MODEL PREDICTIVE CONTROL USING FPGA

Khalil.Mohamed,Ahmed.El Mahdy and Mohamed.Refai

Department of Systems and Computers Engineering, Al_azhar University, Cairo, Egypt

ABSTRACT

Model predictive control (MPC) is an advanced control algorithm that has been very successful in the control industries due to its capability of handling multi input multi output (MIMO) systems with physical constraints. In MPC, the control action are obtained by solving a constrained optimization problem at every sample interval to minimize the difference between the predicted outputs and the reference value through the using of minimum control energy and satisfying the constraints of the physical system. Quadratic programing (QP) problem is solved using QPKWIK method which improves the active set method. The system architecture and design for the implementation of online MPC on the FPGA is taken into consideration in this paper to control a DC motor. This implementation is completed using Spartan6 Nexys3 FPGA chip using simulation environment (EDK tool) and the comparison between MPC and PID controller is also established.

KEYWORDS

MPC, FPGA, MATLAB, QPKWIK Method, PID Controller, DC Motor

1.INTRODUCTION

MPC has become an established control technology through its capability of handling problems with constraints. Many applications become using MPC such as petrochemical industry, chemical process industries, physical processes robotic control system, etc.[1, 2, 3]. MPC is a dynamic optimization problem so, it can be formulated as a QP problem. MPC has the ability to deal with the physical constraints which comes from the industrial applications and control process. MPC computes a vector of optimal control signals by solving QP problems according to a certain constraints to minimize the difference between the reference value and the future outputs predicted from an interesting plant model. Then only the first signal of the optimal input vector is then applied to the plant and this procedure from the prediction and optimization with new optimal input is repeated at the next sampling interval [4, 5].

A successful MPC routine can be obtained when it is mainly dependent on (i) the degree of the precision of the specification of a suitable plant model and (ii) the ability to solve the QP problem online with a prescribed sampling interval to generate a feasible solution within a sampling interval. The implementation of MPC with physical constraints for embedded control has been investigated in [6], where a Handel-C implementation of an MPC algorithm was described and realized on Xilinx Spartan 3 FPGA board. When applying MPC to complex systems with fast response time, the ability to solve the QP problem online become critical where computational resource may be limited when using it in the embedded applications [7].

The essence of the MPC controller is to solve a constrained optimization problem online. Therefore online computational complexity results in most of MPC systems implemented on

DOI : 10.5121/ijctcm.2015.5201

high performance computers, so MPC with various applications is limited on a field controller [8, 9]. The online optimization procedure must be completed in reasonable short time, in order to implement MPC algorithm in field controllers. There are many of hardware platform such as digital signal processing (DSP), application-specific integrated circuit (ASIC) and field programmable gate array FPGA. FPGA technology has several advantages such as flexibility, computing efficiency and contains many programmable logic resources, which can be configured to perform complex functions directly in hardware. Therefore FPGA may achieve very high processing speed [10, 11, 12].

FPGAs are equipped recently with a lot of resources that allow them to hold large digital systems on a single chip. FPGA vendors provide tools that allow the designer to build embedded systems efficiently on FPGAs. This implementation is completed using Xilinx Embedded Development kit (EDK), a tool provided by Xilinx for building an embedded system-on-chip approach (SoC) on its FPGAs. EDK allows the designer to build a complete microprocessor system based on an embedded processor from Xilinx called MicroBlaze. The tool provides a C/C++ compiler for that processor and an Integrated Development Environment (IDE) based on Eclipse framework. The system is implemented first on MATLAB, and the MATLAB code is converted to C code. The C code of the MPC algorithm is compiled into a bitstream file which is then downloaded to the Spartan6 Nexy3 board to configure the FPGA chip to perform the constrained MPC controller for a DC motor.

This paper is organized as follows. Section 2 represents an introduction to MPC. Representation of MPC for speed control of DC motor is introduced in Section 3. The Generation of C Code from MATLAB code is discussed in section 4. A comparison between MPC and PID controller is established in section 5. Section 6, provides FPGA implementation for the system. In Section 7 the paper is concluded.

2.MODEL PREDICTIVE CONTROL

MPC describe an approach to control design not specific algorithm and the interested people interpreted this approach to get the algorithm for their own need. MPC uses model process as shown in figure. 1to predict the future response of a plant in a feed forward manner (i.e. open loop) over specified time interval by solving a Finite-Horizon optimal Control (FHC) problem subjected to the constraints on state, control and output of the system [13, 14, 15].

