Optimized FPGA Implementation of Model Predictive Control for Embedded Systems Using High Level Synthesis Tool

S. Lucia, *Member*, *IEEE*, D. Navarro, O. Lucia, *Senior Member*, *IEEE* , P. Zometa, and R. Findeisen

***Abstract*— Model predictive control is an optimization-based strategy for high-performance control that is attracting increasing interest. While model predictive control (MPC) requires the online solution of an optimization problem, its ability to handle multivariable systems and constraints makes it a very powerful control strategy specially for MPC of embedded systems, which have an ever increasing amount of sensing and computation capabilities. We argue that the implementation of MPC on field programmable gate arrays (FPGAs) using automatic tools is nowadays possible, achieving cost-effective successful applications on fast or resource-constrained systems. The main burden for the implementation of MPC on FPGAs is the challenging design of the necessary algorithms. We outline an approach to achieve a software-supported optimized implementation of MPC on FPGAs using high level synthesis tools and automatic code generation. The proposed strategy exploits the arithmetic operations necessaries to solve optimization problems to tailor an FPGA design, which allows a trade-off between energy, memory requirements, cost, and achievable speed. We show the capabilities and the simplicity of use of the proposed methodology on two different examples and illustrate its advantages over a microcontroller implementation.**

***Index Terms*- Field programmable gate array, model predictive control, high level synthesis.**

I. INTRODUCTION

The amount of required and available sensing and computation capabilities of existing and newly designed systems is growing rapidly. In this paper, we focus on model predictive control (MPC) [1], which is an optimization-based, advanced control strategy that uses a mathematical model to predict the future behavior of a system. The predictions are used to obtain a sequence of control inputs that minimize a desired performance criterion and result in satisfaction of the required constraints. Fig. 1 shows the central idea of MPC, in which the predictions of the model states (x_k) at each sampling time k are used to obtain a sequence of control input vectors (u_k) that minimizes a desired performance criterion and result in satisfaction of the required constraints. The application of MPC demands the real-time solution of a numerical optimization problem. Therefore, its applications are traditionally limited to slow systems, e.g. chemical plants or petrochemical systems.

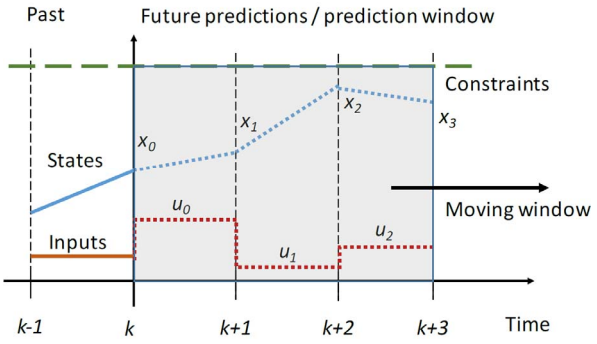


Fig. 1. Model predictive control.

However, due to the significant advances in tailored algorithms and hardware, MPC is increasingly becoming of interest for very fast embedded and resource-limited systems. Some examples include industrial electronics applications [2, 3] such as inverters [4-6], rectifiers [7], matrix converters [8], or engine control [9].

New hardware architectures have been studied for the implementation of MPC [10], including programmable logic controllers (PLCs) [11], low-cost microcontrollers [12, 13], field programmable gate arrays (FPGAs) [14, 15] [16], and application specific integrated circuits (ASICs) [17]. The choice of hardware architecture is often a trade-off between cost, energy consumption and required performance.

Advances in FPGA technology have led to inexpensive devices with increasing digital resources. This opens significantly the spectrum of potential applications. Moreover, FPGA technology enables the optimized use of parallel calculations as well as ad-hoc digital hardware development, increasing the performance to levels that are not achievable using any other fixed architecture implementation. As a result of the technology development, FPGA technology has become an alternative for MPC controllers implementation [18-20] due to its high performance and cost-effectiveness, enabling applications at MHz rates. For cost-efficient solutions, the

Manuscript received February 4th, 2017; revised April 25th, 2017; accepted, June 15th, 2017.

This work was partly supported by the Spanish MINECO under Project TEC2016-78358 and by the DGA-FSE.

S. Lucia is with the Laboratory of Internet of Things for Smart Buildings, Technische Universität Berlin, Ernst-Reuter-Platz 7, 10587 Berlin, Germany and the Einstein Center Digital Future, Wilhelmstr. 67, 10117 Berlin, Germany (e-mail: sergio.lucia@tu-berlin.de). O. Lucia and D. Navarro are with the Department of Electronic Engineering and Communications, University of Zaragoza, SPAIN (phone: +34976761000, e-mail: olucia@unizar.es). P. Zometa and R. Findeisen are with the Laboratory for Systems Theory and Automatic Control, Otto-von-Guericke University Magdeburg, Universitätsplatz 2, 39106, Magdeburg, Germany.

system can be ported to an application-specific integrated circuit (ASIC).

