was first simulated in a software-based environment with results compared against a hardware-based realization. We observed a significant speed-up of up to 5x in the reconfigurable platform-based realization compared to the software-based realization. The values in the hardware-based realization were also more accurate compared to the software-based approach. Statistical error metrics (RMSE, MAE, and NRMSE) further confirmed a close numerical match between the two implementations. Hence, the results conformed well, and thus, this design strategy can be incorporated into the future cytomorphic design on field-programmable gate arrays.

Keywords: bio-chemical reactions; cytomorphic; ordinary differential equations; reconfigurable platforms; soil bacteria network

1. Introduction

The basic cellular structures like neural networks in the nervous system and other cellular networks in biological systems are quite complex and interwoven in nature. In order to understand the mechanisms behind functions performed by them, mathematical and computational modeling techniques play a pivotal role. These techniques enable scientists and researchers to gain some insight into these intricate mechanisms. Likewise, the intracellular biochemical processes might look like simple chemical processes at first, but the interactions and regulation patterns are highly complex, and thus, only mathematical and computational modeling can be deployed to fully understand these cascades of biochemical pathways. The computational or in-silico modeling of biological processes using software tools has been an active research domain called Executable Biology [1,2].

Many different algorithms are based on Finite State Machines as in [3], but the model remains largely theoretical without experimental validation. The biological complexity of real mutations and regulatory networks is seen in [4], where practical biochemical feasibility, dynamics of actual genetic parts, and scalability beyond educational demonstration are untested; and in [5], where the system is greatly simplified and is limited to two inputs and five states only, and the reaction time, though improved, lies on the order of minutes, making scalability and speed bottlenecks in more complex settings.

An artificial-based system in [6] blends mechanistic kinetic model structures with model-free reinforcement learning to adaptively identify both the correct kinetic form and its time-varying parameters, but validation is limited to in-silico scenarios only, lacking real experimental data; and the hybrid structure remains vulnerable to over-parameterization or over-fitting. Similarly, standardized markup languages [7,8] and other computational variants [9,10] are major contributions in this domain. Another interesting approach is to define the network motifs of the most commonly occurring biological networks and create computational algorithms for them [11,12].

All these computational models, however, are phenomenological, and the real molecular level of cellular interactions is not incorporated. Various rates like the rate of product formation, affinities of binding, and rates of transcription are not taken into account or are adjusted to attain certain results. Furthermore, considering a few molecules, surface receptors, and transcription factors is not enough to give a reasonable insight into the phenomenon under consideration. The biological systems like the nervous system and immune system, in particular, rely on molecular heterogeneity. The process of learning requires individual nerve cells to be distinct.

When we consider modeling of neural plasticity, for instance, we consider only the electrical aspects arising from action potentials, their propagation, and synaptic delays; however, these electrical mechanisms take place due to protein-based ion channels, pumps, and receptors present on the cell membranes, and these proteins undergo many variations due to alternative splicing, post translational modifications, and other factors. Hence, there is a possibility that the long term memory formation relies on post translational protein modifications of synaptic proteins [13,14].

Another major issue related to computational modeling is the resources required for intricate and complex simulations of biological processes, such as supercomputer scale machines and cloud facilities. Hence, a growing number of researchers are looking toward other means of modeling like using Graphical Processing Units (GPUs), Field Programmable Gate Arrays (FPGAs), and analog/mixed signal domain electronic substrates [15–17]. GPUs, however, support Single

Instruction Multiple Data (SIMD) scenarios, whereas in biochemical pathways, each ODE represents a number of different operations on the same data. Hence, using GPUs is not a very scalable solution when large and complex biological processes are to be modeled, as a very large number of GPUs would be required in such cases.

Besides digital implementations, some researchers have used analog and mixed-signal platforms to implement biochemical processes. Dr. S. M. Razaul Hasan and his colleagues, at the Center for Research in Analog and VLSI microsystems dEsign (CRAVE), Massey University, Auckland, New Zealand, have made notable contributions to bio-cellular process modeling through mixed IC design. Their work includes a sequential circuit level design for simulating biological processes and cell signaling pathways, with sequencing governed by circadian rhythms [18], a hysteretic electronic switch [19], analogues for mRNA transcription [20], and a comprehensive electronic model of post gene transcription mechanisms [21], all of which represent significant advancements in the field.

Cytomorphic hardware design, which emulates biological processes on an electronic substrate, has been an active research area for several decades. Its growing relevance spans basic medical science, drug testing, pharmacology, molecular biology, and related fields such as synthetic biology, biosensors, prosthetics, and robotics, highlighting its increasing impact and multidisciplinary importance.

Professor Rahul Sarpeshkar at the Research Laboratory of Electronics, MIT, and his research team have contributed to the domain by performing wet experiments (in-vitro) as well as dry experiments on electronic neuromorphic and cytomorphic chips. Their work, ranging from analog synthetic biology circuits, to biological and bio-inspired supercomputers based on analog computation in cells, and ultra-low-power implantable medical devices such as cochlear implants or diagnostic devices, adds great value to the field of Cytomorphic Computing [22]. He has shown with his designs that the presence of a smaller number of transistors in an analog electronic substrate means faster implementations than any digital counterparts [23,24]. In [24], the researchers also provide an excellent review on how cytomorphic electronic chips at sub-threshold analog levels can be used to map biochemical reaction networks.

Waqas et al. in [25] use velocity saturated short-channel MOS transistors to model very frequently seen bio-cellular reactions, namely receptor-ligand binding and Michaelis-Menten reactions. In [26], Waqas et al. further extend their work in [25] and [27], and they implement an analog MOS based design of a complete bio-chemical pathway. In [28], the p53 protein led pathway responsible for cell death, called apoptosis, is designed by Patra et al. using ODEs of the system model. CMOS circuits are implemented for the entire pathway. This design enabled the study of p53 pathways dynamics without the need for biological cells in a dry lab and would benefit cancer treatment and drug control for such diseases. A more recent work from Patra et al. [29] presents a MOS-based cytomorphic SoC for simulating stochastic apoptosis networks, translating nonlinear biochemical dynamics into subthreshold electronic circuits. Simulation results show strong agreement with experimental data and biosimulator outputs, validating the system's accuracy and stability.

Although analog design, being fuzzy and noisy like biological circuits, seems to be a better choice, the design process is tedious and costly; thus, digital and mixed-signal design is often preferred by researchers. In the digital domain, Application Specific Integrated Circuit (ASIC) and FPGA-based design are possible, but ASIC designs are again rigid and costly; hence, reconfigurable platforms like FPGAs are favored over ASICs. We exclude GPU-based realizations from this discussion as they are not scalable and cannot be implemented independent of CPU-based host machines, which incur an additional overhead in the design.

Cytomorphic design at a higher level of abstraction is much easier to carry out using FPGAs, but a bottom-up approach, where biophysical dynamics of cellular processes are taken into account, poses a huge challenge due to limited resources on the FPGA chip. The biological processes have been mapped onto FPGAs for the past few decades, as we can see in the work of [30–32]. A detailed account of FPGA-based design for biological processes is discussed in the proceeding section. The researchers in [33] provide a comprehensive overview of cytomorphic and neuromorphic trends on different electronic platforms and their applications in various emerging fields.

Here, we intend to investigate the realization of a simple biochemical process on a reconfigurable platform such as FPGA. For dynamic modeling of the selected biochemical process on an FPGA, a novel technique is furnished so as to inspect some advantages and disadvantages that can be incurred in doing so.

We have validated the reconfigurable platform-based design against CPU-based simulation by keeping the initial conditions and rate constants identical in both approaches, as defined in the following sections. A straightforward implementation strategy is adopted for solving the Ordinary Differential Equations (ODEs) representing the biological circuit, using Euler's Method as the numerical solver. This choice is intentional to facilitate optimal resource utilization on the FPGA. Higher order solvers such as the Runge Kutta methods, while offering improved numerical accuracy, are not employed due to their significantly higher computational and memory overhead. These methods require multiple intermediate calculations per time step, which would increase the hardware complexity and resource usage on a reconfigurable platform. Given the nature of our application, particularly when modeling large and intricate biochemical processes, the increased cost in terms of FPGA logic, memory blocks, and latency outweighs the benefits in accuracy. Therefore, Euler's Method presents a favorable trade-off between accuracy and hardware efficiency for our use case.

To ensure a direct correspondence between the software-based and FPGA-based implementations, we avoid using built in ODE solvers in software and instead employ a self-written numerical routine. This enables us to maintain consistent computation logic and precisely match the behavior of both implementations. Furthermore, floating-point arithmetic is utilized instead of fixed-point, eliminating the need for scaling and mitigating issues related to precision loss. Fixed-point arithmetic, although resource-efficient, can degrade accuracy and is, thus, avoided where floating-point resources are available.

The FPGA implementation follows a modular design, enabling future enhancements such as time multiplexing of functional units to further optimize resource usage. Additionally, the use of the Advanced eXtensible Interface (AXI) protocol in the floating-point IP cores provides an efficient means to parallelize independent computations within the ODE system. This helps circumvent the need for sequential execution as seen in CPU-based simulations. The AXI protocol also offers robust control over execution flow, making it well suited for handling feedback loops commonly present in biochemical reaction networks.

