Algerian Journal of Engineering and Technology

Journal homepage: https://jetjournal.org/index.php/ajet

Original Article

Motor speed control using a fault tolerance implementation on SRAM-based FPGA

El Habib Bensikaddour* and Aissa Boutte

Satellite Development Center Oran, Algeria

ARTICLE INFO

Article history:
Received 02 February 2023
Revised 14 May 2023
Accepted 01 June 2023

Keywords:
Microblaze Softcore;
Three Modular redundancy;
dependability;
critical-safety applications.

ABSTRACT

DC motor speed control is a critical task in many applications, such as industrial automation, aerospace and robotics. To ensure reliable and robust performance, a fault tolerance implementation is necessary. In this paper, we present a DC motor speed control system using an SRAM-based Field-Programmable Gate Array (FPGA) with a fault tolerance implementation. The control system utilizes a Pulse Width Modulation (PWM) and Proportional Integral Derivative (PID) to regulate the voltage applied to the motor. To ensure the reliability of the system, a MicroBlaze Triple Modular Redundancy is implemented, in which multiple controllers control the motor in parallel and their outputs are compared. The results show that the implementation significantly improves the reliability and robustness of the DC motor speed control system.

1. Introduction

Simply put, the use of any electric motor, regardless of size or function, requires a drive mechanism. These mechanisms can be simple on/off switches or by using a complex controls where dedicated techniques (motor controllers) are called upon to ensure the required operation. Irrespective of the type, a motor controller can manipulate the position, speed or torque of a motor. The key element in a system of DC motor speed control is the controller. This latter drives the motor speed by manipulating the input voltage of the motor. Several controllers are used and proposed in literature where the Proportional-Integral-Derivative (PID) is commonly used [1]. PID and Pulse Width Modulation (PWM) are used together to provide speed control. More precisely, the PID controls the PWM output according to the input parameters to obtain the minimum error between the set speed and the measured speed.

The implementation of PID and PWM is generally carried out on microcontrollers. However, modern FPGAs, specially SRAM-based, are considered as Systems on a Programmable Chip (SoPC) composed of a wide variety of resources (logic gates, processors, memory…etc.), allowing FPGA designers to build sophisticated and complex applications [2]. SRAM-based FPGAs are used in various applications requiring digital electronics (telecommunications, aeronautics, transport, etc.). They offer several advantages to system designers [2]: Hardware parallelism, Flexibility of Design, Low-Cost compared to ASIC and Maintainability [3]. Additionally, SRAM-FPGAs can be designed with fault tolerance techniques, such as redundancy and error correction codes, to ensure the reliability and robustness of the control system even in the event of faults. The dependability of the FPGA-based design is a key concern for safety-critical applications where the most serious problem is the vulnerability to errors [4].

Several works use FPGA chips to achieve PID and PWM functionality. Agarwal. C et al [5] present the results of an implementation of PID and PWM carried out on

* *Corresponding author*.
E-mail address: bensikaddour.elhabib@gmail.com
Peer review under responsibility of University of El Oued.
 https://dx.doi.org/10.57056/ajet.v8i2.137

Spartan-3E FPGA. Moreover, an implementation of a PID and PWM using a FPGA board (Altera DE0 kit) is described in [6]. In [7], a Comparison of High-Level Synthesis and Traditional RTL implementation of PWM is presented. A Design of FPGA-Based Neural Network PID and PWM is proposed in [8].

When errors or malfunctions put people's lives in risk or have the potential to inflict significant environmental and/or material damage, embedded system are always safety-critical [9]. In these types of applications, such as Aerospace, Automobile and Medical, the capability of an embedded system to survive in the presence of faults is essential [10]. It must be able to handle errors from: extreme powers and temperatures, device aging, radiation, ionization and component failures…. etc. [11]. Because of their excellent performance, low development costs, great density, and re-programmability, FPGAs are becoming more valuable for safety-critical applications [3, 12]. Unfortunately, when using commercial off-the-shelf (COTS) electronics in safety-critical applications, malfunctions may be unavoidable. For this, fault tolerance techniques must be used, which means that the system can tolerate faults and continue to operate using specific design techniques.

In this work, the solution provided by Xilinx to achieve fault-tolerant implementations is used to achieve speed control of a DC motor. The simulation and real test results demonstrate that the implementation significantly improves the DC motor speed control system's reliability and robustness, making it a promising solution for high-reliability applications.