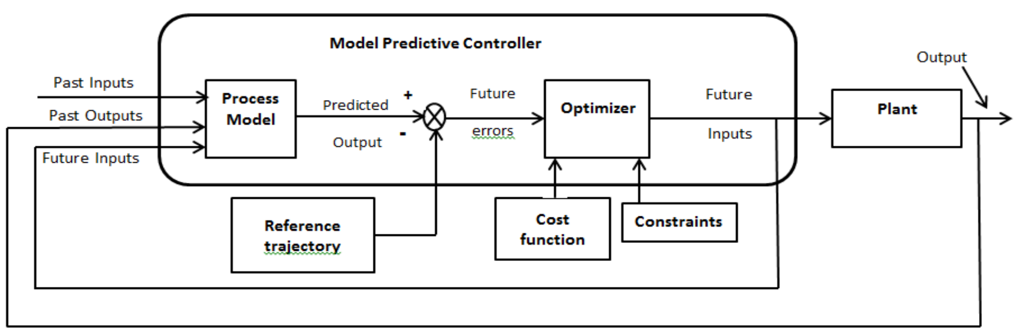

Figure. 1. Structure of Model Predictive Control

MPC can be summarized into three main steps as shown in figure. 2. The first is the Prediction step on [*N*]. At each sampling time instant k, the model of the process is used to predict the future behavior of the controlled plant over a specified time horizon, which is often called the prediction horizon and is denoted by *N*.

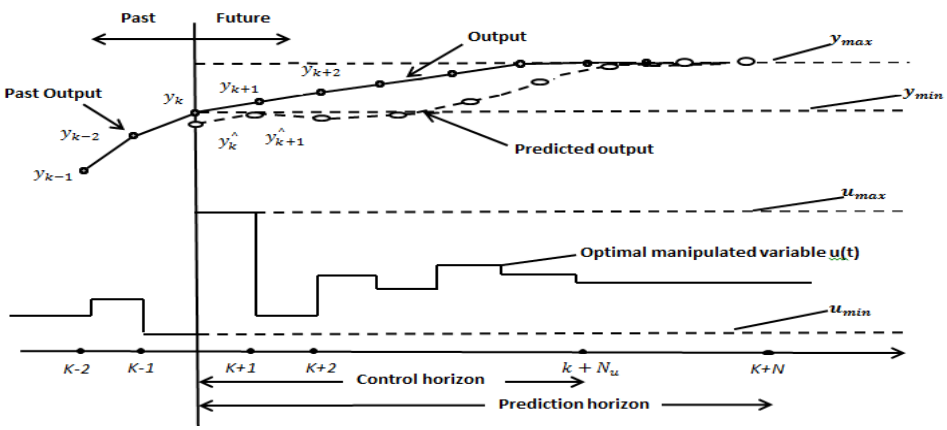

Figure. 2. Model Predictive Control Process

The second one is Control horizon on [N_u], in this step a cost function is minimized subject to constraints to compute an optimal vector *(u (k), u (k+1), u (k+2),…, u (k+*N_u*-1))* of controls of future input signals online at sample k over a specified time horizon, which is usually called control horizon and is denoted by N_u at the end of the control horizon control action become constant. Finally the optimal value *u*(*k*) of control vector is then applied to the plant. At the next sample time *k+1*, the whole process of prediction and optimization will be repeated [16, 17].

2.1 Mathematical Formulation of MPC

MPC is sampled data algorithm so that the state variable model which represents the system can be written as follow.

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned} \qquad (1)$$

Where, $x(k) \in R^n, u(k) \in R^m, y(k) \in R^p$ denote the state, control input, and output $A \in R^{n\times n}$ is the state matrix, $B \in R^{n\times m}$ is the input matrix. As introduced in [17, 18], a vector of optimal control input signals and states can be obtained using MPC controller and represented as follow $U(k) = [u^T(K), u^T(k+1), \dots\dots, u^T(k+N_u-1)]^T$, $X(K) = [x^T(K+1), \dots, x^T(K+N)]^T$. Therefore the main task is to find the optimal control signal $u^*(k)$ such that the performance index (objective function) in eq. 2 is minimized according to physical constraints to move the system from initial state to a final state as follow.

$$min_{u(k)}{}^{J} = \frac{1}{2}x^T(k+N)P_f x(k+N)$$

$$+\sum_{i=0}^{N-1} x^T(k+i)Q_i x(k+i) + \sum_{i=0}^{N_u-1} u^T(k+i)R_i u(k+i) \qquad (2)$$

Subject to

$$y_{min} \leq y \leq y_{max}, \; u_{min} \leq u \leq u_{max}$$

Where N=10, is the prediction horizon, N_u=3, is the control horizon $Q_i \in R^{N\times N}$ and $R_i \in R^{N_u\times N_u}$ are symmetric and positive semi-definite weighting matrices that are specified by the user [19, 20]. The weighting matrices Q and R are the parameters that are tuned until a desired performance is achieved. eq. 2 can be written in the following standard QP form:

$$min_{u(k)}{}^{J} = \frac{1}{2}U_{N_u}^T G U_{N_u} + g^T U_{N_u} \tag{3}$$

$$\textit{Subject to} \quad \Lambda U_{N_u} \leq b.$$

Where $g \in R^{N_u\times n}$ is composed matrix of the state vector *x(k)*, $G \in R^{N_u\times N_u}$ is a constant matrix determined by $G = \Psi_u{}^T \bar{Q}\Psi_u + \bar{R}$, $g = \Psi_u{}^T \bar{Q}\Psi_x x(k)$. At each sampling interval the optimization problem of eq. 3 should be solved to obtain the optimal input signal. Note that $\Psi_u, \bar{Q}, \bar{R}, \Psi_x$ can be calculated as follow:

