IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 8, 2021, accepted March 7, 2021, date of publication April 13, 2021, date of current version April 28, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3073010

Development of a Robotic Hand Using Bioinspired Optimization for Mechanical and Control Design: UnB-Hand

SERGIO A. PERTUZ[1], CARLOS H. LLANOS[1], (Member, IEEE), AND DANIEL M. MUÑOZ[1,2]
[1]Faculty of Technology, Postgraduate in Mechatronic Systems, University of Brasília, Brasília 70910-900, Brazil
[2]Faculty of Gama, Electronics Undergraduate, University of Brasília, Brasília 70910-900, Brazil

Corresponding author: Sergio A. Pertuz (sergio.pertuz@unb.br)

This work was supported in part by the Brazilian Agency for the Improvement of Higher Education Personnel (CAPES) under Grant 88882.383336/2019-01 and in part by the Foundation for Research of the Federal District (FAP-DF) under Grant 00193-00000079/2020-74.

ABSTRACT For the last four decades, the development of robotic hands has been the focus of several works. However, a small part of those approaches consider the exploitation of parallelism of FPGA-based (Field Programmable Gate Arrays) systems or discuss how using bioinspired optimization algorithms could improve the mechanical and controller components. This work considers developing a bioinspired robotic hand that achieves motion and force control with a logic hardware architecture implemented in FPGA intended to be replicated and executed with suitable parallelism, fitting a single device. The developed robotic hand prototype has five fingers and seven DoF (Degrees of Freedom). Using bioinspired optimization, such as PSO (Particle Swarm Optimization), both the rigid finger mechanism and the impedance controller were optimized and incorporated the results in several practical grasping experiments. The validation of this work is done with the Cutkosky grasping taxonomy and some grasping experiments with interference. The tests proved the proficiency of this works for a wide range of power and some precision grasp. The reader can see the experiments in the attached videos.

INDEX TERMS Bioinspired optimization, FPGA, grasping taxonomy, impedance controller, robotic hand, SoC.

I. INTRODUCTION

Over the past four decades, there have been significant contributions in the field of computer science, artificial intelligence, robotics, and other related fields. This progress has allowed the development of robotic systems that are more capable and sophisticated, such as biomimetic robotics. Such robots are more skilled, robust, and efficient than other types of conventional robots when used in unstructured workplaces [1].

There is also much effort towards building biomimetic robotic hands, which have contributed to a better understanding of implementing a human hand into a dexterous gripper/manipulation robot, widening its applications through improved sensors and mechanisms [15]. One of these improvements is tactile sensing, which uses physical signals to identify the phases of object manipulation, namely: (a) non-contact to contact, (b) rotation, and (c) sliding [16]. For the application of robot hands, these phases translate into object recognition, force control, and grasp. Tactile sensors can measure different magnitudes, such as force vectors, vibrations, and contact actions. On the other hand, signal-processing techniques and modeling improve these measurements or even estimate others when sensors cannot read them directly. Such measures are then used for control schemes that perform grasping tasks [17]. Efficient grasping techniques have proven to be complicated on both implementation and computational issues, considering that the diverse robotic hands' components must be controlled and supervised in parallel and real-time. Some works have accomplished these aspects by clustering several data processing devices in parallel. Table 1 presents a summary of some of those works with their main characteristics and achievements. The table includes the following key points: (1) control scheme or

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

TABLE 1. Comparison with some other robotic hand implementations.

Author	Control scheme	Finger gear	Actuators	Sensors	CU	Freq. (Hz)	DoF
Robotic Hand (This Work)	Impedance control	Four-bar linkage and worm gear	7 DC motors	7 angular position, 7 current sensors	SoC ARM+FPGA	>25000	7
KITECH-Hand [2]	PID current control	Spur gears	16 DC motors	16 angular position, 16 current sensors	μCU matrix (16)	333	16
Calderon *et al.* **[3]**	PID control	Four-bar linkage	5 DC motors	5 force sensors	PC	-	5
Jeong *et al.* **[4]**	Neural Networks for position estimation and PID current control	Four-bar linkage	6 DC motors	5 position sensors	PC	-	6
Wang *et al.* **[5]**	Impedance control with internal PID position control	Four-bar linkage	5 DC motors	5 Encoders; 5 hall position; 5 torque sensors	Multiple DSP (2)	10 - 40	5
LMS Hand [6], [7]	Force control with Neural Networks and PID position control	Wire-driven	16 DC motors	16 encoders (for motors), 16 absolute encoders (for joints)	PC	50	16
Lee *et al.* **[8]**	Hybrid PD position/force	Four-bar linkage	9 DC motors	9 incremental encoders, 4 resistive force sensors	PC	-	9
HIT/DLR Prosthetic Hand [9], [10]	Impedance control and PD position control	Four-bar linkage	3 DC motors	3 encoders, 3 hall position, 3 torque, 3 force sensors on fingertips	Multiple DSP (2)	1000	3
HIT/DLR Prosthetic Hand II [11], [12]	Impedance control and PD position control	Wire-driven	15 Brushless DC motors	15 position, 15 torque, 5 force on fingertips, 10 temperature sensors, tactile sensors.	Multi-processor DSP+FPGA (6 DSP & 6 FPGA)	-	20
KNU Hand [13], [14]	Position control with sliding detector	Worm gear and four-bar linkage	2 DC motors	2 position sensors	DSP	-	6

strategy for grasping, (2) finger gear or type of transmission used for the fingers movement, (3) the type of actuators used, (4) the quantity and type of sensors used, (5) the CU or computational control unit, (6) the control loop refresh frequency in Hz, and (7) the number of DoF. It is worth noticing that only works that included the actuators either in the palm or the finger were considered; i.e., robotic hands with actuators in the forearm or outside the hand were not considered.

Table 1 indicates that some implementations use PID for force control [2]–[4]. Although this solution is simple, efficient, and easy to tune, it does not control the dynamics of the contact between the manipulator and object. The impedance controller cannot only do this but also performs well at exerting forces on the environment and achieve good robustness at handling flexible components with unknown stiffness.

On the other hand, Table 1 also shows that the computational unit (CU) of some approaches [2], [5], [12] use several components to achieve the required parallelism of the controllers by using well-known micro-controllers. However, it results in an ample physical space required by the CU and demands implementing communication strategies between the processing devices. Lastly, as expected, more components mean an increase in energy consumption.

Many of the implementations depicted so far include simplifying the complexity of the human hand to achieve embedding. This work's motivation follows that path by scaling down the robotic system's dimensions and complexity, enclosing the CU in a single device or chip. This device should execute complex algorithms fast enough to attend to the control loop and real-time grasping and manipulation requirements. As mentioned before, integrating and centralizing the CU also avoids communication lag between different devices. Finally, reducing the number of components can also aim for a more cheap and energy-efficient robotic hand solution.

Field Programmable Gate Arrays (FPGAs) are a good match in pursuing this motivation. Additionally, these devices can be fundamental in the simultaneous control of several actuators that must be handled synchronously. In conventional processors, the correct synchronization can be hampered by the serial nature of executing instructions in von Neumann-based architectures. In this way, FPGA-based platforms allow the designer to implement efficient digital architectures capable of reading signals in parallel from multiple sensors and generating many output signals through parallel processing. They have been used to implement algorithms for parallel motion control of fingers for piano playing [8]. Another example [18] asserts the importance of the usage of parallelism for tactile sensing in robotic hand applications. Other potential advantages of FPGAs, although not guaranteed, are the possibility of achieving more energy-efficient architectures that can be scalable to more complex systems. Nevertheless, the main drawback of using FPGAs to embed control algorithms is the necessity of proficiency in hardware development from the designer and longer design time, which is greater than developing software.

This work's primary goal is developing a bioinspired robotic hand that achieves motion and force control based on the previously developed logic hardware architecture validated for a single finger [19]. In this sense, this work explores a suitable manner to parallelize the impedance controllers for a complete 7-DoF robotic hand, fitting a single FPGA device. The developed robotic hand prototype has five fingers and seven DoF (Degrees of Freedom). The prototype consists of seven DC motors, seven position-sensors, and six current sensors processed in parallel.

This work's contributions are the following: (a) A novel experimental setup process for implementing a bioinspired robotic hand with an embedded controller using reconfigurable hardware FPGA/SoC (System on a Chip). This approach's main advantage is its capacity to control several DoF in parallel using a single chip that's relatively cheap and energy-efficient, different from other works that need to use a grid of micro-controllers or big processors. (b) The robotic hand fingers are based on the novel bioinspired optimized mechanism in [20], which was vital in reducing the number of needed actuators and allow their inclusion in the palm. Additionally, it also allowed incorporating the sensors and electronics in the palm, avoiding external elements. (c) A new approach for tuning an impedance controller's parameters using a bioinspired Opposition-based learning Particle Swarm Optimization algorithm (OPSO). We portray the OPSO tuning methodology and believe that it be extended for other applications where impedance control can be useful. There is no literature about tuning impedance controllers using bioinspired algorithms to the best of our knowledge. In this work, this controller is used to achieve the robotic hand's dynamic and force control to perform grasp without having tactile sensors. (d) Finally, this work presents real experiments of the proposed FPGA/SoC-based bioinspired robotic hand for the first time. The conveyed experiments consisted of testing several grasping positions using the Cutkosky taxonomy and executing some of those grasps with external interference. With these experiments, we successfully demonstrate the robotic hand capabilities analytically and critically.