The rest of this paper is organized as follows. Section II provides an overview of fault tolerance solutions for FPGA and discusses the key concepts of fault, error and failure. Section III presents the DC motor speed control system achieved in this work. Section IV provides details on the tests and results of the system, including simulations and real tests that were carried out to achieve the objective of this paper. Finally, Section V presents the conclusions.

2. Fault tolerance solutions for FPGA

Dependability refers to the ability of a system to function as intended, even in the presence of faults and failures. Understanding the causes of faults, errors, and failures is essential for designing and maintaining dependable systems [13]. The dependability of an embedded system, including FPGA-based, requires dealing with faults using techniques based on the Fault-avoidance and/or on the Fault-tolerance techniques [3]. Fault-avoidance and fault-tolerance are two related, but distinct, approaches to ensuring the dependability of a system. The main difference between the two is that fault-avoidance aims to prevent faults from occurring, while fault-tolerance aims to handle faults when they do occur [2]. Fault-Avoidance techniques use components that are manufactured with specific processes to achieve high dependability. For example, in the space industry, a Radiation Hardened By Process (RHBP) components are used [14]. Unfortunately, these components are expensive and not easy to purchase as they are protected by the United States International Traffic in Arms Regulations (ITAR) or the United States Export Regulations (EAR) [15]. On the other hand, Fault-Tolerance techniques allow the FPGA to function properly instead of crashing completely [13]. In the literature, many fault tolerance methods of SRAM-based FPGAs are covered [3], which can be summarized as follow:

- Error Masking: Errors are only masked and not corrected. An error that has been masked won't spread throughout the implementation. When an error happens in a specific area of the circuit, its impact is preserved in the design and is not fixed. The system may crash as a result of error accumulation brought on by error masking. In this category, Three Modular Redundancy (TMR) is the most used techniques to mask errors by combing the outputs of three identical modules via a voter [16].
- Error Correction: Errors are corrected by using Error Detection and Correction (EDAC) code or by Configuration scrubbing [2, 12, 17-21]. In configuration scrubbing, the design has the ability to rewrite the correct configuration back to SRAM. It can be classified as follows [21]:

    1. Full scrubbing: The simplest scrubbing methodology is to cyclically reconfigure the FPGA, without performing any error detection. This is called preventive or indiscriminate cleaning.
    2. Partial Scrubbing: Apply a partial reconfiguration of the affected portions of configuration memory of FPGA. A detection mechanism is essential to identify the area to be reconfigured.
    3. It is important to keep in mind that the specific techniques used to improve reliability will depend on the specifics of the system and requirements. A combination of techniques may be needed to achieve the desired level of reliability.

3. DC motor speed control achieved

The objective of this work is to accomplish high-reliability speed control of a DC motor using SRAM-FPGA.

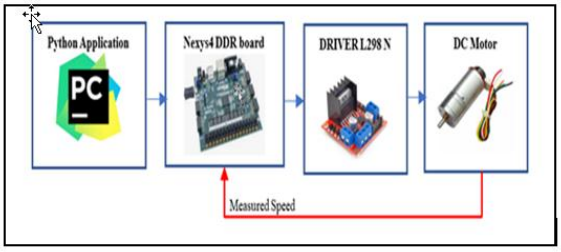

Fig 1. DC Motor Speed Control Achieved.

This system comprises four main components (Figure 1):

- Python application: An application has been developed (Figure 2) using Python language for controlling the speed of a DC motor. This application uses UART communication to configure the various necessary parameters, such as the gains of the PID controller, the sampling time and the speed reference. In addition, it is able to measure the motor speed in real time using suitable sensors and display it in the form of a number and a graph. This application offers great flexibility to adjust the control parameters according to specific application needs, which helps to optimize motor performance in terms of precision, stability and responsiveness.
- Nexys4 DDR board: The Nexys4 DDR is a development board designed by Digilent. It features a Xilinx Artix-7 FPGA, as well as several peripherals including DDR2 memory, Ethernet, USB, and a variety of other interfaces. The board is intended for use in a variety of applications, including embedded systems development, digital signal processing, and more.
- Driver L298N module: The L298N driver has been used in this work. it allows the control of DC motors under a constant voltage between 5 V up to 35 V, with a maximum current can wait 2A.
- DC motor with a rotary encoder: The JGA 25-370 is a DC motor commonly used in small robotics and automation applications.

Fig 2. Python Application.

