IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 25, 2020, accepted May 15, 2020, date of publication May 26, 2020, date of current version June 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2997765

Numerical Simulations and FPGA Implementations of Fractional-Order Systems Based on Product Integration Rules

AMR M. ABDELATY[1], MERNA ROSHDY[2], LOBNA A. SAID[2], (Senior Member, IEEE), AND AHMED G. RADWAN[3,4], (Senior Member, IEEE)

[1]Engineering Mathematics and Physics Department, Faculty of Engineering, Fayoum University, Faiyum 63514, Egypt
[2]Nanoelectronics Integrated Systems Center (NISC), Nile University, Giza 12588, Egypt
[3]Engineering Mathematics and Physics Department, Faculty of Engineering, Cairo University, Giza 12613, Egypt
[4]School of Engineering and Applied Sciences, Nile University, Giza 12588, Egypt

Corresponding author: Ahmed G. Radwan (agradwan@ieee.org)

This work was supported by the Science and Technology Development Fund (STDF) for the Project under Grant # 25977.

ABSTRACT Product integration (PI) rules are well known numerical techniques that are used to solve differential equations of integer and, recently, fractional orders. Due to the high memory dependency of the PI rules used in solving fractional-order systems (FOS), their hardware implementation is very difficult and resources-demanding. In this paper, modified versions of the PI rules are introduced to facilitate their digital implementations. The studied rules are PI rectangular, PI trapezoidal, and predict-evaluate-correct-evaluate (PECE) rules. The three modified versions of the PI rules are validated using a benchmark system of differential equations for different sizes of the memory window to show the effect of the window size on the solution accuracy. Additionally, the three modified versions of the PI rules are used to simulate a novel fractional-order chaotic system (FOCS) where its bifurcation diagrams are discussed with the window length parameter. The chaotic system is then implemented on a Field Programmable Gate Array (FPGA). There are only a few trials in literature to implement the fractional PECE algorithm on FPGA, nevertheless, the proposed FPGA realization is compared with the most recent of these trials. The FPGA implementations of the three PI rules are made using Xilinx ISE 14.7 on Artix 7 kit. The achieved throughput are 1280 Mbits/sec for PI rectangular, 128.8 Mbits/sec for PI trapezoidal, and 129.12 Mbits/sec for fractional-order PECE.

INDEX TERMS Chaotic system, FPGA, fractional calculus, fractional-order systems, product integration rules.

I. INTRODUCTION

The origins of fractional calculus date back to the beginnings of calculus itself. The idea started in a correspondence in 1695 between Leibniz and L'Hopital, where they discussed the possibility of raising the differential operator to the power of 1/2 [1], [2]. Fractional calculus is the study of non-integer order derivatives and integrals where the order can be rational, real, or even complex [1]. Over the last few decades, researchers showed a huge interest in the study of FOS due to their flexibility and their ability to model systems with memory dependency [3]. Also, another advantage of fractional-order modeling is that it regenerates closer responses to the real system while maintaining more compact mathematical formulations than the integer-order counterparts [1]. This advantage is a direct result of the additional tunability attained by introducing fractional orders as new model parameters. Engineering application of FOS includes: bioengineering [4]–[6], control [7]–[10], filters [11], [12], oscillators [13], [14], energy [15], [16], encryption [17], and chaos [18]–[20].

Chaos is an interesting phenomenon that has a quasi-random behavior [21]. It presents nonlinear dynamical systems that are highly sensitive to initial conditions [22], where a minor change in input leads to a significant difference in the output [18]. Chaos theory focuses on systems such as the stock market, weather, neural networks, biology, security, and encryption [23], [24]. A chaotic system can be either

The associate editor coordinating the review of this manuscript and approving it for publication was Jun Shen.

continuous (usually called chaotic attractors) or discrete (usually called chaotic maps) [18], [25]. Lorenz first described the chaos theory in the 1960s by presenting a butterfly attractor while working on weather prediction [26]. In 1975, Li *et al.* have stated simple chaos models for describing similar behavior [27]. Lately, new categories of chaotic systems, self-excited, and hidden attractors have been proposed in [28], [29]. Chaotic systems can be implemented using analog integrated circuits or even at the transistor level [30], [31]. However, due to the non-linearity and to avoid deviation of the chaotic systems parameters, it is preferred to digitally implement the chaotic systems. Mainly, FPGA chips have been utilized in the realization of chaotic generators. FPGAs have a vital position in digital communications and encryption systems due to attributes like reprogrammability and high speed [32].

FPGAs intuitively grant rapid prototyping for Application-Specific Integrated Circuit (ASIC) in low to medium production [33]. The time for design-implement-test-debug cycle may take only hours not months as in ASIC, therefore, the modification in the design will be easier [34]. Also, low power consumption, low cost, easy modification, real-time computing, and high capacity are inherent merits of FPGAs [33]. Additionally, FPGA is characterized by its flexibility due to the predesigned Configurable Logic Blocks (CLB) that is used in industrial applications that match the aimed requirements [35], and the parallel structure which makes it appropriate for high-speed applications, furthermore it outruns microprocessors. Besides, FPGAs clock rates work in the hundreds of MegaHertz (MHz). According to the characteristics of FPGA, it is the most suitable technology for hardware realization of complex systems, that can surpass the microprocessor by a ratio of 100 to 1 based on the system implementation [36].

Fractional-order models increase the complexity of the dynamic behavior of systems. Accordingly, the alliance of chaos theory and fractional calculus to have FOCS, makes the behavior of the systems more complex. Therefore, these systems could be used in various applications such as synchronization and control [37], encryption [38], economics [39], and signal generators [40]. Evaluating solutions of fractional differential equations (FDEs) accurately and efficiently is much more complicated than the traditional integer-order case. Also, popular software packages don't provide this option as built-in functions [41]. So researchers in this field have to write their codes to solve fractional-order differential equations numerically [41]. The persistent memory of the fractional-order differ-integral operators is one of the most difficult problems faced when coding a numerical technique that solves FDEs due to being very computationally exhaustive [42]. Based on this, the numerical methods for solving FDEs are not ready, in its original form, to be implemented on hardware platforms like micro-controllers or FPGAs. So, some manipulations and trade-offs must be made to achieve this goal, like the use of the short memory principle [43], [44].

There are different implementations for fractional-order operators on FPGA. Two different approximations for fractional-order operators to realize FOS on FPGA were presented in [36]. Another approximation to control DC motor using a fractional-order PI controller was implemented on FPGA in [45]. Hardware implementation of the Grünwald-Letnikov (GL) differ-integral was introduced in [44] for different memory sizes on FPGA. Also, based on the GL operator, FPGA implementation of integrator/differentiator was presented in [7]. Furthermore, Infinite Impulse Response (IIR) was used to implement fractional-order operators in [46]. Moreover, the LABVIEW environment was used to implement fractional-order operators using GL technique on FPGA, then connecting FPGA with an external circuit to realize fractional-order Proportional Integral Differential (FOPID) controller. Direct Torque Control (DTC) induction motor was controlled through the implemented controller on FPGA using MATLAB/Simulink tool in [47]. Also, FPGA implementation of digital controllers was realized for power electronics as Space Vector Pulse Width Modulation (SVPWM) using MATLAB/Simulink and HDL coder in [48]. In [49], chaos control of fractional-order motor using sliding mode controllers was implemented on FPGA. The generation of double-scroll chaotic attractor based on fractional-order unstable dissipative systems was introduced in [50]. Various fractional-order chaotic oscillators were implemented on FPGA using the GL method [51]. Based on the auto-regressive filter, a simple design of frequency hopping chaotic generator was implemented on FPGA in [52]. Fractional-order Liu system was realized on FPGA with topological horseshoe analysis in [53].

Most of the time domain based FPGA implementations of chaotic systems that did not use MATLAB/ Simulink code generation were based on the GL fractional differ-integral and its associated numerical method [44]. The other time domain based numerical methods, to the best of the authors' knowledge, were not discussed for FPGA implementation. Hence, this paper discusses the modification of the product integration (PI) rules to prepare them for hardware implementation by employing the short memory principle. This paper presents three different FPGA implementations of a fractional-order multi-scroll attractor based on three different product integration rules. Modified versions of the product integration rules are proposed to allow the usage of the short memory principle, which facilitates the hardware implementation of the FOS on FPGA. Three different algorithms are used to realize the multi-scroll chaotic attractor on FPGA, which are: PI rectangular, PI trapezoidal, and the fractional-order predictor-corrector scheme. The same approaches can be applied to implement other FOS on hardware.

The paper is organized as follows: section II presents the theoretical background of the numerical techniques used in solving FDEs, the proposed modified numerical techniques which are ready for FPGA implementation, and numerical validation of the proposed methods. The multi-scroll chaotic attractor used for hardware realization and FPGA

implementation with its results of the three algorithms is discussed in Section III. Finally, the concluding remarks of this work are summarized in Section IV.

II. THEORETICAL BACKGROUND

The definitions of the fractional-order differ-integrals are numerous and ever-increasing. They have recently been categorized into four classes: the classical definitions (F1), the modified definitions (F2), the local definitions (F3), and the definitions with non-singular kernel (F4) [54]. The Caputo definition belongs to the F1 class and is given as [55]:

$$ {}^{C}D^{\alpha}_{t_0} f(t) = \frac{1}{\Gamma(m-\alpha)} \int_{t_0}^{t} (t-\tau)^{m-\alpha-1} f^{(m)}(\tau) d\tau, \quad (1) $$

where $m-1 < \alpha \leq m, \quad m \in \mathbb{N}$. It can be seen as the $m-\alpha$ order fractional integral of the m-th integer order derivative of the function $f(t)$.

In practice, the Caputo definition is preferred over other definitions of the F1 class as it uses integer order initial condition in its Laplace transform while the others use fractional-order ones [55]. This is due to the fact that researchers are accustomed to measuring integer order initial conditions rather than fractional-order ones.

The initial value problem for an FDE that is described by the Caputo derivative is given by [56]:

$$ \begin{aligned} &{}^{C}D^{\alpha}_{t_0} y(t) = f(t, y(t)), \\ &y(t_0) = y_0, y'(t_0) = y_0^{(1)}, \ldots, y^{(m-1)}(t_0) = y_0^{(m-1)}, \end{aligned} \quad (2) $$

where $f(t, y(t))$ is a continuous function and y_0, $y_0^{(1)}$, $\ldots$, $y_0^{(m-1)}$ are the set of integer order initial conditions.

