2014 UKACC International Conference on Control
9th - 11th July 2014, Loughborough, U.K

A Synthesis Strategy for Nonlinear Model Predictive Controller on FPGA

Bartosz Käpernick*, Sebastian Süß*, Endric Schubert† and Knut Graichen*
*Institute of Measurement, Control, and Microtechnology, University of Ulm, Ulm, Germany
Email: {bartosz.kaepernick,sebastian-1.suess,knut.graichen}@uni-ulm.de
†Missing Link Electronics, Neu-Ulm, Germany
Email: endric@endric.com

***Abstract*—This paper describes an implementation strategy of nonlinear model predictive controller for FPGA systems. A high-level synthesis of a real-time MPC algorithm by means of the MATLAB HDL Coder as well as the Vivado HLS tool is discussed. In order to exploit the parallel processing of FPGAs, the included integration schemes are parallelized using a fixed-point iteration approach. The synthesis results are demonstrated for two different example systems.**

I. INTRODUCTION

The solution of an optimal control problem (OCP) along a moving horizon is the basis of model predictive control (MPC) [1]–[3]. This modern control method has become popular in the recent years due to its ability to handle nonlinear multiple input systems with constraints. However, the computational effort that is typically required to solve the underlying OCP usually limits the use of nonlinear MPC to sufficiently slow and/or low-dimensional systems.

A suitable way to circumvent this problem is the use of real-time MPC approaches and algorithms existing in the literature [4]–[7]. These methods implement different solution strategies, as for instance a continuation method in combination with the generalized minimum residual (GMRES) method [4] or a real-time iteration scheme using a Newton-type framework [5]. While [4]–[7] address nonlinear systems, linear model predictive control approaches with real-time applicability are presented in [8], [9].

A more intuitive way to speed up the computation in MPC frameworks is to use suitable hardware systems that exploit certain algorithmic properties of an applied approach. In terms of computation times in combination with parallel processing, FPGAs (field-programmable gate array) are a serious alternative to common sequentially working CPUs (central processing unit). Due to the inherent parallel processing of programmable logic in FPGAs, a high data throughput with low latency can be achieved [10]. However, the conversion of a general MPC algorithm into digital circuits of an FPGA is a challenging task and hence requires a thorough analysis of the used model predictive control scheme – in particular in the nonlinear MPC case.

There exist already some contributions regarding the implementation of linear model predictive controller on FPGA systems in the literature. An MPC framework based on the solution of a quadratic program (QP) and a combined implementation on an FPGA as well as a CPU was presented in [11]. A similar approach was demonstrated in [12] where only an FPGA implementation was used. The results in both contributions rely on a floating point notation for the number representation within the MPC scheme. In contrast, [13] and [14] demonstrated an FPGA realization of a QP based MPC by means of a fixed-point number representation. A different MPC framework that followed the ideas of a fast gradient method was discussed in [15]. The approach was implemented on common FPGAs from Xilinx where computation times in the microsecond range were achieved. However, all approaches so far used a linear MPC for controlling linear systems. Regarding the nonlinear case, a first discussion on FPGA implementation strategies of a nonlinear MPC scheme can be found in [16].

This paper describes a high-level synthesis strategy for implementing a nonlinear model predictive controller on FPGA systems. The discussion is based on a real-time MPC algorithm that is converted into the register transfer level by means of the MATLAB HDL Coder and the Vivado HLS tool, respectively. The applied algorithm contains the numerical solution of the necessary optimality conditions. For the underlying numerical integrations, an approach to parallelize the computation is presented that follows the ideas of a fixed-point iteration. Synthesis results for two different examples demonstrate the potential of the presented methods.

II. MPC SCHEME AND ALGORITHM

This section introduces the problem formulation and briefly describes the real-time MPC algorithm that is used for a synthesis procedure on an FPGA.

A. Problem formulation

The MPC scheme considered in this contribution repeatedly solves an input constrained OCP of the following form:

$$\min_{u(\cdot)} \quad J(u, x_k) = V(x(t_k + t)) + \int_{t_k}^{t_k+T} L(x(t), u(t))\, \mathrm{d}t \quad (1a)$$

$$\text{s.t.} \quad \dot{x}(t) = f(x(t), u(t)), \;\; x(t_k) = x_k \quad (1b)$$

$$u(t) \in [u^-, u^+], \;\; t \in [t_k, t_k + T], \quad (1c)$$

where $T > 0$ denotes the prediction horizon and $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ are the states and controls of the system, respectively. The functional (1a) contains the terminal cost $V : \mathbb{R}^n \to \mathbb{R}^0_+$ and integral cost $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^0_+$ both being continuous differentiable and positive definite functions. The system function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ in (1b) is also assumed to be continuously differentiable in its arguments. The initial condition $x(t_k) = x_k$ denotes the measured (or observed) state of the system at sampling instance $t_k = t_0 + k\Delta t$ with sampling time Δt. Condition (1c) represents the input constraints.