1.1. Generic three component repressive network of soil bacteria

A simplified model of a generic three-component repressive network, representing a biochemical circuit observed in soil-dwelling bacteria, is known to exhibit oscillatory behavior depending on the parameters selected. This network comprises three proteins A, B, and C that can exist in phosphorylated (A-p, B-p, C-p) or dephosphorylated (A, B, C) states. The biochemical interactions among these proteins form a closed feedback loop, where protein A catalyzes the dephosphorylation of protein B, protein B catalyzes the dephosphorylation of protein C, and protein C, in turn, catalyzes the phosphorylation of protein A. The activation signals in the feedback loop are illustrated in Figure 1, where circles at the end of the arrows denote activation. The reactions involved in this regulatory circuit are governed by Michaelis-Menten kinetics. The system is modeled mathematically using a set of three coupled ordinary differential equations (ODEs) (1), (2), and (3) for the three proteins A, B, and C, respectively, incorporating Michaelis-Menten expressions to describe the enzymatic activity of each component [34,35]. The resulting model is expressed as follows:

$$\frac{d[A]}{dt} = \frac{k_{p1}([A_{total}] - [A])}{K_{p1} + [A_{total}] - [A]} - \frac{k_{k1}[A][C]}{K_{k1} + [A]} \tag{1}$$

$$\frac{d[B]}{dt} = \frac{k_{p2}([B_{total}] - [B])[A]}{K_{p2} + [B_{total}] - [B]} - \frac{k_{k2}[B]}{K_{k2} + [B]} \tag{2}$$

$$\frac{d[C]}{dt} = \frac{k_{p3}([C_{total}] - [C])[B]}{K_{p3} + [C_{total}] - [C]} - \frac{k_{k3}[C]}{K_{k3} + [C]} \tag{3}$$

In these three equations, $[A_{total}]$, $[B_{total}]$, and $[C_{total}]$ represent the total concentrations of proteins A, B, and C, respectively. $[A]$, $[B]$, and $[C]$ give their resulting molecular concentrations, respectively, and $(d[A])/dt$, $(d[B])/dt$, and $(d[C])/dt$ represent the rate of change of concentration for proteins A, B, and C, respectively. k_{p1}, k_{p2}, and k_{p3} are the rate constants for dephosphorylation of A, B, and C, respectively, while k_{k1}, k_{k2}, and k_{k3} are the rate constants for phosphorylation of the three proteins, respectively. K_{p1}, K_{p2}, and K_{p3} are the dissociation constants for the dephosphorylation of A, B, and C, respectively, and K_{k1}, K_{k2}, and K_{k3} are the Michaelis-Menten constants of phosphorylation of A, B, and C, respectively.

The biological motivation for modeling this simplified soil bacterial repressive loop arises from its relevance to naturally occurring gene regulatory networks that govern adaptive responses in microbial populations. Soil bacteria frequently encounter fluctuating environmental conditions, such as changes in nutrient availability, moisture, and pH, necessitating robust regulatory mechanisms to maintain homeostasis and survival. Minimal repressive circuits, such as the one modeled here, capture essential features of such regulation, including nonlinearity, time-delayed feedback, and the potential for oscillatory dynamics. These properties make the system a useful abstraction for studying the fundamental principles of feedback control in biological systems.

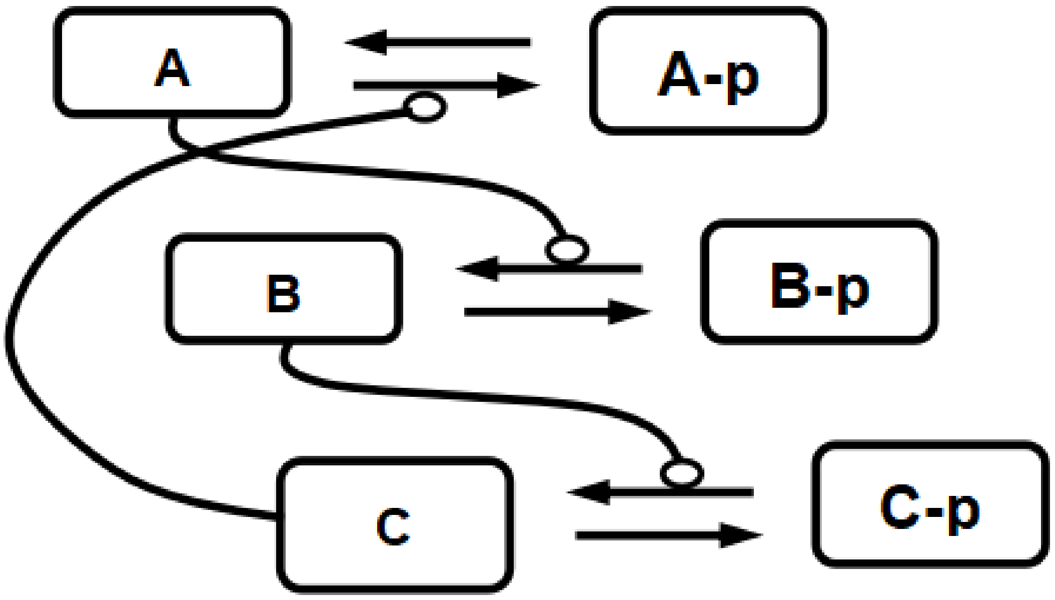


Figure 1. A generic three protein repressive network of soil bacteria [34].

By focusing on a reduced model with only three interacting components, it becomes feasible to analyze the system analytically and computationally, thereby providing insights into the parameter regimes that give rise to oscillatory or steady-state behavior. Furthermore, such simplified models are useful test cases for hardware-based implementations of biological networks, such as those deployed on field-programmable gate arrays (FPGAs), enabling real-time simulation and emulation of biochemical processes. The results obtained from modeling and simulating these circuits not only enhance our understanding of natural regulatory motifs but also contribute to the development of synthetic biological circuits for use in biosensors, drug delivery systems, and programmable bio-devices

1.2. Reconfigurable computing platforms for biochemical pathway realizations

Reconfigurable computing platforms, as the name implies, enable the modification of various design parameters even after successful implementation. Moreover, the design can be easily altered whenever required, with minimal effort and cost. Designs can also be made robust and adaptable by configuring critical parameters as inputs to the system; for instance, by storing initial values in Block RAM (BRAM) or Ultra RAM (URAM) available on many FPGA devices. In such cases, the design does not need to be recompiled; instead, parameter changes can be made simply by writing new values to the memory blocks.

In this work, we explore a technique for implementing a small-scale biological process on an FPGA using Euler's method as the ODE solver. The chosen three-protein repressive network has a limited number of ODEs and parameters, making it ideal for demonstrating the implementation methodology. However, the proposed approach is scalable and can be extended to larger and more complex biological networks, where the benefits of FPGA-based acceleration become more significant.

1.3. Engineering significance and application scope of FPGA-based biochemical circuit implementation

The primary contribution of this work lies in the realization of a biologically inspired regulatory network on reconfigurable hardware. Although the underlying biochemical model, a three-component repressive loop governed by Michaelis-Menten kinetics, is conceptually established in systems biology, we introduce a novel hardware-based implementation that enables real-time and resource-efficient simulation. The circuit is implemented using floating-point arithmetic IP cores to ensure numerical stability and eliminate the need for manual scaling inherent in fixed-point systems. This design decision preserves computational fidelity, which is particularly important in modeling biochemical processes with sensitive dynamic ranges.

The use of the Advanced eXtensible Interface (AXI) protocol further enhances system efficiency by enabling concurrent processing of independent ODE computations. This contrasts with CPU-based implementations, which typically rely on sequential execution and are therefore limited in terms of speed and scalability. Through modular design and IP-level abstraction, the system supports extensibility to larger networks and enables time- -multiplexing strategies for resource optimization on the FPGA.

From an application standpoint, the presented approach holds significant promise for domains such as synthetic biology, where rapid in-silico prototyping of genetic circuits is essential. By enabling hardware-accelerated emulation of gene regulatory networks, the platform facilitates fast validation of circuit dynamics before experimental realization. Furthermore, the ability to simulate complex feedback-driven systems in real time has potential utility in pharmacodynamics and drug discovery workflows, particularly for simulating signaling cascades under varying conditions. Some other emerging fields that may benefit from this work include biomorphic artificial intelligence, neuro-prosthetics and brain-computer interfaces, bio-electronic medicine, DNA-based memory and computing devices, bio-computers, and others [33]. These capabilities make the proposed implementation a generalizable framework for the deployment of biologically relevant models in embedded, high-throughput, or edge-computing environments.