$$\bar{Q} = \begin{bmatrix} Q & 0 & 0 & 0 \\ 0 & Q & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \frac{1}{2}P \end{bmatrix}, \bar{R}\begin{bmatrix} R & 0 & 0 & 0 \\ 0 & R & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & R \end{bmatrix}, \bar{C} = \begin{bmatrix} c & 0 & 0 & 0 \\ 0 & c & 0 & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \dots & c \end{bmatrix}, \Psi_x = \begin{bmatrix} A \\ A^2 \\ \vdots \\ \vdots \\ A^N \end{bmatrix},$$

$$\Psi_u = \begin{bmatrix} B & 0 & 0 & \cdots & \cdots & 0 \\ AB & B & 0 & \cdots & \cdots & 0 \\ A^2B & AB & B & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \sum_{i=0}^{N-1} A^iB & \cdots & \cdots & \cdots & \cdots & \sum_{i=0}^{N-N_u} A^iB \end{bmatrix}.$$ Λ, b are a constant matrices and can be determined by $\Lambda = \begin{bmatrix} I_N\bar{C}\Psi_u \\ -I_N\bar{C}\,\Psi_u \\ I_{N_u} \\ -I_{N_u} \end{bmatrix}$, $b = \begin{bmatrix} I_N y_{max} - I_N\bar{C}\Psi_u x(k) \\ -I_N y_{min} + I_N\,\bar{C}\Psi_u x(k) \\ I_{N_u}U_{max} \\ -I_{N_u}U_{min} \end{bmatrix}$

.

2.2 QPKWIK Method for the Solution of QPs

A multi-step Newton method that is called QPKWIK is used in this paper to solve the QP problem. This method is based on the karush-kuhn-tucker conditions for feasibility test and BFGS formula for optimality test. QPKWIK has been implemented for enhancing efficiency of the active set method and determining search direction if infeasible QP sub-problems are included [21, 22].

$$min \; \frac{1}{2}U_{N_u}{}^T Gx + g^T U_{N_u} \tag{4}$$

$$\textit{Subject to} \quad \Lambda^T U_{N_u} \geq b$$

Taking into account only the active inequalities eq. (4) can be rewritten as

$$min \quad \frac{1}{2}{U_{N_u}}^T Gx + g^T U_{N_u} \tag{5}$$

$$Subject.to \quad \not{A}^T U_{N_u} = b$$

Where, $\not{A}$ represents the active inequality constraint. The optimal control signal is represented by *x*, that is corresponds to the optimization variable instead of U_{N_u}in eq. 5 [23].

2.3 Algorithm Used to Solve QPKWIK Method

Step1: Initialization let $f(x^k)$ is the cost function, $B_k = [\nabla^2 f(x^{k+1})]^{-1}$ is the inverse of the Hessian matrix, $\mu_j$the Lagrange Multiplier, x^* is the optimal point (decision variable), λ_k is the optimal step length, $d_k$is the search direction, δ_k is the difference between two successive decision variable $\delta_k = x^{k+1} - x^k$, γ_k is the difference between two successive gradient point $\gamma_k = \nabla f(x^{k+1}) - \nabla f(x^k)$, m is the number of inequality constraints finally $(.)$ represents the Lagrange multiplier coefficients.

Step2: Find the unconstrained solution, $x = -G^{-1}g$.

Step3: Check for the violation of any of the inactive inequality constraint.

Take the Lagrange multiplier.

$$l(x, \mu, s) = f(x) + \sum_{j=1}^{m} \mu_j (g_j(x) + s_j^2)$$

Apply the necessary condition.

$$\frac{\partial l(.)}{\partial x_k} = \frac{\partial f(x)}{\partial x_k} + \sum_{j=1}^{m} \mu_j \frac{\partial (g_j(x) + s_j^2)}{\partial x_k} = 0$$

$$\frac{\partial l(.)}{\partial \mu_j} = g_j(x) + s_j^2 = 0, \;\; j = 1,2, \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots., m$$

Step4: Computation of $x^*$and μ_j^* by solving KKT system

$$\begin{bmatrix} x^* \\ \mu^* \end{bmatrix} = \begin{bmatrix} H & D \\ D^T & U \end{bmatrix} \begin{bmatrix} -g \\ b \end{bmatrix}$$

Where, *H*, *D* and *U* matrices are explicitly expressed in the following form after using matrix inversion and schur complement.

$$H = -G^{-1} + G^{-1}\not{A}(\not{A}^T G^{-1}\not{A})^{-1} G \not{A}^T G^{-1}$$
$$D = G^{-1}\not{A}(\not{A}^T G^{-1}\not{A})^{-1}$$
$$U = -(\not{A}^T G^{-1}\not{A})^{-1}$$

Determine the feasibility condition.

If $s_j^2 \geq 0$ this implies to $g_j(x) \leq 0$, *j=1,2,………………………..,m*

i. If all are satisfied, the current solution, is both feasible and optimal, stop.
ii. Otherwise, apply switching condition which equal 2^{m} and calculate,

$$\frac{\partial l(.)}{\partial s_j} = 2\mu_j s_j = 0, \ \ j = 1,2, \dots \dots \dots \dots \dots \dots \dots \dots, m$$

If $\mu_j \geq 0$, it implies minimum value of the cost function.

