2005 International Conference on Control and Automation (ICCA2005)
June 27-29, 2005, Budapest, Hungary

Model Predictive Control On A Chip*

Minghua He and Keck-Voon LING

Abstract— **Model Predictive Control (MPC) has become an established control technology in the petrochemical industry, and its use is currently being pioneered in an increasingly wide range of process industries. It is also being proposed for a range of higher bandwidth applications, such as ships, aerospace and road vehicles. To extend its applications to miniaturized devices and/or embedded systems, this paper explores the implementation of MPC technology into reconfigurable hardware such as a FPGA chip. A rapid prototyping environment suitable for exploring the various implementation issues to bring MPC onto a chip is described. Simulation tests were conducted to verify the applicability of the "MPC on a Chip" idea. It is shown that a modest FPGA chip could be used to implement a reasonably sized constrained MPC controller.**

Keywords: predictive control, constrained optimization, reconfigurable hardware, FPGA.

I. INTRODUCTION

Model Predictive Control (MPC) has become an established control technology in the petrochemical industry, and its use is currently being pioneered in an increasingly wide range of process industries. It is also being proposed for a range of higher bandwidth applications, such as ships [6], aerospace [12] [8], and road vehicles [11].

Fundamentally, MPC can be formulated as a quadratic programming (QP) problem. It thus has the natural ability to handle physical constraints arising in industrial applications. Alternatively called receding horizon control, MPC computes optimal current and future control inputs by minimizing the difference between set-points and future outputs predicted from a given plant model. Then only the optimal current input is applied to the plant and this procedure is repeated at the next sampling instance.

Two important factors determines a successful MPC applications. First, is the availability of a suitable plant model. The second, is the ability to solve the quadratic programming problem within the prescribed sampling period. The ability to solve the QP problem online become critical when applying MPC to complex systems with fast response time and/or embedded applications where computational resource may be limited. In the last decade, reconfigurable hardware is becoming a promising alternative to both ASIC and general purpose off-the-shelf processors for embedded applications[2], [3]. As a reconfigurable hardware, Field Programmable Gate Array, or FPGA, is gaining popularity. FPGA-based systems have been applied in applications ranging from signal processing, image processing, to network processors and robotics, just to name a few (see e.g., Andraka [1] and Tessier [9]). FPGA has better flexibility and shorter design cycle than ASIC.

In this paper, we report our attempt and the lessons learned in implementing the constrained MPC technology on a FPGA. Section 2 reviews the constrained MPC problem formulation and its solution. A rapid prototyping environment for developing the MPC algorithm for FPGA implementation is described in Section 3. The resulting MPC on a chip is demonstrated in Section 4 through a simulated aircraft control example. Finally, Section 5 concludes this paper.

II. REVIEW OF CONSTRAINED MPC AND ITS SOLUTION

Constrained MPC can be formulated as a QP problem. Given a discrete linear time-invariant plant in the state space form,

$$\Sigma : \begin{cases} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k), \end{cases} \tag{1}$$

where $y(k) \in \mathrm{R}^p$, $u(k) \in \mathrm{R}^m$ and $x(k) \in \mathrm{R}^n$ represent its system output, input and internal states respectively, the constrained MPC problem is to minimize the cost function,

$$\begin{aligned} &\Phi(y, \Delta u) \\ = &\sum_{j=1}^{N_\mathrm{p}} \|y(k+j) - \omega(k+j)\|_q^2 + \sum_{j=0}^{N_\mathrm{u}-1} \|\Delta u(k+j)\|_r^2 \end{aligned}$$

subject to inequality constraints,

$$y_{\mathrm{LB}} \le J_\mathrm{y} y \le y_{\mathrm{UB}}, \tag{2}$$

$$x_{\mathrm{LB}} \le J_\mathrm{x} x \le x_{\mathrm{UB}}, \tag{3}$$

$$u_{\mathrm{LB}} \le J_\mathrm{u} u \le u_{\mathrm{UB}}, \tag{4}$$

$$\hat{u}_{\mathrm{LB}} \le J_{\hat{\mathrm{u}}} \Delta u \le \hat{u}_{\mathrm{UB}}, \tag{5}$$

Here, ω is the set-point; N_p and N_u are the prediction and control horizons respectively.

