Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 56-2 (2023) 3386–3391

Accelerating the Performance of Fuzzy-FPGA Based Control in LabVIEW for Trajectory Tracking Problems

Ayman A. Nada [*] Victor Parque [**] Mona A. Bayoumi [***]

[*] *School of Innovative Design Engineering, Faculty of Engineering, Egypt-Japan University of Science and Technology E-JUST, Alexandria 21934, Egypt. email:ayman.nada@ejust.edu.eg*
[**] *Department of Modern Mechanical Engineering, Waseda University, Tokyo 169-8555, Japan, email:parque@aoni.waseda.jp*
[***] *Benha Faculty of Engineering, Benha University, 13512 Benha, Egypt. email:mona.elawa@bhit.bu.edu.eg*

Abstract: Fuzzy Logic (FL) is well-known as an intuitive framework to tackle imprecision and uncertainty while allowing to model expert's knowledge in rule-based control. The objective of this study is to investigate the design of embedded fuzzy control systems combining fast execution and parallel processing capabilities provided by Field Programmable Gate Arrays (FPGAs) and Reconfigurable Inputs/Outputs (RIO) boards. We implemented a suitable fixed-point FL to realize a computationally efficient control framework on an FPGA target. As such, the paper provides a concise method to deploy Fuzzy Logic Control (FLC) using RIO-FPGA technology. We also provide a technique for implementing the three stages that constitute the FLC structures in LabVIEW environment using fixed math operations. The experimental work involves tracking the trajectory of a mobile robot and shows the feasibility and the efficiency of the proposed FLC-FPGA controller in comparison to the traditional PID-FPGA controller, along with the added benefits of fuzzy control frameworks.

Keywords: Fuzzy Logic Control, Field Programmable Gate Arrays (FPGAs), Trajectory tracking.

1. INTRODUCTION

Classical and model-based control methods need a mathematical description of the system to realize effective and robust performance. However, control frameworks using Fuzzy Logic (FL) are able to compute with linguistic variables and model if-then rules to control systems as black box functions without the need for mathematical models, Hou and Wang [2013]. Unlike conventional and computationally expensive control algorithms, Fuzzy Logic Control (FLC) does not need precise measurements to get admissible results. Often, FLC are often the preferred system to tackle noisy and imprecise sensor readings in robot systems. At lower costs and computationally efficient mechanisms, FLC provides a simple yet effective way to compute and deal with input-output mappings to tackle uncertain, imprecise, noisy and missing information, Somwanshi et al. [2019].

One of the most desirable features of FLC is the practicality and the intuitiveness of incorporating expert's knowledge in the rule-base control. However, such desirable feature also becomes a bottleneck in realizing highly adaptive systems. Often, it is the case that if-then rules become inaccurate in changing environments. This is one of the key reasons behind the adoption of implementation mechanisms of FLC that could be reformulated upon application, i.e. re-configurable controller.

Field Programmable Gate Array (FPGA) are well-known in the field not only as a valuable tool for control implementation through embedded system, but also as an energy and computationally-efficient framework to realize programs suitable for intelligent control methods, Ruiz-Rosero et al. [2019]. For instance, researchers proposed the FPGA and Real-Time modules in LabVIEW software as an efficient hardware for control implementation, Kandidayeni et al. [2022].

On the practical side, LabVIEW setups can be executed in one of two modes. First, as hardware-in-the-loop (HIL), where the host computer controls the real-time mechatronic system's hardware via wireless network protocols that are used to transmit control actions, Bourhnane et al. [2019]. Second, it is also possible to use the system as a standalone real-time framework that executes the control laws independently on its own, Nada and Shaban [2014], Shaban and Nada [2013].

In this paper, FLC-FPGA stand-alone controller is developed, synthesized, and implemented on a reconfigurable FPGA board for tracking systems. Instead of building the membership functions using lookup tables, as mentioned in, Ponce-Cruz et al. [2016], we developed the set of LabVIEW functions based on logic operations and geometric relationships. The proposed method has the advantage of accelerating fuzzy-based control for trajectory tracking problems by utilising the parallel processing of FPGA boards. As such, we implemented the FLC on a FPGA, and termed the overall framework as FLC-FPGA. We also compared our approach with the results obtained using PID-FPGA on LabVIEW Real-Time environment. The obtained results show the feasibility and efficiency for tracking control in embedded mechatronic systems in which the reference signal varies at a frequency not exceeding 50 $[Hz]$.