Applying the Riemann-Liouville fractional integral to both sides of Eqn.(2) yields [41]:

$$ y(t) = T_{m-1}[y; t_0](t) + \frac{1}{\Gamma(\alpha)} \int_{t_0}^{t} (t-\tau)^{\alpha-1} f(\tau, y(\tau)) d\tau, \quad (3) $$

where $T_{m-1}[y; t_0](t) = y_0$ for $0 < \alpha < 1$. Eqn.(2) is known as weakly singular Volterra integral equation (VIE). VIEs are a well studied topic and there are several theories and numerical techniques associated with them [41].

Step-by-step numerical techniques used for solving differential equations are either one-step or multi-step. In the former, the approximation result of the solution at only the previous step is used to evaluate the current step. Whereas, in the latter, more than one previous step is used to evaluate the current step. Fractional differential equations are known for their memory dependency, which makes choosing multi-step methods for their solutions a must. These multi-step methods can be considered a convolution quadrature given in general as [41]:

$$ y_n = \phi_n + \sum_{j=0}^{n} c_{n-j} f(t_j, y_j), \quad (4) $$

where ϕ_n and c_n are known coefficients that depend on the employed technique and $t_n = t_0 + nh$ is the equally spaced grid with $h > 0$. There are only two classes of multi-step methods that were studied in the literature: product integration rule and fractional linear multi-step methods [41]. This work is concerned with the PI rules only.

A. ORIGINAL PI RULES

Based on the PI rules, the solution of the Eqn.(3) can be written in a piece-wise manner as [56]:

$$ y(t_n) = y_0 + \frac{1}{\Gamma(\alpha)} \sum_{j=0}^{n-1} \int_{t_j}^{t_{j+1}} (t_n - \tau)^{\alpha-1} f(\tau, y(\tau)) d\tau, \quad (5) $$

where $0 < \alpha < 1$. $f(\tau, y(\tau))$ is segmented to sub-intervals $[t_j, t_{j+1}]$ and approximated using polynomials to evaluate the integral. Different choices of the polynomials leads to different PI rules. Implicit or explicit rules are implemented based on the way the integral approximation is carried out. Implicit methods involves solving nonlinear equations as y_n depends on $f(t_n, y_n)$. However, in explicit methods y_n doesn't depend on $f(t_n, y_n)$ but on $f(t_{n-1}, y_{n-1})$ and past approximates of f and y. Hence, explicit methods are better suited for hardware implementations.

If $f(\tau, y(\tau))$ in Eqn.(5) is approximated using a constant value in each sub-interval, two methods will result. The explicit PI rectangular rule is given by [41]:

$$ y_n = y_0 + h^{\alpha} \sum_{j=0}^{n-1} b^{(\alpha)}_{n-j-1} f(t_j, y_j), \quad n = 1, 2, 3, \ldots \quad (6) $$

where $b^{(\alpha)}_n = \frac{(n+1)^{\alpha} - n^{\alpha}}{\Gamma(\alpha+1)}$. The implicit version of the PI rectangular rule is given by [41]:

$$ y_n = y_0 + h^{\alpha} \sum_{j=1}^{n} b^{(\alpha)}_{n-j} f(t_j, y_j), \quad n = 1, 2, 3, \ldots \quad (7) $$

If $f(\tau, y(\tau))$ in Eqn.(5) is approximated using a first order polynomial in each sub-interval [56]:

$$ \begin{aligned} &f(\tau, y(\tau)) \\ &\approx f(t_{j+1}, y_{j+1}) \\ &\quad + \frac{\tau - t_{j+1}}{h} \left(f(t_{j+1}, y_{j+1}) - f(t_j, y_j) \right), \quad \tau \in [t_j, t_{j+1}], \end{aligned} \quad (8) $$

the implicit type trapezoidal rule will result and is given by:

$$ y_n = y_0 + h^{\alpha} \left(\tilde{a}^{(\alpha)}_n f_0 + \sum_{j=1}^{n} a^{(\alpha)}_{n-j} f(t_j, y_j) \right), \quad n = 1, 2, 3, \ldots \quad (9) $$

where

$$ \tilde{a}^{(\alpha)}_n = \frac{(n-1)^{\alpha+1} - n^{\alpha}(n-\alpha-1)}{\Gamma(\alpha+2)}, \quad (10a) $$

$$ a^{(\alpha)}_n = \begin{cases} \dfrac{1}{\Gamma(\alpha+2)} & n = 0 \\ \dfrac{(n-1)^{\alpha+1} - 2n^{\alpha+1} + (n+1)^{\alpha+1}}{\Gamma(\alpha+2)} & n = 1, 2, \ldots \end{cases} \quad (10b) $$

Formulating an explicit variant of the trapezoidal PI rule is not a straight-forward procedure and is rarely encountered in literature. One of these variants is given as [57]:

$$y_n = y_0 + h^\alpha \left(\tilde{a}_n^{(\alpha)} f_0 + \sum_{j=1}^{n-1} a_{n-j}^{(\alpha)} f(t_j, y_j) \right) + h^\alpha (-a_0 f(t_{n-2}, y_{n-2}) + 2a_0 f(t_{n-1}, y_{n-1})), \quad n \geq 2 \tag{11}$$

The predictor corrector scheme is the result of combining the rectangular PI explicit rule with the trapezoidal PI implicit rule as:

$$y_n^P = y_0 + h^\alpha \sum_{j=0}^{n-1} b_{n-j-1}^{(\alpha)} f(t_j, y_j),$$

$$y_n = y_0 + h^\alpha \left(\tilde{a}_n^{(\alpha)} f_0 + \sum_{j=1}^{n-1} a_{n-j}^{(\alpha)} f(t_j, y_j) + a_0^{(\alpha)} f(t_n, y_n^P) \right), \quad n = 1, 2, 3, \ldots \tag{12}$$

This is done to avoid solving nonlinear equations in the implicit PI rule.

B. PROPOSED FPGA READY PI RULES

The coefficients in the original version of the PI rules are heavy-tailed, which leads to an indispensable dependency on almost all the previous calculated samples of the solution. This is a huge drawback when it comes to hardware realization. So, new expressions are proposed to reduce this long memory dependency and allow the hardware to store only a relatively small window of the past estimates. The relative magnitudes of the originally used coefficients and their suggested first-order differences are shown in Fig. 1. It depicts that the magnitude of the coefficient or its first-order difference for any value of α decreases as the iteration number increases except at $\alpha = 1$ where they remain constant. Also, at the same iteration number, the magnitude of the coefficient or its first-order difference decreases as α increases.

The main idea is to evaluate the expression $y_n - y_{n-1}$ for each of the explicit versions of the original PI rules. The difference between the coefficients at iteration n and at $n-1$ is much less in magnitude than the coefficient itself. This reduces the dependency on the oldest values of y.

The proposed version of the PI rectangular rule is given by:

$$y_n = y_{n-1} + h^\alpha \sum_{j=0}^{n-2} (b_{n-j-1}^{(\alpha)} - b_{n-j-2}^{(\alpha)}) f(t_j, y_j) + h^\alpha b_0 f(t_{n-1}, y_{n-1}), \quad n \geq 2, \tag{13}$$

where the first sample, at $n = 1$, is calculated using Eqn. (6). A windowed version of the modified PI rectangular rule is given by:

$$y_n = y_{n-1} + h^\alpha \left[\sum_{j=n-2-TR}^{n-2} (b_{n-j-1}^{(\alpha)} - b_{n-j-2}^{(\alpha)}) f(t_j, y_j) \right] + h^\alpha b_0 f(t_{n-1}, y_{n-1}), \quad n \geq TR + 2, \tag{14}$$

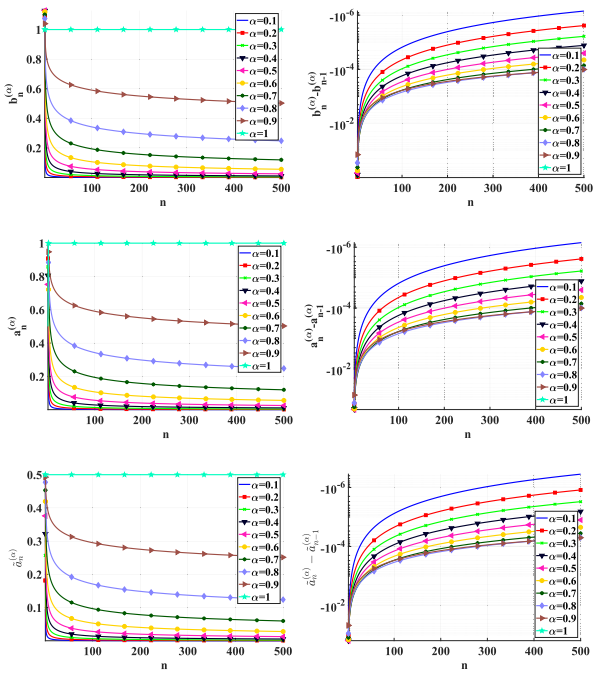


FIGURE 1. Comparison between the original coefficients (left), used in the original numerical methods, and their first order differences which are proposed for easier FPGA implementation (right).

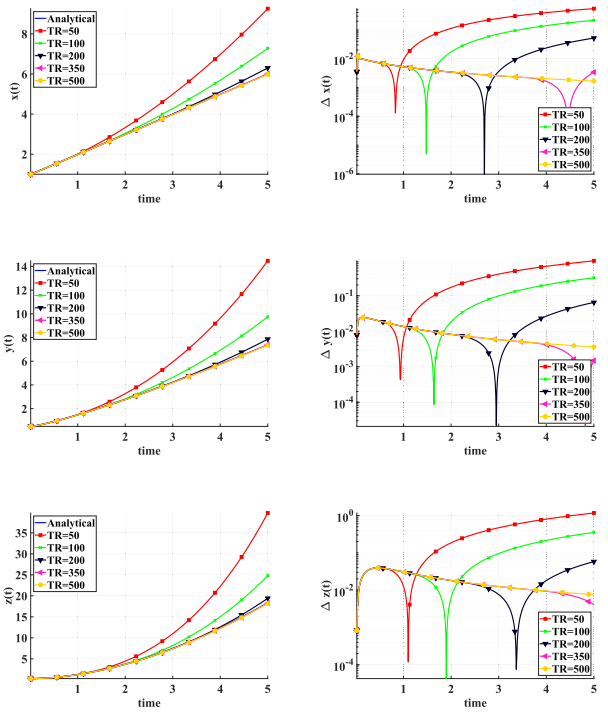


FIGURE 2. Benchmark results of Eqn.(20), numerical solutions on the left and their relative errors on the right, using the modified PI rectangular rule using step size $h = 0.01$ at different window sizes.

where TR is the window size and the first $TR + 1$ samples are calculated using Eqn. (13).