Although theoretical issues of MPC control have been deeply studied [21], there are still many challenges for the fast and economically viable implementation and use of MPC. Among those, the effort to design, optimize and implement such controller is one of the most relevant challenges. In order to enable a rapid and economic FPGA implementation, this paper proposes a design and optimization procedure using high level synthesis (HLS) [22, 23] which has proved to be an effective way to implement industrial controllers [24, 25]. In comparison to [26], which also presents the use of HLS tools for embedded optimization, we present explicitly how to tailor and optimize the FPGA implementation paying special attention to the arithmetic operations of MPC controllers. This allows to consider different trade-offs between energy, speed, and memory requirements, and to provide several guidelines for designers. The main benefits of the proposed design workflow are the fast testing of different designs, which enable a time-effective way to analyze a large set of FPGA implementations, which cannot be done manually.

We focus here on the software-supported implementation of predictive controllers on FPGAs, mainly because of its high-performance, cost effectiveness and flexibility. Currently, there are tools for the implementation of MPC that generate simple code that can be used e.g. on microcontrollers, see CVXGEN [27] or ECOS/QCML [28]. The use of FPGAs for efficient MPC implementations has been presented, e.g., in [29]. However, there the FPGA design is performed manually. While a toolbox for FPGA prototyping is presented in [26], there are currently no available tools to automatically design and optimize an FPGA implementation of MPC starting from a high-level description of the control problem, enabling the application of MPC techniques to non-experts in the field. This represents the main challenge for an optimized use of MPC on FPGAs. This challenge can be efficiently overcome as it is shown in this paper.

This paper is organized as follows. Section II gives a brief overview of the MPC problem formulation and the basic algorithms that are studied for implementation. Section III details the proposed workflow for an efficient and cost-efficient MPC FPGA implementation using HLS as well as the optimization procedure used. Section IV presents a design example including its formulation, implementation and optimization process and the achieved results. Section V summarizes the main conclusions of the paper.

II. Model Predictive Control Problem

A. Formulation and basic solution algorithms

Model Predictive Control is based on the repeated solution of an optimization problem at each sampling time. A mathematical model is used to predict the future behavior of the system until a given prediction horizon and a sequence of optimal control inputs is obtained by minimizing the chosen cost function subject to given constraints (see also Fig. 1). For embedded systems, usually linear models are considered, which represent a linear system or a linearization of the actual nonlinear system, and can be represented as:

$$x^+ = Ax + Bu \tag{1}$$

where $x \in \mathbb{R}^{n_x}$ denotes the states and $u \in \mathbb{R}^{n_u}$ represents the control inputs.

Usually, a quadratic cost function with positive semidefinite weight matrices and affine constraints are considered, so that the optimization problem that needs to be solved at each sampling time is convex and has the form:

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{u}}{\text{minimize}} \quad & \textstyle\sum_{k=0}^{N-1}(x_k^T Q x_k + u_k^T R u_k) + x_N^T P x_N \\ \text{subject to:} \quad & x_{k+1} = Ax_k + Bu_k \\ & c_{min} \le Fx_k + Gu_k \le c_{max}. \end{aligned} \tag{2}$$

Here, Q, R, P are tuning parameters of the cost function which penalize state and input deviations. F, G, c_{min}, c_{max} define the constraints that are considered for the control task, while N is the prediction horizon. Embedding the initial condition x_0 and the linear dynamics (1) in the formulation of the cost function and constraints, the optimization problem can be transformed into a so-called condensed formulation [30]. The condensed problem only has the control inputs as optimization variables, which is beneficial for embedded implementations. The resulting equivalent formulation of the optimization problem (2) can be expressed in condensed form as:

$$\begin{aligned} \underset{\boldsymbol{u}}{\text{minimize}} \quad & \boldsymbol{u}^T H \boldsymbol{u} + g(x_0)^T \boldsymbol{u} \\ \text{subject to:} \quad & V\boldsymbol{u} - v(x_0) \le 0. \end{aligned} \tag{3}$$

Many efficient algorithms have been proposed in the last years to solve (3). First-order methods for solving the quadratic program (3) are, e.g., presented in [12, 31]. Second order approaches are provided in [32, 33]. We focus on the use of a combination of the fast gradient method (FGM) [34] that can be used when only input constraints are considered, combined with the use of an augmented Lagrangian method (ALM), which allows to handle state constraints as proposed in [31]. We review this approach in the following.

For simplicity of notation, in the remainder of the paper we denote the cost function of the considered optimization problem (3) as $f_0(\boldsymbol{u}, x_0)$ and the equality constraints as $f_c(\boldsymbol{u}, x_0) =$

$V\boldsymbol{u} - v(x_0)$ (inequality constraints are transformed into equality constraints by introducing slack variables). As it is common in the field of numerical optimization [30], the augmented Lagrangian $\mathcal{L}(\cdot)$ is defined as:

$$\mathcal{L}(\boldsymbol{u}, x_0, \lambda) = f_0(\boldsymbol{u}, x_0) + \sum_{l=1}^{n_c} \lambda_l f_c(\boldsymbol{u}, x_0) + \frac{\mu}{2} \sum_{l=1}^{n_c} f_c^2(\boldsymbol{u}, x_0). \quad (4)$$

Here λ_l is the Lagrange multiplier associated to the constraint l, μ is a tuning parameter of the regularization term and n_c denotes the number of constraints. The idea of augmented Lagrangian methods is that solving an unconstrained optimization problem with $\mathcal{L}(\cdot)$ as cost function and using the optimal value of the multipliers $\lambda_l^\star$ leads to the solution $\boldsymbol{u}^\star$ of the original (constrained) optimization problem [30]. In the combined FGM+ALM method we use the fast gradient method to find the solution of the unconstrained problem given by the augmented Lagrangian and we use the ALM method to iteratively update the values of the Lagrange multipliers to reach its optimal value.

