Attitude stabilization of Marine Satellite Tracking Antenna using Model Predictive Control

Wang, Yunlong; N. Soltani, Mohsen; Hussain, Dil Muhammad Akbar

Published in:
IFAC Journal of Systems and Control

DOI (link to publication from Publisher):
10.1016/j.ifacsc.2021.100173

Publication date:
2021

Document Version
Accepted author manuscript, peer reviewed version

Attitude Stabilization of Marine Satellite Tracking Antenna Using Model Predictive Control ⋆

Yunlong Wang * **Mohsen Soltani** **
Dil Muhammad Akbar Hussain ** **Roald M. Christensen** ***

* *Maersk Mc-Kinney Moeller Institute, University of Southern Denmark, 55 Campusvej, Odense, 5230, Denmark (e-mail: yun@mmmi.sdu.dk)*
** *Department of Energy Technology, Aalborg University, Esbjerg 6700, Denmark (e-mail: sms@et.aau.dk,akh@et.aau.dk)*
*** *SpaceCom A/S, Hobro 9500, Denmark, (e-mail: rc@spacecom.dk)*

Abstract: Attitude stabilization is a necessary function for a shipboard Marine Satellite Tracking Antenna (MSTA), which is responsible for making the antenna dish track the geostationary satellite in the presence of severe ship dynamics. A control scheme based on Model Predictive Control (MPC) is proposed to stabilize the antenna dish of a MSTA product and the detailed design procedure is stated from algorithm design to Hardware-in-the-loop (HIL) simulation. Since the stepper motor is used as the actuator, the MPC is proposed, for the first time, to handle the problem of velocity acceleration and deceleration of stepper motor, which can save lots of time on the design of velocity profile. The MPC algorithm is implemented in FPGA with three different data types. Besides conventional floating-point and fixed-point data types, a special data type, half-precision floating point, is also explored for the fist time. The comparison results of them are presented and analysed in terms of FPGA resource usage and algorithm execution time. The performance of proposed control scheme is validated in HIL simulation, which is creatively achieved in a low-cost System On Chip (SOC) FPGA. The HIL simulation results demonstrate that the proposed control scheme in fixed-point MPC can satisfy the requirements of MSTA product.

Keywords: Attitude stabilization, MSTA, MPC, FPGA, active set method, half-precision floating point.

1. INTRODUCTION

Marine Satellite Tracking Antenna (MSTA) is an important shipboard device for establishing communication between ships and geostationary satellites. Due to the ship dynamics caused by ocean wave, an attitude stabilization system is necessary for MSTA to keep the antenna dish pointing to the geostationary satellite with high precision. For the Ka-band MSTA, which is the latest technology of MSTA and is also the one in our research, the Root Mean Square (RMS) value of allowable tracking error is very small, only 0.2 *deg*. Moreover, the stepper motor is used as the actuator, whose velocity changing rate cannot be higher than a certain threshold, otherwise step missing will happen.

Many research works have been done about the attitude stabilization system of MSTA. In the paper of Wang et al. (2011), a Permanent Magnet Synchronous Motor (PMSM) servo system based on Digital Signal Processing (DSP) chip and vector control is designed. The DSP and Complex Programmable Logic Device (CPLD) are the main control chips, the software and hardware are designed and completed to realize the digital control of the satellite tracking antenna. In the paper of Tseng and Teo (1998), fuzzy logic is used to cope with the uncertainty and nonlinearity nature of the attitude control of shipboard tracking antenna. The simulations demonstrated the effectiveness of the design which has been implemented on some antennas. A fault tolerant control (FTC) system is proposed in the paper of Soltani et al. (2011) to deal with certain faults from communication system malfunction or signal blocking. The FTC system maintains the tracking functionality by employing proper control strategy. A robust fault diagnosis system is designed to supervise the FTC system. The employed fault diagnosis solution is able to estimate the faults for a class of nonlinear systems acting under external disturbances. In the paper of Ming et al. (2005), a method for constructing the model of MSTA is proposed and the model is established by experiments. According to the model, an H-infinity controller is designed and implemented to the antenna system. These research works have contributed to the design of attitude control system of MSTA, but none of them have considered the constraints in the system. In the attitude stabilization system of MSTA, the actuator is a stepper motor, whose rotation acceleration has to be limited to avoid missing

⋆ This work was supported by Innovation Fund Denmark under the prooject STAR^2 COM (Jnr.060-2013-3).

steps. Previously, research works focus on the design of velocity profile to avoid missing steps but no one had tried to apply MPC to solve this problem.

Recently, there is an increasing interest in the application of FPGA on computation acceleration (Tu et al. (2019); Zhang et al. (2019); Kosan et al. (2018); Lucia et al. (2018)), mainly due to its special hardware architecture which is totally different from CPU. The development of FPGA chips is very quickly in recent five years, which is mainly pushed by the strong need from deep learning and machine vision that require much larger computation capacity than MPC (Rahman et al. (2016); Dundar et al. (2017)). Many research works has been done about applying FPGA for accelerating the computation of MPC (Gulbudak and Santi (2016); Darba et al. (2016); Ramrez et al. (2014); Damiano et al. (2014); Ma et al. (2014); Jerez et al. (2014)). The comprehensive reviews of FPGA application in industrial control system can be found in the papers of (Monmasson and Cirstea (2007); Monmasson et al. (2011)). These previous researches have laid a solid foundation in the application of FPGA on MPC. However, there still exist some points that can be improved.

In this paper, the MPC algorithm based on active set method is designed and implemented in FPGA for the attitude stabilization of MSTA. The contributions of the present work are four-fold: (1) The MPC algorithm is applied to handle the velocity change of stepper motors in MSTA. (2) Half-precision floating-point data type is firstly proposed for MPC implementation. (3) The HIL simulation is achieved in a low-cost SOC FPGA chip, which demonstrates an inexpensive method for HIL simulation. (4) Three parallel fixed-point MPC controllers are implemented in the low cost SOC FPGA chip and are executed simultaneously.