For such a QP problem, Rao et. al. [7] formulated it in a sparse form containing both equality and inequality constraints. Block factorization was used to handle the resulting banded sparse matrix within the interior point method. According to ([5], page 93), although there are computational advantages in factorising a banded matrix with a fixed bandwidth compare to a dense matrix of the same size, this comparison should be done more carefully for

*This research was supported by A*STAR project "Model Predictive Control on a Chip" (Ref: 022-106-0044).

Minghua He and Keck-Voon LING are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798 {emhhe,ekvling}@ntu.edu.sg

any particular applications. This is because the outcome of the comparison would be affected by many factors such as how many constraints that are likely to be active typically. Here, we adopted a compact formulation by eliminating the equality constraints of (1) through replacing predicted system output y as,

$$y = \bar{C}\Psi_{\rm x}x(k) + \bar{C}\Psi_{\rm u}u, \tag{6}$$

with

$$\bar{C} = \begin{bmatrix} C & & & \\ & C & & \\ & & \ddots & \\ & & & C \end{bmatrix}, \quad \Psi_{\rm x} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N_{\rm P}} \end{bmatrix},$$

$$\Psi_{\rm u} = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_{\rm P}-1}B & A^{N_{\rm P}-2}B & \cdots & A^{N_{\rm P}-N_{\rm u}}B \end{bmatrix}.$$

In addition, noting that

$$\Delta u = \Theta u - g_0, \tag{7}$$

with

$$\Theta = \begin{bmatrix} I_m & 0 & 0 & \cdots & 0 \\ -I_m & I_m & 0 & \cdots & 0 \\ & & & & \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -I_m & I_m \end{bmatrix}$$

$$g_0 = J_{\hat{\rm u}} \begin{bmatrix} u(k-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{8}$$

the cost function in (2) can be re-arranged as

$$\Phi(u) = \frac{1}{2}u^{\rm T}Qu + c^{\rm T}u + \Phi_0. \tag{9}$$

Furthermore, the inequality constraints (2)–(5) can be written compactly as

$$Ju \le g, \tag{10}$$

where

$$J = \begin{bmatrix} J_{\rm y}C\Psi_{\rm u} \\ -J_{\rm y}C\Psi_{\rm u} \\ J_{\rm x}\Psi_{\rm u} \\ -J_{\rm x}\Psi_{\rm u} \\ J_{\rm u} \\ -J_{\rm u} \\ J_{\hat{\rm u}}\Theta \\ -J_{\hat{\rm u}}\Theta \end{bmatrix}, \quad g = \begin{bmatrix} y_{\rm UB} - J_{\rm y}C\Psi_{\rm x}x(k) \\ -y_{\rm LB} + J_{\rm y}C\Psi_{\rm x}x(k) \\ x_{\rm UB} - J_{\rm x}\Psi_{\rm x}x(k) \\ -x_{\rm LB} + J_{\rm x}\Psi_{\rm x}x(k) \\ u_{\rm UB} \\ -u_{\rm LB} \\ \hat{u}_{\rm UB} + J_{\hat{\rm u}}g_0 \\ -\hat{u}_{\rm LB} - J_{\hat{\rm u}}g_0 \end{bmatrix}. \tag{11}$$

In this way, the constrained MPC problem is formulated as a compact QP problem in (9) and (10) where Q is a $N_{\rm u}m \times N_{\rm u}m$ matrix and J is $m_{\rm c} \times N_{\rm u}m$ in size. Here $m_{\rm c}$ is the total number of inequality constraints on u.

Active set [4] and Interior Point method [10] are two popular methods used to solve QP problems. In this paper, we adopted the infeasible interior point method for our FPGA implementation. The basic idea of the infeasible interior point method for solving a QP problem follows from the Karush-Kuhn-Tucker (or KKT) conditions,

$$Qu + J^{\rm T}\lambda = -c, \tag{12}$$

$$-Ju - t = -g, \tag{13}$$

$$\lambda \ge 0, \quad t \ge 0, \quad t^{\rm T}\lambda = 0. \tag{14}$$

Equations (12) and (13) are called feasibility conditions while the equation (14) is called complementary condition.