3.1 First Inplementation

First, the implementation shown in Figure 3 has been performed. This implementation controls the motor speed without the high reliability technique. The objective of this first implementation is to achieve a high performance and resource-efficient implementation that can accurately control the speed of a DC motor using the Microblaze processor. This implementation will be extended to be compatible with the TMR Microblaze solution, thereby providing processing redundancy and improved reliability.

The first implementation utilizes several IP cores, including Microblaze Softcore, FIT (Fixed Interval Time), AXI Uartlite, PWM, and Mycounter. The first three IP cores are provided by Xilinx, while the last two are developed using Verilog HDL and created as new IP in the VIVADO software.

The implementation follows a specific operational sequence:

1. The FIT core is programmed to trigger an interrupt signal every 100ms.
2. Upon the signal's triggering, the Microblaze softcore reads the pulse count from the Mycounter core to calculate the motor's rotational speed.
3. The calculated speed is then used to determine the required PWM width, using PID, to achieve the desired speed, which is adjustable via the Python application.

3.2 Second Inplementation

To ensure high reliability, the solution provided by Xilinx has been utilized [22]. The High-Reliability

MicroBlaze Solution is designed for harsh environments and includes the necessary IP cores to create a redundant, triplicated MicroBlaze subsystem. This solution is of the Fault-Tolerant and Fail-Safe type, providing soft error detection, correction, and recovery, ensuring continued operation even after the first failure, and detecting a second failure. This implementation has been extended to be compatible with the TMR Microblaze solution, thereby providing processing redundancy and improved reliability.

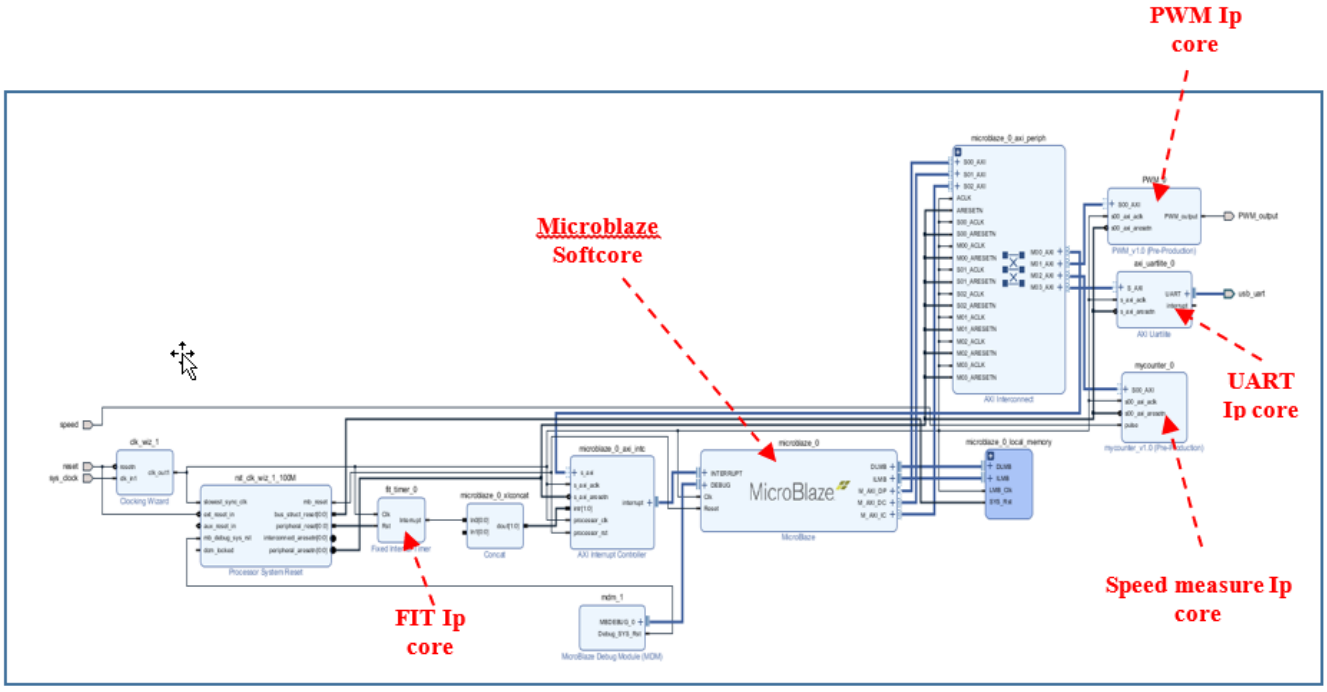

Fig 3. validated implementation to control the motor.

