EasyChair Preprint

№ 10894

FPGA Implementation of Adaptive PID Control for Quadcopter Position Tracking

Harish Bhat N, Shreesha Chokkadi and Satish Shenoy B.

September 13, 2023

AOSEC 2023
Asia Oceania Systems Engineering Conference
Theme: Digitalization for engineering Complex Systems
11th - 14th October, 2023 | Bangalore, Karnataka India
Hosted by INCOSE India Chapter

FPGA Implementation of Adaptive PID Control for Quadcopter Position Tracking

Harish Bhat N.
Center of Excellence in Avionics and Navigation,
Dept. of Aeronautical/Automobile Engg.,
Manipal Institute of Technology,
Manipal Academy of Higher Education,
Manipal, India
+919916022964
Harish.bhat@learner.manipal.edu

Shreesha Chokkadi
Dept. of Instrumentation and Control Engg.,
Manipal Institute of Technology,
Manipal Academy of Higher Education,
Manipal, India
+919448722353
Shreesha.c@manipal.edu
https://manipal.edu/mit/department-faculty/faculty-list/shreesha-c.html

Satish Shenoy B.
Dept. of Aeronautical/Automobile Engg.,
Manipal Institute of Technology,
Manipal Academy of Higher Education,
Manipal, India
+919844232761
Satish.shenoy@manipal.edu
https://manipal.edu/mit/department-faculty/faculty-list/satish-shenoy.html

Corresponding Authors: Shreesha Chokkadi/Satish Shenoy B.

Abstract. PID control is widely used for designing controllers in production plants and automotive and robotic applications. Although it suits linear systems, with minor modifications, it can also be applied to nonlinear plants. As a simple control approach, it can be easily realized conceptually, but there is a large cost involved in the production of the working controller. A digital PID controller based on a fixed point representation of the operands can be considered for approximate realization. The field programmable gate array (FPGA), with its concurrent architecture, is an ideal way of producing programmable, cost-, power-, and speed-optimized controllers with a compromised controller area compared to application-specific approaches. In this research, the PID controller for quadcopters is initially tested for performance using MATLAB/Simulink based on continuous/analog domain operations. Then, FPGA-based architecture mapping is performed using HDL Coder, with fixed-point-based operands produced using the fixed-point conversion toolbox. With satisfactory performance obtained with system generator-based cosimulation of the controller, Vivado IDE is used to implement the controller on a Zynq Ultrascale+ based FPGA device. The realized cost-efficient controller is analysed for logical resources, computation power and controller frequency and inferred to be optimal with reference to all these indices. By adapting the thrust, which is the control input for the translational motion of the quadcopter, robustness against coupled motion is ensured, with higher simultaneous tracking speeds.

Keywords: PID Control, Adaptive Control, FPGA for Digital Control, Quadcopter.

Introduction

Quadcopters are widely used multirotor unmanned aerial vehicles (UAVs) with four rotors to control six degrees of freedom motion, making them underactuated systems. With high degrees of coupling between each direction of motion, control realization for quadcopters is highly complex. As they are easy to build, quadcopters are in high demand to test new, complex control approaches. In this research, a PID controller, which is an easy-to-realize control approach, is considered for a complex quadcopter system.

McCormack John analysed the basic quadcopter model, the parameters for this model, PSO-based gain tuning for SMC-based control, and the control of swarm formation with a limited and large number of agents. For real-world situations, there is a need to consider improved models. Luis E. et al. considered an improved quadcopter model, but the controller was designed only for attitude and altitude based on the PID approach (inner control). The designed controller is also implemented using inertial and ultrasonic sensors. Sabir Abdelhay et al. considered an improved quadcopter model and simulink-based PID controller design for both inner and outer control and tested the performance. The speed of tracking is observed to be very low due to the coupled nature of motions in different directions. Hardware realization of the designed controller was not considered. Ankit Goel et al. extended the PID approach to make it digital as well as an adaptive variable gain controller for quadcopter position tracking and compared the performance with a fixed gain PID controller. In the nominal mode of tracking, fixed gain PID controllers have better tracking than adaptive PID controllers where gains are tuned based on a retrospective cost adaptive controller (RCAC), but when the quadcopter is tracked with inertia matrix variation, RCAC maintains its performance, but the fixed gain PID controller is oscillatory. The tracking speed in both controllers is observed to be limited, as coupling of the motions in different directions was not considered.

To improve robustness, digital PID controllers are the best options, and the realization complexity of such digital controllers can be addressed by considering FPGA-based approaches. Joao Lima et al. covered a detailed methodology to design PID controllers from MATLAB using floating point representation of operands/operators, uniform/specialized fixed point representations, generation of VHDL-based FPGA realization and their effect on logic resources and maximum frequencies of computations. There is a large scope for efficient architectures and better frequency/power performance using variable bit width fixed point representations using the fixed point converter toolbox in MATLAB with similar errors of tracking. Abdesselem Trimeche et al. covered the FPGA implementation of a PID controller for various configurations, such as P, PI, and PID, highlighting the percentage utilization of each type of logic device and concluding that FPGA-based controller realization is best with reference to area, speed, power, error and cost. Akanksha Somani et al. compared a conventional PID controller for DC motor speed control using MATLAB, a DC motor without any controller and a multiplier-based PID controller. These were simulated based on the system generator for testing the FPGA-based implementation. The best frequency of the controller with limited resources for realization indicates an easy method of good PID control realization. A limitation is the limited options available to customize the design for complex systems such as quadcopter. Michal Kocur et al. extended this work to VHDL coding-based digital PID control for a DC motor tested for performance using a black box in a system generator, which only FPGA experts can take up for complex quadcopter-based systems.

To realize an adaptive PID controller for the quadcopter, in this work, a MATLAB m-function-based controller is tested for satisfactory tracking, followed by a variable bit width fixed point m-function-based digital controller, which is also tested by interfacing these controllers to a Simulink-based quadcopter model. Once the tracking is observed to be similar, mapping to FPGA is done using the HDL coder toolbox. By enclosing these Verilog-based controller structures in black boxes in the system generator and cosimulation with a Simulink-based quadcopter model, if performance is observed to match earlier results, automated realization on any suitable FPGA device based on

Vivado IDE is considered, and the resources, power, and controller frequency requirements are analysed. In the following sections, each of these stages is explained in detail, leading to the cost-effective realization of a digital adaptive PID controller for quadcopter position tracking based on FPGA that has better tracking speed, draws less computation power, and has good controller frequency at the nominal controller area due to fixed point-based operand/operations with acceptable error tolerance.

