International Journal of Robotics and Control Systems
Vol. 6, No. 2, 2026, pp. 1079-1099
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

Resource-Efficient Model Predictive Control on a Low-End Field-Programmable Gate Array for DC Motor Speed Control

Aulia Rasyid Pratama [a,1], Jazi Eko Istiyanto [a,2,*], Andi Dharmawan [a,3]

[a] Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada, Yogyakarta, 55281, Indonesia
[1] rasyidpratama23@mail.ugm.ac.id; [2] jazi@ugm.ac.id; [3] andi_dharmawan@ugm.ac.id
* Corresponding Author

ARTICLE INFO

Article history
Received January 04, 2026
Revised March 02, 2026
Accepted April 08, 2026

Keywords
Low-End FPGA;
Resource Efficiency;
Model Predictive Control;
ADMM Optimization;
DC Motor Speed Control

ABSTRACT

MPC offers advantages in controlling dynamic systems by predicting future behavior based on mathematical models while respecting system constraints. However, increasing model complexity raises the computational burden, which becomes a challenge for low-end FPGAs with limited resources and studies on efficient MPC realization for such platforms remain limited. This study addresses that gap by demonstrating a simplified MPC implementation that maintains real-time performance and acceptable control accuracy while operating within the tight resource constraints of a low-end FPGA (Lattice iCE40HX1K). Several optimization strategies are applied, including simplifying the MPC structure using a 16-bit fixed-point representation (Q13.3), performing model coefficient precomputation outside the FPGA, replacing division operations with equivalent multiplications, minimizing the prediction horizon ($N_p = 2$), and employing fixed scalar penalty weights to avoid matrix operations. A fully pipelined architecture is designed to ensure deterministic execution timing. A DC motor speed control system is used as a proof-of-concept and tested under false-edge disturbances (2 kHz, 5 kHz, and 10 kHz). Results show that the design operates within the tight logic and memory constraints of the Lattice iCE40HX1K FPGA, utilizing 99% of available logic cells. The proposed design also meets real-time requirements with a computation time of 1.83 µs and maintains consistently stable control performance with steady-state accuracy above 98%. Nevertheless, limited fixed-point precision, a short prediction horizon, and a simplified EMI noise model may reduce robustness under complex disturbances. Overall, the research contribution is the development of a resource-efficient MPC tailored for low-end FPGAs, broadening MPC accessibility beyond mid- to high-end platforms.

1. Introduction

The fundamental principle of model predictive control (MPC) is to utilize a mathematical model of the system to predict and regulate future control behavior, where at each sampling iteration, the control action is chosen based on the cost function employed [1]. By leveraging the system's mathematical model, MPC can generate an optimal control signal that keeps the system output close

to the reference value [2]. In addition, MPC excels at handling dynamic systems with explicit constraints, while still optimizing control performance within the prediction horizon.

The more accurate the mathematical model of the system is, meaning the closer it reflects the actual system behavior, the more accurate the control signal produced by MPC will be [3]-[5]. Therefore, the mathematical model of the system can be extended to represent uncertainties, such as noise in the system input signal, allowing MPC to address these uncertainties. By using a mathematical model that captures uncertainties while also integrating the current input and previous measurement results, MPC is still able to generate suitable control signals even when the system input signal is inaccurate [6]-[8].

Nevertheless, the more complex the mathematical model of the system is, the heavier the computational load required becomes. A high computational burden in the prediction and control optimization process can sometimes hinder the real-time execution of MPC [2], [9]. A commonly used effective approach to overcome this problem is the use of high-speed hardware technologies [9]-[12], although it does not eliminate the overall complexity of MPC. One such technology is the field-programmable gate array (FPGA). FPGA offers advantages in terms of parallel computation and power efficiency. Compared with other high-speed hardware technologies such as central processing units (CPUs), graphics processing units (GPUs), and application-specific integrated circuits (ASICs), FPGA generally has the advantage of high parallel computation capability, better resource efficiency, and development flexibility due to its reconfigurable nature [9]. This combination of advantages makes FPGA an optimal choice for implementing MPC.

Although FPGAs offer advantages in terms of parallel computation and power efficiency, they still have limitations in the amount of available resources, particularly logic and memory. Meanwhile, the MPC method requires complex, real-time, and repetitive prediction and control computations. The gap between the computational demands of MPC and the limited resources of FPGAs poses a significant challenge in achieving both real-time performance and control accuracy, particularly on low-end platforms [7], [13], [14]. The computational demand of MPC also depends on the model complexity and the prediction horizon. Nevertheless, most FPGA-based MPC implementations reported in the literature continue to rely on mid- to high-end FPGAs (e.g., Xilinx Zynq, Intel Cyclone, or Intel Arria) [15], [16]. These implementations primarily target high-performance control systems. As a result, they are not directly optimized for resource-limited platforms. Consequently, there remains a research gap in exploring how MPC can be realized efficiently on low-end FPGAs for relatively simple control systems. Low-end FPGAs are characterized by very limited hardware resources, including look-up tables (LUTs), flip-flops, block RAM (BRAM), and dedicated DSP blocks, and are typically classified as platforms providing fewer than 2,000 LUTs and no dedicated DSP blocks [14], [17].

This study addresses that gap by demonstrating a simplified MPC implementation that maintains real-time performance and acceptable control accuracy while operating within the tight resource constraints of a low-end FPGA (e.g., Lattice iCE40HX1K FPGA). In contrast to most prior FPGA-based MPC implementations that focus on performance enhancement using mid- to high-end FPGAs, this work redesigns the MPC architecture to operate within the tight resource constraints of a low-end FPGA. Rather than relying on matrix precomputation techniques [18] or adopting a multiparametric Explicit MPC formulation with precomputed region-based lookup tables, the proposed optimizations retain an online optimization structure while emphasizing architectural simplification and numerical refinement to reduce computational complexity and memory usage. The proposed optimizations include simplifying the MPC structure using a 16-bit fixed-point representation (Q13.3), performing model coefficient precomputation outside the FPGA, replacing division operations with equivalent multiplications, minimizing the prediction horizon, and employing fixed scalar penalty weights to avoid matrix operations. The implemented MPC operates with limited fixed-point precision and a short prediction horizon due to the severe hardware constraints.

With this approach, MPC can be effectively implemented on a low-end FPGA. To validate the feasibility and performance of the proposed design, a DC motor speed control problem is used as a

proof-of-concept case study. In this setup, false-edge disturbances are introduced into the feedback signal to emulate EMI noise. A false edge refers to an unintended rising or falling transition detected due to distortion of the encoder signal, which causes errors in pulse detection [19]. The evaluated parameters are total FPGA resource utilization within the Lattice iCE40HX1K FPGA resource limit, MPC computation time with a maximum of 1 ms, and control performance with at least 80% in steady-state accuracy. The results are expected to provide a useful reference for developing MPC implementations in real industrial applications using low-end FPGAs, such as low-cost industrial motor drives, distributed automation systems, and embedded motion control systems. By enabling MPC implementation on low-end FPGA, the proposed optimization approach broadens the accessibility of MPC beyond mid- to high-end FPGAs while maintaining reliable control performance, even in the presence of feedback disturbances.

The main contribution of this paper is the development of a resource-efficient MPC architecture specifically tailored for low-end FPGA platforms. In contrast to most prior FPGA-based MPC implementations that rely on mid- to high-end FPGAs or matrix precomputation techniques, this work demonstrates that online MPC can be restructured to operate within severe logic and memory constraints while maintaining real-time computation time and stable control performance. The feasibility of the proposed design is experimentally validated on a Lattice iCE40HX1K FPGA using a DC motor speed control case study with injected feedback disturbances.

The remainder of this paper is organized as follows: Section 2 surveys related works on MPC implementations across various platforms and resource optimization strategies, emphasizing the advantages of FPGA-based implementations and the research gap addressed in this study. Section 3 Resource-Efficient MPC Implementation on Low-End FPGA describes the implementation of MPC on FPGA, covering system hardware design, MPC modeling, FPGA implementation, and the experimental methodology. Section 4 Results and Discussionis dedicated to experimental results and discussion. Finally, Section 5 Conclusionconcludes this paper.

2. Related Works

2.1. MPC Implementation

MPC has been implemented on various hardware technologies, such as CPUs [20]-[22], GPUs [23]-[26], and FPGAs [1], [2], [27], for different purposes. Several works discuss performance comparisons among these hardware technologies, with a particular focus on parallel computing performance and computation time [11], [28]. In [9], the implementation of MPC on a CPU and an FPGA was compared for trajectory control in autonomous driving, using a double lane-change scenario as the trajectory reference. The results showed that although the control performance was not significantly different, the FPGA achieved up to 27 times faster parallel computation compared to using only the CPU. Similar evaluations were also reported in [12], [29], which compared parallel computing performance between FPGA and CPU. The results consistently demonstrated the superiority of FPGA in terms of parallel computation.

In addition to hardware implementations of MPC, several works have also focused on software implementations to evaluate MPC performance. Such implementations are usually employed as an initial stage before deployment on hardware, with the goal of comparing model accuracy, control performance, and the required computational load. In [30], [31], the results of software implementations were compared with actual FPGA implementations, revealing performance differences. The findings showed that FPGA implementations were able to provide improved computation times compared with software implementations. Software implementations are also often used to test MPC scenarios, such as parameters in the cost function. A relevant example is [32], where MPC was tested in a software-in-the-loop (SIL) environment with four different cost functions.

MPC has been widely applied in various case studies, ranging from motor control [2], [32], [33], robotics [20], [21], [34]-[36], automotive applications [23], [37], [38], and renewable energy systems [39]-[41]. The main advantage of MPC lies in its ability to handle dynamic systems through future

prediction based on the system's mathematical model, while respecting the applied control and state constraints. An interesting approach is presented in [7], [8], [42], where noise in the system input signal is explicitly modeled within the system's mathematical model. As a result, MPC was able to manage this uncertainty while maintaining stable control performance. The implementation of MPC on FPGA also offers advantages by leveraging its available parallel computing capability. Moreover, the use of FPGAs has been proven effective for control systems [43]-[46].