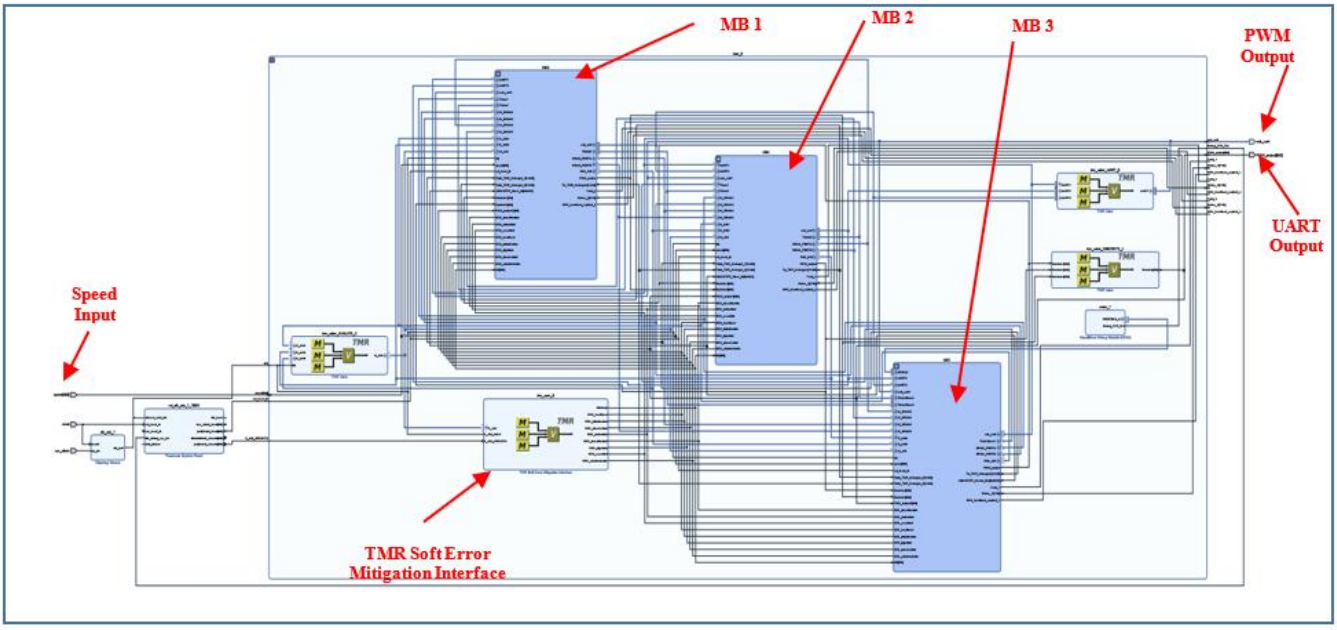

Fig 4. High reliability implementation of DC motor control

In order to implement the solution shown in Figure 4, the steps outlined in the document [22] published by Xilinx have been followed. The three identical modules (MB1, MB2, and MB3) that make up this implementation will work together in redundancy to achieve the previously mentioned Fault-Tolerant - Fail Safe operation.

3.3 Microblaze Program

MicroBlaze is a 32-bit softcore processor developed by Xilinx for use in FPGAs [22]. It is designed to be simple, flexible, and configurable, allowing developers to create custom applications and perform tasks such as data processing, control, and communication.

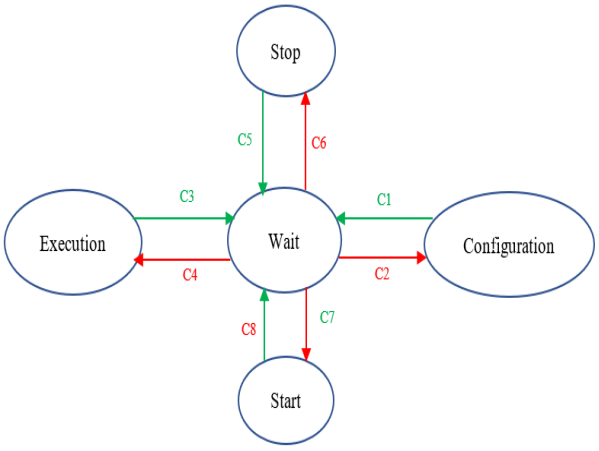

Fig 5. Microblaze states.

