International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 10, No. 3, November 2021, pp. 195~211
ISSN: 2089-4864, DOI: 10.11591/ijres.v10.i3.pp195-211

FPGA based co-design of a speed fuzzy logic controller applied to an autonomous car

Emna Aridhi[1], Decebal Popescu[2], Abdelkader Mami[3]
[1,3]Laboratoire d'Application de l'Efficacité Énergétique et des Énergies Renouvelables-LAPER, Faculté des Sciences de Tunis, Université de Tunis El Manar, Campus Universitaire Farhat Hached, Tunis, Tunisie
[2]Faculté d'Automatique et ordinateurs, Université Politehnica de Bucharest, Romanie

Article Info

Article history:

Received Jun 4, 2021
Revised Jul 27, 2021
Accepted Aug 13, 2021

Keywords:

FPGA
Fuzzy logic
Hardware design
Software design
Speed control

ABSTRACT

This paper invests in FPGA technology to control the speed of an autonomous car using fuzzy logic. For that purpose, we propose a co-design based on a novel fuzzy controller IP. It was developed using the hardware language VHDL and driven by the Zynq processor through an SDK software design written in C. The proposed IP acts according to the ambient temperature and the presence or absence of an obstacle and its distance from the car. The partitioning of the co-design tasks divides them into hardware and software parts. The simulation results of the fuzzy IP and those of the complete co-design implementation on a Xilinx Zynq board showed the effectiveness of the proposed controller to meet the target constraints and generate suitable PWM signals. The proposed hardware architecture based on 6-LUT blocks uses 11 times fewer logic resources than other previous similar designs. Also, it can be easily updated when new constraints on the system are to be considered, which makes it suitable for many related applications. Fuzzy computing was accelerated thanks to the use of digital signal processing blocks that ensure parallel processing. Indeed, a complete execution cycle takes only 7 us.

Corresponding Author:

Emna Aridhi
Department of Physics, Université de Tunis El Manar
Campus Universitaire Farhat Hached, Tunis, Tunisie
Email: emna.aridhi@fst.utm.tn

1. INTRODUCTION

Nowadays, technological advances in hardware and software have facilitated communication and exchange between several objects (machines, parts, peripherals and people) through networks of smart sensors and actuators. This makes them interactive and part of autonomous systems or/and applications. Among the autonomous systems that are currently emerging are electrical autonomous vehicles. They are massively used in several areas like transport, agriculture, residential, industrial maintenance, production, etc. Indeed, they allow reducing the air pollution rate, and improving the reliability of systems in which they integrate by accelerating the execution of required tasks. They also allow access to inaccessible areas by humans and explore them under severe conditions. In this context, many efforts have been made to improve their functionality, especially in detecting the various obstacles they may encounter, improving the trajectory following [1], [2] and especially their good decision-making [3]. Indeed, individual, and collective safety is a very important factor in the technologically open and so active world of today, especially with the emergence of the concept of the Internet of Things and autonomous vehicles. It requires the development of reliable and accurate applications in the detection of obstacles by autonomous vehicles to minimize the risk of accidents.

This is only possible with technological advances in digital and embedded electronics. Several development platforms have been designed to meet the hardware and software needs of such applications, especially since they require large-scale data processing.

For that purpose, reconfigurable architectures such as FPGAs and systems on chip (SoC) are increasingly used due to their high flexibility. This latter characterizes the high ability of functional blocks to be reconfigured statically or dynamically. As a result, these architectures are device-rich platforms suitable for the development of a wide variety of complex real-time applications [4]. They require a lot of computational and processing time and a large capacity of memory to analyze data. The reconfigurable aspect allows optimizing the application hardware architecture and reducing the latency. We find that some SoC include an FPGA circuit and a robust and very fast hardware processor (ARM). Besides, advanced software solutions play an important role in ensuring the high performance of these reconfigurable circuits. In this case, the co-design method is necessary to optimize the applications developed through suitable hardware and software partitioning [5].

The hardware solutions allow rapid prototyping of the envisaged application and ensure its autonomy. While software solutions allow optimization of using the hardware resources and perform the complex computing due to the algorithms and models used [6]. For example, a predictive control model for autonomous cars was developed to predict the path to be followed while considering varying road conditions and small-angle assumptions as measurable disturbances [7]. For the same objective of path tracking, the authors in [8] implemented a fuzzy logic controller into a Xilinx Microblaze soft processor core integrated into a SoC. The control was applied to an actual differential-drive Pioneer 3-DX8 robot. A real-time fuzzy controller for Omni-Mecanum-wheeled autonomous vehicles was developed to achieve an intelligent tracking control [9].

An FPGA embedded processor was used to execute the fuzzy computing and kinematics model-based control. An online learning system of a fuzzy controller was performed to achieve an accurate vehicle cruise control [10]. Moreover, an adaptive fuzzy-neural control of a rotational direction of robotic joints was developed using embedded silicone sensors [11]. The author in [12] developed a neural-fuzzy system applied to a flight vehicle and implemented it into a parallel reconfigurable computer. It is based on FPGA and nine DSP blocks to ensure fast prototyping, reusability, and high-speed run-time data processing [13]. The developed system includes five principal and parallel Mamdani fuzzy controller blocks, which ensure reaching the best flight guidance, navigation, and flight control condition. The flight vehicle is designed to carry out several risk tasks like rescue works, as well as the data collection used in environmental analysis, as also explained in [14].

The fuzzy logic control was also investigated to reach an intelligent vision by autonomous vehicles to park correctly through a braking process [15]-[17]. In these works, FPGA circuits were used to implement the proposed fuzzy logic controllers. The authors in [17] have especially proposed a fully customizable hardware architecture of two fuzzy logic controllers for autonomous cars. The first one was used to control the wheel's orientation and the second controlled the speed of the wheels. The flexibility of the proposed architecture allowed it to be used for any problem application. In [18], the authors developed a hardware design of a fuzzy position correction system. It acts on the latitude and longitude data provided by a GPS Pmod sensor. It is also implemented on a Terasic DE1-SoC development based on an Altera Cyclone V FPGA to perform the speed of autonomous cars. The processing time of the fuzzy system reached 21.5 µs. Furthermore, the authors in [19] proved the efficiency of an FPGA based fuzzy logic controller compared to a soft real-time platform NI CompactRIO to track a given position of a servo-pneumatic actuation system, which becomes more accurate and stable. Moreover, the authors in [20] developed a real-time fuzzy logic controller-based obstacle avoidance approach. It was applied to National Instruments' embedded robotic platform, including an FPGA circuit. The software design was carried out using LabVIEW modules. The fuzzy controller allowed the robot to have a more stable response to both smoother and sudden changes. The previous methodology has been followed to control the movement of a robot arm with six degrees of liberty [21]. Xilinx System Generator (XSG) tool was also used to model and implement fuzzy controllers into FPGA through the MATLAB environment [22], [23]. Indeed, a hardware implementation of a road lanes detection system for intelligent cars was performed on Xilinx Virtex 5 FPGA device using the XSG tool [24], [25]. The purpose is to improve the driver safety on the roads.

Within this framework, we have developed a hardware speed fuzzy logic controller of an autonomous vehicle and implemented it on a Xilinx Zynq board. The controller considers the presence of an obstacle and the value of the ambient temperature (if this is high, the speed must be decreased). In addition, the application ensured that the vehicle provided the necessary information on its status and the decisions that it made. The novel proposed hardware architecture is considered flexible thanks to the FPGA reconfigurable architecture and the fuzzy algorithm developed. Indeed, when the fuzzy input and output variables change,

we have only to modify their values in the algorithm (in the declaration section of signals). Then, the proposed architecture of the fuzzy controller is suitable for many other applications.

The paper is structured as follows: Section 2 gives the Simulink model of the speed fuzzy logic controller to fix the fuzzy rules. Section 3 depicts the method used to develop the hardware/software design of the speed fuzzy logic control. The simulation and implementation results are reported in section 4. The last section is devoted to concluding the paper.

2. MODELLING OF THE FUZZY LOGIC CONTROLLER USNG MATLAB/SIMULINK ENVIRONMENT

The fuzzy logic controller considers the speed error, the surrounding temperature, and the distance from an obstacle through a sonar sensor. The derivative speed error is also considered as an input for the controller in order to have information about the reaction speed of the system. The output signal is the speed expressed in (rpm). The inputs and outputs of the controller, the fuzzy rules and the computation method of the control signal applied to the car are subsequently performed.