2.2. FPGA Resource Optimization

Most existing works on FPGA-based MPC implementations employ mid- to high-end FPGAs, such as the Xilinx Zynq, Intel Cyclone, and Intel Arria families, which provide abundant computational resources. For instance, the work reported in [15] implemented an MPC scheme on a Xilinx Zynq-7000 SoC, utilizing approximately 46,805 LUTs, 4,078 FFs, 2 RAMs, and no DSP blocks for a three-phase voltage source inverter control. Similarly, [16] presented an MPC implementation with a long prediction horizon for power electronics systems, consuming 73,685 LUTs, 134,225 FFs, 134 RAMs, and 68 DSPs on an Intel Cyclone FPGA, and 24,660 LUTs, 62,040 FFs, 127 RAMs, and 121 DSPs on an Intel Arria FPGA.

Although these systems are considerably more complex than simpler control applications such as DC motor speed control, their consistently high FPGA resource utilization indicates that current MPC architectures remain computationally demanding and are primarily designed for platforms with abundant hardware resources. This observation highlights the limited research attention devoted to developing and evaluating resource-efficient MPC structures for low-end FPGAs and less computationally intensive systems. To clearly define this category, low-end FPGAs are characterized by very limited hardware resources, including look-up tables (LUTs), flip-flops, block RAM (BRAM), and dedicated DSP blocks, and are typically classified as platforms providing fewer than 2,000 LUTs and no dedicated DSP blocks [14], [17]. The Lattice iCE40HX1K FPGA contains exactly 1,280 LUTs and no dedicated DSP blocks.

Several studies have proposed strategies to improve the hardware efficiency of MPC implementations. For example, [1] achieved reduced FPGA utilization by employing fixed coefficients instead of adaptive ones, while [27] focused on lowering memory requirements through the adoption of the posit number format, achieving a 50%–70% reduction in memory usage without degrading control performance. However, both approaches were still evaluated on high-end FPGAs with ample resources, leaving their effectiveness on low-end platforms uncertain.

A relevant effort is presented in [18], where MPC resource usage was optimized by precomputing and storing costly matrix factorizations, later implemented on an MCU for robot control. Although this demonstrates a promising direction for lightweight MPC computation, its implementation on an MCU limits direct generalization to FPGA-based systems, where computation is mapped into hardware structures rather than software routines. Some modern MCUs do provide hardware math accelerators. However, the optimization process remains sequential. Since the proposed design maintains online optimization at each sampling period, deterministic execution timing is required to ensure real-time feasibility. A low-end FPGA, such as the Lattice iCE40HX1K, enables this loop to be implemented as dedicated synchronous hardware, ensuring fixed latency. These findings collectively suggest that while prior studies have explored methods to enhance MPC computational efficiency, there remains a lack of investigation into MPC architectures explicitly designed for low-end FPGAs with stringent hardware limitations.

In contrast to the aforementioned studies, the present study specifically targets the development and implementation of an MPC architecture optimized for a low-end FPGA (Lattice iCE40HX1K) under strict resource constraints. Unlike previous approaches that focused on matrix precomputation [18] or adopting a multiparametric Explicit MPC formulation with precomputed region-based lookup tables, this work emphasizes architectural simplification and numerical optimization tailored for a low-end FPGA. The proposed design strategically reduces computational complexity and memory usage by minimizing the prediction horizon, employing precomputed model parameters, and

simplifying mathematical operations to avoid costly divisions and matrix operations. These choices collectively enable a practical and resource-efficient MPC implementation suitable for low-complexity control tasks such as DC motor speed control under noise disturbances.

3. Resource-Efficient MPC Implementation on Low-End FPGA

As shown in Fig. 1, this research was conducted in several stages to develop a resource-efficient MPC on a low-end FPGA (Lattice iCE40HX1K). First, the system hardware was designed. Second, the MPC algorithm was redesigned and optimized for low-end FPGA implementations, emphasizing architectural simplification and numerical refinement to reduce computational complexity and memory usage. Supporting modules such as the UART communication module and clock divider were designed and verified to enable system integration. Finally, experimental validation was conducted to evaluate the designed MPC.

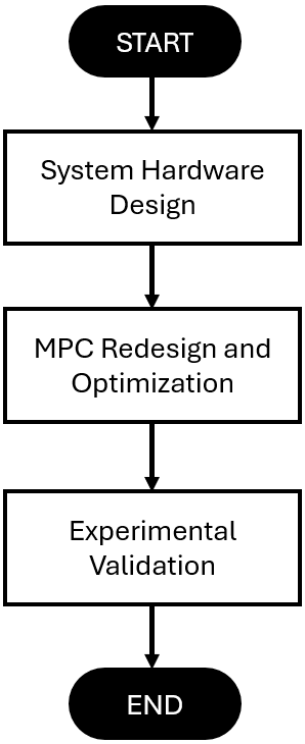

Fig. 1. Research methodology flowchart

3.1. System Hardware Design

The hardware design of the system is shown in Fig. 2. The hardware design consists of the Lattice iCE40HX1K FPGA (Lattice iCEstick), a JGA25-370 1360 RPM DC motor, a hall encoder sensor, an L298N motor driver, a logic level shifter, and 12 V and 5 V adapters. The FPGA is connected to a laptop via USB to receive power while simultaneously monitoring the DC motor speed during operation. The adapter supplies 12 V for the DC motor and 5 V for the system logic and the hall encoder sensor. The hall encoder sensor is used to measure the speed of the DC motor. The output data from the hall encoder sensor is then connected to the arbitrary function generator (AFG) output to introduce EMI noise effects (false edges). The hall encoder sensor output, distorted by the EMI noise effects, is then used as the system input to be processed by the MPC on the FPGA. The EMI noise effects (false edges) on the system input signal will cause the measured DC motor speed to differ from its actual value. The MPC output on the FPGA, which is the optimal control signal in the form of PWM, is then forwarded to the L298N driver to control the DC motor. To ensure logic level compatibility between the FPGA and the L298N, the control signal from the FPGA is first passed

through a logic level shifter, since the Lattice iCE40HX1K FPGA operates at 3.3 V while the L298N operates at 5 V.

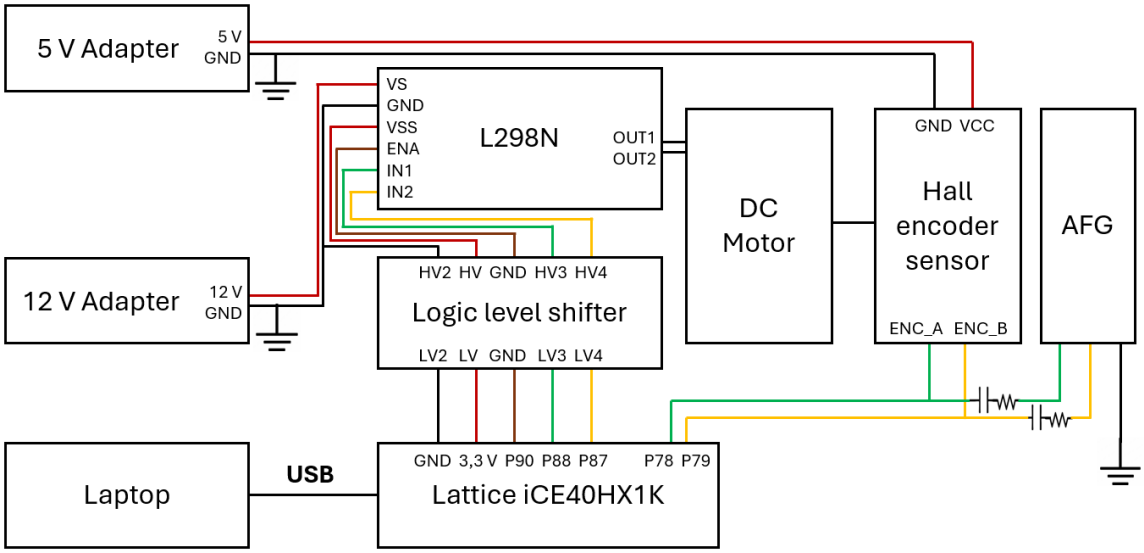

Fig. 2. The hardware design of the system

3.2. MPC Modelling

3.2.1. Mathematical Model of EMI Noise Effects

The fundamental principle of the RPM formula for DC motor is the use of an incremental encoder to calculate the pulses generated per revolution of the rotating shaft [47]-[50]. The ideal DC motor speed measurement (without noise) based on encoder pulses is shown in (1), where $N_{count}[k]$ is the number of pulses detected during the sampling period, T_s is the sampling period used, and PPR is the number of pulses per motor rotation.

$$RPM_k = \frac{N_{count}[k]}{PPR \cdot T_s} 60 \tag{1}$$

In this paper, the EMI noise effect is represented by false edges on the encoder signal. This representation is possible because EMI coupled into the encoder signal can induce transient voltage spikes or distortions that exceed the threshold, thereby generating unintended rising or falling transitions interpreted as valid pulses [19]. Furthermore, this causes the detected pulse count ($N_{count}[k]$) to either increase or decrease compared to the actual value, which in turn leads to inaccurate DC motor speed measurements [51], [52]. False edges are generated by adding an additional signal from an AFG configured to generate a square wave with a certain frequency and an amplitude of 2 V (peak-to-peak).

To maintain consistency with pulse encoder-based speed calculations, a clipping method is applied as a limiting mechanism. This method ensures that the number of pulses detected by the sensor does not exceed the maximum number of pulses theoretically capable of being generated by the sensor based on the actual speed of the DC motor [53]-[55]. This is shown in (2).

$$N_{count}[k] = \begin{cases} N_{max}, & N_{count}[k] > N_{max} \\ N_{count}[k] \end{cases} \tag{2}$$

Where N_{max} is the maximum possible number of pulses corresponding to the actual motor speed. If the detected pulse count ($N_{count}[k]$) exceeds the maximum value that could occur based on the motor speed, the system assumes the signal has been affected by EMI noise and is therefore discarded and replaced with the maximum pulse count limit.

