IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 19 December 2022, accepted 27 January 2023, date of publication 31 January 2023, date of current version 9 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3241243

APPLIED RESEARCH

Implementation of Continuous Control Set Model Predictive Control Method for PMSM on FPGA

BOZHI WANG[1], (Graduate Student Member, IEEE), JIYE JIAO[1], AND ZIYANG XUE[2]
[1]College of Electronic Engineering, Xi'an University of Posts and Telecommunications, Xi'an 710121, China
[2]College of Computer Science, Xi'an University of Posts and Telecommunications, Xi'an 710121, China

Corresponding author: Jiye Jiao (jiaojiye@xupt.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61874087.

ABSTRACT The generalized predictive control (GPC-MPC) algorithm adopts an optimal control strategy, which requires online matrix inversion operation and is difficult to be applied to real-time control systems. Therefore, the permanent magnet synchronous motor (PMSM) drive experiments, using the GPC-MPC algorithm with constraints, are primarily performed in a simulation environment. This paper proposes a new matrix inversion and its circuit design method to complete the above experiments in an embedded environment. By using the calculation steps of matrix block decomposition and reinversion, the parallelism of matrix calculation and the regularity of storage address are improved, and the speed of matrix inversion is accelerated. Also, the model-based design (MBD) method is used to complete the design and verification of the rest of the algorithm, which speeds up the implementation and deployment of the algorithm. Finally, the GPC-MPC algorithm control experiments with current constraints are implemented on the Field programmable gate array (FPGA) experimental board. The experimental results show that the proposed design method has a good control effect and computational efficiency.

INDEX TERMS Matrix inverse, generalized predictive control (GPC-MPC), permanent magnet synchronous motor (PMSM), model-based design (MBD), field programmable gate array (FPGA).

I. INTRODUCTION

Permanent Magnet Synchronous Motors (PMSM) offer higher power density and reliability than conventional motors. As a result, they are widely used in electric drives. To improve the control performance of PMSM, scholars have proposed various control algorithms, such as Constant Voltage to Frequency Ratio Control, Vector Control, Direct Torque Control, and Model Predictive Control (MPC) [1], [2], [3], [4]. Among them, MPC is mainly used to achieve open-loop optimal system control by establishing a predictive model, rolling optimization, and error feedback. However, because of its computing complexity, it has a specific delay characteristic, which is challenging to cope with the control requirements that require a fast response. With the development of processor technology, the MPC method with the ability to perform model prediction and solve constrained problems online has received much attention and discussion from scholars.

To apply MPC algorithms to the control system of PMSM, the current research directions are divided into two categories, Continuous Control Set MPC algorithms (CCS-MPC) and Finite Control Set MPC algorithms (FCS-MPC), depending on the control constraints. CCS-MPC solves the Quadratic Programming (QP) problem with constraints online by designing the cost function, thereby synthesizing vector levels of arbitrary size and direction used to drive the PMSM, which has good dynamic performance [5]. However, the high computational cost of CCS-MPC makes it difficult to be widely used. Therefore Rodriguez et al. [6] proposed the FCS-MPC algorithm based on predictive control. Its main idea is to complete the PMSM control by selecting the better switching signal in the two-level voltage inverter set and applying it directly to the inverter circuit, which can achieve the desired control while reducing the computing complexity of the CCS-MPC algorithm. However, using a limited set of

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

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

switching signals causes the control system to produce large current and torque fluctuations.

To reduce the computational complexity of the CCS-MPC algorithm while the system can also obtain an excellent dynamic response. Researchers have proposed Explicit Model Predictive Control (EMPC) algorithms [7], [8], [9], [10], pairwise constraint set algorithms [11] and combinatorial optimization methods [12], etc. However, these methods for improving CCS-MPC are mainly studied and analyzed from the perspective of simplifying constraints and reducing the number of searches. The optimization and implementation methods for specific operations are less mentioned. With the development of circuit reconfigurability technology and optimization theory, the feasibility of GPC-MPC algorithms using online computation has been greatly improved. Analyzing the computational bottlenecks of the optimization algorithm, selecting appropriate design tools and optimization methods, and performing specific optimization of the algorithm computation can lead to several times improvement of the computational efficiency [13]. The PMSM single-loop MPC algorithm designed in this paper is characterized by few constraints and small changes in the objective function and constraints in a short time. Therefore, the ASM-based GPC-MPC method is chosen to complete the PMSM control algorithm design on FPGA [14].

The ASM algorithm transforms the optimization problem with constraints into a set update problem to be solved in different cases, and an unconstrained optimization problem. In the solution process, the update of the set only requires the judgment of simple conditions. However, the unconstrained optimization problem not only involves more matrix operations, but also requires repeating the computational process after each set update in the solution process to update the computational results, which is the most tedious step in the algorithm calculation and programming process. Therefore, improving the efficiency of matrix operations, especially the efficiency of matrix inverse operations, and reducing the latency of matrix operations is one of the main points considered in this paper.

Kumar et al. [15] implemented the simulation and solution of the inverse matrix on FPGA using the adjugate matrix method, but this method is suitable for the design of matrix inversion operations with small dimensions. When the matrix dimension increases, the design difficulty increases dramatically. Langhammer and Pasca [16] improved the QR decomposition inverse method to make it more suitable for hardware parallelized design and able to handle large dimensional matrices. The experimental results show that it has lower latency and relatively less logic resource usage. However, its designed system and method are complex and costly to implement. Chetan et al. [17] used MATLAB/Simulink tool to design Gauss Jordan hardware inverse architecture to complete the solution of the inverse matrix of order 25. However, there is no decomposition operation for the matrix, which makes the stability of the matrix inversion not guaranteed. Jin et al. [18], analyzed the shortcomings of the RNN model to complete the matrix inversion operation, and solved the convex constraint problem of dynamic matrix inversion. However, its design is laborious and more suitable for the super-scale matrix inversion problem.

The GPC-MPC control of PMSM generally models small and medium-sized matrices. Therefore, this paper optimizes the symmetric positive definite matrix inversion algorithm, proposes a hardware design structure for the ASM algorithm, and simplifies the hardware design difficulty by using the Model-Based Design (MBD) method. There are three key aspects to consider when designing the GPC-MPC algorithm based on the ASM method on FPGA:

1) Online solving speed enhancement.
2) Generation of ASM constraint sets and selection of the number of iterations.
3) Simplification of circuit design methods.

To apply the ASM based GPC-MPC method to control PMSM efficiently and conveniently. In the subsequent sections, the GPC-MPC algorithm is better implemented on FPGA. It drives PMSM by focusing on the above key steps and optimizing the design structure and design process for the algorithm specific features. Therefore, the main contributions of this paper are the following:

1) Based on the Strassen algorithm and the LDLT matrix decomposition method. A more concise form of positive definite symmetric matrix inversion algorithm is proposed, and the circuit design is discussed.
2) The design details of the ASM algorithm with a fixed number of iterations on FPGA are discussed.
3) Simplify complex control logic designs using the MBD method and verify the feasibility of the process through simulation.
4) Experimental implementation of ASM-based PMSM control of the GPC-MPC algorithm on FPGA.

The remainder of this paper is organized as follows: Section II briefly introduces the PMSM state-space model of the single-loop MPC algorithm. It also leads to the ASM algorithm and its solving process. Section III describes an improved positive definite symmetric matrix inversion method in detail. The feasibility of the algorithm was verified using MATLAB, and the computational performance of the algorithm is also tested. Then the matrix inverse circuit architecture design is analyzed and implemented in detail. Section IV analyzes the design points for implementing the ASM based GPC-MPC algorithm on FPGA. The method of completing model simulation and IP (Intellectual Property) core generation using the Simulink/HLS tool is introduced. The experimental results are shown and illustrated. Section V provides a review and summary of our work.