10.1016/j.ifacol.2023.10.1486

2. PROPOSED APPROACH: FLC-FPGA

2.1 Preliminaries

In Fuzzy Logic (FL), a membership function (MF) maps elements of the set X to the interval $[0,1]$. For $x \in X$, a *triangular* MF is defined as follows:

$$\mu_A(x) = \begin{cases} 0 & x \le a \\ \frac{x-a}{b-a} & a \le x \le b \\ \frac{c-x}{c-b} & b \le x \le c \\ 0 & x \ge c \end{cases} \quad (1)$$

a *trapezoidal* MF is defined by

$$\mu_A(x) = \begin{cases} \frac{x-a}{b-a} & a \le x \le b \\ 1 & b \le x \le c \\ \frac{d-x}{d-c} & c \le x \le d \\ 0 & x \le a \cup x \ge d \end{cases} \quad (2)$$

a *singleton* MF is

$$\mu_A(x) = \begin{cases} 1 & x = c \\ 0 & otherwise \end{cases} \quad (3)$$

An equivalent form for Eq.(1) is

$$\mu_A(x) \equiv A(x) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \quad (4)$$

Also, Eq.(2) has an equivalent form as follows

$$\mu_A(x) \equiv A(x) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right) \quad (5)$$

2.2 Fuzzification

Rather using lookup tables to build MFsPonce-Cruz et al. [2016], we developed LabVIEW functions based on logic operators and fixed geometrical parameters while avoiding negative effects on subsequent control operations.

To overcome the limitation of normal divide and delay in quotient and remainder functions in LabVIEW-FPGA, Eq. (4, 5) can be expressed, without loss of generality, as

$$A(x) = \max(\min(2(x-a), 2(c-x)), 0) \quad (6)$$
$$\Longleftrightarrow b-a = c-b = 0.5$$
$$A(x) = \max(\min(4(x-a), 1, 4(d-x)), 0) \quad (7)$$
$$\Longleftrightarrow b-a = d-c = 0.25$$

Left and right *triangle* MFs can be constructed by setting $a = b$ and $b = c$ respectively. Using Eqs. (6)-(7), the membership values of the input/output variable can be obtained. Thus, given an input x, $A : x \to [0,1]$. To allow using variables in integer format, the corresponding output values are scaled considering the user-preferences on the resolution, as follows:

$$A_{Scaled} = A \times (2^n - 1), \quad (8)$$

where n is the number of bits in the system. As such, floating point values scale to resolution $(2^n - 1)$. In order to ease calculations, the probability scale is suggested to be 10000, whose equivalent n approximates to 14, denoting the saturation limit for the computation.

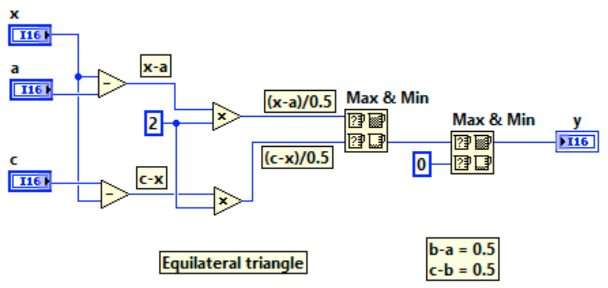

Fig. 1. LabVIEW *TriangleMF.vi*

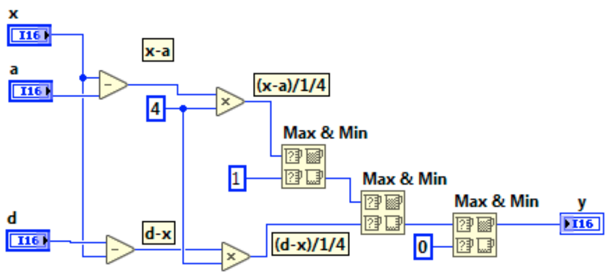

Fig. 2. LabVIEW *TrapezoidalMF.vi*

In LabVIEW-FPGA module, the *triangle* MF can be established through the programming of Eqs. (6)-(7), as presented in Fig. 1. Also, the *trapezoidal* MF is presented in Fig. 2. The `max` & `min` function compares *two* inputs and returns the larger (smaller) value at the top (bottom) output terminal.

