2013 European Control Conference (ECC)
July 17-19, 2013, Zürich, Switzerland.

Predictive Control for Spacecraft Rendezvous in an Elliptical Orbit using an FPGA

Edward N. Hartley and Jan M. Maciejowski

Abstract— **A field programmable gate array (FPGA)-based predictive controller for a spacecraft rendezvous manœuvre is presented. A linear time varying prediction model is used to accommodate elliptical orbits, and a variable prediction horizon is used to facilitate finite time completion of manœuvres. The resulting constrained optimisation problems are solved using a primal dual interior point algorithm. The majority of the computational demand is in solving a set of linear equations at each iteration of this algorithm. To accelerate this operation, a custom circuit is implemented, using a combination of Mathworks HDL Coder and Xilinx System Generator for DSP, and used as a peripheral to a MicroBlaze soft core processor. The system is demonstrated in closed loop by linking the FPGA with a simulation of the plant dynamics running in Simulink on a PC, using Ethernet.**

I. Introduction

IMPLICIT generation of a control law through solution of a constrained optimisation problem makes model predictive control (MPC) particularly suitable for control scenarios where reconfiguration is necessary. The prediction model, cost function and constraints can be updated online to reflect changes in system parameters or control objectives. In [1]–[4], linear time-varying (LTV) prediction models are used to perform spacecraft manœuvres in elliptical orbits. Simulations show that fuel consumption using MPC is favourable in comparison to other methods. Variable horizon predictive control (VH-MPC) is a variant of MPC, suited to "completion" type control problems, such as aerial vehicle manœuvres [5] and spacecraft rendezvous [4], [6], [7] due to its finite-time completion properties. Exploiting this, [4] presents a family of VH-LTV-MPC controllers to perform a spacecraft rendezvous operation, with a controlled chaser spacecraft performing rendezvous with a passive target in an orbit around Mars [8].

VH-LTV-MPC has a high computational burden, and a contemporary guidance, navigation and control computer would require a dedicated co-processor to support the MPC. Instead of using an extra high-performance general purpose microprocessor, an alternative would be to employ a custom peripheral circuit, designed to carry out all or part of the predictive control function, trading high clock frequencies (problematic in space applications due to radiation exposure) for parallelism, and offering opportunities to reduce power consumption.

This work was supported by the EPSRC (Grant EP/G030308/1) as well as industrial support from Xilinx, Mathworks and the European Space Agency.

The authors are with the University of Cambridge (email: {enh20,jmm}@eng.cam.ac.uk).

A. Background: FPGA based predictive control

Field programmable gate arrays (FPGAs) are configurable silicon devices that enable a system designer to implement custom circuits without the need to produce custom chips.

Recently there has been interest in using FPGAs to exploit opportunities for parallelism in the numerical algorithms used to implement MPC, with a variety of approaches proposed. A hybrid software-hardware design is advocated by [9]–[11]. The first uses a primal barrier method and the custom accelerator is used to calculate the gradient vector and Hessian matrix of the augmented cost function at each iteration, whilst the latter two use an active set method, with a custom circuit accelerating matrix/vector-vector multiplication. Full hardware implementations of MPC controllers are documented in [12] (active set), [13]–[16] (interior point), although [16] uses a soft-core microprocessor to bridge between the MPC and the outside world. At the other end of the spectrum, [17] implements MPC using a custom soft-core processor with non-standard numerical representations to minimise the FPGA resource (and power) requirements. With the exception of [15], [16], systems with one or two inputs are considered, and no prior FPGA-based predictive controllers consider variable prediction horizons and time-varying prediction models.

B. Summary

This paper presents an FPGA-based implementation of a variable horizon predictive controller, for a phase of the spacecraft rendezvous application in an elliptical orbit, based on a simplified version of one modes of operation from [4]. The design is implemented and tested on a Xilinx ML605 Evaluation Board [18], using Ethernet to communicate with a simulation of the spacecraft dynamics running in Simulink on a PC. Resource usage and power consumption are analysed.

An uncondensed MPC form is used. This scales favourably in terms of prediction horizon [19] and is advantageous for model reconfiguration, since once the time varying state space matrices are computed, the remaining effort in constructing the optimisation problem is data transfer rather than the multiplication of large, dense matrices.

A primal-dual interior point (PD-IP) algorithm [20] is implemented to solve the constrained optimisation problems. Following the hybrid software-hardware paradigm, the algorithm is co-ordinated by a Xilinx MicroBlaze soft-core processor, and a custom peripheral circuit is designed to assist in the construction and to perform the solution of the set of linear equations that arise at each iteration of the PD-IP algorithm. This is the bottleneck in optimisation problems

of the scale considered. In contrast to prior designs, where register transfer languages (RTL) such as Verilog [10], VHDL [12], [14]–[16], or high level C-like languages (System-C, [13]) have been used, the custom peripheral presented here is designed using a combination of Mathworks HDL Coder 2012a and Xilinx System Generator for DSP 14.2. This gives almost the same fine-grained clock-cycle accurate control of the design that an RTL would offer, but, being based upon Simulink, enables high-level visualisation of connectivity between components and rapid simulation.