3.2.2. Mathematical Model of DC Motor

A DC motor is an electromechanical system that receives an electrical signal as input and produces mechanical displacement as output [56]-[58]. The mathematical model of a DC motor consists of two main parts: the electrical system and the mechanical system [53], [59], [60]. In the electrical system, the relationship between voltage, current, and motor speed is shown in (3), where V_m is the armature voltage, R_m is the armature resistance, L_m is the armature inductance, and V_b is the back electromotive force (EMF).

$$V_m = R_m i_m + L_m \frac{di_m}{dt} + V_b \tag{3}$$

Since the back EMF is proportional to the motor constant (K_m) and the motor angular speed (ω_m), it can be reformulated as (4).

$$V_m = R_m i_m + L_m \frac{di_m}{dt} + K_m \omega_m \tag{4}$$

In the mechanical system, the relationship among the moment of inertia, the mechanical damping coefficient, and torque is shown in (5), where J_m is the motor's moment of inertia, $\frac{d\omega_m}{dt}$ is the angular acceleration, B_m is the mechanical damping coefficient, ω_m is the motor angular speed, T_{load} is the external load torque, and T_m is the torque generated by the motor.

$$J_m \frac{d\omega_m}{dt} + B_m \omega_m + T_{load} = T_m \tag{5}$$

Since the motor torque is proportional to the motor torque constant (K_t) and the armature current (i_m), it can be reformulated as (6).

$$J_m \frac{d\omega_m}{dt} + B_m \omega_m + T_{load} = K_t i_m \tag{6}$$

The continuous model is then transformed into a state-space form to represent the system dynamics linearly. The state variable (x) used is the motor angular speed (ω_m), while the control input variable (u) is the armature voltage signal (V_m), which is converted into a PWM signal to be applied to the DC motor. Since only one state variable is used, namely the motor angular speed (ω_m), the motor armature current (i_m) must be substituted from the DC motor electrical system equation into the DC motor mechanical system equation. The motor armature current (i_m) is shown in (7), where L_m is very small so that $L_m \frac{di_m}{dt} \approx 0$. The T_{load} used in this paper is set to zero. Therefore, the DC motor model in state-space form is shown in (8).

$$i_m = \frac{V_m - K_m \omega_m}{R_m} \tag{7}$$

$$\frac{d\omega_m}{dt} = -\left(\frac{B_m}{J_m} + \frac{K_t K_m}{J_m R_m}\right)\omega_m + \frac{K_t}{J_m R_m} V_m - \frac{T_{load}}{J_m} \tag{8}$$

Next, since MPC operates in the discrete-time domain, (8) needs to be expressed in matrix form and discretized. The discretization is performed exactly using the matrix exponential, under the assumption that the input $u[k]$ remains constant within each sampling interval T_s. Consequently, matrices A, B, and C are obtained as shown in (9)-(11), while α and β are presented in (12) and (13). The state vector $x[k+1]$ and the output signal $y[k]$ are given in (14) and (15).

$$A = e^{\alpha T_s} \tag{9}$$

$$B = \beta \cdot \frac{1 - e^{\alpha T_s}}{\alpha} \tag{10}$$

$$C = 1 \tag{11}$$

$$\alpha = -\left(\frac{B_m}{J_m} + \frac{K_t K_m}{J_m R_m}\right) \tag{12}$$

$$\beta = \frac{K_t}{J_m R_m} \tag{13}$$

$$x[k+1] = Ax[k] + Bu[k] \tag{14}$$

$$y[k] = Cx[k] \tag{15}$$

3.2.3. MPC Design

The block diagram of the MPC design on the FPGA for the DC motor control system is shown in Fig. 3. The MPC input is the output signal from the hall encoder sensor, which is affected by EMI noise (false edges). This signal is processed into DC motor speed data, which is then used in the mathematical system model (including the mathematical model of EMI noise effect and the DC motor model) to predict the motor speed. The prediction result is then compared with the reference speed value, producing a prediction error. This error is processed by the MPC optimizer to generate the optimal control signal based on the cost function and the applied system constraints.

The cost function in the MPC method is designed to determine the most optimal control signal so as to minimize the difference between the reference speed value and the actual speed of the DC motor. Equation (16) shows the cost function used.

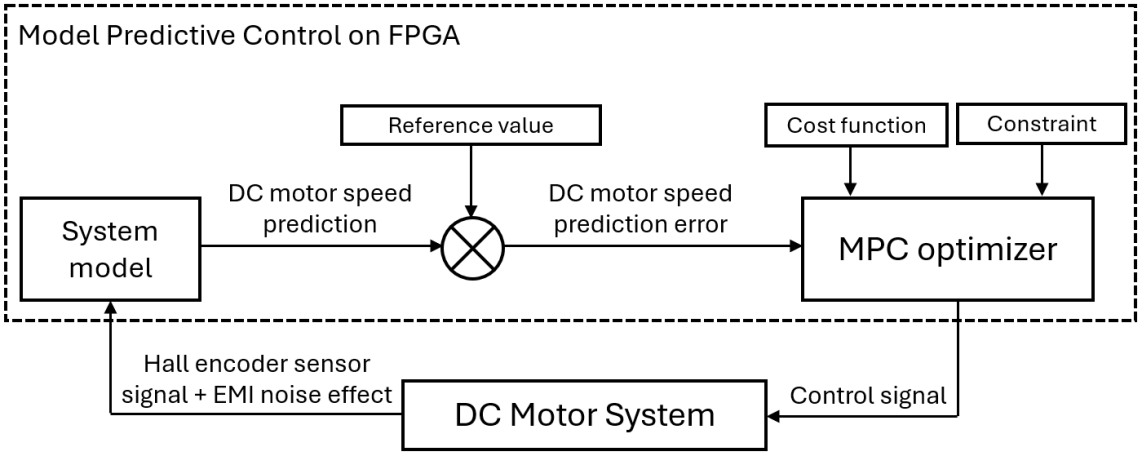

Fig. 3. The block diagram of the MPC design on the FPGA for DC motor control system

$$J = \sum_{i=0}^{N_p-1} (Q \cdot (\omega_m[k+i] - r[k+i])^2 + R \cdot u[k+i]^2) \tag{16}$$

Where N_p is the prediction horizon, Q is the penalty weight for the DC motor speed error, R is the penalty weight for the input voltage, and r is the reference value at time $k+i$. In this cost function, the first term reduces the difference between the desired system output and the actual output value. The second term is then used to regulate the variation of the control signal. Next, to be solved using

the alternating direction method of multipliers (ADMM), (16) is transformed into a quadratic programming (QP) form as shown in (17).

$$J = \left(W - R_{ref}\right)^T Q\left(W - R_{ref}\right) + U^T RU \tag{17}$$

$$W = \phi X[k] + \Gamma U \tag{18}$$

Where W is the prediction vector of the motor speed over N_p steps ahead, which is generally represented in (18), R_{ref} is the reference speed vector at each prediction step, Q is the diagonal weight matrix for the motor speed prediction error, U is the vector of control input sequences, and R is the diagonal weight matrix for the input voltage. The result of substituting (17) into (18) is shown in (19).

$$J = \left(\phi X[k] + \Gamma U - R_{ref}\right)^T Q\left(\phi X[k] + \Gamma U - R_{ref}\right) + U^T RU \tag{19}$$

Where ϕ is the matrix that represents how the initial state influences the predicted DC motor speed over N_p steps ahead, and Γ is the matrix that represents how the sequence of control signals influences the predicted DC motor speed over N_p steps ahead.

The optimization problem can then be represented in the standard quadratic programming form, which is subsequently modified by adding the augmented lagrangian so that ADMM can efficiently perform separate solutions between the primal and dual, as shown in (20).

$$J = \min_U \ \frac{1}{2}U^T HU + f^T U + \frac{\rho}{2}\left\|U - Z + \frac{\lambda}{\rho}\right\|^2 + constant \tag{20}$$

Where the control signal vector is the primal variable, ρ is the ADMM penalty weight, Z is the auxiliary variable in ADMM, and λ is the dual variable. The constant term at the end can be ignored since it does not depend on the primal. The hessian matrix (H) is shown in (21), and the linear vector (f) is shown in (22).

$$H = 2(\Gamma^T Q\Gamma + R) \tag{21}$$

$$f = 2\Gamma^T Q(\phi X[k] - R_{ref}) \tag{22}$$

To obtain the primal, (20) is differentiated with respect to U, as shown in (23). Next, the auxiliary variable (Z) is updated through a projection (clipping) operation onto the admissible constraint set. This step enforces the inequality constraint directly within the optimization process rather than applying simple output saturation. The constraint limits the motor speed prediction error to the range $-100 \le e \le e_{ref}$. Then, the dual variable for the next iteration is updated according to (24).

$$U = \frac{\rho \cdot Z - \lambda - f}{H + \rho} \tag{23}$$

$$\lambda^{k+1} = \lambda + \rho(U - Z) \tag{24}$$

3.3. FPGA Implementation

The MPC method for DC motor control must be designed efficiently enough to be implemented on a low-end FPGA such as the Lattice iCE40HX1K (Lattice iCEstick), while still maintaining computation time and control performance even when the system input signal is affected by EMI noise effects (false edges). In this paper, the FPGA resource optimization strategies employed include simplifying the MPC structure using a 16-bit fixed-point representation (Q13.3), performing model coefficient precomputation outside the FPGA, replacing division operations with equivalent multiplications, minimizing the prediction horizon, and employing fixed scalar penalty weights to

avoid matrix operations. Since the Lattice iCE40HX1K does not provide dedicated DSP blocks, all arithmetic operations were implemented using LUT-based (soft) logic structures.

The flow diagram of the MPC method on FPGA is shown in Fig. 4. The flow diagram summarizes the steps of the implemented MPC scheme, including initialization of MPC parameters and model coefficients, sensor data acquisition, motor speed prediction, prediction error computation, cost function evaluation, primal and dual variable updates, and implementation of optimal control signal in PWM format.