If $\mu_j < 0$, it implies maximum value of the cost function, go to switching condition.

Step5: Start with the optimal point, x^* and with $n \times n$ positive definite matrix, B_k.

Step6: Compute the gradient of the cost function, $\nabla f(x^k)$ at point x^k.

Step7: Determine the search direction, d_k.

$$d_k = -[B_k]\nabla f(x^k)$$

Step8: Compute the optimal step length, λ_k^* in the direction, d_k .

$$\lambda_k^* = -\frac{\nabla^T f\left(x^k\right).\, d_k}{{d_k}^T.\, \nabla^2 \nabla f(x^k).\, d_k}$$

Step9: Set $\quad x^{k+1} = x^k + \lambda_k^* \, d_k$

Step10: Test the point for optimality. If $||\nabla f(x^{k+1})\,|| \leq \varepsilon$, where ε is a small positive quantity.

Take $x^* \approx x^{k+1}$ and apply it to the prediction equation to get the next state.

$$X(k+1) = Ax(k) + Bu(k)$$

Where, $x^* = u(k)$.

Otherwise, go to Step11.
Step11: Update the Hessian matrix as

$$[B_{k+1}] = [B_k] + \frac{(\delta_k - B_k\gamma_k)(\delta_k - B_k\gamma_k)^T}{\gamma^T(\delta_k - B_k\gamma_k)}$$

Step12: Set the new iteration number as $k = k+1$ go to Step5.

3.IMPLEMENTATION OF MPC FOR SPEED CONTROL

In this section the representation of MPC for motor speed control utility is implemented in figure. 3. This representation mainly has two phases the first one is done in the MATLAB engine which contain the first four blocks from the flowchart which are:

- Develop MPC, in this step MPC is represented as a function it has three inputs and one output, internally contain the state space model of DC motor and the QP solver that used to solve the system optimization problem.
- Verifying design functionality, to ensure that the output of the MATLAB code is the same as the output of MATLAB simulation we take the output file from MATLAB code and draw it using MATLAB command program, this is done by using MATLAB command "load".

- Setting up model parameters with MATLAB coder to generate C code, in this two cases we use MATLAB coder to convert MATLAB code into C code to prepare the system algorithm that will be implemented on an FPGA board.

.

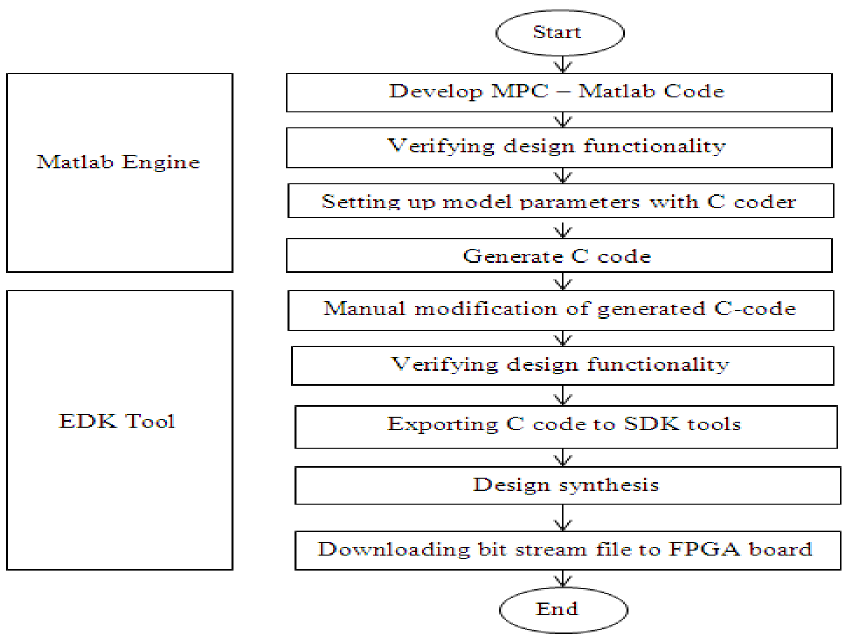

Figure. 3. Flowchart for realizing MPC algorithm in MATLAB code

The second one is done in the EDK tools which contain the last five blocks from the flowchart shown in figure. 3. as follow:

- Manual modification of the generated C code, we add some portions of C code for example open a file to write the output of the algorithm on it, also open another file to write the output of the controlled plant on it, after that print this data on the output of the console of the program to be ready to draw it as mentioned before with MATLAB output.
- Verifying design functionality, we take the output file of the C code and draw it by using the MATLAB command "load".

- Exporting C code to SDK tools, after adjusting the C code we export it into SDK tools to be ready to down load it on an FPGA board.

- Design synthesis and downloading bit stream file to FPGA board, the implemented C code is synthesized ensuring that the algorithm is true and generate embedded executable file (ELF), which like a hex file for some controllers, after that download this algorithm on the FPGA board to control the speed of the DC motor.