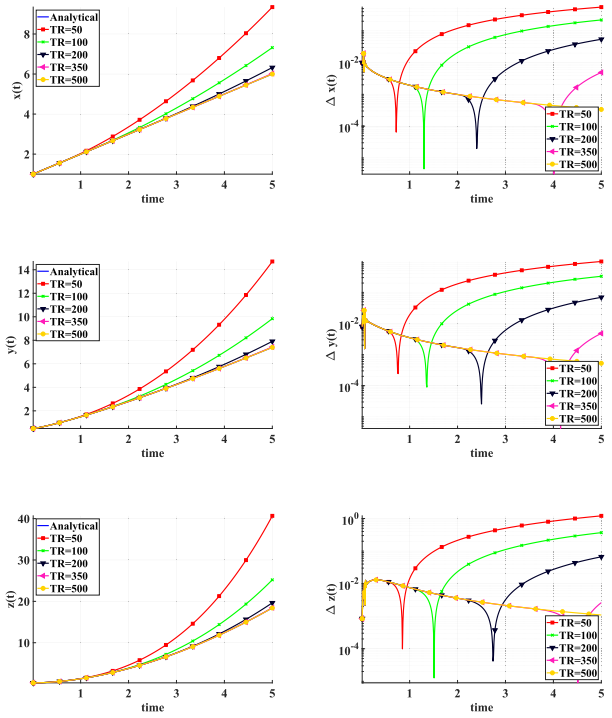


FIGURE 3. Benchmark results of Eqn.(20), numerical solutions on the left and their relative errors on the right, using the modified PI trapezoidal rule using step size $h = 0.01$ at different window sizes.

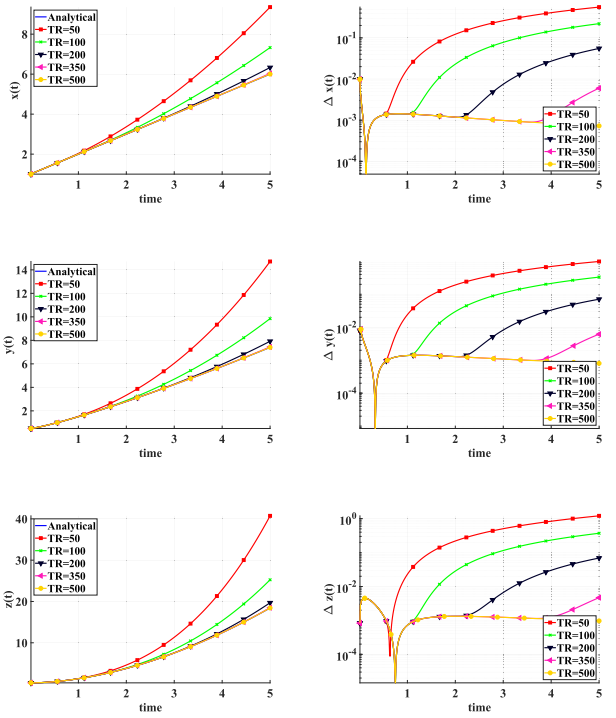


FIGURE 4. Benchmark results of Eqn.(20), numerical solutions on the left and their relative errors on the right, using the modified PECE algorithm using step size $h = 0.01$ at different window sizes.

The modified version of the PI trapezoidal rule:

$$\begin{aligned} y_n &= y_{n-1} + h^\alpha(\tilde{a}_n^{(\alpha)} - \tilde{a}_{n-1}^{(\alpha)})f_0 \\ &\quad + h^\alpha \sum_{j=1}^{n-2}(a_{n-j}^{(\alpha)} - a_{n-j-1}^{(\alpha)})f(t_j, y_j) \\ &\quad + h^\alpha a_1 f(t_{n-1}, y_{n-1}) - h^\alpha a_0(f(t_{n-2}, y_{n-2}) \\ &\quad - f(t_{n-3}, y_{n-3})) + 2h^\alpha a_0(f(t_{n-1}, y_{n-1}) \\ &\quad - f(t_{n-2}, y_{n-2})), \quad n \geq 3, \end{aligned} \tag{15}$$

where the first two samples, at $n = 1$ and $n = 2$, are calculated using Eqn. (11). A windowed version of the modified PI trapezoidal rule is given by:

$$\begin{aligned} &y_n \\ &= y_{n-1} + h^\alpha(\tilde{a}_n^{(\alpha)} - \tilde{a}_{n-1}^{(\alpha)})f_0 \\ &\quad + h^\alpha \sum_{j=n-2-TR}^{n-2}(a_{n-j}^{(\alpha)} - a_{n-j-1}^{(\alpha)})f(t_j, y_j) + h^\alpha a_1 f(t_{n-1}, y_{n-1}) \\ &\quad - h^\alpha a_0(f(t_{n-2}, y_{n-2}) - f(t_{n-3}, y_{n-3})) \\ &\quad + 2h^\alpha a_0(f(t_{n-1}, y_{n-1}) - f(t_{n-2}, y_{n-2})), \quad n \geq TR + 3, \end{aligned} \tag{16}$$

where the first $TR + 2$ samples are calculated using Eqn.(15). Due to hardware window limitations, the expression $\tilde{a}_n^{(\alpha)} - \tilde{a}_{n-1}^{(\alpha)}$ is approximated by a rational function of the form:

$$\tilde{a}_n^{(\alpha)} - \tilde{a}_{n-1}^{(\alpha)} = \frac{p_1 n + p_2}{n^2 + q_1 n + q_2}. \tag{17}$$

The rational function coefficients are extracted using MATLAB curve fitting toolbox.

As mentioned before, the predictor-corrector scheme is a combination between the explicit PI rectangular rule and the implicit PI trapezoidal rule. The modified version of the PECE is given by:

$$\begin{aligned} y_n^P &= y_{n-1} + h^\alpha \sum_{j=0}^{n-2}(b_{n-j-1}^{(\alpha)} - b_{n-j-2}^{(\alpha)})f(t_j, y_j) \\ &\quad + h^\alpha b_0 f(t_{n-1}, y_{n-1}), \\ y_n &= y_{n-1} + h^\alpha(\tilde{a}_n^{(\alpha)} - \tilde{a}_{n-1}^{(\alpha)})f_0 \\ &\quad + h^\alpha \sum_{j=1}^{n-2}(a_{n-j}^{(\alpha)} - a_{n-j-1}^{(\alpha)})f(t_j, y_j) \\ &\quad + h^\alpha a_0^{(\alpha)}(f(t_n, y_n^P) - f(t_{n-1}, y_{n-1})) + h^\alpha a_1^{(\alpha)} f(t_{n-1}, y_{n-1}), \\ &\quad n \geq 3, \end{aligned} \tag{18}$$

where the first two samples, at $n = 1$ and $n = 2$, are calculated using Eqn. (12). A windowed version of the modified predictor corrector scheme is given by:

$$\begin{aligned} y_n^P &= y_{n-1} + h^\alpha \sum_{j=n-2-TR}^{n-2}(b_{n-j-1}^{(\alpha)} - b_{n-j-2}^{(\alpha)})f(t_j, y_j) \\ &\quad + h^\alpha b_0 f(t_{n-1}, y_{n-1}), \\ y_n &= y_{n-1} + h^\alpha(\tilde{a}_n^{(\alpha)} - \tilde{a}_{n-1}^{(\alpha)})f_0 \\ &\quad + h^\alpha \sum_{j=n-1-TR}^{n-2}(a_{n-j}^{(\alpha)} - a_{n-j-1}^{(\alpha)})f(t_j, y_j) \end{aligned}$$

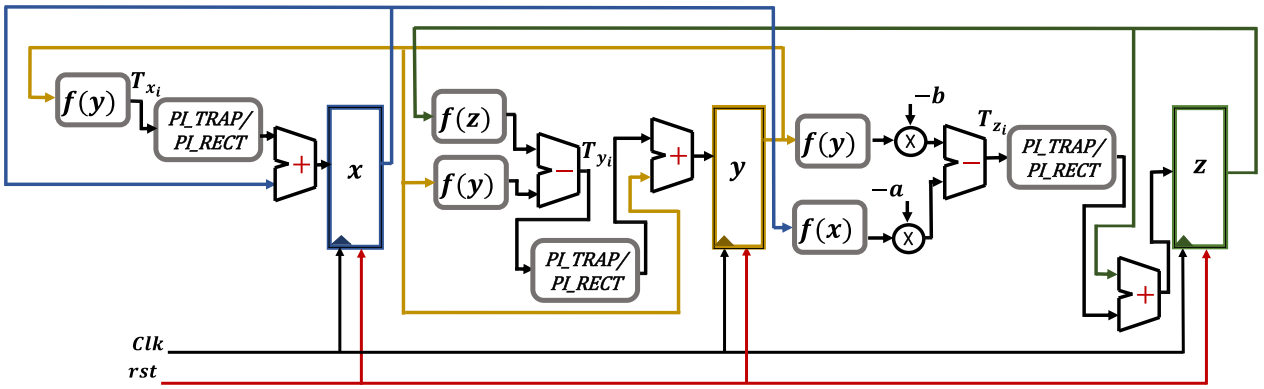


FIGURE 5. Block diagram of multi-scroll chaotic attractor.

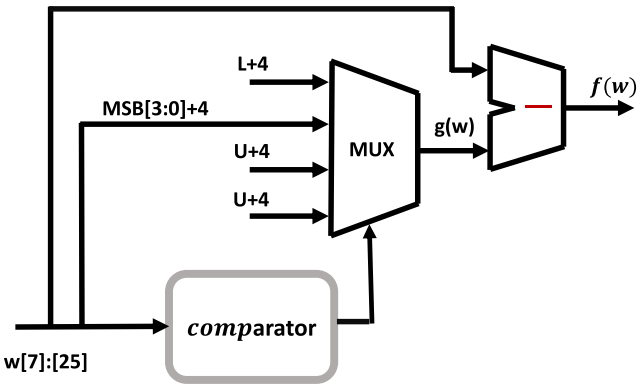


FIGURE 6. Block diagram of the non-linear function $f(w)$.

$$+h^{\alpha}a_0^{(\alpha)}(f(t_n,y_n^P)-f(t_{n-1},y_{n-1}))+h^{\alpha}a_1^{(\alpha)}f(t_{n-1},y_{n-1}),\quad n\geq TR+3, \tag{19}$$

where the first $TR+2$ samples are calculated using Eqn. (18).