The infeasible interior point method perturbs the complementary condition in (14) with the following scalar

$$\mu^k = (t^k)^{\rm T}\lambda^k/m_{\rm c}, \tag{15}$$

where k is the iteration sequence and $m_{\rm c}$ is the number of inequality constraints in (10). When the iteration goes on, the infeasibility and μ^k are gradually reduced to zero.

According to the infeasible interior point framework introduced by Wright [10], an optimal control signal can be computed by solving the QP problem using the following algorithm:

Step 1:
Choose an initial condition (u^0, λ^0, t^0) with $(\lambda^0, t^0) > 0$.

Step 2:
At the k-th iteration step, solve for the increments $(\Delta u^k, \Delta\lambda^k, \Delta t^k)$ with

$$\begin{bmatrix} Q & J^{\rm T} \\ J & \Gamma \end{bmatrix} \begin{bmatrix} \Delta u^k \\ \Delta\lambda^k \end{bmatrix} = \begin{bmatrix} r_1^k \\ r_2^k \end{bmatrix}, \tag{16}$$

and

$$\Delta t^k = -t^k + (\Lambda^k)^{-1}(\sigma^k\mu^k e - T^k\Delta\lambda^k). \tag{17}$$

Here Q is a symmetric matrix, $\Gamma = -(\Lambda^k)^{-1}T^k$,

$$\Lambda = \begin{bmatrix} \lambda_1^k & & \\ & \ddots & \\ & & \lambda_{m_{\rm c}}^k \end{bmatrix}, \quad T = \begin{bmatrix} t_1^k & & \\ & \ddots & \\ & & t_{m_{\rm c}}^k \end{bmatrix}, \quad e = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{18}$$

In addition,

$$\begin{aligned} r_1^k &= -Qu^k - J^{\rm T}\lambda^k - c, \\ r_2^k &= Ju^k - g - \sigma^k\mu^k(\Lambda^k)^{-1}e. \end{aligned} \tag{19}$$

Step 3:
Increment the variables by

$$(u^{k+1}, \lambda^{k+1}, t^{k+1}) = (u^k, \lambda^k, t^k) + \alpha^k(\Delta u^k, \Delta\lambda^k, \Delta t^k), \tag{20}$$

for some $\alpha^k \in (0, 1]$ subject to $(\lambda^{k+1}, t^{k+1}) > 0$.

Step 4:
Judge the convergence. If the iterations converges, stop the process and the optimal control u^{k+1} is obtained; otherwise, go back to Step 2 with $(u^{k+1}, \lambda^{k+1}, t^{k+1})$ and continue the iteration process.

Remark 2.1: It can be seen that solving such a QP problem is an iterative process and the main computational load is in solving equation (16) at each iteration.

Equation (16) can be solved either as

$$\left\{\begin{array}{rcl}\Delta u^k &=& Q^{-1}r_1^k - Q^{-1}J^{\mathrm{T}}\Delta\lambda^k \\ \Delta\lambda^k &=& (\Gamma - JQ^{-1}J^{\mathrm{T}})^{-1}(r_2^k - JQ^{-1}r_1^k),\end{array}\right. \tag{21}$$

or

$$\left\{\begin{array}{rcl}\Delta\lambda^k &=& \Gamma^{-1}r_2^k - \Gamma^{-1}J\Delta u^k \\ \Delta u^k &=& (Q - J^{\mathrm{T}}\Gamma^{-1}J)^{-1}(r_1^k - J^{\mathrm{T}}\Gamma^{-1}r_2^k).\end{array}\right. \tag{22}$$

Both schemes above need to inverse a matrix. For (21), the matrix to be inverted at each iteration is $\Gamma - JQ^{-1}J^{\mathrm{T}}$, which has dimensions m_{c} by m_{c}. In (22), the matrix is $Q - J^{\mathrm{T}}\Gamma^{-1}J$, which has dimensions $N_{\mathrm{u}}m$ by $N_{\mathrm{u}}m$. $\Gamma - JQ^{-1}J^{\mathrm{T}}$ has a special structure in that only the diagonal elements change at each iteration. In most practical MPC applications, m_{c} is generally much larger than $N_{\mathrm{u}}m$. In such a situation, (22) is more attractive for FPGA implementation and is adopted for the rest of the paper.