The remainder of the paper is organised as follows: Section II develops the control problem; Section III describes the FPGA-based implementation; Section IV analyses the FPGA resource usage and power consumption; Section V describes the test-bench setup and presents closed-loop performance results; and Section VI concludes.

II. Predictive control problem

The scenario considered here is the medium-short range phase of the rendezvous in the Mars Sample Return mission [4], [8], starting at the point where the controlled spacecraft (chaser) has been brought into approximately the same orbit as the spacecraft with which it must rendezvous (target), but with a significant separation in true anomaly (i.e. an in-track separation of a few tens of kilometres). The control objective is to guide the chaser towards the vicinity of the target via a sequence of pre-determined "holding points" at which it must remain until given clearance. The time spent at holding points is unknown *a priori*, so at a given instant the goal is to enter the next holding point. The scenario terminates once the final holding point has been reached.

The Yamanaka-Ankersen (Y-A) equations [21] describe the dynamics of the relative position and velocity of the chaser with respect to the target in a local-vertical local-horizontal (LVLH) reference frame centred on the target, with the z-axis pointing towards the central body, the y-axis normal to the orbital plane, and the x-axis completing a right-handed set. Unlike the LTI Clohessey-Wiltshire (C-W) equations (e.g. [22]), these apply to elliptical orbits, but are parameter varying in terms of the true anomaly, ν of the target. In the MSR capture scenario, the target is passive, so the Y-A equations are LTV, since ν is only a function of time.

As in [4], [6], a 1-norm cost is used to reflect propellant consumption being directly proportional to delivered thrust. A variable control horizon is used to enable a compromise between completion time and propellant usage. Letting $x_i \in \mathbb{R}^{n_x}$ and $u_i \in \mathbb{R}^{n_u}$ denote the predicted state and control input i time steps into the future, $x_T \in \mathbb{R}^{n_x}$ denote a target setpoint, $x(k)$ denote the actual measured (estimated) state at time $k \in \mathbb{Z}_+$, $\gamma > 0 \in \mathbb{R}$ be a weighting parameter, and $\theta(k) \triangleq (x_0, u_0, \ldots, x_N)$, the optimisation problem posed is

$$J^* = \min_{N,\theta} \textstyle\sum_{i=0}^{N-1} (1 + \gamma \|u_i\|_1) \tag{1a}$$

$$\text{s.t.} \quad x_0 = x(k), \tag{1b}$$

$$x_{i+1} = A_i x_i + B_i u_i, \quad i \in \{0, \ldots, N-1\}, \tag{1c}$$

$$x_N = x_T(k+N), \tag{1d}$$

$$0 \le u_i \le u_{\max}, i \in \{0, \ldots, N-1\}, \tag{1e}$$

$$N \le N_{\max}, \tag{1f}$$

where A_i denotes the Y-A state transition matrix corresponding to a prediction from time step i to $i+1$. The plant input is modelled as an impulsive change in velocity, partitioned into positive and negative components aligned with the three axes, so $B_i = A_i \begin{bmatrix} 0 & 0 \\ I_3 & -I_3 \end{bmatrix}$. Control $u(k) = u_0$ is applied to the plant, the rest of $\theta(k)$ is discarded and the process is repeated at the next sampling instant. To improve numerical conditioning of the matrices within the PD-IP algorithm, the state and input vectors are scaled so that relative positions are in units of 20 m, velocities in units of $1/60$ m/s and inputs in units of $1/330$ m/s.

The variable horizon problem (1) can be posed as a sequence of $N_{\max}$ linear programs (LPs), each with fixed N. An LP is QP with a zero Hessian matrix, and the PD-IP algorithm for QP does not require a strictly positive definite Hessian matrix. Therefore, to facilitate future component reuse (for example for trajectory tracking during the final few metres of the rendezvous, for which a quadratic cost is more appropriate), a QP solver is implemented. The QP representation of (1), with N fixed is