For further considerations it is assumed that OCP (1) possesses an optimal solution that is denoted by

$$u_k^*(t) := u^*(t; x_k),\ x_k^*(t) := x^*(t; x_k),\ t \in [t_k, t_k + T], \quad (2)$$

where the subindex k indicates the current sampling instance t_k. Typical MPC approaches aim at computing the optimal solution (2) in each sampling instance and inject the first part of the optimal control trajectory to the system, i.e.

$$u(t) = u_k^*(t),\ \ t \in [t_k, t_k + \Delta t). \quad (3)$$

In the next sampling instance $t_{k+1} = t_k + \Delta t$, OCP (1) is solved again with the new state $x(t_{k+1})$.

B. Real-time algorithm

The algorithm that is used in this paper for solving (1) relies on the optimality conditions based on the definition of the Hamiltonian[1]

$$H(x, u, \lambda) = L(x, u) + \lambda^{\mathsf{T}} f(x, u) \quad (4)$$

with the costates $\lambda \in \mathbb{R}^n$. Given an optimal solution (x_k^*, u_k^*), Pontryagin's Maximum Principle [17] states that there exists a costate trajectory λ_k^* satisfying the following conditions:

$$\begin{aligned}
&\dot{x}_k^* = f(x_k^*, u_k^*), && x_k^*(t_k) = x_k && \text{(5a)}\\
&\dot{\lambda}_k^* = -H_x(x_k^*, u_k^*, \lambda_k^*), && \lambda_k^*(t_k + T) = V_x(x_k^*(t_k + T)) && \text{(5b)}\\
&u_k^* = \underset{u \in [u^-, u^+]}{\arg\min}\ H(x_k^*, u, \lambda_k^*), && t \in [t_k, t_k + T], && \text{(5c)}
\end{aligned}$$

where $H_x := \partial H / \partial x$ and $V_x := \partial V / \partial x$ denote the partial derivatives of the Hamiltonian H and the terminal cost V w.r.t. the states x, respectively. The separated boundary conditions in (5a), (5b) are due to the OCP formulation (1) without terminal conditions.

The optimality conditions (5) can be solved by means of the (projected) gradient method [18], [19]. In view of (5), the following gradient steps are performed for solving OCP (1):

- Initialization of input trajectory $u_k^{(0)}$
- Gradient iterations for $j = 0, \ldots, N_{\text{grad}} - 1$:
  1) Forward integration of (5a) to obtain $x_k^{(j)}$
  2) Backward integration of (5b) to obtain $\lambda_k^{(j)}$
  3) Solution of the line search problem

$$\alpha^{(j)} = \underset{\alpha > 0}{\arg\min}\ J\left(\psi\left(u_k^{(j)} - \alpha g_k^{(j)}\right), x_k\right) \quad (6)$$

with search direction $g_k^{(j)} = H_u(x_k^{(j)}, u_k^{(j)}, \lambda_k^{(j)})$ and projection function

$$\psi(u) = \begin{cases} u^- : u < u^- \\ u \ \ \ : u \in [u^-, u^+] \\ u^+ : u > u^+ \end{cases} \quad (7)$$

  4) Control update $u_k^{(j+1)} = \psi\left(u_k^{(j)} - \alpha^{(j)} g_k^{(j)}\right)$

The solution of the line search problem (6) is determined by means of the adaptive approach discussed in [19]. Following these ideas, the cost (1a) is evaluated at three sample step lengths $\alpha_1 < \alpha_2 < \alpha_3$ constructing a quadratic approximation that is subsequently minimized to compute $\alpha^{(j)}$. In addition, the corresponding minimum is tracked along each gradient iteration as well as MPC step by adapting the sample interval accordingly.

In order to achieve a real-time feasibility of the approach, a limited number of gradient iterations N_{grad} is performed per MPC step, i.e. the input to the system is

$$u(t) = u_k^{(N_{\text{grad}})}(t), \quad t \in [t_k, t_k + \Delta t) \quad (8)$$

where the last iteration is additionally used in the next sampling instance to re-initialize the controls. Convergence and stability analysis regarding the projected gradient method as well as the (prematurely stopped) MPC scheme can be found in [18] and [20], respectively.