III. A PROTOTYPING ENVIRONMENT FOR MPC ON A CHIP

The main factors to be considered when implementing MPC on reconfi gurable hardware include computational speed, hardware resource usage, power consumption, etc. For a particular application, specifi c requirements on these factors need to be met and the fi nal implementation is usually a compromise between all these factors. Hence, an effective and effi cient rapid prototyping environment which allows for experimentation and verifi cation of various algorithm confi gurations, architecture and implementation schemes would be useful. The tools we employed in achieving our “MPC on a chip” includes a RC200 FPGA prototyping board, the DK3 Design Suite from Celoxica and Matlab/Simulink software from Mathworks.

The core of RC200 is a Xilinx Virtex II (XC2V1000-4-fg456) FPGA chip which has one million logic gates and some useful on-chip resources such as multiplier and on-chip memory. Celoxica DK3 design suite is an integrated environment for FPGA implementation using the Handel-C programming language. It provides a complete tool set which includes a compiler, a debugger, an optimizer and a simulator. MATLAB/Simulink provides an excellent platform for plant modelling, MPC algorithm design and simulation, Handel-C/MATLAB co-simulation and hardware-in-the-loop verifi cation.

Handel-C is a high level FPGA implementation language with an ANSI C syntax and some hardware related language features such as parallel execution, channel communication, interface defi nition, etc. Compared with other hardware description languages such as VHDL or Verlog, Handel-C is more convenient for rapid prototyping of control-centric algorithms.

The main procedure of prototyping of our “MPC on a Chip” design is illustrated in Figure 1.

In the following, we briefly describe main steps in prototyping MPC to a FPGA implementation.

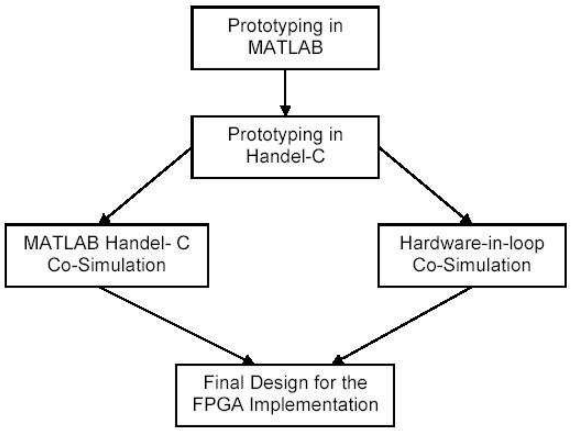
Prototyping in MATLAB
Prototyping in Handel-C
MATLAB Handel- C Co-Simulation
Hardware-in-loop Co-Simulation
Final Design for the FPGA Implementation

Fig. 1. Prototyping of MPC on a Chip

Step 1: Prototyping in MATLAB Code
MATLAB provides an excellent computation and simulation environment for designing and implementing control algorithms. The MPC algorithm is fi rst prototyped in MATLAB code and then simulated and verifi ed in the MATAB/SIMULINK environment.

Step 2: Prototyping in Handel-C Code
The prototype MPC in the form of MATLAB code is translated into Handel-C code for FPGA realisation. The code is then compiled and optimized in the DK3 design suite. It is mapped, placed and routed by Xilinx ISE to a target FPGA. The Xilinx tool would report hardware resource usage and timing performance. If the results do not meet the specifi ed requirements, design iterations would need to be carried out.

Step 3: Handel-C/MATLAB Co-Simulation
Two options are available for algorithm verifi cation: software or hardware verifi cation. For software verifi cation, the Handel-C code could be packaged into a DLL fi le and then be called by Simulink as a S-function (see Fig. 2).