2.1. Principle of the fuzzy logic control

The fuzzy logic control consists of three major modules: the fuzzifier, the inference, which is driven by the fuzzy rules, and the defuzzifier, as shown in Figure 1.

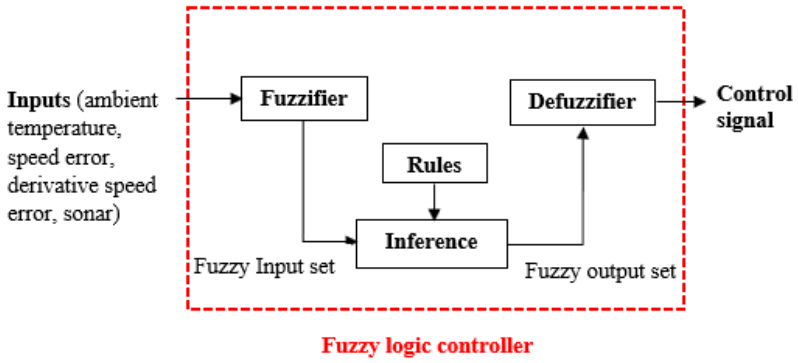

Figure 1. Architecture of the fuzzy logic controller

The fuzzifier encodes the crisp measured values of each system parameter (ambient temperature, speed and its derivative errors, and sonar) into a fuzzy term using the given membership functions. The inference module allows identifying the subset of fuzzy rules that can be fired and draw adequate fuzzy consequences [17]. Herein, we have used the Mamdani Min-Max process. Finally, the defuzzifier decodes the fuzzy variable (speed control signal) and computes the corresponding crisp value using the center of gravity method. These modules are described next.

2.2. Model of the speed fuzzy controller

2.2.1. Fuzzifier

For all fuzzy input variables, as well as the fuzzy output, we have chosen triangular membership functions, which are illustrated in Figure 2. Furthermore, the linguistic terms and the corresponding universes of the discourse of each input and output variable are summarized in Table 1.

For triangular membership functions, the determination of the truth degree μ of each linguistic term is carried out using (1) or (2) according to having an increasing or decreasing slope, respectively, as shown in Figure 3.

$$\mu = \frac{(input\_value - p1)}{max\_pt - p1} \quad (1)$$

$$\mu = 1 - \frac{(input\_value - max\_pt)}{p2 - max\_pt} \quad (2)$$

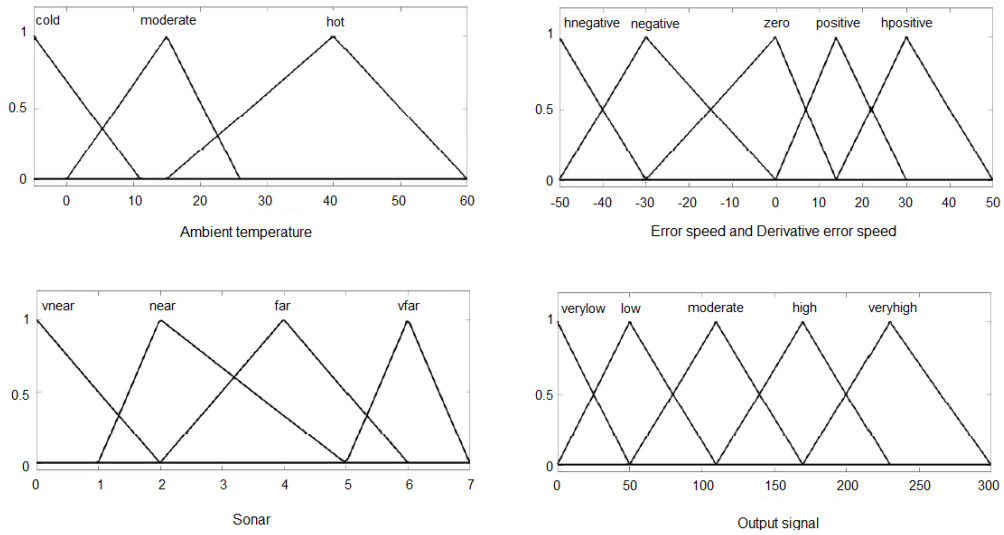

Figure 2. Membership functions of the fuzzy input and output variables

Table 1. Universes of the discourse of input and output variables

	Fuzzy input and output variables	Universe of the discourse
Ambient temperature (°C)	cold	xcold ϵ [-5 11]
	moderate	xmoderate ϵ [0 26]
	hot	xhot ϵ [15 60]
Sonar (m)	vnear (very near)	xvnear ϵ [0 2]
	near	xnear ϵ [1 5]
	far	xfar ϵ [2 6]
	vfar (very far)	xvfar ϵ [5 7]
Speed error/derivative speed error (cm/s)	hneg (high negative)	xhneg ϵ [-50 -30]
	neg (negative)	xneg ϵ [-50 0]
	zero	xzero ϵ [-30 14]
	pos (positive)	xpos ϵ [0 30]
	hpos (high positive)	xhpos ϵ [14 50]
Control signal (rpm)	verylow	xverylow ϵ [0 50]
	low	xlow ϵ [0 110]
	moderate	xmoderate ϵ [50 170]
	high	xhigh ϵ [110 230]
	veryhigh	xveryhigh ϵ [170 300]

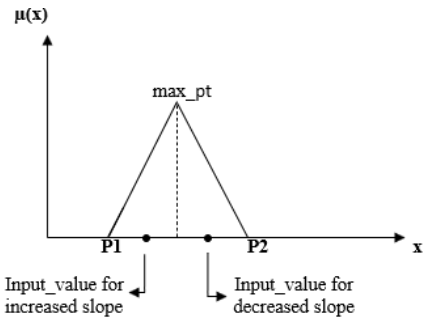

Figure 3. Triangular type membership function

2.2.2. Fuzzy rules

To thoroughly control the behavior of the car, we have defined 44 rules that mainly consist of: (i) If the ambient temperature is hot, then the speed is very low. (ii) If the sonar is very near or near, then the speed is very low or low, respectively. For a cold or moderate temperature, the rules are given in Table 2.

Table 2. Fuzzy rules when the ambient temperature is cold or moderate

		Speed error				
		hneg	neg	zero	pos	hpos
Derivative speed error	hneg	verylow	verylow	verylow	low	moderate
	neg	low	low	low	high	veryhigh
	zero	verylow	(very)low	moderate	high	veryhigh
	Pos	verylow	low	moderate	very(high)	veryhigh
	hpos	verylow	low	low	low	veryhigh
	hneg	verylow	verylow	verylow	low	moderate

It is worth noting that when the error speed is negative and the derivative one is zero, the speed is low or very low when the obstacle (referred by the sonar signal) is far or very far, respectively. However, when both the error speed and the derivative one is positive, the speed is high or very high according to the distance from the obstacle respectively far or very far.

The use of the 'and' operator means that the truth degree μ of each output linguistic term (verylow, low, moderate, high, veryhigh) takes the lowest truth degree of the input linguistic terms according to the verified one or many rules.

2.2.3. Defuzzifier

The crisp value of the control signal xspeed is determined using the center of gravity method according to (3).

$$x_{\text{speed}} = [x_{\text{verylow}}\mu(x_{\text{verylow}}) + x_{\text{low}}\mu(x_{\text{low}}) + x_{\text{moderate}}\mu(x_{\text{moderate}}) + x_{\text{high}}\mu(x_{\text{high}}) + x_{\text{veryhigh}}\mu(x_{\text{veryhigh}})]/[\mu(x_{\text{verylow}}).\mu(x_{\text{low}}).\mu(x_{\text{moderate}}).\mu(x_{\text{high}}).\mu(x_{\text{veryhigh}})] \quad (3)$$

where μ is the truth degree of each fuzzy output variables (verylow, low, moderate, high, veryhigh) corresponding to the values (x_{verylow}, x_{low}, x_{moderate}, x_{high}, x_{veryhigh}).

2.3. System model

Before carrying out the hardware/software design of the proposed fuzzy controller, we have tested it by applying a car plant [26] according to our car dimensions, shown in Figure 5. It is a discrete transfer function given in (4). The sample time is equal to 0.2 s.

$$H(z) = \frac{\Delta v}{\Delta u} = \frac{0.0007198z^2 + 0.00217z + 0.0003955}{z^3 - 2.18z + 1.487z - 0.3009} \quad (4)$$

where Δu is the control signal that is considered as throttle control and used to increase or decrease the engine driver force [26] and Δv is the car speed in cm/s. Figure 4 presents the system model developed in MATLAB/Simulink environment.