III. FPGA SYNTHESIS STRATEGY

Synthesis strategies map an algorithmic description into digital hardware. Typically, the hardware realization is then characterized by means of a hardware description language (HDL) including the corresponding algorithmic behavior. This section discusses the synthesis procedure of the real-time MPC algorithm from Section II-B.

A. High-level synthesis

A well known synthesis strategy for implementing algorithms into digital circuits is the high-level synthesis [21] also known as behavioral synthesis. The algorithmic description (algorithmic level) is generally provided in form of a high-level programming language as for instance C/C++ or SystemC. After analysing the code a hardware design is performed including various optimization strategies resulting in a hardware description (register transfer level) also allowing to verify the hardware. As shown in Figure 1, the MATLAB HDL Coder as well as the software tool Vivado HLS are used in this paper for the high-level synthesis of the real-time MPC algorithm described in Section II-B.

1) Synthesis procedure with MATLAB *HDL Coder:* The MATLAB HDL Coder requires the MPC algorithm to be provided as MATLAB code. Based on this MATLAB implementation, the coder then optionally converts all data into a fixed-point number representation. Subsequently, a related HDL code of the algorithm is generated and simulated. This procedure may contain a few iterations until a suitable number of bits for the number representation is chosen. Finally, the

[1] In the following lines, the time argument is omitted where it is convenient to maintain readability.

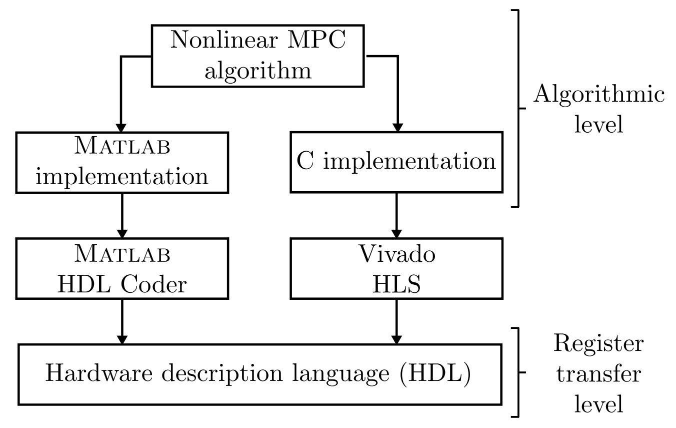


Fig. 1. Procedure of the high-level synthesis.

generated HDL code can be implemented and tested on an FPGA system.

2) Synthesis procedure with Vivado HLS: In order to use the high-level synthesis tool Vivado HLS from Xilinx, the MPC algorithm has to be implemented in C/C++ or SystemC. The Vivado HLS tool basically performs the same steps within the synthesis procedure as the MATLAB HDL Coder. However, optimization objectives can be formulated by so-called constraints and directives. Constraints incorporate information with regard to the target device such as the associated size of the chip or a desired clock rate. However, constraints do not represent direct limitations of the FPGA and hence may be violated throughout the synthesis procedure. To this end, the hardware limitations can be taken into account by formulating corresponding directives. Vivado HLS also provides some optimization strategies for variables, functions, and loops included in the code.

B. Parallelization of the integration scheme

The MPC algorithm from Section II-B relies on the forward and backward integration of the system and adjoint dynamics, respectively. Generally, the steps within numerical integration schemes are performed in a sequential manner and thus cannot take advantage of the parallel processing of an FPGA system. To this end, a parallel integration approach is discussed where ideas of the fixed-point iteration method are used.

An integration step of both explicit and implicit integration schemes (e.g. Euler forward/backward approach, Runge-Kutta methods) with the step size h is given by

$$x^{i+1} = x^i + h\phi\big(x^i, x^{i+1}, u^i, u^{i+1}\big), \quad i = 0, \dots, N-1 \quad (9)$$

where N is the number of discretization points and x^i and u^i denote the (approximated) states $x(t_k + ih)$ and controls $u(t_k + ih)$ at these discretization points, respectively. In order to derive a more suited form for the FPGA implementation, the integration steps (9) are summed up in the following way

$$\underbrace{\begin{bmatrix} x^1 \\ \vdots \\ x^N \end{bmatrix}}_{=:z} = a + Kz + h \underbrace{\begin{bmatrix} \phi\left(x^0, x^1, u^0, u^1\right) \\ \vdots \\ \phi\left(x^{N-1}, x^N, u^{N-1}, u^N\right) \end{bmatrix}}_{=:\Phi(z)} \quad (10)$$