2. Related work

Recent advances in cytomorphic circuits implemented on reconfigurable platforms, particularly Field-Programmable Gate Arrays (FPGAs), have demonstrated the potential for high-speed and parallel simulation of biochemical processes. However, much of the work emphasizes implementation than addressing the broader engineering trade-offs, resource constraints, or biological fidelity gaps that we aim to bridge. Early contributions in this area include the work by Osana et al. [36–39], where FPGA-based simulators were developed using deterministic (ODE-based) and stochastic approaches to model metabolic and biochemical pathways. These efforts culminated in the development of ReCSiP (Reconfigurable Cell Simulation Platform) [39,40], which demonstrated considerable speedup compared to conventional software implementations. Despite these advances, the ReCSiP platform relied on now-obsolete FPGA hardware and legacy toolchains, limiting its relevance to modern programmable logic devices.

Soleimani and Drakakis, in [41], extended this line of research by implementing a cellular calcium signaling model on FPGA, capable of pipelining between 10,000 and 40,000 calcium units.

This demonstrated not only a significant improvement in execution time compared to CPU-based simulations but also the feasibility of modeling large-scale cellular dynamics on reconfigurable hardware. However, their focus remained largely on performance metrics, without much emphasis on hardware scalability or generalization to broader biochemical networks.

Further work by Soma et al. [42], and later extended by Acharya et al. [43], introduced hardware-software co-design methodologies using MATLAB Simulink and Xilinx System Generator. These tools facilitated rapid prototyping of biochemical circuits through modular design libraries. While this approach enabled greater flexibility and ease of implementation, it introduced software dependencies and version compatibility issues that could affect design portability and long-term reproducibility. Moreover, both works acknowledged that such abstraction tends to omit key biological intricacies, thereby limiting its biological interpretability. In [44], Acharya et al. demonstrated a cytomorphic engineering approach by modeling key cellular biochemical reactions as low-power electronic circuits using FPGA-based implementation. The synthetic models of p53 signaling pathways are validated against cell culture data and efficiently realized on a Zynq-7000 board, consuming only 310 mW of power.

Several challenges persist across these studies. First, much of the prior work depended on Systems Biology Markup Language (SBML) for model representation. While SBML is a widely accepted standard, it poses several practical issues, such as dependency conflicts (e.g., libSBML and libstdc++ compatibility with MATLAB), memory management concerns, and difficulties in parsing complex expressions; issues documented on the SBML.org website. These limitations restrict automation and integration into modern design workflows. Second, despite the demonstrated speedup, most platforms use fixed-point arithmetic to reduce resource utilization. However, fixed-point methods can compromise numerical accuracy, particularly for systems with wide dynamic ranges or sensitive kinetic parameters. This trade-off is not well addressed in the literature, especially in the context of biologically faithful circuit emulation.

In contrast, we propose a modular, floating-point FPGA implementation that addresses these limitations on several fronts. First, it eliminates reliance on SBML or external software layers by directly implementing the system of ODEs derived from biochemical reactions. Second, it uses floating-point arithmetic to preserve numerical precision without requiring result scaling, which is crucial for accurate biological modeling. Third, it leverages the AXI protocol-based IP cores to exploit the parallelism inherent in biochemical systems, providing scalability and real-time execution capabilities. In doing so, this work contributes not only a performance-optimized design but also a biologically consistent and hardware-portable framework that fills key gaps in current cytomorphic circuit implementations.

3. Experiments to implement biological processes of repressive networks of soil bacteria

3.1. CPU-based implementation

For a software-based approach, we selected MATLAB R2019a running on an Intel i7 Core machine. We used Euler's method to implement the three-way repressive network of soil bacteria as given by equations (1), (2), and (3), using the code structure defined below.

Table 1. Rate constants and Michaelis-Menten reaction constants.

Constant symbol	Constant name	Value
k_{p1}	Constant rate for dephosphorylation for protein A	20 sec^{-1}
K_{p1}	Dissociation constant for phosphorylation of A	0.2 mM
k_{k1}	Constant rate for phosphorylation for protein A	5 sec^{-1}
K_{k1}	Michaelis–Menten constant for phosphorylation of protein A	8 mM
k_{p2}	Constant rate for dephosphorylation for protein B	15 sec^{-1}
K_{p2}	Michaelis-Menten constant for dephosphorylation of protein B	10 mM
k_{k2}	Constant rate for phosphorylation for protein B	30 sec^{-1}
K_{k2}	Dissociation constant for phosphorylation of protein B	3 mM
k_{p3}	Constant rate for dephosphorylation for protein C	5 sec^{-1}
K_{p3}	Michaelis-Menten constant for dephosphorylation of protein C	40 mM
k_{k3}	Constant rate for phosphorylation for protein C	50 sec^{-1}
K_{k3}	Dissociation constant for phosphorylation of protein C	3 mM

We first defined all constants, set the time step and simulation time, and set all the initial conditions, as shown in Table 1. The initial molecular concentrations of proteins A, B, and C were kept at 5 mM each. We implemented a loop to handle the progression of time at each time-step and thus, compute the rates of change in concentration of proteins at each time step denoted as d[A]/dt, d[B]/dt, and d[C]/dt in equations (1), (2), and (3), respectively. We used a very small timestep of 0.005 seconds. Based on the attained values, we calculated new values of protein concentrations, appended them to a vector storing all the new values, and plotted these values to display the resultant concentration curve.

3.2. FPGA-based implementation

For FPGA-based implementation, we selected Xilinx Vivado 2020.2 and Artix 7 device version xc7a100tcsg324-1. When it comes to FPGA-based realization of ODE solvers, one of the primary concerns is the use of fixed-point arithmetic or floating-point arithmetic to solve the equations. Though we could use fixed-point arithmetic in Verilog, it is known that such an attempt results in errors related to precision in the results, especially when the molecular concentrations can be subject to very minute changes in each time-step. Moreover, the synthesis of multiplication and division operations poses a major concern in such implementations, as the division and multiplication operators are not synthesizable in Verilog. Thus, a self-reliant synthesizable circuit based on some adequate algorithm, such as Booth's algorithm for multiplication, may be required. This is not only tedious, but also error-prone and expensive from the perspective of resource utilization. One possibility is to come up with some multiplier-less implementation and use a multiplication-based approach for division as well. Another way is to use DSP (Digital Signal Processor) slices for multiplication using fixed-point arithmetic. Since Verilog does not support any fixed-point representation, it considers all the operations as integer operations, and the results need to be scaled to get the correct results.

In our implementation, we opted for floating-point operations available as Intellectual Property (IP) cores in Xilinx. IP cores are available through an IP catalog that comes with the Xilinx software. Various IP cores can be instantiated in the design as and when required. One major advantage of using these IP cores for floating-point operations is that it enables users to optimize the use of DSP slices. Moreover, since it works according to the Advanced eXtensible Interface (AXI) protocol, better control of the operations is possible, which is discussed in the following text. Once an IP core for an operation is generated, it can be used as many times as required by simply instantiating it in the design. We used Floating-Point IP Core in XILINX Floating-Point Operator V7.1. The Xilinx Floating-Point Operator core enables a range of floating-point arithmetic operations to be performed on FPGA. The operation is specified when the core is generated, and each operation has similar interface. This core complies with much of the IEEE-754 Standard. Some standards supported are listed as below. For this research work, we took binary 32 (single precision format).

- *binary16 (Half Precision Format):* 16 bits, with an 11-bit fraction and 5-bit exponent.
- *binary32 (Single Precision Format):* 32 bits, with a 24-bit fraction and 8-bit exponent.
- *binary64 (Double Precision Format):* 64 bits, with a 53-bit fraction and 11-bit exponent.

This standard necessitates the results to be accurate to half of one Unit in the Last Place (ULP). This Floating-Point Operator enables multiplication, addition/subtraction, fused multiply-add, division, square-root, and conversion operators. Thus, multiplication, addition/subtraction, accumulator, fused multiply-add, division, and others are used as building blocks for various modules in our design.

3.2.1. Verilog modules for implementation of repressive networks

The required floating-point IP cores were configured and generated first and then the modules for solving equations and producing outputs were created. The implemented modules were as follows:

- Module fp_add_sub: As the name suggests, this IP Core module can be used to add or subtract two floating-point operands in IEEE 754 Single Precision Standard.
- Module fp_multiply : This module can be used to multiply two single precision floating-point numbers.
- Module fp_mult3 : This module comprises two fp_multiply modules to enable multiplication of 3 floating-point quantities.
- Module fp_divide: This module enables single precision division.
- Module fp_mult_add: This module takes three operands as inputs, multiplies the first two, and then adds the third one to generate the result.

3.2.2. Repressive network modules

Using the modules of different floating-point operators described in the previous subsection, various modules are written to implement the ODE solver using Euler's method. These modules are as follows.