C. NUMERICAL VALIDATION OF THE PROPOSED METHODS

A set of five benchmark Caputo FDEs were proposed to compare performances of different numerical algorithms [58]. From these five problems, the following is chosen to benchmark the modified PI rules:

$$\begin{aligned} {}^{C}D_0^{0.5}x(t) &= \frac{1}{\sqrt{(\pi)}}\left(\sqrt[6]{(y(t)-0.5)(z(t)-0.3)}+\sqrt{t}\right),\\ {}^{C}D_0^{0.2}y(t) &= \Gamma(2.2)(x(t)-1),\\ {}^{C}D_0^{0.6}z(t) &= \frac{\Gamma(2.8)}{\Gamma(2.2)}(y(t)-0.5), \end{aligned} \tag{20}$$

where the initial conditions are $x(0)=1.0$, $y(0)=0.5$, and $z(0)=0.3$. The analytical solutions are given by [58]:

$$x(t)=t+1,\quad y(t)=t^{1.2}+0.5,\ z(t)=t^{1.8}+0.3 \tag{21}$$

The benchmark results of Eqn. (20) using the modified numerical methods in Eqn. (14), Eqn. (16), and Eqn(19) and their relative errors are shown in Fig. 2, Fig. 3, and Fig. 4, respectively. The simulations are made between $t=0$ and $t=5$ at a step size $h=0.01$ and different window sizes namely: $TR=50$, 100, 200, 350, and 500(the total number of samples). The relative error levels vary significantly over the time interval of interest and between different implementations. The relative error levels in the modified PI rectangular implementation is higher than the modified PI trapezoidal rule, which in turn is higher than the modified PECE. In case of the modified PI rectangular and trapezoidal rules, the relative error drops down significantly around the position where the window effect takes place, at $t=h*TR$, and then rises up till it reaches its maximum value at the end of the interval ($t=5$, $n=500$), except for the case when the window is the same size as the studied time interval (window effect is removed, $TR=n$). However, in the case of the modified PECE rule, the relative error rises immediately at $t=h*TR$ without going down as in the previous two cases.

The GL based approach is not included in the comparison because it gives complex values results when it is used to simulate the benchmark system in Eqn. 20. The GL approach is a generalization of the classical Euler method used in the simulation of the integer order system [41], [56]. This can be considered one of the main drawbacks of the GL approach due to its lower accuracy levels when compared with other approaches like the FO-PECE method for example [59].

III. FPGA IMPLEMENTATION OF MULTI-SCROLL CHAOTIC OSCILLATOR

A. THEORETICAL ANALYSIS

In this section, the chaotic attractor used for hardware realization is discussed. Some of the chaotic attractors generate multi-scrolls, where they are related to a modified version of double-scroll attractors like Chua [60]. The employed chaotic attractor is defined as follows [61]:

$${}^{C}D_0^{q_1}x = f(y), \tag{22a}$$
$${}^{C}D_0^{q_2}y = f(z)-f(y), \tag{22b}$$
$${}^{C}D_0^{q_3}z = -bf(y)-af(x), \tag{22c}$$

where $q_i\in(0,1)$, $i=1,2,3$. $f(w)$ is defined as $f(w)=w-g(w)$ where $g(w)$ is defined in a piecewise-linear (PWL)

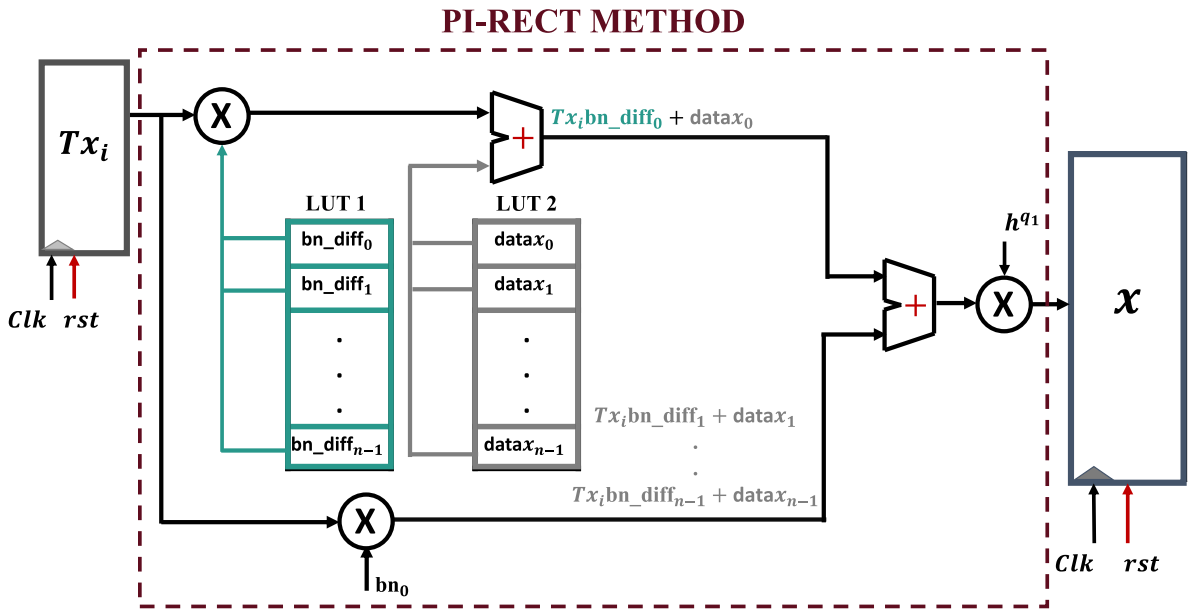


FIGURE 7. Block diagram of PI-RECT method.

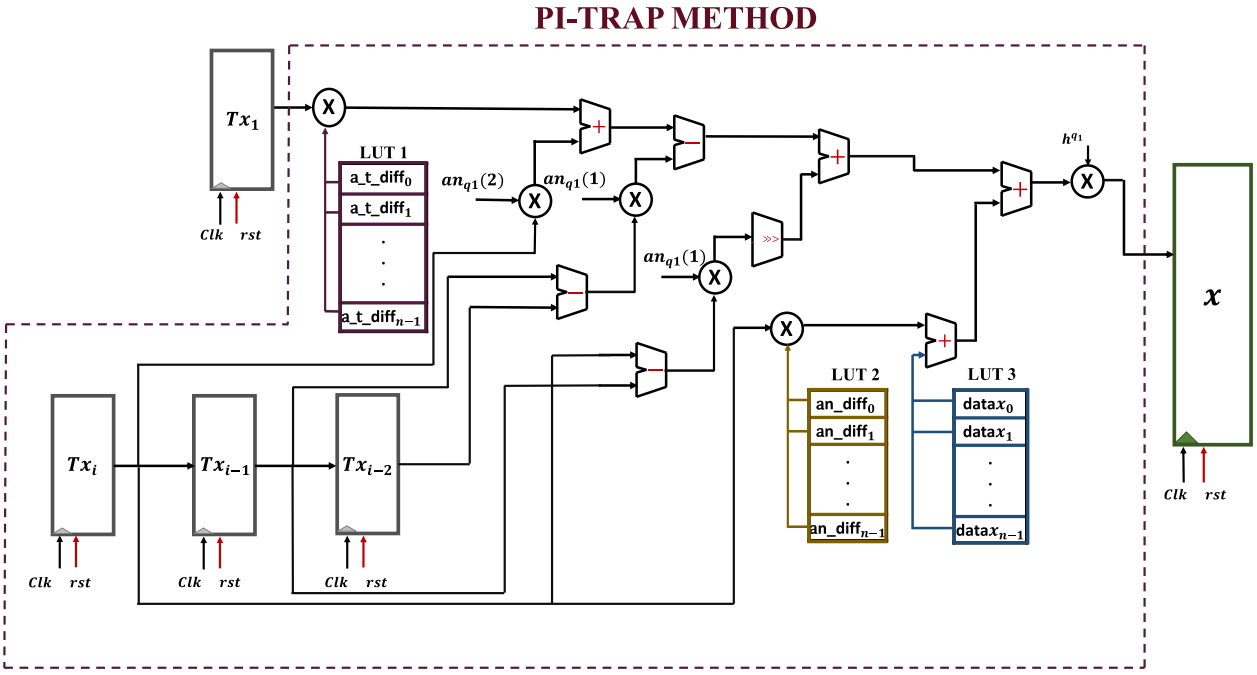


FIGURE 8. Block diagram of PI-TRAP method.

form as follows:

$$g(w) = \begin{cases} L+4, & w < L \\ 8floor(w/8)+4, & L \geq w \leq U \\ U+4, & w > U \end{cases}, \quad (23)$$

where $U = 24$ and $L = -24$ are integers multiples of eight, represent the lower and upper bound of the step function. Also, q_1, q_2, and q_3 are the non-integer orders, and $a = 1.25$ and $b = 0.75$ are the system parameters. The output of the floor function returns a maximum integer not greater than $x/8$.

The equilibrium points of this system can be found by solving [61]:

$$\begin{aligned} f(y) &= 0, \\ f(z) - f(y) &= 0, \\ -bf(y) - af(x) &= 0. \end{aligned} \quad (24)$$

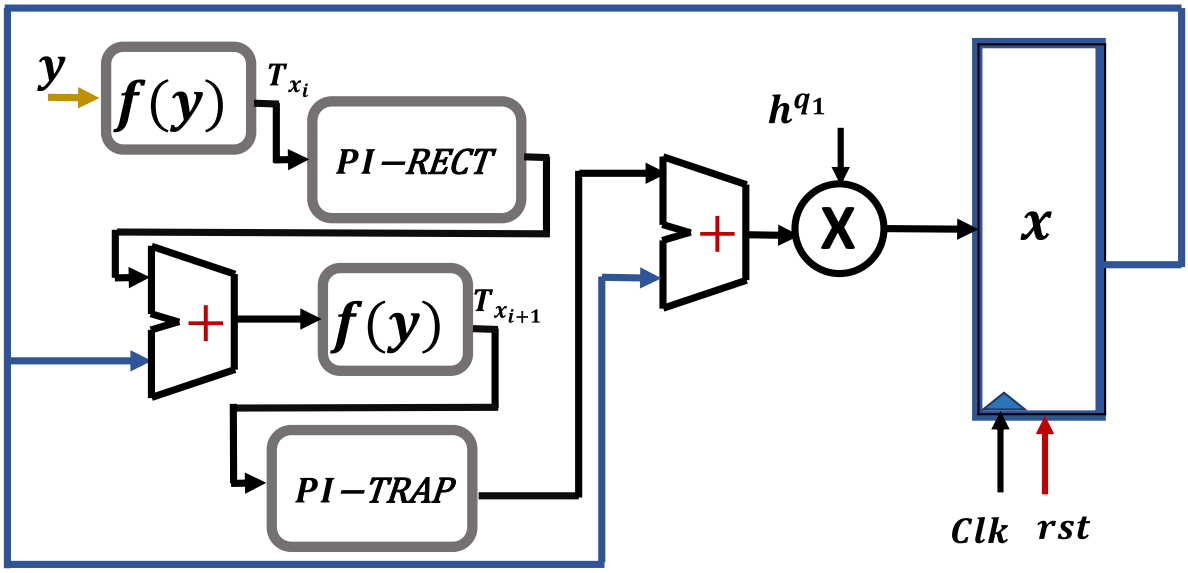


FIGURE 9. Block diagram of PI-RECT-TRAP method.