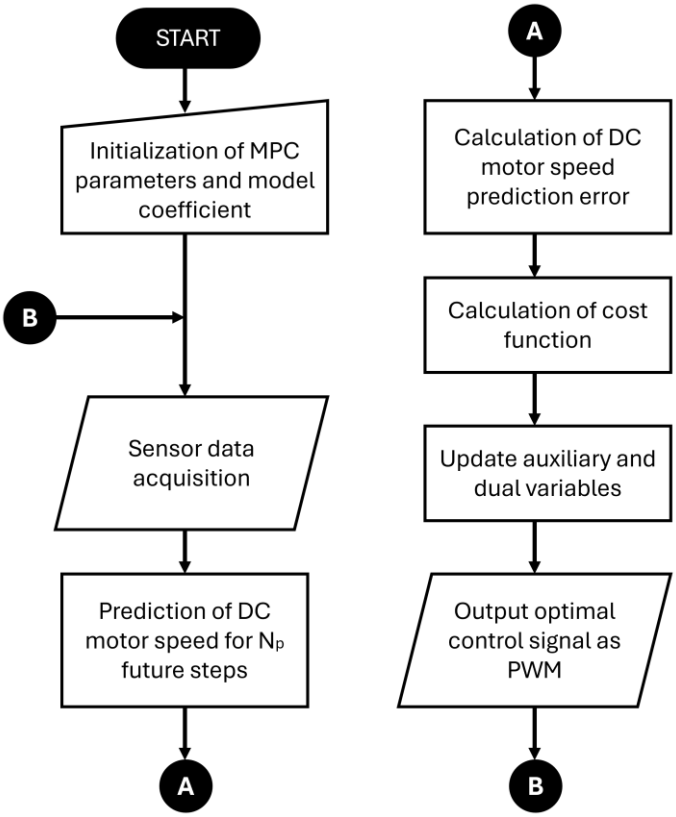

Fig. 4. Flow diagram of the MPC

The use of a 16-bit fixed-point representation (Q13.3) in the MPC implementation was chosen for several reasons. First, the fixed-point format has simpler arithmetic operations, enabling more efficient utilization of FPGA resources compared to floating-point format [61], [62], particularly on low-end FPGAs such as the Lattice iCE40HX1K. Second, the Q13.3 format provides a numerical range of approximately −4096 to 4095.875 with a resolution of 0.125, which is sufficient to represent all MPC computations without overflow while preserving control performance.

Furthermore, to reduce the computational load on the FPGA, the calculation of model coefficients is performed outside the FPGA. These coefficients include N_{max} in the mathematical model of EMI noise effects, as well as coefficients A and B in the mathematical model of the DC motor. The controller is based on a linear time-invariant (LTI) model, where the coefficients depend on the hall encoder and motor physical parameters. Since the motor's physical condition is relatively stable, the motor parameters remain constant, allowing these coefficients to be precomputed outside the FPGA. These coefficients are computed once outside the FPGA and implemented as fixed constants in the FPGA. In the case of significant load variations or parameter changes, the model coefficients must be recalculated accordingly.

Division is avoided in the FPGA implementation due to costly and is instead replaced by equivalent multiplication by precomputed reciprocals. This approach lowers resource consumption and shortens computation latency on the FPGA. Due to the same hardware constraints, a single

ADMM iteration is performed at each sampling step and the prediction horizon is intentionally limited to $N_p = 2$. A short horizon reduces the number of predicted states and optimization variables, thereby reducing overall computational complexity and resource usage.

In addition, to simplify computation on the FPGA, the MPC cost function employs fixed scalar penalty weights for the motor speed error (Q) and the control signal (R), instead of using full penalty matrices. This approach helps avoid complex matrix multiplication operations, thereby reducing the need for logic and memory resource. Although the penalty weights are scalar, selecting appropriate values can still maintain system performance in terms of computation time and control accuracy.

3.4. Experimental Methodology

The FPGA design was developed and synthesized using the open-source Apio framework (version 0.6.7). The toolchain integrates Yosys for logic synthesis, nextpnr-ice40 for place-and-route, and IceStorm for bitstream generation targeting the Lattice iCE40HX1K FPGA. All source files, build scripts, and configuration settings were maintained under version control to ensure full reproducibility of the experimental results. The hardware setup is shown in Fig. 5.

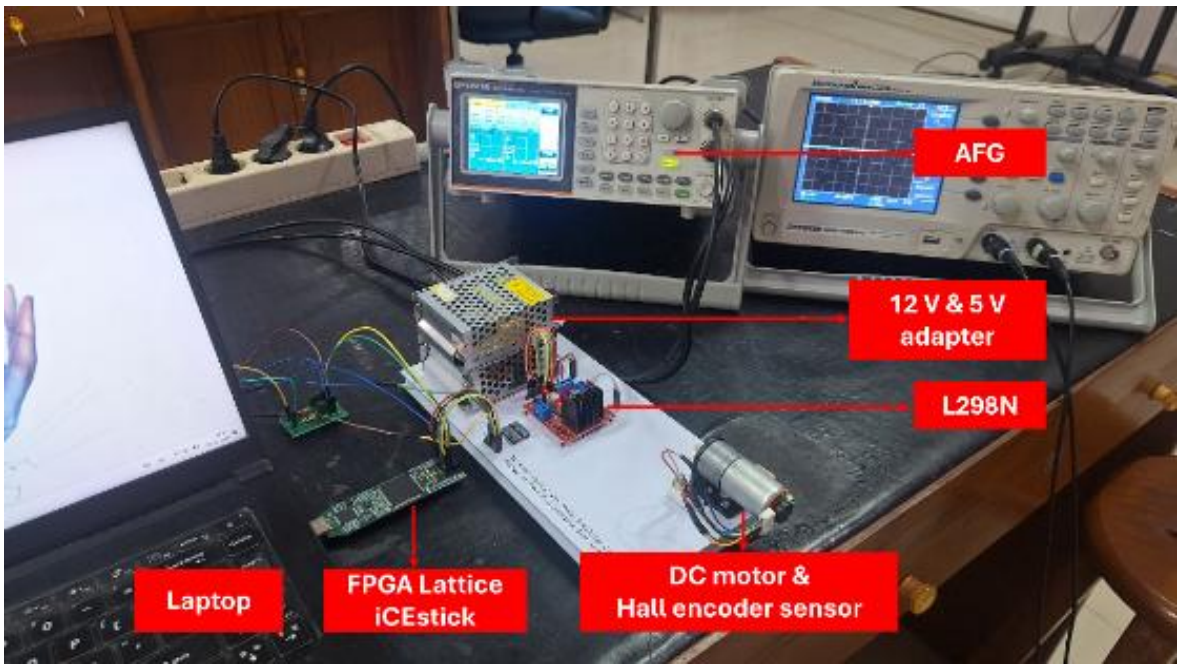

Fig. 5. Hardware setup of the system for testing the designed MPC

After synthesis, the optimized MPC design was evaluated on the Lattice iCE40HX1K FPGA (Lattice iCEstick) board through a DC motor speed control experiment as a proof-of-concept experiment. Three main performance variables were observed: total FPGA resource utilization, MPC computation time, and control performance. The MPC optimization strategy is considered successful if the total resource utilization remains within the resource limits of the Lattice iCE40HX1K FPGA, achieves a MPC computation time of no more than 1 ms (1,000 µs), and the steady-state control accuracy achieves a minimum of 80%. FPGA resource utilization was evaluated based on the synthesis report generated by the FPGA development software, while MPC computation time was evaluated by multiplying the total number of clock cycles required by the entire pipeline to complete one MPC iteration by the FPGA clock period (83.3 ns at 12 MHz). The pipeline consists of DC motor speed prediction, DC motor speed prediction error computation, optimal control signal computation, and updating of auxiliary and dual variables. Finally, system control accuracy performance was evaluated based on the average steady-state DC motor speed after transient effects had decayed, and compared with the reference value set at 104.72 rad/s (1,000 rpm), allowing a tolerance of 5%.

Testing was performed with three EMI noise effect scenarios in the form of false edges generated using the AFG output signal. The AFG was configured to generate a square-wave signal with adjustable frequencies of 2 kHz, 5 kHz, and 10 kHz, and a peak-to-peak amplitude of 2 V. The injected noise signal was conditioned through a passive RC filter network before being coupled to the system input to emulate false edges disturbances and control the noise characteristics. Each test scenario was repeated five times with fifteen data samples were recorded for each test. Fig. 6 (a) shows the system input signal under ideal conditions, while Fig. 6 (b)-(d) show the system input signals affected by EMI noise (false edges) at each corresponding frequency.

