Alternative Implementations of a Fractional Order Control Algorithm on FPGAs

Cristina I. Muresan, George Mois, Silviu Folea, Clara Ionescu
Department of Automation
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Cristina.Pop@aut.utcluj.ro

***Abstract*— Traditionally, microprocessor and digital signal processors have been used extensively in controlling simple processes, such as direct current motors. The Field Programmable Gate Arrays (FPGA) are currently emerging as an alternative to the previously used devices in controlling all sorts of processes. The fractional order proportional-integrative control algorithm has the advantage of enhancing the closed loop performance as compared to traditional proportional-integrative controllers, but the implementation requires a higher number of computations. Implementations of control algorithms on FPGAs are nowadays much faster than implementations on microprocessors. This allows for a more accurate digital realization of the fractional order controller. The paper presents nine alternative implementations of such control algorithm on two different FPGA targets. The experimental results, considering DC motor speed control, show that double, fixed-point and integer data representation may be used efficiently for control purposes.**

Keywords—Field Programmable Gate Arrays, DC motor speed control, data representation, fractional order controller

I. INTRODUCTION

Fractional order PID (FO-PI$^{\mu}$D$^{\lambda}$ – fractional order proportional-integral-derivative) control represents a generalization of the traditional PID control, meaning that the integration and differentiation of the error signal is performed to any order, rather than an integer order which is specific to the classical approach. It is generally considered that FO-PID can boost the performance of a closed loop system, as compared to the traditional PID. This is mainly due to the ability to meet two extra performance requirements, by proper tuning of the fractional orders – λ and μ [1-3]. Despite the advantages they ensure as compared to the traditional PIDs, very few research has been conducted towards the actual implementation of FO-PI$^{\mu}$D$^{\lambda}$s on real processes.

In general, the realization of fractional-order differentiators and integrators can be performed in two ways: a continuous time and a discrete time approximation. In the continuous time realization, the fractional order element is approximated as a higher order rational system which maintains a constant phase within a chosen frequency band [4-6]. The continuous time realization is achieved by applying several iterative techniques, such as Carlson's method [7], Oustaloup's method [8] or Charef's method [9]. The discrete time realization, more suitable for a hardware implementation, can be done either directly or indirectly. In an indirect approach, firstly the frequency domain fitting is done in continuous time domain and then the fitted continuous time transfer function is discretized. The direct discretization methods are based on power series expansion, continued fraction expansion and MacLaurin series expansion, combined with a suitable generating function that maps the continuous time operator into its discrete equivalent. Various types of generating functions, such as recursive Tustin, Simpson, Al-Alaoui, mixed Tustin-Simpson, mixed Euler-Tustin-Simpson, impulse response based and other higher order generating functions, have been proposed [10-15]. The direct discretization methods are generally preferred instead of the indirect discretization methods. In the indirect approach, the approximated transfer function with a constant phase is discretized with a suitable generating function, which may result in a deviation from the original desired transfer function depending on the generating function. Direct discretization rules are, on the other hand, based upon maintaining a constant phase of the fractional order element directly in the frequency domain. Hence, direct discretization is generally preferred for digital realization over its indirect counterpart [16].

The present paper tackles the problem of reaching the best implementation of a fractional order PI control algorithm on a Field Programmable Gate Array (FPGA) target, for controlling the speed of a DC motor, in terms of computation speed, area overhead and overshoot. Fractional order control algorithms for DC motors have been previously implemented on programmable logic controllers [17]. However, only the digital implementation of the fractional order controller is given, without its testing on an actual process. Traditionally, microprocessors and digital signal processors are the most widely used in low-rate applications. However, recent advances in technology, as well as the simplification in programming FPGAs have lead to an increasing trend in using such devices in control applications [18]. The resulted controllers can take advantage of their native characteristics, such as increased parallelism and fast time-to-market. In [18], the digital realization of the fractional order operator is presented without considering experimental closed loop results using a fractional order controller.