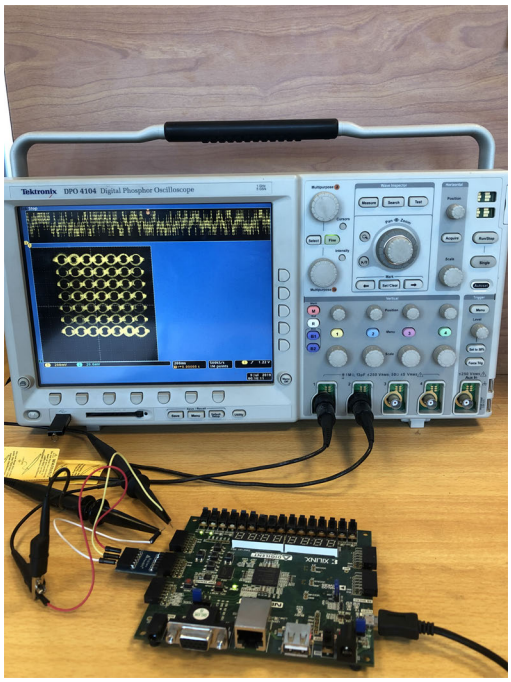


FIGURE 10. Hardware setup of PI-TRAP algorithm.

The solution is found to be [61]:

$$x^* = y^* = z^* = 8i + 4, \quad i \in \{\pm 3, \pm 2, \pm 1, 0\}. \quad (25)$$

This means that the system under investigation has 7^3 equilibrium points. These points appear in the simulations as the center points of the scrolls where the attractor rotates about but never crosses them.

The proposed system can be solved using any of the three proposed methods; Product Integration Rectangular (PI-RECT) method, Product Integration Trapezoidal (PI-TRAP) method, and Product Integration Rectangular-Trapezoidal

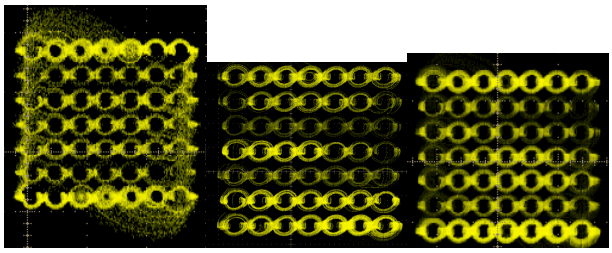


FIGURE 11. Hardware implementation of (a) PI-RECT Algorithm (b) PI-TRAP Algorithm (c) PI-RECT-TRAP Algorithm.

(PI-RECT-TRAP) method as previously discussed in Section II

There are different types of data representation, fixed-point or floating-point. Fixed-point is described as follows one bit for sign, bits for integer part and bits for the fractional part, while floating-point is divided as one bit for sign, bits for exponent and bits for mantissa [62]. Subsequently, the floating-point seems to be similar to scientific notation and more accurate, but fixed-point is commonly used for hardware implementations to save cost and increase the speed. Therefore, fixed-point representation is chosen to implement the proposed systems to save the hardware resources by maintaining the quality of the implementation.

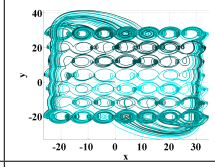
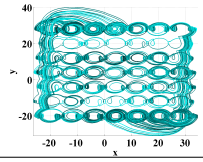
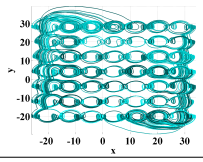
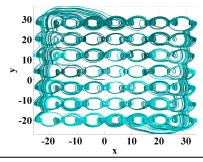
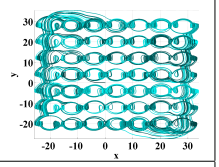
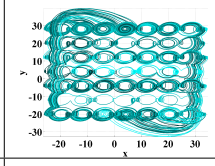
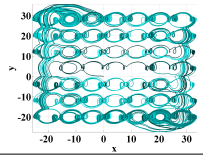
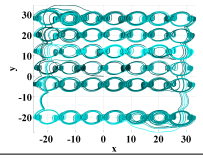
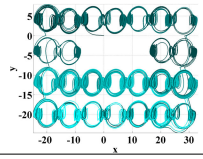
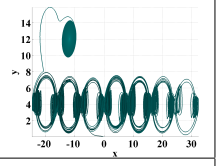
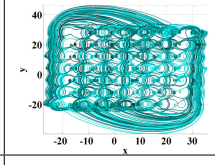
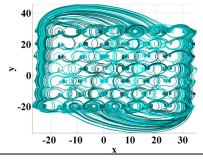
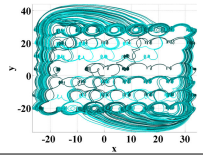
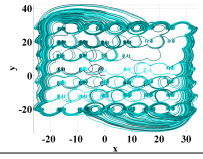
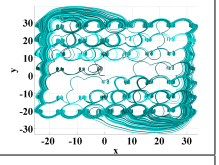
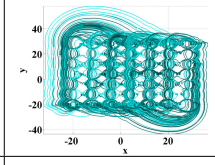
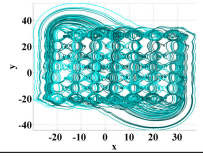
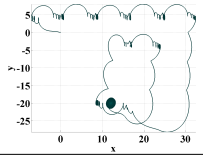
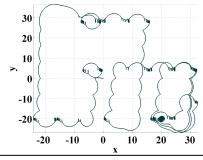
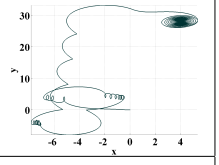
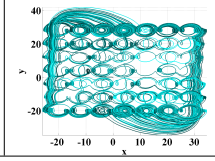
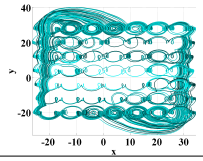
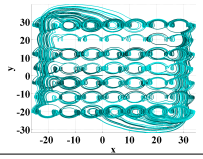
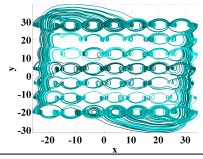
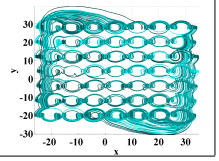
Figure 5 shows the overall block diagram describing the proposed hardware realization of the FOCS by PI-RECT or PI-TRAP methods. For implementing the proposed system on FPGA, three registers are needed to save the primary states of the system x, y, and z. Each state is computed by a combinational circuit using 32-bit fixed-point for each register, divided into 7-bit and 25-bit for the integer and fractional parts, respectively.

The nonlinear function of the system $g(w)$ is implemented in the PWL form with three pieces as described by Eqn. (23);

TABLE 1. FPGA hardware resources summary.

	PI-RECT	PI-TRAP	PI-RECT-TRAP
No of LUT	2156 out of 63400	19038 out of 63400	20961 out of 63400
No of slice registers	1305 out of 126800	2263 out of 126800	2524 out of 126800
Clock speed (MHz)	40	4.025	4.035
Throughput (Mbit/sec)	1280	128.8	129.12

TABLE 2. The impact of parameters variation on the chaotic behavior using PI-RECT Algorithm.

	TR_P=5	TR_P=10	TR_P=20	TR_P=30	TR_P=40
$\alpha = 1$ $\beta = 0.8$ $\gamma = 1$ $a = 1.1$ $b = 0.7$					
$\alpha = 0.9$ $\beta = 0.8$ $\gamma = 1$ $a = 1.1$ $b = 0.7$					
$\alpha = 0.9$ $\beta = 0.8$ $\gamma = 1$ $a = 1.5$ $b = 0.7$					
$\alpha = 0.9$ $\beta = 0.8$ $\gamma = 1$ $a = 2.5$ $b = 1.7$					
$\alpha = 1$ $\beta = 0.8$ $\gamma = 1$ $a = 1.25$ $b = 0.75$					

therefore, a multiplexer is required to select the operation, where the selection lines depend on the value of w. The hardware realization of $8floor(w/8)$ in Eqn. (23) is executed by taking the four Most Significant Bits (MSBs) of the input of the function $g(w)$ as shown in Fig.6.

B. RECTANGULAR ALGORITHM

In Fig.7, an illustration of the PI-RECT module is presented. The first values of the three main states of the system are initialized as ($x = 0.01, y = 0.01$, and $z = 0.01$). Then the second iteration is calculated as in Eqn.6

For implementing the proposed algorithm, LUT1 (see Fig. 7) is used to store the difference of $(b^{(\alpha)}_{n-j-1} - b^{(\alpha)}_{n-j-2})$ as stated by Eqn.14 labeled by *bn_diff* in Fig.7.

Tx_i is the combinational circuit required to compute the numerical solution of the system in Eqn.22. In Fig.7, Tx_i is multiplied by the coefficients from bn_diff_0 to bn_diff_{n-1}. $datax_i$ in LUT2 is used to save the values of the adder from $Tx_i bn\_diff_0 + datax_i$ to $Tx_i bn\_diff_{n-1} + datax_{n-1}$. Then, the result from the adder is added to $Tx_i bn_0$, and multiplied by h^{q_1}. The output of every clock cycle is taken as $Tx_i bn\_diff_0 + datax_0$.

C. TRAPEZOIDAL ALGORITHM

The first values of the three main states of the system are initialized as ($x\_c = 0.01$, $y\_c = 0.01$, and $z\_c = 0.01$). Then, the second and the third iterations are implemented as in Eqn.11.

Figure 8 shows the implementation of the proposed algorithm where three LUTs are employed. LUT1 is used to store the difference of $(\tilde{a}^{(\alpha)}_n - \tilde{a}^{(\alpha)}_{n-1})$, while LUT2 is used to store the difference of $(a^{(\alpha)}_{n-j} - a^{(\alpha)}_{n-j-1})$ as stated by Eqn. 16. LUT3 is used to store the values from $Tx_i an\_diff_0 + datax_i$ to $Tx_i an\_diff_{n-1} + datax_{n-1}$. Also, three registers are used to save Tx_{i-1},Tx_{i-2}, and Tx_1.