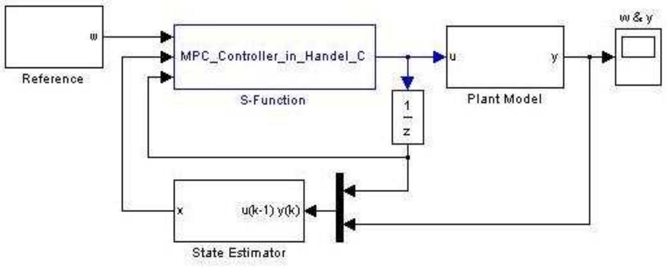

Fig. 2. Simulink/Handel C Co-Simulation

Step 4: Hardware-in-the-loop Verifi cation
For hardware verifi cation, the Handel-C code can be compiled into a bitstream fi le which would confi gure the FPGA as a “MPC on a chip”. The bitstream fi le can be downloaded to the RC200 prototyping board and a test suite can be written in MATLAB script to verify the MPC implementation on the FPGA. Test data and test results can be transferred between MATLAB on the PC and FPGA on the RC200 board

through the RS232 serial link (see Fig. 3). Table I shows a typical MATLAB test suite. If there is any error in the hardware-in-the-loop verification, it would be trapped and investigated.

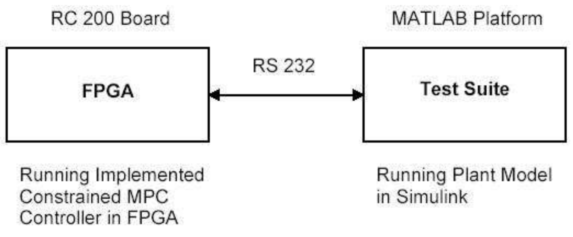
RC 200 Board
MATLAB Platform
FPGA
RS 232
Test Suite
Running Implemented Constrained MPC Controller in FPGA
Running Plant Model in Simulink

Fig. 3. Hardware-in-the-loop Verification

TABLE I
A SAMPLE TEST SUITE FOR FPGA IMPLEMENTATION OF CONSTRAINED MPC

```
clear all

TOL = 1e-3; %acceptable accuracy
N = 50;     %how many test
n = 6;      %size of QP
mc = 32;
...

for i = 1:N
  % generate a QP program randomly
  H = rand(n,n); Q = H'*H;
  ...

  u_qp = quadprog(Q,c,J,g); %use MATLAB's QP solver

  disp('Download program and data to RC200 board...')
  for i=1:n
    for j=1:n, fwrite(s,Q(i,j),'single'); end
  end
  ...

  disp('Read back results from RC200 board...')
  for i=1:n, u_RC200(i) = fread(s,1,'single'); end
  ...

  err = max(abs(u_qp-u_ip));

  if (err > TOL)
    disp('-----Error!----'), break
  end
end
```

IV. RESULTS AND DISCUSSIONS

Although FPGA implementation of MPC is highly application dependent, there exists some common core components. The first, is an efficient floating point library provided by the DK3 design suite. Next, is a matrix inversion core. Our experience showed that a one million gate Virtex II FPGA could easily handle a 128x128 matrix inversion problem with IEEE single precision floating point arithmetic (see Table II). Better result may be achieved with further optimisation.

When using the interior point method to solve the constrained MPC problem, the solution depends on the precision of the floating point arithmetic and the criteria used in

TABLE II
FPGA IMPLEMENTATION OF 128X128 MATRIX INVERSION USING IEEE SINGLE PRECISION FLOATING POINT ARITHMETIC

Slice	LUT	FF	Block Memory
2037 (39%)	3220 (31%)	1100 (10%)	32 (80%)
System Clock	Maximum Delay	Clock Cycles	Execute Time
30MHz	1.183ns	38,528,305	1.2843s

the convergence test. From our experience, $\mu < 10^{-5}$ appeared to work well. Table III shows an implementation of constrained MPC of size $N_{\mathrm{u}}m = 45$ and $m_{\mathrm{c}} = 128$. It was implemented on a Virtex II (XC2V1000-4-fg456) FPGA chip with IEEE single precision floating point arithmetic. It can be seen that this one million gate FPGA could be used to implement a reasonable size MPC comfortably. Better result may be achieved with further optimisation.