with the vector $a \in \mathbb{R}^{nN}$ and the matrix $K \in \mathbb{R}^{nN \times nN}$ given by

$$a = \begin{bmatrix} x^0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad K = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ I & 0 & \cdots & 0 & 0 \\ 0 & I & \cdots & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & \cdots & I & 0 \end{bmatrix} \quad (11)$$

where $I \in \mathbb{R}^{n \times n}$ is the unit matrix and $z \in \mathbb{R}^{nN}$ comprises the states at all discretization points $i = 1, \dots, N$. The integration is then approximated by means of the fixed-point iteration

$$z^{(j+1)} = a + Kz^{(j)} + h\Phi(z^{(j)}) =: F\big(z^{(j)}\big), \quad j = 1, \dots, M \quad (12)$$

allowing to obtain approximations of the discretized states in a parallel fashion. The maximum number of fixed-point iterations M has to be chosen appropriately for adequate integration results. Note that the bracketed superscript j in (12) indicates now the fixed-point iteration. Additionally, a damped formulation

$$z^{(j+1)} = \varepsilon F(z^{(j)}) + (1 - \varepsilon)\, z^{(j)}, \quad \varepsilon \in (0, 1] \quad (13)$$

of the fixed-point integration (12) can be used to improve the convergence properties.

For simplicity and an efficient FPGA implementation, the explicit Euler forward approach is used in this contribution for the numerical integration of the system as well as the adjoint dynamics. Thus, the related integration step (9) has the form

$$x^{i+1} = x^i + hf\left(x^i, u^i\right), \ i = 0, \dots, N-1 \quad (14)$$

with the fixed step size $h = \frac{T}{N}$. The associated fixed-point iterations (12) and (13), respectively, then follow by simply using the system function f as iteration function, i.e. $\phi = f$. The fixed-point formulation for the integration of the adjoint dynamics (5b) can be obtained in a similar way. According to (5b) and the Euler forward method (cf. (14)), the related iteration function for integrating the adjoint dynamics is $\phi = -H_x$. Also note that due to the backward integration a negative step size $h = -\frac{T}{N}$ has to be chosen.

IV. EXAMPLES AND SYNTHESIS RESULTS

The synthesis of the nonlinear MPC algorithm (cf. Section II-B) is demonstrated for two examples. In addition, the results are evaluated by means of two reference FPGAs in the sense that the required number of logic components is compared.

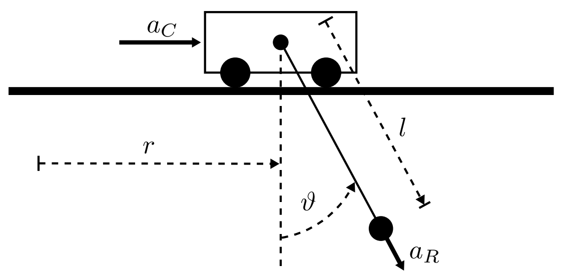


Fig. 2. Schematics of the overhead crane.

A. Overhead crane

Figure 2 shows the schematics of a two-dimensional overhead crane consisting of a moving cart and a mounted rope that can also be altered in length. The nonlinear system dynamics of the overhead crane is given by [22]

$$\ddot{r} = a_C,\ \ddot{l} = a_R,\ \ddot{\vartheta} = -\frac{1}{r}\left(g\sin\vartheta + a_C\cos\vartheta + 2\dot{l}\dot{\vartheta}\right) \quad (15)$$

with the gravitational constant g. The states $x \in \mathbb{R}^6$ of the system are the cart position $x_1 = r$, the rope length $x_3 = l$ and the angle $x_5 = \vartheta$ to the vertical direction as well as the corresponding velocities $x_2 = \dot{r}$, $x_4 = \dot{l}$ and $x_6 = \dot{\vartheta}$. The cart and the rope accelerations $u_1 = a_C$ and $u_2 = a_R$ serve as control inputs which are subject to the constraints

$$u^- = \left[-3\,\text{m/s}^2, -3\,\text{m/s}^2\right]^\mathsf{T}, \quad u^+ = \left[3\,\text{m/s}^2, 3\,\text{m/s}^2\right]^\mathsf{T}. \quad (16)$$