After preparing the required LUTs (LUT1, LUT2, and LUT3) and registers (Tx_i, Tx_{i-1}, and Tx_{i-2}), the combinational circuit is ready to calculate the main states of the FOCS using the trapezoidal method.

The implementation of PI trapezoidal algorithm depends on the computed Tx_i, where i represents the real-time, which makes PI-TRAP unrealizable on FPGA after the window size. Therefore, Eqn.17 is a curve fitted equation used after the

TABLE 3. The impact of parameters variation on the chaotic behavior using PI-TRAP Algorithm.

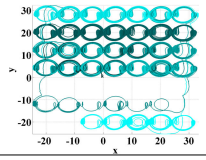
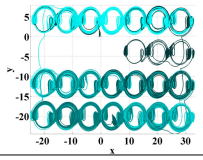
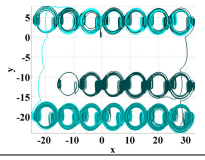
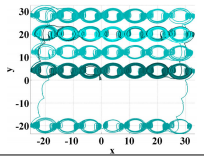
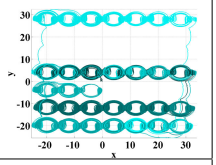
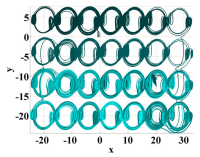
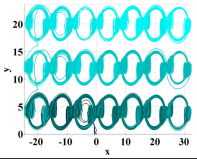
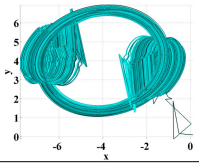
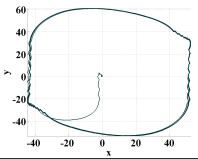
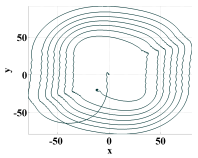
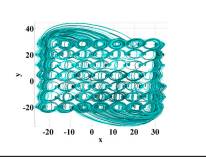
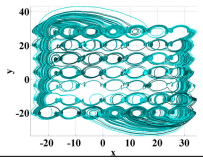
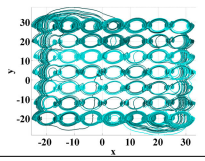
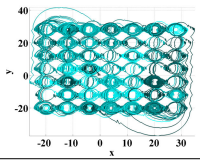
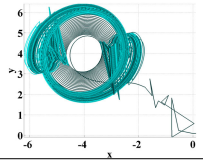
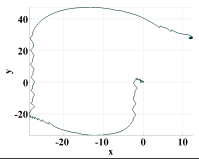
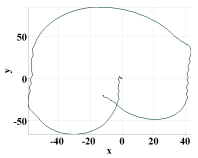
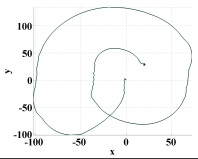
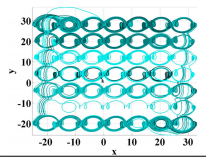
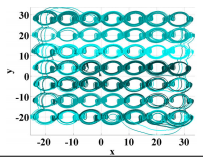
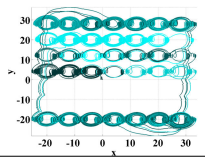
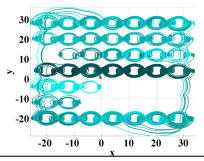
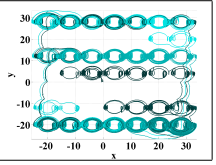
	TR_C=5	TR_C=10	TR_C=20	TR_C=30	TR_C=40
$\alpha = 1$ $\beta = 0.8$ $\gamma = 1$ $a = 1.1$ $b = 0.7$					
$\alpha = 0.9$ $\beta = 0.8$ $\gamma = 1$ $a = 1.1$ $b = 0.7$					
$\alpha = 0.9$ $\beta = 0.8$ $\gamma = 1$ $a = 1.5$ $b = 0.7$				*UNSTABLE*	*UNSTABLE*
$\alpha = 0.9$ $\beta = 0.8$ $\gamma = 1$ $a = 2.5$ $b = 1.7$					
$\alpha = 1$ $\beta = 0.8$ $\gamma = 1$ $a = 1.25$ $b = 0.75$					

TABLE 4. The impact of parameters variation on the chaotic behavior using PI-RECT-TRAP Algorithm.

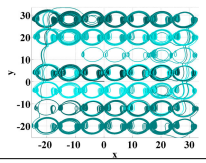
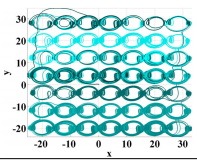
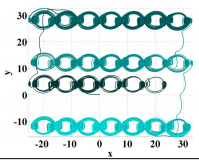
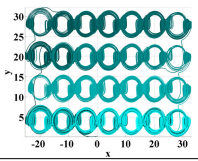
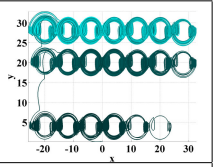
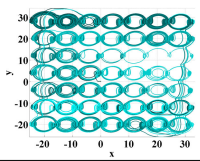
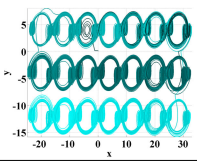
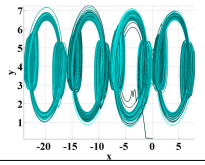
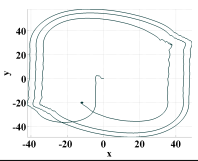
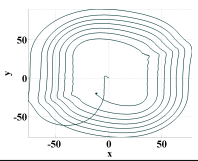
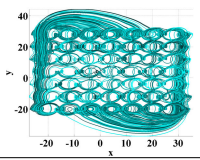
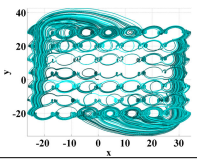
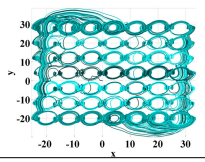
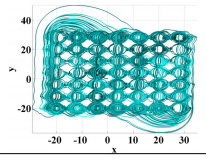
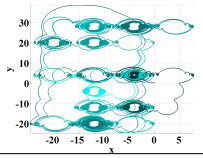
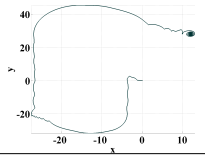
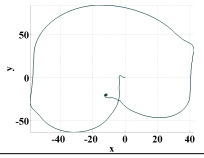
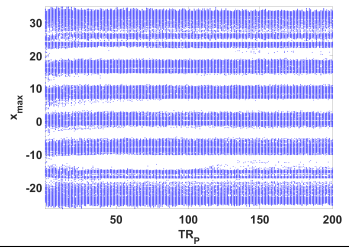
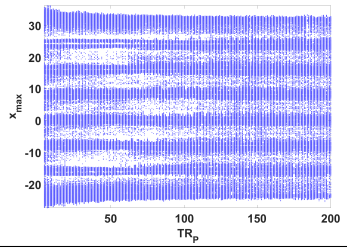
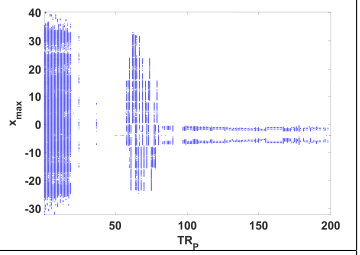
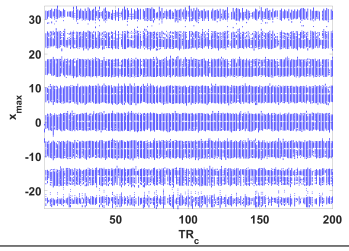
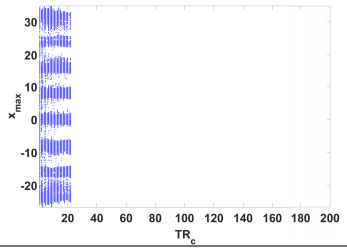
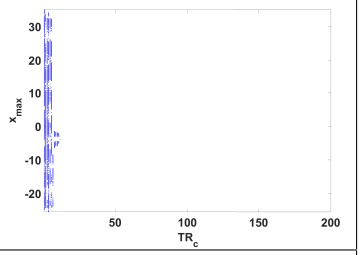
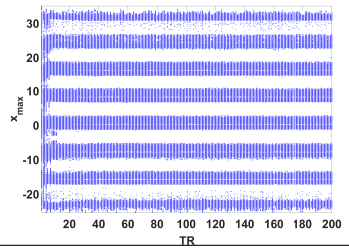
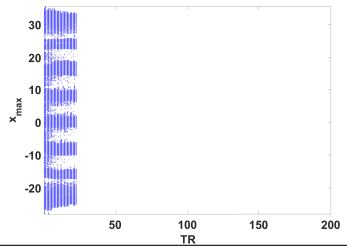
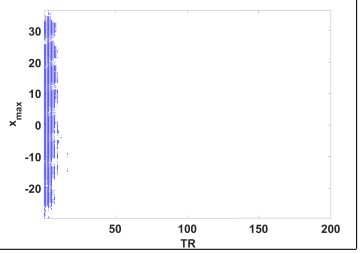
	TR_P=TR_C=5	TR_P=TR_C=10	TR_P=TR_C=20	TR_P=TR_C=30	TR_P=TR_C=40
$\alpha = 1$ $\beta = 0.8$ $\gamma = 1$ $a = 1.1$ $b = 0.7$					
$\alpha = 0.9$ $\beta = 0.8$ $\gamma = 1$ $a = 1.1$ $b = 0.7$					
$\alpha = 0.9$ $\beta = 0.8$ $\gamma = 1$ $a = 1.5$ $b = 0.7$				*UNSTABLE*	*UNSTABLE*
$\alpha = 0.9$ $\beta = 0.8$ $\gamma = 1$ $a = 2.5$ $b = 1.7$					
$\alpha = 1$ $\beta = 0.8$ $\gamma = 1$ $a = 1.25$ $b = 0.75$					

TABLE 5. Bifurcation diagrams of the three algorithms (window size is the bifurcation parameter).

	$\alpha = 1, \beta = 0.8$ $\gamma = 1, a = 1.1, b = 0.7$	$\alpha = 0.9, \beta = 0.8$ $\gamma = 1, a = 1.5, b = 0.7$	$\alpha = 0.9, \beta = 0.8$ $\gamma = 1, a = 2.5, b = 1.7$
PI-RECT			
PI-TRAP			
PI-RECT-TRAP			

window size to make the PI-TRAP algorithm applicable on FPGA.