This paper is organized as follows: Section I describes the robotic hand mechanical and electronic design, describing the fingers' optimization and listing the used electronic devices. Section II depicts the impedance controller that implements the force control and its tuning using bioinspired algorithms. Section III develops the embedded FPGA/SoC-based system that executes this control, using hardware/software co-design. Section IV unfolds a performance analysis of the previously mentioned embedded system, comparing it to other solutions. Finally, Section V describes the experiments that were carried out to validate this work's results.

II. HAND DESIGN

Human hands have 31 muscles and 19 articulations that actuate at least 25 DoF (Degrees of Freedom) [21]: four in every upper finger, of which three are responsible for flexion-extension and one for adduction-abduction; four in the thumb, where two are for flexion-extension and two for opposability. The rests lay on the palm and wrist's rotation and translation.

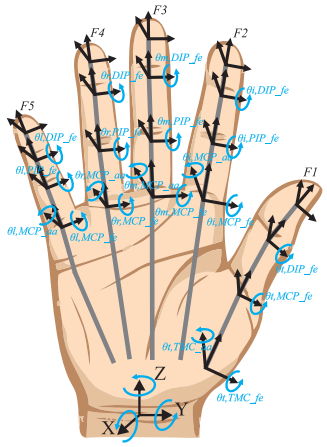


FIGURE 1. DoF of a human hand. Where DIP is Distal-Inter-Phalangeal joint, PIP is Proximal-Inter-Phalangeal joint, MCP is MetaCarpal-Phalangeal joint, TMC is CarpoMetaCarpal Joint, *fe* is flexion-extension, and *aa* is abduction-adduction.

Fig. 1 illustrates the large number of joints or DoF of a human hand. Ideally, a biomimetic hand should emulate all DoF. However, this achievement has proven to be very difficult due to space, energy, and other physical restrictions. Recent solutions to this problem have involved reducing the number of fingers [2], locating the actuators outside the hand to achieve a similar amount of DoF in the robot [22], [23], transforming and reducing the number of DoF [3], [4], [16], [24].

For the final solution, this work takes advantage of the fact that some joints in the human hand restrict the state of others [21]. Therefore, their movement set is constrained and defined mathematically regarding other joints. With this in mind, different human hands' DoF can be hierarchized according to the relevance of its actions when performing particular tasks, such as grasping [25]. Table 2 lists the most relevant joints of the human hand, where *aa* and *fe* states the movements of the MCP and TMC joints for the front and thumb fingers, respectively. The joints with a "no" are sub-actuated according to the others' state, and *yes* indicates the actuated joints. Table 3 defines the mathematical relation of the constrained ones.

Figure 2 presents the proposed kinematic structure. It depicts the five fingers and their joints as sixteen rotatory joints, seven actuated while the rest behaves according to the constraints seen in Table 3.

Table 4 shows the ranges of the joints and phalangeal length of the fingers, which were established based on real human hand for Latin-American individuals [26].

The parameters on Table 4 establish the robot's workspace, which is the total volume the fingertips can reach when combining every possible joint configuration [27]. Fig. 3 exhibits the proposed robotic hand workspace using a cloud

TABLE 2. Ten more relevant human hand joints when performing grasping tasks. Refer to Fig. 1 for the positions of the joints. Adapted from Cobos [25].

No.	Finger	Articulation	Is used in this work
1	Thumb	θ_{t,TMC_aa}	yes
2	Thumb	θ_{t,TMC_fe}	yes
3	Index	θ_{i,MCP_aa}	yes
4	Index	θ_{i,PIP_fe}	yes
5	Middle	θ_{m,MCP_fe}	yes
6	Middle	θ_{m,PIP_aa}	no
7	Ring	θ_{r,MCP_fe}	yes
8	Ring	θ_{r,PIP_aa}	no
9	Little	θ_{l,MCP_fe}	yes
10	Little	θ_{l,PIP_aa}	no

TABLE 3. Interphalangeal constraints for the actuated joints. Adapted from Cobos [25].

Thumb	Index	Middle	Ring	Little
θ_{t,TMC_aa}				
	θ_{i,MCP_aa}			
$\theta_{t,MCP\_fe} \approx \frac{4}{5}\theta_{t,DIP\_fe}$	$\theta_{i,MCP\_fe} \approx \frac{4}{3}\theta_{i,PIP}$	$\theta_{m,MCP\_fe} \approx \frac{4}{3}\theta_{m,PIP}$	$\theta_{r,MCP\_fe} \approx \frac{4}{3}\theta_{r,PIP}$	$\theta_{l,MCP\_fe} \approx \frac{4}{3}\theta_{l,PIP}$
$\theta_{t,DIP}$	$\theta_{i,PIP} \approx \frac{3}{2}\theta_{i,DIP}$	$\theta_{m,PIP} \approx \frac{3}{2}\theta_{m,DIP}$	$\theta_{r,PIP} \approx \frac{3}{2}\theta_{r,DIP}$	$\theta_{l,PIP} \approx \frac{3}{2}\theta_{l,DIP}$
	$\theta_{i,DIP}$	$\theta_{m,DIP}$	$\theta_{r,DIP}$	$\theta_{l,DIP}$

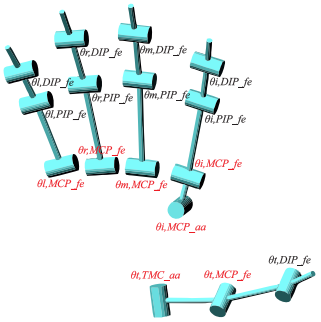


FIGURE 2. Kinematic structure of 7 DoF simplified robotic hand. Actuated joints are highlighted in red, the other ones are sub-actuated according to Table 3

TABLE 4. Kinematic parameters of the robotic hand.

Joint	Phalangeal lengths of the finger [mm]					range [^{o}]
	Thumb	Index	Middle	Ring	Little	
MCP_aa	-	17	-	-	-	$0-15$
TMC_aa	38	-	-	-	-	$0-90$
MCP_fe	37.67	45.77	51.36	47.60	37.67	$0-90$
PIP_fe	-	25.75	30.30	29.51	20.84	$0-67.5$
DIP_fe	20.84	18.20	20.02	19.90	18.36	$0-45$

of fingertips' points. It illustrates the intersections between the upper fingers and the thumb, validating its opposability. This figure also denotes that the middle, ring, and little fingers' workspace is a 2-dimensional spline due to the single DoF they possess. The following subsection describes the resulted design for the fingers using a four-bar linkage mechanism and how it was optimized to perform the constraints depicted in Table 3.

A. MODELLING AND OPTIMIZATION OF ROBOTIC FINGER MECHANISM

Pertuz *et al.* [20] proposed a robotic finger mechanism that located the actuators in the palm of the robot hand, which is useful when the robotic hand needs to be adapted to larger and

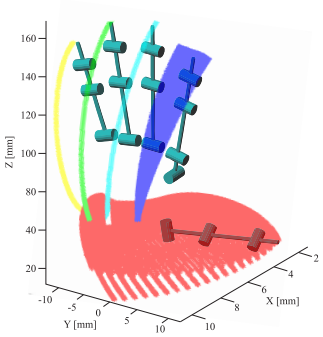


FIGURE 3. Robotic hand workspace.

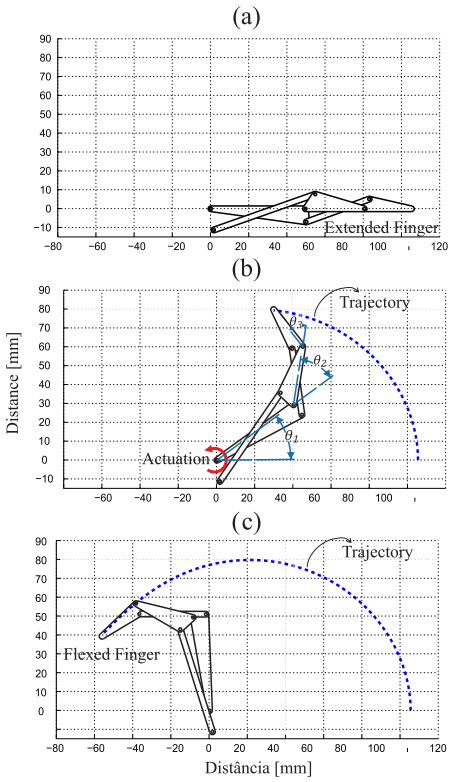


FIGURE 4. Coupled 4-bar mechanism. (a) Finger fully extended; due to the mechanisms self-lock feature, in this position, it only allows counter-clock wise movement input, (b) Finger upon input movement in θ_1, and (c) Fully flexed finger.

more complex systems [15]. The drawback of this approach is that the space in the palm and finger of a human-sized robot hand is minimal, thus the necessity of reducing the number of DoF.

The *fe* movements of the fingers are sub-actuated with 1 DoF, i.e. they have a single motion input and the other joints move at a certain proportion, through a four-bar linkage mechanism. Fig. 4 presents the mechanism and its motion path, the input of the mechanism is θ_1 (from here onwards the *MCP*, *PIP*, and *DIP* joints will be referred as θ_1, θ_2, and θ_3, see Fig. 1).