$$\min_{\theta} \frac{1}{2} \theta^T H \theta + h^T \theta \tag{2}$$

subject to $F\theta = f$ and $G\theta \le g$ where, letting $\oplus$ denote the direct matrix sum, $\otimes$ denote the Kronecker product, I_p denote the p-dimensional identity matrix, $\mathbf{0}_{q\times r}$ denote a $q \times r$ matrix of zeros and $\mathbf{1}_{q\times r}$ denote a $q \times r$ matrix of ones, $H = \left(\bigoplus_{i=0}^{N-1} H_i\right) \oplus H_N$, $h = \begin{bmatrix} \mathbf{1}_N \otimes h_i \\ h_N \end{bmatrix}$, $F = \begin{bmatrix} I_N \otimes \begin{bmatrix} -I_{n_x} & \mathbf{0}_{n_{xu}} \end{bmatrix} & \mathbf{0}_{Nn_x \times n_x} \\ \mathbf{0}_{n_x \times Nn_{xu}} & -I_{n_x} \\ \mathbf{0}_{n_T \times Nn_{xu}} & F_N \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{n_x \times N(n_{xu})} & \mathbf{0}_{n_x} \\ \bigoplus_{i=0}^{N-1} \begin{bmatrix} A_i & B_i \end{bmatrix} & \mathbf{0}_{Nn_x \times n_x} \\ \mathbf{0}_{n_T \times Nn_{xu}} & \mathbf{0}_{n_T \times n_x} \end{bmatrix}$, $f = \begin{bmatrix} -x(k) \\ \mathbf{0}_{Nn_x \times 1} \\ f_N \end{bmatrix}$, $G = \left[\left(\bigoplus_{i=0}^{N} G_i\right), \mathbf{0}_{Nn_c \times n_x}\right]$, $g = \left[\mathbf{1}_N \otimes g_i\right]$. In (1), with fixed N, every $H_i = \mathbf{0}_{n_{xu} \times n_{xu}}$, $h_i = \begin{bmatrix} \mathbf{0}_{1\times n_x} & \gamma \mathbf{1}_{1\times n_x} \end{bmatrix}^T$, $F_N = I_{n_x}$, $f_N = x_N$, $G_i = \begin{bmatrix} \mathbf{0}_{n_u \times n_x} & -I_{n_u} \\ \mathbf{0}_{n_u \times n_x} & I_{n_u} \end{bmatrix}$, and $g_i = \begin{bmatrix} \mathbf{0}_{n_u \times 1} \\ u_{\max} \end{bmatrix}$.

III. MPC controller architecture

Since the cost function (1a) in problem (1) is not convex with respect to the N, each of the convex constrained optimisation problems corresponding to horizons $N = \{1, \ldots, N_{\max}\}$ is solved in sequence according to Algorithm 1. A Xilinx MicroBlaze soft-core processor (with floating point unit) is used to calculate the LTV prediction matrices, and provide data to a custom peripheral core designed to solve the set of linear equations that occur at each iteration of the PD-IP algorithm, which solves the optimisation problem occuring at step **3** of Algorithm 1. Using the annotation **(M)** to denote a process performed on the MicroBlaze and **(F)** to indicate that the process happens in the custom logic implemented in the FPGA fabric, Algorithm 2 shows the software-hardware

Algorithm 1: Variable horizon MPC controller

Initialisation:

1. $J_{opt} = \infty$, $N_{opt} = 0$, $\theta_{opt} = 0$;

for $j = 1$ **to** $N_{\max}$

2. Calculate A_j, B_j using [21];

3. Solve (1) s.t. $N = j$. If feasible, $J_j = J^*$, else $J_j = \infty$;

if $J_j < J_{opt}$

4. $J_{opt} = J_j$, $N_{opt} = j$, $\theta_{opt} = \theta^*$;

end if

end for.

5. Return u_0 from θ_{opt}.

partition. The notation $n_c (= 2Nn_u)$ is used to denote the total number of inequality constraints.

Algorithm 2: PD-IP Algorithm

1. Initialisation: θ_0, λ_0, s_0, z_0, σ, I_{IP} **(M)**

for $k = 0$ **to** $I_{\text{IP}} - 1$

Linearisation:

2. $\mu_k = z_k^T s_k / Nn_c$ **(M)**

3. $w_k = z_k ./ s_k$ (elementwise) **(M)**

4. $m_k = \begin{bmatrix} \theta_k \\ \lambda_k \\ \sigma\mu_k \mathbf{1}_{n_c} ./ s_k - \text{diag}(w_k) g_k + z_k \end{bmatrix}$ **(M)**

5. $\Phi_k = G^T \text{diag}(w_k) G$ **(F)**

6. $\mathscr{A}_k = \begin{bmatrix} H + \Phi_k & F^T \\ F & 0 \end{bmatrix}$ **(F)**

7. $b_k = \begin{bmatrix} -h \\ f \end{bmatrix} - \begin{bmatrix} H + \Phi_k & F^T & G^T \\ F & 0 & 0 \end{bmatrix} m_k$ **(F)**

Solve:

8. $\mathscr{A}_k c_k = b_k$, where $c_k = \begin{bmatrix} \Delta\theta_k \\ \Delta\lambda_k \end{bmatrix}$ **(F)**

9. Calculate $G(\theta_k + \Delta\theta_k) - g$ **(M)**

10. $\Delta z_k = \text{diag}(w_k)(G(\theta_k + \Delta\theta_k) - g) + \sigma\mu_k s_k^{-1}$ **(M)**

11. $\Delta s_k = -G(\theta_k + \Delta\theta_k) - s_k$ **(M)**

Line search:

12. $\alpha_k = \max_{[0,1]} \alpha : \begin{bmatrix} \lambda_k + \alpha\Delta\lambda_k \\ s_k + \alpha\Delta s_k \end{bmatrix} > 0$ **(M)**

Update:

13. $(\theta, \lambda, z, s)_{k+1} = (\theta, \lambda, z, s)_k + \alpha\Delta(\theta, \lambda, z, s)_k$ **(M)**

end for

The number of iterations is fixed, since in a real time setting, the time to obtain a solution (even if not optimal) should be bounded. Some of the optimisation problems might be infeasible for shorter horizons N. Three heuristics are used to attempt to identify infeasibility and/or a poor solution after I_{IP} iterations: the solution is assumed infeasible if for any j, $(\|b_j\|_\infty + Nn_c\mu_j) > 10^4 \min_i(\|b_i\| + Nn_c\mu_i)$, [23] and the solution is also rejected as likely infeasible if $\mu_{I_{IP}-1} > 10^{-3}$ or $\|b_{I_{IP}-1}\|_\infty > 0.01\|b_0\|_\infty$ (i.e. the result has not converged sufficiently).

As in [15], [16], the system of linear equations is solved using the minimum residual (MINRES) algorithm. In the present context, this iterative method can be considered preferable to factorisation-based methods, being dominated by matrix-vector multiplications, which are very easily parallelised, and the number of iterations (i.e. the solution time) can be traded for solution accuracy.

At a high level, the custom peripheral core (PCORE) is split into two major parts. The first component performs steps (5-7) of the Algorithm 2, building the linear system, whilst the second, more complex component performs step 8.

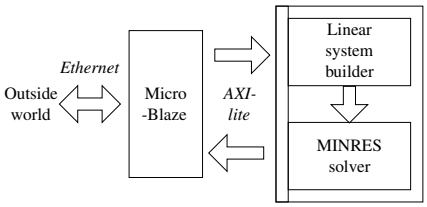

Fig. 1. High-level architecture of controller implementation

The PCORE is connected to the MicroBlaze using the AXI4-lite bus, which presents data and status registers as shared memory locations (Figure 1).

A. Linear system builder

The linear system builder is designed using Xilinx System Generator for DSP, and consists of four main modes of operation. In the first mode, the vectorised matrices A_i, B_i, G_i, F_i, H_i (in row major form) and vectors m_k, w_k and $[-h^T, f^T]^T$ are communicated from the MicroBlaze to the peripheral via a shared FIFO RAM. Five additional shared RAMs of length $(N_{\max} + 1)$ are used, with element i of each of these RAMs containing the address of the first element of A_i, B_i, etc. (Figure 2). Consequently A_i etc. do not need to be stored

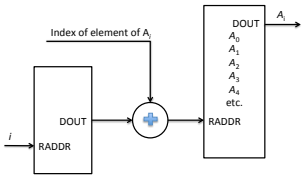

Fig. 2. Indexing of time varying prediction matrices

sequentially and can be repeated (e.g. for LTI cases) without the overhead of re-writing the time invariant matrices to the PCORE.

The second mode triggers a state machine which calculates $G^T \text{diag}(w_k) G + H$ and stores this row-wise in a further RAM of size $N_{\max}(n_x + n_u)^2 + n_x^2$. This calculation requires $N_{\max}(n_x + n_u)^2$ dot products per PD-IP iteration. To achieve a throughput of one scalar addition per clock cycle within each dot product (after an initial latency), elementwise multiplication is performed in single precision floating point using a pipelined IP-core, then converted to 64-bit signed fixed point with a 32-bit fractional part [11]. The accumulated result is transformed back to single precision floating point.

The third mode triggers a second state machine which addresses the RAMs to construct a sequence of numbers comprising the rows of a compact block representation of

$\mathscr{A}_k$:

$$\begin{bmatrix} H_{\Phi xx,0} & H_{\Phi xu,0} & -I & A_0^T \\ H_{\Phi ux,0} & H_{\Phi uu,0} & 0 & B_0^T \\ H_{\Phi xx,1} & H_{\Phi xu,0} & -I & A_1^T \\ H_{\Phi ux,1} & H_{\Phi uu,0} & 0 & B_1^T \\ \vdots & \vdots & \vdots & \vdots \\ H_{\Phi xx,N} & 0 & -I & [F_N^T, 0] \\ -I & 0 & 0 & 0 \\ A_0 & B_0 & -I & 0 \\ \vdots & \vdots & \vdots & \vdots \\ F_N & 0 & 0 & 0 \end{bmatrix}. \quad (3)$$

Whilst less generic than interleaving the primal, dual and slack variables corresponding to each stage of the prediction horizon forming a banded matrix [19], to perform dot products of the rows of (3) with appropriately sized vectors with a throughput of one dot product per clock cycle requires a bank of $(3n_x + n_u)$ multipliers in parallel, in constrast to $(2n_u + 4n_x - 1)$ if a band structure were exploited.