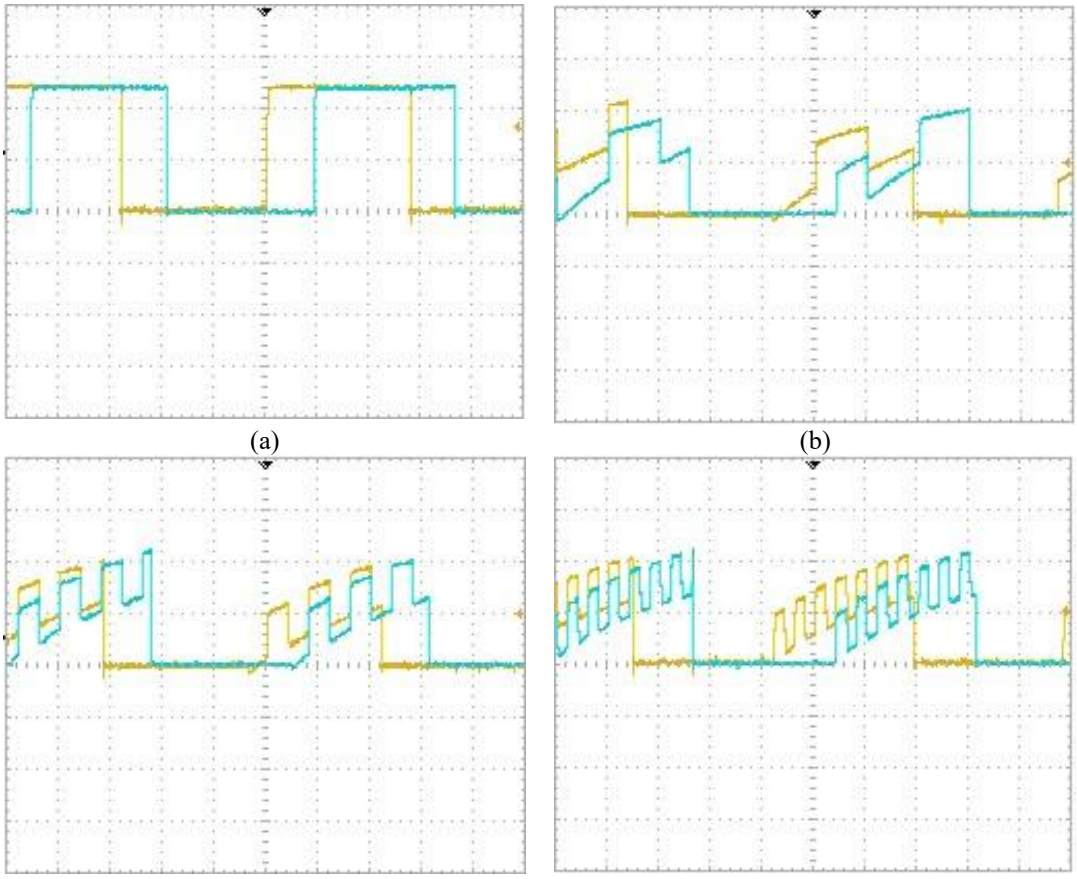

Fig. 6. System input signal under (a) ideal condition and affected by EMI noise at (b) 2 kHz, (c) 5 kHz, and (d) 10 kHz

4. Results and Discussion

This section summarizes the key findings of the proposed MPC implementation on a Lattice iCE40HX1K FPGA. Three main performance variables were observed: total FPGA resource utilization, MPC computation time, and control performance. The results demonstrate that the controller can be realized within the tight logic and memory constraints of the Lattice iCE40HX1K FPGA while maintaining real-time computation time and stable control performance. This indicates that MPC implementation can be extended to more accessible and resource-limited platforms through appropriate optimization strategies. In line with this evaluation scope, dynamic power consumption was not explicitly measured, as energy optimization was beyond the scope of the present study.

4.1. FPGA Resource Utilization

The report on total FPGA resource utilization for the implementation of MPC in the DC motor control is shown in Table 1. The data include the usage of input/output blocks (SB_IO), global buffers (SB_GB), warm boot (SB_WARMBOOT), phase-locked loop (ICESTROM_PLL), internal memory

(ICESTROM_RAM), and logic cells (ICESTROM_LC). Based on the table, the utilization of resources remains within the available capacity of the Lattice iCE40HX1K FPGA, with logic cell usage has reached 99%. While the system operates successfully under this condition, the extremely high utilization introduces critical vulnerability, leaving virtually zero margin for timing closure adjustments, manufacturing variations, minor design modifications, or future design expansion. Therefore, the design operates at the practical limit of the selected FPGA.

Table 1. FPGA resource utilization

Resource	Used	Percentage
SB_IO	7/112	6%
SB_GB	5/8	62%
SB_WARMBOOT	0/1	0%
ICESTROM_PLL	0/1	0%
ICESTROM_RAM	0/16	0%
ICESTROM_LC	1272/1280	99%

The heaviest load comes from the computation of the optimal control signal using the ADMM method, which requires arithmetic operations such as multiplication, addition, and subtraction, all of which are executed using the FPGA's logic and memory. Although full matrix multiplications have been eliminated to reduce complexity, the computation process still demands significant resources because a series of scalar arithmetic operations representing matrix elements. In addition, the processes of sensor data acquisition and UART communication for monitoring the DC motor speed also consume a considerable amount of logic and memory, since the period calculation, signal accumulation per period, and comparator functions for reset and synchronization are performed directly within the FPGA.

The relatively high utilization of global buffers indicates that the designed MPC architecture requires a considerable number of signals, in addition to the main clock, with large fanout. As a result, the FPGA synthesis software automatically promotes these signals to global buffers. This utilization of global buffers serves as an optimization mechanism provided by the FPGA synthesis tool to reduce clock skew and ensure uniform signal distribution across all registers. The presence of many signals with large fanout is a consequence of the design architecture, which demands wide signal distribution to various logic blocks within the FPGA. Interestingly, these signals mainly originate from the sensor data acquisition module and the clock divider module used for synchronizing computation cycles and managing data transmission via UART, rather than from the MPC core itself. This is due to the requirement of both modules to widely distribute signals for sensor reading synchronization, timing generation, and serial communication control.

Meanwhile, several FPGA resources such as warm boot, phase-locked loop (PLL), and internal memory (RAM) were not utilized in this implementation. This is because the designed architecture only employs a single bitstream and does not require recovery or reloading mechanisms during system operation. Similarly, the PLL was not used since the design relies solely on the internal FPGA clock (12 MHz) without the need for frequency multiplication or division. Although some modules require lower frequencies, these were achieved using a clock divider module based on enable signals with manual counters, rather than relying on the PLL. The use of manual counters to generate enable signals was chosen to ensure that all logic operates under a single clock source, thereby avoiding potential clock domain crossing issues that could arise if multiple clock sources were used. Additionally, RAM was left unused since all computations and variable storage were performed directly using logic and memory.

4.2. MPC Computation Time

The number of clock cycles required in each pipeline stage for a single MPC iteration is shown in Table 2. One MPC iteration consists of several pipeline stages, namely motor speed prediction, motor speed prediction error computation, optimal control signal computation, and the adjustment of

auxiliary and dual variables. This iteration begins immediately after sensor acquisition and concludes with the update of the PWM output. The total latency of one MPC iteration is 22 clock cycles. The FPGA frequency is 12 MHz, which is equivalent to a clock period of 83.3 ns. Accordingly, the overall execution time of one MPC iteration is 1.83 μs (22 × 83.3 ns). Given that the maximum allowable MPC computation time is 1 ms, the measured latency of 1.83 μs offers a substantial timing margin of roughly 546 times, ensuring real-time operation.

It can be observed that the optimal control signal computation stage requires the largest number of clock cycles, while the motor speed prediction and prediction error computation stages require the fewest. The variation in clock cycle requirements across pipeline stages corresponds to the complexity of the logic and memory resources used. Stages that require a large number of clock cycles generally demand more logic and memory, whereas stages with fewer clock cycles are relatively lighter in resource usage.

Table 2. MPC computation time

Pipeline Stage	Clock Cycle
DC motor speed prediction	2
DC motor speed prediction error computation	2
Optimal control signal computation	12
Updating auxiliary and dual variables	6

4.3. Control System Performance

The experimental results of the FPGA-based MPC implementation for DC motor speed control under three EMI noise effect (false edge) scenarios, 2 kHz, 5 kHz, and 10 kHz, are shown in Table 3. The reference speed was set at 104.72 rad/s (1,000 rpm) with a tolerance of 5%. To simulate the effect of EMI noise, false-edge disturbances were applied to the encoder signal during operation. Each test scenario was repeated five times with fifteen data samples were recorded for each test. The DC motor speed graph for each test scenario is shown in Fig. 7, where the dashed lines represent the ±5% tolerance limits around the reference speed.

At noise frequencies of 2 kHz and 5 kHz, the DC motor speed experienced a significant decrease immediately after the noise was applied, with a larger deviation observed at 5 kHz than at 2 kHz. The average measured speed dropped by approximately 86% at 2 kHz and 52% at 5 kHz relative to the reference value. This occurs because low-frequency noise produces long-duration false edges that occasionally reduce the number of detected encoder pulses. In this case, the implemented EMI noise effect model applies an upper bound through a clipping mechanism to prevent excessive overshoot but does not impose a lower bound on the minimum pulse count. Consequently, missing pulses caused by low-frequency noise directly lead to underestimated speed measurements. The lower-bound check was omitted because its implementation would require additional logic and memory resources that are not available on the Lattice iCE40HX1K FPGA. As a result, the system temporarily interprets the motor as rotating more slowly, which leads to performance degradation.

The standard deviation at 5 kHz (3.07) is larger than that at 2 kHz (0.70) because missing pulses at 2 kHz occur more consistently, producing a relatively uniform reduction in the measured speed. In contrast, at 5 kHz, the number of missing pulses varies more from one sampling period to another, causing larger fluctuations around the mean value. At a higher noise frequency of 10 kHz, the motor speed closely followed the reference value throughout the test, with a standard deviation of 2.85, which is lower than that observed at 5 kHz. High-frequency noise generates short-duration false edges that tend to appear as additional pulses rather than missing ones. These excessive pulses are effectively clipped by the upper-bound clipping mechanism implemented in the EMI noise effect model. Because the clipping mechanism actively limits unrealistic pulse increments, the speed measurement remains valid and stable. Consequently, in the 10 kHz noise test, the designed MPC method was able to maintain stable and accurate control performance even when the system input was affected by EMI noise effects (false edges). These results also indicate that control performance does not systematically

improve as noise frequency increases. Instead, the system response depends on how the noise characteristics interact with the implemented clipping mechanism in the EMI noise effect model.

Based on the experimental results shown in Fig. 7, the designed MPC method maintained good control performance. During noise application, a noticeable speed drop occurred in the 2 kHz and 5 kHz scenarios, while no significant reduction was observed at 10 kHz. After the noise was removed, the system quickly returned toward the reference speed. The steady-state condition was restored within a relatively short duration, with an error below 2%. In all cases, overshoot was absent in the 2 kHz and 5 kHz scenarios, whereas the 10 kHz case exhibited only minimal overshoot.