Fig. 4(c) also describes two fingertip trajectories: 1) the one developed by the mechanism using trivial link lengths named as *mechanism's fingertip path*, and 2) the one generated by the

TABLE 5. Optimized decision variables for the coupled 4-bar mechanisms of the robotic hand for all fingers. Values are in mm.

	Proximal-Medial Link				Medial-Distal Link			
	$P_{a2}\hat{i}$	$P_{a2}\hat{j}$	L_{a1}	L_{b1}	$P_{a2}\hat{i}$	$P_{a2}\hat{j}$	L_{a1}	L_{b1}
Ring	3.26	3.44	46.09	5.98	3.12	3.44	27.64	6.59
Index	3.14	3	44.32	5.75	3	3	24.12	5.75
Middle	2	2	50.1	5.56	3.75	3.29	28.52	5.75
Little	2.56	2.93	35.95	6.33	4.1	2.82	17.64	6.14
Thumb	2.56	2.93	35.95	6.33				

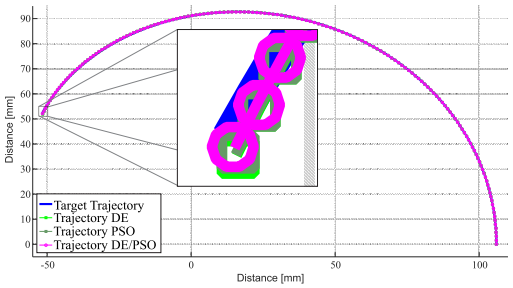


FIGURE 5. Optimized fingertip paths using PSO and DE.

constraints on Table 3 referred as *desired fingertip path*. For both trajectories, the proximal, medial, and distal phalanges have the values listed in Table 4. The first path depends on the mechanism's link lengths and is expected to fit the second. The solution to this problem is not straightforward; however, it can be seen as an optimization problem where the variables are the link's positions, and the objective is to fit the fingertip path.

In order to optimize the four-bar mechanism, the bioinspired optimization algorithms PSO (Particle Swarm Optimization) [28], DE (Differential Evolution) [29] and GA (Genetic Algorithm) [30] were used. The results of each algorithm were analyzed, compared, and selected for a final prototype. The optimization was performed for one finger and adapted in proportion to the others. The decision variables are listed in Table 5 and Fig.5 illustrates the optimized mechanism's fingertip path for the index finger. The readers are referred to [20] for more details regarding the mechanical design optimization procedure.

B. DESIGN AND ASSEMBLING

This project enclosed the novel four-bar finger mechanism optimized in previous work [20]. It allowed to reduce the DoF and thus the mechanical components of the flexion-extension and adduction abduction movements, permitting them to be enclosed in the palm. The robotic hand was built using mainly 3D printed parts; however, we used other manufacturing processes for the metallic pieces. The finger mechanism depicted in the previous subsection is anthropomorphized for a better appearance, as shown in Fig. 6.

The motors used in this project are low-power brushed DC motors (200 RPM and 0.28 Nm) and actuate the mechanism through a worm gear to bring more grasping torque. The worm mechanism also allows rotation of the movement axis 90^o, easing the location of the motors (see Fig. 6(c)).

The thumb and index fingers' configurations have extra actuated joints for their *aa* movements. This project proposes

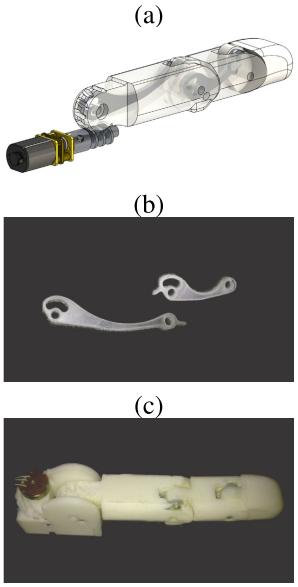


FIGURE 6. Antropomorphed index finger mechanism. (a) CAD with transparent finger depicting the phalanges and the links of the mechanism, and photos of (b) the proximal-medial and medial-distal links, and (c) the assembled finger. Most of the robotic hand was 3D printed with ABS plastic except for the worm gear and links that are fabricated in aluminum.

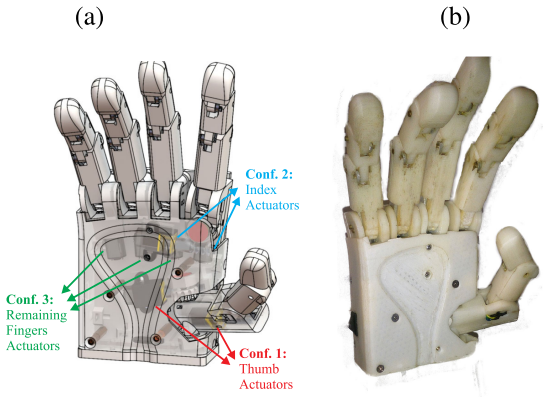


FIGURE 7. Assembled Robotic Hand. (a) Render of assembled robotic hand with the locations of the actuators and extra DoF of the index and thumb. (b) Real picture of the assembled robotic hand.

three different finger configurations (see Fig. 2 for viewing the joints), as follows. (1) Configuration 1 (Thumb): 2 DoF (*aa* and *fe*) and 3 joints. (2) Configuration 2 (Index): 2 DoF (*aa* and *fe*) and 4 joints. (3) Configuration 3 (Remaining fingers): 1 DoF (*fe*) and 3 joints.

The extra DoF of the thumb and index fingers were added as seen in Fig. 7(a). Fig. 7(b) also illustrates the prototype robotic hand's final result.

The thumb opposability of this work can be quantified using an adaptation of the well-known Kapandji clinical test [31]. This test assesses the thumb's opposition by checking its ability to touch a specific part of the hand. They are ten tests/positions in increasing difficulty, with the easiest consists of touching the proximal phalanx of the index finger and the hardest is to touch the distal palmar crease.

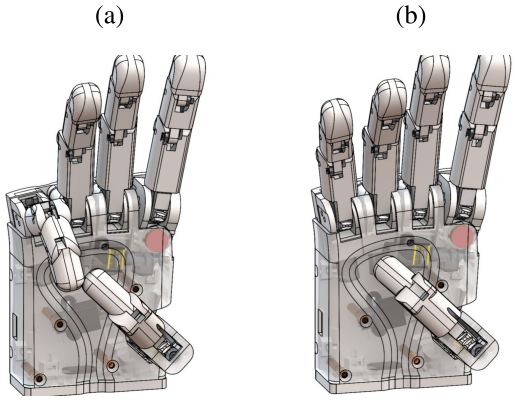


FIGURE 8. Kapandji Test for robotic hand. (a) Position six: thumb touches the little finger. (b) Position ten: thumb touches distal palmar crease.

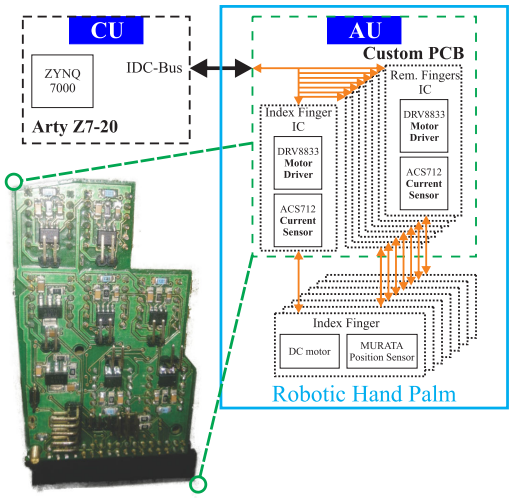


FIGURE 9. Schematic diagram the Robotic Hand Acquisition Unit (AU) and Controller Unit (CU). The entire AU is embedded in the palm while the CU is the arty board that is located outside. On the south-west corner is a photo of the custom PCB of robotic hand.

Fig. 8 depicts two of the most representative Kapandji tests (six and ten) validating this work's thumb opposable ability by achieving the most challenging position.

C. ELECTRONICS

The robotic hand electronics are embedded in the palm and, excepting the position sensor, are located in a custom PCB. The signals are accessed through an IDC 34-pin header that connects to the controller device. Figure 9 illustrates the schematic of the controller dividing it into two parts the Acquisition Unit (AU) and the Controller Part (CU).

The AU includes the motor drivers (DRV8833), the current sensors (ACS712), and some other power regulation ICs (Integrated Circuits). It also has connection pins for the IDC breakthrough and sockets for the DC motors and the analog angular position sensors. The PCB has dimensions of approximately $80 \times 52mm$ and has an irregular form designed to fit in the palm (See Fig. 9). The CU consists of the

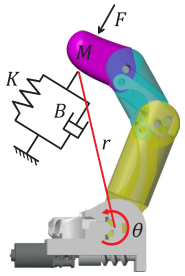


FIGURE 10. Schematic diagram of the impedance model in the robotic finger.

Arty Z7-20 development board. It contains an XC7Z010 SoC (System on a Chip) that includes a Zynq®-7000, an Artix™-7 FPGA (Field Programmable Gate Arrays), and a Dual-Core ARM®Cortex®-A9 at 667MHz. The chip also includes a XADC (Analog to Digital Converter) necessary for the current and position sensors.

