Int. J. Electron. Commun. (AEÜ) 78 (2017) 162–172

Contents lists available at ScienceDirect

International Journal of Electronics and Communications (AEÜ)

journal homepage: www.elsevier.com/locate/aeue

Regular paper

FPGA implementation of two fractional order chaotic systems

Mohammed F. Tolba [a], Amr M. AbdelAty [b], Nancy S. Soliman [a], Lobna A. Said [a,*], Ahmed H. Madian [a,c], Ahmad Taher Azar [a,d], Ahmed G. Radwan [a,e]

[a] NISC Research Center, Nile University, Cairo, Egypt
[b] Dept. of Engineering Mathematics and Physics, Fayoum University, Egypt
[c] Radiation Engineering Dept., NCRRT, Egyptian Atomic Energy, Authority, Egypt
[d] Faculty of Computers and Information, Benha University, Egypt
[e] Dept. of Engineering Mathematics and Physics, Cairo University, Egypt

ARTICLE INFO

Article history:
Received 14 February 2017
Accepted 24 April 2017

Keywords:
Chaotic systems
FPGA
Fractional order
Liu system
Multi-scroll
V-Shape

ABSTRACT

This paper discusses the FPGA implementation of the fractional-order derivative as well as two fractional-order chaotic systems where one of them has controllable multi-scroll attractors. The complete hardware architecture of the Grünwald-Letnikov (GL) differ-integral is realized with different memory window sizes. As an application of the proposed circuit, a complete fractional-order FPGA implementation of Liu chaotic system is introduced with different fractional-orders. Moreover, a fractional-order controllable heart and V-shape multi-scrolls chaotic systems are verified in the case of symmetric and asymmetric cases. Different interesting attractors are realized under various parametric changes with distinct step sizes for different fractional-orders. To verify the chaotic behavior of many generating attractors, the Maximum Lyapunov Exponent (MLE) is calculated for such systems. The designs have been simulated using Xilinx ISE 14.5 and realized on Xilinx FPGA Virtex 5 *XC5VLX50T*. The achieved throughputs are: 4.4 Gbit/s for GL, 1.986 Gbit/s for Liu system, and 2.921 Gbit/s for V-Shape multi-scroll attractor.

1. Introduction

In 1960s, Lorenz has described the chaos phenomenon by introducing the butterfly attractor which is highly sensitive to initial conditions [1]. A small change in the system parameters leads to a huge difference in its behavior. Inspired by Lorenz attractor, many researchers developed different chaotic attractors and implemented simple circuits for exhibiting these nonlinear dynamical phenomena [2,3]. Chaos emerges in many applications such as signal generators [4], synchronization [5], random number generator [6,7], communication security [8] and others [9,10]. Nowadays, there is a great need to have secure data especially in IOT related applications which inspired the researchers to develop more complicated chaotic systems [8]. Multi-scrolls attractor [11,12] is considered one of these complicated systems. Also, more complexity is achieved by generalizing the conventional integer-order systems into the fractional-order domain [13]. One of the advantages of fractional-order chaotic systems is the added parameters that can be used to extend the encryption key.

Before fractional calculus was introduced, system modeling was confined to the integer order domain. This restriction often caused either imprecise models of low order or extremely complex higher order ones. However, these exact systems could now be described with higher accuracy and fewer number of parameters. Also, the non-locality property of most fractional calculus operators allowed modeling of systems exhibiting memory dependency more accurately. Moreover, variable order fractional operators add more degrees of freedom to model systems with changing models in time, space or both. All these interesting features inspired researcher to utilize this new tool in many areas such as: control [14,15], bioengineering [16,17], PV modeling [18], analog filters [19–23], oscillators [24–26], stability analysis [27], Smith chart [28], chaotic systems [29], super-capacitor modeling [30] and viscoelasticity [31].

Fractional-order systems are complicated to translate into hardware due to their memory dependency that requires the use of high-order integer order systems. Field Programmable Gate Array (FPGA) technology is suitable for implementing complex systems. That is why hardware implementation of fractional-order differentiators and integrators requires careful consideration of some issues such as system quality, hardware cost and speed. Recent approaches in technology have made digital hardware

* Corresponding author.
E-mail address: L.a.Said@ieee.org (L.A. Said).

http://dx.doi.org/10.1016/j.aeue.2017.04.028
1434-8411/

implementations less expensive, faster, and easier to design. The FPGA is an efficient platform to implement high quality, high throughput approximations to fractional order systems that are low in cost and require only short design times [32].

There have been several trials in the literature to implement fractional order operators on FPGAs [32–36]. A high order Finite Impulse Response (FIR) filter or a sum of first-order Infinite Impulse Response (IIR) sections were used to implement the fractional order operator in [32–34]. In [35], LabVIEW was used for the implementation of fractional order integrator/differentiator on FPGA. The design based on LabVIEW does not optimize the hardware resources on FPGA. Another FPGA implementation was introduced in [36] based on the short memory principle by Podlubny [37].

The aim of this work is to provide a more compact hardware implementation of the Grünwald-Letnikov's (GL) fractional order differ-integral operator and then use it as a building block to simulate systems of fractional order chaotic differential equations efficiently on FPGAs. The paper also aims to investigate the effect of the fractional order, memory window length and other parameters on the MLE of the implemented systems. These systems are established on Xilinx FPGA Virtex 5 *XC5VLX50T*. The proposed design utilizes the parallel structure and the flexibility of FPGAs to produce high-performance and yet low-cost implementations with faster delay times than the ones reported in literature.

This paper is organized as follows, Section 2 presents a summary of the GL fractional order operator and its FPGA implementation. Liu system discussion and hardware implementation are introduced in Section 3. Controllable V-shape multi-scroll system discussion and the circuit are illustrated in Section 4, and finally Section 5 concludes the work.