- Module *dAdt:* This module computes *d[A]/dt* for protein A using (1). This module takes the constants and initial values required for calculating differential output at each time step. *fp_multiply*, *fp_add_sub*, *fp_divide* are instantiated multiple times to carry-out the operations required. Control of operations is established through handshake signals of AXI4 stream protocol.
- Module *dBdt* (used as *dCdt* as well)*:* Similar to *d[A]/dt* module but the relevant operations and their order is maintained according to the repressive circuit model described in (2) through (3) for *d[B]/dt* and *d[C]/dt*, respectively. The same module is used to instantiate both *d[B]/dt* as well as *d[C]/dt* calculations as the working mechanics of proteins B and C are the same.
- Module *A_dtxdAdt:* This module is for the calculation of new values of protein A used as initial value for the next time step. Modules *dAdt* and *fp_mult_add* are instantiated with required control signals for handshaking purposes.
- Module *B_dtxdBdt:* Similar to the module defined above, this is for the computation of proteins B and C. Hence, this module is instantiated twice; once for protein B and then for protein C.
- Module *ABC_NW:* This is the top module of this design where the concentrations of all the three proteins are computed by setting the initial conditions and constants of equations along with the provision of keeping the new values at each time step and produced as outputs. The initial values are the same as described in Table 1.

In Figure 2, we present the block diagram illustrating various modules derived from equation (1) for protein A. The floating-point operation modules were arranged to enable parallel execution, as the resulting values were independent of each other, as shown in equation (1). Once the dependent values were computed, the next-stage modules were triggered using control signals generated within the architecture. This figure demonstrates the independent initiation and completion of operations for numerators and denominators, and the subsequent stages begin execution as soon as all required operands become available. To enable computational overlapping, multiple instances of each arithmetic operation were instantiated as needed. However, we could also provide the flexibility to time-multiplex these modules, optimizing FPGA resource utilization.

The *fp_add_sub* module was used to compute $[Atotal]$-$[A]$, and another instance computes the denominator of the first term in equation (1). The numerator's multiplication operation was executed in parallel. Once the numerator and denominator values were obtained, the division was carried out. The second term's numerator involved a three-operand floating-point multiplication of k_{k1}, $[A]$, and $[C]$, while the denominator was computed using *fp_add_sub* to add K_{k1} and $[A]$; these operations were initiated simultaneously with the first term's subtraction. Upon computing the numerator and denominator, the second division was performed. A final subtraction between the two computed terms yielded *dA/dt*, which was then multiplied by *dt* and added to the previous value of $[A]$ to update the concentration, stored as *accA*. Figure 2 shows the *A_values* at every time step *dt*.

Figures 3 and 4 depict the modules *B_values* and *C_values*, respectively. Figure 5 provides a top-level overview (ABC_NW) that simultaneously calculates *A_values*, *B_values*, and *C_values*, thereby fully parallelizing the execution in contrast to the sequential computation of a CPU-based realization, illustrated in the flow diagram of Figure 6. This comparison demonstrated that the CPU processes the values for proteins $[A]$, $[B]$, and $[C]$, sequentially, using Euler's method, whereas the FPGA implementation parallelizes the computation of all three protein concentrations and the independent arithmetic operations within equations (1), (2), and (3).

3.2.3. Synthesis results

All the instantiated modules in the proposed design are illustrated in Figures 7 and 8. Figure 7 presents the top-level schematic of the *ABC_NW* module, which implements the system of ODEs given in (1)–(3). This module accepts clock, reset, and start signals as inputs and outputs the updated concentrations *A_new*, *B_new*, and *C_new* for the three proteins at each simulation step.

Figure 8 provides an expanded schematic of the *A_values* instance, highlighting the *A_dtxdAdt* t module, which contains the *dAdt* and *fp_mult_add* units. The *dAdt* module was further decomposed into floating-point operation units, namely *fp_multiply*, *fp_add_sub*, and *fp_divide*. For completeness, detailed expanded schematics for *A_values*, *B_values*, and *C_values* are included in the supplementary material (Figures S1 and S2), showing the internal arrangement of the floating-point operator modules for each protein.

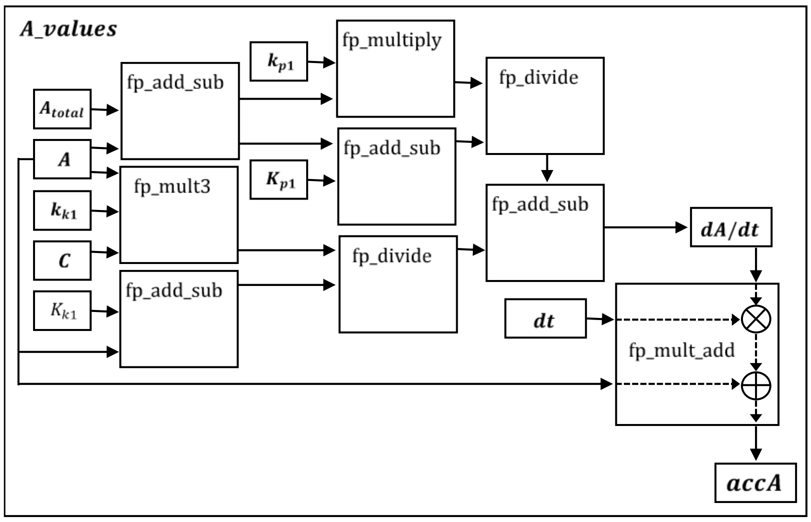


Figure 2. A_values module block diagram based on floating-point modules according to (1) and (6).

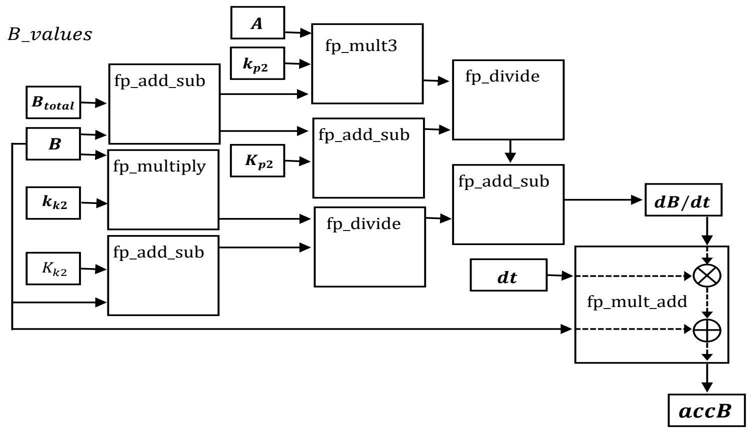


Figure 3. B_values module block diagram based on floating-point modules according to (2) and (6).

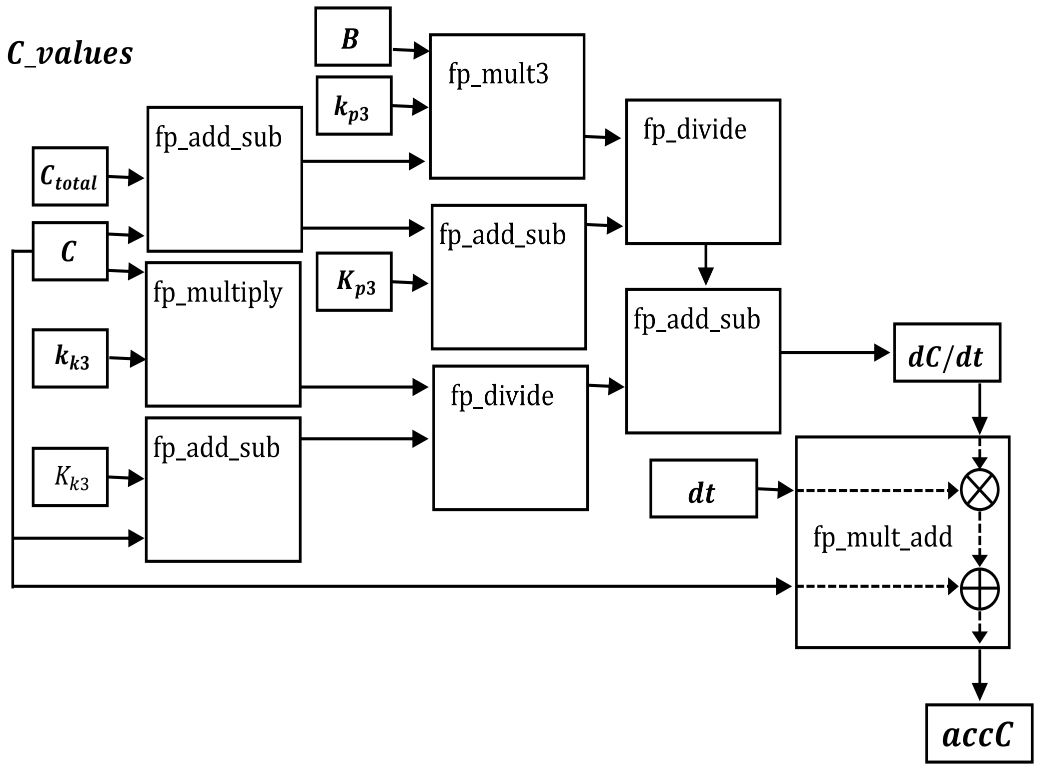


Figure 4. C_values module block diagram based on floating-point modules according to (3) and (6).