In [19], the authors proposed a first implementation of the fractional order controller on an FPGA, but several problems regarding the implementation were left unsolved. Such implementation problems that can arise are related to the data representation. Floating-point computations are highly

978-1-4799-2079-2/13/$31.00 ©2013 IEEE

accurate, but difficult to be implemented in hardware applications. On the other hand, fixed-point operations have the advantage of increasing the computation speed and are fairly easy to be implemented. The main drawback of fixed-point representation consists in the loss of precision. The paper presents the comparative results obtained when implementing the fractional order control algorithm on two different FPGA targets using various data representations, including integer order, fixed-point and double format.

The paper is organized as follows. In section II, the implementation of the fractional order, as well as the tuning of the FO-PI controller for a DC motor are discussed. Section III presents the implementation of the designed FO-PI control algorithm on two different FPGA devices with different data representations, together with the comparative experimental closed loop. Finally, Section IV presents the concluding remarks.

II. From a Fractional Order PI Transfer Function to an FPGA Ready Algorithm

A. Design of a fractional order PI controller for DC motor

The transfer function of the fractional order PID controller is given by:

$$H_{FO-PID}(s) = k_p\left(1+\frac{k_i}{s^{\mu}}+k_d s^{\lambda}\right) \tag{1}$$

with λ and μ being any arbitrary real numbers, k_p, k_i and k_d being the proportional, integrative and derivative gains. If $\lambda=\mu=1$, the traditional PID controller is obtained. If $k_d=0$, the transfer function in (1) becomes:

$$H_{FO-PI}(s) = k_p\left(1+\frac{k_i}{s^{\mu}}\right) \tag{2}$$

which is the general transfer function of an FO-PI controller. Tuning of the FO-PI controller in (2) implies the unique determination of all three parameters: k_p, k_i and the integration order μ based on three independent equations. These equations are derived based on a set of three performance specifications that refer to a gain crossover frequency ω_{gc} – to ensure a certain settling time -, a phase margin φ_m – to ensure a certain overshoot – and a robustness condition to gain variations. Using these three performance specifications applied to the open loop transfer function $H_d(s)$, a system of three equations with three unknown variables may be constructed:

$$\begin{cases} \left|H_d(j\omega_{gc})\right| = 1 \\ \angle H_d(j\omega_{gc}) = -\pi+\varphi_m \\ \dfrac{d\left(\angle H_d(j\omega_{gc})\right)}{d\omega_{gc}} = 0 \end{cases} \tag{3}$$

The first equation in (3) implies that the modulus of the open loop phase margin be zero dB at the gain crossover frequency; the second equation specifies that the phase margin of the open loop system equals φ_m at the gain crossover frequency; the third and final equation in (3) specifies the robustness condition, meaning that the phase of the open loop system should not vary significantly at the gain crossover frequency. This last specification ensures that the overshoot of the closed loop system is close to the one obtained in nominal conditions, even when the open loop gain is modified. Since system (3) may have one unique solution, an infinity number of solutions or no exact solution at all, an optimization routine to yield the final results has to be implemented. Such a routine may be based on graphical methods, predefined Matlab functions, genetic algorithms, etc [1, 19].

The process to be controlled is the speed of a DC motor. The performance specifications, apart from ensuring robustness to gain variations, are: ω_{cg} =15 rad/s and φ_m =70°. The process transfer function is identified using experimental data [19]:

$$H(s) = \frac{27.5}{0.26s+1} \tag{4}$$

Applying the system of equations in (3), with the values imposed for the performance specifications, leads to:

$$\begin{cases} k_p\left|1+15^{-\mu}\cdot k_i\left(\cos\dfrac{\pi\mu}{2}-j\sin\dfrac{\pi\mu}{2}\right)\right| = 0.144 \\ \dfrac{k_i\sin\left(\dfrac{\pi\mu}{2}\right)}{15^{\mu}+k_i\cos\left(\dfrac{\pi\mu}{2}\right)} = 0.0984 \\ \dfrac{\mu k_i 15^{-\mu-1}\sin\dfrac{\pi\mu}{2}}{1+2k_i 15^{-\mu}\cos\dfrac{\pi\mu}{2}+k_i^2 15^{-2\mu}} = 0.016 \end{cases} \tag{5}$$

where the relation:

$$H_{FO-PI}(j\omega_{gc}) = k_p\left[1+k_i\omega_{gc}^{-\mu}\left(\cos\frac{\pi\mu}{2}-j\sin\frac{\pi\mu}{2}\right)\right] \tag{6}$$

has been used for the FO-PI controller.