II. BACKGROUND AND MODEL

This section briefly introduces the PMSM single-loop MPC design model and the ASM solution steps. Details about the single-loop MPC algorithm, the PMSM current and speed

equations derivation, and its state space model are described in [19] and [20].

A. SINGLE-LOOP MODEL OF PMSM

In the ideal case, the current and mechanical equation of motion of a surface-mounted permanent magnet synchronous motor (SPMSM) in the rotating coordinate system is expressed in equation Eq. (1).

$$\begin{aligned}\frac{di_d}{dt} &= -\frac{R_s}{L_s}i_d + n_p i_q \omega_m + \frac{u_d}{L_s}\\ \frac{di_q}{dt} &= -\frac{R_s}{L_s}i_q - n_p i_d \omega_m - \frac{n_p \psi_f}{L_s}\omega_m + \frac{u_q}{L_s}\\ \frac{d\omega_m}{dt} &= -\frac{3n_p\psi_f}{2J_r}i_q - \frac{B_v}{J_r}\omega_m - \frac{T_L}{J_r}\end{aligned} \tag{1}$$

where u_d and u_q are stator voltages components of the d and q axes, respectively. i_d and i_q are the stator currents of the d and q axes, respectively. ψ_f is permanent magnet magnetic chain. ω_m is the rotor mechanical angular velocity of the motor. n_p is the number of pole pairs. R_s is stator phase resistance. L_s is stator phase inductance. J_r is the moment of rotational inertia. T_L is the load torque. B_v is the coefficient of viscous friction.

The single-loop MPC algorithm uses a vector control strategy with $i_d = 0$ to achieve approximate decoupling of the i_d-axis currents. At this point, the equation di_d/dt tends to zero. Therefore the di_d/dt term is omitted. The $d-q$-axis current loop di_q/dt and the velocity loop $d\omega_m/dt$ terms are combined to design the algorithm using a second-order model of the i_q-axis voltage and current velocity ω_m of the PMSM. The scheme structure is shown in Fig. 1.

By discretizing the above single-loop PMSM model, expanding it with new state variables, and expressing it in matrix form, we obtain the discretized PMSM single-loop state space expression Eq. (2).

$$\begin{aligned}x(k+1) &= Ax(k) + B\Delta u(k)\\ y(k) &= Cx(k)\end{aligned} \tag{2}$$

where

$$A = \begin{bmatrix} 1 - \frac{R_s T_s}{L_s} & -\frac{n_p \psi_f T_s}{L_s} & 0 \\ \frac{3n_p\psi_f T_s}{2J_r} & 1 - \frac{B_v T_s}{J_r} & 0 \\ \frac{3n_p\psi_f T_s}{2J_r} & 1 - \frac{B_v T_s}{J_r} & 1 \end{bmatrix}^T$$

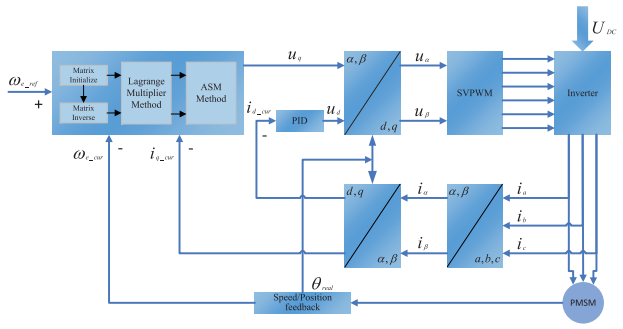


FIGURE 1. Single-loop GPC-MPC structure diagram.

$$x = \begin{bmatrix} \Delta i_q & \Delta\omega_m & \omega_m \end{bmatrix}^T$$
$$B = \begin{bmatrix} \frac{T_s}{L_s} & 0 & 0 \end{bmatrix}^T$$
$$C = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

T_s is the sampling frequency of the PMSM single-loop MPC algorithm. After the above generalization, the PMSM single-loop model required for the design is derived.

B. SINGLE-LOOP MPC MODEL OF PMSM

Suppose that the control quantity Δu does not change outside the control time domain. Let the prediction step be N_p and the control step be N_c. Using the moment k as the starting point for predicting the future dynamic changes of the system, the future state of the system is expected using the PMSM single-loop model, and the prediction model expression of the system is obtained Eq.(3).

$$Y_e = Y_s x(k) + Y_u \Delta U \tag{3}$$

where

$$Y_s = \begin{bmatrix} CA & CA^2 & \cdots & CA^{Np} \end{bmatrix}^T$$
$$Y_u = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{Np-1}B & CA^{Np-2}B & \cdots & CA^{Np-Nc}B \end{bmatrix}$$
$$\Delta U = \begin{bmatrix} \Delta u(k) & \Delta u(k+1) & \cdots & \Delta u(k+N_c-1) \end{bmatrix}^T$$

Suppose that the output error e_c is constant over the visibility horizon, it is expressed in the matrix form of Eq. (4).

$$E_c = \begin{bmatrix} e_c(k) & e_c(k) & \cdots & e_c(k) \end{bmatrix}^T \tag{4}$$

where $e_c(k) = y_c(k) - y_e(k)$. $y_c(k)$ is the current actual mechanical angular velocity and $y_e(k)$ is the mechanical angular velocity obtained from the prediction of the previous cycle.

Set the reference trace of the system speed as a smooth curve from the current actual speed $y_c(k)$ to the reference speed $y_r(k)$. The expression is in the form of a first-order exponential change as in Eq. (5).

$$y_l(k+i) = \sigma^i y_c(k) - (1-\sigma^i) y_r(k), i = 1, \ldots, N_p \tag{5}$$

where σ is the adjustment coefficient of the rate of change of the velocity of the reference trajectory. From equation (5), it can be seen that as σ increases, the initial slope of the set reference trajectory gradually decreases, and vice versa, the larger the initial slope of the reference trajectory. It is expressed as the vector form of Eq. (6).

$$Y_l = \begin{bmatrix} y_l(k+1) & y_l(k+2) & \cdots & y_l(k+N_p) \end{bmatrix}^T \tag{6}$$

To obtain excellent speed tracking performance while reducing energy losses, the output error and control magnitude are used as performance indicators, and the cost function is defined as Eq. (7).

$$J = \left\| \Gamma_y (Y_l - Y_e - E_c) \right\|^2 + \left\| \Gamma_u \Delta U \right\|^2 \tag{7}$$

Both Γ_y and Γ_u are the weight matrices of the output error and the control quantity, respectively, expressed as in Eq. (8).

$$\begin{aligned}\Gamma_y &= diag(\gamma_{y,1}, \gamma_{y,2}, \cdots, \gamma_{y,p})\\ \Gamma_u &= diag(\gamma_{u,1}, \gamma_{u,2}, \cdots, \gamma_{u,n})\end{aligned} \tag{8}$$

In addition, the output constraint on the i_q-axis current obtained from the next stage of the model prediction is expressed in Eq. (9).

$$i_{\min} \le i_q(k+1) \le i_{\max} \tag{9}$$

where $i_{\min}$ and $i_{\max}$ denote the constrained minimum and maximum currents. Substitution of the PMSM model and the current constraint into Eq. (7) yields the QP problem with constraints, expressed as Eq. (10).

$$\begin{aligned}&\min \frac{1}{2} z^T Q z + c^T z\\ &s.t.\ Pz \ge \xi\end{aligned} \tag{10}$$

where $Q = 2({Y_u}^T \Gamma_y Y_u + \Gamma_u)$ is a positive definite symmetric matrix, $z \in \mathbb{R}^n$ and is equivalent to ΔU.

$$\begin{aligned}c &= -2{Y_u}^T \Gamma_y (Y_l - Y_s x(k) - E_c)\\ P &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}\\ \xi &= \begin{bmatrix} (i_{\min} - \beta)\frac{L_s}{T_s} \\ -(i_{\max} - \beta)\frac{L_s}{T_s} \end{bmatrix}\\ \beta &= i_q(k) + \left(1 - \frac{R_s T_s}{L_s}\right)\Delta i_q(k) - \frac{n_p \psi_f T_s}{L_s}\Delta\omega_m(k)\end{aligned}$$

The open-loop optimal control is achieved by using the optimal solution of the above QP problem as the control input to the PMSM. The ASM is introduced in the following subsection to solve the QP problem.