The saturation block limits the control signal value to 300 and the second one allows a maximum speed value of 90 cm/s.

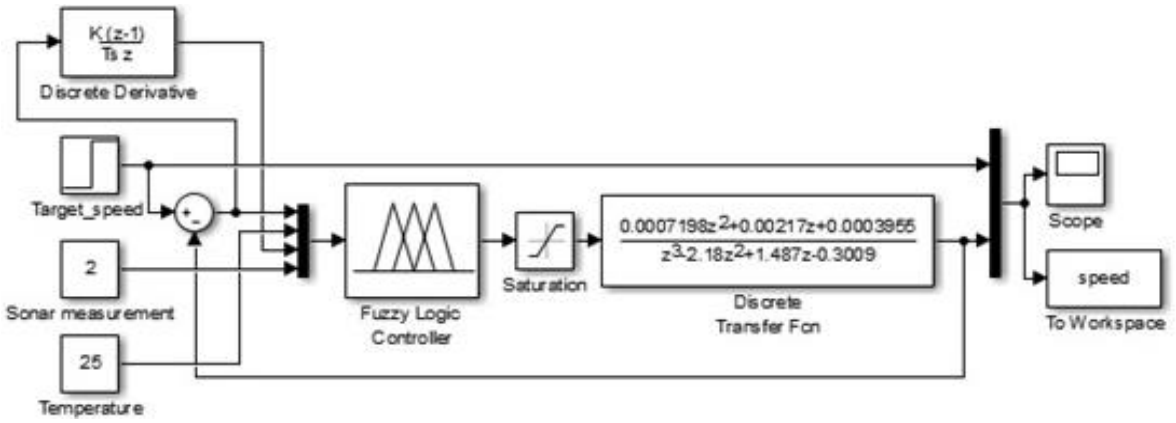

Figure 4. Simulink model of the car speed fuzzy control

3. HARDWARE/SOFTWARE CO-DESIGN OF THE FUZZY LOGIC CONTROLLER

In this section, we depict a synthesizable hardware and software co-design of the proposed fuzzy controller dedicated to being applied to the autonomous car, which is presented in Figure 5. Herein, the control signal is considered as a speed expressed in rpm, as mentioned earlier. The design generates three PWM signals that will drive the car left and right wheels, as well as the servo motor that controls the front steering. However, the steering control is already achieved [27] and herein used.

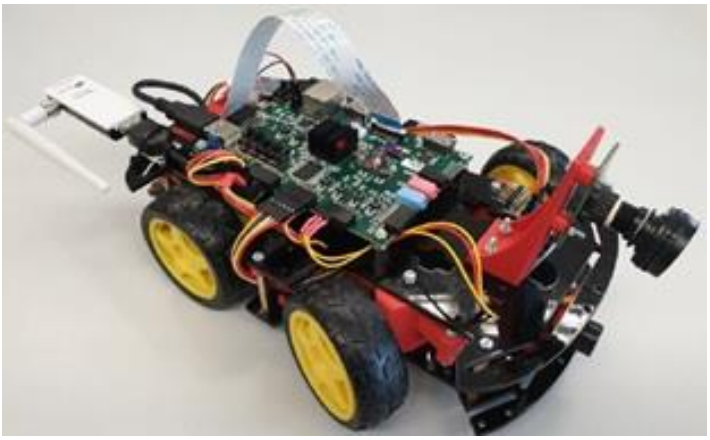

Figure 5. The autonomous car [27]

The partitioning of the application tasks consists of dividing them into a hardware part and a software part. In the hardware part, we developed the hardware architecture of the fuzzy controller IP and used the IPs of the temperature and sonar sensors. However, the software part uses the Zynq processor to accede the registers of sensors to read the distance and ambient temperature values. It also sends them to the fuzzy controller, which computes the car speed according to the control signal and generates the corresponding PWM signals.

3.1. Hardware design

The hardware design was developed using VHDL and some specifications were made to efficiently program the fuzzy algorithm.

3.1.1. Specifications of the programming process

To get a synthesizable code, we adopt the hexadecimal format to present the data values. Thus, the signals have the type std_logic_vector where their width is chosen as follows:

- 1 byte for signal positive values.
- 2 bytes for signal, which can have negative values.
- 1 byte for the output signal. Therefore, the truth degrees are between 0 and 255.

In addition, we use the min-max method to compute the signals of the fuzzy output linguistic terms (verylow, low, etc.), as well as the center of gravity to compute the control signal. This latter is used to determine the PWM signal that drives the car and then its corresponding duty cycle. Indeed, each output linguistic term has a specific duty cycle width, as reported in Table 3. The system clock frequency and those of the PWM signal are fixed to 50 MHz and 100 kHz, respectively. Therefore, a counter is used to count up to 500 to reach a PWM period.

3.1.2. Hardware architecture of the fuzzy controller

Here, the objective is to prepare an IP that describes the operation of the fuzzy controller. This was accomplished by writing the fuzzy controller's VHDL code based on the Mamdani method. The entity block of the fuzzy controller, as well as its architecture, are shown in Figure 6. It is worth noting that each block consists of a separate VHDL module, and all of them are interconnected using the 'port map' instruction.

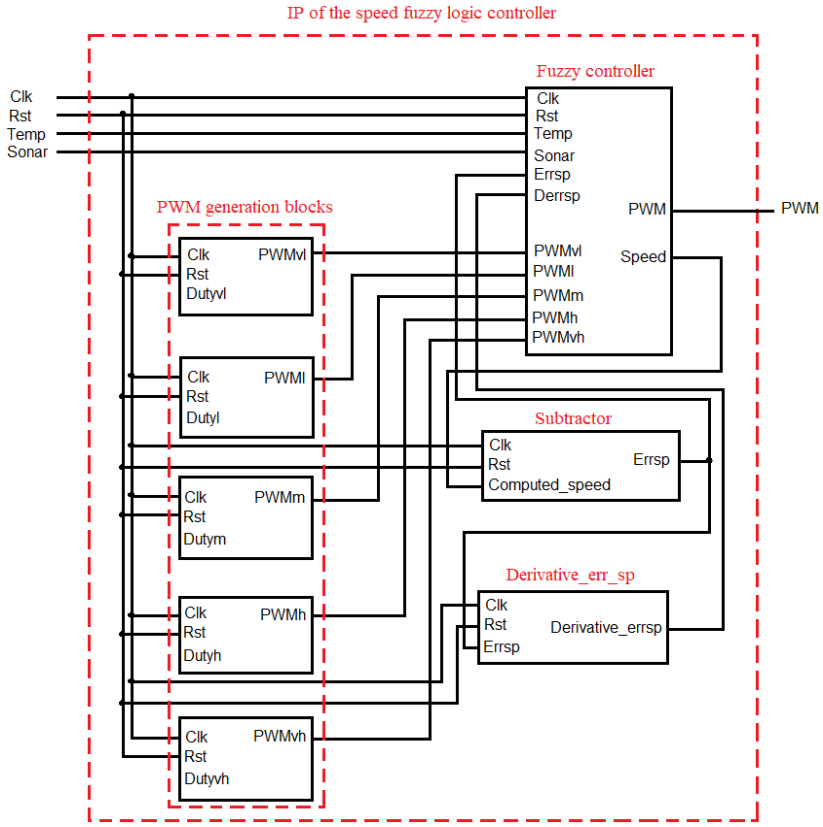

Figure 6. Hardware architecture of the speed fuzzy controller IP

The speed error value (signal *Errsp*) is obtained using a subtractor, which receives the computed speed and includes the target speed as an internal signal. Furthermore, the derivative speed error value (signal *Derrsp*) is considered as the difference between the current speed error value and the previous one using a finite state machine. It is presented in Figure 7.