III. DESIGN OF THE ROBOTIC FINGER FORCE CONTROLLER

Several approaches can be implemented when attempting to grasp with a robotic hand [32]. This work considers a scheme that plans to grasp an object whose physical characteristics are unknown using only position and motor current to estimate the fingers' torque, avoiding tactile sensors. The amount of torque/force is controlled by the algorithms in order to sustain a stable grasp. Previous works have already considered this approach [33], [34].

Impedance control is adopted to control the torque produced by DC motors of the fingers. An impedance controller aims to control the dynamics a robot has when interact with its environment [35], [36]. Fundamentally, the impedance controller is defined as a second-order dynamic system, such as a mass-damper-spring system, with adjustable parameters. A decoupled and linear behavior is considered in this work, i.e., every finger has an independent controller with torque and position feedback, see Fig. 10. The controller is represented by Eq. 1

$$Mr\ddot{\theta}(t) + Br\dot{\theta}(t) + Kr\theta(t) = F(t), \tag{1}$$

where F is the resulting force of the system, M, B and K are mass, damping, and spring coefficients, respectively, and r is the distance between the first joint and the fingertip.

The control parameters, M, B, and K, can have any value; however, their tuning can be complicated for stable system response. The tuning of control parameters of several types (PID, Neural Networks, and others) using bioinspired optimization algorithms have been implemented in [37], [38]. This work proposes implementing an OPSO (Opposition-based learning Particle Swarm Optimization) algorithm to tune the impedance controller's parameters with torque feedback and a closed-loop to achieve the desired response for a step input.

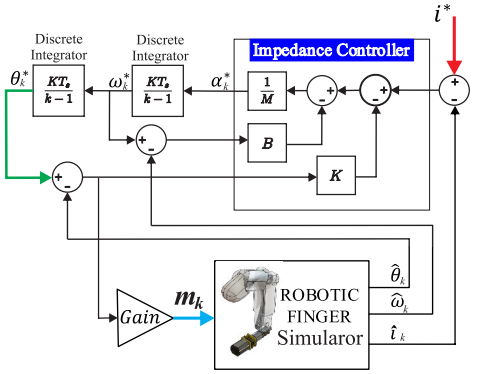


FIGURE 11. Dataflow of finger simulator impedance controller (k=1 to 7). The colored lines highlight the data-path that the control interface override for testing (See Chapter III.C).

A. IMPEDANCE CONTROLLER SCHEME AND SIMULATION

The implementation of Eq. 1 as the controller of the system is done using a discrete integration method [39]. This is described through three steps:

1) Discretize Eq. 1 as follows:

$$\alpha^*(t+1) = (Mr)^{-1}(\overline{F}(t) - Kr\,\overline{\theta}(t) - Br\,\overline{\omega}(t)) \quad (2)$$

where the Force F on Eq. 1 is replaced with the Force Error at a moment t, $\overline{F}(t)$, $\overline{\theta}(t)$ is the tracking position error and $\overline{\dot{\theta}}$ is the tracking angular speed error.

2) The product of the previous step is the target angular acceleration at the next step $t+1$; $\ddot{\theta}^*(t+1)$, which is integrated to obtain the tracking angular velocity:

$$\omega^*(t+1) = \int_{t-1}^{t} \alpha^*(t+1)d\theta \quad (3)$$

3) Finally, the tracking angular position, $\theta^*(t+1)$, is obtained integrating the tracking velocity.

$$\theta^*(t+1) = \int_{t-1}^{t} \omega^*(t+1)d\theta \quad (4)$$

Fig. 11 presents the block diagram for the control scheme for a k DoF and its data-flow. This model aims to detect objects that obstruct the movement upon closing the finger and maintain a desired gripping force i^* (measured with the motor current $\hat{i}_k$ directly considering it is directly proportional to the force F in Eq. 1).

The *Robotic Finger Simulator Block* includes the friction forces of the mechanism and the action of an object that hampers its movement. The simulator was designed in Matlab/Simulink and includes current, position, and speed transducers. It performs a flexion movement with an object that hampers its path. The simulator is better described in [40].

B. OPTIMIZATION OF IMPEDANCE CONTROLLER

The tuning of the impedance controller parameters is a complicated and expensive procedure. One of the main reasons is that the designer must consider coupling effects between multiple joints. Tuning can become even more complex when the robot dynamic has to behave with high accuracy and high speed. The reader is referred to [41]–[43] to find several approaches for auto-tuning impedance controllers. This problem can be addressed with bioinspired optimization algorithms such as PSO (Particle Swarm Optimization), as explored in this work's mechanical design. Some related applications of PSO tuning for robot controllers can be found in [44]–[46]. With these examples as a basis, it is possible to extend PSO to tune impedance controllers with little effort.

Algorithm 1 Pseudo-Code for the OPSO Algorithm

function OPSO(S, N, c_1, c_2, Max_{iter}, $OBLMax_{iter}$, $threshold$)
  Start swarm;
  $iter = 1$
  $OBL_{iter} = 1$
  repeat
    for i **do** 1 S
      if $f(x_k) \leq f(y_{ik})$ **then**
        $y_{ik} \leftarrow x_k$
        $OBL_{iter} = 1$
    calculate y_s using the S fitness values $f(y_{ik})$
    for i **do** 1 S
      for j **do** 1 N
        $v_{ij}^{(t+1)} \leftarrow wv_{ij}^{(t)} + c_1U_{1j}(y_{ij}^{(t)} - x_{ij}^{(t)}) + c_2U_{2j}(y_{sj}^{(t)} - x_{ij}^{(t)})$
        $x_{ij}^{(t+1)} \leftarrow x_{ij}^{(t)} + v_{ij}^{(t+1)}$
    $OBL_{iter} = OBL_{iter} + 1$
    if $OBL_{iter} > OBLMax_{iter}$ **then**
      $x^{(t+1)} \leftarrow a + b - x^{(t)}$
      $OBL_{iter} \leftarrow 1$
    $iter = iter + 1$
  until $(f(y_{y_s}) < threshold)||(Iter <= Max_{iter})$
  return y_{best}

The central part of an optimization problem is the design of the cost function. For this case, it is to track the desired force i_k^* with the decision variables being the controller's coefficients (M, B, and K). It is calculated by observing the estimated motor current and extracting some characteristics from the response, such as (a) the over-impulse percentage (O_i), (b) the necessary time for the finger to reach collision with the object (t_o), (c) the settling time after the collision is reached (t_e) and, (d) the force tracking Mean Squared Error (MSE). The selection of these criteria was made empirically and are weighted according to

$$f_{cost} = 0.05(O_i) + 0.15(t_o) + 0.15(t_e) + 0.65(MSE), \quad (5)$$

where the weights were decided empirically.

Due to the lack of knowledge about the coefficients, a random initial position is fed to the algorithm. OPSO is a modification to the original PSO algorithm that attempts to avoid falling on local minimums. This diversification uses the opposition-based learning (OBL) method [47] when an individual best is not improved after a certain quantity of

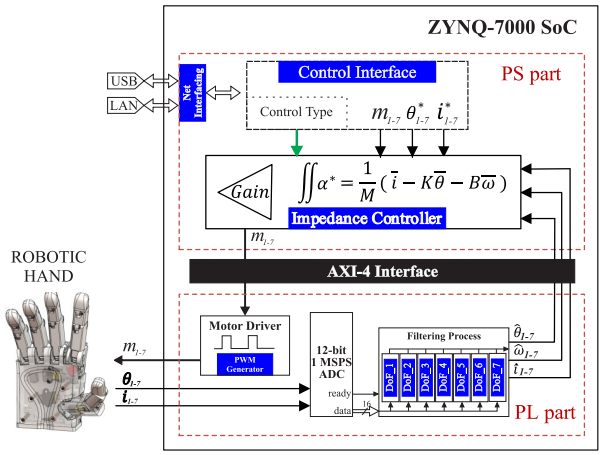


FIGURE 12. Internal architecture of the CU. The Filtering Process block is replicated for the same amount of DoF (Degrees of Freedom) in the robotic hand. These blocks, including the Motor Driver, are controlled and synchronized via the AXI interface with the PS-part. The PS-part also executes the impedance controller algorithm and the control interface.

iterations. The following algorithm describes how the OPSO works.

The minimum error achieved by the PSO after 500 iterations was 0.0725 with the decision variables of $K = 5.0774$, $B = 0.3450$ and $M = 0.0052$. Finally, the resulting coefficients were tested for the simulator's controller with a collision time of 3.565 *seconds*, a settling time of 0.335 *seconds*, and a steady-state error of 0.012.

IV. IMPLEMENTATION OF THE EMBEDDED ROBOTIC FINGER FORCE CONTROLLER

The robotic hand aims to embed the low-level control algorithms. Some authors approach this issue by distributing the computation between several devices to comply with real-time constraints [2]. This work proposes, in novelty, implementing the filtering process and the impedance controller as a reconfigurable architecture in an FPGA/SoC device. Doing so enables implementing computing-intensive functions in parallel, in a single chip, with little compromise to time-execution performance and energy consumption.

The Filtering Process block (presented in Fig. 12) is the most expensive part of the control scheme. So, this part is determined to be described in VHDL (VHSIC ''Very High-Speed Integrated Circuit'' Hardware Description Language) on the CU's FPGA (Field Programmable Gate Arrays), from now on referred to as the Programmable Logic Part / PL-part). It is essential to point out that hardware implementation has some advantages, like logic architecture's intrinsic parallelism. On the other hand, a drawback is that the designer must be mindful not to surpass the number of logical resources on the chip.