System (5) has three unknown variables, the three parameters of the FO-PI controller. Solving (6), leads to the following values: $\mu = 0.7371$, k_i = 7.85 and k_p = 0.09. The resulting FO-PI controller has the transfer function:

$$H_{FO-PI}(s) = 0.09\left(1+\frac{7.85}{s^{0.7371}}\right) \tag{7}$$

B. *Implementation of the fractional order*

To implement the fractional order controller on a FPGA, the discrete form of (7) is required. The direct discretization method adopted in this paper is based on the 9th order recursive Tustin method [13]:

$$s^{\mu} = \left(\frac{2}{T}\right)^{\mu} \frac{A(z^{-1},\mu)}{A(z^{-1},-\mu)} \tag{8}$$

with T- the sampling time, equal to 0.015 seconds, and the polynomial A is computed as in [13]. Then, the discrete form of (7), based on (8) and (9), is given by:

$$H_{FO-PI}(z^{-1}) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + ... + a_{10} z^{-10}}{1 + b_1 z^{-1} + b_2 z^{-2} + ... + b_{10} z^{-10}} \tag{10}$$

where the polynomial coefficients are given in Table 1.

TABLE 1.COEFFICIENTS OF THE FO-PI CONTROL ALGORITHM

	a_0=0.10111236572265625
b_1=0.72219846328125	a_1=-0.0458221435546875
b_2=0.243499755859375	a_2=-0.0246200561523437
b_3=-0.0606842041015625	a_3=0.00385284423828125
b_4=0.074066162109375	a_4=-0.0074920654296875
b_5=-0.0364608764648437	a_5=0.0023117065429687 5
b_6=0.04343414306640625	a_6=-0.004394531250000
b_7=-0.026763916015625	a_7=0.00170135498046875
b_8= 0.032135009765625	a_8=-0.0032501220703125
b_9=-0.0222930908203125	a_9=0.00141143798828125
b_{10}=0.0308685302734375	a_{10}=-0.0031204223632812

To implement the FO-PI controller in (10) on FPGA targets, a recursive relation that computes the command signal based on the error signal is derived. Relation (10) may be written as:

$$H_{FO-PI}(z^{-1}) = \frac{c(z^{-1})}{\varepsilon(z^{-1})}$$

(11) where $c(z^{-1})$ is the command signal and $\varepsilon(z^{-1})$ is the error signal. Thus, using the coefficients in Table 1 and equations (10) and (11), the recurrent relation for the command signal is obtained as:

$$c(k) = a_0\varepsilon(k) + a_1\varepsilon(k-1) + ... + a_{10}\varepsilon(k-10) - b_1 c(k-1) - b_2 c(k-2) - ... - b_{10} c(k-10) \tag{12}$$

The schematic diagram of the DC motor controlled using the previously designed fractional order controller is given in Figure 1.

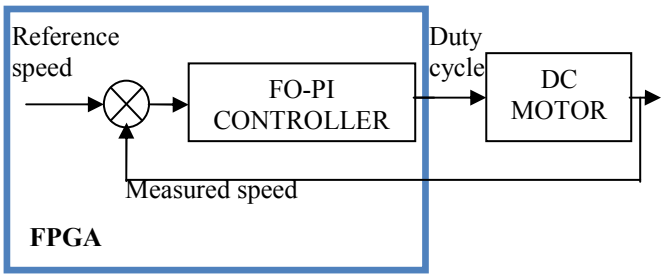


Fig. 1. Block diagram of the closed loop implementation using FPGA

III. COMPARATIVE FPGA IMPLEMENTATIONS AND EXPERIMENTAL RESULTS

The FO-PI control algorithm in (12) was further implemented on two different FPGAs, a high performance chip, on a CompactRIO™ platform and a second one, a cheaper device, a Spartan-3E, on a Digital Electronics FPGA board.