C. ACTIVE SET METHOD FOR SINGLE-LOOP MPC DESIGN

To solve the QP problem with constraints obtained by simplification in the previous subsection, it is designed to be solved iteratively using ASM. Suppose that z_{cur} is a feasible point of the QP problem Eq. (10) and the corresponding active set is S_k, the iterative steps are as follows.

1) Select the initial value. Initialize the number of iterations $k = 0$.
2) Calculate the search direction and determine the set that meets the constraints of the equation. Solve for the minimal point z_k and calculate $z_k - z_{cur} = p_k$. Turn to step 3 when $p_k = 0$, otherwise turn to step 4.
3) Check the algorithm termination condition. By calculating the Lagrange multiplier λ_k. In the case of $\lambda_k \ge 0$, the global minima are obtained as z_k. Otherwise, the current constraint is excluded from the constraint set and turned to step 2 to calculate the new search direction.
4) Identify the step size p_k. Determine the step length based on the search direction and the constraints within the unconstrained set.

TABLE 1. Parameters and sample times for PMSM.

Rated Voltage U_{dc}	$24V$
Rated Current I_{dc}	$11.5A$
Rated Speed ω_m	$3000RPM$
PolePairs P_n	5
Stator inductance L_s	$0.46mH$
Stator resistance R_s	0.165Ω
Rated Flux Linkage ψ_f	$0.0119Wb$
Rotor Intertia J_r	$2.8^{-4}kgm^2$
Friction coefficient B_v	1.0^{-6}
Sample time T_s	$100us$

5) Update the constraint set. When the search step length is 1, the constraint set is unchanged; when the search step is less than 1, the constraint set is updated, and new constraints are added.
6) Update the number of iterations $k = k + 1$, and go to step 2.

In step 3, the Lagrange multiplier λ_k and the minimal value z_k can be calculated by equation Eq. (11).

$$\begin{aligned}z_k &= -Nc + E^T \xi\\ \lambda_k &= -Ec - M\xi\end{aligned} \tag{11}$$

where

$$\begin{aligned}N &= Q^{-1} - Q^{-1}P^T(PQ^{-1}P^T)^{-1}PQ^{-1}\\ E &= (PQ^{-1}P^T)^{-1}PQ^{-1}\\ M &= -(PQ^{-1}P^T)^{-1}\end{aligned}$$

The designs N_p and N_c are both 6 steps. By analyzing the model and constraints, it is concluded that $PQ^{-1}P^T$ is a positive definite symmetric matrix of order 2, which can be directly inverted. Q is a positive definite symmetric matrix of order N_p. In case N_p is greater than order 3, a complex concomitant matrix solution is required. Therefore, the operation of inverting the matrix Q is the most computationally intensive and challenging part of the ASM algorithm. In order to simplify the complexity of the numerical calculation, Keep the weight matrix Γ_y of the output error as a unit diagonal matrix, and adjust the speed error and the weight of the control input by changing the control weight matrix Γ_u. At this point Q and c are expressed as equation Eq. (12).

$$\begin{aligned}Q &= Q_c + 2\Gamma_u\\ c &= C_c(Y_l - Y_s x(k) - E_c)\end{aligned} \tag{12}$$

where $Q_c = 2{Y_u}^T \Gamma_y Y_u$, $C_c = -2{Y_u}^T \Gamma_y$ and Y_s are constant matrices when the PMSM parameters N_p and N_c have been determined.

According to the above model and calculation steps, the single-loop GPC-MPC model is built in Simulink. The data

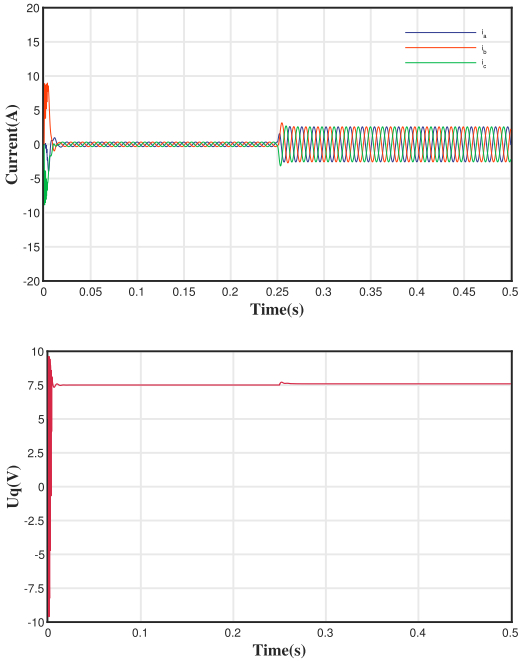


FIGURE 2. Single-loop MPC algorithm simulation for the sample current $i_{a,b,c}$ and control voltage U_q.

type adheres to the IEEE-754 double-precision floating-point standard. Simulation is performed to verify the correctness of the algorithm. The parameters of the PMSM used are shown in Table 1. The simulation results are shown in Fig. 2.

A load of $T_L = 0.2N$ was imposed at $0.25s$. The results show that the MPC single-loop control algorithm of PMSM has a good control effect. In the next section, the high-order matrix inversion method is discussed in detail, and the matrix inversion algorithm improved in this paper is introduced.

III. ALGORITHM DESIGN FOR COMPUTING THE INVERSE MATRIX

In order to solve the ASM, the inverse operation of the high-order positive definite symmetric matrix needs to be completed. In engineering, to speed up the solution of the inverse matrix of positive definite symmetric matrix and improve the computing stability, the operational method of trigonometric decomposition [21], [22], [23] and then the inverse is widely used. The main idea is to decompose the matrix to be inverse by solving the equation into the product of upper and lower triangular matrices, then inverse the decomposed triangular matrices and multiply them together to obtain the inverse of the matrix. It has been proved that the decomposition of the matrix inverse method is an excellent method for solving the inverse matrix. This section proposed the Strassen 2-order LDLT (S2-LDLT) inverse algorithm, which also has the characteristics of matrix decomposition. It is mainly used to solve the inverse of a positive definite symmetric matrix.

A. ALGORITHM DESIGN FOR INVERSE MATRIX SOLUTION

The S2-LDLT method decomposes the matrix with 2-order blocks as the smallest unit and then backtracks to calculate the inverse of the matrix.

For analogy between the S2-LDLT algorithm and the traditional triangular decomposition method of the positive definite symmetric matrix inversion process. Considering the common LDLT decomposition steps, the matrix to be solution is first expressed as a triangular matrix multiplied by a diagonal matrix, as shown in Eq. (13).

$$A_{org} = LDL^T \tag{13}$$

where A_{org} is a positive definite symmetry matrix of order n, L is the lower trigonometric matrix of order n, the main diagonal line is all 1, and D is the diagonal matrix of order n.

By multiplying the factors of the decomposed matrix and equating them with the original matrix, the values of the L and D matrices can be calculated by solving the equation, respectively. Then the inverse of the L and D matrices are calculated by Gauss-Jordan et al. Finally, the inverse of the original matrix can be found by multiplying the inverse of the triangular and diagonal matrices obtained by solving the equation. The detailed procedure is described in the paper [24].