Consequently, five triangular membership functions subjected to input's variable can be constructed. Each membership MF_1 to MF_5 describes the probability of the input variable as $(-\Downarrow)$ Negative Big, $(-\downarrow)$ Negative Small, $(\longleftrightarrow)$ Zero, $(+\uparrow)$ Positive Small, and $(+\Uparrow)$ Positive Big. The Labview programming of FPGA-Fuzzification process is constructed via LabVIEW sub-VIs labeled as *5MFTri.vi* by using five *TriangleMF.vi* with the scaled boundaries from -10000 to 10000, see Fig.(3). The output of the *5MFTri.vi* is a 5-element array which denotes the probability of the input conditions.

In the case of manipulating more than one input variable, i.e., two inputs, such as the error term e and the change of the error $\dot{e}$, one can estimate the measure of the influence of membership functions. This can be done by estimating the strength of each rule, w^i, according to the corresponding probabilities, as follows

$$w^i = A_1^i \cap A_2^i, \quad (9)$$

where $A_{1,2}^i$ is the probability of membership function i for inputs 1 and 2 respectively. The fuzzy OR/AND operator selects the maximum/minimum of the two probabilities. Figure (4) shows the LabVIEW block diagram of *CompactMF.vi* that estimates Eq.(9) using the AND operator by utilizing the `max` & `min` Function. The output of *CompactMF.vi* is a vector includes the strength of the rules of one probability of input 1, with all probabilities of the input 2.

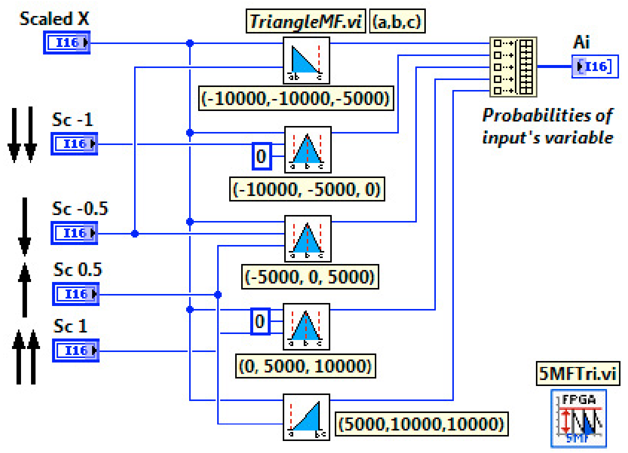

Fig. 3. Input error fuzzy sets - $5MFTri.vi$

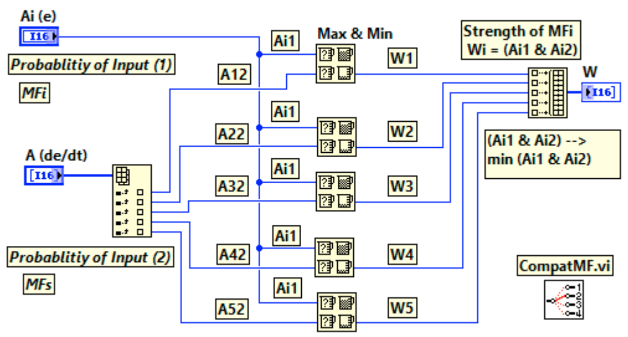

Fig. 4. Strength of inputs' MFs - $CompatMF.vi$

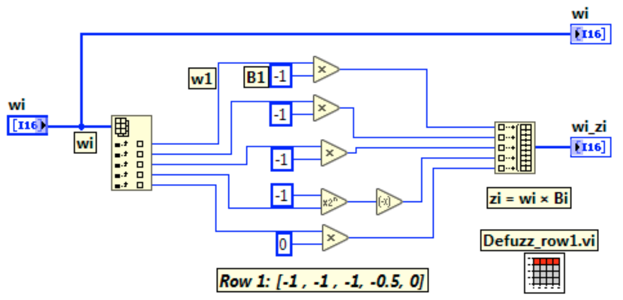

Fig. 5. 1st row of 0-order Sugeno model $Defuzz-row1.vi$

Table 1. Rules Table: $(-\Downarrow)$ Negative Big, $(-\downarrow)$ Negative Small, $(\longleftrightarrow)$ Zero, $(+\uparrow)$ Positive Small, $(+\Uparrow)$ Positive Big.