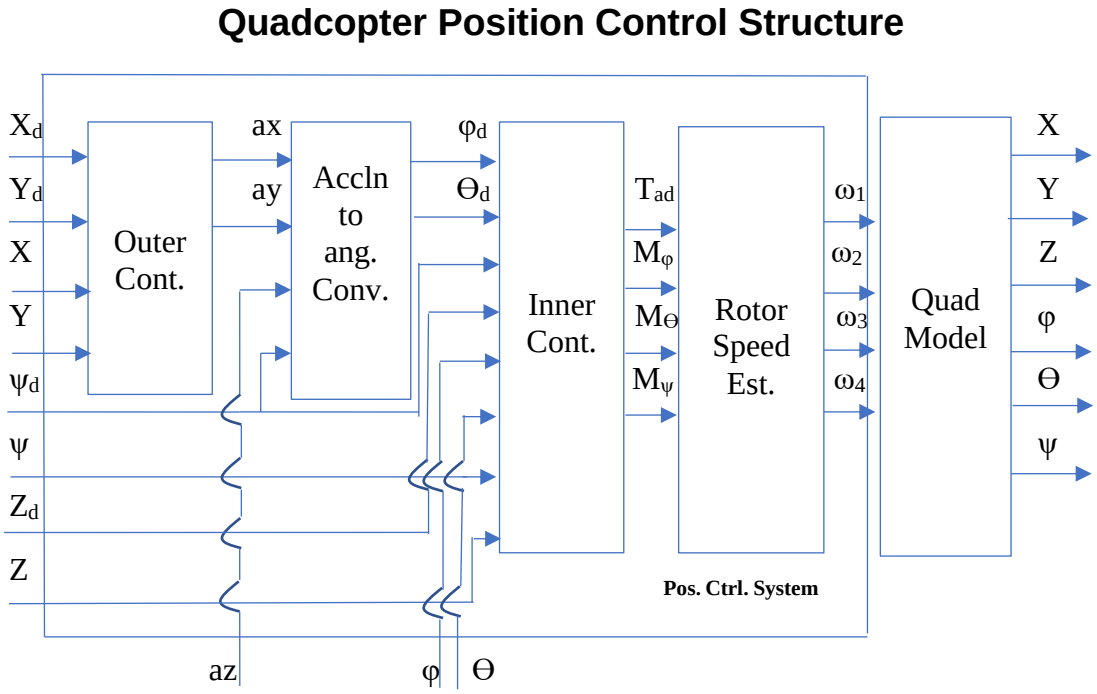

Fig. 1(a) Quadcopter position control system (analog based) (Sabir Abdelhay et al.)

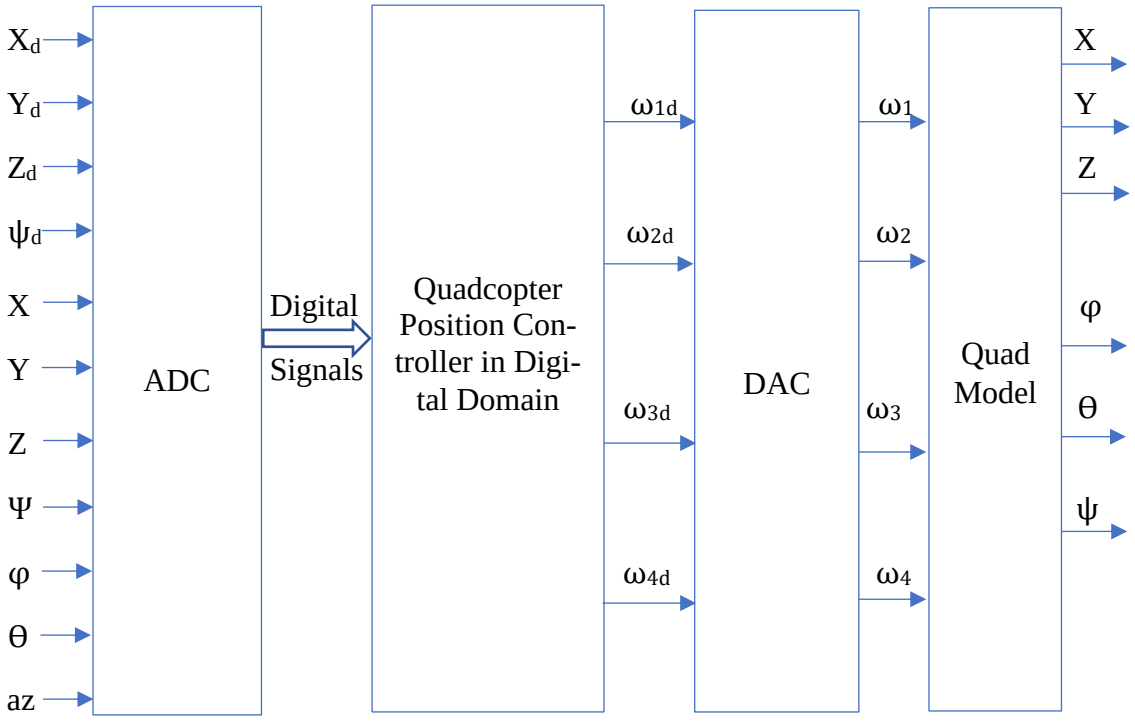

Fig. 1(b) Digital quadcopter position controller

Quadcopter, due to its underactuated structure, has to be tracked with a few of the variables jointly. The X and Y references initially have to be converted to corresponding accelerations, which are converted to pitch/roll angle references by the outer layer controllers. These angles, along with

yaw and altitude references, are applied to inner layer controllers (Sabir Abdelhay, et al.). Thrust and angular moment outputs produced by inner controllers are converted to desired rotor speeds by the rotor speed estimator that can be applied to the quadcopter model to produce desired motions in the X/Y/Z directions. Fig. 1(a) is the conventional system in the analog domain, whereas Fig. 1(b) is the same system in the digital domain. $\omega_{1\text{-}4d}$ are the digital rotor speeds, which, after converting to analog values using DAC, can be applied to the same quadcopter model. X_d, Y_d, and Z_d are the desired input positions, ψ_d is the desired yaw angle, and X, Y, Z, and ψ are the corresponding feedback/actual positions. Φ_d and Θ_d are the desired roll and pitch angles, φ and Θ are the feedback roll and pitch angles, ax, ay, and az are the accelerations in the x, y, and z directions, and T_{ad}, $M_{\varphi,}$ $M_{\Theta, \text{ and}}$ M_{ψ} are the thrust and roll, pitch, and yaw moments, respectively.