The difficulty of the implementation of (12) on the FPGA resides in the increased number of multiplications and summations. Additionally, the data representation in fixed-point and integer format may lead to a deterioration of the closed loop results. The integer data representation was implemented using a scaling procedure that involved multiplying the coefficients in Table 1 with a scalar, performing the computations in integer approximated values and finally dividing the result to the scalar prior to sending the command signal to the DC motor. Figure 2 shows an example of such a scaling procedure and the implementation using LabVIEW™, while Figure 3 shows the implementation in fixed-point, using the same programming environment.

LabVIEW was chosen because it provides a user-friendly configuration environment and a short project development time, advantages which allow the controller designer to implement the algorithm in hardware without needing a thorough understanding of digital design principles. The resources occupied in the FPGA, considering the fractional order control implementation, are presented in Tables 2 and 3. The results show that a maximum performance is achieved when using the CompactRIO™ device, which includes an FPGA Virtex II (2v3000fg676-4) board.

However, in terms of resources used, the implementation of the FO-PI control algorithm on the Spartan-3E device requires less resources as compared to the CompactRIO™ implementation, but the drawback resides in an increase in the computation time. The CompactRIO™ clock is at 40 MHz, while Spartan-3E works on a 50 MHz clock. With integer order implementation, the occupied resources increase. This is due to the supplementary computations required for the scaling procedure. Nevertheless, the number of multipliers used is at its minimum value when considering integer order data representation. In the particular example given in Tables 2 and 3, the coefficients in Table 1 are firstly multiplied by 10000 and then converted to integer.

The computations in (12) are then performed using integer data representation and the final value for the comand signal is obtained by dividing the result with 10000. Figure 2 shows the implementation of the FO-PI control algorithm considering integers scaled with 10000, as explained.

The implementation of the FO-PI control algorithm with non-reentrant (NR) subVIs in LabVIEW™ ensures a decreased number of necessary multipliers, without significantly increasing the used resources. This is achieved using data pipelining. This type of implementation was only possible on the CompactRIO™ platform, since the implementation on Spartan-3E required a number of

multipliers that exceed the 4 times the resources available. Nine different implementations have been considered: Int16 and Int32, with the values scaled with 100, Int32 scaled with 1000, Int32 scaled with 10^4, Int32 scaled with 10^5, Int32 scaled with 10^6, Int32 scaled with 10^7, double 64 bit and fixed-point 15.32.

The experimental results obtained in all 9 cases are presented in Figure 4. The DC motor speed is given in Figure 4a), while the duty ratio is given in Figure 4b). Table 4 presents the closed loop performance obtained in all implementations, when the reference changes from 500 rot/min to 1400 rot/min. It can be seen that a steady state position error exists when considering integer data representation, ranging from 2.5% up to 8.5% for the Int16 and Int32 representations, scaled with 100. The data representation has little effect upon the overshoot, while the settling time is increased more than twice in the case of Int16 and Int32, scaled with 100, compared to the other cases.