$e/\dot{e}$	$-\Downarrow$	$-\downarrow$	$\longleftrightarrow$	$+\uparrow$	$+\Uparrow$
$-\Downarrow$	-1	-1	-1	$-1/2$	0
$-\downarrow$	-1	-1	$-1/2$	0	$1/2$
$\longleftrightarrow$	-1	$-1/2$	0	$1/2$	1
$+\uparrow$	$-1/2$	0	$1/2$	1	1
$+\Uparrow$	0	$+1/2$	1	1	1

2.3 FPGA - Inference Engine and DeFuzzification

The rule base encodes if−then relationships of variables in the linguistic space. For example, rule 1: if Error e is $(-\Downarrow)$ & Change of error $\dot{e}$ is $(-\Downarrow)$ then the appropriate value from the rule table c^1 is -1, where $(-\Downarrow)$ means Negative Big. Table (1) shows the appropriate value of each rule, i.e., $c^i s$, where $i \in [N]$, in which N is the number of rules.

Although the well-known Mamdani-type inference engine renders MFs that require defuzzification, it is possible to use a single spike as the output MF rather than a distributed fuzzy set, also known as a singleton output membership function. The above-mentioned concept enhances the efficiency of the defuzzification process because it simplifies the computation required by the more general Mamdani method. Instead of integrating across the two-dimensional function to find the centroid, one can use the weighted average of a relevant subset of data points to calculate the control output, e.g. Sugeno-type systems, whose approach is amenable to symmetric MFs. As such, a rule in a Sugeno fuzzy model is computed as follows:

$$Final\ Output = \sum_{i=1}^{N} w^i z^i \ / \ \sum_{i=1}^{N} w^i \tag{10}$$

where w^i is computed by using Eq.(9) and N is the number of rules according to Table (1).

Three stages of calculations are required to construct the FPGA-Fuzzy Logic controller on LabVIEW. First, in the fuzzification stage, the two inputs are checked to obey the saturation limits ($\pm$10000) and then introduced, individually, to two $5MFTri.vi$ to generate five probabilities of the each input (error and the change of error). Each probability output (one value per membership) of the input 1, i.e., error signal, is fed-forward to $CompactMF.vi$ with all probabilities of the input 2 (change of error), to generate the strength of each pair corresponding to the rule table. For example, the probability of the $MF1$ of the error signal, $A_i(e)$ in the block of Fig.(4), with the probabilities of $MF1$ to $MF5$ of the change of error, $A(de/dt)$, generates the strength vector corresponding to first row of the rule table, Table (1), and therefore the weighted output can be estimated using the block diagram shown in Fig.(5).

Figure (6) shows the complete block diagram of the 3-stages of the FLC-FPGA controller proposed in the paper. In the second stage, step 1., invloves five calls of $CompactMF.vi$ are carried out to generate the strength vector of each row of the rule table, followed by step 2. that estimate the weighted outputs of each row according zero-order Sugeno model. The Defuzzification satge estimates the final output according to Eq. (10). The reader may note that the output is defined as a singleton MF, which is computationally more efficient compared to approaches based on computation of the centroid of mass or interpolations.

2.4 Execution-Time Calculation

The VIs code is mapped to gates and slices on an FPGA-based target, allowing concurrent loops to be realised on separate regions of the FPGA fabric. This enables all computations to perform concurrently (in parallel). Each loop's time is independent of the other loops, with the flexibility to add operations that let loops to interact and exchange data. To measure the execution time of the proposed controller, different timing VIs can be implemented in parallel on the FPGA without affecting the code's performance. Defining the execution time as the difference between consecutive loop iterations, the execution time of different VIs are shown in Table 2, in which the largest consumed time is 2000 $[\mu s]$.