The remainder of this paper is organized as follows: In Section 2, the attitude stabilization system of MSTA is introduced and the model for MPC design is presented. The design procedure of MPC algorithm is stated in Section 3. Then, the designed MPC is implemented in FPGA in Section 4 and a HIL simulation is made in Section 5 to verify performace of proposed control scheme. Lastly, Section 6 gives the conclusion and discusses future works.

2. CONTROL SYSTEM INTRODUCTION AND MODELLING

The MSTA under development is a commercial product, which has three rotational parts, that is, the elevation part, cross-elevation part and azimuth part. The attitude stabilization design of each part is independent and the controller design method of each part is the same. Hence, only the elevation part is used to show the design procedure of the attitude stabilization system.

2.1 Diagram of the Attitude Stabilization System

The diagram of the attitude stabilization system of the elevation part of MSTA is shown in Fig. 1.

In Fig. 1, the reference angle is determined by the position of the geostationary satellite that the MSTA needs to track

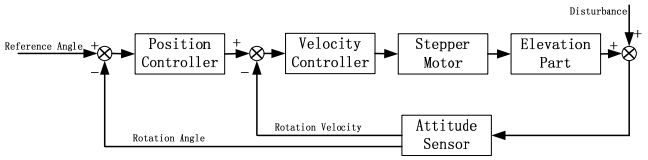

Fig. 1. Diagram of the attitude stabilization system of the elevation part

and can be considered as a constant value in a short period. There are two closed loops in this attitude stabilization system, inner loop and outer loop. The inner loop is the velocity loop, which is responsible to control the rotation velocity of the elevation part. And the outer loop is the position loop, which is responsible to control the rotation angle of the elevation part. The rotation angle and velocity can be obtained from an attitude sensor. The disturbance is from the rotation movements of ships caused by ocean waves.

The position controller is a PI controller. It is the design of the velocity controller (MPC) that will be focused on in this paper. The velocity controller is responsible for compensating most part of the disturbance from ships and the remaining parts is left to the position controller.

2.2 State Space Model For MPC

In the attitude stabilization system of elevation part, the stepper motor is the actuator and the elevation part is the load. For MPC design, the model of stepper motor has to be obtained first.

In previous formulation of model predictive control, finite impulse response (FIR) models, step response models, and transfer function models were favoured. In recent years, there is growing popularity of designing MPC based on state-space models (Wang (2009)). The main theoretical results of MPC related to stability come from a state space formulation, which can be used for both monovariable and multivariable processes and can be extended to nonlinear processes (Camacho and Alba (2007)).

The stepper motor is used in close-loop control system and its rotation velocity is obtained from a high-precision gyroscope mounted on the elevation part. By conducting experimental analysis, it is found that the velocity model of the stepper motor can be approximated as a second-order linear system, as shown below,

$$G_{vs}(s) = \frac{V(s)}{R(s)} = \frac{w_n^2}{s^2 + 2 \cdot \xi_v \cdot w_n \cdot s + w_n^2} \tag{1}$$

where $R(s)$ is the command rotation velocity and $V(s)$ is the actual rotation velocity of stepper motor shaft. $w_n = 27.3 rad/s$ is the undamped natural frequency and $\xi_v = 0.022$ is the damping ratio, which are obtained by applying model identification method on sampled experimental data.

The transfer function model in (1) is converted into continuous-time state-space model, and then discretized by Euler method to get discrete-time state space model, as shown in (2).

$$\boldsymbol{x}_d(k+1) = \boldsymbol{A}_d \cdot \boldsymbol{x}_d(k) + \boldsymbol{B}_d \cdot u_d(k) \quad (2)$$
$$y_d(k) = \boldsymbol{C}_d \cdot \boldsymbol{x}_d(k)$$

where Δt is the sampling interval, k is the current time step, $\boldsymbol{x}_d(k) = [x_1(k)\ x_2(k)]^T$ where $x_1(k)$ is the rotation velocity and $x_2(k)$ is the rotation acceleration. $\boldsymbol{A}_d = \exp(\boldsymbol{A}_c \cdot \Delta t) \approx \boldsymbol{I} + \boldsymbol{A}_c \cdot \Delta t$, where $\boldsymbol{A}_c$ is

$$\boldsymbol{A}_c = \begin{bmatrix} 0 & 1 \\ -w_n^2 & -2 \cdot \xi \cdot w_n \end{bmatrix},$$

and

$$\boldsymbol{B}_d = \begin{bmatrix} 0 \\ w_n^2 \end{bmatrix}, \boldsymbol{C}_d = [\,1\ 0\,].$$

The model in (2) is changed to make the MPC embed an integrator. The changed model is called augmented model, as shown below. The details about augmented model can be found in the paper of Wang (2009).

$$\boldsymbol{x}(k+1) = \boldsymbol{A} \cdot \boldsymbol{x}(k) + \boldsymbol{B} \cdot \Delta u(k) \quad (3)$$
$$y(k) = \boldsymbol{C} \cdot \boldsymbol{x}(k)$$

where $\boldsymbol{x}(k+1) = \begin{bmatrix} \boldsymbol{\Delta x}(k+1) \\ y(k+1) \end{bmatrix}$, $A = \begin{bmatrix} \boldsymbol{A_d} & \boldsymbol{o}_d^T \\ \boldsymbol{C}_d\boldsymbol{A}_d & 1 \end{bmatrix}$, $\boldsymbol{x}(k) = \begin{bmatrix} \Delta\boldsymbol{x}(k) \\ y(k) \end{bmatrix}$, $\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_d \\ \boldsymbol{C}_d\boldsymbol{B}_d \end{bmatrix}$, $\boldsymbol{C} = [\,\boldsymbol{o}_d\ 1\,]$, $\boldsymbol{o}_d = [\,0\ 0\,]$, $\Delta\boldsymbol{x}(k) = \boldsymbol{x}_d(k+1) - \boldsymbol{x}_d(k)$, $\Delta u(k) = u_d(k+1) - u_d(k)$. Note that, Δu is the change of controlled variable.

Equation (3) is the model that will be used for MPC design in Section 3.

3. MPC ALGORITHM

The pseudo-code of the MPC algorithm for velocity control loop in Fig. 1, is stated in Algorithm 1 and the detailed explanation of each line is given next. It should be note that this pseudo code is written in Maltab M language style.