The fourth mode triggers the second state machine again with an additional configuration flag enabled, and in combination with a second multiply-accumulate device and a subtractor to calculate b_k as in step 7, and presents this sequence as a separate output.

B. MINRES solver

The majority of the computation in the MINRES algorithm comprises the Lanczos process. This is implemented using fixed-point arithmetic, enabled by the preconditioner proposed by [24], [25], which ensures that key variables are in the range $[-1, 1]$. Using the notation $Z_{k,mn}$ to denote the nth element of the mth row of matrix Z at iteration k, by letting

$$\mathscr{M}_{k,mn} = \begin{cases} \left(\sum_p |\mathscr{A}_{k,mp}|\right)^{-1/2} & \text{if } m = n \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

instead of directly solving $\mathscr{A}_k c_k = b_k$, MINRES is used to solve $(\mathscr{M}_k \mathscr{A}_k \mathscr{M}_k)\tilde{c}_k = (\mathscr{M}_k b_k)/\|\mathscr{M}_k b_k\|_2$, and $c_k = \mathscr{M}_k \|\mathscr{M}_k b_k\|_2 \tilde{c}_k$. The preconditioner is calculated from the row-wise sequence of the elements of (3) and stored in a dual-port RAM. The sequence (3) is also simultaneously stored temporarily in a single-port RAM. After the preconditioner is calculated, the contents of the RAM are read in sequence and multiplied by the corresponding two elements of the preconditioner (determined by a state machine addressing the dual port RAM containing the preconditioner), converted to a fixed point representation and written to a bank of block RAMs, each of which will contain a single column of the preconditioned version of (3).

The key operation within the Lanczos process is a matrix-vector multiplication, $(\mathscr{M}_k \mathscr{A}_k \mathscr{M}_k)v$ performed at each iteration. This is implemented as a bank of $(3n_x + n_u)$ multipliers, a bank of $(3n_x + n_u)$ single port RAMs comprising the columns of the preconditioned fixed point copy of (3) and a bank of $2n_x$ dual port RAMs and n_u single port RAMs, containing appropriately ordered elements of v. Simulink HDL coder transparently handles element-wise vector operations and automatically creates the tree-reduction structure for adding the products together, and perform register balancing between tree stages. However, to efficiently use the 25×18-bit hardware multiplier units (DSP48E) on the FPGA and to avoid timing issues at the implementation stage, the operands are split and long multiplication is performed in stages, with pipeline registers (delay blocks) inserted in between. Using the notation (u/s)Fix*xx_yy* to denote an (un)signed *xx*-bit number with *yy*-bit fractional part, a data type of sFix25_23 is used for the matrix and sFix35_33 for the elements of v. This uses 2 DSP48Es per vector element.

The Lanczos process also requires a reciprocal square root operation to be performed. This is implemented as an integer square root operation followed by calculation of the reciprocal of the result. Due to the word lengths involved (sFix52_50), the Simulink built-in blocks generate a design with many layers of logic between registers even if automatic register balancing is used in the synthesis tools. These operations need only be performed on scalars and therefore do not need to be pipelined, so to circumvent this limitation, a standard integer division algorithm and an integer square root algorithm are implemented as state machines using M-code. The result of this operation is stored as type sFix64_32.

Outside of the Lanczos process, the remainder of the algorithm is implemented using floating point arithmetic since the bounds on the magnitudes are not known *a priori*. This is designed using Xilinx System Generator for DSP to instantiate pipelined IP-cores performing floating point operations and type conversions, with the HDL description of the Lanczos process imported as a "black box". A single precision (32-bit) floating point data type is used in preference to the more common double precision (64-bit) representation, this decision being motivated by FPGA resource usage.

IV. Resource and power usage

Table I shows the resource usage of the MicroBlaze and custom circuit on the FPGA on the ML605 Evaluation Board. The resources used specifically in the Lanczos process are also shown in italic. An estimate of utilisation for a selection of smaller FPGAs is also presented.

TABLE I
FPGA RESOURCE USAGE

Component	FF	LUT	DSP48E	BRAM
Microblaze	8608	9864	6	41
Custom PCORE	28873	26359	199	67
(Lanczos)	*(9806)*	*(8198)*	*(89)*	*(34)*
Total	37481	36223	205	108
As % of available resources				
Virtex 6 VLX240T	**12.4%**	**24.0%**	**26.7%**	**26.0%**
Virtex 6 VLX75T	*40.3%*	*77.8%*	*71.2%*	*69.2%*
Virtex 5 VSX95T	*63.7%*	*61.5%*	*32.0%*	*44.3%*
Artix 7 A100T	*29.6%*	*57.1%*	*85.4%*	*80.0%*
Kintex 7 K70T	*45.7%*	*88.3%*	*85.4%*	*80.0%*

(FF: Flip-flop, LUT: Look-up table, BRAM: Block RAM, DSP48E: Hardware multiplier)

Using Xilinx Power Analyzer's default operating conditions, power consumption of the system is estimated. This shows

that the majority of power consumption is due to "leakage" (i.e. due to quiescent currents) and the custom PCORE. The power consumption of the PCORE and leakage are estimated for a selection of smaller FPGAs, using the Timing and Power Analysis workflow in System Generator for DSP and presented in Table II. (Totals in brackets exclude the MicroBlaze). The estimated consumption of the PCORE is consistent over the FPGA models considered, so to gain a further reductions in power consumption, improving the sharing of resources between tasks would be worthwhile.