In this work, the Microblaze switches between four states, shown in the Figure 5:

- Wait Sate: This is the idle state where the Microblaze processor waits for events before transitioning to other states. The processor can be configured through the Configuration state by receiving a request (C2) from the Python application. This request sets the PID parameters and the desired speed. The Start state is entered when the processor receives a start request (C7) from the Python application. The execution state is activated at each interrupt generated by the FIT core (C4). The operation can be stopped at any time by receiving a stop request (C5) from the Python application, which transitions the processor to the Stop state.
- Configuration State: The Configuration state is dedicated to setting the PID parameters (Kp, Ki, Kd, and Tm) and the speed of the motor. The necessary information is received from the application via UART communication. Upon completing the configuration process, the state transitions automatically to its previous state (C1).
- Start state: The Run state is responsible for initiating the operation. Upon completion, the state returns automatically to the Wait state.
- Execution State: The Microblaze processor calculates the motor speed based on the number of pulses counted by the Mycounter core. The processor then utilizes a PID control algorithm to determine the necessary PWM width. The transition from this state to the Wait state is seamless, taking place automatically once the execution is finished
- Stop State: This state is validated when a request is transmitted by the python application. The return to wait state is generated automatically.

4. Tests and Results

To validate the PWM IP core and ensure its accuracy, a simulation has been performed. The results in Fig. 6 show the PWM duty cycles for three settings (a: 50%, b: 25%, and c: 75%), which were obtained by simulation using VIVADO. Based on the obtained results, we can conclude that the IP core can generate PWM signals accurately according to the required duty cycle.

The results in Fig. 6. show the PWM duty cycles for three settings (a: 50%, b: 25%, and c: 75%), which were obtained by simulation using VIVADO. Based on the obtained results, we can conclude that the IP core can generate PWM signals accurately according to the required duty cycle.

Fig. 6. Simulation Results of PWM IP core; (a): 50% Duty Cycle, (b) 25% Duty Cycle, 75% Duty Cycle.

Efficient use of resources is important for designing high-performance and cost-effective embedded systems. In Table 1, we can see the percentages of resources consumed in the ARTIX-7 chip used for the implementation.

Table 1: Device Utilization

Resources	**Utilisation (%)**	
	High-Reliability	**Simple**
LUT	19.05	4.46
LUTRAM	5.07	1.72
FF	8.89	2.16
BRAM	25.56	13.33
BUFG	12.5	12.5
MMCM	16.67	16.67

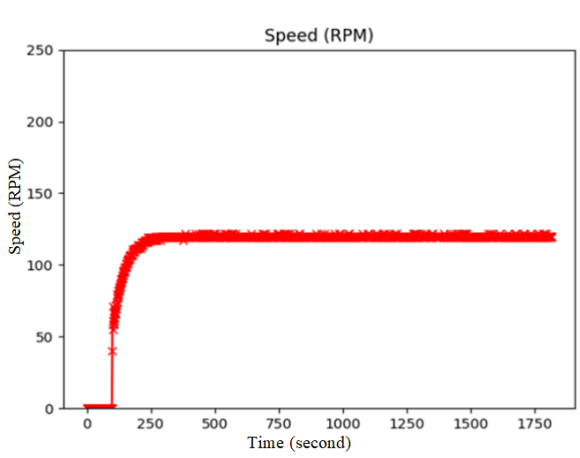

Fig 7. Test carried out at 120 RPM.

In the second stage, experimental test results were provided to objectively evaluate the success of the implementation and identify any potential issues or areas for improvement. Figure 7 displays a portion of a trial test conducted at 120 RPM. The implemented application enables accurate regulation of the motor speed based on the setpoint speed, indicating the effectiveness of the design.

5. Conclusion

In conclusion, selecting the right FPGA for critical applications, such as aerospace, is a vital task. The selection is based primarily on the possibility of surviving in the target environment and the reliability required for the intended application. This paper has presented a DC motor speed control system that utilizes an SRAM-based FPGA with a fault tolerance implementation. By using Pulse Width Modulation and Proportional Integral Derivative, the system can regulate the voltage applied to the motor, which is a critical task in many industrial, aerospace and robotic applications. The fault tolerance implementation is achieved by using MicroBlaze Triple Modular Redundancy, which ensures the system's reliability and robustness. The simulation and real test results demonstrate that the implementation significantly improves the DC motor speed control system's reliability and robustness, making it a promising solution for high-reliability applications.