3.1 Definition of MPC Problem

The MPC problem is defined in this section. Firstly, the following vectors are defined,

$$\boldsymbol{Y} = [\,y(k+1|k)\ y(k+2|k)\ \ldots\ y\,(k+N_p|k)]^T$$
$$\Delta\boldsymbol{U} = [\,\Delta u(k)\ \Delta u(k+1)\ \ldots\ \Delta u(k+N_c-1)\,]^T$$

where the elements of $\boldsymbol{Y}$ are the predicted outputs based on current state $\boldsymbol{x}(k)$. Thus they are also the predicted rotation velocity of the elevation part of MSTA. The elements of $\boldsymbol{\Delta U}$ are the increment of future control trajectory that can be obtained by solving convex optimization problem with constraints. N_p is the prediction horizon, and N_c is the control horizon.

The predicted output variables are,

$$\boldsymbol{Y} = \boldsymbol{F} \cdot \boldsymbol{x}(k) + \boldsymbol{\Phi} \cdot \boldsymbol{\Delta U} \quad (4)$$

where

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{CA}\ \boldsymbol{CA}^2\ \ldots\ \boldsymbol{CA}^{N_p} \end{bmatrix}^T \quad (5)$$

Algorithm 1: MPC controller algorithm

MPCControllerOneStep ($\boldsymbol{R}_s$,$\boldsymbol{x}_d(k)$,$y_d(k)$)
$\boldsymbol{x}(k) \leftarrow \begin{bmatrix} \boldsymbol{x}_d(k) - \boldsymbol{x}_d(k-1) \\ y_d(k) \end{bmatrix}$
$\boldsymbol{\Delta U} \leftarrow (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \bar{\boldsymbol{R}})^{-1}\boldsymbol{\Phi}^T(\boldsymbol{R}_s - \boldsymbol{F} \cdot \boldsymbol{x}(k_i))$
$N_{cv} \leftarrow 0$
for $i = 1, ..., 2 \cdot N_c$ **do**
  if $(\boldsymbol{M}(i,:) \cdot \boldsymbol{\Delta U} > \boldsymbol{\gamma}(i))$ **then**
    $N_{cv} \leftarrow N_{cv} + 1$
  end
end
if $N_{cv} > 0$ **then**
  $\boldsymbol{K} \leftarrow \boldsymbol{M} \cdot (-\boldsymbol{\Delta U}) + \boldsymbol{\gamma}$
  for $cy = 1, ..., N_1$ **do**
    for $i = 1, ..., N_c$ **do**
      $s1 \leftarrow 0$
      $s2 \leftarrow 0$
      for $j = 1, ..., i-1$ **do**
        $s1 \leftarrow s1 + \boldsymbol{H}(i,j) \cdot \lambda(j)$
      end
      for $j = i+1, ..., N_c$ **do**
        $s2 \leftarrow s2 + \boldsymbol{H}(i,j) \cdot \lambda_p(j)$
      end
      $\omega(i,1) \leftarrow -(1/\boldsymbol{H}(i,i)) \cdot (\boldsymbol{K}(i,1) + s1 + s2)$
      $\lambda(i,1) \leftarrow max(0, w(i,1))$
    end
    $E_\lambda^2 \leftarrow (\lambda - \lambda_p)^T \cdot (\lambda - \lambda_p)$
    if $\boldsymbol{E}_\lambda^2 < E_c$ **then**
      break;
    end
    $\lambda_p \leftarrow \lambda$
  end
  $\boldsymbol{\Delta U} \leftarrow \boldsymbol{\Delta U} - (\boldsymbol{\Phi}^T\ \boldsymbol{\Phi} + \bar{\boldsymbol{R}})^{-1} \cdot \boldsymbol{M}^T \cdot \boldsymbol{\lambda}$
end
$\Delta \mathrm{u} \leftarrow \boldsymbol{L}_0 \cdot \boldsymbol{\Delta U}$
$u \leftarrow u + \Delta \mathrm{u}$
$\boldsymbol{x}_d(k-1) \leftarrow \boldsymbol{x}_d(k)$
return u
End Of MPCControllerOneStep

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{CB} & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{CAB} & \boldsymbol{CB} & \ldots & \boldsymbol{0} \\ \ldots & & & \\ \boldsymbol{CA}^{N_p-1}\boldsymbol{B}\ \boldsymbol{CA}^{N_p-2}\boldsymbol{B} & \ldots & \boldsymbol{CA}^{N_p-N_c}\boldsymbol{B} & \end{bmatrix} \quad (6)$$

Using (4), the cost function of MPC, J, is then defined as,

$$J = (\boldsymbol{R}_s - \boldsymbol{Y})^T \cdot (\boldsymbol{R}_s - \boldsymbol{Y}) + \boldsymbol{\Delta U} \cdot \bar{\boldsymbol{R}} \cdot \boldsymbol{\Delta U} \quad (7)$$

where $\boldsymbol{R}_s \in \mathbb{R}^{N_p \times 1}$ is the vector of reference values. $\bar{R} = r_w \cdot \boldsymbol{I}_{N_c \times N_c} (r_w \geq 0)$, where r_w is the tuning parameter that determines the desired closed-loop performance.

Using (7), the MPC problem is defined as,

$$\min_{\boldsymbol{\Delta U}} \quad J \quad (8)$$
$$subject\ to: \boldsymbol{x}(k+1) = \boldsymbol{A} \cdot x(k) + \boldsymbol{B} \cdot \Delta u(k) \quad (9)$$
$$\boldsymbol{M} \cdot \Delta\mathrm{U} \leq \boldsymbol{\gamma} \quad (10)$$

where

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{I}_{N_c \times N_c} \\ -\boldsymbol{I}_{N_c \times N_c} \end{bmatrix}, \boldsymbol{\gamma} = \begin{bmatrix} \boldsymbol{\Delta U^{max}} \\ -\boldsymbol{\Delta U^{min}} \end{bmatrix}, \quad (11)$$

$\mathbf{\Delta U^{min}} = \boldsymbol{I}_{N_c\times 1}\cdot\Delta u^{\min}, \mathbf{\Delta U^{max}} = \boldsymbol{I}_{N_c\times 1}\cdot\Delta u^{\max}$. The change rate of the command rotation velocity of stepper motor has to be constrained in the range of $[\Delta u^{min}\Delta u^{max}]$, whose exact values are explained in Section 3.4. Equation (10) is equivalent to $\mathbf{\Delta U^{min}} \leq \mathbf{\Delta U} \leq \mathbf{\Delta U^{max}}$. This inequality constraint arises from the stepper motor, whose rotation acceleration has to be limited, otherwise, the stepper motor would miss some steps.