TABLE II
ESTIMATED POWER CONSUMPTION

FPGA	Total	PCORE	Leakage
Virtex 6 LX240T	5.247 W	0.6 W	3.481 W
Virtex 6 LX75TL	(2.176 W)	0.6 W	1.081 W
Artix XC7A100TL	(0.988 W)	0.6 W	0.041 W
Kintex XC7K70TL	(0.965 W)	0.6 W	0.042 W

V. CLOSED-LOOP SIMULATION

The design is used to control a simulation of the plant dynamics on a PC connected to the FPGA using UDP/IP over 100 Mbit ethernet. The target orbital parameters, the chaser relative position and velocity and parameters configuring the number of algorithm iterations are transmitted to the FPGA, which returns the optimal cost, optimal prediction horizon and the full predicted state and input trajectory.

A. Simulator and scenario

Continuous-time target orbit dynamics and chaser relative dynamics are simulated in Simulink. The target is modelled as being in a Keplerian orbit [22] around Mars, with semimajor axis $a = 4643$ km and eccentricity $e = 0.2044$. The orbital radius at perigee is $r_p = 3.6939 \times 10^6$ m. The the open-loop chaser relative dynamics are modelled using the continuous time-varying equations from which the Y-A equations [21] derive. Navigation uncertainty is modelled as zero-mean additive Gaussian white noise on each channel. With r denoting the Euclidean distance from the target, the standard deviation is $0.35/1000r + 0.0028$ (m) for position, and $0.0115/1000r + 0.003$ (m/s) for velocity measurements [4]. The impulsive thrust input request from the predictive controller delivered as a 1 s pulse of acceleration, 1 s after the control instruction is received from the FPGA (this delay is unmodelled in the prediction).

In elliptical scenarios, points on the x-axis in the relative reference frame, with zero instantaneous velocity, are not unforced equilibria. The terminal state $x_T(k+N)$ is chosen as a point on a periodic holding trajectory. Letting h_p be the average distance the target, $\nu_N = \nu(k+N)$, and $\rho_N = (1 + e\cos\nu_N)$, then $x_T(k+N) = h_p\left[\rho_N, 0, -e\sin\nu_N, -k_2 e\rho_N^2 \sin\nu_N, 0, k_2(\rho_N^2 - \rho_N^3)\right]^T$, where $k_2 = \mu^2 / \left(r_p\sqrt{2\mu\left(r_p^{-1} - 0.5a^{-1}\right)}\right)^3$, and μ is the gravitational parameter.

The scenario demonstrated starts at a position of $x(0) = [10000\,\text{m}, 0\,\text{m}, 0\,\text{m}, 0\,\text{m/s}, 0\,\text{m/s}, 0\,\text{m/s}]^T$) and the hold points are scheduled to be centred at 5000 m, 2000 m, 1000 m, 500 m, 200 m and 100 m in front of the target. The chaser is released from a holding point when $N_{opt} \leq 2$ for at least 5 sampling instants. Based on [4], the sampling period is chosen to be $T_s = 300$ s, and the weighting parameter $\gamma = 35$. The impulsive control input bound $u_{\max} = 1$ m/s.

B. Control results

Table III shows metrics for the quality of the control performance, with respect to variation in the number of interior point iterations I_{IP}, and the number of MINRES iterations, $I_{MR} = \lfloor \eta\,(N(2n_x + n_u) + n_x + n_T) \rfloor$. These include the completion time of the whole scenario, the solution time for the full VH-LTV-MPC problem at each sampling instant, the cumulative impulsive velocity change ΔV applied, and the maximum and mean 2-norm of the difference between the control action from the FPGA-based strategy and an "ideal" solution solved on the PC using GLPK [26]. (For context, the MicroBlaze runs at 100 MHz, and the peripheral MINRES accelerator at 200 MHz). Figure 3 shows an example closed loop trajectory with $I_{IP} = 22$ and $\eta = 1.1$.

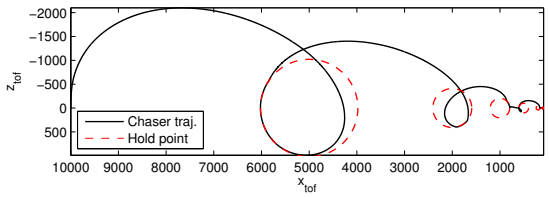

Fig. 3. Closed loop trajectory with FPGA in the loop

C. Computational load (for one fixed horizon)

The apportionment of computation time for a fixed horizon $N = 20$ is presented in Table IV, obtained empirically using a 100 MHz Timer Counter peripheral connected to the MicroBlaze.