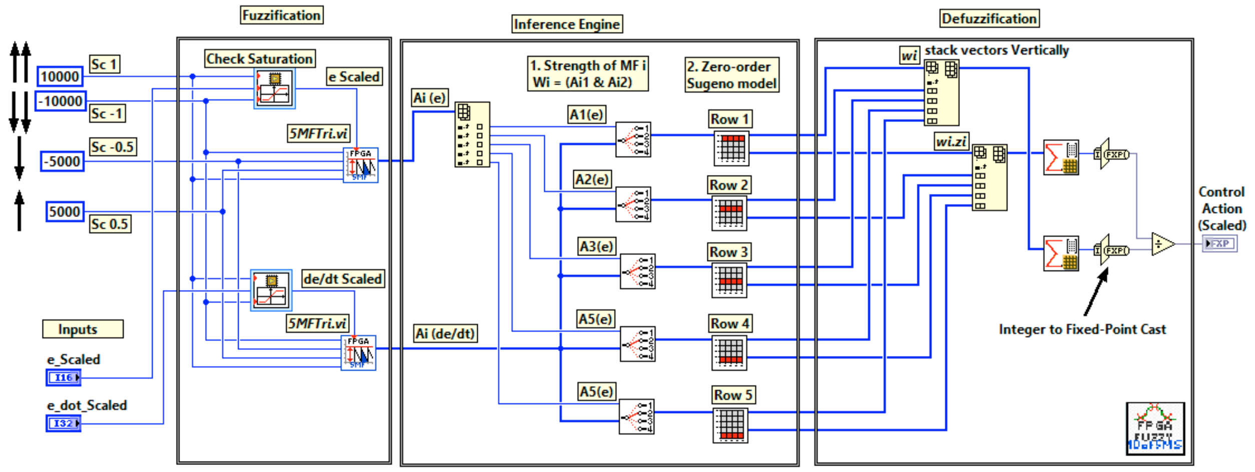

Fig. 6. LabVIEW Block diagram of FLC-FPGA, $FLC-FPGA-5mf-2I-1O.vi$

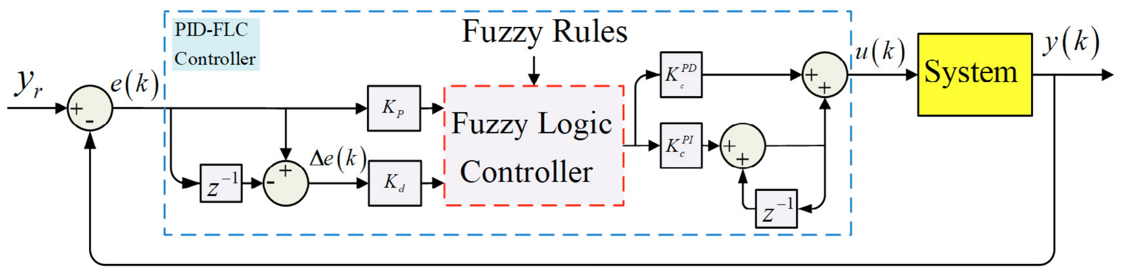

Fig. 7. Block diagram PID-FLC control system

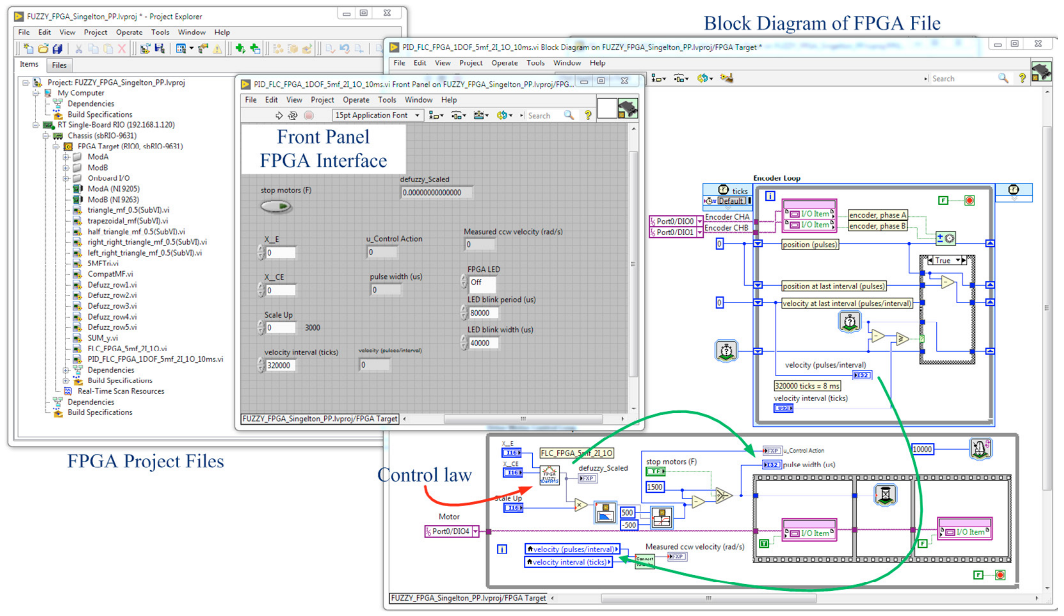

Fig. 8. LabVIEW FPGA Project

Table 2. Execution Time of FLC's VIs.