Quadcopter Model

Quadcopter positions can be specified in two reference frames. The translational positions X, Y and Z can be measured with reference to inertial reference frame O_E, whereas all the velocities, accelerations and angles can be measured with reference to body frame O_B. The translational accelerations $\dot{U}, \dot{V}$ and $\dot{W}$ are given by (Sabir Abdelhay et al.)

$$\dot{U} = (\sin(\varphi)\sin(\psi)+\cos(\varphi)\sin(\Theta)\cos(\psi))\frac{T}{m} - \frac{A_x U}{m}$$

$$\dot{V} = (-\sin(\varphi)\cos(\psi) + \cos(\varphi)\sin(\Theta)\sin(\psi))\frac{T}{m} - \frac{A_y V}{m} \quad (1)$$

$$\dot{W} = -g + \cos(\varphi)\cos(\Theta)\frac{T}{m} - \frac{A_z W}{m}$$

$A_{x,}$ A_y and A_z are drag coefficients in the X, Y and Z directions, respectively, and U, V and W are the corresponding velocities. m is the mass in kg of the quadcopter, and g is the gravitational acceleration in m/s^2.

T is the thrust and is given by

$$T = K\sum_{i=1}^{4}\omega_i^2 \quad (2)$$

K is the thrust constant, and ω_i's are rotor speeds.

Rotational accelerations (roll, pitch and yaw) are given by

$$\dot{P} = \left(\frac{I_{xx}-I_{yy}}{I_{zz}}\right)QR - \frac{I_R}{I_{xx}}Q\Omega + \frac{M_\varphi}{I_{xx}} - \frac{A_r P}{I_{xx}}$$

$$\dot{Q} = \left(\frac{I_{zz}-I_{xx}}{I_{yy}}\right)PR - \frac{I_R}{I_{yy}}P\Omega + \frac{M_\Theta}{I_{yy}} - \frac{A_r Q}{I_{yy}} \quad (3)$$

$$\dot{R} = \left(\frac{I_{xx}-I_{yy}}{I_{zz}}\right)QP + \frac{M_\psi}{I_{zz}} - \frac{A_r R}{I_{zz}}$$

P, Q and R are the roll, pitch and yaw velocities in the body frame, I_{xx}, I_{yy} and I_{zz} are the moment of inertia of the rotor, I_R is the rotational inertia of each motor, and A_r is the rotational drag coefficient. Ω is the net rotor speed given by

$$\Omega = -\omega_1 + \omega_2 - \omega_3 + \omega_4 \quad (4)$$

M_φ, M_Θ and M_ψ are roll, pitch and yaw moments, respectively, given by

$$M_\varphi = kl(\omega_4^2 - \omega_2^2)$$

$$M_\Theta = kl(\omega_3^2 - \omega_1^2) \quad (5)$$

$$M_\psi = B(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2)$$

Here, B is the torque constant.

The body frame velocities P, Q and R are converted to inertial frame velocities $\dot{\varphi}, \dot{\theta}$ and $\dot{\psi}$ by the matrix

$$\begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\varphi\tan\theta & \cos\varphi\tan\theta \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \frac{\sin\varphi}{\cos\theta} & \frac{\cos\varphi}{\cos\theta} \end{bmatrix} \begin{bmatrix} P \\ Q \\ R \end{bmatrix} \quad (6)$$

Values of the parameters for the quadcopter are summarized in Table 1.

Table 1 Quadcopter model parameters (McCormack John)

m	0.98 kg	k	$3.13*10^{-5}$ N-s^2
g	9.81 m/s^2	B	$3.13*10^{-5}$ N-m-s^2
l	0.17 m	A_x	0.3 N-s/m
I_{xx}	0.01548 kgm^2	A_y	0.3 N-s/m
I_{yy}	0.01565 kgm^2	A_z	0.25 N-s/m
I_{zz}	0.03024 kgm^2	A_r	0.2 N-s
I_R	$6*10^{-5}$kgm^2		

Rotor Speed Estimator

The control inputs produced by the inner controller (thrust and moments from altitude and attitude controllers, respectively) are mixed and converted to rotor speeds for the 4 rotors of the quadcopter by a rotor speed estimator. The equations of mixing are

$$\omega_1^2 = \frac{T_{ad}}{4k} - \frac{M_\theta}{2kl} - \frac{M_\psi}{4B}$$
$$\omega_2^2 = \frac{T_{ad}}{4k} - \frac{M_\varphi}{2kl} + \frac{M_\psi}{4B} \quad (7)$$
$$\omega_3^2 = \frac{T_{ad}}{4k} + \frac{M_\theta}{2kl} - \frac{M_\psi}{4B}$$
$$\omega_4^2 = \frac{T_{ad}}{4k} + \frac{M_\varphi}{2kl} + \frac{M_\psi}{4B}$$

Inner Controller

The inner controller is a PID controller that aids in tracking roll, pitch, and yaw (attitude) along with the altitude signal. PID controller equations for attitude are given in (8), whereas the same for altitude is given in (9) (Sabir Abdelhay et al.).

$$M_{\varphi,\theta,\psi} = k_p e + k_d \frac{de}{dt} + k_i \int e dt \quad (8)$$

With a sampling period of 0.1 sec and a delay of 2 samples,

$\frac{de}{dt} = 5(e(t) - e(t - 0.2))$

Additionally, $e(t) = \varphi_d - \varphi$ for roll angle,

$e(t) = \theta_d - \theta$ for pitch angle,

$e(t) = \psi_d - \psi$ for yaw angle.

k_p, k_d and k_i are proportional, derivative and integral gains tuned using the simulator and are 0.65, 0.2 and 0.08, respectively, for all 3 angles tracking.

For the altitude PID controller, the equation is

$$T_{ad} = \frac{k_p e + k_d \frac{de}{dt} + k_i \int e dt + 9.61}{\cos\varphi \cos\theta} \quad (9)$$

The numerator term in (9) is a conventional PID controller equation, whereas the denominator term demands additional thrust to meet the thrust demand for X and Y tracking without affecting the altitude. This is the adaptive thrust that can ensure better speeds of simultaneous tracking in all 3 directions. Here also,

$e(t) = Z_d - Z,$

$\frac{de}{dt} = 5(e(t) - e(t - 0.2))$

With a similar process of simulator-based gain tunings, k_p, k_d and k_i are found to be 0.45, 2 and 0.02. The constant term in the numerator of (9) is the force due to gravity.

Acceleration to Angle Converter

Accelerations in the X and Y directions produced by the outer controller have to be converted to pitch and roll reference angles, respectively, that can be tracked by the inner controller. The equations (McCormack John) are

$$\theta_d = \arctan \frac{ax.\cos(\psi_d) + ay.\sin(\psi_d)}{az + 9.81}$$

$$\varphi_d = \arcsin \frac{ax.\sin(\psi_d) - ay.\cos(\psi_d)}{\sqrt{ax^2 + ay^2 + (az + 9.81)^2}} \quad (10)$$

ax and ay are the accelerations produced by the outer PID controllers, ψ_d is the yaw reference angle, and az is the altitude acceleration.