TABLE IV
COMPUTATION TIME APPORTIONMENT FOR $N = 20$, $I_{IP} = 22$, $\eta = 1.1$

Operation	Runs	Ticks	Total	(ms)
Prediction model	1	16829	16829	0.17
Prediction model transfer	1	6867	6867	0.07
Memory initialisation	1	3594	3594	0.04
Vector transfer	I_{IP}	31403	690866	6.9
Microblaze computation	I_{IP}	86676	1906872	19.06
PCORE computation	I_{IP}	224464	4938208	49.38
Total			7563236	75.6

In comparison, for a *fixed horizon* $N = 20$, an LP solver generated using CVXGEN [27], configured with prediction matrices starting from $\nu = \pi/2$ with x_0 on the x axis at 10000 m, and scheduled holding point at 5000 m takes 314 ms to run 23 iterations directly on the MicroBlaze using single precision arithmetic. The FPGA-based solver offers a 4× speedup (or 2.5× cycle-per-cycle due to the different clock rates). The FPGA-based solver can also handle dense quadratic stage and input costs with no modifications. A QP solver for such costs from CVXGEN takes 1.79 s with single precision arithmetic (14× more cycle-per-cycle).

TABLE III

CLOSED LOOP PERFORMANCE RESULTS

(a) Completion time (s)

	I_{MR} scaling, η			
I_{IP}	0.9	1.0	1.1	1.2
20	36000	35700	36300	35700
22	37200	35700	35700	35700
24	36600	35700	35700	35700
26	37200	35700	35700	36000
28	37200	36000	35700	35700

(b) Solution time (ms)

	I_{MR} scaling, η			
I_{IP}	0.9	1.0	1.1	1.2
20	626	655	684	713
22	687	720	752	784
24	749	785	820	854
26	811	850	887	925
28	873	914	955	996

(c) ΔV usage (m/s)

	I_{MR} scaling, η			
I_{IP}	0.9	1.0	1.1	1.2
20	4.85	4.79	4.85	4.78
22	4.84	4.84	4.76	4.76
24	4.81	4.84	4.76	4.83
26	4.88	4.76	4.76	4.79
28	4.82	4.84	4.76	4.76

(d) Max norm input error (m/s)

	I_{MR} scaling, η			
I_{IP}	0.9	1.0	1.1	1.2
20	3.56×10^{-2}	3.56×10^{-2}	1.45×10^{-2}	2.90×10^{-3}
22	3.76×10^{-3}	3.47×10^{-3}	3.33×10^{-3}	6.19×10^{-4}
24	3.29×10^{-2}	2.41×10^{-3}	4.54×10^{-4}	1.93×10^{-1}
26	8.41×10^{-3}	2.28×10^{-3}	2.68×10^{-3}	1.75×10^{-3}
28	1.45×10^{-2}	3.26×10^{-3}	2.93×10^{-4}	2.11×10^{-3}

(e) Mean norm input error (m/s)

	I_{MR} scaling, η			
I_{IP}	0.9	1.0	1.1	1.2
20	8.20×10^{-4}	3.84×10^{-4}	1.72×10^{-4}	2.54×10^{-5}
22	2.88×10^{-4}	8.20×10^{-5}	4.06×10^{-5}	6.25×10^{-6}
24	7.30×10^{-4}	1.11×10^{-4}	1.36×10^{-5}	1.62×10^{-3}
26	3.05×10^{-4}	7.49×10^{-5}	3.08×10^{-5}	1.56×10^{-5}
28	5.59×10^{-4}	1.43×10^{-4}	4.38×10^{-6}	1.87×10^{-5}

VI. CONCLUSIONS AND FUTURE DEVELOPMENTS

This paper has presented the design and evaluation of a predictive controller for a phase of spacecraft rendezvous on an FPGA using a mixed hardware-software approach, with the custom hardware component accelerating the critical path of the algorithm. The controller supports a LTV prediction model and a variable horizon, and the custom hardware components are implemented using Simulink HDL Coder and Xilinx System Generator for DSP. Further work will investigate the application to additional rendezvous phases, and improving the efficiency of the custom hardware component.

REFERENCES

[1] M. Rossi and M. Lovera, "A predictive approach to formation keeping for constellations of small spacecraft in elliptic orbits," in *Proc. 5th ESA Int. Conf. Spacecraft Guidance, Navigation and Control Systems*, Frascati, Rome, Oct. 22–25 2003, pp. 209–216.

[2] R. Larsson, S. Berge, P. Bodin, and U. Jönsson, "Fuel efficient relative orbit control strategies for formation flying and rendezvous within PRISMA," in *Proc. 29th Annual AAS Guidance and Control Conf.*, 2006.