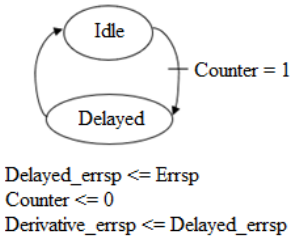

Figure 7. Finite state machine that computes the derivative error speed

However, the signals *Temp* and *Sonar* refer to the measured values of the ambient temperature and the distance from the obstacle, respectively, using sensors. These values are used to compute the control signal, and then the car speed value using (5).

$$Speed[m/s] = \frac{(2.\pi.r)}{60}.control\_signal[rpm] \quad (5)$$

where *r* is the wheel ray and is equal to 3 cm.
It is worth noting that the car speed can maximum reach 90 cm/s for a control value of 300 rpm.

As shown in Table 3, each PWM generation block generates a different PWM corresponding to one of the output linguistic terms to which the output signal 'PWM' of the fuzzy controller is assigned. For instance, this signal is assigned to PWMl when the speed should be low.

Table 3. PWM signal of each output linguistic term.

Output linguistic terms	Duty signal	Number of cycles	PWM signals
verylow	Dutyvl	20 → 4 %	PWMvl
low	Dutyl	125 → 25 %	PWMl
moderate	Dutym	250 → 50 %	PWMm
high	Dutyh	375 → 75 %	PWMh
veryhigh	Dutyvh	480 → 96 %	PWMvh

Each Duty signal (*Dutyvl*, *Dutyl*, *Dutym*, *Dutyh*, and *Dutyvh*) indicates the number of cycles for which the corresponding PWM signal becomes high. It is worth noting that the PWM period corresponds to 500 cycles. We assume that the maximum speed rate corresponds to only 96 % for security matters.

The algorithm developed in the 'Fuzzy controller' block consists of the following successive steps:

– Defining the values of the linguistic terms in hexadecimal format.

For example, the first two values of the sonar sensor are defined as follows:

```
constant sonp1:STD_LOGIC_VECTOR (7 downto 0):=x"00";--0
constant sonp2:STD_LOGIC_VECTOR (7 downto 0):=x"01";--1
```

Also, for the temperature sensor, the first two values are:

```
constant tp1:STD_LOGIC_VECTOR (15 downto 0):=x"FFFB";--(-5)
constant tp2:STD_LOGIC_VECTOR (15 downto 0):=x"0000";--0
```

– Building the triangular membership function of each input linguistic term using equations of rising and falling slopes.

For example, the VHDL code of computing the membership degree 'mvnear' of the linguistic term 'vnear' using the falling slope equation is:

```
Mvnear<=conv_std_logic_vector(conv_integer(sonar-sonp1)/conv_integer(sonp3-
sonp1),8);
```

Likewise, the VHDL code of computing the membership degree 'mnear' of the linguistic term 'near' using the rising slope equation is:

```
mnear<=conv_std_logic_vector(conv_integer(sonar-sonp2)/conv_integer(sonp3-sonp2),8);
```

Determination of the truth degree of each input linguistic term according to its membership function.

For instance, the truth degree 'dvnear' of the linguistic term 'vnear' is determined using the following VHDL code:

```
if rising_edge(clk)then
   ---mvnear
      if sonar > sonp3 then -- sonp3 = 2 m
         dvnear <= x"00";
      else dvnear <= x"FF"-mvnear; -- falling slope
      end if;
end if;
```

– Elaboration of the fuzzy rules using if-else statements. Each rule result takes the minimum input truth degree.

For example, the 18th rule 'r18' describes the case: when the distance from the obstacle is between 1 and 5 m, its output takes the value of the truth degree 'dnear'. The corresponding VHDL code is:

```
if sonar >= sonp2 and sonar < sonp4 then r18 <= "00"& dnear; -- sonp4 = 5 m
else r18<=(others=>'0');
end if;
```

In this case, the fuzzy output of the controller should be assigned to the linguistic term 'verylow'.

– Determination of the output truth degrees also using if-else statements. Each output truth degree takes the maximum result of the fuzzy rule used to determine it.

For example, the truth degree ‘dcmoderate’ of the output linguistic term ‘moderate’ is computed according to the fuzzy rules (r9, r10, r43, and r44). Each of them has an output assigned to this linguistic term:

```
if rising_edge(clk)then
if r10 < r9 and r43 < r9 and r44 < r9 then dcmoderate <= r9;
elsif r9 < r10 and r43 < r10 and r44 < r10 then dcmoderate<=r10;
elsif r9 < r43 and r10 < r43 and r44 < r43 then dcmoderate<= r43;
elsif r9 < r44 and r10 < r44 and r43 < r44 then dcmoderate<= r44;
else dcmoderate <= (others=>'0');
end if;
end if;
```

- Determination of the control signal using equation (3).
- Computation of the speed value using equation (5).
- Determination of the PWM signal according to the obtained control.

3.1.3. Complete hardware design of the speed fuzzy control

The design of the speed controller was performed using the Xilinx Vivado Design Suite 2018. It includes the IP of the fuzzy controller (myipFLC_0), the Zynq processor, and the sensors, as shown in Figure 8. The fuzzy controller starts operating when the enable signal is high. İt is an asynchronous reset active low (enable <= not rst).

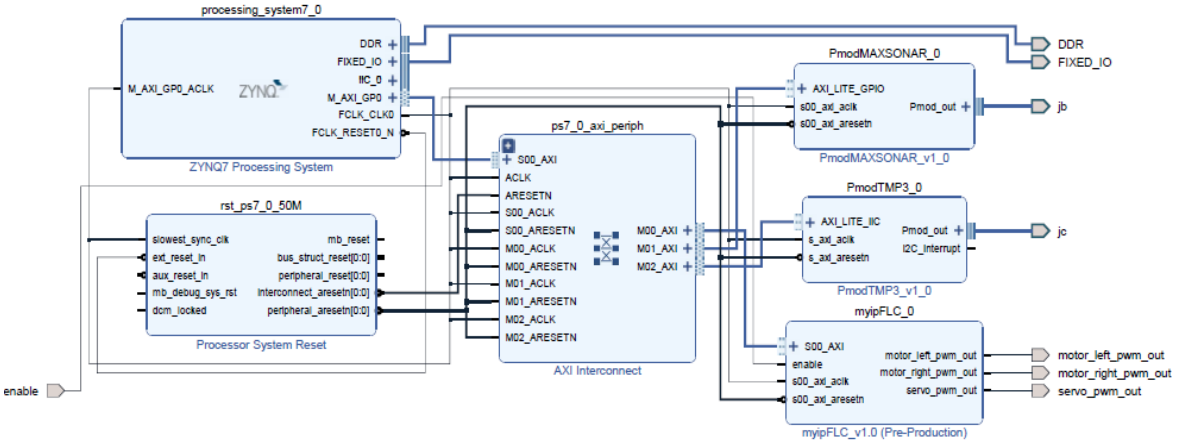

Figure 8. Hardware design of the speed fuzzy control using Vivado design suite

The ambient temperature and the distance from the sonar sensor are computed and read by the Zynq processor at the level of its 32-bit data registers through C files. The development of these files was performed in the software section. Then, they are sent to the fuzzy controller through the AXI bus to generate the suitable PWM signal that is determined by the ‘Fuzzy controller’ block presented in Figure 6. This signal is driven to each of the output PWM signals *motor_left_pwm_out*, *servo_pwm_out*, and *motor_right_pwm_out*, which drive the car movement. Herein, the Vivado design suite assigns a 32-bit range of addresses to registers of each IP. Therefore, the Zynq processor can act on the fuzzy controller and the sensors by accessing their registers through the AXI bus. For the ZedBoard, four slave registers are defined for the fuzzy IP.

3.2. Software design

The software design consists of developing the C files. They allow reading the temperature value, as well as, calculating the distance from the obstacle according to the PWM signal generated by the sonar sensor. Indeed, a duty width of 147 us corresponds to 2.54 cm. No obstacle is detected if the distance exceeds 6.5 m. We also added various messages that the vehicle must generate to inform us about the state of its operation and the decisions that were made, as explained in Figure 9.