Table 3. Test results of the MPC implementation on the FPGA for DC motor control with three test scenarios

Parameter	2 kHz	5 kHz	10 kHz
Average speed	104.22 rad/s	102.89 rad/s	102.87 rad/s
Average accuracy	99.52%	98.25%	98.23%
Standard deviation	0.70	3.07	2.85

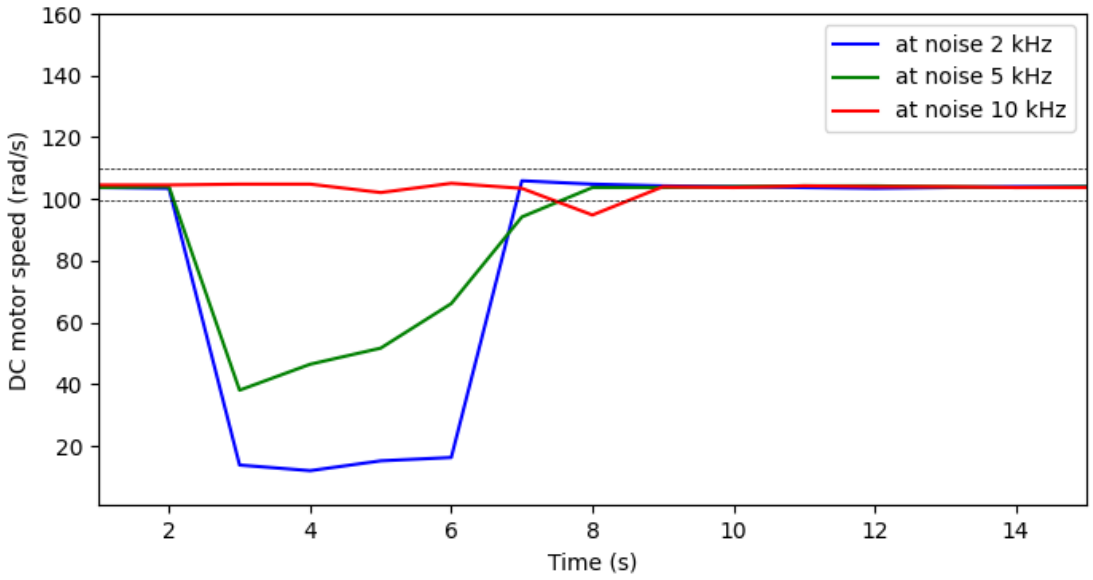

Fig. 7. The DC motor speed graph

Next to further evaluate the impact of the 16-bit fixed-point representation (Q13.3), a zoomed view of the steady-state response is shown in Fig. 8. Although small steady-state fluctuations are observed in the measured speed signal, no sustained periodic limit-cycle behavior is detected. The variation is primarily attributed to encoder resolution and measurement noise rather than the 3-bit fractional precision. The system remains stable and convergent, indicating that the selected fixed-point format provides sufficient numerical precision for the implemented MPC under the given hardware constraints.

The system's control accuracy across the three EMI noise effect conditions is shown in Fig. 9. The control system achieved more than 98% steady-state accuracy in all test scenarios, exceeding the predefined minimum requirement of 80%. This demonstrates that the proposed MPC design successfully meets the specified control performance objective. These results confirm that the optimized MPC implementation maintains control performance, even in the presence of EMI noise effects (false edges), while satisfying both the real-time constraint and the available FPGA resource limits. Nevertheless, the current implementation exhibits limitations in handling low-frequency noise that causes pulse loss, as no lower-bound check is implemented due to resource constraints on the Lattice iCE40HX1K. This trade-off represents the design decision to prioritize feasibility within strict hardware limits.

Future work may focus on investigating architectural redesign strategies to enable scalability toward more complex control systems, while preserving resource efficiency. In addition, exploring implementation on larger FPGA platforms may provide additional hardware margin for extending prediction horizons or incorporating more sophisticated disturbance-mitigation techniques. Further research may also include a comparative evaluation against similar resource-efficient MPC approaches to better quantify performance-resource trade-offs. Extending the proposed framework to multi-axis control systems represents another important direction, particularly for coordinated motion applications. Moreover, exploring alternative numerical representations and investigating the minimal viable hardware resource margins required for stable MPC operation could provide deeper insight into design optimization boundaries. The current results also demonstrate the potential for real-world applications, such as low-cost industrial motor drives and embedded motion control systems.

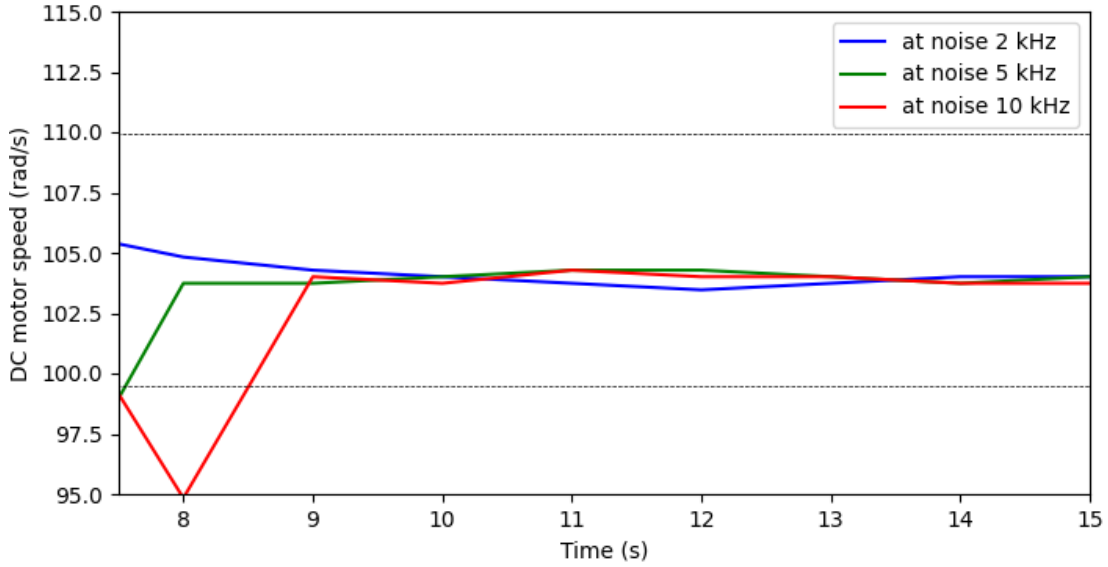

Fig. 8. Zoomed steady-state response of DC motor speed

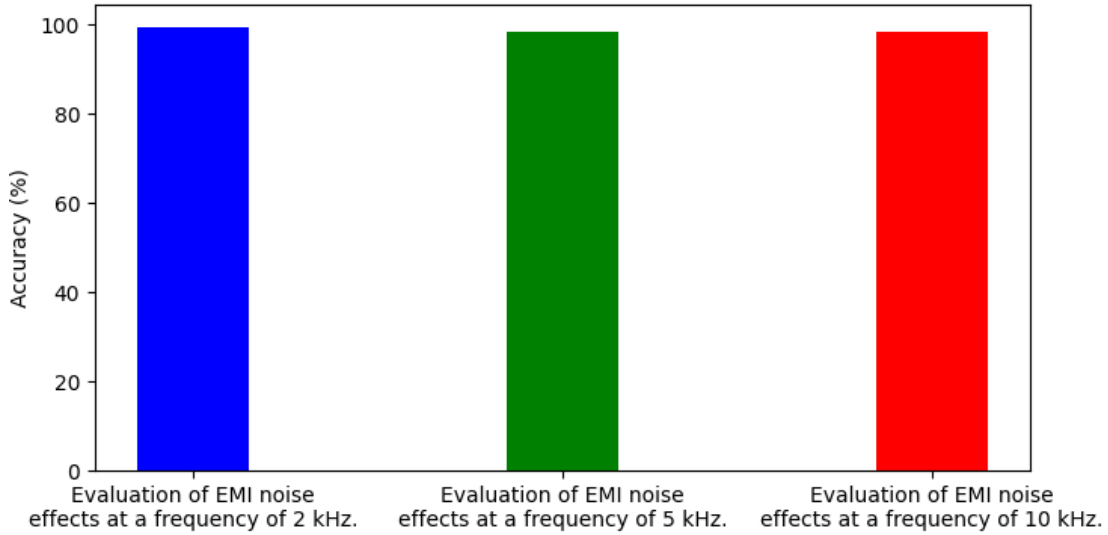

Fig. 9. Accuracy graph of the test results

5. Conclusion

This paper presents a resource-efficient implementation of MPC tailored for low-end FPGA (Lattice iCE40HX1K), validated using a DC motor speed control problem as a proof-of-concept case study with injected false-edge disturbances to emulate EMI noise. In contrast to prior studies that rely

on larger-resource FPGA platforms, this work focuses on optimizing MPC for implementation under a severely resource-limited platform. Key strategies to achieve resource-efficient MPC implementation include simplifying the MPC structure using a 16-bit fixed-point representation (Q13.3), performing model coefficient precomputation outside the FPGA, replacing division operations with equivalent multiplications, minimizing the prediction horizon ($N_p = 2$), and employing fixed scalar penalty weights to avoid matrix operations. The implemented MPC operates with limited fixed-point precision, a short prediction horizon, and a simplified EMI noise model effect due to the severe hardware constraints. The results demonstrate that the FPGA resource utilization remains within the available capacity of the Lattice iCE40HX1K, with logic cell usage reaching 99%. The implementation achieves a computation time of 1.83 µs, satisfying real-time requirements. Furthermore, it maintains DC motor speed under false-edge disturbances in the feedback signal, achieving steady-state accuracy above 98% and stable transient performance. Although the proposed MPC operates on the Lattice iCE40HX1K FPGA, the 99% logic cell utilization introduces a critical vulnerability, leaving virtually zero margin for manufacturing variations, minor design modifications, or future design expansion. Consequently, this severely limits direct scalability to more complex systems, such as PMSM with Field-Oriented Control (FOC), without architectural redesign or migration to a larger FPGA platform. Overall, the results confirm that MPC can operate effectively on low-end FPGA for low-complexity control. Future work may focus on investigating architectural redesign strategies to enable scalability toward more complex control systems, while preserving resource efficiency. In addition, exploring implementation on larger FPGA platforms may provide additional hardware margin for extending prediction horizons or incorporating more sophisticated disturbance-mitigation techniques. Further research may also include a comparative evaluation against similar resource-efficient MPC approaches to better quantify performance-resource trade-offs. Extending the proposed framework to multi-axis control systems represents another important direction, particularly for coordinated motion applications. Moreover, exploring alternative numerical representations and investigating the minimal viable hardware resource margins required for stable MPC operation could provide deeper insight into design optimization boundaries.