D. RECTANGULAR-TRAPEZOIDAL ALGORITHM

The Predict Evaluate Correct Evaluate (PECE) method is the combination between the two previous algorithms, PI-Rectangular and PI-Trapezoidal. For implementing the proposed algorithm, three LUTs are needed to store the coefficients of both rectangular and trapezoidal methods. 26-bit registers are used, divided into 2-bit and 24-bit for the integer and the fractional parts, respectively.

The first values of the three main states of the system are initialized as ($x\_c = 0.1$, $y\_c = 0.1$, and $z\_c = 0.2$). The second iteration of this algorithm is calculated as in Eqn. 12. After that the algorithm is as follows: Tx_i is computed then transferred to the PI-RECT section. Then, it is added to the previously calculated value from the PI-TRAP algorithm. The calculated value then passes by the $f(y)$ block to compute the Tx_{i+1}. Last, Tx_{i+1} passes by PI-TRAP and is added to the value of the previously calculated value from the PI-TRAP algorithm (x_{i-1}).

For the PI-RECT-TRAP algorithm, $g(w)$ is implemented as a function in the same module not as a separate module as in PI-RECT and PI-TRAP algorithms. Also, in Fig. 5, the PI-RECT/PI-TRAP part, and in Fig. 9, the PI-RECT and the PI-TRAP parts, are sections from the system module not separated ones.

E. RESULTS AND DISCUSSION

The proposed algorithms (PI-RECT, PI-TRAP, and PI-RECT-TRAP) are realized using Verilog hardware description language, Xilinx ISE 14.7, Pmod DAC2, digital Oscilloscope DPO 4104, and Xilinx FPGA Artix-7 XC7A100T. Each algorithm is verified using a testbench, by importing the data from RTL simulation and plotting it using MATLAB. An Oscilloscope is used to plot the fractional-order multi-scroll chaotic attractor, where a digital to analog converter must be used to be connected to the JTAG interface of FPGA. This interface produces two serial data outputs, and each is 12-bit sent serially to the Pmod DAC2 that is connected to the oscilloscope for showing the chaotic attractor.

Figure 10 shows the hardware setup of one of the proposed algorithms while Fig. 11 presents the $x - y$ plane of the chaotic attractor generated using the proposed algorithms (PI-RECT, PI-TRAP, and PI-RECT-TRAP) for window size equal to 20. Hardware resources comparison between the three methods is presented in Table 1. Regarding the LUTs, the utilization percentages out of 63400 LUTs are 3.4%, 30%, and 33.06% for PI-RECT, PI-TRAP, and PI-RECT-TRAP, respectively. For the slice registers, the utilization percentages out of 126800 are 1.029%, 1.78%, and 1.99% for PI-RECT, PI-TRAP, and PI-RECT-TRAP, respectively. As Expected, the PI-RECT-TRAP implementation uses the most resources due to its complexity as it can be seen as a modified combination of the PI-RECT and PI-TRAP implementations.

The performance can also be measured using the throughput parameter, which represents the output data per second. It is calculated by multiplying the clock speed with the number of output bits (32-bit). Consequently, the achieved throughput are 1280 Mbits/sec for PI rectangular, 128.8 Mbits/sec for PI trapezoidal, and 129.12 Mbits/sec for fractional-order PECE.

Tables 2, 3 and 4 present the chaotic system behavior at different values of the system parameters and window sizes. The color of each graph changes from dark blue to cyan as time progresses. All simulations in these tables are made at $h = 0.0625$ and simulation duration equals to $t_f = 10000$ seconds. It can be seen that the newly introduced window size parameter can have a significant effect on the number and the size of scrolls generated during fixed simulation time and the stability of the system. The system response can change from chaotic to stable or from chaotic to unstable by only changing the window size.

Bifurcation diagrams in Table 5 are used to validate the effect of window size on the chaotic system behavior. From the window size perspective, three different cases are chosen to be discussed. For ($\alpha = 1$, $\beta = 0.8$ $\gamma = 1$, $a = 1.1$ and $b = 0.7$), the system is chaotic for all window sizes (TR, TR_P, or TR_C) from 1 to 200, which is also seen from the bifurcation diagram in the first column of Table 5. Regarding the horizontal spaces in the bifurcation diagrams, in Table 5, they represent the equilibrium points of the system that are mentioned in Eqn.25. For ($\alpha = 0.9$, $\beta = 0.8$ $\gamma = 1$, $a = 1.5$ and $b = 0.7$), the system is chaotic when it is solved by PI-RECT method. However, in the case of PI-TRAP and PI-RECT-TRAP, the system is only chaotic for a limited range of TR_C and TR. Bifurcation diagrams, in Table 5, make it more obvious that after 20, when window size equals 23, the system starts to be unstable. Finally, For ($\alpha = 0.9$, $\beta = 0.8$ $\gamma = 1$, $a = 2.5$ and $b = 1.7$), the system becomes stable as the window size increases.

When comparing the proposed design in this work with its recent counterpart introduced in [62], two important distinctions are found. The first one is the usage of memory. The authors in [62] used a memory of length 256 to represent the window size of the fractional operator while the proposed FPGA design in the current manuscript doesn't use memory at all. The second difference the latency. The total latency in [62] is 272 clock cycles which is very big compared to the latency of only 1 clock cycle observed in the proposed FPGA design.

IV. CONCLUSION

This work presented an FPGA realization of the fractional-order operator based on the product integration rules, with a modification in the PI rules to make it suitable for hardware realization. A multi-scroll chaotic attractor was realized on FPGA using the proposed methods (PI rectangular, PI trapezoidal, and predict-evaluate-correct-evaluate). The results of the systems based on the three algorithms were presented on a digital Oscilloscope. A comparison between the three algorithms from the hardware resources perspective was presented. The proposed algorithms are generic methods that can be used to realize any FOS on FPGA. Future directions might include using the proposed FPGA-ready techniques in other applications like encryption. Also, the proposed modifications can be applied to the multi-corrector numerical techniques.

ACKNOWLEDGMENT

Authors would like to thank Nile University for facilitating all procedures required to complete this study.

REFERENCES

[1] J. A. Tenreiro Machado and V. Kiryakova, "Recent history of the fractional calculus: Data and statistics," in *Basic Theory*, A. Kochubei and Y. Luchko, Eds. Berlin, Germany: De Gruyter, Feb. 2019, pp. 1–22.
[2] W. Malesza, M. Macias, and D. Sierociuk, "Analytical solution of fractional variable order differential equations," *J. Comput. Appl. Math.*, vol. 348, pp. 214–236, Mar. 2019.
[3] I. Petráš and J. Terpák, "Fractional calculus as a simple tool for modeling and analysis of long memory process in industry," *Mathematics*, vol. 7, no. 6, p. 511, Jun. 2019.
[4] D. A. Yousri, A. M. Abdelaty, L. A. Said, A. AboBakr, and A. G. Radwan, "Biological inspired optimization algorithms for cole-impedance parameters identification," *AEU-Int. J. Electron. Commun.*, vol. 78, pp. 79–89, Aug. 2017.
[5] B. M. Aboalnaga, L. A. Said, A. H. Madian, A. S. Elwakil, and A. G. Radwan, "Cole bio-impedance model variations in *daucus carota sativus* under heating and freezing conditions," *IEEE Access*, vol. 7, pp. 113254–113263, 2019.
[6] A. AboBakr, L. A. Said, A. H. Madian, A. S. Elwakil, and A. G. Radwan, "Experimental comparison of integer/fractional-order electrical models of plant," *AEU-Int. J. Electron. Commun.*, vol. 80, pp. 1–9, Oct. 2017.
[7] M. F. Tolba, L. A. Said, A. H. Madian, and A. G. Radwan, "FPGA implementation of the fractional order integrator/differentiator: Two approaches and applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 4, pp. 1484–1495, Apr. 2019.
[8] M. R. Homaeinezhad and A. Shahhosseini, "High-performance modeling and discrete-time sliding mode control of uncertain non-commensurate linear time invariant MIMO fractional order dynamic systems," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 84, May 2020, Art. no. 105200.
[9] R. Shalaby, M. El-Hossainy, and B. Abo-Zalam, "Fractional order modeling and control for under-actuated inverted pendulum," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 74, pp. 97–121, Jul. 2019.
[10] Y. Xu, Q. Li, and W. Li, "Periodically intermittent discrete observation control for synchronization of fractional-order coupled systems," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 74, pp. 219–235, Jul. 2019.
[11] E. M. Hamed, A. M. Abdelaty, L. A. Said, and A. G. Radwan, "Effect of different approximation techniques on fractional-order KHN filter design," *Circuits, Syst., Signal Process.*, vol. 37, no. 12, pp. 5222–5252, Dec. 2018.
[12] N. A. Khalil, L. A. Said, A. G. Radwan, and A. M. Soliman, "Generalized two-port network based fractional order filters," *AEU-Int. J. Electron. Commun.*, vol. 104, pp. 128–146, May 2019.
[13] O. Elwy, L. A. Said, A. H. Madian, and A. G. Radwan, "All possible topologies of the fractional-order wien oscillator family using different approximation techniques," *Circuits, Syst., Signal Process.*, vol. 38, no. 9, pp. 3931–3951, Feb. 2019.
[14] L. A. Said, A. G. Radwan, A. H. Madian, and A. M. Soliman, "Three fractional-order-capacitors-based oscillators with controllable phase and frequency," *J. Circuits, Syst. Comput.*, vol. 26, no. 10, Oct. 2017, Art. no. 1750160.
[15] A. Allagui, T. J. Freeborn, A. S. Elwakil, M. E. Fouda, B. J. Maundy, A. G. Radwan, Z. Said, and M. A. Abdelkareem, "Review of fractional-order electrical characterization of supercapacitors," *J. Power Sources*, vol. 400, pp. 457–467, Oct. 2018.
[16] A. M. Abdelaty, A. G. Radwan, A. S. Elwakil, and C. Psychalinos, "Transient and steady-state response of a fractional-order dynamic PV model under different loads," *J. Circuits, Syst. Comput.*, vol. 27, no. 02, Feb. 2018, Art. no. 1850023.
[17] S. M. Ismail, L. A. Said, A. A. Rezk, A. G. Radwan, A. H. Madian, M. F. Abu-Elyazeed, and A. M. Soliman, "Generalized fractional logistic map encryption system based on FPGA," *AEU-Int. J. Electron. Commun.*, vol. 80, pp. 114–126, Oct. 2017.