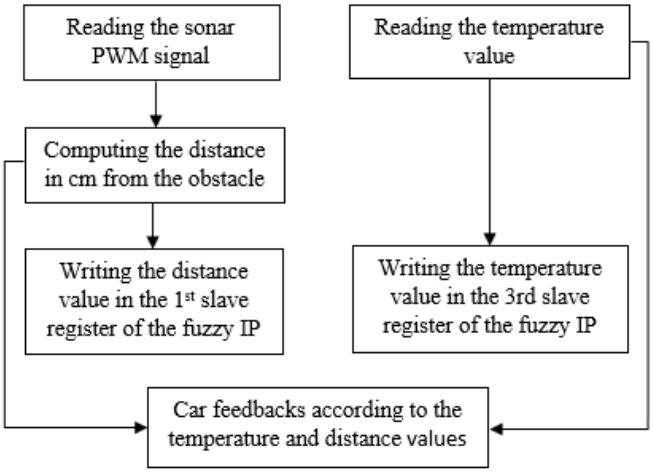

Figure 9. Software design tasks to send data to the fuzzy controller through the Zynq processor and to get feedback from the car

This step is performed using the Xilinx SDK tool after exporting the hardware and the bitstream file (.bit). After that, the building of the SDK project allows the generation of the executable file (.elf). This latter allows applying the software application to the exported hardware after loading the corresponding bitstream into the FPGA circuit to configure it.

4. RESULTS AND DISCUSSION

4.1. Simulation results of the Simulink model

The simulation was carried out for a target speed variation from 30 to 40 cm/s, an ambient temperature equal to 25 °C, and an obstacle distance of 6 m. In this case, the car reaches the target speed, as shown in Figure 10.

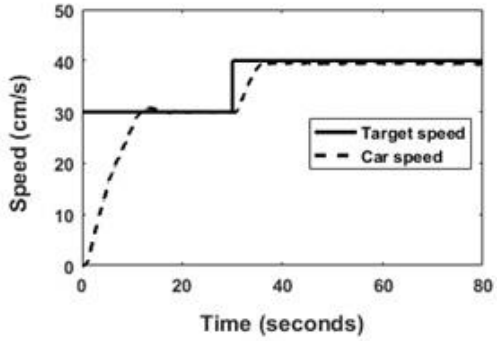

Figure 10. Car speed behavior when the ambient temperature is moderate, and the obstacle is far

When this distance decreases by 1 m and then by 4 m after 30 s, the car speed also decreases from 40 to 27 and 10 cm/s, respectively, as shown in Figure 11(a) and Figure 11(b). In cases where the temperature becomes hot (40 °C), the speed decreases to 8.5 cm/s and no more follow-up of the target speed, as shown in Figure 11(c). The same behavior is reached when, at the beginning, the obstacle distance is 2 m near or very near (1 m) from the car. However, the speed starts by increasing at the beginning seen that the ambient temperature is moderate. Then, the speed was immediately decreased due to the very near distance between the obstacle and the car. This scenario is illustrated in Figure 11(d). Consequently, the fuzzy controller's operation complies with the proposed specifications.

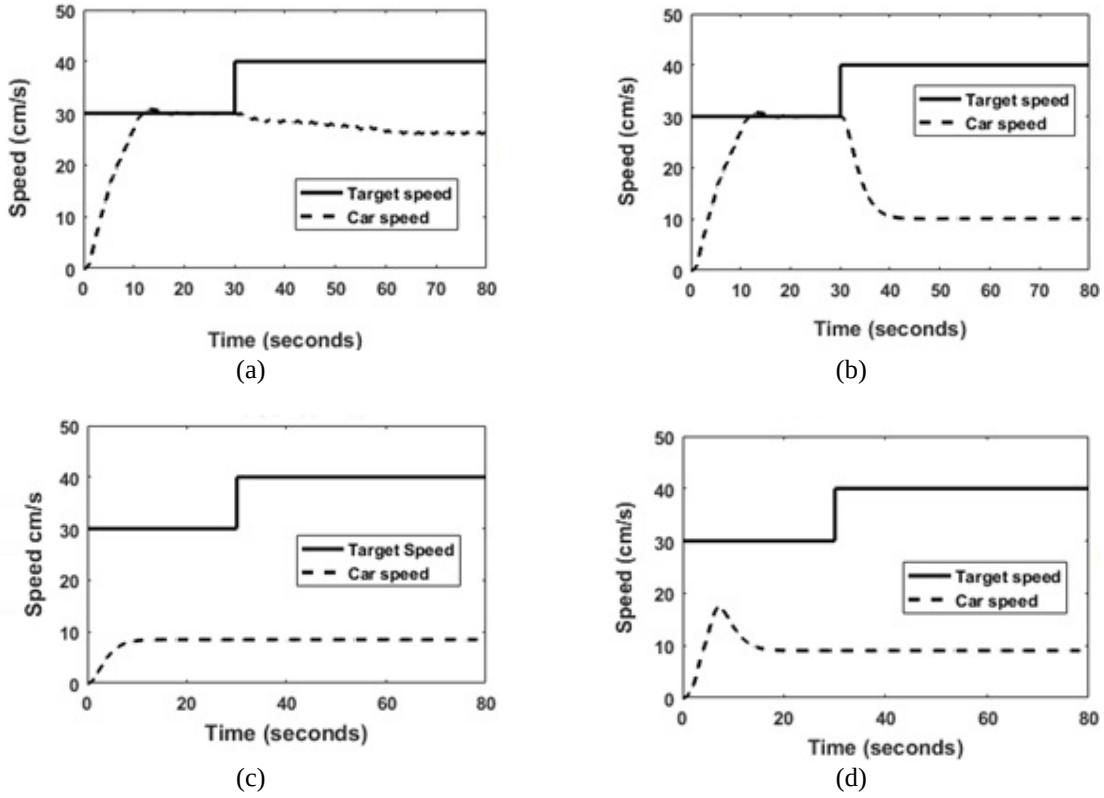

Figure 11. Car speed behavior when the obstacle distance decreases after 30 s by (a) 1 m and (b) 4 m, (c)the ambient temperature becomes hot, and (d) the obstacle is very near the car during the simulation

4.2. Simulation results of the hardware design

The fuzzy logic algorithm developed in VHDL was tested for several scenarios during 20 us, which correspond to two complete cycles (or control periods). The scenarios prove to achieve the desired behavior. Indeed, Figure 12 illustrates the case of a target speed equal to 40 cm/s, an ambient temperature equal to 25 °C, and an obstacle distance of 6 m from the sonar sensor. Hence, only the moderate truth degree is active, as well as those of the far/veryfar fuzzy variables of the sonar. Consequently, no constraints are imposed on the car, which should accelerate to reach the target speed thanks to the high state of the generated PWM signal.

As shown in Figure 12, the speed control is high. It varies between 110 and 170 rpm according to the values of the input fuzzy variables. Therefore, only the output truth degrees of the fuzzy terms moderate, far, and veryfar are active. Furthermore, the speed average value is about 42 cm/s, which meets with the target speed. Indeed, the error speed and the derivative one is the output truth degrees of the fuzzy terms moderate, far, and veryfar are both low. Nonetheless, Figure 13 illustrates the cases for which the car should brake or stop to avoid the collision with the obstacle. Indeed, the scenario I shows that the control signal decreased to 2 rpm corresponding to the output fuzzy set verylow. It is due to the hot ambient temperature (the truth degree of the hot variable is active) despite that the obstacle is far. Thus, the PWM signal is low, and then the speed decreased considerably, and the car stops. Moreover, scenario II shows that for a moderate temperature, the control signal decreases as long as the obstacle is closer to the sensor. It is also considered low/verylow. Then, the PWM signal is inhibited, and the car speed drops in order to avoid the collision with the obstacle.

As for the results achieved by the fuzzy model developed on MATLAB, the operation of the hardware speed fuzzy controller also satisfies the required specifications.