VIs	$Fig./Eq.$	μs
$Fuzzy\_FPGA\_2DOF\_5mf$	$Fig.(10)$	6
SUM_y	$Eq.(10)$	0
$DaNi\_Robotic\_5mf\_2I...$	$Fig.(14)$	2000

2.5 *Fuzzy Controller*

In line with the main tenet of FL systems, the mathematical model of the system is unavailable; instead, only the system output, y, can be monitored and the system's states, particularly; the error can be extimated and change of error can be approximated. Depending on the type of control type required, i.e., PD, PI, or PID type. the control law of the fuzzy controller can be estimated using the error, $e(k)$, change of error $\Delta e(k)$, and the sum of error $\sum e(k)$ as inputs and control input $u(k)$ as output, where k is the sampling time. Thus, typical proportional differential PD-like FLC can be developed as, Perry et al.

[2007]:

$$u(k) = K_p.e(k) + K_d.\Delta e(k) \tag{11}$$

for proportional gain K_p, differential gain K_d, and error terms defined as follows

$$e(k) = y_r - y(k) \tag{12}$$

$$\Delta e(k) = e(k) - e(k-1) \tag{13}$$

where y_r is the reference input,i.e., desired output and $y(k)$ is the real output. The inverse $z-$ operator, z^{-1}, denotes the unit time delay. The error terms are first normalized by gains K_p and K_d, respectively. The normalized terms are used by the FLC, by which after defuzzification, the FLC renders the control signal within the pre-described range. The control action is then obtained by multiplying the control signal by a additional constant, K_c, known as the output scaling gain. PI control is often expressed as follows:

$$\frac{d}{dt}u(t) = K_p\frac{d}{dt}e(t) + K_I e(t) \tag{14}$$

Using the forward differential approximation, the discrete form of Eq.(14) takes the form of

$$\left(1-z^{-1}\right)u(z) = K_p.\left(1-z^{-1}\right)e(z) + K_I.e(z)$$

Taking the inverse $z-$ transform, yields:

$$\Delta u(k) = K_p\Delta e(k) + K_I e(k) \tag{15}$$

Equation (15) is similar to Eq.(11), however, the output is the difference of the control action and the integral constant is multiplied by the error signal while the proportional constant is muliplied by the error difference. PI-FLC controller accumulate of the output of the PD-FLC, to produce the FLC with the PI effect. The control signal $u(k)$ can be approximated by digital means as follows

$$u(k) = \Delta u(k) + u(k-1) \tag{16}$$

A fuzzy controller structure that resembles a PID-like controller can be created by combining PD-FLC with PI-FLC control actions. Figure (7) depicts the FLC-PID control system's entire structure. The output of the PID-FLC is the sum of the outputs of the PD-FLC and PI-FLC, and it has two scaling gains, $\left(K_c^{PD}, K_c^{PI}\right)$, for the FLCs of type PI and PD, respectively,

$$u(k)|^{PID-FLC} = K_p.e(k) + K_d.\Delta e(k) + \Delta u(k) + u(k-1) \tag{17}$$

2.6 Embedded PID-FLC-FPGA in LabVIEW

To realize parallel processing, one should separate the code into different and independent segments. The set of self-governing loops can be programmed to acquire and quickly process measured analogue or digital data at distinct loop rates. The FPGA layout implements two basic independent loops:

- a loop for measurement and feedback of the measured signal (typically digital signals),
- a loop for generation of the driving signal (typically a Pulse Width Modulation signal) to drive the actuator given the control action.

Along with the above-mentioned fundamental loops, additional loops must be incorporated together during the implementation process. One loop is used to calculate the control law, and another loop is used to pass the collected data to the real-time processor for the updating process. The details of measuring and driving loops are found in literature. Whereas the focus of this paper is concerned in the calculation of the FLC law within the FPGA environment, that is referred to as FLC-FPGA.

In LabVIEW, one can use FPGA project (.lvproj) to manage targets and VIs, references to project files, configuration data; deployment data; build data; and other data. This project uses the sbRIO chassis. The controller (RT Single-Board RIO) attaches instantly to the board chassis and communicates either directly or via a network with the development computer. The FPGA VIs (subroutines of the FLC in the previous section) are the VIs that would be downloaded and run over the FPGA target. Figure (8) shows the FPGA project files as well as the main FLC-FPGA control file, the VI is called as $PID\_FLC\_FPGA\_1DOF\_5mf\_2I\_1O\_10ms.vi$ that call all the above VIs with loop rate of 5 $[ms]$ that is greater than the largest execution time given in Tab.(2). In last, project files are configured and translated throughout the compilation process into a bitfile that can be downloaded to the FPGA target.