Although the above matrix inversion method eliminates the calculation of the square root of the inverse of the Cholesky decomposition, it still requires complex matrix multiplication and division operations and lacks computational regularity. In order to simplify the solution steps, in this paper, the matrix A_{org} is regarded as a block matrix consisting of the 2-order matrix in the upper left corner of the original matrix, the $(n-2)$-order matrix in the lower right corner, and the remainder. Then the matrix A_{org} is decomposed by block in LDLT, expressed as Eq. (14), as shown at the bottom of the next page.

Where I denotes the unit matrix, O denotes the matrix with all zeros, and the subscripts of the block matrix denote the dimensions of the matrix rows and columns. Same as LDLT decomposition, multiply the triangular matrix and the diagonal matrix on the right side of Eq. (14) and make them equal to the corresponding values of the original matrix. The decomposed block matrix can be calculated by Eq. (15), as shown at the bottom of the next page.

$$\begin{aligned} \hat{R}_{2,2} &= \hat{A}_{2,2} \\ S_{n-2,2} &= F_{n-2,2}^T \hat{R}_{2,2}^{-1} \\ \tilde{R}_{n-2,n-2} &= \tilde{A}_{n-2,n-2} - S_{n-2,2} \hat{R}_{2,2} S_{2,n-2}^T \end{aligned} \tag{15}$$

To compute the inverse of matrix A_{org}, the inverse is taken simultaneously for both sides of the Eq. (14). In this way, the inverse matrix A_{org} can be expressed as Eq. (16), as shown at the bottom of the next page, by the inverse rule of the block matrix.

From Eq. (16), although the values of matrices $\hat{R}_{2,2}$, $\tilde{R}_{n-2,n-2}$ and $S_{n-2,2}$ are known. However, it is still necessary to solve the inverse matrices of $\hat{R}_{2,2}$ and $\tilde{R}_{n-2,n-2}$ to obtain the inverse matrix of the A_{org}. Therefore, the inverse matrix of $\hat{R}_{2,2}$ is calculated using the formula for the inverse of a matrix

of order 2, and the $\tilde{R}_{n-2,n-2}$ matrix is subjected to the same matrix decomposition as the A_{org} matrix until $\tilde{R}$ becomes a 2-order matrix positive definite symmetric matrix. At this point, a total of $\frac{n}{2}-1$ matrix decompositions are performed on the A_{org} matrix, and the decomposition process is expressed as Eq. (17).

$$\begin{aligned} \hat{A}_{2,2} &= \hat{R}_{2,2} \\ F_{2,n-2} &= \hat{R}_{2,2} S_{2,n-2} \\ \tilde{A}_{n-2,n-2} &= S_{n-2,2} \hat{R}_{2,2} S_{2,n-2}^{T} + decomp(\tilde{R}_{n-2,n-2}) \end{aligned} \tag{17}$$

$decomp(\tilde{R})$ denotes doing the same decomposition of the decomposed $(n-2)$-order $\tilde{R}$ matrix as the original matrix. Suppose that $\tilde{R}$ is the 2-order positive definite symmetric matrix obtained from the $(\frac{n}{2}-1)$th matrix decomposition, the inverse of $\tilde{R}$ is found using the 2-order matrix inverse formula. Then the inverse matrix of the matrix before each decomposition is computed recursively according to Eq. (16), until the inverse matrix of A_{org} is obtained.

It should be noted that before solving the inverse matrix, to keep the regularity of matrix decomposition and calculation, when the order of the original matrix is odd, it is extended to $(n+1)$-order, and make the main diagonal elements of the extended matrix of $(n+1)$-order to 1 and the rest elements are 0. After finishing the operation of solving the inverse matrix, the result of the n-order calculation is taken to obtain the result of the inverse of the original matrix.The algorithm is a special case of Strassen's inverse algorithm [25] with 2-order matrices as the minimum block and decomposition of the inverse according to the LDLT method, which is further explained and extended in this paper to make it more regular and suitable for implementation on hardware. Following the above steps, the pseudo-code of the S2-LDLT algorithm can be summarized as follows in Algorithm 1.

Algorithm 1 S2-LDLT Positive Definite Symmetric Matrix Inversion Algorithm

Input: Positive definite symmetric matrix A_{org} with dimension $n \times n$.
Initialize: $N \leftarrow n, i \leftarrow 1, j \leftarrow 1$
if $n \; mod \; 2 = 1$ **then**
  $A_{org}(i \rightarrow n+1, n+1) \leftarrow 0$
  $A_{org}(n+1, j \rightarrow n+1) \leftarrow 0$
  $A_{org}(n+1, n+1) \leftarrow 1$
  $N \leftarrow n+1$
end if
$\tilde{A}_{org}(\frac{N}{2}) \leftarrow A_{org}$
for $k = N \rightarrow 4 \; step \; -2$
  $\hat{R}_{2,2}(\frac{K}{2}) \leftarrow \hat{A}_{2,2}^{-1}(\frac{K}{2})$
  $S_{k-2,2}(\frac{K}{2}) \leftarrow -F_{2,k-2}^{T}(\frac{K}{2})\hat{R}_{2,2}(\frac{K}{2})$
  $\tilde{R}(\frac{K}{2}-1) \leftarrow \tilde{A}_{k-2,k-2}(\frac{K}{2}) + S_{k-2,2}(\frac{K}{2})F_{k-2,2}(\frac{K}{2})$
  Take $\tilde{R}(\frac{K}{2}-1) \; as \; A_{org}(\frac{K}{2}-1)$
end for
$\tilde{R}_{2,2}(1) \leftarrow \hat{A}_{2,2}^{-1}(1)$
for $k = 4 \rightarrow N \; step \; 2$
  $F_{k-2,2}^{T} \leftarrow \tilde{R}_{k-2,k-2}(\frac{K}{2}-1)S_{k-2,2}(\frac{K}{2})$
  $\hat{A}_{2,2} \leftarrow \hat{R}_{2,2}(\frac{K}{2}) + S_{2,k-2}^{T}(\frac{K}{2})F_{k-2,2}^{T}$
  $\tilde{A}_{k-2,k-2} \leftarrow \tilde{R}_{k-2,k-2}(\frac{K}{2}-1)$
  $\tilde{R}_{k-2,k-2}(\frac{K}{2}) \leftarrow \begin{bmatrix} \hat{A}_{2,2} & F_{2,k-2} \\ F_{k-2,2}^{T} & \tilde{A}_{k-2,k-2} \end{bmatrix}$
end for
  $A_{org}^{-1} \leftarrow \tilde{R}_{n,n}$
Output: The inverse of matrix A_{org}.

B. SIMULATION AND EVALUATION OF THE S2-LDLT ALGORITHM

The design generates 1000 random positive definite symmetric matrices each from order 2 to order 72 (step value of 2) in MATLAB 9.9. The LDLT, SPMI [26] and S2-LDLT methods are used for matrix inversion operations, respectively, and their average time spent is calculated. The computing complexity of the three algorithms is shown in Table 2 and the time consumed is shown in Fig. 3.

As shown in Fig. 3, the inverse matrix calculation speed of the S2-LDLT method is better than that of the LDLT and the SPMI method. In addition, since both the SPMI method and the S2-LDLT method use the block decomposition calculation method, they have faster computation speed compared with the traditional algorithm.

TABLE 2. Inverse algorithm time complexity comparison.