Outer Controller

The outer controller is also a PID controller, with X and Y references and actual positions fed back from the quadcopter, computes, error, its derivative and integral produces ax and ay, which are the X and Y accelerations, respectively, as in (11) (Sabir Abdelhay et al.).

$$ax, ay = k_p e(t) + k_d \frac{de}{dt} + k_i \int e(t) dt \quad (11)$$

Here e(t) = Desired – Actual positions for X, Y,

$\frac{de}{dt} = 5(e(t) - e(t - 0.2))$

With the same procedure of simulator-based tuning, proportional, derivative and integral gains are found to be, $k_p = 0.45$, $k_d = 2$, $k_i = 0.02$

All these systems are algorithmically tested for performance in the analog domain (Fig. 1(a)) based on floating point operand/operations using m-functions in MATLAB/Simulink. Once the results are observed to be acceptable, the system is converted to the digital domain shown in Fig. 1(b) by converting external data to a fixed-point representation using the data type converter block available in the Simulink library. Internal operands/operators are also converted to a fixed-point format, resulting in a fixed-point m-function using the fixed-point conversion (Fixed Point Designer User's Guide for MATLAB 2018a) toolbox, also in the Simulink library. Digital outputs produced by this system, the four rotor speeds after converting to double precision floating point data, can be applied to the same quadcopter model as in Fig. 1(a). The results with limited precision possible with fixed point representation should be within acceptable tolerance as obtained with a floating point-based analog control system.

Fixed Point-based Digital Controller

For FPGA-based realization of the controller, initially, all the m-functions have to be converted to fixed point-based m-functions that are in the digital domain. For efficient resource utilization during FPGA-based realization, the data width should be variable, which can be proposed by the user while using a fixed-point conversion toolbox. The same provision is available for operations as well, where instead of using default fixed precision operations, which, although having better accuracy, can be optimized for tolerance within acceptable margins using a specify precision option for product and sum operations, if user-specified precisions lead to overflow, the tool will indicate error, and a fixed point-based function will not be generated.

Fixed point representation requires specification of the sign bit (1 = signed, 0 = only positive/unsigned), total bits and the location of the decimal point from the LSB. A specified fixed point (0, m, n) represents the maximum decimal number $2^{m-n-1} - 1 + (2^{-1} + 2^{-2} + 2^{-3} + \cdots + 2^{-n})$, whereas (1, m, n) represents the minimum decimal number of -2^{m-n-1}. Precision is given by the number of fractional bits (n in the specification). As n increases, precision improves, but computation as well as its realization complexity increases. Hence, there should be a balance between these two requirements to ensure final implementation with optimal controller area. The flowchart in Fig. 2 is the summary of steps for generating fixed point-based operands and operators that lead to the digital system in Fig. 1(b).

Fixed-point algorithm to FPGA-based Digital architecture mapping using the HDL Coder Toolbox

Once the fixed point (floating point should be used for precise operations such as trigonometric operations)-based model is ready for the entire controller, the HDL coder toolbox can be used for the conversion from algorithm to architecture, which is the first step of FPGA-based realization of the position controller. Verilog-based modules are generated for the fixed-point model of the controller. Asynchronous reset with a proper clock and synchronous enable are the strategies used while generating the Verilog-based modules. The flowchart in Fig. 3 summarizes the steps of Verilog code generation using the HDL coder toolbox.

Cosimulation of the FPGA-based Digital Controller using System Generator Tool

With the Verilog-based controller module generation, before implementation, functionality should be verified by comparing the tracking performance with MATLAB-based analog algorithms

and MATLAB-based fixed-point represented digital controller algorithms. The well-known IDE Vivado or any others can simulate only Verilog-based FPGA controllers, whereas MATLAB/Simulink does not have a provision for FPGA-based simulation, but the quadcopter model can be tested. With the collaboration of these two developers, a new blockset has been added to the Simulink library for Xilinx-based devices, known as the Xilinx blockset. It is a simulink environment where the FPGA-based controller as per Xilinx standards after enclosure in the tool known as the blackbox represents the FPGA module equivalent to m-functions. This blackbox-based Xilinx module can be simulated with any Simulink function, including the MATLAB level 2 S function quad model. The FPGA portion is simulated using Vivado, and the quadcopter model is simulated by Simulink by prior linking of Simulink and Vivado using system generator. This concept aids the FPGA-based quadcopter position control system shown in Fig. 1(b), where the digital position controller system is realized using FPGA-based modules enclosed in blackbox, external data at the input should be passed through ADC widely known in system generator terminology as gateway in, and at the output should be converted back to analog using DAC before applying to quadcopter model known as, gateway out. The flowchart in Fig. 4 is a summary of the steps to be followed in the system generator-based co-simulation.