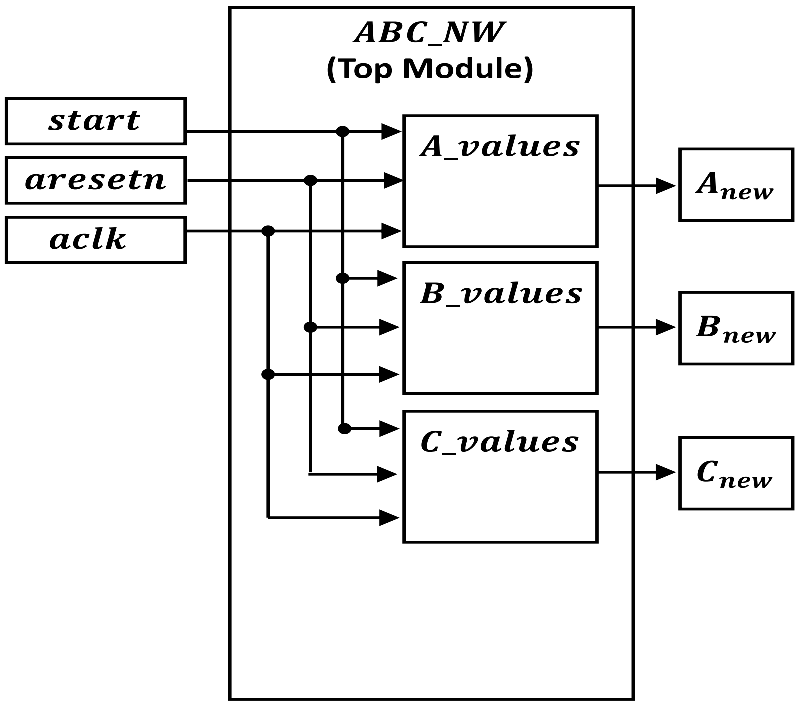


Figure 5. Top module ABC_NW comprising of A_values, B_values, and C_values.

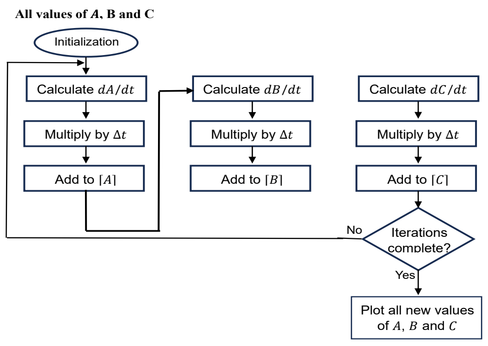


Figure 6. CPU-based realization of a sequential execution of the three loops for proteins A, B, and C unlike parallel execution of the new values for the three proteins, as shown in Figure 4.

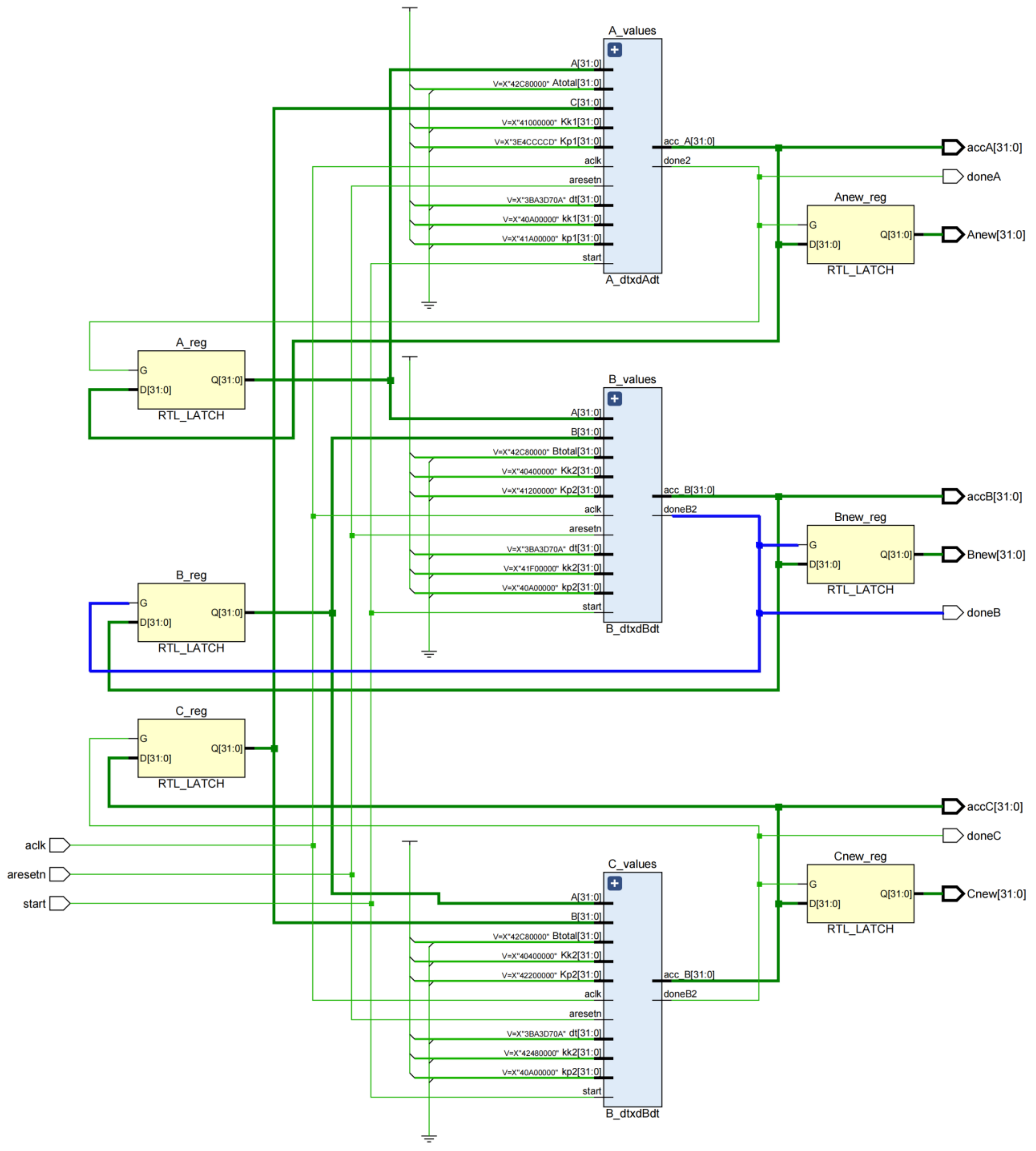


Figure 7. RTL Schematic of the ABC_NW Module.

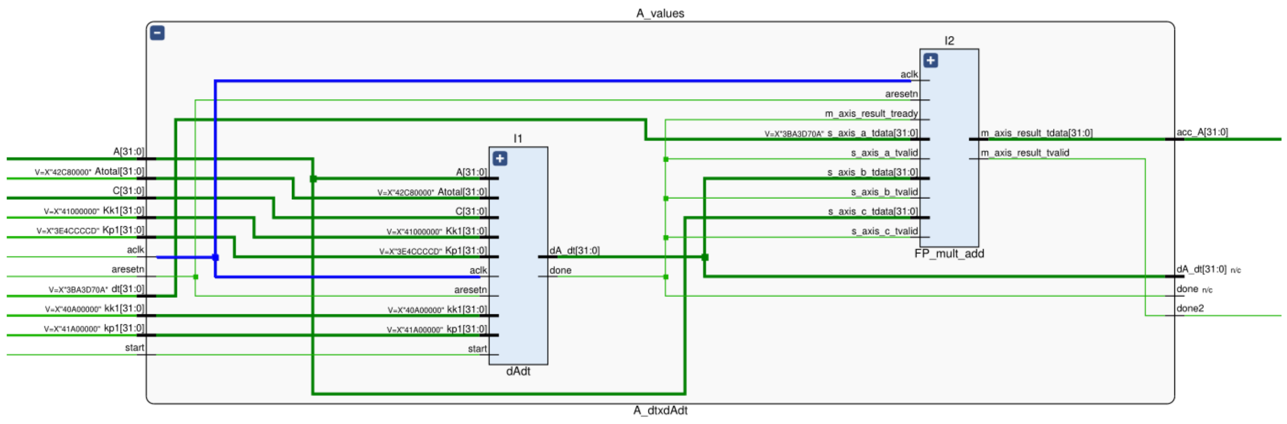


Figure 8. *A_values* expanded to see instances of the *dAdt* and *fp_mult_add* modules.

3.2.4. Resource utilization

Figure 9 shows the estimated post-synthesis resource utilization percentage of FPGA resources used in the design. Look-Up-Table (LUT) and LUTRAM were the basic resources through which FPGA-based design elements were realized. Thus, we observed an adequate percentage of these resources estimated to be used in the design. Since we opted for Digital Signal Processor (DSP) slices when configuring IP cores to facilitate multiplication and division operations, 10% of DSP slices were utilized to optimize the design. Since we latched the concentration values and constants with the design modules, we observed 11% percent of flip-flops (FF) also estimated to be used, which was again acceptable resource utilization. However, we observed that 32-bit single precision floating-point outputs were taxing the IO buffers to 94% but that could be significantly reduced if intermediate values were not included in the output interface such as $accA$, $accB$, and $accC$ buses. These buses could be used within the design without any loss of information required in the design. Table 2 lists a comparison of the resources available and estimated utilization for the Artix 7 FPGA resources in this design.