Moreover, the Impedance Controller block in Fig. 11 is executed in the CU's Processor Unit (from now referred to as Programmable Software Part / PS-part). The composition of the CU's architecture and the data flow is described in Fig. 12. Similarly, the robotic hand *Control Interface* block is implemented in the PS-part. It can switch the control technique between manual, position, and impedance control; this is described more thoroughly in the following subsection.

The platform-chip used in the CU allows communication between the PS- and PL-part, enabling the creation of hybrid systems combining the advantages of software and hardware, i.e., the possibility of implementing hardware accelerators in the PL-part and the flexibility of software.

A. PL-PART: SENSORS FILTERING PROCESS

The PL-part includes an ADC converter, the motor PWM signals generator (Motor Drivers), communication protocol (AXI Interface), and the sensors Filtering Process. The latter implements two filters for the analog current and position sensors: (1) a Kalman filter [48] that estimates position and velocity for the analog position sensor; and (2) a second-order low-pass filter for the current sensor.

The architecture uses FSM to control data flow and operations. There are four arithmetical operators (two adders, one multiplier, and one divider) that can be used in parallel and shared between states and two filters. Additionally, the operators are custom-made [49] and use a 27-bit floating-point numeric representation. This bit resolution is used because it preserves the best *resource utilization/precision* ratio for the selected FPGA device. A more thorough description of the Filtering process is depicted in [19], [50].

B. PS-PART: IMPEDANCE CONTROLLER

The controller block involves the impedance controller scheme introduced in Section III. It performs all the scheme steps using the sensor filter outputs and the motor's current reference. It is important to note that it cannot be executed in parallel given the step's equations' data dependency, which is another reason not to implement it on hardware. The controller's output is fed to the motor driver block as seen in Fig. 12. The PS-part is also in charge of the data synchronization between modules through the AXI Interface.

C. PS-PART: CONTROL INTERFACE

This work uses a text-based visual interface (see Fig. 13) that enables the user to control the robotic hand. The interface allows setting a series of predefined control values ($i^*_{1,:,:7}$) for different grasps and manually controlling each finger separately. The ''***'' below the fingers' names indicates the currently controlled one. The control variable of that finger can be increased or decreased. The user can also declare which control variable will be adjusted between three options by overriding it in the impedance controller dataflow. This circumvention of the dataflow is depicted in Fig. 11 with the red, green and blue lines for i^*, θ^* and m^* respectively. Resuming the control can be done with three strategies:

1) MANUAL_MOTORS mode overrides the m^* variable (Seen as ''Volt'' on Fig. 11), allowing to control the voltage of the motors of the robotic hand manually; and hence, the flexion-extension movements of every finger. It sends a constant pulse to the motors vetoing the output of the proportional gain.

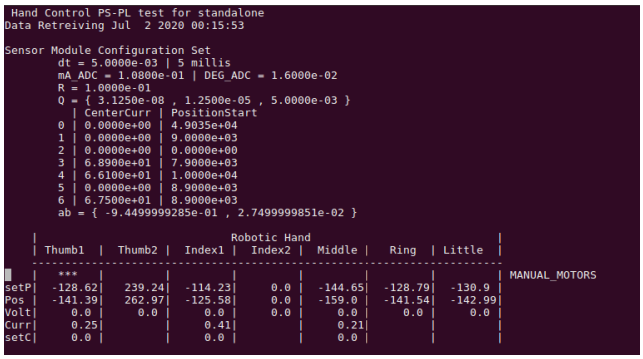


FIGURE 13. Robotic hand visual interface. It illustrates some relevant data for the sensors calibration and filters parameters. It also shows sensors and control data: *setP* and *Pos* are the desired and filtered angular position of a finger, *Volt* is the voltage of a finger's motor, and *Curr* and *setC* are the filtered sensor and desired current respectively. Finally, it also specifies the current control strategy for testing (MANUAL_MOTORS, P_ONLY or FULL_IMPEDANCE).

TABLE 6. Synthesis utilization of full controller system for one finger and the full hand.

Resource Utilization	PSPL-Arty 1 Finger (%)	PSPL-Arty Full Hand (%)	PL-Arty 1 Finger (%)	PL-Arty Full Hand (%)	ArtyZ720 Available
LUT	1976 (3,71)	12252 (23,03)	2888 (5,43)	22521 (42,33)	53200
LUTRAM	0 (0,00)	0 (0,00)	0 (0,00)	62 (0,36)	17400
FF	1917 (1,80)	10714 (10,07)	3522 (3,31)	25702 (24,16)	106400
DSP	2 (0,91)	14 (6,36)	3 (1,36)	21 (9,55)	220

2) P_ONLY: Bypasses the desired position of a finger (θ_k^* and shown as "setP"on Fig. 11). Setting this control variable makes it possible to set a finger in the desired position, nullifying the impedance controller's action.
3) FULL_IMPEDANCE: Is the full proposed impedance controller scheme. It updates the control variable i^* ("setC" in Fig. 11) into the impedance controller. It feeds the desired current for grasping objects.

D. RESULTS AND SYNTHESIS ANALYSIS

An FPGA has limited resources, such as LUTs (Look-Up Tables), slice registers, DSPs (Digital Signal Processors), and BRAM (Block RAM). Table 6 presents the consumption of those resources for the implementation of the system described in Fig. 12 in the column under the name *PSPL-Arty*. Additionally, for contrast, the table also presents the resource consumption of the full-hardware implementation (without using the PS-part); these results are described in the column under the name *PL-Arty*.

For full-logic implementation, almost half of the FPGA resources were used, allowing little space for further development and other characteristics the work might need in the future. In contrast, the hardware-software implementation results (PSPL-Arty) drastically reduced the resource utilization of the FPGA.

TABLE 7. Control system implementation on different platforms.

ID	Type of Arch	Platform	OS	Clock Frequency
PS-Arduino	Software -Only	ATmega2560	Bare metal	16MHz
PS-ArtyBM	Software -Only	ARM Cortex-A9	Bare metal	650MHz
PS-ArtyLnx	Software -Only	ARM Cortex-A9	Linux	650MHz
PL-Arty	Full -HW	XC7Z020	-	100MHz
PSPL-ArtyBM	Hybrid	ARM Cortex-A9 / 1CLG400C	Bare metal	650MHz / 100MHz
PSPL-ArtyLnx	Hybrid	ARM Cortex-A9 / 1CLG400C	Linux	650MHz / 100MHz

TABLE 8. Control system implementation on different platforms.

ID	1 DoF t[us]	1 DoF Freq[kH]	7 DoF t[us]	7 DoF Freq[kH]
PS-Arduino	1017.00	0.98	7219.00	0.14
PS-ArtyBM	5.24	190.84	32.92	30.38
PS-ArtyLnx	13.72	72.88	87,80	11.39
PL-Arty	1.20	830.00	1.44	692.52
PSPL-ArtyBM	3.38	303.03	21.16	47.26
PSPL-ArtyLnx	5.85	170.94	39.00	25.64

V. PERFORMANCE ANALYSIS OF THE CONTROLLER

This section compares the implementation of this control system using various architectures and platforms. The controller is implemented as three different structures, exploiting the modules created in the previous subsection: 1) The architectures named as "software-only" perform the systems using only software. 2) Similarly, the architectures named "Full-HW" perform the system using only the PL-part. 3) Finally, the architecture proposed in Fig. 12 is named Hybrid.

The software-only implementations are replicated in different platforms, comparing cost and performance. Table 7 lists the different architectures, platforms, and Operating Systems (OS) implemented in this work.

In total, six approaches were performed in this work. The comparison consists of executing the control loop and measuring the execution time of each iteration for one DoF (Degree of Freedom) and then for the seven. Table 8 lists the execution time consumed for every architecture.

All the PS (Programmable Software) approaches used the same code with minor adaptations for some functions that were not compatible between platforms. Table 8 shows a big timing gap between the PS-Arduino approach and the rest, which is an unfair comparison in many cases. However, it allows the execution time to be compared with a much cheaper solution than the *Arty-Z720* used on the other approaches. The PS-Arduino strategy's execution demonstrates that it can only achieve a maximum control frequency of 140 *Hz*.

The different approaches in the *Arty-Z720* have each their advantages. For instance, the applications in Bare-Metal (*PS-ArtyBM* and *PSPL-ArtyBM*) can be executed in real-time; this guarantees the correct execution of the controller at a fixed frequency. Real-time implementation is not as simple for the cases with the Linux kernel used in the OS (*PS-ArtyLnx* and

PSPL-ArtyLnx) because it is not real-time. However, using Linux in the platform provides more flexibility for interfacing and networking for future project applications. Connectivity for this project is essential given that this work is intended as part of a more extensive application where several elements need to communicate with each other.

The performance of the approaches using software-only is better on bare-metal than with Linux-OS. Nevertheless, the hybrid system of Linux with PL (*PSPL-ArtyLnx*) stretches this difference in the execution time. Additionally, the issue regarding real-time in Linux is minimized in this approach, given that the PL (Programmable Logic) part executes the control in real-time.