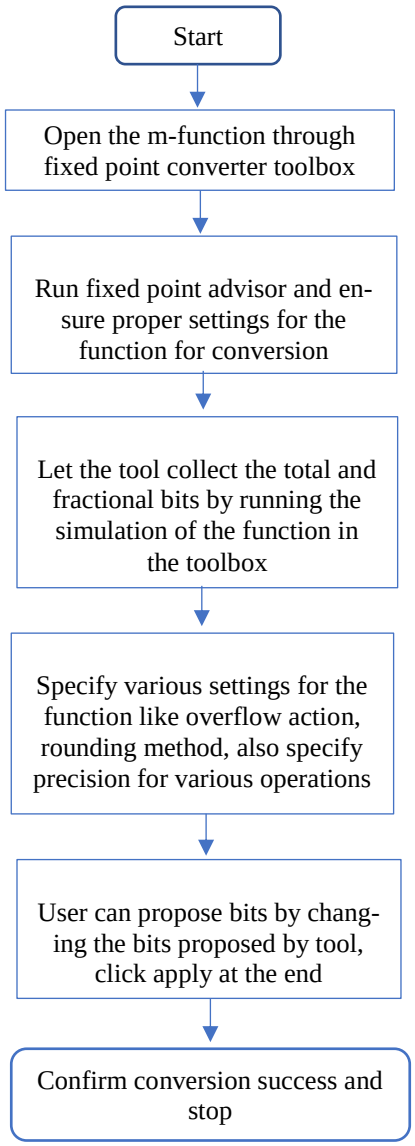

Fig. 2 Flowchart for fixed point conversion of operands/operators

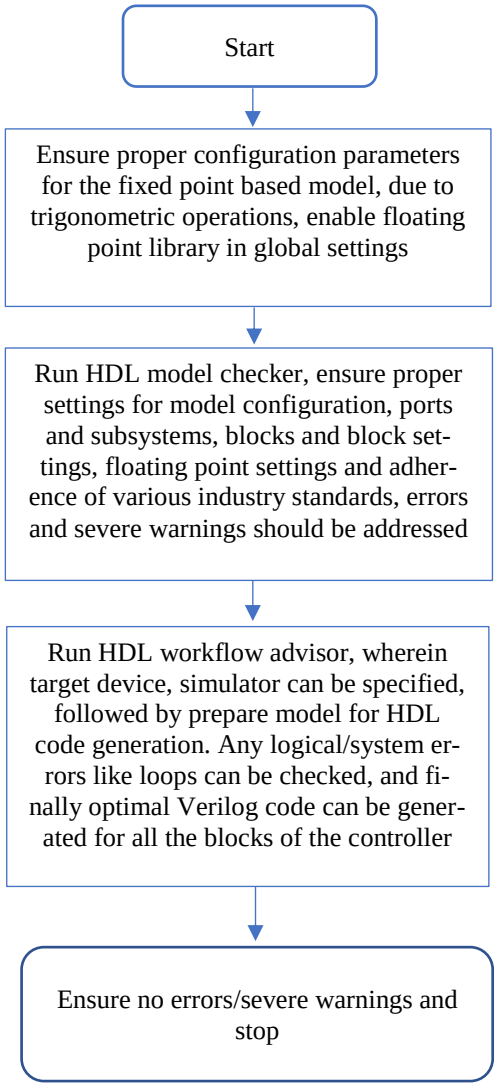

Fig. 3 Flowchart for algorithm to FPGA based architecture mapping using HDL Coder toolbox

FPGA-based Digital Controller Implementation using Vivado v2018.3

With all three-stage simulation results (analog, digital algorithms and FPGA-based cosimulation) within acceptable tolerances, the final step of the position controller implementation using Vivado IDE has to be carried out. As it is completely an FPGA-based implementation without any simulink in the loop, higher clock speeds can be considered (no timing violations below 8 MHz clock speeds have been tested, above which there are violations). The flowchart in Fig. 5 summarizes the steps.

Simulation/Implementation Results

There are 3 stages of simulations, as explained earlier:

- Simulink-based simulation of the floating point (analog) algorithm.
- Simulink-based simulation of the fixed point (digital) algorithm.
- Cosimulation of an FPGA-based controller with a Simulink-based quadcopter model using a system generator.