Table 2. Total number of resources utilized of available resources.

Resource	Estimated number of units	Available number of units	Resource utilization %
LUT	7537	63400	11.89
LUTRAM	360	19000	1.89
FF	13990	126800	11.03
DSP	24	240	10.00
IO	198	210	94.29
BUFG	1	32	3.13

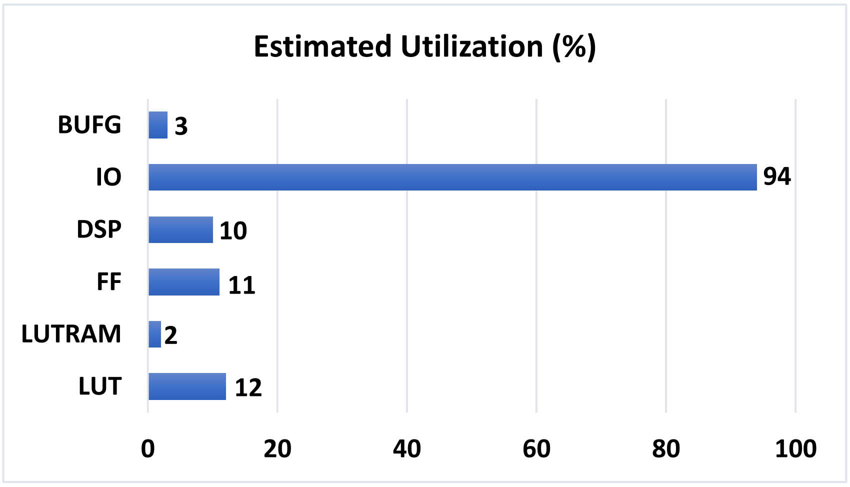


Figure 9. Percentage of resources used in the design.

4. Results and discussions

The comparison between simulation results obtained from MATLAB-based and FPGA-based implementations, both in terms of waveform behavior and numerical values, as illustrated in Figure 10, yields promising insights. We successfully replicate the dynamic biochemical behavior of the three-protein regulatory network. The FPGA implementation, based on IEEE-754 single-precision floating-point format, produces results with higher numerical precision at each time step in contrast to the fixed-point representation employed in the MATLAB simulation.

Figure 10(a) presents the concentration waveforms of proteins A, B, and C, where the green trace denotes A, the blue trace denotes B, and the red trace denotes C. The dynamic interactions between the proteins, governed by the ODE system defined in Equations (1)–(3), are visible. The evolution of concentrations is parameter-dependent, as defined in Table 1. Notably, an initial increase in protein A concentration facilitates a delayed increase in protein B. The concurrent initial decrease in protein C permits this rise in A. As B concentration accelerates, it promotes the increase of C, which in turn stabilizes and reverses the growth of A, exhibiting a negative feedback loop characteristic of the modeled biological circuit.

The corresponding FPGA-based analog waveforms, captured in Vivado Simulator and shown in Figure 10(b), reproduce the same dynamical behavior with high temporal resolution. Again, the green, blue, and red traces represent the concentrations of proteins A, B, and C, respectively. The analog-style display of digital simulation values was enabled via Vivado's analog waveform visualization settings. This feature supports the broader feasibility of using digital reconfigurable platforms to simulate analog biological processes, establishing a viable bridge between continuous-time biological models and digital hardware realization.

The simulation results from the software-based (CPU) implementation extend over a biological timescale of 1.5 seconds. In contrast, the FPGA-based realization achieves equivalent system dynamics in only 0.3 seconds, demonstrating a 5× acceleration in simulated time. This performance gain is depicted in Figure 11(a) and Figure 11(b), which show the MATLAB and FPGA simulation timelines, respectively, along with closely matching concentration values.

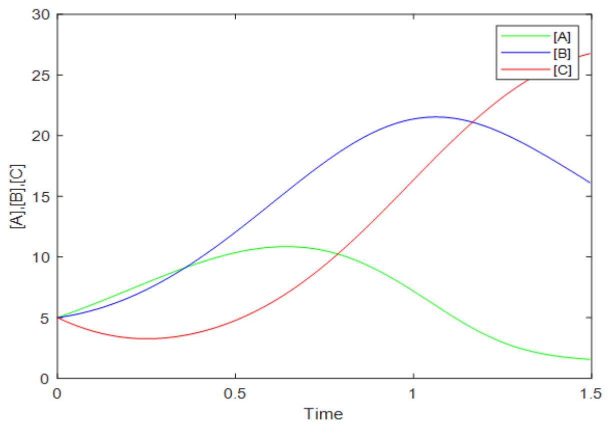
These results not only validate the numerical correctness of our hardware design but also confirm its practical viability. The use of floating-point arithmetic via Xilinx's IP cores, interfaced through the AXI protocol, enabled parallel evaluation of ODE terms, significantly improving simulation throughput compared to sequential CPU execution. Moreover, the system was implemented in a modular fashion to enable scalability. Resource utilization remain within acceptable bounds for a Xilinx Artix 7 FPGA, ensuring that the design is feasible on mid-range hardware platforms. The use of Euler's method, selected for its simplicity and low hardware overhead, provides a favorable trade-off between numerical efficiency and real-time execution constraints.

Together, these findings underscore the accuracy and feasibility of implementing biologically inspired nonlinear ODE systems on reconfigurable digital platforms. The work highlights the potential of such FPGA-based architectures in enabling high-speed, resource-efficient simulations of biochemical pathways (relevant for real-time embedded systems in synthetic biology and other related domains).

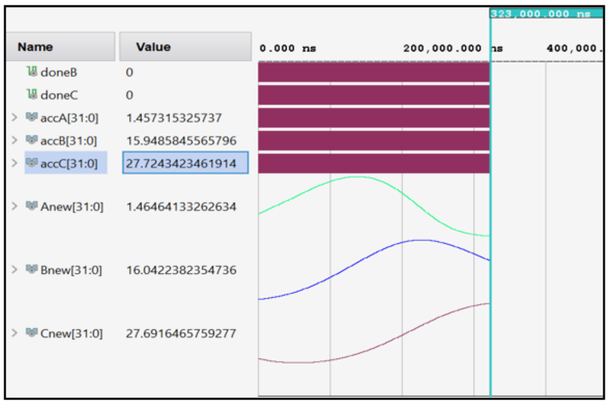
4.1. Statistical analysis of CPU-based and FPGA-based results

To quantitatively validate the accuracy of the FPGA-based implementation, statistical error metrics were computed between the CPU-based and FPGA-based simulation results for protein concentrations A, B, and C. Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) were used as indicators of numerical fidelity as depicted in Figure 12(a). To express the FPGA–CPU deviations relative to typical concentration magnitudes, we also computed the normalized RMSE (NRMSE) using the CPU-based mean concentration as the normalization factor. The NRMSE values are 1.35% (Protein A), 1.85% (Protein B), and 1.97% (Protein C), indicating that the FPGA implementation reproduces the MATLAB reference with deviations below 2% of the typical concentration values. These low percentages confirm the high numerical fidelity of the FPGA-based simulation besides significant execution-time improvements. Figure 12(b) illustrates the NMRSE values of the three proteins.

Furthermore, execution time measurements in Figure 12(c) show that the CPU-based simulation requires 1.500 s, whereas the FPGA-based implementation completes the same simulation in 0.315 s, yielding a 5× acceleration. This performance improvement highlights the suitability of FPGA architectures for real-time biochemical network simulation.