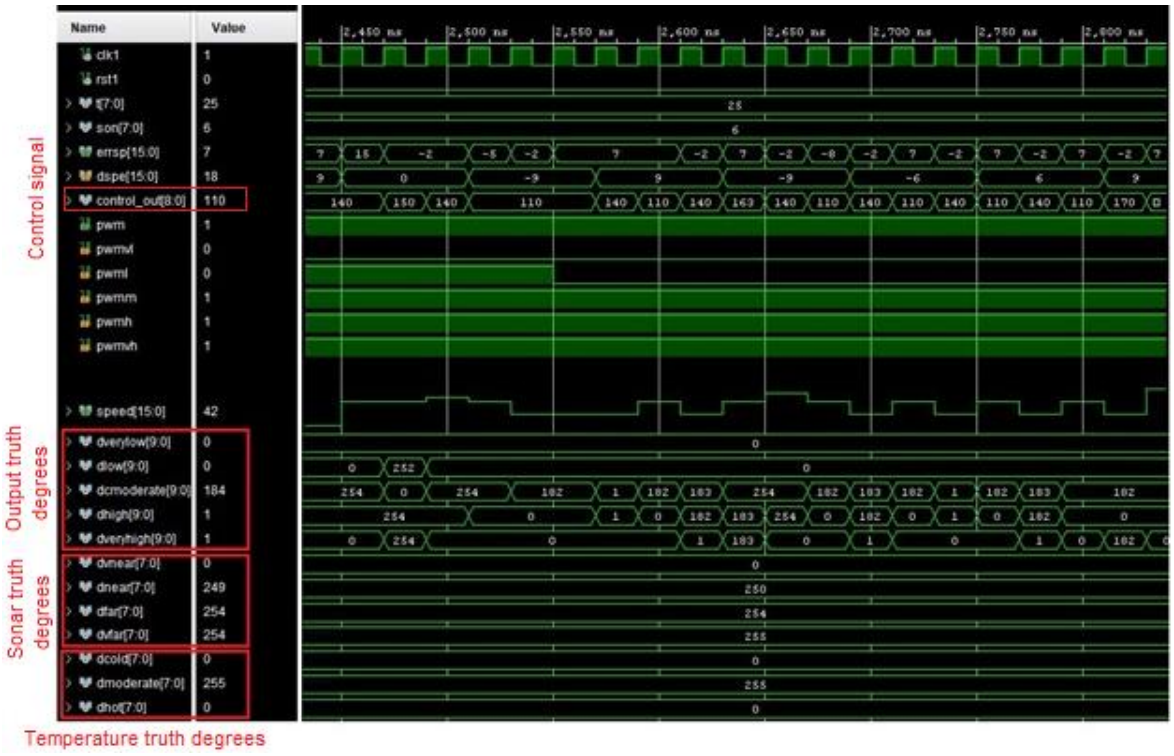

Figure 12. Simulation results of the speed fuzzy control when the ambient temperature is moderate, and the obstacle is very far

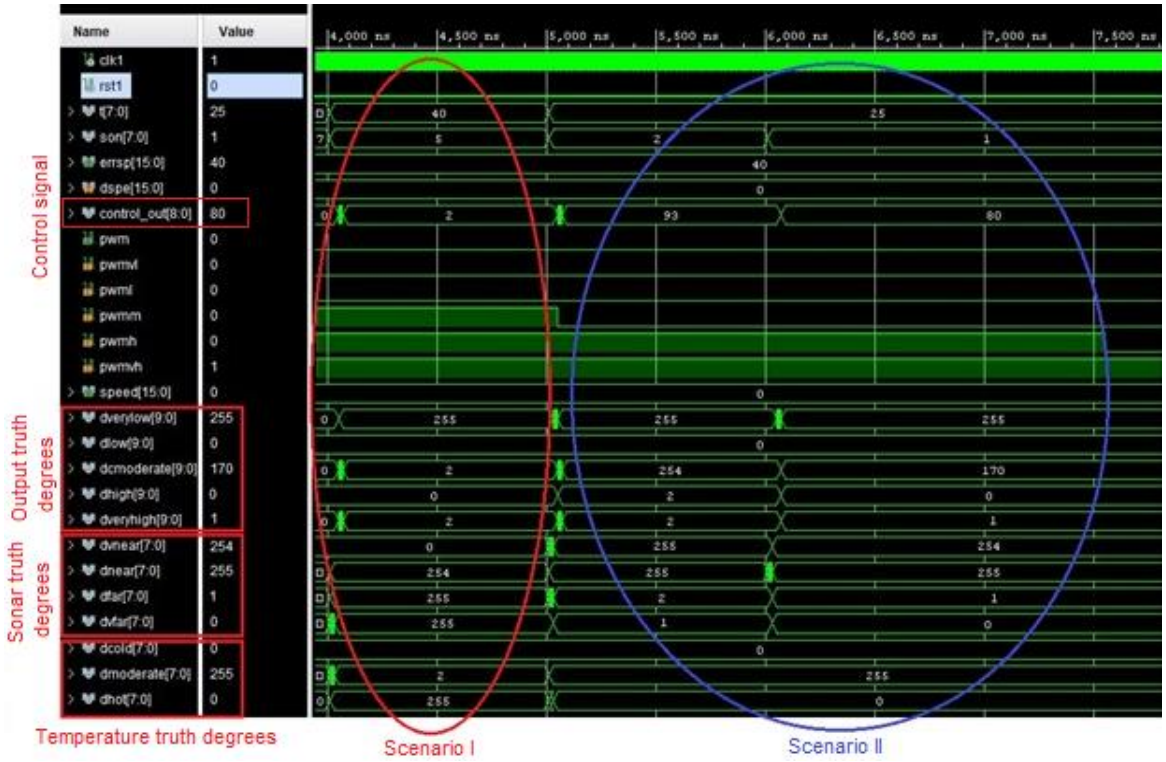

Figure 13. Simulation results of the speed fuzzy control when constraints on temperature and obstacle are imposed

4.3. Implementation results

The implementation of the proposed co-design was carried out on the ZedBoard 7z020-clg484 while choosing the clock frequency 50 MHz. After generating and verifying the schematic design of the hardware design mentioned in Figure 8, we executed the synthesis and implementation processes, which allow placing the hardware architecture in the ZedBoard one, as shown in Figure 14. The Zynq processor I/O (here FIXED_IO) is routed to the multiplexed I/O (MIO)).

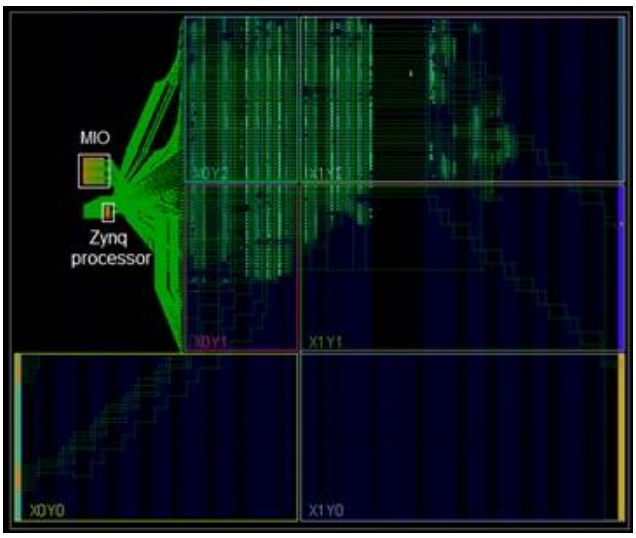

Figure 14. Implementation design of the speed fuzzy control

Moreover, Table 4 summarizes the postimplementation results of the hardware design that fits into the selected SoC's logic, leaving enough resources available for further design applications or improvements. As shown in Table 4, DSP blocks were used to perform the calculation operations of the membership functions, the truth degrees of fuzzy inputs and outputs, the crisp output, and the car speed. They allow reducing the calculation time thanks to parallel processing. An output value is computed at each clock cycle. However, as soon as the sensors' measurements are available, it is necessary to run the application for at least 7 us to complete the execution of the PWM blocks used in the IP of the fuzzy controller.

Table 4. Hardware resources used by the design

Resource	Utilization	Available	Used (%)
6 -LUT	9789	53200	18.4
LUTRAM	72	17400	0.41
FF	2757	106400	2.59
DSP	14	220	6.36
IO	20	200	10
BUFG	1	32	3.13

We also note that the proposed design does not require a lot of hardware resources compared to previous similar hardware fuzzy logic designs, such as in [17]. For instance, the utilization of the 6-LUTs is reduced 11 times compared to those used in this reference. This is achieved thanks to the optimization of the fuzzy algorithm and the high performance of the FPGA circuit. The power consumption of the chip is about 1.68 W of which 96 % is consumed by the Zynq processor.

Moreover, the reconfiguration of the FPGA is achieved by generating the bitstream file, which was exported with the hardware design to the SDK environment, where we developed the software design and then built it to get the executable file (.elf). Indeed, Figure 15 and Figure 16 show the messages sent by the car using the UART peripheral according to values of sonar and ambient temperature sensors. These values are sent to the fuzzy controller through the slave registers 1 and 3, respectively, in hexadecimal format.