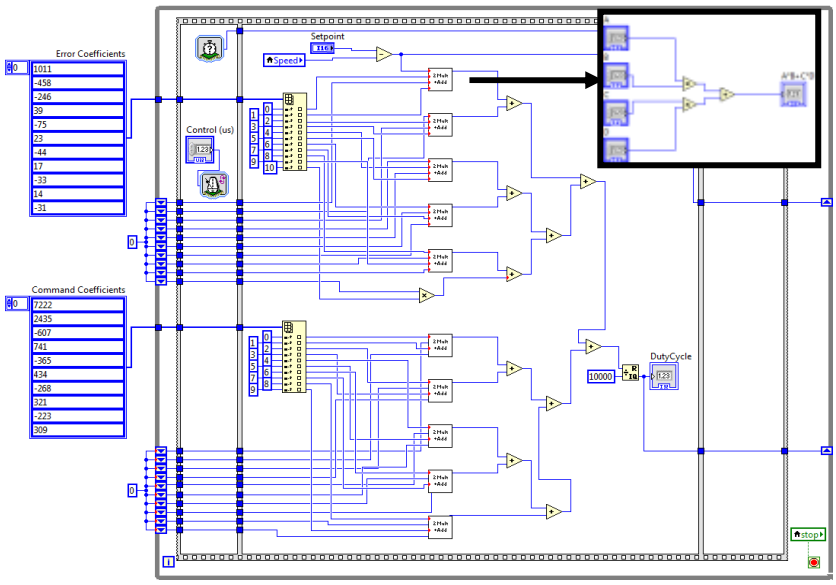


Figure 2. Implementation of the FO-PI control algorithm using integer data representation

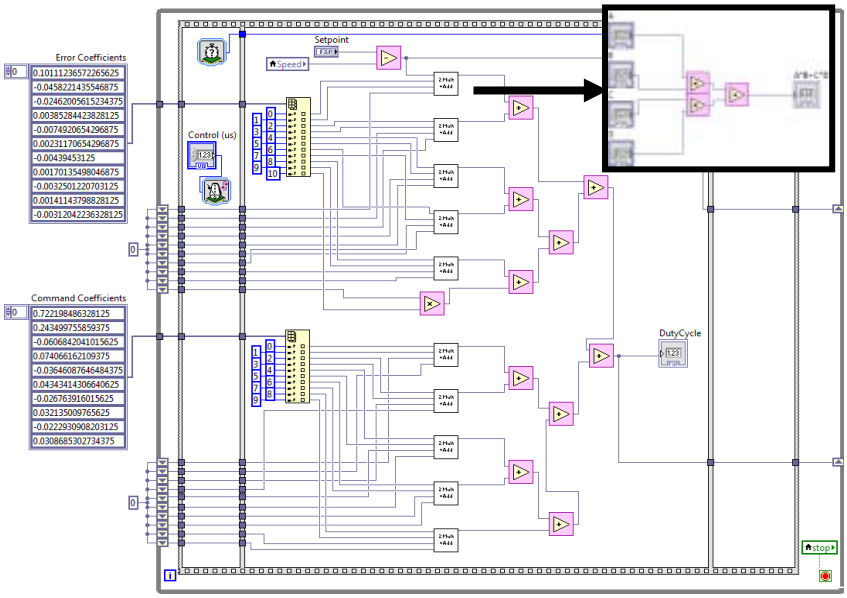


Figure 3. Implementation of the FO-PI control algorithm using fixed-point data representation

TABLE 2. FPGA - DEVICE UTILIZATION ON COMPACTRIO™

2v3000fg676-4 / Virtex II	Total Slices (14336)	Slice Registers (28672)	Slice LUTs (28672)	No. of MULT18X18s (96)	Exec. time (ns)	Clock max. (MHz)
VI (Timed Loop)	25%	10%	19%	90%	50	20.0