Part of the implementation is developed using MATLAB, and the other part is completed using Xilinx EDK tools supported by FPGA board as indicated in figure. 3.

3.1 System Modeling for Speed Control of DC Motor

The model of the DC motor is shown in figure. 4. This consists of a DC motor, shaft and a load. The system can be represented by the continuous time state space equations:

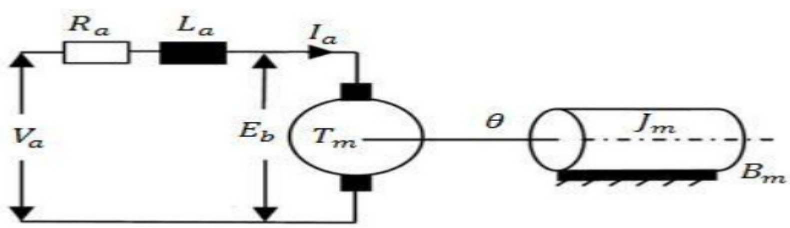

Figure. 4. DC motor model

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{k_b}{L_m} \\ \frac{k_t}{J_m} & -\frac{B_m}{J_m} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} \\ 0 \end{bmatrix} V_a(t) \qquad (9)$$

Based on the parameters setting in Table 1, the discrete linear time invariant system of state space model can be written as, where sampling time $T_s = 0.1$sec.

$$X(k+1) = \begin{bmatrix} -0.0001 & -0.0000 \\ 3.3864 & 0.9974 \end{bmatrix} x(k) + \begin{bmatrix} 0.0025 \\ 0.2594 \end{bmatrix} u(k) \qquad (10)$$

$$Y(k) = [0 \quad 1]x(k) + [0]u(k) \qquad (11)$$

The only measurement available for feedback is.x_2, which is the motor speed. Where the prediction horizon=10 and control horizon=3. The speed of the DC motor must be set at a desired value by adjusting the applied voltage $V_a(t)$ which is the motor input. The applied voltage must stay within the range $-24\text{v} \leq V_a \leq 24\text{v} \ \forall t$. The plant has a single input $V_a(t)$, which is manipulated by the MPC controller and single output $x_2(t)$, which is the motor speed that is fed back to the controller [24, 25].

3.2 MPC Representation for Speed Control of DC Motor

In order to improve control system's robustness and reduce time of execution process, as shown in figure. 5, the representation of MPC is designed as a function written and programmed using MATLAB. This function has three inputs (the reference point, actual motor speed and the optimized signal it called manipulated variable and one output which is the optimized signal.

Table 1. Description of the parameters and their corresponding

Symbol	Value	Definition
k_t	0.01 N-m/A	Motor constant
k_b	0.01 V/rad/sec	Back emf constant
J_m	9.277e-6 kg-m^2	Motor inertia
B_m	7.1e-10 N-m/rad/sec	Motor viscous friction coefficient
R_a	400 Ω	Armature resistance
l_a	1.26 H	Motor inductance

This function internally depends primarily on the optimization process that implemented using (QPs) solver of MATLAB. The implemented MPC is converted into C code using code generation to implement it on an FPGA board using EDK tools. And the output of the generated C code is tested and compared with resulted data of MPC simulator. Then the generated C code was implemented on the FPGA.

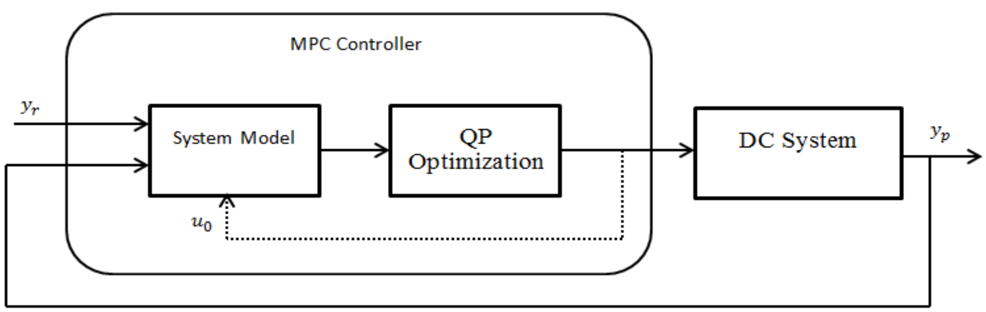

Figure. 5. MPC representation for speed control

4.GENERATION OF C CODE FROM MATLAB CODE

MATLAB coder is a new tool which comes with MATLAB software package can be used to generate C code. The coder brings the Model-Based Design approach into the domain of FPGA development. After the manual modification on the C code generated by the MATLAB coder the result show that the output of the C code is very closed to the MATLAB simulation. MPC algorithm take 46 clock cycle for PC computer which equivalent to 0.0046ms and the time needed by SDK tool when used to execute a constrained MPC algorithm that is implemented on FPGA to we can't compute it because the algorithm is floating point and it run offline to compute the control actions (signals) and online control is implemented as a lookup table. figure. 6 shows the comparison between the MATLAB simulation and the generated C code for the motor speed, the response of the generated C code is approximately the same as the response of MATLAB simulation.