2. Implementation of fractional order derivative

2.1. Grünwald-Letnikov's theoretical analysis

The GL definition of the fractional order derivative is [37]:

$$ {}_a^{GL}D_t^{\alpha} f(t) = \lim_{h\to 0} \frac{1}{h^{\alpha}} \sum_{j=0}^{\lfloor (t-a)/h \rfloor} w_j^{(\alpha)} f(t-jh), \tag{1} $$

where $w_j^{(\alpha)}$ are the binomial coefficients, calculated recursively as:

$$ w_0^{(\alpha)} = 1, \quad w_j^{(\alpha)} = \left(1 - \frac{\alpha+1}{j}\right) w_{j-1}^{(\alpha)}, \quad j = 1,2,3,\ldots. \tag{2} $$

In order to implement this operator on the FPGA, an approximate FIR version of length L is given by:

$$ {}_{t-L}^{GL}D_t^{\alpha} f(t) = \frac{1}{h^{\alpha}} \sum_{j=0}^{L} w_j^{(\alpha)} f(t-jh), \tag{3} $$

where h is the step size, L is the window size. According to the short memory principle, the error in calculating this approximated derivative is bounded by [37]:

$$ \Delta(t) = \left| {}_a^{GL}D_t^{\alpha} f(t) - {}_{t-L}^{GL}D_t^{\alpha} f(t) \right| \leqslant \frac{ML^{-\alpha}}{|\Gamma(1-\alpha)|}, \tag{4} $$

where $(a+L<t<b)$ and $|f(t)|<M$ when $a<t<b$. So, it can be inferred that the error is reduced by increasing the window size. Fig. 1 shows how the magnitude of the $w_j^{(\alpha)}$ decreases with increasing j. It can be seen that the dependency on past values of the function decreases as the order α approaches the integer value.

The general form of a fractional order system of three differential equations is given as:

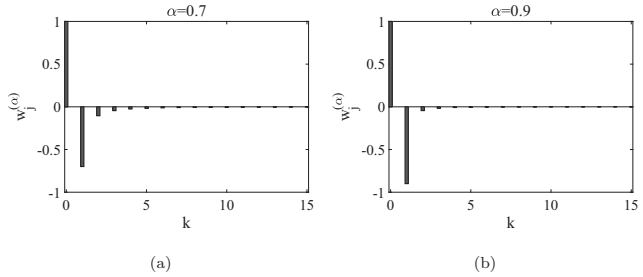

Fig. 1. The value of the binomial coefficient weights at (a) $\alpha = 0.7$ and (b) $\alpha = 0.9$.

$$ D^{q_1}x = P(x,y,z,t), \tag{5a} $$
$$ D^{q_2}y = Q(x,y,z,t), \tag{5b} $$
$$ D^{q_3}z = R(x,y,z,t). \tag{5c} $$

To simulate this system based on GL definition, the following set of equations are used [13]:

$$ x_{t_k} = P(x(t_{k-1}), y(t_{k-1}), z(t_{k-1}))h^{q_1} - \sum_{j=1}^{m} w_j^{(q_1)} x(t_{k-j}), \tag{6a} $$
$$ y_{t_k} = Q(x(t_{k-1}), y(t_{k-1}), z(t_{k-1}))h^{q_2} - \sum_{j=1}^{m} w_j^{(q_2)} y(t_{k-j}), \tag{6b} $$
$$ z_{t_k} = R(x(t_{k-1}), y(t_{k-1}), z(t_{k-1}))h^{q_3} - \sum_{j=1}^{m} w_j^{(q_3)} z(t_{k-j}), \tag{6c} $$

where $m = L$ for the approximated window variation of GL operator and $m = k$ when the entire state memory is used in calculations. This approach is quite similar to Euler method for integer order DEs.

2.2. Grünwald-Letnikov's FPGA implementation

GL equation consists of two parts, the first is the binomial coefficients in (2), which is implemented by using look up table (LUT) to store the coefficients. The second part depicted in Fig. 2 is the dot product of the row and column vectors presented by (3). From Fig. 2, each input is multiplied by all the binomial coefficients. The implementation of the proposed GL design is explained through the flowchart depicted in Fig. 3. Two LUTs are required, the first one stores the binomial coefficients w_0 to w_n, and the second one stores the output of the addition from $V_{in}w_1 + d_1$ to $V_{in}w_n + d_n$. The input signal is multiplied with all the binomial coefficients that are stored in the first LUT then the output is added to the data previously stored in the second LUT. The first results of the addition, $(V_{in}w_0 + d_0)$ is not stored and directly taken as an output. The rest

$$ \mathbf{w_0x_0} $$
$$ \mathbf{w_0x_1} + \mathbf{w_1x_0} $$
$$ w_0x_2 + \mathbf{w_1x_1} + \mathbf{w_2x_0} $$
$$ w_0x_3 + w_1x_2 + \mathbf{w_2x_1} + \mathbf{w_3x_0} $$
$$ w_0x_4 + w_1x_3 + w_2x_2 + \mathbf{w_3x_1} + \mathbf{w_4x_0} $$
$$ w_0x_5 + w_1x_4 + w_2x_3 + w_3x_2 + \mathbf{w_4x_1} + \mathbf{w_5x_0} $$
$$ w_0x_6 + w_1x_5 + w_2x_4 + w_3x_3 + w_4x_2 + \mathbf{w_5x_1} + \mathbf{w_6x_0} $$
$$ \vdots $$
$$ w_0x_n + w_1x_{n-1} + w_2x_{n-2} + \quad \cdots\cdots\cdots\cdots \quad + \mathbf{w_{n-1}x_1} + \mathbf{w_nx_0} $$

Fig. 2. Row and column vector dot products output.