Algorithm	Complexity
$LDLT$	$O(n^3)$
$SPMI$	$O(n^3)$
$S2-LDLT$	$O(n^{2.807})$

The S2-LDLT algorithm uses a block recursive decomposition method to improve the speed of matrix decomposition while having certain computational regularity, which is in

$$\begin{bmatrix} \hat{A}_{2,2} & F_{2,n-2} \\ F_{n-2,2}^{T} & \tilde{A}_{n-2,n-2} \end{bmatrix} = \begin{bmatrix} I_{2,2} & O_{2,n-2} \\ S_{n-2,2} & I_{n-2,n-2} \end{bmatrix} \begin{bmatrix} \hat{R}_{2,2} & O_{2,n-2} \\ O_{n-2,2} & \tilde{R}_{n-2,n-2} \end{bmatrix} \begin{bmatrix} I_{n-2,n-2} & S_{2,n-2}^{T} \\ O_{n-2,2} & I_{2,2} \end{bmatrix} \tag{14}$$

$$\begin{bmatrix} \hat{A}_{2,2} & F_{2,n-2} \\ F_{n-2,2}^{T} & \tilde{A}_{n-2,n-2} \end{bmatrix}^{-1} = \begin{bmatrix} \hat{R}_{2,2}^{-1} + S_{2,n-2}^{T}\tilde{R}_{n-2,n-2}^{-1}S_{n-2,2} & -S_{2,n-2}^{T}\tilde{R}_{n-2,n-2}^{-1} \\ -\tilde{R}_{n-2,n-2}^{-1}S_{n-2,2} & \tilde{R}_{n-2,n-2}^{-1} \end{bmatrix} \tag{16}$$

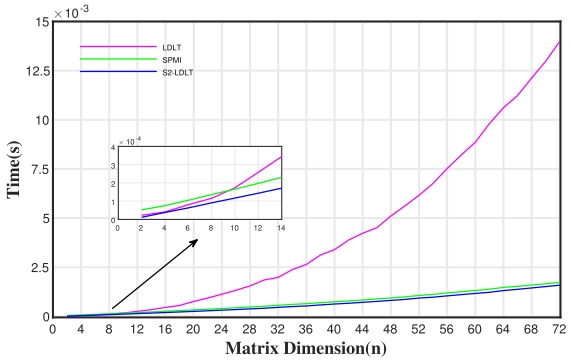


FIGURE 3. Comparison of the time consumed by the inverse matrix calculation.

line with the FPGA parallel computing and resource reuse. In addition, in the matrix back inversion stage, the inverse method of diagonal and triangular matrix blocks is used to transform the matrix inversion operation into recursive multiplication and addition of known values, which reduces the number of division operations in inverse substitution and improves the efficiency of matrix inversion operation.

C. HARDWARE ARCHITECTURE DESIGN OF S2-LDLT ALGORITHM

Before designing the architecture of the S2-LDLT matrix inversion algorithm, a brief analysis of the algorithm structure is first performed. This determines the appropriate pipeline length and balances the algorithm's computational resources and data parallelism. The design takes a 6-order matrix as an example and makes the following three analyses.

1) Data adventure of the pipeline: The depth of the pipeline is proportional to the data parallelism when there is no adventure. In the S2-LDLT inverse algorithm, the data adventure mainly occurs in the part of the matrix decomposition process to solve the 2-order matrix inverse. Also, due to the limitation of the matrix order, a 3-level structure is designed to complete the operation.
2) Reuse of resources: The S2-LDLT algorithm does not change its primary decomposition and backtracking form when decomposing or backtracking matrices of different orders. The design shares data computation units and uses a state machine to judge the order of the matrix and control the read/write addresses of data and data distribution to improve the reuse of resources.
3) Data parallelism: Data parallelism is mainly determined by the resources of the data computing unit. To complete the operation quickly with less resource usage, the data operation unit with two parallel degrees is designed concerning the minimum decomposition unit of the matrix inversion algorithm.

After the above analysis, combined with the model prediction step, the matrix inversion circuit designed in this paper adopts a three-stage operation structure: data distribution, data calculation, and data write-back. The data computation unit is designed as two parallel computation units, including two floating-point multiplication units and two floating-point addition units. A floating-point division unit is added to the second parallel computation unit, which is implemented in 3-stage, 3-stage, and 6-stage pipelines, respectively. The data distribution and write-back are mainly done by the state machine. In addition, the BRAM interface is used to read the original matrix data and write back the matrix inverse result. The overall architecture of the S2-LDLT algorithm is shown in Fig. 4.

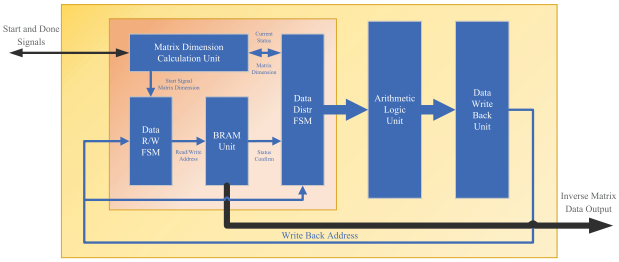


FIGURE 4. Hardware architecture of S2-LDLT algorithm.

As shown in Fig. 4, the core of the S2-LDLT algorithm architecture design is the matrix data read/write finite state machine and the data distribution finite state machine. The data read/write finite state machine is divided into two components depending on the different operational processes. The first half is responsible for controlling the matrix decomposition address, and the latter is controlling the matrix backtracking address. The data distribution finite state machine is responsible for assigning different data to the corresponding computational units. They collaborate to complete the matrix inversion operation. The state transfer diagram of the matrix decomposition address control part is shown in Fig. 5. The design of the S2-LDLT algorithm can be seen to be relatively simple, and the calculation of the access address is the most complicated part of the design.

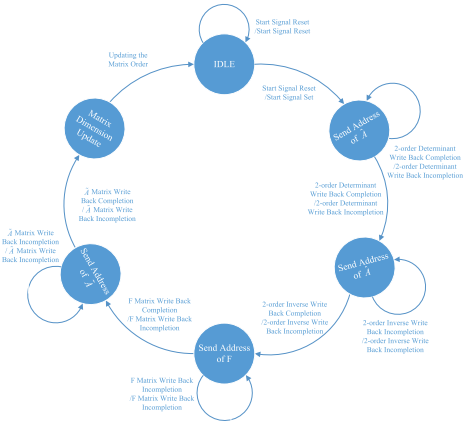


FIGURE 5. Matrix decomposition FSM of S2-LDLT algorithm.

Simulation of the above matrix inverse algorithm circuit design was verified and the resources were evaluated. The computation of the 6-order matrix inversion algorithm can be completed within 4us at 100MHz, while it takes about 120us to complete the same matrix inverse operation using the software. It also has a large speedup compared to the literature [26] which completes the 5-order matrix inversion operation in about 57us. The resource usage is shown in Table 3.

TABLE 3. S2-LDLT algorithm resource utilization.

XC7A200T	Resources	Design usage
LUT	8754	6.5%
FF	6200	2.3%
DSP48	8	1.08%
BRAM	3	0.82%

Finally, the matrix inversion algorithm circuit cooperates with the Simulink/HLS generated IP as a separate IP, the implementation details of which are described in the next section.

IV. SINGLE-LOOP MPC CIRCUIT DESIGN BASED ON PMSM

This section discusses the implementation of the ASM algorithm on FPGA. The sequential computation steps of the ASM algorithm are decomposed into 5 parallel computation processes for improving hardware parallelism. The choice of warm start and fixed iteration number are also discussed.