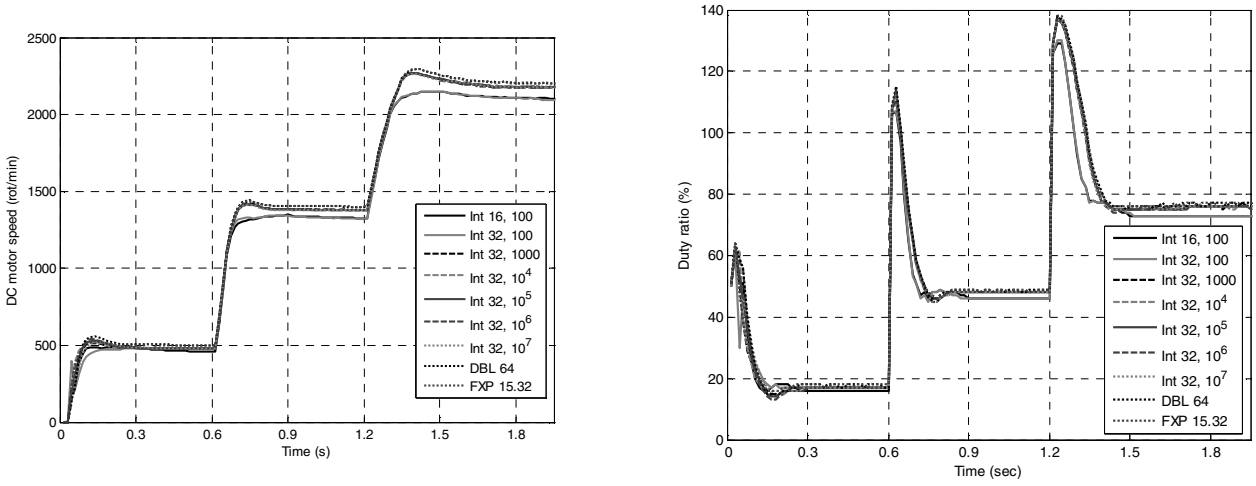
FXP 16.32	(3,679)	(3,120)	(5,723)	(87)		
VIs (SubVI 1) FXP 15.32	24% (3,443)	14.0% (4,236)	19% (5,453)	90% (87)	275	45.55
VIs (SubVI 1 NR) FXP 15.32	21.3% (3,047)	13.4% (3,854)	15.2% (4,351)	15% (15)	1675	47.84
VIs (SubVI 2 NR) Int 32	21.4% (3,070)	12.9% (3,709)	15.2% (4,363)	10% (10)	2575	54.25

TABLE 3. FPGA – DEVICE UTILIZATION ON NI ELVIS DIGITAL ELECTRONICS BOARD

xc3s500e-ft256-4 Spartan-3E	Total Slices (4,656)	Slice Registers (9,312)	Slice LUTs (9,312)	No. of MULT18X18s (20)	Exec. time (ns)	Clock max. (MHz)
VIs (SubVI 1 NR) FXP 15.32	63.2% (2,944)	40.5% (3,774)	50.6% (4,714)	75% (15)	1,340	62.50
VIs (SubVI 2 NR) Int 32	66.1% (3,076)	41.7% (3,881)	53% (4,931)	55% (11)	3,000	66.67

TABLE 4. CLOSED LOOP PERFORMANCE

	Int16(10^2)	Int32(10^2)	Int32(10^3)	Int32(10^4)	Int32(10^5)	Int32(10^6)	Int32(10^7)	DBL	FXP15.32
Overshoot	3%	3%	5%	5%	5%	5%	5%	4%	4%
Settling time	0.36s	0.36s	0.165s	0.165s	0.165s	0.165s	0.165s	0.165s	0.165s
Steady state error	8.5%	8.5%	2.5%	2.5%	2.5%	2.5%	2.5%	0%	0%



a) b)

Figure 4. Closed loop comparative experimental results using different data representations a) DC motor speed b) duty ratio

IV. CONCLUSION

The purpose of the present paper was to compare the implementation of a fractional order PI control algorithm on two different FPGA targets, for controlling the speed of a DC motor. A total of nine data representations were considered throughout the paper, each one of them having a different impact on computational speed, area overhead and overshoot.

The results obtained show that the implementation of the FO-PI control algorithm on Spartan-3E requires less resources compared to the CompactRIO™. In terms of data representation, the integer order representation requires more resources, due to the supplementary computations required for the scaling procedure. In this case, however, the number of multipliers used has the minimum value.

In terms of closed loop performance, the experimental results show that the overshoot, steady state error and settling time are quite similar when considering Int32 scaled

with 10^3, 10^4, 10^5, 10^6, 10^7 and double or fixed point representation. This justifies the possibility of using these integer order data representations as an alternative to double or fixed point. Nevertheless, poor results are obtained in the case of Int16 and Int32, scaled with 100, for the steady state error, overshoot and settling time. Consequently, such data representation is not efficient for the FO-PI control algorithm implementation.

ACKNOWLEDGMENT

This work was supported by a grant of the Romanian National Authority for Scientific Research, CNDI–UEFISCDI, project number 155/2012 PN-II-PT-PCCA-2011-3.2-0591.