3.2 Active Set Methods

From (8) to (10), it can be seen that solving the MPC problem is actually solving quadratic programming (QP) problem with equality and inequality constraints. The active set method is one of the methods for solving MPC problem (Gu et al. (2009); Cannon et al. (2001); Bartlett et al. (2000); Lau et al. (2009)) and is employed in our case. The idea of active set methods is to define, at each step of MPC algorithm, a set of constraints, termed the working set, that is to be treated as the active set (Wang (2009)). If the active set is found, the original optimization problem with inequality constraints can be converted into an optimization problem with equality constraints. In the active set methods, finding the active constraints is a time consuming task, especially when there are many constraints. Here, the primal-dual method is used to find the active set. A dual method is applied systematically to find the inactive constraints, which can then be removed in the solution. The dual problem is written as below, whose detailed derivation procedure can be found in the paper of Wang (2009).

$$\max_{\boldsymbol{\lambda}\geq 0}\left(-\frac{1}{2}\boldsymbol{\lambda}^T\boldsymbol{H}\boldsymbol{\lambda}-\boldsymbol{\lambda}^T\boldsymbol{K}-\frac{1}{2}\boldsymbol{F}_d^T\boldsymbol{E}^{-1}\boldsymbol{F}_d\right) \tag{12}$$

where $\boldsymbol{\lambda}$ is the Lagrange multipliers, also called dual variables in the optimization literature. And,

$$\boldsymbol{H} = \boldsymbol{M}\cdot\boldsymbol{E}^{-1}\cdot\boldsymbol{M}^T \tag{13}$$

$$\boldsymbol{K} = \boldsymbol{\gamma} + \boldsymbol{M}\cdot\boldsymbol{E}^{-1}\cdot\boldsymbol{F}_d \tag{14}$$

where $\boldsymbol{E} = \boldsymbol{\Phi}^T\boldsymbol{\Phi} + r_w\cdot\boldsymbol{I}_{N_c\times N_c}$, $\boldsymbol{F}_d = -\boldsymbol{\Phi}^T(\boldsymbol{R}_s - \boldsymbol{F}\cdot\boldsymbol{x}(k))$

From (12), it can be seen that the dual is also a quadratic program problem with λ as the decision variable. (12) is equivalent to,

$$\min_{\boldsymbol{\lambda}\geq 0}\left(\frac{1}{2}\boldsymbol{\lambda}^T\boldsymbol{H}\boldsymbol{\lambda}+\boldsymbol{\lambda}^T\boldsymbol{K}+\frac{1}{2}{\boldsymbol{F}_d}^T\boldsymbol{E}^{-1}\boldsymbol{F}_d\right) \tag{15}$$

that is solved by the Hildreth's quadratic programming procedure, which is based on an element-by-element search, therefore, no matrix inversion is required.

The active set method corresponds to Line $10-32$ in Algorithm 1.

3.3 Explanation of Algirithm 1

In this section, the Algorithm 1 is explained with details, from algorithm initialization to the functions of some important lines.

Before the execution of Algirithm 1, some variables have to be set and initialized. The following variables are set: $N_c = 40$, $N_p = 40$, $\Delta u^{min} = -3\ deg/s$, $\Delta u^{max} = 3\ deg/s$, and $\Delta t = 0.002$. How to get these parameters are explained in Section 3.4 with details. These parameters, together with the identified model parameters, $\xi_v = 0.022$ and $w_n = 27.3\ rad/s$, are used to calculate $\boldsymbol{F}$, $\boldsymbol{\Phi}$, $\boldsymbol{M}$, $\boldsymbol{\gamma}$, $\boldsymbol{H}$, by (5), (6), (11), and (13), respectively. Then, the following parameters have to be initialized before execution of Algorithm 1: $u = 0, \boldsymbol{x}_d(k-1) = \boldsymbol{0}, \boldsymbol{\lambda} = \boldsymbol{0}_{2N_c\times 1}, \boldsymbol{\lambda}_p = \boldsymbol{0}_{2N_c\times 1}, \boldsymbol{L}_0 = [1\ 0\ ...\ 0]_{N_c\times 1}, N_1 = 100, E_c = 0.001$, where u is the controller output, $\boldsymbol{x}_d(k-1)$ is the previous state, $\boldsymbol{\lambda}$ is the Lagrange multipliers, $\boldsymbol{\lambda}_p$ is the previous $\boldsymbol{\lambda}$, $\boldsymbol{L}_0$ is a vector that is used to select the first element of $\boldsymbol{\Delta U}$, N_1 is just a constant that determines the number of cycles for finding $\boldsymbol{\lambda}$ that satisfies (15), E_c is an error threshold that determines when the cyclic section should be left. N_1 and E_c are specified by trial and error.

After parameters setting and initialization, the main parts of Algorithm 1 are explained. There are three inputs in the algorithm, which are $\boldsymbol{R}_s$, $\boldsymbol{x}_d(k)$ and $y(k)$. $\boldsymbol{R}_s$ is the reference velocity of the elevation part. $\boldsymbol{x}_d(k)$ is the current state, whose first element is the current rotation velocity of the elevation part, and whose second element is the current rotation acceleration of the elevation part. $y_d(k)$ is the model output in (2). $\boldsymbol{x}_d(k)$ and $y_d(k)$ are used to get the state variable, $\boldsymbol{x}(k)$ in (3), which can be found in Line 2. In Line 3, the MPC without inequality constraints is solved and $\boldsymbol{\Delta U}$ is obtained. From Line 4 to 9, the elements of $\boldsymbol{\Delta U}$ are checked one by one to see whether the inequality constraints are violated. If it is, the active set method is applied to find the active set which is used to correct $\boldsymbol{\Delta U}$, shown in Line 31. The whole procedure of active set method corresponds to Line 10 to 32. Line 33 is used to get the first element of $\boldsymbol{\Delta U}$. As Δu is the change of u, Line 34 is used to get the latest u, which is then returned as the controller output. Then, $\boldsymbol{x}_d(k-1)$ is updated in Line 35 for next cycle.