List of Acronyms

ADC Analog-To-Digital Converter.
ASIC Application-Specific Integrated Circuit.
BRAM Block RAM.
BUFG Buffer Global clock.
CPU Central Processing Unit.
FIT Fixed Interval Time.
FF Flip-Flop.
FPGA Field-Programmable Gate Array.
IP Intellectual Property.
LUT Look-Up Table.
LUTRAM Look-Up Table Random Access Memory.
MMCM Mixed-Mode Clock Manager.
RAM Random Access Memory.

Conflict of Interest

The authors declare that they have no conflict of interest.

References

1. Joseph SB, Dada EG, Abidemi A, Oyewola DO, Khammas BM. Metaheuristic algorithms for PID controller parameters tuning: Review, approaches and open problems. Heliyon. 2022.
2. Kastensmidt F, Rech P. FPGAs and parallel architectures for aerospace applications. Soft Errors and Fault-Tolerant Design. 2016.
3. Bernardeschi C, Cassano L, Domenici A. SRAM-based FPGA systems for safety-critical applications: A survey on design standards and proposed methodologies. Journal of Computer Science and Technology. 2015;30:373-390.
4. Zhu ZK, Yuan Y, Zhang XH. Theodolite-camera videometrics system based on total station. InInternational Symposium on Photoelectronic Detection and Imaging 2011: Advances in Imaging Detectors and Applications 2011 Aug 18 (Vol. 8194, pp. 706-715). SPIE.
5. Agarwal C, Gupta A, Modeling, simulation-based DC motor speed control by implementing PID controller on FPGA. 2013.
6. Jain RV, Aware MV, Junghare AS. Implementation of a PID control PWM module on Altera DE0 kit using FPGA. In2016 IEEE First International Conference on Control, Measurement and Instrumentation (CMI) 2016 Jan 8 (pp. 341-345). IEEE.
7. Kangas N, A Comparison of High-Level Synthesis and Traditional RTL in Software and FPGA Design. 2020.
8. Wang J, Li M, Jiang W, Huang Y, Lin R. A design of FPGA-based neural network PID controller for motion control system. *Sensors*. 2022;22(3):889.
9. Fowler K. Mission-critical and safety-critical systems handbook: Design and development for embedded applications2009: Newnes.
10. Leroux P. Radiation tolerant electronics. Electronics. 2019;8(7):730.
11. Michael Henze MME. FPGAs enable development of safety-critical embedded systems. Safety & Security, 2018. EmbeddedWorld2018.
12. Kastensmidt FL, Carro L, da Luz Reis RA. Fault-tolerance techniques for SRAM-based FPGAs. Dordrecht: Springer; 2006.
13. Brosser F, Milh E. SEU mitigation techniques for advanced reprogrammable FPGA in space. Chalmers University of Technology, 2014.
14. Cao Y, Leroux P, Steyaert M. Radiation-tolerant Delta-sigma Time-to-digital Converters. Springer International Publishing; 2015.
15. León AF. Trends and patterns of ASIC and FPGA use in European space missions. 2013.
16. Carcheri JC. N-Modular Redundancy Techniques for Fault Tolerance in Reconfigurable Logic Devices. Master's Project Report. 2015.
17. Kastensmidt, F. and R. Reis, Soft error rate and fault tolerance techniques for FPGAs, in Circuit Design for Reliability 2015, Springer. p. 207-221.
18. Yang M, Hua G, Feng Y, Gong J. Fault-tolerance techniques for spacecraft control computers. John Wiley & Sons; 2017.
19. Bridgford B, Carmichael C, Tseng CW. Single-event upset mitigation selection guide. Xilinx Application Note, XAPP987 (v1. 0). 2008:69.
20. Assurance SP. Techniques for Radiation Effects Mitigation in AASIC and FPGAs Handbook. Technical report, ESA Requirements and Standards Division; 2016.
21. Stoddard AG. Con guration Scrubbing Architectures for High-Reliability FPGA Systems. Brigham Young University; 2015.
22. Xilinx, MicroBlaze Triple Modular Redundancy (TMR) Subsystem v1.0. Product Guide, 2018.

Recommended Citation

Bensikaddour E, Boutte A. Motor speed control using a fault tolerance implementation on SRAM-based FPGA. *Alger. J. Eng. Technol. 2023*, 8(2): 302-308. https://dx.doi.org/10.57056/ajet.v8i2.137