On the other hand, the dynamic energy is related to the user design's power consumption on the FPGA (Field Programmable Gate Arrays). In contrast, the static power represents the steady-state intrinsic leakage of the transistors on the device. For the hybrid system *PSPL*, the used static power is $0.115W$, and the dynamic is $1.467W$. Most of the latter is related to the *PS-Part* ($1.39W$), given that the Cortex A9 included in the chip is a high-performance multicore processor. Nonetheless, the energy consumption of the proposed architecture is only $0.077W$.

A. COMPARISON TO RELATED WORKS

According to table 1, previous works [9], [10] have achieved the most considerable control loop frequency (around 1 kHz) for a robotic hand with 3 DoF, using two DSP devices, directly sensing position, torque, and force on fingertips. In contrast, this work accomplishes a control loop frequency of approximately 25 kHz of a robotic hand with 7 DoF, using a single chip device and sensing position and current to estimate torque.

In terms of DoF and control technique, works in [13], [14] developed a robotic hand with 6 DoF using only two DC motors and only one DSP device for controlling position with sliding detection. However, control loop frequency is not reported, and the dexterous is limited since only two DC motors are used. A robotic hand with 16 DoF with force and position control was developed in [6], [7]; however, it is limited in terms of space given that it uses a PC, achieving a control loop frequency of 50 Hz. An embedded solution to force control of a robotic hand with 16 DoF was developed in [2]. The authors used a PID current control and embedded the solution in a matrix of 16 uCs and achieving a control loop frequency of 333 MHz. Finally, works in [11] and [12] implemented an embedded solution for impedance and position control of a robotic hand with 20 DoF using 6 DSPs and 6 FPGAs devices. This massive parallel solution does not report the control loop frequency but probably increases the energy consumption and the controller board complexity.

Using a single chip for all DoF is a novel contribution of this work that saves the embedded controller board's real-estate and can be more efficient in energy consumption and performance. Additionally, the current system can be scaled; i.e., if the number of DoF increases, the hardware architecture can also be adapted. As reported in Table 6, there are enough hardware resources for implementing more filters and processing more DoF, still achieving a high control loop frequency.

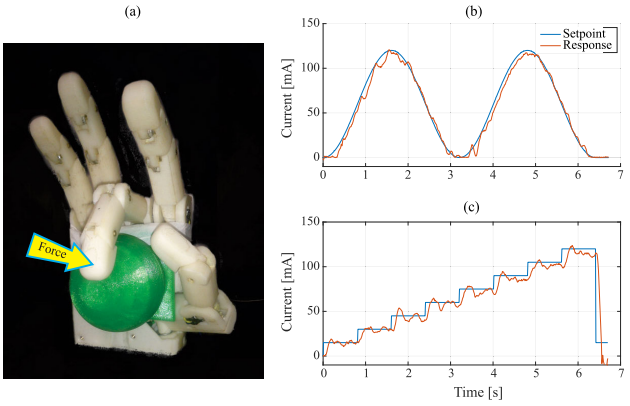


FIGURE 14. Control behavior of impedance controller of one finger. (a) Robotic hand experimental setup, (b)-(c) Current response to a sine wave and step input.

VI. EXPERIMENTS

The finger's control response using the proposed PSPL-ArtyLnx architecture is smooth and is tested to grasp various objects. Fig. 14 illustrates the control response of the sensors for one finger.

It can be seen it stabilizes for a set current that is proportional to the grip torque. The impedance controller sends the position signal to the proportional controller that calculates the motor's input voltage.

A. IMPEDANCE CONTROLLER INTERFERENCE TESTING

The *Arty-Z720*'s available ADC channels allow controlling the five fingers of the robotic hand simultaneously, enabling experiments that involve grasping objects with specific interference.

The test consists of setting a current setpoint for the impedance controller and then see how it behaves upon contact with a cylindrical object and with some type of interference afterward. It is important to highlight that the setpoint values are selected empirically. Figure 15 illustrates these experiments.

Figure 15(a)-(e) illustrates the first test of grasping a rigid cylindrical object, where immediately after contact (around the 20th second), a user carries out interference to evaluate if the robotic hand can maintain the grasp. First, in Fig. 15(b) the object is pushed outwards and then sideways Fig. 15(c)-(d). The opposable thumb is the most affected finger in this type of interference, and it can be seen in its curve. However, the thumb can maintain the grasp and is stabilized after the interference is stopped, around the second 40.

The second experiment (Fig. 15(f)-(h)) details the curves for when the finger enters in contact with a non-rigid object on its surface. It is then performed a series of increased

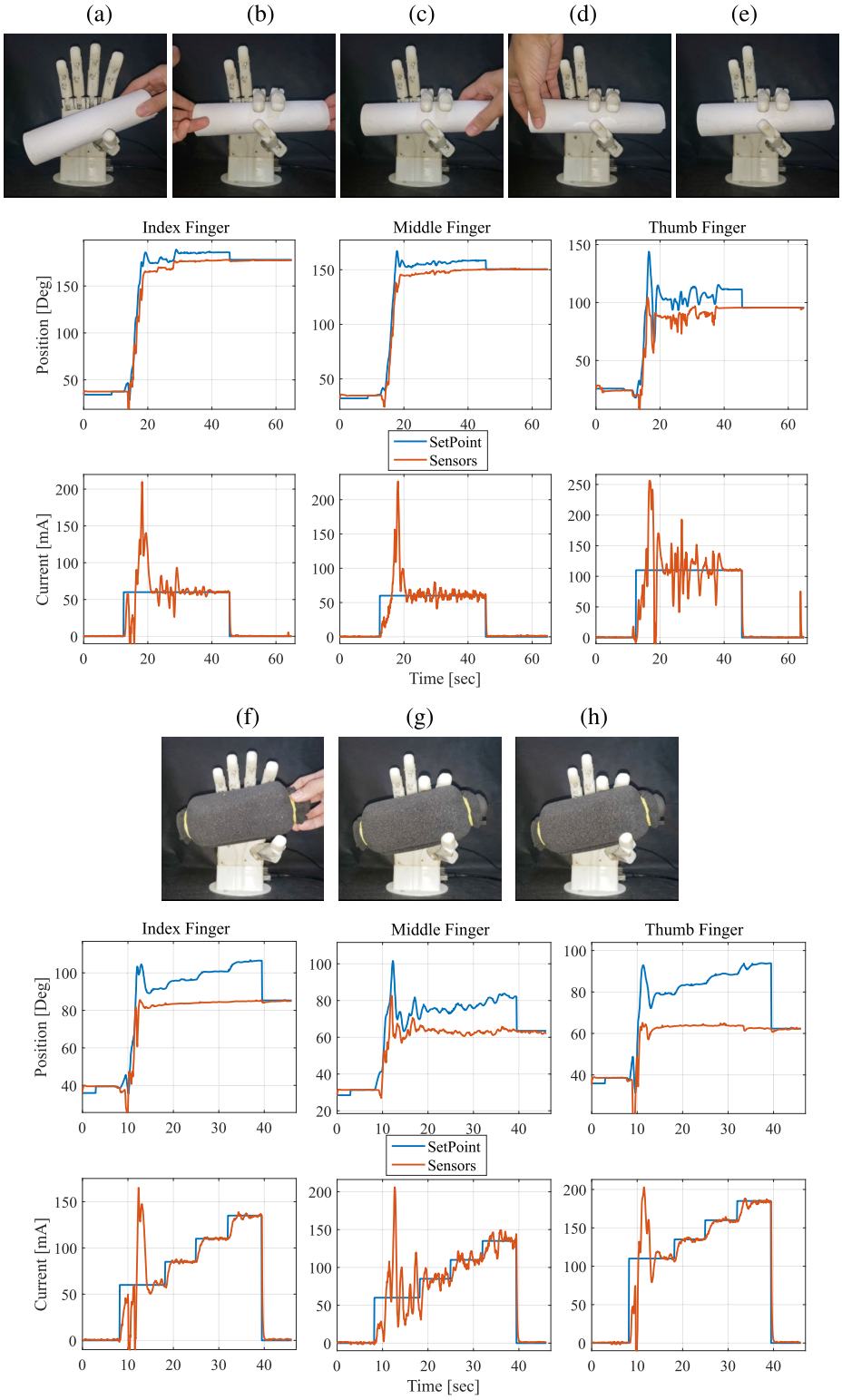


FIGURE 15. Grasping tests with interference. (a)-(e) is the first experiment and (f)-(h) is the second. Bellow the experiments are the curves for current and position for the index, middle and thumb fingers. The pictures are in chronological order. See video.

steps to see if the fingers interject more on the object upon increasing the grasping force. It can be seen that this is not the case because the initial setpoint is already high enough to interject the object the maximum amount. This value cannot be less because the fingers will not start moving because of the work gear inertia.