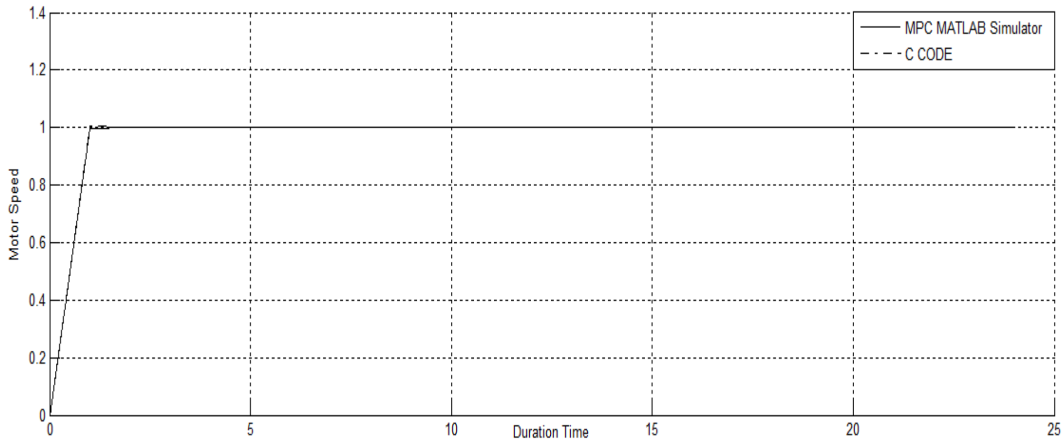

Figure. 6. Output response from MATLAB and C code

The comparison between the MATLAB simulation and the generated C code for the applied voltage signal is shown in figure. 7. The optimal signal of MATLAB code is greater than the generated C code approximately by one at each sample.

5.COMPARISON BETWEEN MPC AND PID CONTROLLERS

One of the most widely used algorithm in industrial control application is the PID controller due to their simple structures, extensive control algorithms and low cost. The essence of PID control is to compare the system output with the reference points and minimize the error depending on the tuning of its three parameters to compute the control signal which is applied to the plant [26, 27]. The response of the PID controller when used to control the speed of the DC motor is shown in figure. 8.The response of the MPC from the MATLAB code and MATLAB simulation is shown in figure. 9. The transient response of the two controllers can be obtained as shown in table 2. The rise time of the PID is smaller than MPC controller, over shoot of MPC is very small and almost equal to zero where PID over shoot is biggish and settling time of PID is larger than MPC controller.