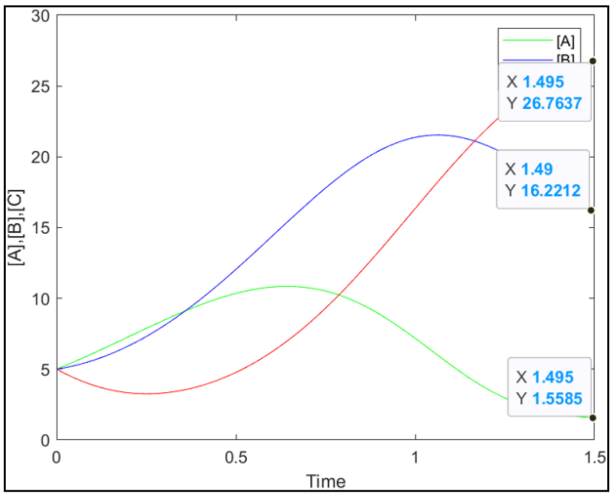
(a)

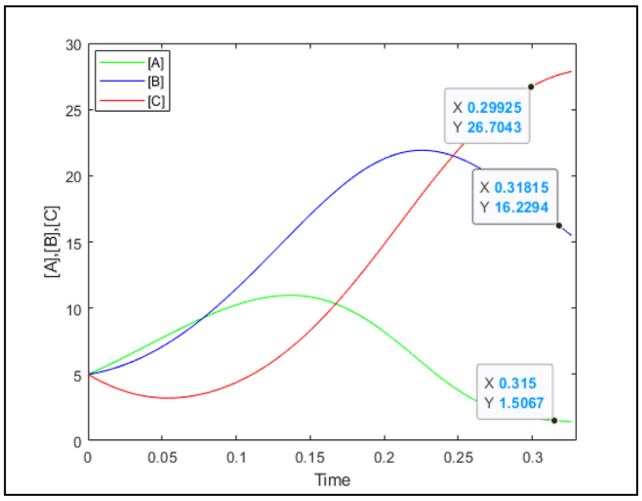


Name	Value
doneB	0
doneC	0
accA[31:0]	1.457315325737
accB[31:0]	15.9485845565796
accC[31:0]	27.7243423461914
Anew[31:0]	1.46464133262634
Bnew[31:0]	16.0422382354736
Cnew[31:0]	27.6916465759277

(b)

Figure 10. Simulation Results. (a) Results from MATLAB-based implementation; and (b) results from FPGA-based implementation in Vivado Simulator using analog waveform.





(a) (b)

Figure 11. Simulation Results. (a) Results from MATLAB-based implementation with time scale from 0 to 1.5 seconds; and (b) results from FPGA-based implementation with time scale from 0 to 0.3 seconds.

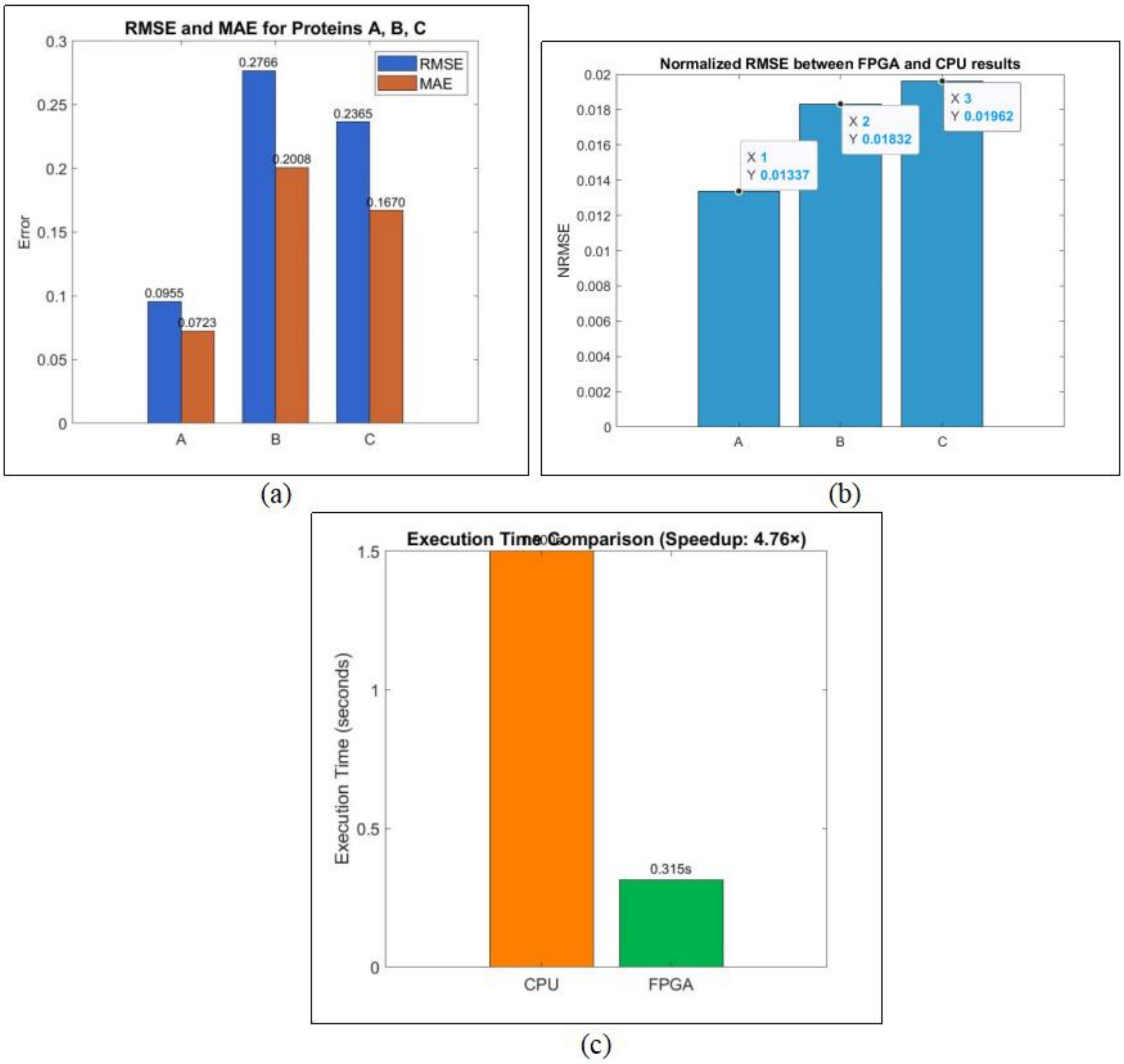


Figure 12. Statistical Analysis of CPU-based and FPGA-based results. (a) RMSE and MAE for proteins A, B, and C (CPU-based and FPGA-based simulations); (b) NRMSE for proteins A, B, and C; and (c) execution time comparison between CPU and FPGA-based simulations.

5. Conclusion

In this work, we propose a reconfigurable platform-based hardware realization using floating-point IP cores for various arithmetic operations in ordinary differential equations to simulate a biological process of soil bacteria. We design our modules to capitalize on the parallelism that can be achieved by starting independent arithmetic operations in a single ordinary differential equation simultaneously, as well as starting the solution of ordinary differential equations at the same time, in contrast to CPU-based algorithms, where solutions are attained sequentially. The AXI protocol enables us to control the start and end of various arithmetic functions. The results are validated against CPU-based simulation, and we observe a speed-up of 5 against software-based realization. Moreover, the results are more accurate due to single floating-point representation of numbers instead of fixed-point representation in software-based implementation. To further strengthen the validation, we conduct a rigorous statistical comparison between FPGA- and CPU-based results using Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Normalized RMSE (NRMSE) for all simulated protein concentrations. The low RMSE values (A: 0.09546, B: 0.27655, C: 0.23648) and NRMSE values below 2% confirm a close numerical match between the two implementations, demonstrating that the hardware realization preserves the accuracy of the original simulation. Furthermore, this work can be improved by using Block RAMs (BRAMs) and Ultra RAMs (URAMs) for constants, initial values, and intermediate values instead of internal registers, thus making the design optimized through parallel implementation techniques. With the techniques and resources available on FPGA chips, it is very likely to continue using such platforms, either for the actual design or for networking of such designs or for both.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

All authors declare no conflicts of interest in this paper.

Authors contributions

We would like to undertake that both authors of this research paper have directly participated in the planning, execution, or analysis of this study. All authors of this research paper have read and approved the final version submitted

References

1. Fisher J, Henzinger TA (2007) Executable cell biology. *Nat Biotechnol* 25: 1239–1249. https://doi.org/10.1038/nbt1356
2. Fisher J, Piterman N (2010) The executable pathway to biological networks. *Briefings Funct Genomics* 9: 79–92. https://doi.org/10.1093/bfgp/elp054
3. Gao R, Hu W, Tarn TJ (2013) The application of finite state machine in modeling and control of gene mutation process. *IEEE Trans NanoBiosci* 12: 265–274. https://doi.org/10.1109/TNB.2013.2260866
4. Fuente D, Garibo i Orts Ó, Conejero JA, et al. (2020) Rational design of a genetic finite state machine: combining biology, engineering, and mathematics for bio-computer research. *Mathematics* 8: 1362. https://doi.org/10.3390/math8081362
5. Liu L, Hong F, Liu H, et al. (2022) A localized DNA finite-state machine with temporal resolution. *Sci Adv* 8: eabm9530. https://doi.org/10.1126/sciadv.abm9530