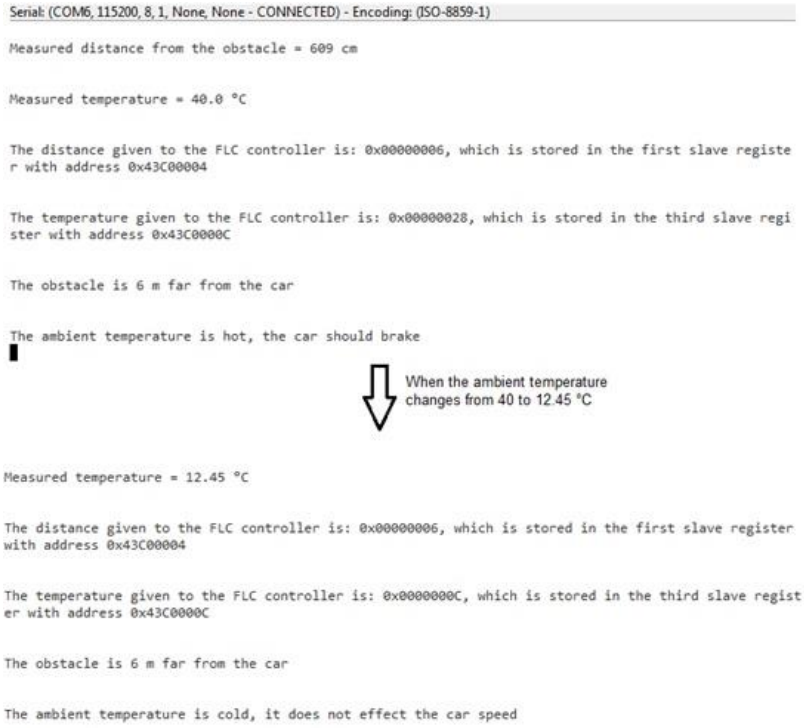

Figure 15. Results of the software design displayed on the serial terminal (COM6) when the ambient temperature changes from 40 to 12.45 °C and the obstacle distance is about 6 m

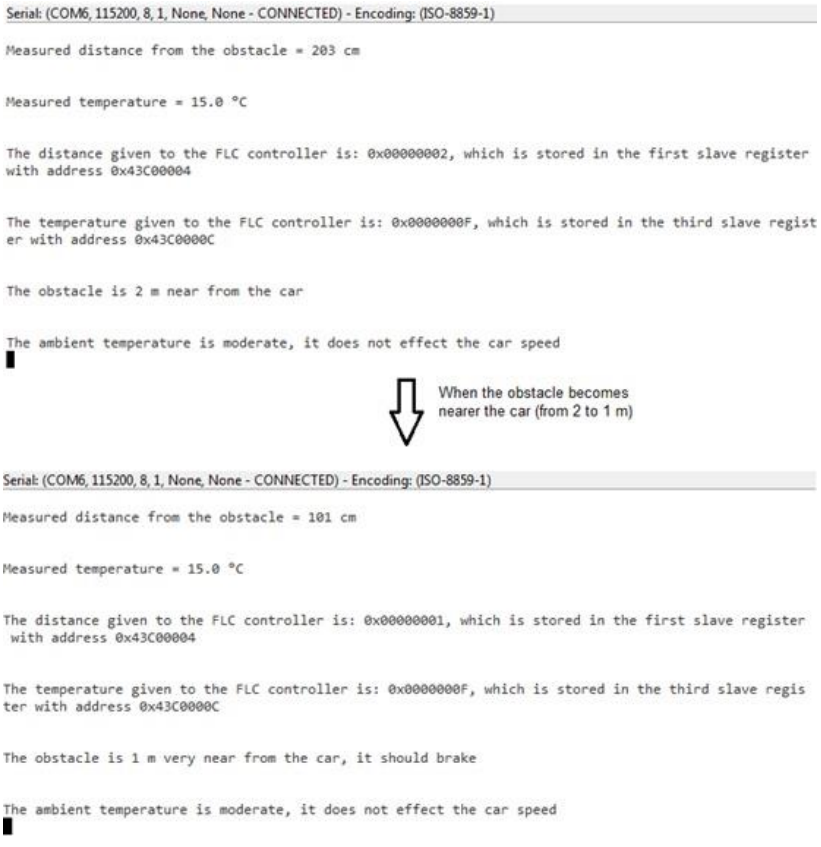

Figure 16. Results of the software design displayed on the serial terminal (COM6) when the obstacle distance is reduced from 2 to 1 m and the ambient temperature is 15 °C

Furthermore, Figure 17 illustrates the ZedBoard configuration when no constraints are imposed on the car (the ambient temperature and the obstacle distance are equal to 25 °C and 6 m, respectively). In fact, the enable switch is enabled, then, the PWM signals are active seen that the car does not encounter constraints.

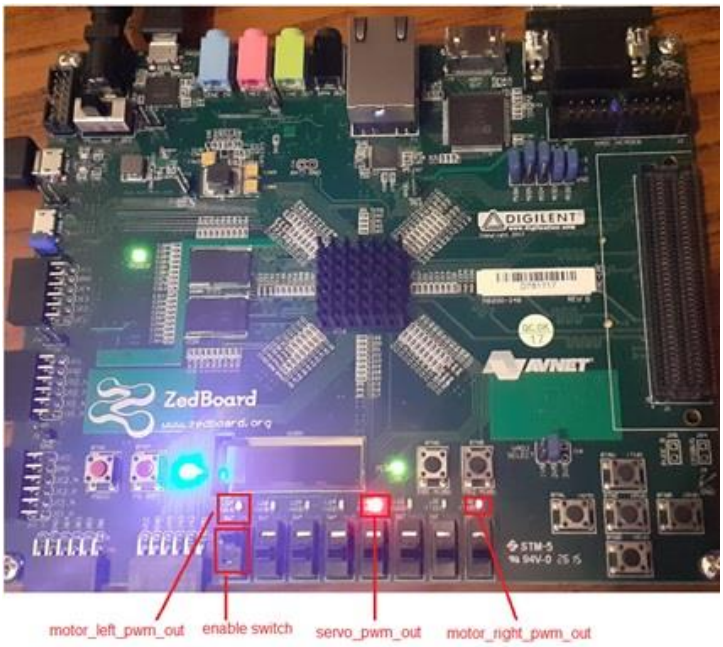

Figure 17. ZedBoard configuration and result obtained

5. CONCLUSION

In the present paper, we have developed a hardware/software co-design to control the speed of an autonomous car using FPGA technology and a fuzzy logic controller. This latter is performed by considering the variation of the ambient temperature and the distance from a given obstacle. It makes the car brake or stop for hot temperature or a near obstacle. Otherwise, the controller ensures reaching the target speed. The simulation and implementation results showed that these specifications were achieved. In addition, the novel hardware/software solution allowed accelerating the fuzzy computing. It is due to the optimized fuzzy algorithm developed and the availability of DSP blocks on the ZedBoard that perform the parallel processing. Hence, a complete execution cycle takes only 7 us. The proposed solution is also considered flexible thanks to the Zynq processor. This latter can be configured without modifying the hardware architecture. Furthermore, the utilization rate of hardware resources is considered lower than other previous hardware solutions. In future work, we aim to build an embedded Linux solution to detect and recognize a given obstacle that encounters the car.

REFERENCES

[1] J. P. Fernández, M. A. Vargas, J. M. V. García, J. A. C. Carrillo and J. J. C. Aguilar, "Low-cost FPGA-based electronic control unit for vehicle control systems," *Sensors*, vol. 19, no. 8, p. 1834, 2019, doi: 10.3390/s19081834.

[2] V. Kapse, B. Jharia and S. S. Thakur, "FPGA based fuzzy control technique for obstacle avoidance," *TENCON 2017 - 2017 IEEE Region 10 Conference*, 2017, pp. 1245-1250, doi: 10.1109/TENCON.2017.8228048.

[3] O. Mata-Carballeira, J. Gutiérrez-Zaballa, I. del Campo and Victoria Martínez, "An FPGA-based neuro-fuzzy sensor for personalized driving assistance," *Sensors*, vol. 19, no. 18, p. 4011, 2019, doi: 10.3390/s19184011.

[4] W. Andre dan O. Couillard, "Design and implementation of a new architecture of a real-time reconfigurable digital modulator (DM) into QPSK, 8-PSK, and 16-PSK on FPGA," *International Journal of Reconfigurable and Embedded Systems*, vol. 7, no. 3, pp. 173-185, 2018, doi: 10.11591/ijres.v7.i3.pp173-185.