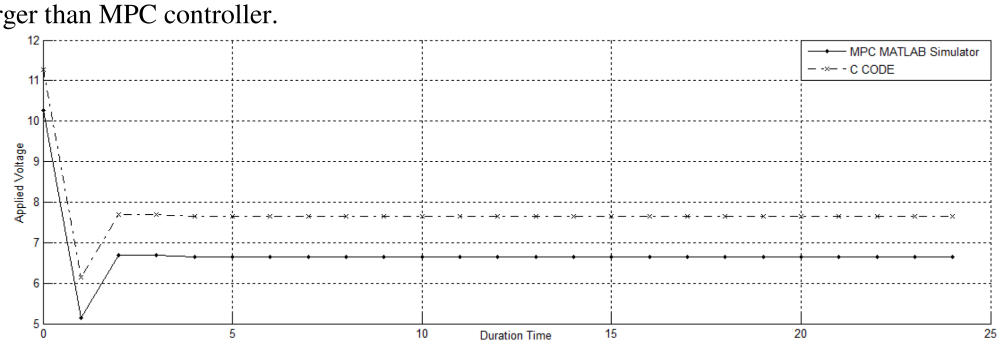

Figure. 7. applied voltage from MATLAB and C code

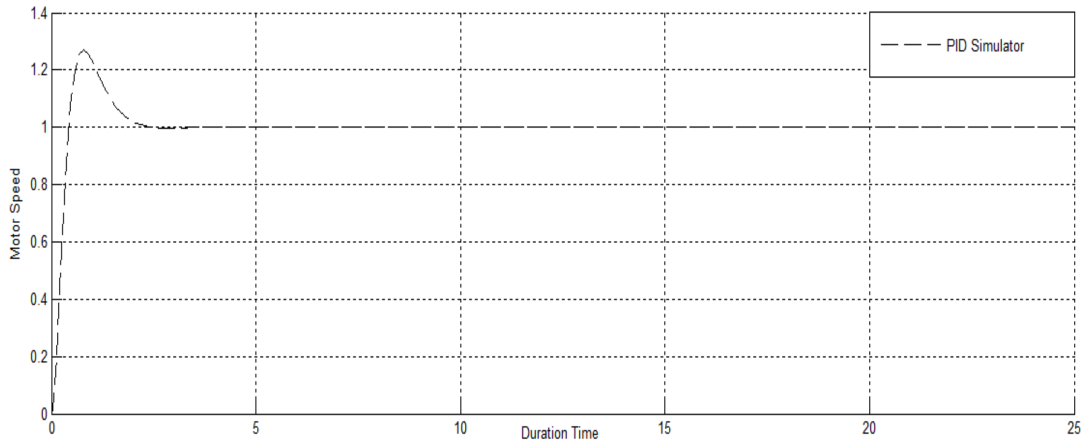

Figure. 8. Output response from MATLAB PID controller

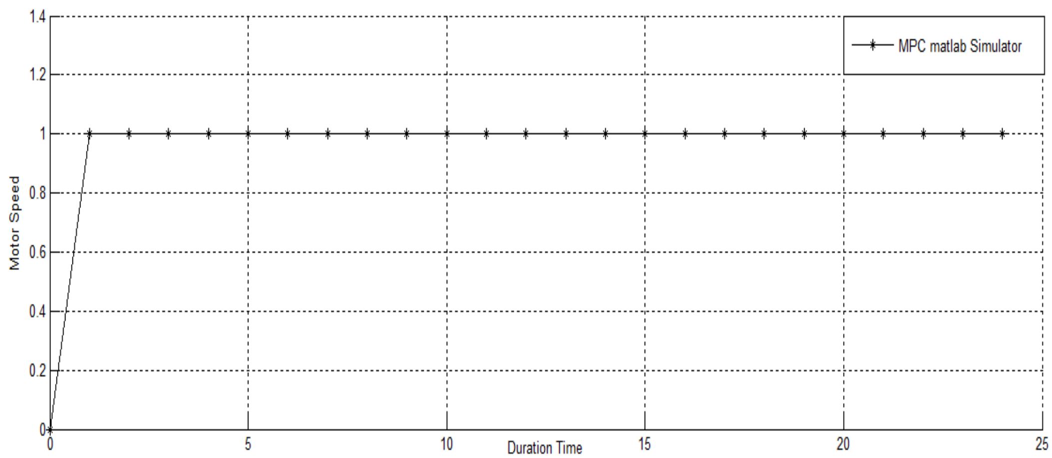

Figure. 9. Output response of the MPC from the MATLAB code

Table 2. Comparison between PID and MPC controller

controller	rise time sec	Settling time sec	peak time	maximum over shoot
PID	0.469	2.39	0.791	0.23
MPC	0.949	2	1.898	0.0014

6.FPGA IMPLEMENTATION

A complete microprocessor system based on an embedded processor MicroBlaze from Xilinx is built in the Xilinx Platform Studio (XPS) that is provided by EDK tools. The optimization problem needed to be solved at each sampling time so MPC controller is limited the application dynamic response. Therefore the concept of explicit MPC is used to compute the entire control action offline and the online controller can be implemented as lookup table in the FPGA board.

Using one output port from FPGA board to write the output of the algorithm which is the manipulated variable (the first signal from the optimal control vector signals generated by the optimizer) to the motor, write user constraint file (UCF) that is corresponding to the selected ports according to its pin numbers. Then generate bitstream corresponding to the hardware chosen and launch SDK to the software C code algorithm on the IDE eclipse framework developed by FPGA board. The logic resources utilization of the MPC algorithm that implemented on FPGA to solve the optimization problem is shown in Table 3.

Table 3. Resource utilization for MPC

Flip-flop	Look-up table	Dsp18 A1s	RAM block
1765/2503 (70%)	1931/9112 (21%)	3/32 (9%)	32/32 (100%)

The whole design is implemented on a spartn6 Nexys3 FPGA board as shown in figure. 9. The C code implementation of the MPC algorithm is compiled into a bitstream file which is then downloaded to the Spartan6 Nexy3 board to configure the FPGA chip to perform the constrained MPC calculations and it is tested on a speed control of DC motor.

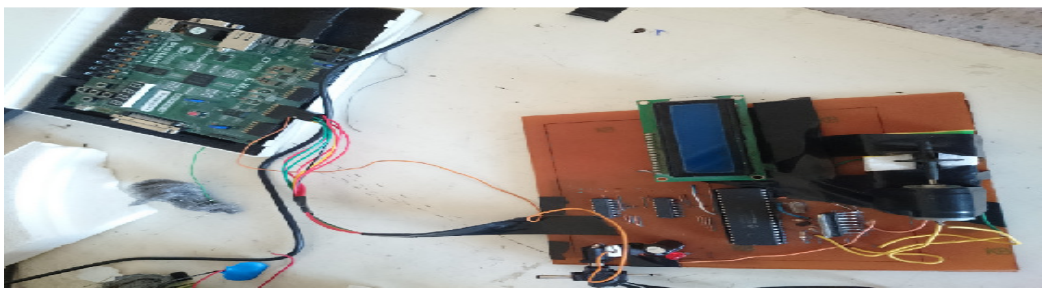

Figure. 9. MPC implementation on spartan6 nexys3 FPGA board

7.CONCLUSION

MPC controller is designed and implemented using FPGA board. A QP solver to system design is used to accelerate the optimization process in MPC algorithm it is much faster than traditional approaches. The MPC suggested algorithm is implemented to control DC motor speed. A comparison between MPC and PID controller is established, that indicate MPC is better than PID in both maximum overshoot and settling time and PID is better than MPC in rise time and peak time. The proposed MPC algorithm can achieve satisfactory performance when appropriate parameters are chosen, and hence this can be applied to large fields of industrial process controllers.