TABLE III
IMPLEMENTATION RESULTS OF MPC ON AN FPGA CHIP

Slice	LUT	FF	Block Memory
3826 (74%)	7028 (68%)	1708 (16%)	38 (95%)
System Clock	Maximum Delay	Clock Cycles	Execute Time
28MHz	1.188ns	65,966,362	2.3559s

V. APPLICATION OF MPC ON A CHIP TO AN AIRCRAFT EXAMPLE

To verify our MPC on a chip, we test it using the Cessna Citation 500 aircraft model from [5]. It has the following continuous-time state space form

$$A = \begin{bmatrix} -1.2822 & 0 & 0.98 & 0 \\ 0 & 0 & 1 & 0 \\ -5.4293 & 0 & -1.8366 & 0 \\ -128.2 & 128.2 & 0 & 0 \end{bmatrix}, \; B = \begin{bmatrix} -0.3 \\ 0 \\ -17 \\ 0 \end{bmatrix},$$
$$C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -128.2 & 128.2 & 0 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{23}$$

The model has the elevator angle (rad) as its input, and the pitch angle (rad), altitude (m) and altitude rate (m/s) as outputs.

A MPC controller was designed with a sampling interval of 0.5s, $N_{\mathrm{p}} = 10$, $N_{\mathrm{u}} = 3$. The following constraints,

$$|u| \le 0.262, \quad |\Delta u| \le 0.524, \quad |y_1| \le 0.349.$$

were also included. Hence, the total number of constraints is $m_{\mathrm{c}} = 32$ and $N_{\mathrm{u}}m = 3$.

Table IV shows the implementation result using the IEEE single precision arithmetic (8-bit exponent and 23-bit mantissa). It can be seen from Table IV that the FPGA implementation of MPC met the prescribed sampling time of 0.5s comfortably.

A lower precision MPC would also work for this example. For comparison, Table V shows a similar MPC implementation with a slightly lower precision using 18 bits for mantissa and 9 bits for exponent. It can be seen that the lower

TABLE IV

IMPLEMENTATION RESULTS WITH (8,23) FLOATING POINT ARITHMETIC (IEEE SINGLE PRECISION)

Slice	LUT	FF	Block Memory
3707 (72%)	6817 (66%)	1629 (15%)	17 (42%)
System Clock	Maximum Delay	Clock Cycles	Execute Time
28MHz	1.187ns	171,460	0.0061s

precision implementation consumed less hardware resource because fewer bits were used to represent oating point numbers. In addition, since there are numerous 18×18 on-chip hardware multipliers on the chosen FPGA, this implementation is likely to benefi t from the more effi cient use of the available on-chip resources if further optimisation is carried out.

TABLE V

FPGA IMPLEMENTATION RESULTS WITH (9,18) FLOATING POINT ARITHMATIC

Slice	LUT	FF	Block Memory
3301 (64%)	5916 (57%)	1596 (15%)	17 (42%)
System Clock	Maximum Delay	Clock Cycles	Execute Time
28MHz	1.187ns	171,460	0.0061s

To verify, we downloaded the MPC implementation onto the RC200 board. The aircraft model was simulated in MATLAB/SIMULINK on a PC and controlled by the FPGA implementation of MPC on the RC200 board. The controller and plant interacted through the RS232 serial link and satisfactory hardware-in-the-loop simulation result was obtained as shown in Fig. 4.