The integral and terminal cost in (1a) are set to

$$V(x) = \Delta x^\mathsf{T} P \Delta x,\ L(x,u) = \Delta x^\mathsf{T} Q \Delta x + u^\mathsf{T} R \Delta u \quad (17)$$

with $P = Q = \operatorname{diag}(10,1,10,1,10,1)$ and $R = \operatorname{diag}(0.01, 0.01)$. Additionally, $\Delta x := x - x_\text{SP}$ and $\Delta u := u - u_\text{SP}$ denote the distance to a desired setpoint x_SP and u_SP, respectively. The corresponding prediction horizon, the sampling time, as well as the number of discretization points and gradient iterations are

$$T = 1\,\text{s}, \quad N = 30, \quad \Delta t = 1\,\text{ms}, \quad N_\text{grad} = 2. \quad (18)$$

Simulation results for the stabilization of the setpoint

$$x_\text{SP} = [0, 0, 1.2\,\text{m}, 0, 0, 0, 0]^\mathsf{T}, \quad u_\text{SP} = [0, 0]^\mathsf{T} \quad (19)$$

starting from the initial condition

$$x_0 = [-2\,\text{m}, 0, 0.2\,\text{m}, 0, 0, 0, 0] \quad (20)$$

can be seen in Figure 3. The related cost function is also illustrated revealing an asymptotic decrease. The integration of the system (5a) as well as the adjoint dynamics (5b) is performed by means of the fixed-point scheme presented in Section III-B with $M = 5$ iterations.

B. Continuous stirred tank reactor

The continuous stirred tank reactor (CSTR) (e.g. see [23]) depicted in Figure 4 is fed by the dilution rate q with the temperature ϑ_in and the inlet concentration c_in. The cooling jacket's temperature ϑ_c is affected by the cooling power $\dot{Q}$ applied to the colling jacket. The CSTR comprises the

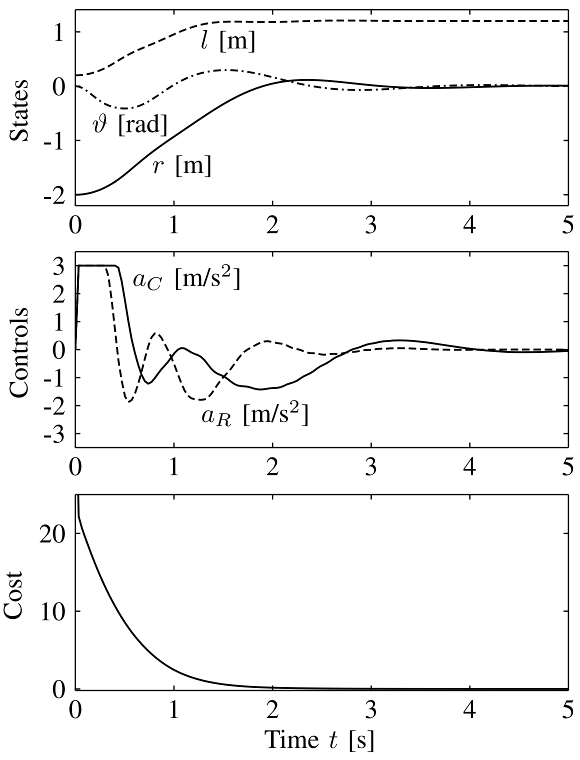
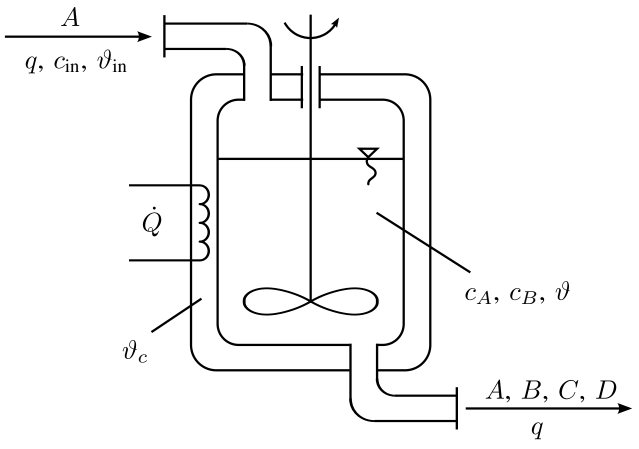


Fig. 3. MPC trajectories for the overhead crane.