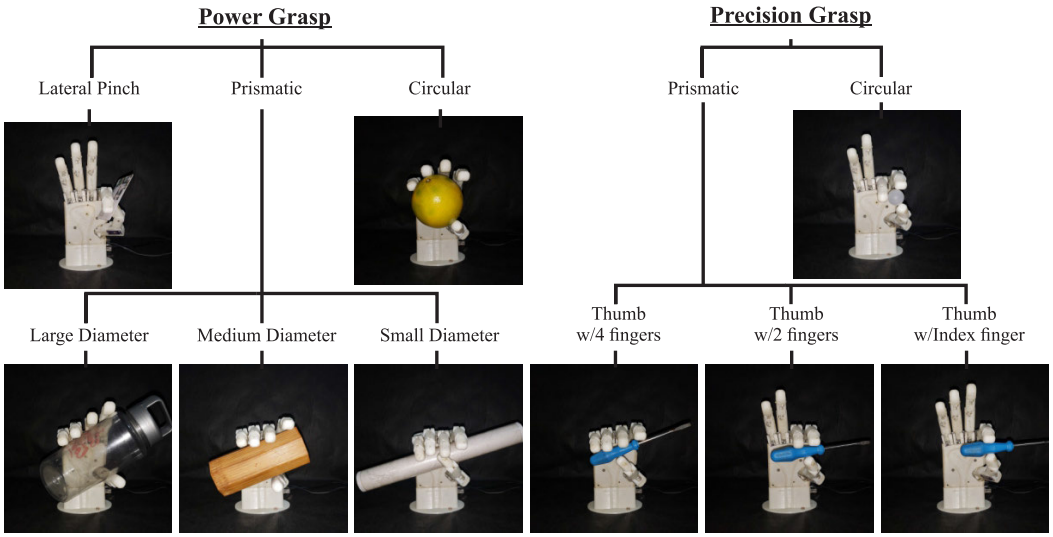


FIGURE 16. Robotic hand performing grasping poses according to the Cutkosky taxonomy. See video.

B. CUTKOSKY TAXONOMY TESTING

Additional experiments are performed to test if the present work can simulate human grasping motion despite DoF reduction (Degrees of Freedom). The Cutkosky taxonomy [51] is a human grasping classification that has been extensively used for manipulation and machining tasks. The different types of grasping poses are classified into power and precision grasps. The experiments are executed by this work's robotic hand by maintaining an object's static grasping pose from the taxonomy.

Figure 16 shows the robotic hand performing some poses from said taxonomy. The robotic hand can perform five power-grasp and four precision-grasp successfully, demonstrating its ability to grasp and manipulate some objects, enabling a vast amount of applications. These results also show the relevance of the *TMC_aa* for most grasping poses.

VII. CONCLUSION

In this project, a biomimetic robotic hand was implemented using an FPGA/SoC-based (Field Programmable Gate Arrays / System on a Chip) approach using a Zynq chip from Xilinx. The biomimetic hand's mechanical design contains 7 DoF (Degrees of Freedom) compared to the 24 an actual human hand has. The number and specific joints of the project were selected according to the most significant DoF and how well they perform gripping objects. Given that some movements are constrained by other DoF in this project, the fingers execute the flexion-extension movements with 4-link mechanisms. A bioinspired optimization was applied to solve a mechanism that generates the necessary trajectories for emulating a human finger's action. The optimization process reached an average quadratic error of $0,00266$. This result was extended for the five fingers' mechanical design according to their size, allowing the full robotic hand's implementation similar to a real hand. Reaching dimensions of $210 \times 84 \times 38mm$ with $413.13gr$ (for the robotic hand design) versus $175 \times 89 \times 43mm$ with $409,5gr$ (for a regular male hand). Additionally, the thumb's ability of this project was demonstrated with the Kapandji test. The prototype was able to reach all ten levels of finger positioning.

The impedance controller developed in this work was first evaluated via numerical simulation. For this action, a robotic finger simulator was implemented for tuning the controller parameters without endangering the prototype. The impedance controller's tuning was performed with bioinspired optimization algorithms, precisely the OPSO (Opposition-based learning Particle Swarm Optimization). The optimization process resulted in an underdamped motor current response, efficiently achieving a rising time of $355ms$ and a steady-state error of $1,2\%$. This bioinspired tuning method for an impedance controller of a robotic system is a new approach and one of this work's main contributions.

As reported in Section V, the embedded controller was developed through several approaches: PS-Arduino, PS-ArtyBM, PS-ArtyLnx, PL-Arty, PSPL-ArtyBM, PSPL-ArtyLnx. The experiments conducted with the proposed PSPL architectures of the impedance controller proved that the tuning performs correctly outside of the simulator. However, complex physical phenomena such as dynamic friction, gaps, and tolerances in the physical prototype (specifically the worm gear), introduced several oscillations in the system's response. This is also derived from the oscillations of the position set point. Despite that, the response of the controller remains stable for most of the experiments.

The execution time had to achieve a maximum of one millisecond for every DoF, and for this reason, the PS-Arduino approach was ruled out to implement the full hand. The PSPL-ArtyLnx method applies the hybrid system using software for interfacing and communication. Programmable logic enables the robotic hand's control in real-time while

achieving an execution time much higher than the required (25.64 kHz). It is essential to highlight that even though the achieved frequencies are not required for the current system, this architecture opens a path for other platforms with faster dynamics.

Finally, this project's prototype could grasp objects with different geometries with power and precision grip, highlighting the robotic hand's dexterity and flexibility. Also, The results of the experiment in Fig. 15(f)-(h) showed that, for low force grasping, it is still necessary a complementary strategy or data to the impedance controller, such as tactile sensing. One could be a combined impedance/position control that performs a position control until contact with an object is reached, switching to the impedance control. These are objectives for future works.

REFERENCES

[1] R. Vepa, "Biomimetic robotics," in *Engineered Biomimicry*, Newnes, Ed. Amsterdam, The Netherlands: Elsevier, 2013, pp. 81–105.

[2] D.-H. Lee, J.-H. Park, S.-W. Park, M.-H. Baeg, and J.-H. Bae, "KITECH-hand: A highly dexterous and modularized robotic hand," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 2, pp. 876–887, Apr. 2017.

[3] C. A. Calderon, C. Ramirez, V. Barros, and G. Punin, "Design and deployment of grasp control system applied to robotic hand prosthesis," *IEEE Latin Amer. Trans.*, vol. 15, no. 2, pp. 181–188, Feb. 2017.

[4] S. H. Jeong, K.-S. Kim, and S. Kim, "Designing anthropomorphic robot hand with active dual-mode twisted string actuation mechanism and tiny tension sensors," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1571–1578, Jul. 2017.

[5] X. Wang, Y. Liu, D. Yang, N. Li, L. Jiang, and H. Liu, "Progress in the biomechatronic design and control of a hand prosthesis," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 5880–5885.

[6] N. Daoud, J. P. Gazeau, S. Zeghloul, and M. Arsicault, "A real-time strategy for dexterous manipulation: Fingertips motion planning, force sensing and grasp stability," *Robot. Auto. Syst.*, vol. 60, no. 3, pp. 377–386, Mar. 2012.

[7] J. P. Gazeau, S. Zehloul, M. Arsicault, and J. P. Lallemand, "The LMS hand: Force and position controls in the aim of the fine manipulation of objects," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, vol. 3, May 2001, pp. 2642–2648.

[8] S. Lee, S. Noh, Y. Lee, and J. H. Park, "Development of bio-mimetic robot hand using parallel mechanisms," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2009, pp. 550–555.

[9] D. W. Zhao, L. Jiang, H. Huang, M. H. Jin, H. G. Cai, and H. Liu, "Development of a multi-DOF anthropomorphic prosthetic hand," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, Dec. 2006, pp. 878–883.

[10] H. Huang, L. Jiang, Y. Liu, L. Hou, H. Cai, and H. Liu, "The mechanical design and experiments of HIT/DLR prosthetic hand," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, Dec. 2006, pp. 896–901.

[11] H. Liu, K. Wu, P. Meusel, N. Seitz, G. Hirzinger, M. H. Jin, Y. W. Liu, S. W. Fan, T. Lan, and Z. P. Chen, "Multisensory five-finger dexterous hand: The DLR/HIT hand II," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 3692–3697.

[12] T. Lan, Y. W. Liu, M. H. Jin, S. W. Fan, H. G. Fang, J. J. Xia, and H. Liu, "DSP&FPGA-based joint impedance controller for DLR/HIT II dexterous robot hand," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechtron.*, Jul. 2009, pp. 1594–1599.

[13] J.-U. Chu, D.-H. Jung, and Y.-J. Lee, "Design and control of a multifunction myoelectric hand with new adaptive grasping and self-locking mechanisms," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2008, pp. 743–748.

[14] D.-H. Jeong, J.-U. Chu, and Y.-J. Lee, "Development of myoelectric hand with infrared led-based tactile sensor," *J. Inst. Control, Robot. Syst.*, vol. 15, no. 8, pp. 831–838, Aug. 2009.

[15] C. Piazza, G. Grioli, M. G. Catalano, and A. Bicchi, "A century of robotic hands," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 2, no. 1, pp. 1–32, May 2019.

[16] E. Mattar, "A survey of bio-inspired robotics hands implementation: New directions in dexterous manipulation," *Robot. Auton. Syst.*, vol. 61, no. 5, pp. 517–544, May 2013.

[17] Z. Kappassov, J.-A. Corrales, and V. Perdereau, "Tactile sensing in dexterous robot hands—Review," *Robot. Auton. Syst.*, vol. 74, pp. 195–220, Dec. 2015.

[18] O. Oballe-Peinado, J. A. Hidalgo-Lopez, J. Castellanos-Ramos, J. A. Sanchez-Duran, R. Navas-Gonzalez, J. Herran, and F. Vidal-Verdu, "FPGA-based tactile sensor suite electronics for real-time embedded processing," *IEEE Trans. Ind. Electron.*, vol. 64, no. 12, pp. 9657–9665, Dec. 2017.