A. DESIGN OF SINGLE-LOOP MPC ACTIVE SET METHOD FOR PMSM

First, a combination of hardware and software was used to complete the ASM in [27]. Yet, more time is wasted on data transfer, so the data transfer between hardware and software should be avoided or reduced as much as possible when designing and dividing the ASM unit. In this paper, the Simulink/HLS toolbox is used to complete the algorithm design of ASM (except matrix inversion). It is also simulated and generates IP to implement the algorithm as a whole in hardware. Secondly, in this paper, only i_q-axis currents are constrained during the algorithm design, and two constraints are formed for any initial point. At most, one constraint is valid at a time, and only one search step needs to be adjusted to find the optimal value point on the active constraint. Therefore, the number of iterations is fixed to twice. Finally, the warm start method is used for initial point selection. The optimal value point obtained from the last search is used as the initial iteration point for the next ASM to reduce the number of the algorithm. The hardware design flow of the ASM algorithm based on the above three aspects is shown in Fig. 6.

Fig. 6 is divided into five steps from top to bottom to calculate the optimal value of ASM. In the first step, the input data is needed to initialize the QP problem. In the second step, the inverse of the Q matrix is calculated, which is used to support the Lagrange multiplier method and the iteration of the optimal value. In the third step, the optimal value points and Lagrange multipliers are solved using the formulas of the Lagrange multiplier method. In the fourth step, the active set and the new optimal value points are computed in parallel. Due to the parallel nature of the hardware circuit, the iterative process of the ASM algorithm in the FPGA does not correspond exactly to its algorithmic steps. Therefore the design uses conditional judgment and selection of outputs to accomplish the design goals. The separated parts of the design block diagram are the corresponding output conditions. At the end of the fourth step, the output selection is made by calculating and judging the incremental size of the step, completing a single iteration of the algorithm. In the fifth step, the number of iterations is checked to see if the set value is reached, thus returning to step three for another iteration or outputting the optimal value point. This not only divides the computational units for different functions but also reduces the computational latency, which is an undeniable advantage when designing hardware. In this design, the number of iterations is fixed at two, and the warm start is done by unit delay.

B. FPGA IMPLEMENTATION AND SIMULATION OF GPC-MPC ALGORITHM

By constructing the parallel computing logic according to the hardware implementation block diagram of ASM and combining it with the vector control, the single-loop GPC-MPC algorithm can be implemented on the FPGA.

In order to speed up the design of hardware circuits and reduce the difficulty of hardware circuit design, this paper adopts the design method of MBD. It uses Simulink/HLS to complete the design of the main computational flow of the GPC-MPC algorithm. The MBD method takes the algorithm model building as the core and pays more attention to the computational process of the algorithm itself, avoiding redundant code writing and debugging steps. It should be noted that, unlike the traditional MBD design method, the HLS module has a relatively single function, so when designing the algorithm, the original algorithm should be modeled and simulated using the Simulink basic module first. After obtaining the desired simulation results, the HLS module is used to replace them one by one until the design algorithm is completed using the HLS library. The design flow using the MBD method is shown in Fig. 7.

Because the algorithm is ultimately implemented on a hardware platform, the following two design points must be considered when building the ASM-based GPC-MPC algorithm using Simulink/HLS.

1) Serially executed algorithms vs. parallel computed modules: While attention to hardware parallelism is critical to HLS design, the algorithmic logic is serial.

<table>
<tr><td>STEP 1</td><td colspan="10">Input z_{cur}, Y_l and initialize Q, c, P, ξ, S_0 matrix</td></tr>
<tr><td>STEP 2</td><td colspan="10">Calculate the inverse matrix Q^{-1} using the S2-LDR algorithm</td></tr>
<tr><td>STEP 3</td><td colspan="10">Use Equation $z_k = -Nc + E^T\xi$, $\lambda_k = -Ec - M\xi$ to calculate the optimal value of the equation constraint</td></tr>
<tr><td rowspan="6">STEP 4</td><td colspan="5"></td><td colspan="5">$Den_{1,2} = P_{1,2} * z_k$ $\quad \bar{P}_{1,2} = \frac{(\xi_{1,2} - P_{1,2}) * z_{cur}}{P_{1,2} * z_k}$</td></tr>
<tr><td colspan="5"></td><td colspan="5">$\hat{P}_{1,2} = \min(1, \bar{P}_{1,2})$ $\quad \tilde{P} = \min(\hat{P}(1), \hat{P}(2))$</td></tr>
<tr><td colspan="2">$S_k(1)=1 \,\&\, S_k(2)=1$</td><td>$S_k(1)=1$</td><td>$S_k(2)=1$</td><td>$else$</td><td colspan="2">$S_k(1,2)=0 \,\&\, \tilde{P}\neq 1$ $\,\&Den(1,2)<0$</td><td colspan="2">$S_k(1)=0 \,\&\, \hat{P}(2)\neq 1$ $\,\&Den(1)<0$</td><td>$S_k(2)=0 \,\&\, \hat{P}(2)\neq 1$ $\,\&Den(2)<0$</td><td>$else$</td></tr>
<tr><td>$\lambda_k(1)<\lambda_k(2)$</td><td>$else$</td><td></td><td></td><td></td><td>$Den(1)<Den(2)$ $p_k=\tilde{P}$</td><td>$else$</td><td>$p_k=\hat{P}(1)$</td><td>$p_k=\hat{P}(2)$</td><td>$p_k=1$</td></tr>
<tr><td>$S_k(1)=0$</td><td>$S_k(2)=0$</td><td>$S_k(1)=0$</td><td>$S_k(2)=0$</td><td>$S_k(1,2)=S_k(1,2)$</td><td>$S_k(1)=1$</td><td>$S_k(2)=1$</td><td>$S_k(1)=1$</td><td>$S_k(2)=1$</td><td>$S_k(1,2)=S_k(1,2)$</td></tr>
<tr><td colspan="5"></td><td colspan="5">$z_{cur} = z_{cur} + p_k * z_k$</td></tr>
<tr><td colspan="10">Select output by $sum(z_k(i)^2) < 0, \quad i = 1, \cdots, 6;$</td></tr>
<tr><td>STEP 5</td><td colspan="10">Output the optimal value or optimize again from step 3 using the optimal value</td></tr>
</table>

FIGURE 6. Hardware parallel calculation step diagram of ASM algorithm.

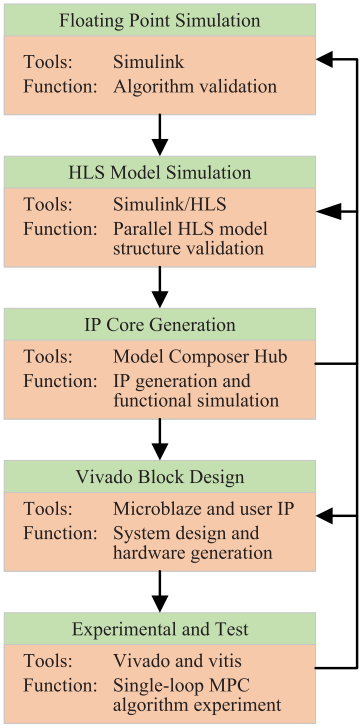


FIGURE 7. Design flow of MBD method based on Simulink/HLS.

Therefore, when designing HLS modules, more attention should be paid to the part of the algorithm that can be computed in parallel during the serial execution of the algorithm.

2) Data transfer between modules: Since the design is completed using both Verilog and Simulink/HLS, a unified data interface must be used to complete the data transfer between modules. After analysis, it can be seen that data transfer mainly occurs in the process of numerical transfer after solving the inverse matrix, and the amount of data is large, so the BRAM interface is designed to complete the storage and transit of the inverse matrix.

Based on the above design rules, an ASM-based PMSM single-loop MPC model suitable for implementation in FPGA is built using the Simulink/HLS toolbox. The data type adheres to the IEEE-754 single-precision floating-point standard. Unlike the HDL toolbox, which focuses on the design idea of circuit structure, the HLS toolbox focuses more on algorithm implementation with higher abstraction capability, which makes hardware design easier. The correctness of the algorithm is verified in Simulink, and the simulation waveforms are shown in Fig. 8.