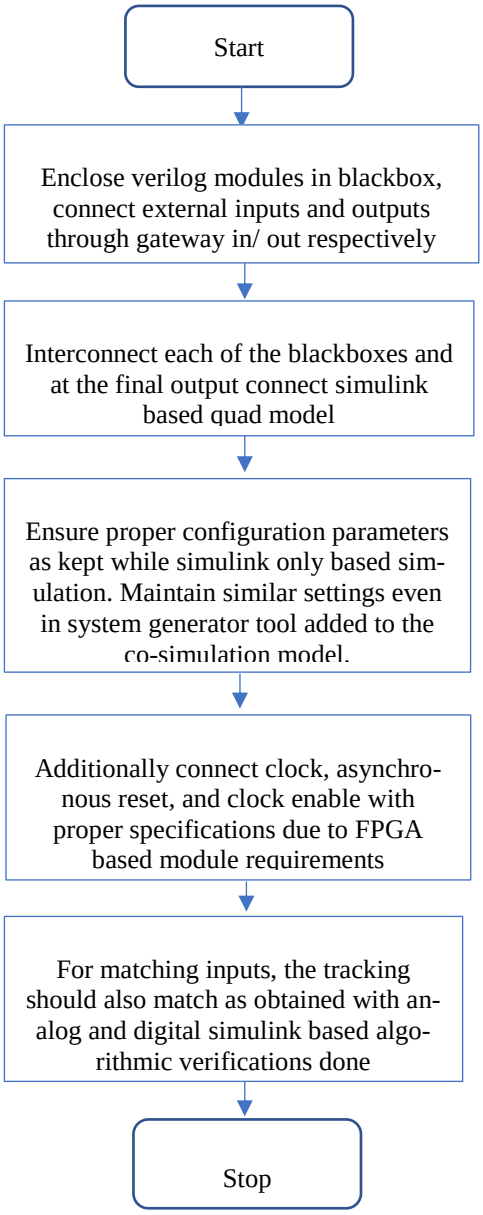

Fig. 4 System generator based co-simulation steps

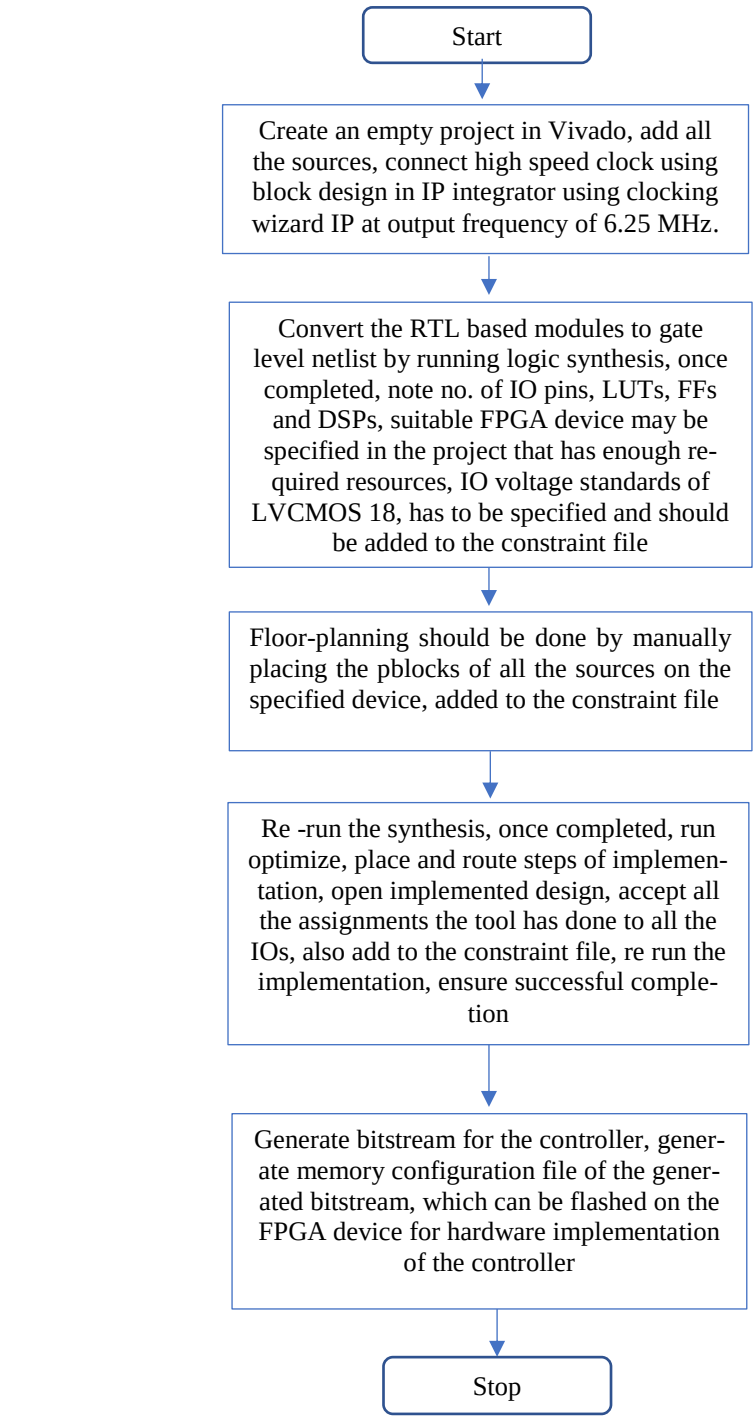

Fig. 5 FPGA based implementation procedural flowchart for the controller

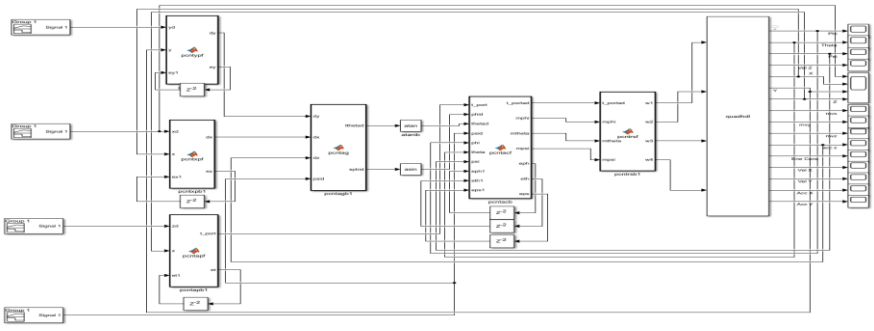

Fig. 6(a) Analog (floating point) based controller model simulation

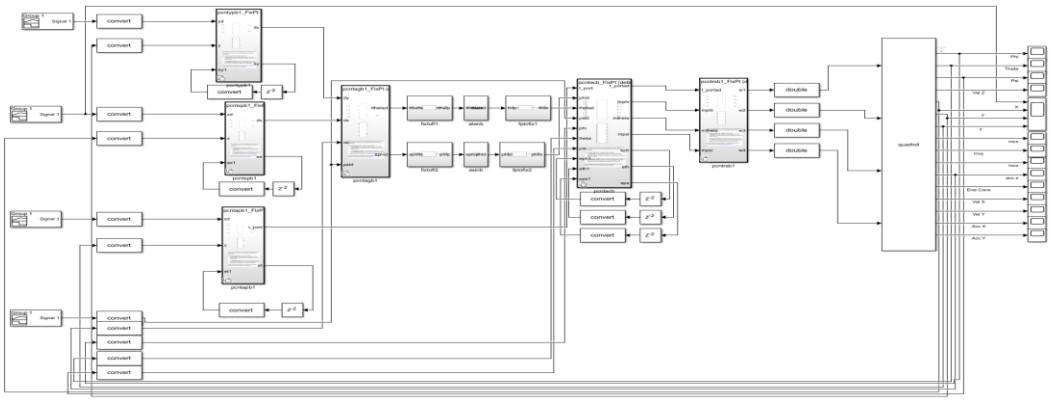

Fig. 6(b) Fixed point digital controller model simulation

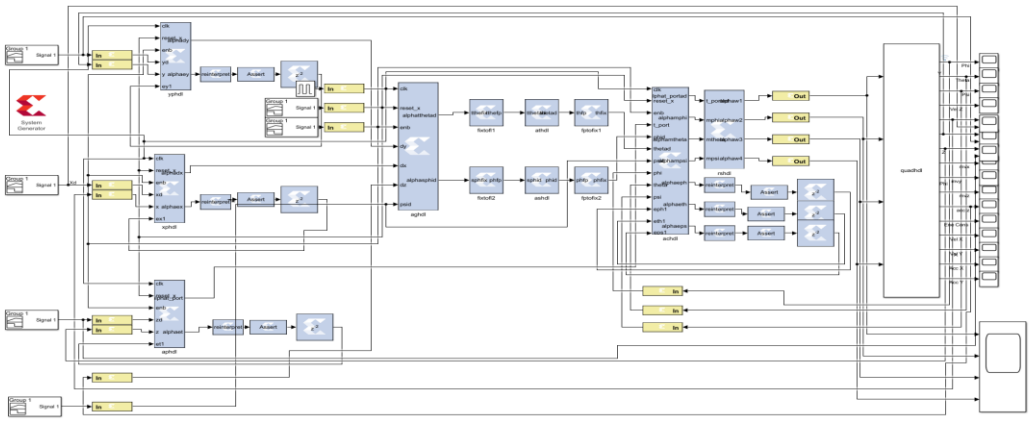

Fig. 6(c) FPGA based digital controller and quad model co-simulation

If all three stages of results are within acceptable tolerance, Vivado-based synthesis/implementation/bitstream generation has to be taken up on a Zynq Ultrascale+ based FPGA device.

Simulation is carried out for a linear (ramp) input at a rate of 1 m/s (3 m in 3 sec) for all three position references. A yaw reference of 0.1 rad (step nature) is also applied externally for all three stages of simulation. Figs. 6(a), 6(b) and 6(c) are for the respective controllers controlling the quadcopter model.

Tracking is evaluated by comparing the settling time, percentage overshoot, steady state error and propulsion energy drawn for all three stages. If these results match for all three stages, then the FPGA-based controller is validated for performance, and the last stage of the controller implementation can be taken up. If the synthesis/implementation resources used (LUTs, FFs and DSPs) are less than the maximum available and power and timing errors are not observed, the controller implementation is successful at the applied frequency.

As covered in the flowchart in Fig. 5, the clock has a maximum frequency chosen by taking care of the data path delay for the critical path ensuring no timing violation. (For this controller, the critical path delay is observed with FPGA-based synthesis to be 123.67 ns, leading to a maximum clock frequency of 8 MHz.) Similarly, with the FPGA device selection that has all the resources slightly more than the utilized resources observed with the synthesis stage, the area is also kept at a minimum. With various Verilog code generation optimizations carried out with reference to operand/operation width, resulting in the simplest Verilog code leading to fewer required resources, the power demand is also kept at a minimum.