3.4 Parameters in MPC algorithm

There are several parameters in MPC algorithm, which will be discussed here with details.

The first parameter is Δt, which is sampling interval of MPC algorithm and determines the value of N_p and N_c. Currently, the setting is, $\Delta t = 0.002\ sec$. The bandwidth of position loop of the attitude control system has to be above $16Hz$. As the rule of thumb, the bandwidth of the velocity loop is least 5 times larger than that of position loop, which means the bandwidth of velocity loop should be above 80 Hz. The sampling frequency of a control loop is 5 times faster than the bandwidth of the control loop, which means the sample frequency of the velocity loop should be above 400Hz. That is why 500 KHz is applied as the sampling frequency of MPC algorithm.

Other parameters are N_p and N_c. They are set to be the same value in our MPC. Currently, the setting is, $N_p = N_c = 40$. The reason is that the desired closed-loop response time of the velocity loop is 80 ms and $\Delta t = 0.002\ sec$, the N_p is chosen in such a way that makes $N_p\cdot\Delta t = 80\ ms$.

Finally, the settings of Δu^{min} and Δu^{max}, are explained. They are the bound of the velocity change of u in every 2 ms. Currently, the settings are $\Delta u^{min} = -3\ deg/s$

and $\Delta u^{max} = 3\ deg/s$, which are obtained through experiments.

4. FPGA IMPLEMENTATION

The designed MPC algorithm is firstly verified in simulation environment in Matalb/Simulink. After successfully doing that, the next step is to implement the MPC algorithm in real hardware. This is a very challenging task, because, in out case, the sampling rate is only $2ms$ and the prediction horizon and control horizon is very large, $N_p = N_c = 40$.

The FPGA chip is selected to execute MPC mainly because of its parallel computational capacity. Through the utilization of more FPGA resources, more parallel computation can be achieved, which results in less time needed for the execution of algorithm. For example, assume that there is one for loop with the iteration times of 100. And every iteration will cost 1 ms and $N_s\%$ of the FPGA resources. The total time and resources needed for finishing this for loop would be about $100ms$ and $N_s\%$, respectively, when not applying parallel mechanism. By comparison, those number would be $1ms$ and $100 \cdot N_s\%$ when applying fully parallel mechanism. That is the feature of FPGA and the resources can be used to save time.

4.1 Hardware Setup

The MPC algorithm is implemented on a Xilinx Zynq chip, xc7z020, which is a low cost SOC FPGA chip which integrates two ARM processors (Cortex-A9 with $866MHz$) and FPGA architectures into a single device. Consequently, they provide higher integration, lower power, smaller board size, and higher bandwidth communication between the processor and FPGA. They also include a rich set of peripherals, on-chip memory, an FPGA-style logic array, and high speed transceivers. Xc7z020 is called All Programmable System on Chip (SoC), whose software and hardware can all be programmed. Moreover, a kind of high-speed in-chip data bus, Advanced Extensible Interface (AXI), is used to exchange data between ARM cores and FPGA, whose transmission rate could be at least 800 MByte/sec.

The MPC algorithm coded in C language was imported into Xilinx Vivado High-Level Synthesis (HLS) software, which is a software that can convert C code into an Intellectual Property (IP) block. This IP block is then used in Vivado software and converted into bitstream together with other IP blocks. The bitstream is the file that can be downloaded into FPGA chip.

4.2 MPC Implementation in Different Data Types

The resources in one FPGA chip is fixed and limited. And every calculation operation would consume certain amount of resources. However, for the same operation, the resources consumed by different data types can vary greatly. That is, the less number of bits occupied by the data type, the less resources consumed by the operation. The data types with less bits are preferred, and its limitation is whether the precision and range of this data type is high enough for algorithm.

Table 1. Fixed-point data types of some variables

	Word Length (bit)	Fraction Length(bit)	Signed	Range	Precision
$\boldsymbol{R_s}$	17	9	1	[-128,128)	0.0039
$\boldsymbol{x_d}$	16	8	1	[-128,128)	0.0039
y_d	17	10	1	[-64,64)	9.7656e-04
$\boldsymbol{\Delta U}$	16	9	1	[-64,64)	0.0020
Δu	18	13	1	[-16,16)	1.2207e-04
λ	16	11	0	[0,32)	4.8828e-04
ω	16	9	1	[-64,64)	0.0020

The FPGA in xc7z020 chip supports a broad range of data types from binary (1 bit) to double precision (64 bit), which provides great flexibility for optimizing resource usage while meeting design performance targets. In this paper, three kinds of data types are explored for the execution of MPC algorithm. They are single-precision floating-point data type (32 bit), half-precision floating-point data type (16 bit) and fixed-point data type (customized bit). The half-precision floating-point data type is released in 2002 and has 1 sign bit, 5 bits exponent width, and 11 bits mantissa width (10 explicitly stored).

The difficulties in the implementation of MPC algorithm in FPGA with different data types varies greatly. Among them, the single-precision floating point version is the easiest one. After implementing the algorithm in C code, the algorithm can be implemented in FPGA with help of Vivado and HLS software. The half-precision floating-point version can also be easily implemented. The works needed are only changing the data type from "float" into "half" in variable definition in HLS and modifying the function parameters into array or pointer. Converting the MPC algorithm from floating point to fixed point is very difficult for complicated algorithm, such as MPC, because of the limited tool support. Although the Vivado HLS software can be used to reduce the needed time, it is still a challenging task. We need to find the suitable bit width for each variable in our algorithm, according to the range and precision needed by that variable, to avoid overflow and rounding that would seriously affect controller performance. By using fixed-point data type, what the benefit you can get is the obvious reduction of the consumed resources and algorithm execution time. The fixed-point data types of some important variables in Algorithm 1 are shown in Tab. 1. The variables in Tab. 1 are only small part of the variables involved in Algorithm 1, where about 50 variables are defined and all of them have to be converted into fixed-point data type. Many temporary variables are not listed here to limit the length of the paper.