reactions of the educt A to the desired product B and the parallel reactions to the undesired byproducts C and D. A nonlinear model of the reactor based on mass and energy balances is given by [23]

$$\dot{c}_A = -k_1(\vartheta)c_A - k_2(\vartheta)c_A^2 + (c_\text{in} - c_A)\,q \quad (21a)$$
$$\dot{c}_B = k_1(\vartheta)\,(c_A - c_B) - c_B q \quad (21b)$$
$$\dot{\vartheta} = h(c_A, c_B, \vartheta) + \alpha\,(\vartheta_c - \vartheta) + (\vartheta_\text{in} - \vartheta)\,q \quad (21c)$$
$$\dot{\vartheta}_c = \beta\,(\vartheta - \vartheta_c) + \gamma\dot{Q} \quad (21d)$$

where c_A and c_B denote the concentrations of educt A and product B and ϑ and ϑ_c are the temperatures within the reactor and the cooling jacket, respectively. The dilution rate and the



A
q, c_in, ϑ_in
$\dot{Q}$
c_A, c_B, ϑ
ϑ_c
A, B, C, D
q

Fig. 4. Schematics of the CSTR [23].

cooling power are the control inputs $u = [q, \dot{Q}]^\mathsf{T} \in [u^-, u^+]$ constrained by

$$u^- = \left[3\,\mathrm{h}^{-1}, -9000\,\mathrm{kJ/h}\right]^\mathsf{T}, \quad u^+ = \left[35\,\mathrm{h}^{-1}, 0\,\mathrm{kJ/h}\right]^\mathsf{T}. \tag{22}$$

The enthalpy $h(c_A, c_B, \vartheta)$ in (21c) is

$$\begin{aligned} h(c_A, c_B, \vartheta) = -\delta\big[&k_1(\vartheta)\,(c_A \Delta H_{AB} + c_B \Delta H_{BC}) \\ &+ k_2(\vartheta) c_A^2 \Delta H_{AD}\big] \end{aligned} \tag{23}$$

and the reaction rates $k_1(\vartheta)$ and $k_2(\vartheta)$ are modelled with the Arrhenius functions

$$k_i(\vartheta) = k_{i0} \exp\left(\frac{-E_i}{\vartheta/{}^\circ\mathrm{C} + 273.15}\right), \quad i = \{1, 2\}. \tag{24}$$

The parameters of the CSTR can be found in [23]. Investigations in view of an FPGA implementation revealed that the exponential functions in the Arrhenius terms (24) require many bits for the number representation. Hence, the functions are approximated within the region of operation $\vartheta \in [80\,{}^\circ\mathrm{C}, 120\,{}^\circ\mathrm{C}]$ by the 4-th order polynomials

$$k_i(\vartheta) \approx \sum_{j=0}^{4} a_{ij}\,(\vartheta/{}^\circ\mathrm{C} - 100)^j\,. \tag{25}$$

The cost (1a) is also chosen quadratically (cf. (17)) with the weights $P = Q = \mathrm{diag}\,(0.2, 1, 0.5, 0.2)$ and $R = \mathrm{diag}\,\left(0.5, 5 \cdot 10^{-3}\right)$. The prediction horizon, the sampling time, as well as the number of discretization points and gradient iterations are set to

$$T = 1200\,\mathrm{s}, \quad N = 30, \quad \Delta t = 1\,\mathrm{s}, \quad N_{\text{grad}} = 2. \tag{26}$$

Figure 5 illustrates the simulation results for the model predictively controlled CSTR performing a setpoint change from the initial state

$$x_0 = \left[2.02\,\mathrm{kmol/m^3}, 1.07\,\mathrm{kmol/m^3}, 100\,{}^\circ\mathrm{C}, 97.1\,{}^\circ\mathrm{C}\right]^\mathsf{T} \tag{27}$$

to the desired setpoint

$$\begin{aligned} x_{\text{SP}} &= \left[1.37\,\mathrm{kmol/m^3}, 0.95\,\mathrm{kmol/m^3}, 110\,{}^\circ\mathrm{C}, 108.6\,{}^\circ\mathrm{C}\right]^\mathsf{T} \\ u_{\text{SP}} &= \left[5\,\mathrm{h}^{-1}, -1190\,\mathrm{kJ/h}\right]^\mathsf{T}. \end{aligned} \tag{28}$$

The fixed-point scheme with $M = 5$ iterations was also used for the integration procedure (cf. Section IV-A).

C. Synthesis results

In this section the synthesis results for the overhead crane as well as the CSTR are presented using the standard (sequential) and the fixed-point based (cf. Section III-B) iteration scheme. The results are verified for feasibility in the sense that the required number of logic elements is compared to available FPGAs. To this end, a Zynq FPGA of type *XC7Z020 CLG484-1* as well as an Artix FPGA of type *XC7A200T-2FBG676C* are used as reference components. The corresponding number of logic elements is given in Table I.