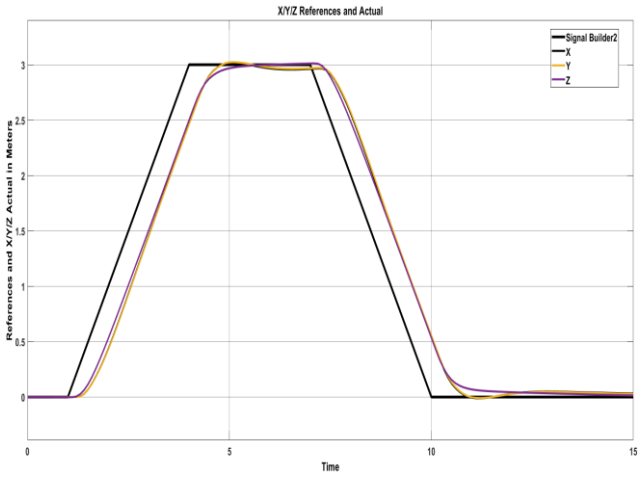

Fig. 7(a) Simulation results for the analog based controller on simulink

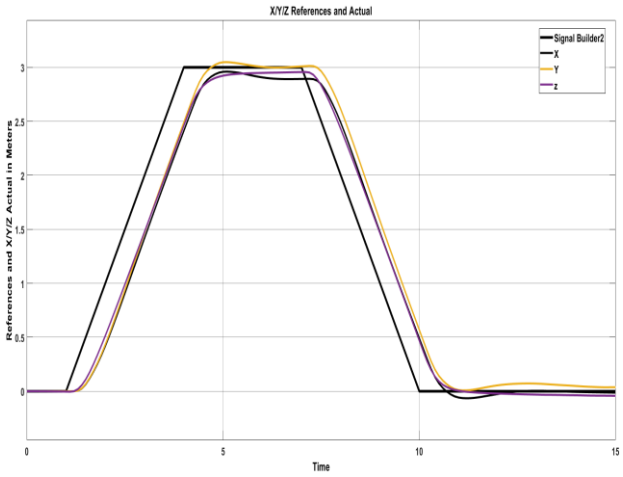

Fig. 7(b) Simulation results for the digital based controller on simulink

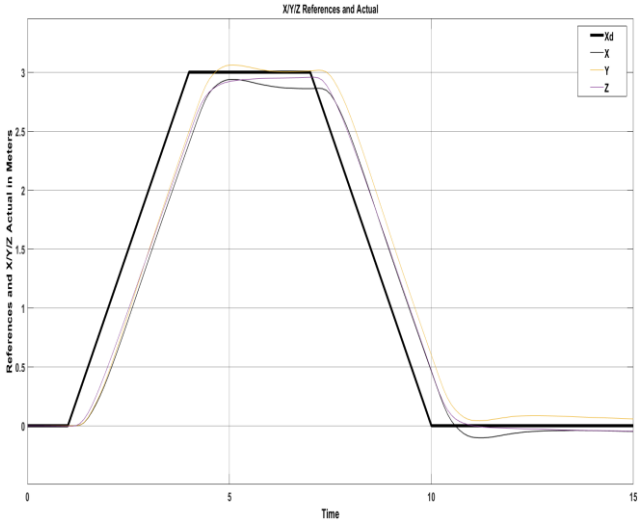

Fig. 7(c) Co-simulation results for the FPGA based controller on system generator

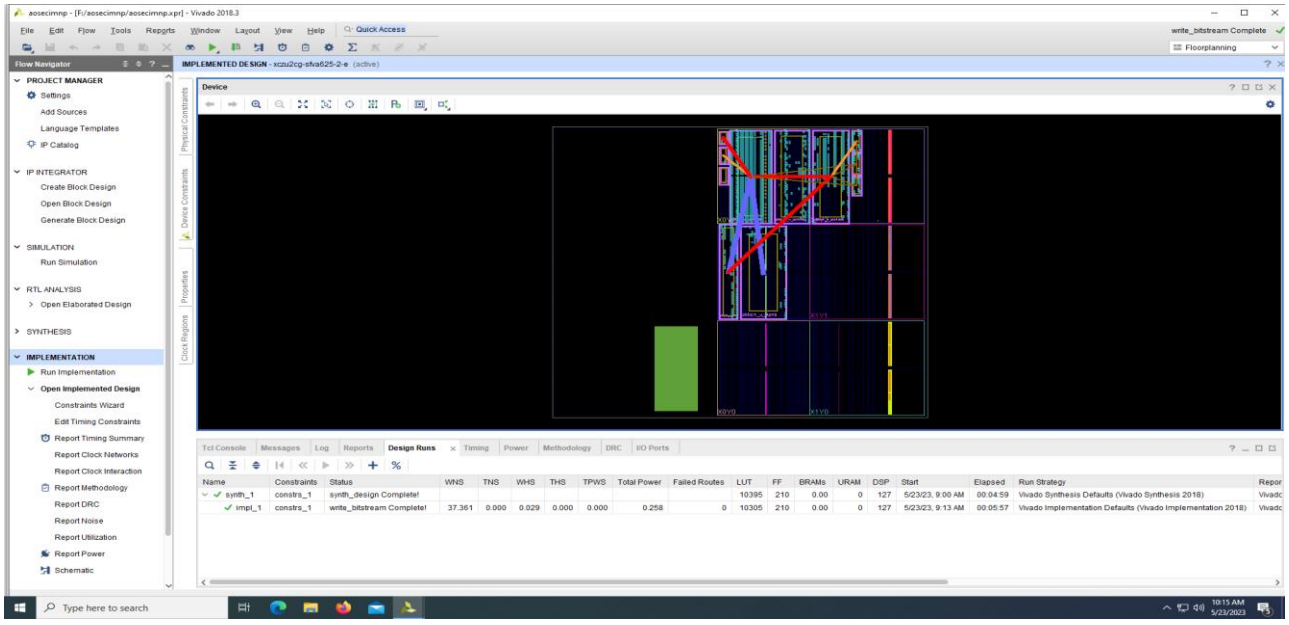

Fig. 8 Vivado v2018.3 based implementation of the controller on Zynq Ultrascale+ device

Waveforms in Figs. 7(a), 7(b) and 7(c) are simulation results for the three stages, and Fig. 8 is the controller implemented on the Zynq Ultrascale + FPGA device (xczu2cg-sfva625-2e)

Tables 2 and 3 summarize the results of the simulation and implementation of the controller, respectively.