[18] A. M. Abdelaty, A. T. Azar, S. Vaidyanathan, A. Ouannas, and A. G. Radwan, "Applications of continuous-time fractional order chaotic systems," in *Mathematical Techniques of Fractional Order Systems*. Amsterdam, The Netherlands: Elsevier, 2018, pp. 409–449.

[19] M. Chen, S.-Y. Shao, P. Shi, and Y. Shi, "Disturbance-observer-based robust synchronization control for a class of fractional-order chaotic systems," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 4, pp. 417–421, Apr. 2017.

[20] N. S. Soliman, M. F. Tolba, L. A. Said, A. H. Madian, and A. G. Radwan, "Fractional X-shape controllable multi-scroll attractor with parameter effect and FPGA automatic design tool software," *Chaos, Solitons Fractals*, vol. 126, pp. 292–307, Sep. 2019.

[21] G. Williams, *Chaos Theory Tamed*. Boca Raton, FL, USA: CRC Press, 1997.

[22] B. Muthuswamy and S. Banerjee, *A Route to Chaos Using FPGAs*. Cham, Switzerland: Springer, 2015.

[23] S. H. Strogatz, *Nonlinear Dynamics And Chaos: With Applications To Physics, Biology, Chemistry, And Engineering*. Boca Raton, FL, USA: CRC Press, 2018.

[24] M. A. Zidan, A. G. Radwan, and K. N. Salama, "Random number generation based on digital differential chaos," in *Proc. IEEE 54th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2011, pp. 1–4.

[25] A. G. Radwan, "On some generalized discrete logistic maps," *J. Adv. Res.*, vol. 4, no. 2, pp. 163–171, Mar. 2013.

[26] E. N. Lorenz, "Deterministic nonperiodic flow," *J. Atmos. Sci.*, vol. 20, no. 2, pp. 130–141, Mar. 1963.

[27] T.-Y. Li and J. A. Yorke, "Period three implies chaos," *Amer. Math. Monthly*, vol. 82, no. 10, pp. 985–992, Dec. 1975.

[28] G. A. Leonov and N. V. Kuznetsov, "Hidden attractors in dynamical systems. from hidden oscillations in Hilbert–Kolmogorov, Aizerman, and Kalman problems to hidden chaotic attractor in Chua circuits," *Int. J. Bifurcation Chaos*, vol. 23, no. 01, Jan. 2013, Art. no. 1330002.

[29] A. T. Azar, C. Volos, N. A. Gerodimos, G. S. Tombras, V.-T. Pham, A. G. Radwan, S. Vaidyanathan, A. Ouannas, and J. M. Munoz-Pacheco, "A novel chaotic system without equilibrium: Dynamics, synchronization, and circuit realization," *Complexity*, vol. 2017, pp. 1–11, 2017.

[30] A. G. Radwan, A. M. Soliman, and A.-L. El-Sedeek, "An inductorless CMOS realization of Chua's circuit," *Chaos, Solitons Fractals*, vol. 18, no. 1, pp. 149–158, Sep. 2003.

[31] A. G. Radwan, A. M. Soliman, and A. El-Sedeek, "MOS realization of the double-scroll-like chaotic equation," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 50, no. 2, pp. 285–288, Feb. 2003.

[32] M. Tuna, M. Alçın, İ. Koyuncu, C. B. Fidan, and İ. Pehlivan, "High speed FPGA-based chaotic oscillator design," *Microprocessors Microsyst.*, vol. 66, pp. 72–80, Apr. 2019.

[33] K. Rajagopal, A. Akgul, S. Jafari, A. Karthikeyan, and I. Koyuncu, "Chaotic chameleon: Dynamic analyses, circuit implementation, FPGA design and fractional-order form with basic analyses," *Chaos, Solitons Fractals*, vol. 103, pp. 476–487, Oct. 2017.

[34] W. J. MacLean, "An evaluation of the suitability of FPGAs for embedded vision systems," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)-Workshops*, Sep. 2005, p. 131.

[35] E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan, and M. W. Naouar, "FPGAs in industrial control applications," *IEEE Trans. Ind. Informat.*, vol. 7, no. 2, pp. 224–243, May 2011.

[36] C. X. Jiang, J. E. Carletta, and T. T. Hartley, "Implementation of fractional-order operators on field programmable gate arrays," in *Advances in Fractional Calculus*. Dordrecht, The Netherlands: Springer, 2007, pp. 333–346.

[37] S. Vaidyanathan, "Generalized projective synchronization of vaidyanathan chaotic system via active and adaptive control," in *Advances and Applications in Nonlinear Control Systems*. Cham, Switzerland: Springer, 2016, pp. 97–116.

[38] P. Muthukumar and P. Balasubramaniam, "Feedback synchronization of the fractional order reverse butterfly-shaped chaotic system and its application to digital cryptography," *Nonlinear Dyn.*, vol. 74, no. 4, pp. 1169–1181, Dec. 2013.

[39] I. Tejado, D. Valério, E. Pérez, and N. Valério, "Fractional calculus in economic growth modelling: The spanish and portuguese cases," *Int. J. Dyn. Control*, vol. 5, no. 1, pp. 208–222, Mar. 2017.

[40] K. T. Alligood, T. D. Sauer, and J. A. Yorke, *Chaos*. New York, NY, USA: Springer, 1996.

[41] R. Garrappa, "Numerical solution of fractional differential equations: A survey and a software tutorial," *Mathematics*, vol. 6, no. 2, p. 16, Jan. 2018.

[42] G. E. Karniadakis, *Numerical Methods*, vol. 3. Berlin, Germany: De Gruyter, 2019.

[43] I. Podlubny, *Fractional Differential Equations: An Introduction to Fractional Derivatives, Fractional Differential Equations, to Methods of Their Solution and Some of Their Applications*, vol. 198. Amsterdam, The Netherlands: Elsevier, 1998.

[44] M. F. Tolba, A. M. AbdelAty, N. S. Soliman, L. A. Said, A. H. Madian, A. T. Azar, and A. G. Radwan, "FPGA implementation of two fractional order chaotic systems," *AEU-Int. J. Electron. Commun.*, vol. 78, pp. 162–172, Aug. 2017.

[45] C. I. Muresan, S. Folea, G. Mois, and E. H. Dulf, "Development and implementation of an FPGA based fractional order controller for a DC motor," *Mechatronics*, vol. 23, no. 7, pp. 798–804, Oct. 2013.

[46] C. X. Jiang, J. E. Carletta, T. T. Hartley, and R. J. Veillette, "A systematic approach for implementing fractional-order operators and systems," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 3, no. 3, pp. 301–312, Sep. 2013.

[47] F. Ricci and H. Le-Huy, "Modeling and simulation of FPGA-based variable-speed drives using simulink," *Math. Comput. Simul.*, vol. 63, nos. 3–5, pp. 183–195, Nov. 2003.

[48] Y. P. Siwakoti and G. E. Town, "Design of FPGA-controlled power electronics and drives using MATLAB simulink," in *Proc. IEEE ECCE Asia Downunder*, Jun. 2013, pp. 571–577.

[49] K. Rajagopal, F. Nazarimehr, A. Karthikeyan, A. Srinivasan, and S. Jafari, "Fractional order synchronous reluctance motor: Analysis, chaos control and FPGA implementation," *Asian J. Control*, vol. 20, no. 5, pp. 1979–1993, Sep. 2018.

[50] E. Zambrano-Serrano, J. M. Muñoz-Pacheco, and E. Campos-Cantón, "Chaos generation in fractional-order switched systems and its digital implementation," *AEU-Int. J. Electron. Commun.*, vol. 79, pp. 43–52, Sep. 2017.

[51] A. D. Pano-Azucena, B. Ovilla-Martinez, E. Tlelo-Cuautle, J. Manuel Muñoz-Pacheco, and L. G. de la Fraga, "FPGA-based implementation of different families of fractional-order chaotic oscillators applying Grünwald–Letnikov method," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 72, pp. 516–527, Jun. 2019.

[52] L. Cong and W. Xiaofu, "Design and realization of an FPGA-based generator for chaotic frequency hopping sequences," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 48, no. 5, pp. 521–532, May 2001.

[53] E. Dong, M. Yuan, F. Han, J. Tong, and S. Du, "Topological horseshoe analysis and FPGA implementation of a classical fractional order chaotic system," *IEEE Access*, vol. 7, pp. 129095–129103, 2019.

[54] G. Sales Teodoro, J. A. Tenreiro Machado, and E. Capelas de Oliveira, "A review of definitions of fractional derivatives and other operators," *J. Comput. Phys.*, vol. 388, pp. 195–208, Jul. 2019.

[55] A. A. Kilbas, H. M. Srivastava, and J. J. Trujillo, *Theory and Applications of Fractional Differential Equations* (North-Holland Mathematics Studies), vol. 204. Amsterdam, The Netherlands: Elsevier, 2006.

[56] R. Garrappa, "Trapezoidal methods for fractional differential equations: Theoretical and computational aspects," *Math. Comput. Simul.*, vol. 110, pp. 96–112, Apr. 2015.

[57] R. Garrappa, "On linear stability of predictor–corrector algorithms for fractional differential equations," *Int. J. Comput. Math.*, vol. 87, no. 10, pp. 2281–2290, Aug. 2010.

[58] D. Xue and L. Bai, "Benchmark problems for caputo fractional-order ordinary differential equations," *Fractional Calculus Appl. Anal.*, vol. 20, no. 5, pp. 1305–1312, Jan. 2017.

[59] K. Diethelm, "Fundamental approaches for the numerical handling of fractional operators and time-fractional differential equations," in *Numerical Methods*, G. E. Karniadakis, Ed. Berlin, Germany: De Gruyter, Feb. 2019, pp. 1–22.

[60] C. H. Wang, H. Xu, and F. Yu, "A novel approach for constructing high-order Chua's circuit with multi-directional multi-scroll chaotic attractors," *Int. J. Bifurcation Chaos*, vol. 23, no. 2, Feb. 2013, Art. no. 1350022.

[61] D. Chang, Z. Li, M. Wang, and Y. Zeng, "A novel digital programmable multi-scroll chaotic system and its application in FPGA-based audio secure communication," *AEU-Int. J. Electron. Commun.*, vol. 88, pp. 20–29, May 2018.

[62] E. T. Cuautle, A. D. Pano-Azucena, O. Guillén-Fernández, and A. Silva-Juárez, *Analog/Digital Implementation of Fractional Order Chaotic Circuits and Applications*. Cham, Switzerland: Springer, 2019.