TABLE I. DATA OF THE REFERENCE FPGAS.

	No. of DSPs	No. of LUTs	No. of FFs	No. of BRAMs
Zynq	220	53200	106400	280
Artix	740	129000	258000	730

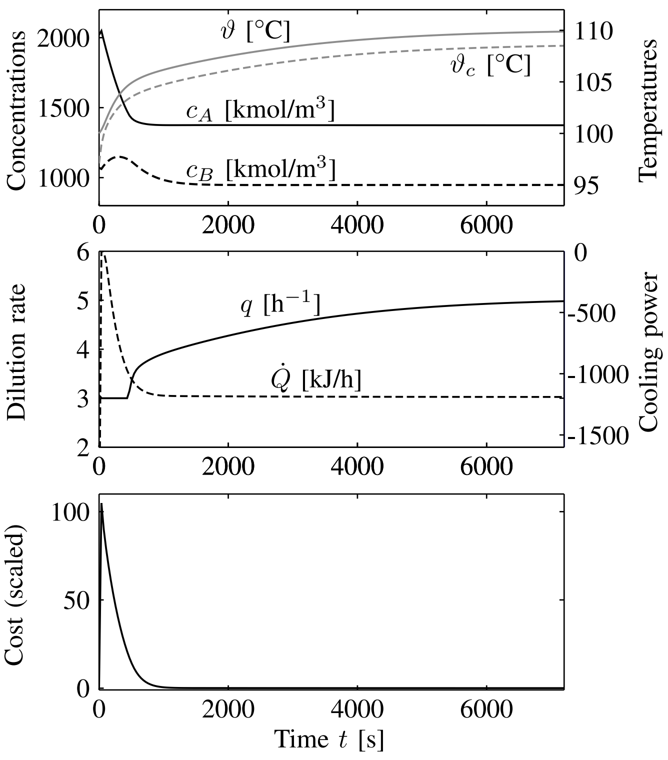


Fig. 5. MPC trajectories for the CSTR.

1) MATLAB *HDL Coder:* The synthesis results for the overhead crane obtained with the MATLAB HDL Coder are illustrated in Table II where the sequential integration scheme was used. The approximated number of required elements was estimated after the HDL code generation by means of the software tool Vivado (version 2013.2, 64-bit) from Xilinx.

TABLE II. MATLAB HDL CODER SYNTHESIS RESULTS FOR THE OVERHEAD CRANE.

Examples	No. of DSPs	No. of LUTs	No. of FFs	No. of BRAMs
Crane	18128	1623531	1557	1

It can be seen that the results for the overhead crane cannot be implemented on any of the both reference FPGAs with regard to the number of DSPs and LUTs. Due to this considerable number of required resources, a synthesis procedure of the CSTR with the MATLAB HDL Coder was omitted. The synthesis of both examples in combination with the fixed-point integration scheme was also not further investigated.

2) Vivado HLS: Alternative synthesis results can be achieved by using the tool Vivado HLS. Compared to the results from Section IV-C1, the estimated computation time in each MPC step on the FPGA is also provided at this point. Based on the clock rate f_{clk}, the corresponding computation time is given by

$$T_{\text{MPC}} = \frac{n_{\text{clk}}}{f_{\text{clk}}} \tag{29}$$

where n_{clk} is the corresponding number of clock pulses. For the following discussion a clock rate $f_{\text{clk}} = 200\,\mathrm{MHz}$ is used.

TABLE III. Vivado HLS synthesis results for the overhead crane and the CSTR.

Examples	DSPs	LUTs	FFs	BRAMs	Comp. time [ms]	Integration scheme
Crane	102	37704	39828	19	50.6	sequential
	160	94220	109999	19	10.4	fixed-point
CSTR	168	32627	33468	12	60.4	sequential
	330	120460	116493	12	20.0	fixed-point

With regard to the synthesis procedure, the word length for each state and control variable was set to 28 bit (20 fractional bit) for the crane example and to 24 bit (14 fractional bit) for the CSTR, respectively. The remaining synthesis settings (pipeline techniques, degree of parallelism) were chosen in such a way as to obtain an acceptable tradeoff between required resources and computation times. The related results are illustrated in Table III. The results reveal that the fixed-point integration scheme requires more resources than the sequential procedure due to the parallelization. It can also be observed that the model predictively controlled crane as well as the CSTR in combination with the sequential integration scheme can be implemented on both reference FPGAs (cf. Table I). However, a corresponding fixed-point version can only be used on the Artix FPGA.