[3] L. Breger and J. P. How, "Gauss's variational equation-based dynamics and control for formation flying spacecraft," *J. Guidance, Control, and Dynamics*, vol. 30, no. 2, pp. 437–448, 2007.

[4] E. N. Hartley, P. A. Trodden, A. G. Richards, and J. M. Maciejowski, "Model predictive control system design and implementation for spacecraft rendezvous," *Control Eng. Pract.*, vol. 20, no. 7, pp. 695–713, Jul 2012.

[5] A. Richards and J. P. How, "Robust variable horizon model predictive control for vehicle maneuvering," *Int. J. Robust and Nonlinear Control*, vol. 16, no. 7, pp. 333–351, February 2006.

[6] A. Richards and J. How, "Performance evaluation of rendezvous using model predictive control," in *AIAA Guidance, Navigation and Control Conf. and Exhibit*, Austin, TX, Aug 11–14 2003.

[7] J. A. Starek and I. V. Kolmanovsky, "Nonlinear model predictive control strategy for low thrust spacecraft missions," *Optim. Control Appl. and Meth.*, vol. (In press), Sep 2012.

[8] D. Beaty, M. Grady, L. May, and B. Gardini, "Preliminary planning for an international Mars Sample Return mission," Report of the International Mars Architecture for the Return of Samples (iMARS) Working Group, Jun 2008.

[9] P. D. Vouzis, L. G. Bleris, M. G. Arnold, and M. V. Kothare, "A system-on-a-chip implementation for embedded real-time model predictive control," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 3, pp. 1–12, 2009.

[10] N. Yang, D. Li, J. Zhang, and Y. Xi, "Model predictive controller design and implementation on FPGA with application to motor servo system," *Control Eng. Pract.*, vol. 20, no. 11, pp. 1129–1235, Nov 2012.

[11] H. Chen, F. Xu, and Y. Xi, "Field programmable gate array/system on a programmable chip-based implementation of model predictive controller," *IET Control Theory Appl.*, vol. 6, no. 8, pp. 1055–1063, Jul 2012.

[12] A. Wills, A. Mills, and B. Ninness, "FPGA implementation of an interior-point solution for linear model predictive control," in *Proc. 18th IFAC World Congress*, Milan, Italy, 2011.

[13] K. V. Ling, B. F. Wu, and J. M. Maciejowski, "Embedded model predictive control (MPC) using a FPGA," in *Proc. 17th IFAC World Congress*, Seoul, Korea, July 2008.

[14] A. G. Wills, G. Knagge, and B. Ninness, "Fast linear model predictive control via custom integrated circuit architecture," *IEEE Trans. Control Syst. Technol.*, vol. 20, no. 1, pp. 59–71, Jan 2012.

[15] J. L. Jerez, K.-V. Ling, G. A. Constantinides, and E. C. Kerrigan, "Model predictive control for deeply pipelined field-programmable gate array implementation: algorithms and circuitry," *IET Control Theory Appl.*, vol. 6, no. 8, pp. 1029–1041, 2012.

[16] E. N. Hartley, J., A. Suardi, J. M. Maciejowski, E. C. Kerrigan, and G. A. Constantinides, "Predictive control of a Boeing 747 aircraft using an FPGA," in *Proc. IFAC Conf. Nonlinear Model Predictive Control*, Aug 23–27 2012.

[17] X. Chen and X. Wu, "Design and implementation of model predictive control algorithms for small satellite three-axis stabilization," in *Proc. Int. Conf. on Information and Automation*, Shenzhen, China, Jun 2011, pp. 666–671.

[18] *ML605 Hardware User Guide*, Xilinx, February 15 2011.

[19] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *J. of Optim. Theory and Appl.*, vol. 99, no. 3, pp. 723–757, December 1998.

[20] S. J. Wright, "Interior-point method for optimal control of discrete-time systems," *J. Optim. Theory Appl.*, vol. 77, pp. 161–187, 1993.

[21] K. Yamanaka and F. Ankersen, "New state transition matrix for relative motion on an arbitrary elliptical orbit," *J. Guidance Control and Dynamics*, vol. 25, no. 1, pp. 60–66, 2002.

[22] W. Fehse, *Introduction to Automated Rendezvous and Docking of Spacecraft*. Cambridge University Press, 2003.

[23] E. M. Gertz and S. Wright, *OOQP User Guide*, Mathematics and Computer Science Division, Argonne National Laboratory, Oct 2001.

[24] E. C. Kerrigan, J. L. Jerez, S. Longo, and G. A. Constantinides, "Number representation in predictive control," in *Proc. IFAC Conf. Nonlinear Model Predictive Control*, Noordwijkerhout, NL, Aug 23–27 2012, pp. 60–67.

[25] J. L. Jerez, G. A. Constantinides, and E. C. Kerrigan, "Fixed-point Lanczos with guaranteed variable bounds," 2012.

[26] A. Makhorin, "GNU linear programming kit," 2000.

[27] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.