[5] N. Hou, X. Yan and F. He, "A survey on partitioning models, solution algorithms and algorithm parallelization for hardware/software co-design," *Design Automation for Embedded Systems*, vol. 23, pp.57–77, 2019, doi: 10.1007/s10617-019-09220-7.

[6] B. El Kari, H. Ayad, A. El Kari, M. Mjahed and C. Pozna, "Design and FPGA implementation of a new intelligent behaviors fusion for mobile robot using fuzzy logic," *International Review of Automatic Control*, vol. 12, no. 1, pp. 1-10, 2019, doi: 10.15866/ireaco.v12i1.14802.
[7] H. Guo, D. Cao, H. Chen, Z. Sun and Y. Hu, "Model predictive path following control for autonomous cars considering a measurable disturbance: Implementation, testing, and verification," *Mechanical Systems and Signal Processing*, vol. 118, pp.41–60, 2019, doi: 10.1016/j.ymssp.2018.08.028.
[8] S.G. Tzafestas, K.M. Deliparaschos and G.P. Moustris, "Fuzzy logic path tracking control for autonomous non-holonomic mobile robots: Design of system on a Chip," *Robotics and Autonomous Systems*, vol. 58, no. 8, pp.1017-1027, 2010, doi: 10.1016/j.robot.2010.03.014.
[9] H. C. Huang, C.W, Tao, C. C. Chuang and J. J. Xu, "FPGA-based mechatronic design and real-time fuzzy control with computational intelligence optimization for omni-mecanum-wheeled autonomous vehicles," *Electronics*, vol. 8, no. 11, p. 1328, 2019, doi: 10.3390/electronics8111328.
[10] E. Onieva, J. Godoy, J. Villagrá, V. Milanés and J. Pérez, "On-line learning of a fuzzy controller for a precise vehicle cruise control system," *Expert Systems with Applications*, vol. 40, no. 4, pp. 1046–1053, 2013, doi: 10.1016/j.eswa.2012.08.036.
[11] D. Petkovic, M. Issa, N. D. Pavlović and L. Zentner, "Intelligent rotational direction control of passive robotic joint with embedded sensors," *Expert Systems with Applications*, vol. 40, no. 4, pp. 1265–1273, 2013, doi: 10.1016/j.eswa.2012.08.064.
[12] G. M. A. Cordoba, "Autonomous intelligent fuzzy logic guidance, and flight control system for the EFIGENIA EJ-1B MOZART unmanned aerial vehicle UAV," *IFAC Proceeding*, vol. 40, no. 7, pp. 31–36, 2007, doi: 10.3182/20070625-5-FR-2916.00007.
[13] G. T. Tchendjou, E. Simeu and R. Alhakim, "Fuzzy logic based objective image quality assessment with FPGA implementation," *Journal of Systems Architecture*, vol. 82, pp. 24-36, 2018, doi: 10.1016/j.sysarc.2017.12.002.
[14] M. S. Gharajeh, "Implementation of an autonomous intelligent mobile robot for climate purposes," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 31, no. 3, pp. 200-218, 2019, doi: 10.1504/ijahuc.2019.100737.
[15] C. Y. Chen and H. M. Feng, "Hybrid intelligent vision-based car-like vehicle backing systems design," *Expert Systems with Applications*, vol. 36, no. 4, pp. 7500–7509, 2009, doi: 10.1016/j.eswa.2008.09.057.
[16] T. -. S. Li, Shih-Jie Chang and Yi-Xiang Chen, "Implementation of human-like driving skills by autonomous fuzzy behavior control on an FPGA-based car-like mobile robot," in *IEEE Transactions on Industrial Electronics*, vol. 50, no. 5, pp. 867-880, Oct. 2003, doi: 10.1109/TIE.2003.817490.
[17] N. Nedjah, P. R. S.S. Sandres and L. de Macedo Mourelle, "customizable hardware design of fuzzy controllers applied to autonomous car driving," *Expert Systems with Applications*, vol. 41, no. 16, pp.7046-7060, 2014, doi: 10.1016/j.eswa.2014.05.032.
[18] P. J. Correa-Caicedo, A. I. Barranco-Gutiérrez, E. I. Guerra-Hernandez, P. Batres-Mendoza, J. A. Padilla-Medina and H. Rostro-González, "An FPGA-based architecture for a latitude and longitude correction in autonomous navigation tasks," *Measurement*, vol. 182, p. 109757, 2021, doi: 10.1016/j.measurement.2021.109757.
[19] P. M. Soares dos Santos and J.A.F. Ferreira, "Novel intelligent real-time position tracking system using FPGA and fuzzy logic," *ISA Transactions*, vol. 53, 2, pp. 402-414, 2014, doi: 10.1016/j.isatra.2013.09.003.
[20] M. Dirik, A. F. Kocamaz and E. Donmez, "Implementation of fuzzy controller for mobile robot navigation on NI's Embedded-FPGA robotic platform," *Anatolian Journal of Computer Science*, vol. 4, no. 2, pp. 80-87, 2019. [Online]. Available: https://dergipark.org.tr/en/pub/bbd/issue/49546/496309.
[21] A. N. A-Amir and H. A. R. Akkar, "Artificial intelligent fuzzy logic controller applied on 6DOF robot Arm using LabVIEW and FPGA," *European Journal of Engineering Research and Science*, vol. 3, no. 5, pp. 1-8, 2018, doi: 10.24018/ejers.2018.3.5.661.
[22] O. Montiel, J. Camacho, R. Sepúlveda and O. Castillo, "Embedding a fuzzy locomotion pose controller for a wheeled mobile robot into an FPGA, soft computing for intelligent control and mobile robotics," in *Studies in Computational Intelligence book series*, vol. 318, pp. 465-481, 2010, doi: 10.1007/978-3-642-15534-5_28.
[23] O. Montiel, Y. Maldonado, R. Sepulveda and O. Castillo, "Simple tuned fuzzy controller embedded into an FPGA," *NAFIPS 2008 - 2008 Annual Meeting of the North American Fuzzy Information Processing Society*, 2008, pp. 1-6, doi: 10.1109/NAFIPS.2008.4531339.
[24] I. Millan, O. Montiel, R. Sepúlveda and O. Castillo, "Design and implementation of a hybrid fuzzy controller using VHDL," in *Soft Computing for Hybrid Intelligent Systems*, vol. 154, pp. 437-446, 2008, doiL 10.1007/978-3-540-70812-4_27.
[25] A. Hechri, R. Hmida, A. B. Abdelali and A. Mtiba, "Real time road lane markers detection for intelligent vehicles," *Advances in Environmental Biology*, vol. 8, no. 7, pp. 2266-2272, 2014. [Online]. Available: link.gale.com/apps/doc/A385069474/AONE?u=anon~fc71f2e5&sid=googleScholar&xid=a9f2109b.
[26] O. Munyaneza, B. B. Munyazikwiye and H. R. Karimi, "Speed control design for a vehicle system using fuzzy logic and PID controller," *2015 International Conference on Fuzzy Theory and Its Applications (iFUZZY)*, 2015, pp. 56-61, doi: 10.1109/iFUZZY.2015.7391894.
[27] C. C. Bitire, A. Ionașcu and M. M. Cojocaru, "Zybo autonomous car," *Digilent Design Contest Europe*, Romania, 2019, pp. 1-78. [Online]. Available: https://github.com/Catabit/Self-Driving-Car-ReleaseRepo-04BURO/blob/master/Documentation/Design%20Report.pdf.

BIOGRAPHIES OF AUTHORS

Emna Aridhi was born in Tunis, Tunisia. She is an assistant professor at the University of Carthage, Tunisia. She obtained her Ph.D. in Electronics from the Faculty of Sciences of Tunis (FST) in 2015. She is member of LAPER and works on developing and implementing control algorithms on embedded systems like FPGA and SoC.

Decebal Popescu graduated Computer Science at Universitatea Politehnica Bucureşti in 1998. He received M.Sc. and PhD degrees from the same institution in 1999 and 2003 respectively. Currently, he is Associated Professor at the Faculty of Automatic Control and Computer Science. His main research interests cover distributed computing, expert systems, network architectures, digital circuit design.

Abdelkader Mami received his Dissertation H.D.R (Enabling to Direct Research) from the University of Lille, France, in 2003. He is currently a Professor and a Scientific Advisor at the Faculty of Sciences of Tunis (FST), as well as the Header of LAPER.