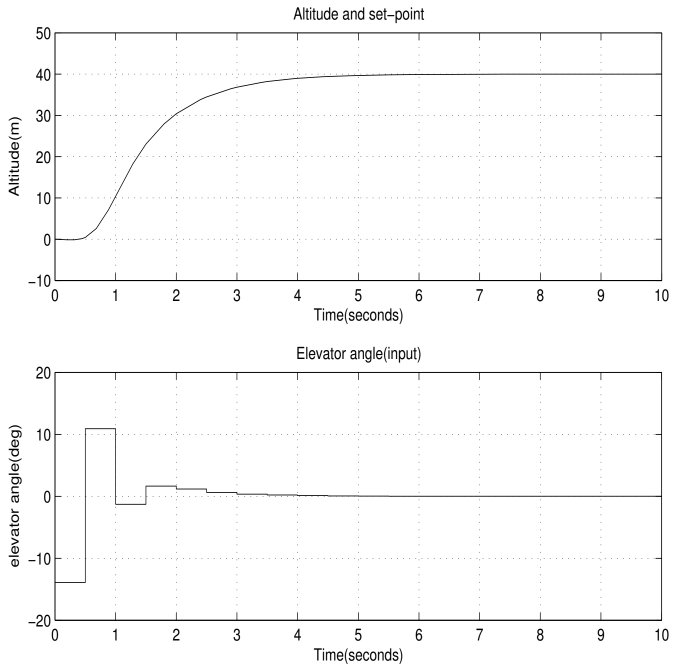

Fig. 4. Aircraf example, hardware-in-the-loop simulation results.

VI. CONCLUSIONS

In this paper, we explored the implementation of constrained MPC algorithm using a FPGA chip. Interior point method, with dense matrix formulation, was employed to solve the resulting QP problem. A rapid prototyping environment suitable for exploring the various implementation issues to bring MPC onto a FPGA chip was described. Simulation tests were conducted to verify the applicability of the MPC on a Chip idea. It was shown that a modest FPGA chip could be used to implement a reasonably sized constrained MPC controller. Further work is needed to investigate the possible parallelising of MPC computations to take advantage of the available on-chip resources on the FPGA chip.

REFERENCES

[1] R. Andrak, A survey of CORDIC algorithms for FPGA based computers , *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*, March 1998, pp. 191-200

[2] M. Baleani, F. Gennari, Y. Jiang, Y. Patel, R. K. Brayton, A. Sangiovanni-Vincentelli, System partitioning and timing analysis: HW/SW partitioning and code generation of embedded control applications on a reconfi gurable architecture platform , *Proceedings of the tenth international symposium on Hardware/software codesign*, May 2002, pp.151-156.

[3] K. Compton and S. Hauck, Reconfi gurable computing: a survey of systems and software , *ACM Computing Surveys (CSUR)*, Vol. 34, Issue 2, June 2002, pp.171-210.

[4] R. R. Fletcher, *Practical Methods of Optimization*, Wiley, 2nd Edition, 1987.

[5] J. M. Maciejowski, *Predictive Control with Constraints*, Prentice Hall, 2002.

[6] T. Perez, G.C. Goodwin and C.W. Tzeng, Model predictive rudder roll stabilization control for ships , In: *Proc. 5th IFAC Conf. on Manoeuvring and Control of Marine Craft*, Aalborg, Denmark, 2000.

[7] C. V. Rao, S. J. Wright and J. B. Rawlings, Application of interior-point methods to model predictive control, *Journal of Optimization Theory and Applications* , Vol. 99, 1998, pp. 723-757.

[8] A. Richards and J.P. How., Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility , In: *Proc. American Control Conference*, Denver, 2003.

[9] R. Tessier and W. Burleson, Reconfi gurable computing for digital signal processing: a survey , *Journal of VLSI Singnal Processing*, Vol. 28, 2001, pp. 7-27

[10] S. J. Wright, Applying new optimizaiton algorithms to model predictive control, *Chemical Process Control-V*, CACHE, AIChE Symposium Series No. 316, Vol. 93, 1997, pp. 147-155.

[11] M. Morari, M. Baotić and F. Borrelli, "Hybrid systems modeling and control", *European Journal of Control*, Vol. 9, 2003, pp.177-27.

[12] R.M. Murray, J. Hauser, A. Jadbabie, M.B. Milam, N. Petit, W.B. Dunbar and R. Franz, "Online control customization via optimization-based control", In: *Software-Enabled Control* (T. Samad and G. Balas, Eds.). IEEE Press and Wiley, 2003.