Beyond experimental validation, this work provides insight into how computationally intensive MPC algorithms can be systematically adapted to resource-limited FPGA platforms through resource-aware architectural design principles. Although demonstrated on a DC motor speed control problem, the proposed optimization strategies illustrate a general resource-constrained design methodology, rather than a case-specific adjustment. However, further architectural refinement is required to enable scalability toward higher-complexity systems.

Author Contribution: All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

Acknowledgment: The authors extend their appreciation to the Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada for funding this research work through the Department Grant 2026.

Funding: The authors extend their appreciation to the Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada for funding this research work through the Department Grant 2026.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] V. K. Singh, R. N. Tripathi, and T. Hanamoto, "Implementation Strategy for Resource Optimization of FPGA-Based Adaptive Finite Control Set-MPC Using XSG for a VSI System," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 9, no. 2, pp. 2066–2078, 2021, https://doi.org/10.1109/jestpe.2020.2999267.

[2] O. Gulbudak and M. Gokdag, "FPGA Implementation of Model Predictive Control for Driving Multi-Induction Motors," *2023 5th Global Power, Energy and Communication Conference (GPECOM)*, pp. 21–26, 2023, https://doi.org/10.1109/gpecom58364.2023.10175693.

[3] E. Zafra, S. Vazquez, T. Geyer, R. P. Aguilera, and L. G. Franquelo, "Long Prediction Horizon FCS-MPC for Power Converters and Drives," *IEEE Open Journal of the Industrial Electronics Society*, vol. 4, pp. 159–175, 2023, https://doi.org/10.1109/OJIES.2023.3272897.

[4] X. Zhang, R. Tao, Z. Dou, S. Qiu, J. Liu, and Y. Liu, "Fast two-vector-based model-free predictive current control of permanent magnet synchronous motor," *IET Electric Power Applications*, vol. 17, no. 2, 217-231, 2022, https://doi.org/10.1049/elp2.12257.

[5] Y. Luo, M. A. Awal, W. Yu, and I. Husain, "FPGA-Based High-Bandwidth Motor Emulator for Interior Permanent Magnet Machine Utilizing SiC Power Converter," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 9, no. 4, pp. 4340–4353, 2021, https://doi.org/10.1109/jestpe.2020.3015179.

[6] M. Leuer and J. Böcker, "Real-time implementation of an online Model Predictive Control for IPMSM using parallel computing on FPGA," *2014 International Power Electronics Conference (IPEC-Hiroshima 2014 - ECCE ASIA)*, pp. 346-350, 2014, https://doi.org/10.1109/IPEC.2014.6869605.

[7] I. Mishra, R. N. Tripathi, V. K. Singh, and T. Hanamoto, "Step-by-Step Development and Implementation of FS-MPC for a FPGA-Based PMSM Drive System," *Electronics*, vol. 10, no. 4, p. 395, 2021, https://doi.org/10.3390/electronics10040395.

[8] S. Mohammd Taher, A. Halvaei Niasar and S. Abbas Taher, "A New MPC-based Approach for Torque Ripple Reduction in BLDC Motor Drive," *2021 12th Power Electronics, Drive Systems, and Technologies Conference (PEDSTC)*, pp. 1-6, 2021, https://doi.org/10.1109/PEDSTC52094.2021.9405871.

[9] Y. Li, S. E. Li, X. Jia, S. Zeng, and Y. Wang, "FPGA accelerated model predictive control for autonomous driving," *Journal of Intelligent and Connected Vehicles*, vol. 5, 2, pp. 63-71, 2022, https://doi.org/10.1108/jicv-03-2021-0002.

[10] K. V. Ling, B. F. Wu, and J. M. Maciejowski, "Embedded Model Predictive Control (MPC) using a FPGA," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 15250–15255, 2008, https://doi.org/10.3182/20080706-5-kr-1001.02579.

[11] B. Plancher, S. M. Neuman, T. Bourgeat, S. Kuindersma, S. Devadas and V. J. Reddi, "Accelerating Robot Dynamics Gradients on a CPU, GPU, and FPGA," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2335-2342, 2021, https://doi.org/10.1109/LRA.2021.3057845.

[12] F. Xu, Z. Guo, H. Chen, D. Ji and T. Qu, "A Custom Parallel Hardware Architecture of Nonlinear Model-Predictive Control on FPGA," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 11, pp. 11569-11579, 2022, https://doi.org/10.1109/tie.2021.3118427.

[13] S. Lucia, D. Navarro, Ó. Lucía, P. Zometa and R. Findeisen, "Optimized FPGA Implementation of Model Predictive Control for Embedded Systems Using High-Level Synthesis Tool," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 1, pp. 137-145, 2018, https://doi.org/10.1109/TII.2017.2719940.

[14] A. Ravera *et al*., "Co-Design of a Controller and Its Digital Implementation: The MOBY-DIC2 Toolbox for Embedded Model Predictive Control," *IEEE Transactions on Control Systems Technology*, vol. 31, no. 6, pp. 2871–2878, 2023, https://doi.org/10.1109/tcst.2023.3254133.

[15] E. Zafra *et al*., "Efficient FPSoC Prototyping of FCS-MPC for Three-Phase Voltage Source Inverters," *Energies*, vol. 13, no. 5, pp. 1074–1074, 2020, https://doi.org/10.3390/en13051074.

[16] M. Jeong, M. Schoen and J. Biela, "When FPGAs Meet ADMM with High-level Synthesis (HLS): A Real-time Implementation of Long-horizon MPC for Power Electronic Systems," *2023 11th International Conference on Power Electronics and ECCE Asia (ICPE 2023 - ECCE Asia)*, pp. 1704-1711, 2023, https://doi.org/10.23919/ICPE2023-ECCEAsia54778.2023.10213796.

[17] B. Wang, J. Jiao and Z. Xue, "Implementation of Continuous Control Set Model Predictive Control Method for PMSM on FPGA," *IEEE Access*, vol. 11, pp. 12414-12425, 2023, https://doi.org/10.1109/access.2023.3241243.

[18] K. Nguyen, S. Schoedel, A. Alavilli, B. Plancher and Z. Manchester, "TinyMPC: Model-Predictive Control on Resource-Constrained Microcontrollers," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1-7, 2024, https://doi.org/10.1109/ICRA57147.2024.10610987.

[19] N. Lusardi, N. Corna, F. Garzetti, S. Salgaro and A. Geraci, "Cross-Talk Issues in Time Measurements," *IEEE Access*, vol. 9, pp. 129303-129318, 2021, https://doi.org/10.1109/ACCESS.2021.3113033.

[20] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza and M. Ryll, "Real-Time Neural MPC: Deep Learning Model Predictive Control for Quadrotors and Agile Robotic Platforms," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2397-2404, 2023, https://doi.org/10.1109/lra.2023.3246839.

[21] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model Predictive Contouring Control for Time-Optimal Quadrotor Flight," *IEEE Transactions on Robotics*, pp. 1–17, 2022, https://doi.org/10.1109/tro.2022.3173711.

[22] R. Grandia, F. Jenelten, S. Yang, Farbod Farshidian, and M. Hutter, "Perceptive Locomotion Through Nonlinear Model-Predictive Control," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3402–3421, 2023, https://doi.org/10.1109/tro.2023.3275384.

[23] E. Chajan, J. Schulte-Tigges, M. Reke, A. Ferrein, D. Matheis and T. Walter, "GPU based model-predictive path control for self-driving vehicles," *2021 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1243-1248, 2021, https://doi.org/10.1109/iv48863.2021.9575619.

[24] H. Gao, Z. Kan and K. Li, "Robust Lateral Trajectory Following Control of Unmanned Vehicle Based on Model Predictive Control," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 3, pp. 1278-1287, 2022, https://doi.org/10.1109/tmech.2021.3087605.

[25] Y. Lee, K. H. Choi, and K.-S. Kim, "GPU-Enabled Parallel Trajectory Optimization Framework for Safe Motion Planning of Autonomous Vehicles," *IEEE Robotics and Automation Letters*, vol. 9, no. 11, pp. 10407–10414, 2024, https://doi.org/10.1109/lra.2024.3471452.

[26] P. C. N. Verheijen, A. H. Derkani, Y. A. Agarwal, M. Lazar, and D. Goswami, "Multilevel parallel GPU implementation of SQP solvers for Nonlinear MPC," *2024 European Control Conference (ECC)*, pp. 2292–2298, 2024, https://doi.org/10.23919/ecc64448.2024.10590998.

[27] C. Jugade, D. Ingole, D. N. Sonawane, M. Kvasnica and J. Gustafson, "A Memory Efficient FPGA Implementation of Offset-Free Explicit Model Predictive Controller," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 6, pp. 2646-2657, 2022, https://doi.org/10.1109/tcst.2022.3168493.

[28] R. Ramakrishnan, A. K. V. Dev, A. S. Darshik, R. Chinchwadkar and M. Purnaprajna, "Demystifying Compression Techniques in CNNs: CPU, GPU and FPGA cross-platform analysis," *2021 34th International Conference on VLSI Design and 2021 20th International Conference on Embedded Systems (VLSID)*, pp. 240-245, 2021, https://doi.org/10.1109/vlsid51830.2021.00046.

[29] R. Miyagi, N. Takagi, S. Kinoshista, M. Oda and H. Takase, "Zytlebot : FPGA integrated ros-based autonomous mobile robot," *2021 International Conference on Field-Programmable Technology (ICFPT)*, pp. 1-4, 2021, https://doi.org/10.1109/icfpt52863.2021.9609883.