The FPGA resource utilizations of MPC implementations in three different data types are shown in Tab. 2, where LUT is Lookup Table, FF is Flip Flops, DSP48E is a digital signal processing logic element included on FPGA, and BRAM_18K is a configurable memory module for data storage. The clock cycles and consumed time are also listed here. It can be clearly seen that, from floating-point data type to fixed-point data type, the consumed resources and time are all decreased.

Besides different data types, there are other calculation optimization mechanisms in FPGA for shorting algorithm

Table 2. FPGA Resources usage and Consumed time

	Floating Point	Half Floating Point	Fixed Point
LUT	34624(65%)	26242(49%)	1952(3%)
FF	22901(21%)	22492(21%)	1528(1%)
DSP48E	203(92%)	54(24%)	11(5%)
BRAM_18K	88(31%)	60(21%)	24(8%)
Clock Cycles	185409	128435	86346
Consumed Time	1.85 ms	1.28ms	0.86ms

execution time, such as pipeline, unroll, array partition, etc.

5. HARDWARE-IN-THE-LOOP (HIL) SIMULATION

The HIL simulation is performed in this section to test the proposed control scheme, especially the designed MPC.

The diagram of HIL simulation is illustrated in Fig. 2.

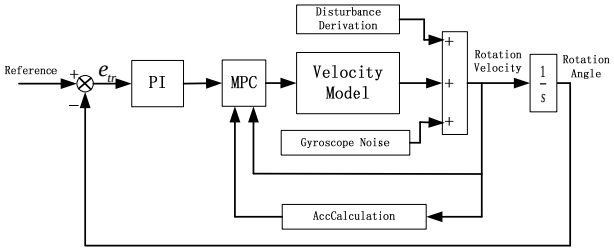

Fig. 2. Diagram of simulation of attitude stabilization control system

In Fig. 2, the "PI" block is the position loop controller, which is actually a simple PI controller. The "MPC" block is the velocity loop controller. The velocity model used in simulation is (2). The disturbance from ship dynamics is described as $A \cdot sin(\omega \cdot t)$, where $A = 25\ deg$ and $\omega = 2\pi/6\ sec$, whose derivation is the "Disturbance Derivation" block. The "Gyroscope Noise" block is added to increase the simulation reliability. The noise data is actually the measurement of gyroscope sensor chip when the system is in static state. The "AccCalculation" block is applied to calculate the rotation acceleration directly from rotation velocity, whose detailed description can be found in Section **??**. The reference angle of the position control loop is 50 deg. The simulation step is set to be $0.002sec$.

In HIL simulation, the MPC block in Fig. 2, is implemented in FPGA while other blocks are run in other real-time computation platform. Usually, a computer together with Matlab/Simulink or LabView is used as the real-time platform Xu et al. (2016); Hartley et al. (2014). Here, a creative method for HIL simulation is proposed because of the powerful architecture of SOC FPGA chip.

The proposed method of HIL simulation is illustrated in Fig. 3. The MPC block in Fig. 2 is implemented in FPGA, and the other blocks are implemented in ARM core 1 with bare mental. A timer, AXI Timer, is defined in FPGA with the interruption interval of $2ms$, whose parameters are set by ARM core 1 through AXI data bus. Every $2ms$, ARM core 1 would receive an interrupt from AXI Timer, and the corresponding Interrupt Service Routine (ISR) will be called. Inside ISR, the following works are done:1) Set the inputs of MPC through AXI data bus; 2) Start MPC calculation through AXI data bus; 3) Wait until the calculation is done; 4) Get the MPC calculation results through AXI data bus. Finally, exit ISR and wait for next interrupt. The obtained calculation results are put in DDR SDRAM by ARM core 1, which can also be read by ARM core 2 through shared memory. In ARM core 2, the Linux is run and is responsible for high-level tasks, such as data displaying through GUI, network communication, data storage on SD card, etc. The benefits of this method are: 1) Real-time performance can be guaranteed, as no operation system exists in ARM core 1 and only a timer is used; 2)No data packet needs to be parsed, as it is known that parsing a data packet is much harder than sending a data packet; 3) No expensive device or software is required.

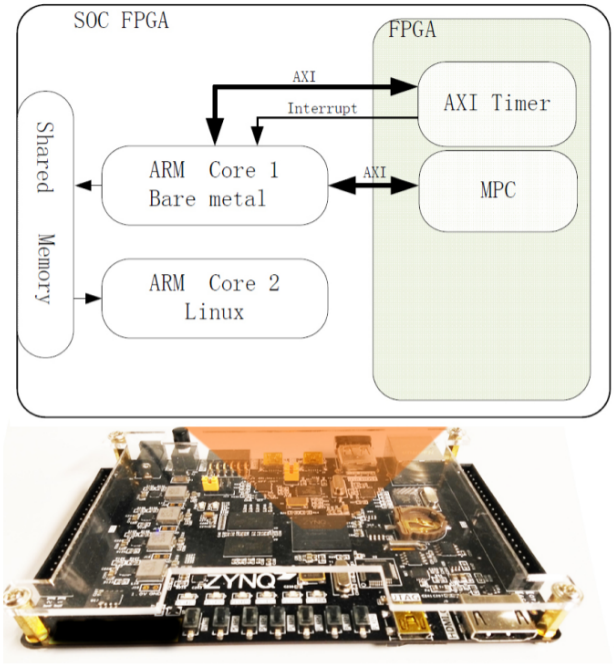

Fig. 3. Illustration of proposed HIL simulation in one SOC FPGA chip

In Section 4.2, three data types are explored for MPC implementation, which are tested here in HIL simulation to check their performance. The MPC implementation in each data type is tested in HIL simulation one by one, and the testing results are collected and shown in Fig. 4.