Table 2 Summary of results of 3 stages of simulation

Controller	Settling time (Desired 3 sec, with reference to Input rate)			% Overshoot			Steady state error (m)			Energy drawn KJ
	X	Y	Z	X	Y	Z	X	Y	Z	
Analog	3.4 s	3.4 s	3.4 s	0	0.7	0	0.04	0.035	-0.004	15.17
Fixed point	3.5 s	3.4 s	3.6 s	0	1.5	0	0.11	0.005	0.056	15.17
FPGA	3.6 s	3.4 s	3.6 s	0	2	0	0.14	0.004	0.047	15.17

Table 3 Implementation results for the controller using Vivado v2018.3

Resources			Power drawn W	Timing violations at 6.25 MHz Clock	Device dimensions
LUTs	FFs	DSPs			
10305 (22%)	210 (0.22%)	127 (53%)	0.258 (< 1% of max.)	Nil	21 mm*21mm (Data Sheet)

From Table 2, with the observed settling time near the desired time and all the results observed for all three cases within acceptable margins, the analog algorithm to digital and then to FPGA realization is verified based on simulation. Additionally, Table 3 results with limited resources consumed (with no over utilization for the Zynq Ultrascale+ device considered xczu2cg-sfva625-2e, percentage utilization less than 100%), with no crossing of power margin or without any timing violations, with successful bitstream generation as indicated in Fig. 8. The designed PID controller for quadcopter position tracking is realized on the FPGA hardware.

Conclusion/Future Scope

An FPGA-based complete PID controller for quadcopter position tracking has been designed and validated for specified performances. A complete automated process based on various tools leads to a quickest time to market with cost-efficient hardware realization and acceptable speed, area and power performance, as has been observed with performance analysis of the controller.

Complete hardware realization of the controller by flashing the bitstream generated on the target FPGA device with various sensors interfaced to sense the real quadcopter can be considered. Advanced controllers with better propulsion power efficiency for better endurance may also be considered.

References

McCormack, John 2019, "Quadcopter attitude control optimization and multiagent coordination", *Master's Theses and Capstones*. 1277. https://scholars.unh.edu/thesis/1277

Luis E. Romero, David Omar Zurita Pozo and Jorge A. Rosales 2014, “Quadcopter stabilization by using PID controllers”, *In proceedings, Romero 2014 quadcopter SB.*

Sabir Abdelhay, Alia Zakriti, 2019, "Modelling of a quadcopter trajectory tracking system using PID controller", *12th International Conference Inter-Disciplinarity In Engineering, Inter Eng 2018*, 4th-5th October 2018, Tirgu Mores Romania, https://doi.org/10.1016/j.promfg.2019.02.253.

Ankit Goel, Abdulazeez Mohammed Salim, Ahmad Ansari, Sai Ravela and Dennis Bernstein, 2020, "Adaptive digital PID control of a quadcopter with unknown dynamics", *arXiv eprint* 2006.00416.

Lima J., Monotti R., Cardoso J.M.P. and Marques E., 2006, "A methodology to design FPGA based PID controllers", *IEEE international conference on systems, man and cybernatics,* Taipei, Taiwan, pp. 2577-2583, doi: 10.1109/ICSMC.2006.385252

Abdesselem T., Anis S., Abdelatif M. and Mohamed B., 2011, "Advances in PID control: PID controller using FPGA technology", Valery D. Yurkevich (ed.), *InTechOpen*, https://doi.org/10.5772/18295.

Somani, A., Kokate, R., 2018, "Realization of FPGA based PID controller for speed control of DC motor using Xilinx sysgen", Satapathy, S., Joshi, A. (eds) *Information and Communication Technology for Intelligent Systems (ICTIS 2017)* - Volume 2. ICTIS 2017, Smart innovation, Systems and Technologies, vol 84, Springer, Cham, https://doi.org/10.1007/978-3-319-63645-0_9

Kocur M., Kozak S. and Dvorscak B., 2014, "Design and implementation of FPGA-digital based PID controller", *Proceedings of the 2014 15th International Carpathian Control Conference (ICCC),* Velke Karlovice, Czech Republic, pp. 233-236, doi: 10.1109/CarpathianCC.2014.6843603

Fixed point designer user's guide, MATLAB, R2018a, *Mathworks*.

HDL coder user's guide, MATLAB & Simulink, R2018a, *Mathworks*.

UG897, Vivado design suite user guide: Model based DSP design using system generator, v2018.3, *Xilinx*.

UG904, Vivado design suite user guide, implementation, v2018.3, *Xilinx*.

DS891 v1.9, May 26, 2021, *Zynq Ultrascale+ MPSoC Datasheet*.

Biography

HARISH BHAT N. received B.E. in Electronics and Communication Engineering and M.Tech in Digital Electronics and Communication from Visvesvaraya Technological University, Belgaum, India. He is currently pursuing a PhD degree with the Department of Aeronautical and Automobile Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, India. His current research interests include Digital control system design, Autonomous control, Evolvable Hardware, Artificial Intelligence for Evolvable Hardware, Computer vision for state estimation, Fault Tolerant Operation and Power Efficient algorithms for quadcopters. He is an active member of ISTE.

Shreesha Chokkadi is currently working as a Professor and Head, Department of Instrumentation and Control engineering, Manipal Institute of Technology (MIT), Manipal Academy of Higher Education, Manipal. He obtained his PhD from Indian Institute of Technology, Bombay, India in 2002 from Systems and Control IDP. His research interests are Control Engineering. He has authored more than 25 articles in reputed journals and is also a reviewer for International Journal of Control Automation and Systems.

SATISH SHENOY B. is currently working as a Professor with the Department of Aeronautical and Automobile Engineering, Manipal Institute of Technology (MIT), Manipal Academy of Higher Education, Manipal. He is on the Mentor Board of the Center for Excellence in Avionics and Navigation Systems, MIT, Manipal. He is an active researcher in the field of CFD, FEM, and CAD. He has published more than 88 articles in reputed peer-reviewed journals. He is a Life Member of TSI. He is on the board of reviewers of journals such as Tribology International, Proceedings of the Institution of Mechanical Engineers, Part J: Journal of Engineering Tribology, Computer Methods in Biomechanics and Biomedical Engineering, Journal of the Brazilian Society of Mechanical Sciences and Engineering, Simulation Modelling Practice and Theory, Tribology Transactions, Industrial Lubrication and Tribology, Journal of Tribology, Polymer Testing, Acta Radiologica, Journal of Bionic Engineering, Journal of Marine Science and Application, Mechanics and Industry, and Polymer Composites.