[30] A. K. Madsen and D. G. Perera, "Toward Composing Efficient FPGA-Based Hardware Accelerators for Physics-Based Model Predictive Control Smart Sensor for HEV Battery Cell Management," *IEEE Access*, vol. 11, pp. 106141-106171, 2023, https://doi.org/10.1109/access.2023.3319288.

[31] Y. Zhou, X. Jin, and T. Wang, "FPGA Implementation of A∗ Algorithm for Real-Time Path Planning," *International Journal of Reconfigurable Computing*, vol. 2020, pp. 1–11, 2020, https://doi.org/10.1155/2020/8896386.

[32] M. Sahin, "Designing MPC algorithms for velocity control of brushed DC motor and verification with SIL tests," *Automatika*, vol. 64, no. 3, pp. 399–407, 2023, https://doi.org/10.1080/00051144.2023.2170022.

[33] M. F. Elmorshedy, W. Xu, F. F. M. El-Sousy, M. R. Islam and A. A. Ahmed, "Recent Achievements in Model Predictive Control Techniques for Industrial Motor: A Comprehensive State-of-the-Art," *IEEE Access*, vol. 9, pp. 58170-58191, 2021, https://doi.org/10.1109/access.2021.3073020.

[34] S. Katayama, M. Murooka, and Y. Tazaki, "Model predictive control of legged and humanoid robots: models and algorithms," *Advanced Robotics*, vol. 37, no. 5, pp. 298-315, 2023, https://doi.org/10.1080/01691864.2023.2168134.

[35] R. Benotsmane and G. Kovács, "Optimization of Energy Consumption of Industrial Robots Using Classical PID and MPC Controllers," *Energies*, vol. 16, no. 8, p. 3499, 2023, https://doi.org/10.3390/en16083499.

[36] T. Zeng, A. Mohammad, A. G. Madrigal, Dragos Axinte, and M. Keedwell, "A Robust Human–Robot Collaborative Control Approach Based on Model Predictive Control," *IEEE Transactions on Industrial Electronics*, vol. 71, no. 7, pp. 7360–7369, 2023, https://doi.org/10.1109/tie.2023.3299046.

[37] H. Liu, S. Yan, Y. Shen, C. Li, Y. Zhang, and F. Hussain, "Model predictive control system based on direct yaw moment control for 4WID self-steering agriculture vehicle," *International Journal of Agricultural and Biological Engineering*, vol. 14, no. 2, pp. 175–181, 2021, https://doi.org/10.25165/j.ijabe.20211402.5283.

[38] G. Cavone, A. Bozza, R. Carli and M. Dotoli, "MPC-Based Process Control of Deep Drawing: An Industry 4.0 Case Study in Automotive," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1586-1598, 2022, https://doi.org/10.1109/tase.2022.3177362.

[39] D. Allahseh, J. Böttner, M. Al-Addous, and V. Lenz, "Advancements in hybrid heating systems for residential applications," *Energy Exploration & Exploitation*, vol. 43, no. 5, pp. 2221-2275, 2025, https://doi.org/10.1177/01445987251336405.

[40] O. Kabouri, M. Azeroual, A. Bagayogo, and H. El Markhi, "Optimized energy management in hybrid PV-wind microgrids: Fuzzy logic controller-based strategy with hydrogen system integration for DC bus stability," *Wind Engineering*, vol. 49, no. 6, pp. 1417–1438, 2025, https://doi.org/10.1177/0309524x251367997.

[41] M. Hamdi, O. Gtari, and M. Hazami, "Techno-environmental optimal sizing and dynamic behavior of a hybrid system feeding a large-scale SWRO desalination system: The case of Djerba Island, Tunisia," *Energy Exploration & Exploitation*, vol. 43, no. 1, pp. 378–409, 2024, https://doi.org/10.1177/01445987241285463.

[42] T. Brüdigam, M. Olbrich, D. Wollherr and M. Leibold, "Stochastic Model Predictive Control With a Safety Guarantee for Automated Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 22-36, 2023, https://doi.org/10.1109/TIV.2021.3074645.

[43] J. Pérez, A. Alabdo, J. Pomares, G. J. García, and F. Torres, "FPGA-based visual control system using dynamic perceptibility," *Robotics and Computer-Integrated Manufacturing*, vol. 41, pp. 13–22, 2016, https://doi.org/10.1016/j.rcim.2016.02.005.

[44] C. V. Cong, "Industrial robot arm controller based on programmable System-on-Chip device," *FME Transactions*, vol. 49, no. 4, pp. 1025–1034, 2021, https://doi.org/10.5937/fme2104025c.

[45] J. P. Queralta *et al*., "FPGA-based Architecture for a Low-Cost 3D Lidar Design and Implementation from Multiple Rotating 2D Lidars with ROS," *2019 IEEE SENSORS*, pp. 1-4, 2019, https://doi.org/10.1109/sensors43011.2019.8956928.

[46] J. Wang, M. Li, W. Jiang, Y. Huang, and R. Lin, "A Design of FPGA-Based Neural Network PID Controller for Motion Control System," *Sensors*, vol. 22, no. 3, pp. 889–889, 2022, https://doi.org/10.3390/s22030889.

[47] M. F. Elnaggar, A. Flah, M. Kaana, and D. Mourad, "A better quality of high-speed software encoder for an aerogenerator-based PMSM machine," *Wind Engineering*, vol. 49, no. 3, pp. 756–770, 2025, https://doi.org/10.1177/0309524x241310141.

[48] R. Shenbagalakshmi, S. K. Mittal, J. Subramaniyan, V. Vengatesan, D. Manikandan, and K. Ramaswamy, "Adaptive speed control of BLDC motors for enhanced electric vehicle performance using fuzzy logic," *Scientific Reports*, vol. 15, no. 1, pp. 1–22, 2025, https://doi.org/10.1038/s41598-025-90957-6.

[49] C. P. Y. Chan, "In-Pipe Maintenance Robot Using Spray-In-Place Pipe Technique for Long-Distance and Complex Pipe Environment," *Journal of Field Robotics*, vol. 42, no. 2, pp. 1226-1243, 2024, https://doi.org/10.1002/rob.22440.

[50] H. C. C. Michel, C. M. P. Braga, P. B. Costa, C. E. L. Silva, M. A. Câmara, and P. V. Trevizoli, "A servomotor-based alternative to traditional torque measurement systems: calibration and performance," *Measurement Science and Technology*, vol. 36, no. 10, p. 105901, 2025, https://doi.org/10.1088/1361-6501/ae131c.

[51] R. Gandhi, E. Sankararao, P. Mishra, and R. Roy, "A Novel Technique to Correct the Unwanted Zero Position Signal of Faulty Encoder for Accurate Speed and Position Estimation of PMSM Drive," *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, vol. 48, no. 1, pp. 289–301, 2023, https://doi.org/10.1007/s40998-023-00662-1.

[52] B.-D. Park, S.-J. Kim, J.-H. Moon, D.-W. Kang, S.-C. Go, and K.-H. Su, "Study on Compensation Method of Encoder Pulse Errors for Permanent Magnet Synchronous Motor Control," *Mathematics*, vol. 12, no. 19, p. 3019, 2024, https://doi.org/10.3390/math12193019.

[53] Y. Leng *et al*., "Design and test of reel speed control system for soybean combine harvester in the strip intercropping mode," *Scientific Reports*, vol. 14, no. 1, 2024, https://doi.org/10.1038/s41598-024-73835-5.

[54] R. E. García-Chávez *et al*., "Sliding Mode and PI-Based Control for the SISO "Full-Bridge Buck Inverter–DC Motor" System Powered by Renewable Energy," *IEEE Access*, vol. 12, pp. 27399-27410, 2024, https://doi.org/10.1109/access.2024.3365136.

[55] P. V. Minh, T. D. Chuyen, D. Q. Du, and H. D. Co, "Development of position tracking electric drive system to control BLDC motor working in very low mode for industrial machine application," *International Journal of Power Electronics and Drive Systems/International Journal of Electrical and Computer Engineering*, vol. 14, no. 2, pp. 688–688, 2023, https://doi.org/10.11591/ijpeds.v14.i2.pp688-697.

[56] S. Ahmad, S. Aslam, S. Khalid, U. Shabbir and M. Qaiser, "Investigation of MPC for MIMO system in presence of both input and output constraints with relative parametric variation," *2023 International Multi-disciplinary Conference in Emerging Research Trends (IMCERT)*, pp. 1-5, 2023, https://doi.org/10.1109/imcert57083.2023.10075329.

[57] Y. Song, Q. Li, and M. Zhang, "Electromechanical coupling modeling and fractional-order control of the seeker stabilization platform," *Scientific Reports*, vol. 14, no. 1, 2024, https://doi.org/10.1038/s41598-024-73478-6.

[58] J. Yang, D. Hou, X. Sun and J. Zuo, "Force Equalization Control of Redundant Electromechanical Actuation System by FOPID and Current Feedforward," *IEEE Access*, vol. 11, pp. 109283-109293, 2023, https://doi.org/10.1109/access.2023.3320275.

[59] C. Yin, S. Wang, J. Gao, G. Xu, and J. Wu, "An improved adaptive position tracking strategy for automatic shift actuator with uncertain parameters," *Scientific Reports*, vol. 14, no. 1, 2024, https://doi.org/10.1038/s41598-024-59952-1.

[60] H. Jin, H. Xu, and S. Wang, "Design and test of electromechanical disc brake controller for mine hoist," *Measurement + control/Measurement and control*, vol. 55, no. 3–4, pp. 146–154, 2022, https://doi.org/10.1177/00202940221091270.

[61] S. Gerkšič and B. Pregelj, "Finite-word-length FPGA implementation of model predictive control for ITER resistive wall mode control," *Fusion Engineering and Design*, vol. 169, p. 112480, 2021, https://doi.org/10.1016/j.fusengdes.2021.112480.

[62] Y. Wang, M. Soltani, and D. M. A. Hussain, "Attitude stabilization of Marine Satellite Tracking Antenna using Model Predictive Control," *IFAC Journal of Systems and Control*, vol. 17, p. 100173, 2021, https://doi.org/10.1016/j.ifacsc.2021.100173.