From Fig. 4, it can be seen that the controller performance of these three MPC controllers are similar. The RMS tracking angle errors are all less than the product requirement, 0.2 deg. The ability of MPC to handle constraints can be found in Fig. 5. Because of the high similarity of Δu in all three kinds of MPC implementation, only the one in fixed-point MPC is shown. From the top part of Fig. 5, it can be seen that Δu can be successfully handled by MPC and be limited in allowable range. To show the details of Δu, the area, indicated by a red frame in the top part of Fig. 10, is plotted in the bottom part of Fig. 5.

There are advantages and disadvantages in each kind of MPC implementation. The floating-point data type has the largest data range and highest data precision. The half-precision floating-point data type can satisfy

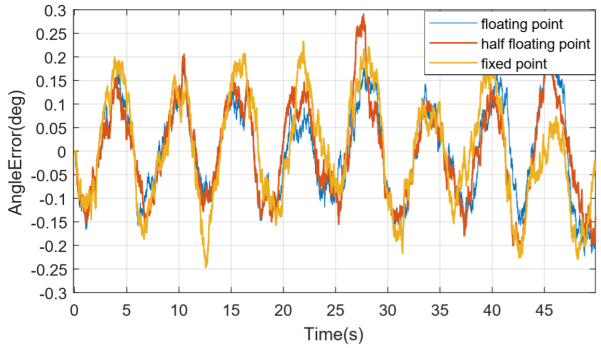

Fig. 4. HIL simulation results of three different kinds of MPC implementation.

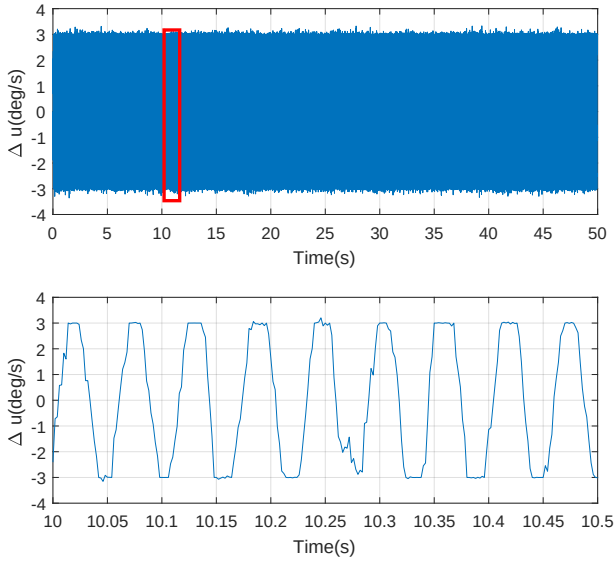

Fig. 5. Top: Δu output of fixed-point MPC during the whole HIL simulation process. Bottom: Δu output of fixed-point MPC between $10sec$ to $10.5sec$.

most cases of application requirements in terms of data range and precision while can significantly decrease the consumed FPGA resources. Moreover, the conversation from floating-point to half-precision floating-point data type is very easy. Among these data types, the fixed-point data type possess absolute advantages in terms of consumed time and resources, which is achieved at the cost of long design time. Which data type is the best depends on the specific control system needed to be handled.

The whole control system of MSTA becomes more challenging after realizing that, in the attitude stabilization system of MSTA, there are totally three independent control system, which are the control systems of elevation part, cross-elevation part, and azimuth part. Each control system has the same structure as shown in Fig. 2. Hence, there are totally three MPC controllers, which require huge amount of computation capacity. The computation capacity of the FPGA is only limited by its resources and

Table 3. Resources utilization after implementation of three fixed-point MPCs

Resources	Utilization	Available	Utilization %
LUT	4500	532000	8.5
FF	4977	106400	4.7
DSP48E	36	220	16.4
BRAM_18K	34.5	140	24.6

xc7z020 is tested to see whether it can cope with the computation task of three MPCs. After implementation of all three fixed-point MPCs in the FPGA, the FPGA resource utilization is shown in Fig. 6, where the area in bright yellow color indicates the used resources and the dark area means unused resources. It can be clearly seen that there are still lots of unused resources, which means the FPGA can handle the computation of three parallel MPCs. The utilization of FPGA resources is also summarized in Tab. 3. As the MSTA under development is a commercial product, the price of every component has to be as low as possible. From the percentage of the used FPGA resources, it can be concluded that a cheaper FPGA chip with smaller FPGA resources, such as xc7z010, will be able to implement three MPC controllers.

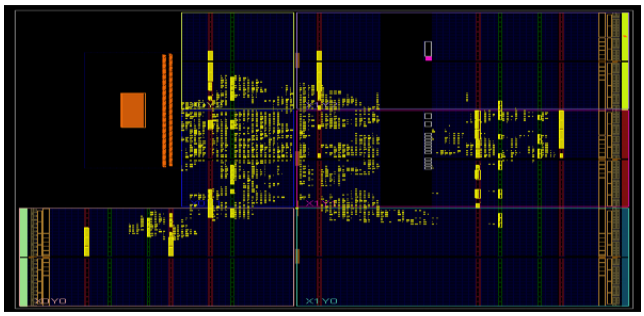

Fig. 6. The FPGA resources consumed after implementation of three MPCs in FPGA

Another test is done to check whether the control system performance will be affected if three MPC controllers are implemented in the SOC FPGA. The running results of fixed-point MPC of the elevation part in the case of three-MPC implementation are compared with that in the case of one-MPC implementation. And there is no difference in the data collected in both cases.

6. CONCLUSION

In this paper, a control scheme based on MPC for attitude stabilization of MSTA is proposed and tested in FPGA. The modelling of stepper motor is stated. The MPC algorithm based on active set method is given and explained with details. Three data types are explored for the implementation of MPC in FPGA. A creative HIL simulation is proposed to verify the performance of each MPC implementation. The HIL simulation is achieved in one SOC FPGA, which doesn't require any expensive tools or software. From HIL simulation results, it can been seen that the proposed control scheme can satisfy the product requirements and the RMS tracking angle error is within 0.2 deg. The application of MPC can handle of the velocity change problem of stepper motor. The implementation of three MPCs in one FPGA is also tested and its feasibility is verified.