3. EXPERIMENTAL WORK

We compare the results of the proposed controller with PID control of the National Instruments robot platform (DaNI 1). The robot platform uses the sbRIO-9631 embedded control board, which integrates a user reconfigurable field programmable gate array being programmable through LabVIEW-FPGA module. The kit includes a ready-made optimal PID control employed in the FPGA environment and referred to in this investigation as PID-FPGA. The controller uses the error terms to render a Pulse Width Modulated signal (PWM) as controls. The frequency of the PWM signal takes a value from $1000-2000$ us. In this section, the obtained results of the proposed PID-FLC-FPGA controller will be compared to the ready-made PID-FPGA controller to demonstrate the efficiency of the proposed PID-FLC-FPGA controller.

The autonomy of the two DaNI motors allows not only driving but also manoeuvrability of the robot. Thus, the controller for each motor should be independent, and the control signal should be calculated for each motor separately. Here, the effectiveness of the FPGA in programming and communicating with the hardware targets becomes clear; the control action can be calculated by adding one more parallel loop or through the For loop structure.

We used a trajectory tracking scenario with a profile that denotes the changing of curvature, see Fig.(10). In particular, the robot will move from the rest position at point A to point B through a straight line and then along the Lemniscate profile, Eq.(18), then again to point B, and finally to point C, where the final velocity and acceleration are zeros. The Lemniscate profile can be expressed as:

$$\mathbf{R}(t) = \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} \frac{a\sqrt{2}\cos t}{\sin^2 t+1} \\ \frac{a\sqrt{2}\cos t\sin t}{\sin^2 t+1} \\ \frac{dy}{dt}\cdot\frac{dt}{dx} \end{bmatrix} \tag{18}$$

Defining the wheel radius as r, and the wheel has a distance ℓ from the origin point. Given r, ℓ, θ, the velocity kinematics models can be constructed to estimate the velocity of the robot

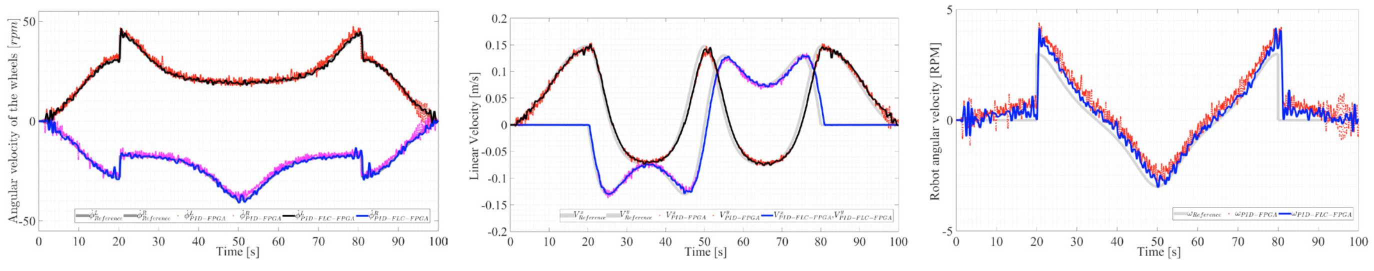
(a) Angular velocity of the robot's wheels (b) Robot's linear velocity (c) Robot's angular velocity

Fig. 9. Overview of velocities.

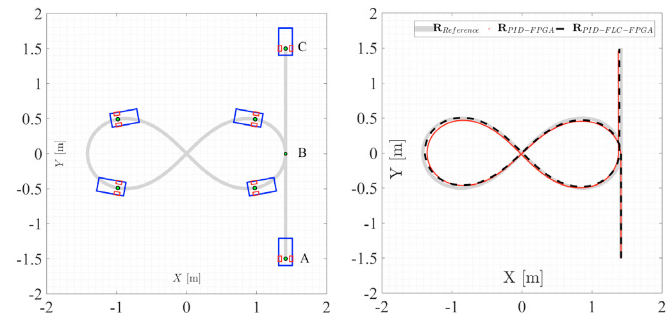
(a) Desired trajectory (b) Achieved trajectory