REFERENCES

[1] C.A. Monje, Y. Chen, B.M. Vinagre, D. Xue, V. Feliu, *Fractional-order systems and controls: Fundamentals and applications*, Springer, London, 2010

[2] C.I. Pop (Muresan), C.M. Ionescu, R. De Keyser, E.H. Dulf, E.-H, Robustness evaluation of Fractional Order Control for Varying Time Delay Processes, *Signal, Image and Video Processing*, vol. 6 pp. 453-461, 2012

[3] I. Podlubny, Fractional-order systems and PIλDμ- controllers, *IEEE Trans. Automat. Control,* vol. 44, pp.208–214, 1999

[4] B.T. Krishna, Studies on fractional order differentiators and integrators: a survey, *Signal Process*, vol. 91, no. 3, pp. 386–426, 2011

[5] A.S. Elwakil, Fractional-order circuits and systems: an emerging interdisciplinary research area, *IEEE Circuits Syst. Mag.*, vol. 10, no. 4, pp. 40–50, 2010

[6] B. Maundy, A.S. Elwakil, T.J. Freeborn, On the practical realization of higher-order filters with fractional stepping. *Signal Process,* vol. 91, no. 3, pp. 484–491, 2011

[7] G. Carlson, C. Halijak, Approximation of fractional capacitors (1/s)ˆ(1/n) by a regular newton process, *IEEE Trans. Circuit Theory*, vol. 11, no. 2, pp. 210–213, 1964

[8] A. Oustaloup, F. Levron, B. Mathieu, F.M. Nanot, Frequency-band complex noninteger differentiator: characterization and synthesis, *IEEE Trans. Circuits Syst. I Fundam. Theory Appl.*, vol. 47, no. 1, pp. 25–39, 2000

[9] A. Charef, H.H. Sun, Y.Y. Tsao, B. Onaral, Fractal system as represented by singularity function, *IEEE Trans. Autom. Control*, vol. 37, no. 9, pp. 1465–1470, 1992

[10] M. Tenreiro, A.M. Galhano, A.M. Oliveira, J.K. Tar, Approximating fractional derivatives through the generalized mean, *Communications in Nonlinear Science and Numerical Simulation*, vol. 14, no. 11, pp. 3723-3730, 2009

[11] B.M. Vinagre, I. Podlubny, A. Hernandez, V. Feliu, Some approximations of fractional order operators used in control theory and applications, *Fractional Calculus and Applied Analysis*, vol. , 3, no. 3, pp. 231-248, 2000

[12] B.M. Vinagre, Y.Q. Chen, I. Petras, Two direct Tustin discretization methods for fractional – order differentiator / integrator, *Journal of the Franklin Institute: Engineering and applied mathematics*, vol. 340, 349-362, 2003

[13] Y.Q. Chen, K.L. Moore, Discretization schemes for fractional-order differentiators and integrators, *IEEE T. Circuits.-I.*, vol. 49, pp.363-367, 2002

[14] Y. Chen, B.M. Vinagre, I. Podlubny, Continued fraction expansion approaches to discretizing fractional order derivatives—an expository review, *Nonlinear Dyn*, vol. 38, no. 1, pp. 155–170, 2004

[15] M. Gupta, P.Varshney, G.S. Visweswaran, Digital fractional-order differentiator and integrator models based on first-order and higher order operators, *Int. J.Circuit Theory Appl*, vol. 39, no. 5, pp.461–474, 2011

[16] S. Das, I. Pan, Fractional order signal processing: Introductory concepts and applications, Springer Briefs in Applied Sciences and Technology, 2012

[17] I. Petras, Fractional-order feedback control of a DC motor, *Journal of Electrical Engineering*, vol. 60, no. 3, pp.117-128, 2009

[18] R. Caponetto, G. Dongola, A. Gallo, Fractional integrative operator and its FPGA implementation, *Proceedings of the ASME 2009 International Design Engineering Technical Conferences*, San Diego, USA, 2009

[19] C.I. Muresan, S. Folea, G. Mois, E.H. Dulf, Development and Implementation of an FPGA Based Fractional Order Controller for a DC Motor, *Mechatronics*, article accepted for publication, December 2013