[19] S. A. Pertuz, C. Llanos, C. Peña, and D. Muñoz, "A parallel system-on-chip approach for impedance controller for a 7-DoF robotic hand," *Anal. Integr. Circuits Signal Process.*, vol. 106, no. 1, pp. 195–204, Apr. 2020.

[20] S. A. Pertuz, C. H. Llanos, and D. M. Munoz, "Bioinspired optimization of a robotic finger mechanism," in *Proc. XIII Latin Amer. Robot. Symp. IV Brazilian Robot. Symp. (LARS/SBR)*, Oct. 2016, pp. 199–204.

[21] H. van Duinen and S. C. Gandevia, "Constraints for control of the human hand," *J. Physiol.*, vol. 589, no. 23, pp. 5583–5593, Dec. 2011.

[22] Shadow Robot Company. (Aug. 2015). *Shadow Dexterous Hand Technical Specification*. [Online]. Available: http://www.shadowrobot.com/products/dexterous-hand/

[23] A. A. M. Faudzi, J. Ooga, T. Goto, M. Takeichi, and K. Suzumori, "Index finger of a human-like robotic hand using thin soft muscles," *IEEE Robot. Autom. Lett.*, vol. 3, no. 1, pp. 92–99, Jan. 2018.

[24] M. Tavakoli and A. T. de Almeida, "Adaptive under-actuated anthropomorphic hand: ISR-SoftHand," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 1629–1634.

[25] S. Cobos, M. Ferre, and R. Aracil, "Simplified human hand models based on grasping analysis," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 610–615.

[26] O. Binvignat, A. Almagià, P. Lizana, and E. Olave, "Aspectos biométricos de la mano de individuos chilenos," *Int. J. Morphol.*, vol. 30, no. 2, pp. 599–606, Jun. 2012.

[27] K. Waldron and J. Schimiedeler, "Kinematics," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Germany: Springer-Verlag, 2008, ch. 1, pp. 9–33.

[28] H. R. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *Proc. CIMCA-IAWTIC*, vol. 1, 2006, pp. 695–701.

[29] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[30] D. Whitley, "A genetic algorithm tutorial," *Statist. Comput.*, vol. 4, no. 2, pp. 65–85, Jun. 1994.

[31] A. Kapandji and M. Torres, *Fisiología Articular: Esquemas Comentados de Mecánica Articular. Hombro, Codo, Pronosupinacion, Muñeca, Mano. 1*, (Fisiología Articular. Tomo 1. Hombro, Codo, Pronosupinación, Muñeca, Mano), M. Torres, Ed. Madrid, Spain: Editorial Médica Panamericana, 2006.

[32] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—A survey," *IEEE Trans. Robot.*, vol. 30, no. 2, pp. 289–309, Apr. 2014.

[33] M. Regoli, U. Pattacini, G. Metta, and L. Natale, "Hierarchical grasp controller using tactile feedback," in *Proc. IEEE-RAS 16th Int. Conf. Hum. Robots (Humanoids)*, Nov. 2016, pp. 387–394.

[34] S. B. Godfrey, M. Bianchi, A. Bicchi, and M. Santello, "Influence of force feedback on grasp force modulation in prosthetic applications: A preliminary study," in *Proc. 38th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Aug. 2016, pp. 5439–5442.

[35] N. Hogan, "Impedance control: An approach to manipulation: Part II—Implementation," *J. Dyn. Syst., Meas., Control*, vol. 107, no. 1, pp. 8–16, 1985.

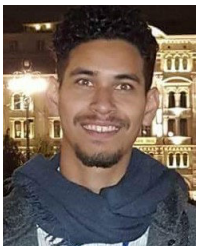
[36] N. Hogan, "Impedance control: An approach to manipulation: Part I—Theory," *J. Dyn. Syst., Meas., Control*, vol. 107, no. 1, pp. 1–7, 1985.

[37] W. He, W. Ge, Y. Li, Y.-J. Liu, C. Yang, and C. Sun, "Model identification and control design for a humanoid robot," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 1, pp. 45–57, Jan. 2017.

[38] J. Sergey, S. Sergei, and Y. Andrey, "Comparative analysis of global optimization-based controller tuning methods for an exoskeleton performing push recovery," in *Proc. 20th Int. Conf. Syst. Theory, Control Comput. (ICSTCC)*, Oct. 2016, pp. 107–112.

[39] F. Caccavale, C. Natale, B. Siciliano, and L. Villani, "Integration for the next generation: Embedding force control into industrial robots," *IEEE Robot. Autom. Mag.*, vol. 12, no. 3, pp. 53–64, Sep. 2005.

[40] C. H. Llanos, D. Munoz, and S. A. P. Mendez, "Simulation and implementation of impedance control in robotic hand," in *Proc. 24th ABCM Int. Congr. Mech. Eng.*, 2017, pp. 1–10.
[41] S. Dehghani, H. D. Taghirad, and M. Darainy, "Self-tuning dynamic impedance control for human arm motion," in *Proc. 17th Iranian Conf. Biomed. Eng. (ICBME)*, Nov. 2010, pp. 1–5.
[42] B. Maldonado, M. Mendoza, I. Bonilla, and I. Reyna-Gutiérrez, "Stiffness-based tuning of an adaptive impedance controller for robot-assisted rehabilitation of upper limbs," in *Proc. 37th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Aug. 2015, pp. 3578–3581.
[43] P. Balatti, D. Kanoulas, G. F. Rigano, L. Muratore, N. G. Tsagarakis, and A. Ajoudani, "A self-tuning impedance controller for autonomous robotic manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 5885–5891.
[44] Y. Raza, S. F. Ahmed, A. Ali, M. K. Joyo, and K. A. Kadir, "Optimization of PID using PSO for upper limb rehabilitation robot," in *Proc. IEEE 5th Int. Conf. Eng. Technol. Appl. Sci. (ICETAS)*, Nov. 2018, pp. 1–4.
[45] J. Campos, S. Jaramillo, L. Morales, O. Camacho, D. Chávez, and D. Pozo, "PSO tuning for fuzzy PD + I controller applied to a mobile robot trajectory control," in *Proc. Int. Conf. Inf. Syst. Comput. Sci. (INCISCOS)*, 2018, pp. 62–68.
[46] E. Mendez-Flores, E. M. Martinez-Galicia, J. D. J. Lozoya-Santos, R. Ramirez-Mendoza, R. Morales-Menendez, I. Macias-Hidalgo, A. Vargas-Martinez, and A. Molina-Gutierrez, "Self-balancing robot control optimization using PSO," in *Proc. 5th Int. Conf. Control Robot. Eng. (ICCRE)*, Apr. 2020, pp. 7–10.
[47] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, and M. Ventresca, "Enhancing particle swarm optimization using generalized opposition-based learning," *Inf. Sci.*, vol. 181, no. 20, pp. 4699–4714, Oct. 2011.
[48] G. Einicke, *Smoothing, Filtering and Prediction: Estimating the Past, Present and Future*, A. Garry, Ed. Rijeka, Croatia: InTech, Feb. 2012.
[49] D. M. Muñoz, D. F. Sanchez, C. H. Llanos, and M. Ayala-Rincón, "Trade-off of FPGA design of a floating-point library for arithmetic operators," *J. Integr. Circuits Syst.*, vol. 5, no. 1, pp. 42–52, Nov. 2010.
[50] S. A. Pertuz, C. Llanos, C. A. Pena, and D. Munoz, "A modular and distributed impedance control architecture on a chip for a robotic hand," in *Proc. 31st Symp. Integr. Circuits Syst. Design (SBCCI)*, Aug. 2018, pp. 1–6.
[51] M. R. Cutkosky, "On grasp choice, grasp models, and the design of hands for manufacturing tasks," *IEEE Trans. Robot. Autom.*, vol. 5, no. 3, pp. 269–279, Jun. 1989.



SERGIO A. PERTUZ received the B.Sc. degree in mechatronics engineering from the University of Pamplona (UPA), and the M.Sc. degree in mechatronics systems from the University of Brasilia (UnB). He has experience in Systems on Chip (SoC), microchip programming and FPGA, mechanical and electrical designs, and process automation.

CARLOS H. LLANOS (Member, IEEE) received the B.S. degree in electrical engineering from the University of Valle, Cali, Colombia, in 1983, the M.Sc. degree in computer science from the University of Minas Gerais-UFMG, Minas Gerais, Brazil, in 1990, and the Ph.D. degree in electrical engineering from the University of São Paulo, Brazil, in 1998. Since 2003, he has been lecturing in the graduate program in Mechatronics Systems with the Department of Mechanical Engineering, University of Brasilia-UnB, Brasilia, Brazil. His current research interests include reconfigurable systems for automation, intelligent systems, instrumentation, and embedded systems design.

DANIEL M. MUÑOZ received the B.S. degree in physics engineering from Universidad del Cauca, Colombia, in 2003, and the M.Sc. and Ph.D. degrees in mechatronics systems from the University of Brasília, Brazil, in 2006 and 2012, respectively. Since 2012, he has been with the University of Brasília. He is currently an Assistant Professor with the Electronics Undergraduate Program, Faculty of Gama, and with the Mechatronics Graduate Program, Department of Mechanical Engineering. His research interests include digital circuit design using FPGAs, artificial intelligence, and bio-inspired optimization algorithms.