Fig. 10. Overview desired and achieved trajectory.

in the global reference frame, i.e., $\dot{\mathbf{q}} = [V_x \ V_y \ \omega]^T$. The velocity transformation between the motors angular velocities and the generalized velocities of the robot can be expressed as, Nada and Bashiri [2018]:

$$\begin{bmatrix}\dot{\phi}_R \\ \dot{\phi}_L\end{bmatrix} = \begin{bmatrix} 2/r\cos\theta & 2/r\cos\theta \\ 2/r\sin\theta & 2/r\sin\theta \\ 2\ell/r & -2\ell/r \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} \tag{19}$$

Given the generalized velocity vector, $\dot{\mathbf{q}}$, the tracking problem's reference values for the robot's controller for the right and left wheels, $\dot{\phi}_R$ and $\dot{\phi}_L$, are estimated. Fig. 9-(a) shows the results of the assinged controllers upon the robot's spinning velocities $\dot{\phi}_R$ and $\dot{\phi}_L$. In the case of implementing PID-FPGA, the two motors are controlled by the identical set of PID control parameters, providing that both motors have the same dynamic model. However, since the dynamics of two motors differ, the regulated controlled results clearly distinguish between the right and left motors. The left motor in the PID-FPGA in Fig. 9-(a) deviates more in relation to the reference value. With the PID-FLC-FPGA, because the controller design is independent of the system model, the results are more accurate. Fig. 9-(b,c) show the results of the assinged controllers upon the generalized velocities of the robot, and Fig. 10-(b) shows the real time trajectory of the robot. It should be mentioned that the scope is FPGA implementation of FLC rather than minimization of error with respect to the trajectory baseline, which will be involved in the future work.

4. CONCLUSION

This paper provides a method implementing the three stages that constituent the FLC and PID-FLC-FPGA structure in LabVIEW-FPGA environment. We use fixed math operations on integer data types since complex mathematics is difficult to implement on FPGA due to the target device's lack of floating point operations. Sugeno-type FLC with five triangular membership functions with two inputs and nine singleton output MFs are constructed. The controller is evaluated in tracking a differential-wheeled mobile robot. The experimental results and performances of the PID-FLC-FPGA are compared with the PID-FPGA controller for tracking problems. Our results validate the control design procedure and implementation and additionally show several benefits for high-speed embedded applications. Future work will include controller implementation of large scale control problems and overall system performance improvement.

REFERENCES

Bourhnane, S., Abid, M.R., Lghoul, R., Zine-Dine, K., Elkamoun, N., Bakhouya, M., and Benhaddou, D. (2019). Real-time control of smart grids using ni compactrio. 1–6. IEEE.

Hou, Z.S. and Wang, Z. (2013). From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235, 3–35.

Kandidayeni, M., Macias, A., Trovão, J.P., and Boulon, L. (2022). Control systems with compact-rio implementation. In *Reference Module in Materials Science and Materials Engineering*. Elsevier.

Nada, A.A. and Bashiri, A.H. (2018). Integration of multibody system dynamics with sliding mode control using fpga technique for trajectory tracking problems. volume 3. American Society of Mechanical Engineers.

Nada, A.A. and Shaban, E.M. (2014). The development of proportional-integral-plus control using field programmable gate array technology applied to mechatronics system. *American Journal of Research Communication*, 2, 14.

Perry, A.G., Feng, G., Liu, Y.F., and Sen, P.C. (2007). A design method for pi-like fuzzy logic controllers for dc-dc converter. *IEEE Transactions on Industrial Electronics*, 54, 2688–2696.

Ponce-Cruz, P., Molina, A., and MacCleery, B. (2016). *Fuzzy Logic Type 1 and Type 2 Based on LabVIEW^{TM} FPGA*, volume 334. Springer International Publishing. doi: 10.1007/978-3-319-26656-5.

Ruiz-Rosero, J., Ramirez-Gonzalez, G., and Khanna, R. (2019). Field programmable gate array applications—a scientometric review. *Computation*, 7, 63.

Shaban, E.M. and Nada, A.A. (2013). Proportional integral derivative versus proportional integral plus control applied to mobile robotic system. *Journal of American Science*, 9, 583–591.

Somwanshi, D., Bundele, M., Kumar, G., and Parashar, G. (2019). Comparison of fuzzy-pid and pid controller for speed control of dc motor using labview. *Procedia Computer Science*, 152, 252–260.