3) Comparison: Computation times of the MPC with sequential integration running on a standard PC[2] and a dSPACE[3] system are shown in Table IV. The results of the FPGA implementation with the fixed-point integration scheme show that lower computation times can be achieved and hence demonstrate the potential of the nonlinear MPC on an FPGA.

TABLE IV. Computation times of the model predictively controlled crane and CSTR.

Examples	Computation time [ms]	
	PC	dSPACE
Crane	26	172
CSTR	29	180

V. Conclusion

This paper described an implementation strategy of a nonlinear model predictive control scheme on FPGA systems. Based on a real-time MPC algorithm, two different synthesis procedures using the Matlab HLD Coder as well as the tool Vivado HLS were discussed. Synthesis results for two examples were illustrated in order to demonstrate the potential of an FPGA implementation of a nonlinear MPC.

Future work concerns improving the synthesis results by using further optimization strategies in order to reduce the required number of logic elements as well as the computation time. Moreover, the discussed real-time MPC scheme shall be implemented and tested on a common used FPGA system.

References

[1] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[2] E. F. Camacho and C. Bordons, *Model Predictive Control*. London: Springer, 2003.

[3] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*. London: Springer, 2011.

[4] T. Ohtsuka, "A continuation/GMRES method for fast computation of nonlinear receding horizon control," *Automatica*, vol. 40, no. 4, pp. 563–574, 2004.

[5] B. Houska, H. J. Ferreau, and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[6] D. DeHaan and M. Guay, "A real-time framework for model-predictive control of continuous-time nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 52, no. 11, pp. 2047–2057, 2007.

[7] V. M. Zavala and L. Biegler, "The advanced-step NMPC controller: Optimality, stability and robustness," *Automatica*, vol. 45, no. 1, pp. 86–93, 2009.

[8] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[9] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," in *Proceedings of the 51th IEEE Conference on Decision and Control*, 2012, pp. 668–674.

[10] P. K. Chan and S. Mourad, *Digital Design Using Field Programmable Gate Arrays*. New Jersey: Prentice Hall, 1994.

[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 and Applications*, vol. 6, no. 8, pp. 1055–1063, 2012.

[12] J. L. Jerez, G. A. Constantinides, E. C. Kerrigan, and K.-V. Ling, "Parallel MPC for real-time FPGA-based implementation," in *Proceedings of the 18th IFAC World Congress*, 2011, pp. 1338–1343.

[13] K. Basterretxea and K. Benkrid, "Embedded high-speed model predictive controller on a FPGA," in *Proceedings of the 2011 NASA/ESA Conference on Adaptive Hardware and Systems*, 2011, pp. 327–335.

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

[15] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded predictive control on an FPGA using the fast gradient method," in *Proceedings of the 2013 European Control Conference*, 2013.

[16] G. Knagge, A. Wills, A. Mills, and B. Ninness, "ASIC and FPGA implementation strategies for model predictive control," in *Proceedings of the 2009 European Control Conference*, 2009, pp. 144–149.

[17] D. E. Kirk, *Optimal Control Theory: An Introduction*. Mineola: Dover Publications, 1970.

[18] J. C. Dunn, "On L^2 sufficient conditions and the gradient projection method for optimal control problems," *SIAM Journal on Control and Optimization*, vol. 34, no. 4, pp. 1270–1290, 1996.

[19] K. Graichen and B. Käpernick, "A real-time gradient method for nonlinear model predictive control," in *Frontiers of Model Predictive Control*, T. Zheng, Ed. InTech, 2012, pp. 9–28, http://www.intechopen.com/books/frontiers-of-model-predictive-control.

[20] K. Graichen and A. Kugi, "Stability and incremental improvement of suboptimal MPC without terminal constraints," *IEEE Transactions on Automatic Control*, vol. 55, no. 11, pp. 2576–2580, 2010.

[21] J. P. Elliott, *Understanding Behavioral Synthesis: A Practical Guide to High-Level Design*, 2nd ed. Boston: Kluwer Academic Publishers, 2000.

[22] B. Käpernick and K. Graichen, "Model predictive control of an overhead crane using constraint substitution," in *Proceedings of the 2013 American Control Conference*, 2013, pp. 3979–3984.

[23] R. Rothfuss, J. Rudolph, and M. Zeitz, "Flatness based control of a nonlinear chemical reactor model," *Automatica*, vol. 32, no. 10, pp. 1433–1439, 1996.

[2]Intel Core i5 @ 2.67 GHz and 4 GB memory.

[3]dSPACE DS1103 with a PowerPC 750 GX @ 1 GHz and 96 MB memory.