By contrasting Fig. 8 with Fig. 2, it can be seen that using a fixed number of iterations and a warm start significantly reduces the fluctuations in the output of the MPC algorithm. Still, at the same time, the drive voltage of the motor increases considerably in a short period. In general, the algorithms designed using Simulink/HLS toolbox and Simulink base module use different computational accuracy, though. However, the simulation results are the same, and the expected design results are achieved. This also verifies the correctness of the design approach in this paper. Finally, the

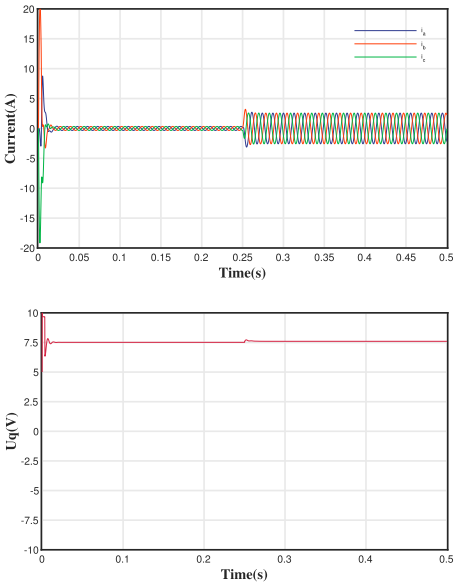


FIGURE 8. Single-loop MPC algorithm simulation of sampling current $i_{a,b,c}$ and control voltage U_q using two iterations and warm start in Simulink/HLS model.

simulation module is divided into four parts: matrix initialization (without matrix inversion module), Lagrange multiplier calculation, ASM iteration, and vector generation, and the independent IP of the AP interface is generated in turn, which is interconnected and called in FPGA Block Design with the matrix inversion algorithm IP designed in Section III-C.

TABLE 4. Resource utilization of the CCS-MPC algorithm.

XC7A200T	Resources	Design usage
LUT	33135	24.62%
FF	33720	12.53%
DSP48	132	17.84%
BRAM	16	4.50%

In Vivado 2020.2, after synthesis and routing, the algorithm resource usage is obtained, as shown in Table 4. The table shows that the computational circuits obtained by using Simulink/HLS design do not use overly redundant computational resources. It can even be used on FPGAs that are more resource constrained.

C. EXPERIMENTAL RESULTS OF FPGA BASED GPC-MPC ALGORITHM

The FPGA implementation of the GPC-MPC algorithm is completed. Then the XADC acquisition circuit and PWM generation circuit design are completed separately as the input and output interfaces of the peripheral circuits. It should be noted that the output PWM duty cycle is converted to unsigned int (32bit) format and amplified by a factor of 2^{15}, which helps to reduce the current ripple due to numerical errors. In this section, the design is evaluated and experimented with computational speed and control effect.

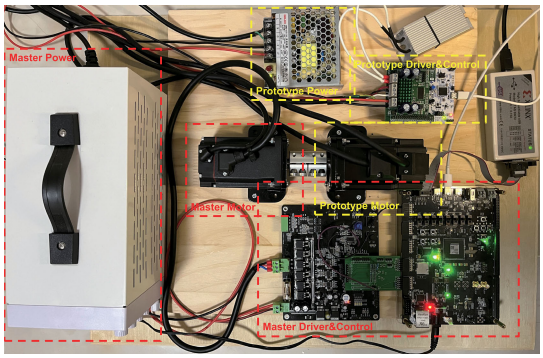
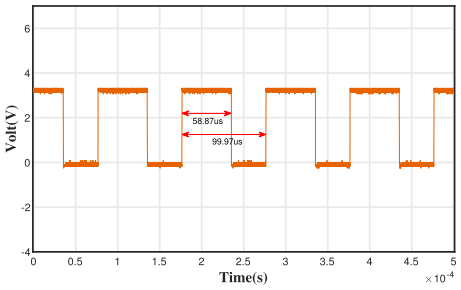


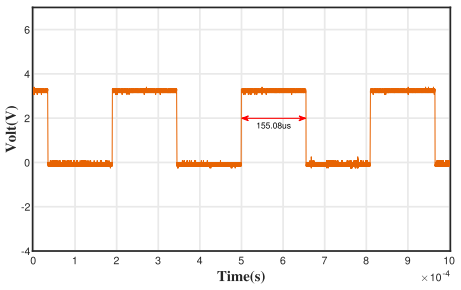
FIGURE 9. Experimental setup of FPGA-based PMSM drive system.

The experimental equipment for the GPC-MPC algorithm is shown in Fig. 9. It mainly consists of three parts: FPGA experimental board, motor driver board, and PMSM. This section uses experimental equipment to test and evaluate the computational speed and control effect.

The algorithm runs at a clock frequency of 100 MHz. The single-bit logic is designed to set the logic level high at the moment of triggering the XADC sampling and low when the calculation is completed to update the PWM duty cycle. The single-bit signal is output using the IO port and sampled using an oscilloscope. The calculation time consumption is shown in Fig. 10a. As a comparison, the software program using the same algorithm is implemented on an STM32G474 microcontroller with a level-1 optimization and with the floating point unit (FPU) turned on. The IO level is



(a) Calculated time consuming on FPGA



(b) Calculated time consuming on STM32

FIGURE 10. Time consumption of GPC-MPC algorithm (Contains time consumed by XADC/ADC acquisition and conversion).

flipped after each calculation is completed. The computation elapsed time is shown in Fig. 10b.

As shown in Fig. 10, the computation speed of the GPC-MPC algorithm is significantly improved compared to the microcontroller implementation by using a full hardware implementation. The calculation can be completed in about $60us$, which meets the design requirement of 10kHz sampling rate. Start the PMSM and make it stable at $500RPM$, then add $T_L = 0.1N$ load. The sampled current i_a are shown in Fig. 11.

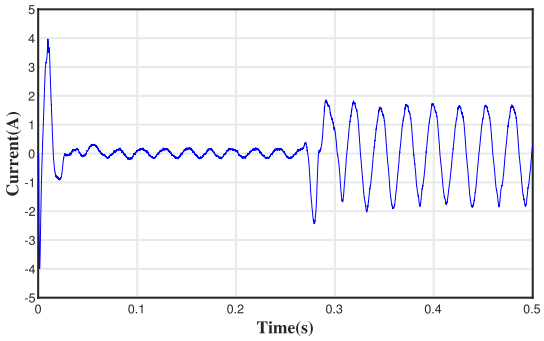


FIGURE 11. Sampling current i_a for single-loop MPC algorithm experiments implemented on FPGA.

V. REVIEW AND CONCLUSION

In this paper, the implementation of single-loop PMSM control method based on GPC-MPC algorithm on FPGA is discussed in detail. Firstly, the single-loop PMSM MPC algorithm model and the ASM solution procedure are introduced. The S2-LDLT algorithm is proposed to speed up the computation of the matrix inverse of a positive definite symmetric matrix in the ASM algorithm. Compared with the traditional matrix inversion algorithm, it is more efficient in computation and read/write address regularity, which is suitable for implementation on FPGA. Secondly, by analyzing the differences in hardware and software implementation methods of the ASM algorithm, the Simulink/HLS toolbox is used to establish a parallel GPC-MPC algorithm computational model suitable for implementation in hardware. Finally, the PMSM control experiments are completed on an FPGA experimental board using the proposed design method. The experimental results show that the computational efficiency of the proposed matrix inversion algorithm and MBD design method achieves more than 2.6 times the computational speed of the embedded software platform. The proposed method can be effectively applied to actual industrial production.