REFRENCESES

[1] S.Lucia, & S. Engell. (2013)" Robust nonlinear model predictive control of a batch bioreactorusing multi-stage stochastic programming", 2013 European Control Conference (ECC) July 17-19, Zürich, Switzerland.
[2] P. Bumroongsri & S. Kheawhom. (2012)"Robust constrained MPC based on nominal perfor-mance cost with applications in chemical processes", 20th International Congress of Chemical and Process Engineering CHISA 2012 25 – 29, Prague,Czech Republic, 1561 – 1571.
[3] M. Farrokhsiar & H. Najjaran, (2013)" A Robust probing motion planning scheme: A Tube-based MPC approach", American Control Conference (ACC) Washington, DC, USA, June 17-19.
[4] L. Wang, (2008)"Model predictive control system design and implementation using mat-lab", Advances in Industrial Control ISSN 1430-9491, Melbourne, Australia.
[5] T Zheng, (2010)" Model predictive control", Janeza Trdine 9,51000 Rijeka, Croatia, Sciyo.
[6] Ling, K.V, Yue, S.P, and Maciejowski, J.M. (2006), "A fpga implementation of model pred-ictive control", American Control Conference, Minnesota, USA.
[7] T. Zheng, (2012)"Frontiers of model predictive control", Published by InTech, Janeza, Trdine 9, 51000, Rijeka, Croatia.
[8] N. Yang, D. Li , J. Zhang & Y. Xi, (2011)" Model predictive control system based on fpga & A case study", Preprints of the 18th IFAC World Congress Milano Italy.
[9] K.V. Ling, B.F. Wu & J.M. Maciejowski, (2008)" Embedded model predictive control using a FPGA", Proceedings of the 17th World Congress the International Federation of Automatic Control, Seoul, Korea, July 6-11.
[10] E. K. H. Chen, (2012)" High-level abstractions for fpga-based control systems to improve usability and reduce design time", phd, Edward Kuan-Hua Chen Simon Fraser University Fall, Spring 2012.
[11] M. Leuer and J. Bocker, (2012)" Fast online model predictive control of IPMSM using parallel computing on FPGA", University of Paderborn, D-33095 Paderborn, Germany.
[12] R. Nema, R. Thakur & R. Gupta, (2013)" Design & Implementation of FPGA based on PID controller", International Journal of Inventive Engineering and Sciences (IJIES) ISSN: 2319–9598, Volume-1, Issue-2.
[13] Dr. rer. Nat, (2009)"MPC/LQG-Based optimal control of nonlinear parabolic PDEs",phd, Technische Universitat Chemnitz.
[14] Y. Yu, (2012)" Model-Based multivariate control of conditioining systems for office buil-dings", PH.D, Carnegie Mellon University.
[15] M. A. Hossain, (2013)" A Dual-Rate model predictive controller for fieldbus based dis-trbuted control systems", M.Science, University of Western Ontario, London, Ontario, Canada.
[16] C. Tricaud & Y.Q Chen, (2008)" Linear and nonlinear model predictive control using a general purpose optimal control problem solver RIOTS 95", IEEE, 978-1-4244-1734.
[17] Y.Wang & S. Boyd, (2010)" Fast model predictive control using online optimization", IEEE transactions on control systems technology, VOL. 18, NO. 2.
[18] F. Borrelli, A. Bemporad & M. Morari, (2011)" Predictive control for linear and hybrid systems", Control Book.
[19] S. Rao, (2009)"Engineering optimization", John Wiley & Sons, Inc.Book.
[20] J. Nocedal & S. J. Wright, (1999) "Numerical optimization", Springer-Verlag New York, Inc.Book
[21] C.Schimed & L. T. Bieglert," Quadratic programming methods for reduced hessian SQP", Computers them.Engng Vol. 18, No. 9, pp. 817432. 1994.
[22] D. Goldfarb & A. Idnani, (1983)" A numercally sable dual method for solving strictly convex quadratic programs", Mathematical Programming 27 (1983) 1-33, North-Holland.
[23] N Yang ,D. Li ,J Zhang & Y Xi, (2012)"Model predictive controller design and implementation on FPGA with application to motor servo system", Control Engineering Practice.
[24] C.Zhao & X.Zhan, (2008)" The application of fractional order pid controller to position servomechanism", Proceedings of the 7th,World Congress on Intelligent Control and Automation China.
[25] R. Singhal, S. Padhee & G.Kaur, (2012)" Design of fractional order pid controller for speed control of dc motor", International Journal of Scientific and Research Publications, Volume 2, Issue 6, ISSN 2250-3153.

[26] M. Kushwah & A. Patra, (2014)" Tuning pid controller for speed control of dc motor using soft computing techniques-a review", Advance in Electronic and Electric Engineering, SSN 2231-1297, Volume 4, Number 2, pp. 141-148.

[27] S. Dubey & S.K. Srivastava, (2013)" A pid controlled real time analysis of dc motor" International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization), Vol. 1, Issue 8.