ACKNOWLEDGEMENTS

Authors would like to thank Innovation Fund Denmark for financial support. Thanks also goes to SpaceCom A/S for providing assistance.

REFERENCES

Bartlett, R.A., Wachter, A., and Biegler, L.T. (2000). Active set vs. interior point strategies for model predictive control. In *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*, volume 6, 4229–4233 vol.6.

Camacho, E.F. and Alba, C.B. (2007). *Model Predictive Control*. 1439-2232. Springer-Verlag London.

Cannon, M., Kouvaritakis, B., and Rossiter, J.A. (2001). Efficient active set optimization in triple mode mpc. *IEEE Transactions on Automatic Control*, 46(8), 1307–1312.

Damiano, A., Gatto, G., Marongiu, I., Perfetto, A., and Serpi, A. (2014). Operating constraints management of a surface-mounted pm synchronous machine by means of an fpga-based model predictive control algorithm. *IEEE Transactions on Industrial Informatics*, 10(1), 243–255.

Darba, A., Belie, F.D., D'haese, P., and Melkebeek, J.A. (2016). Improved dynamic behavior in bldc drives using model predictive speed and current control. *IEEE Transactions on Industrial Electronics*, 63(2), 728–740.

Dundar, A., Jin, J., Martini, B., and Culurciello, E. (2017). Embedded streaming deep neural networks accelerator with applications. *IEEE Transactions on Neural Networks and Learning Systems*, 28(7), 1572–1583.

Gu, R., Bhattacharyya, S.S., and Levine, W.S. (2009). Improving the performance of active set based model predictive controls by dataflow methods. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, 339–344.

Gulbudak, O. and Santi, E. (2016). Fpga-based model predictive controller for direct matrix converter. *IEEE Transactions on Industrial Electronics*, 63(7), 4560–4570.

Hartley, E.N., Jerez, J.L., Suardi, A., Maciejowski, J.M., Kerrigan, E.C., and Constantinides, G.A. (2014). Predictive control using an fpga with application to aircraft control. *IEEE Transactions on Control Systems Technology*, 22(3), 1006–1017.

Jerez, J.L., Goulart, P.J., Richter, S., Constantinides, G.A., Kerrigan, E.C., and Morari, M. (2014). Embedded online optimization for model predictive control at megahertz rates. *IEEE Transactions on Automatic Control*, 59(12), 3238–3251.

Kosan, T., Talla, J., Janous, S., and Blahnik, V. (2018). Fpga-based accelerator for model predictive control of induction motor drive. In *2018 18th International Conference on Mechatronics - Mechatronika (ME)*, 1–6.

Lau, M.S.K., Yue, S.P., Ling, K.V., and Maciejowski, J.M. (2009). A comparison of interior point and active set methods for fpga implementation of model predictive control. In *2009 European Control Conference (ECC)*, 156–161.

Lucia, S., Navarro, D., Lucía, ., Zometa, P., and Findeisen, R. (2018). Optimized fpga implementation of model predictive control for embedded systems using high-level synthesis tool. *IEEE Transactions on Industrial Informatics*, 14(1), 137–145. doi:10.1109/TII.2017.2719940.

Ma, Z., Saeidi, S., and Kennel, R. (2014). Fpga implementation of model predictive control with constant switching frequency for pmsm drives. *IEEE Transactions on Industrial Informatics*, 10(4), 2055–2063.

Ming, A., Yamaoka, T., Kida, T., Kanamori, C., and Satoh, M. (2005). Accuracy improvement of ship mounted tracking antenna for satellite communications. In *IEEE International Conference Mechatronics and Automation, 2005*, volume 3, 1369–1374 Vol. 3.

Monmasson, E. and Cirstea, M.N. (2007). Fpga design methodology for industrial control systems – a review. *IEEE Transactions on Industrial Electronics*, 54(4), 1824–1842.

Monmasson, E., Idkhajine, L., Cirstea, M.N., Bahri, I., Tisan, A., and Naouar, M.W. (2011). Fpgas in industrial control applications. *IEEE Transactions on Industrial Informatics*, 7(2), 224–243.

Rahman, A., Lee, J., and Choi, K. (2016). Efficient fpga acceleration of convolutional neural networks using logical-3d compute array. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, 1393–1398.

Ramrez, R.O., Espinoza, J.R., Mel?n, P.E., Reyes, M.E., Espinosa, E.E., Silva, C., and Maurelia, E. (2014). Predictive controller for a three-phase/single-phase voltage source converter cell. *IEEE Transactions on Industrial Informatics*, 10(3), 1878–1889.

Soltani, M.N., Izadi-Zamanabadi, R., and Wisniewski, R. (2011). Reliable control of ship-mounted satellite tracking antenna. *IEEE Transactions on Control Systems Technology*, 19(1), 221–228.

Tseng, H.C. and Teo, D.W. (1998). Ship-mounted satellite tracking antenna with fuzzy logic control. *IEEE Transactions on Aerospace and Electronic Systems*, 34(2), 639–645.

Tu, W., Luo, G., Chen, Z., Liu, C., and Cui, L. (2019). Fpga implementation of predictive cascaded speed and current control of pmsm drives with two-time scale optimization. *IEEE Transactions on Industrial Informatics*, 1–1. doi:10.1109/TII.2019.2897074.

Wang, A., Zhang, L., and Wei, L. (2011). The design and implementation of the digital servo system for the satellite antenna. In *2011 International Conference on Electrical Machines and Systems*, 1–5.

Wang, L. (2009). *Model Predictive Control System Design and Implementation Using MATLAB*. Springer-Verlag London.

Xu, F., Chen, H., Gong, X., and Mei, Q. (2016). Fast nonlinear model predictive control on fpga using particle swarm optimization. *IEEE Transactions on Industrial Electronics*, 63(1), 310–321.

Zhang, L., Zhou, Z., Chen, Q., Long, R., and Quan, S. (2019). Model predictive control for electrochemical impedance spectroscopy measurement of fuel cells based on neural network optimization. *IEEE Transactions on Transportation Electrification*, 5(2), 524–534. doi: 10.1109/TTE.2019.2909687.