REFERENCES

[1] P. Perera, F. Blaabjerg, J. K. Pedersen, and P. Thogersen, "A sensorless, stable V/f control method for permanent-magnet synchronous motor drives," *IEEE Trans. Ind. Appl.*, vol. 39, no. 3, pp. 783–791, May/Jun. 2003.

[2] L. Jiao, Y. Luo, H. Jia, N. Cao, B. Yu, Y. Wang, Y. Liu, and X. Zhang, "Vector control strategy of PMSM servo system based on auto-disturbances rejection controller," in *Proc. IEEE 2nd Inf. Technol., Netw., Electron. Autom. Control Conf. (ITNEC)*, Dec. 2017, pp. 1332–1336.

[3] A. Nasr, C. Gu, X. Wang, G. Buticchi, S. Bozhko, and C. Gerada, "Torque-performance improvement for direct torque-controlled PMSM drives based on duty-ratio regulation," *IEEE Trans. Power Electron.*, vol. 37, no. 1, pp. 749–760, Jan. 2022.

[4] F. Niu, X. Chen, S. Huang, X. Huang, L. Wu, K. Li, and Y. Fang, "Model predictive current control with adaptive-adjusting timescales for PMSMs," *CES Trans. Electr. Mach. Syst.*, vol. 5, no. 2, pp. 108–117, Jun. 2021.

[5] A. A. Ahmed, B. K. Koh, and Y. Il Lee, "A comparison of finite control set and continuous control set model predictive control schemes for speed control of induction motors," *IEEE Trans. Ind. Informat.*, vol. 14, no. 4, pp. 1334–1346, Apr. 2018.

[6] J. Rodriguez, J. Pontt, C. Silva, M. Salgado, S. Rees, U. Ammann, P. Lezana, R. Huerta, and P. Cortes, "Predictive control of three-phase inverter," *Electron. Lett.*, vol. 40, pp. 561–563, May 2004.

[7] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, Jan. 2002.

[8] P. Tøndel, T. A. Johansen, and A. Bemporad, "An algorithm for multi-parametric quadratic programming and explicit MPC solutions," *Automatica*, vol. 39, no. 3, pp. 489–497, Mar. 2003.

[9] Z. Mynar, L. Vesely, and P. Vaclavek, "PMSM model predictive control with field-weakening implementation," *IEEE Trans. Ind. Electron.*, vol. 63, no. 8, pp. 5156–5166, Aug. 2016.

[10] S. Yuntao, X. Xiang, Z. Yuan, Z. Hengjie, and S. Dehui, "Design of explicit model predictive control for PMSM drive systems," in *Proc. 29th Chin. Control Decis. Conf. (CCDC)*, May 2017, pp. 7389–7395.

[11] O. Arpacik and M. M. Ankarali, "An efficient implementation of online model predictive control with field weakening operation in surface mounted PMSM," *IEEE Access*, vol. 9, pp. 167605–167614, 2021.

[12] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, Mar. 2010.

[13] M. Herceg, C. N. Jones, and M. Morari, "Dominant speed factors of active set methods for fast MPC," *Optim. Control Appl. Methods*, vol. 36, no. 5, pp. 608–627, Sep. 2015.

[14] M. S. K. Lau, S. P. Yue, K. V. Ling, and J. M. Maciejowski, "A comparison of interior point and active set methods for FPGA implementation of model predictive control," in *Proc. Eur. Control Conf. (ECC)*, Aug. 2009, pp. 156–161.

[15] G. A. Kumar, T. V. Subbareddy, B. M. Reddy, N. Raju, and V. Elamaran, "An approach to design a matrix inversion hardware module using FPGA," in *Proc. Int. Conf. Control, Instrum., Commun. Comput. Technol. (ICCICCT)*, Jul. 2014, pp. 87–90.

[16] M. Langhammer and B. Pasca, "High-performance QR decomposition for FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2018, pp. 183–188.

[17] S. Chetan, V. Lekshmi, S. Sudhakar, and J. Manikandan, "Design and implementation of a floating point matrix inversion module using model based programming," in *Proc. IEEE 16th India Council Int. Conf. (INDICON)*, Dec. 2019, pp. 1–4.

[18] L. Jin, S. Li, and B. Hu, "RNN models for dynamic matrix inversion: A control-theoretical perspective," *IEEE Trans. Ind. Informat.*, vol. 14, no. 1, pp. 189–199, Jan. 2018.

[19] M. Zhou, X. Liu, and S. Li, "Composite single-loop model predictive control design for PMSM servo system speed regulation based on disturbance observer," in *Proc. Chin. Control Decis. Conf. (CCDC)*, Aug. 2020, pp. 2886–2892.

[20] Z. Fan and S. Li, "Model predictive control method based on permanent magnet synchronous motor speed regulation system," in *Proc. 31st Chin. Control Conf.*, 2012, pp. 4412–4417.

[21] A. Krishnamoorthy and D. Menon, "Matrix inversion using Cholesky decomposition," in *Proc. Signal Process., Algorithms, Architectures, Arrangements, Appl. (SPA)*, 2013, pp. 70–72.

[22] I. P. Stanimirovic and M. B. Tasic, "Computation of generalized inverses by using the LDL^* decomposition," *Appl. Math. Lett.*, vol. 25, no. 3, pp. 526–531, 2012.

[23] G. R. Prabhu and J. S. Rani, "Fixed point pipelined architecture for QR decomposition," in *Proc. IEEE Int. Conf. Adv. Commun., Control Comput. Technol.*, May 2014, pp. 468–472.

[24] X.-W. Zhang, L. Zuo, M. Li, and J.-X. Guo, "High-throughput FPGA implementation of matrix inversion for control systems," *IEEE Trans. Ind. Electron.*, vol. 68, no. 7, pp. 6205–6216, Jul. 2021.

[25] M. Petkovic and P. Stanimirovic, "Block recursive computation of generalized inverses," *Electron. J. Linear Algebra*, vol. 26, pp. 394–405, Jan. 2013.
[26] Y. Xu, D. Li, Y. Xi, J. Lan, and T. Jiang, "An improved predictive controller on the FPGA by hardware matrix inversion," *IEEE Trans. Ind. Electron.*, vol. 65, no. 9, pp. 7395–7405, Sep. 2018.
[27] N. Yang, D. Li, J. Zhang, and Y. Xi, "Model predictive controller design and implementation on FPGA with application to motor servo system," *Control Eng. Pract.*, vol. 20, no. 11, pp. 1229–1235, Jun. 2012.

BOZHI WANG (Graduate Student Member, IEEE) received the B.S. degree in electronic information engineering from the Xi'an Innovation College, Yan'an University, in 2019. He is currently pursuing the master's degree with the Xi'an University of Posts and Telecommunications. His recent research interests include embedded processor design for control systems and low-power circuit design.

JIYE JIAO received the B.S. and M.S. degrees from the Xi'an University of Science and Technology, Xi'an, China, in 2000 and 2003, respectively, and the Ph.D. degree from the Xi'an University of Electronic Science and Technology, Xi'an, in 2013. In 2004, he worked as a Design Manager at Xi'an Advanced Microsystems Technology Company. In 2008, he worked at Teralane Semiconductor, as a Design Director. Since 2013, he has been working at Shanxi Province Specialized Integrated Circuit Design Engineering Center, Xi'an University of Posts and Telecommunications, where he is responsible for graphics processor chip design. His current research interests include computer architecture, high performance computing units, graphics processor design, and digital-analog hybrid circuit design.

ZIYANG XUE received the B.S. degree in information engineering and the IoT engineering from Xi'an University, in 2020. He is currently pursuing the master's degree in engineering with the Xi'an University of Posts and Telecommunications. His current research interests include sensor less control design of control systems and FPGA circuit design.