6. Mowbray MR, Wu C, Rogers AW, et al. (2023) A reinforcement learning-based hybrid modeling framework for bioprocess kinetics identification. *Biotechnol Bioeng* 120: 154–168. https://doi.org/10.1002/bit.28262
7. Zi Z, Klipp E (2006) SBML-PET: a systems biology markup language-based parameter estimation tool. *Bioinformatic* 22: 2704–2705. https://doi.org/10.1093/bioinformatics/btl443
8. Novère NL (2009) The systems biology graphical notation. *Nat Biotechnol* 27: 735–741. https://doi.org/10.1038/nbt.1558
9. Schaub MA, Henzinger TA, Fisher J (2007) Qualitative networks: a symbolic approach to analyze biological signaling networks. *BMC Syst Biol* 1: 4. https://doi.org/10.1186/1752-0509-1-4
10. Palshikar MG, Min X, Crystal A, et al. (2023) Executable network models of integrated multiomics data. *J Proteome Res* 22: 1546–1556. https://doi.org/10.1021/acs.jproteome.2c00730
11. Milo R, Shen-Orr S, Itzkovitz S, et al. (2002) Network motifs: simple building blocks of complex networks. *Science* 29: 824–827. https://doi.org/10.1126/science.298.5594.824
12. Shen-Orr SS, Milo R, Mangan S, et al. (2002) Network motifs in the transcriptional regulation network of Escherichia coli. *Nat Genet* 31: 64–68. https://doi.org/10.1038/ng881
13. Bray D (2015) Limits of computational biology. *In Silico Biol* 12: 1–7. https://doi.org/10.3233/ISB-140461
14. Routtenberg A (2008) Long-lasting memory from evanescent networks. *Eur J Pharmacol* 585: 60–63. https://doi.org/10.1016/j.ejphar.2008.02.047
15. Chalkidis G, Nagasaki M, Miyano S (2010) High performance hybrid functional petri net simulations of biological pathway models on cuda. *IEEE/ACM Trans Comput Biol Bioinform* 8: 1545–1556. https://doi.org/10.1109/TCBB.2010.118
16. Osana Y, Yoshimi M, Iwaoka Y, et al. (2007) ReCSiP: an FPGA-based general-purpose biochemical simulator. *Electron Commun Jpn Part II Electron* 90: 1–10. https://doi.org/10.1002/ecjb.20370
17. De Rubertis G, Davies SW (2003) A genetic circuit amplifier: design and simulation. *IEEE Trans Nanobioscience* 2: 239–246. https://doi.org/10.1109/TNB.2003.820283
18. Hasan SMR (2010) A digital cmos sequential circuit model for bio-cellular adaptive immune response pathway using phagolysosomic digestion: a digital phagocytosis engine. *J Biomed Sci Eng* 3: 470–475. https://doi.org/10.4236/jbise.2010.35065
19. Hasan SMR (2008) A micro-sequenced CMOS model for cell signaling pathway using G-protein and phosphorylation cascade. *2008 15th International Conference on Mechatronics and Machine Vision in Practice,* IEEE, 2008: 57–62. https://doi.org/10.1109/MMVIP.2008.4749507
20. Hasan SMR (2008) A novel mixed-signal integrated circuit model for DNA-protein regulatory genetic circuits and genetic state machines. *IEEE Trans C*ircuits Syst *Regul Pap* 55: 1185–1196. https://doi.org/10.1109/TCSI.2008.925632
21. Alam S, Hasan SMR (2013) Integrated circuit modeling of biocellular post-transcription gene mechanisms regulated by microRNA and proteasome. *IEEE Trans Circuits Syst Regul Pap* 60: 2298–2310.https://doi.org/10.1109/TCSI.2013.2245451
22. Sarpeshkar R (2014) Analog synthetic biology. *Philos Trans R Soc Math Phys Eng Sci* 372: 20130110. https://doi.org/10.1098/rsta.2013.0110

23. Sarpeshkar R, Lu TKT, Danial R, et al. (2015) Analog and mixed-signal computation and circuits in living cells: U.S. Patent Application 14/391,817[P]. 2015-3-26.
24. Beahm DR, Deng Y, Riley TG, et al. (2021) Cytomorphic electronic systems: a review and perspective. *IEEE Nanotechnol Mag* 15: 41–53. https://doi.org/10.1109/MNANO.2021.3113192
25. Waqas M, Khurram M, Hasan SMR (2017) Bio-cellular processes modeling on silicon substrate: receptor-ligand binding and Michaelis Menten reaction. *Analog Integr Circuits Signal Process* 93: 329–340. https://doi.org/10.1007/s10470-017-1044-x
26. Waqas M, Ainuddin U, Iftikhar U (2022) An analog electronic circuit model for cAMP-dependent pathway-towards creation of Silicon life. *AIMS Bioeng* 9: 145–162. https://doi.org/10.3934/bioeng.2022011
27. Waqas M, Khurram M, Hasan SMR (2020) Analog electronic circuits to model cooperativity in hill process. *Mehran Univ Res J Eng Technol* 39: 678–685. https://doi.org/10.22581/muet1982.2004.01
28. Patra T, Chatterjee S, Barman Mandal S (2023) Cytomorphic electrical circuit modeling of tumor suppressor p53 protein pathway. *Trans Indian Natl Acad Eng* 8: 363–377. https://doi.org/10.1007/s41403-023-00403-0
29. Patra T, Dey S, Barman S (2025) Cytomorphic electronic system design of apoptosis pathway with extrinsic and intrinsic perturbations. *Comput Biol Chem* 119: 108545. https://doi.org/10.1016/j.compbiolchem.2025.108545
30. Yamaguchi Y, Azuma R, Konagaya A, et al. (2003) An approach for the high speed Monte Carlo simulation with FPGA-toward a whole cell simulation. *2003 46th Midwest Symposium on Circuits and Systems,* IEEE, 2003, 1: 364–367. https://doi.org/10.1109/MWSCAS.2003.1562294
31. Yamaguchi Y, Maruyama T, Azuma R, et al. (2007) Mesoscopic-level simulation of dynamics and interactions of biological molecules using monte carlo simulation. *J VLSI Signal Process Syst Signal Image Video Technol* 48: 287–299. https://doi.org/10.1007/s11265-007-0072-7
32. Mak TST, Rachmuth G, Lam KP, et al. (2006) A component-based FPGA design framework for neuronal ion channel dynamics simulations. *IEEE Trans Neural Syst Rehabil Eng* 14: 410–418. https://doi.org/10.1109/TNSRE.2006.886727
33. Fatima SR, Waqas M (2025) Trends of modeling bio-cellular processes and neural pathways on analog, mixed-signal and digital hardware-a review. *AIMS Bioeng* 12: 177–208. https://doi.org/10.3934/bioeng.2025008
34. Mogilner A, Wollman R, Marshall WF (2006) Quantitative modeling in cell biology: what is it good for?. *Dev Cell* 11: 279–287. https://doi.org/10.1016/j.devcel.2006.08.004
35. Igoshin OA, Goldbeter A, Kaiser D, et al. (2004) A biochemical oscillator explains several aspects of Myxococcus xanthus behavior during development. *Proc Natl Acad Sci* 101: 15760–15765. https://doi.org/10.1073/pnas.0407111101
36. Osana Y, Fukushima T, Amano H (2003) Implementation of recsip: a reconfigurable cell simulation platform. *International Conference on Field Programmable Logic and Applications. Berlin,* Heidelberg: Springer Berlin Heidelberg, 2003: 766–775. https://doi.org/10.1007/978-3-540-45234-8_74
37. Osana Y, Fukushima T, Yoshimi M, et al. (2004) An FPGA-based acceleration method for metabolic simulation. *IEICE Trans Inf Syst* 87: 2029–2037.

38. Osana Y, Fukushima T, Yoshimi M, et al. (2005) An FPGA-based, multi-model simulation method for biochemical systems. *19th IEEE International Parallel and Distributed Processing Symposium,* IEEE, 2005: 4.
39. Osana Y, Yoshimi M, Iwaoka Y, et al. (2007) ReCSiP: an FPGA-based general-purpose biochemical simulator. *Electron Commun Jpn Part II Electron* 90: 1–10. https://doi.org/10.1002/ecjb.20370
40. Amano H, Kitano N, Iwanaga N, et al. (2006) Performance evaluation of an FPGA-based biochemical simulator ReCSiP. *2006 International Conference on Field Programmable Logic and Applications,* IEEE, 2006: 1–6. https://doi.org/10.1109/FPL.2006.311327
41. Soleimani H, Drakakis EM (2017) A compact synchronous cellular model of nonlinear calcium dynamics: simulation and FPGA synthesis results. *IEEE Trans Biomed Circuits Syst* 11: 703–713. https://doi.org/10.1109/TBCAS.2016.2636183
42. Soma BM, Moumita A, Samik B, et al. (2021) FPGA implementation of different stochastic biochemical reactions involved in a cell. *2021 25th International Symposium on VLSI Design and Test (VDAT),* IEEE, 2021: 1–4. https://doi.org/10.1109/VDAT53777.2021.9601094
43. Acharya M, Dey S, Chakrabarti A, et al. (2023) FPGA based library subset design for different biochemical reactions involved in a cell. *Sādhanā* 48: 264. https://doi.org/10.1007/s12046-023-02314-w
44. Acharya M, Chakrabarti A, Dey S, et al. (2024) Designing Lab Prototype of Synthetic p53 Protein-Based Tumor Suppressor Pathway. *2024 1st International Conference on Cognitive, Green and Ubiquitous Computing (IC-CGU),* IEEE, 2024: 1–6. https://doi.org/10.1109/IC-CGU58078.2024.10530855