The fast gradient method is a first order iterative method to efficiently solve input-constrained optimization problems. It is especially suited for embedded platforms due to its low memory requirements and high convergence rate. It consists of two main steps. In the first step, the next candidate of the optimal solution $\boldsymbol{u}^+$is computed:

$$\boldsymbol{u}^+ = \mathcal{P}_{\mathcal{U}}\left(\boldsymbol{w} - \frac{1}{L}\nabla_u \mathcal{L}(\boldsymbol{w}, x_0, \lambda)\right). \quad (5)$$

Here L is a Lipschitz constant that can be calculated from the problem data, i.e., the system description and the control task description, and $\mathcal{P}_{\mathcal{U}}$ denotes a projection onto the set $\mathcal{U}$, which is defined by the input constraints. In most cases, the inputs are box-constrained, and such projection is a simple (and computationally cheap) saturation operation given the minimum and maximum possible values of the control inputs. The operator $\nabla_u(\cdot)$ denotes the partial derivative of the augmented Lagrangian with respect to the vector of control inputs $\boldsymbol{u}$.

In the second part of the fast gradient method an extra-step is computed [34]:

$$\boldsymbol{w} = \boldsymbol{u}^+ + v(\boldsymbol{u}^+ - \boldsymbol{u}), \quad (6)$$

where $v = \left(\sqrt{L} - \sqrt{\phi}\right)\left(\sqrt{L} - \sqrt{\phi}\right)^{-1}$. Here, $\phi > 0$ is a strong convexity constant which can be computed offline [31]. The fast gradient method for a number of iterations j_{in} and a Lagrange multiplier λ is summarized in Algorithm 1.

Table I. Algorithm 1: Fast gradient method (FGM)

Require: initial guess $\boldsymbol{u}$, multiplier $\boldsymbol{\lambda}$, state $\boldsymbol{x_0}$

1. **set** $\boldsymbol{w} = \boldsymbol{u}$
2. **for** $\boldsymbol{j = 0}$ **until** $\boldsymbol{j_{in} - 1}$ **do**:
3. $\quad \boldsymbol{u^+ = \mathcal{P}_{\mathcal{U}}\left(w - \frac{1}{L}\nabla_u \mathcal{L}(w, x_0, \lambda)\right)}$
4. $\quad \boldsymbol{w = u^+ + v(u^+ - u)}$
5. **end for**

Return $\boldsymbol{u^+}$

If a given optimization problem only has input constraints, Algorithm 1 is directly able to obtain a solution (the value of the multiplier λ is then irrelevant). In the case of state constraints, an optimal value of the multiplier has to be found to obtain the solution of the original (constrained) problem. This is achieved via an iterative update of the multiplier:

$$\lambda \leftarrow \lambda + \mu\big(V\boldsymbol{u} - v(x_0)\big), \quad (7)$$

where $\mu > 0$ is a penalty parameter.

When state constraints are present, a combination of the fast gradient method presented in Algorithm 1 and the multiplier update in (7) is used. A summary of the ALM + FGM scheme with i_{ex} iterations is given in Algorithm 2.

Table II. Algorithm 2: ALM + FGM

Require: initial guess $\boldsymbol{u}$, multiplier $\boldsymbol{\lambda}$, state $\boldsymbol{x_0}$

1. **for** $\boldsymbol{i = 0}$ **until** $\boldsymbol{i_{ex} - 1}$ **do**:
2. $\quad$ compute $\boldsymbol{u^+}$ using Algorithm 1
3. $\quad \boldsymbol{\lambda \leftarrow \lambda + \mu\big(Vu - v(x_0)\big)}$
4. **end for**

Return $\boldsymbol{u^+, \lambda}$

III. FPGA IMPLEMENTATION OF MPC USING HLS

A. Design workflow

A two-step approach is proposed (Fig. 2(a)) to enable an optimized yet simple FPGA design for MPC controllers. As a first step, we use the tailored MPC code generation tools µAOMPC [35] and included novel extensions to transform a simple description of the control problem to code that can be used in a second step by a High Level Synthesis (HLS) tool, such as Vivado HLS [36]. The required tailored code is automatically generated by the extended version of µAOMPC, which implements the ALM+FGM algorithm presented in Table II relying only on additions and multiplications. The proposed extended version modifies µAOMPC 0.4.0 by including definitions of each needed operation (see Table III) with fixed-size arrays. This is necessary because the Vivado HLS tool does not allow variable-size array optimization. Variable size arrays were used in µAOMPC to achieve generated codes of smaller size. However, the modified version of the code will take advantage of fixed-size arrays to optimize the FPGA

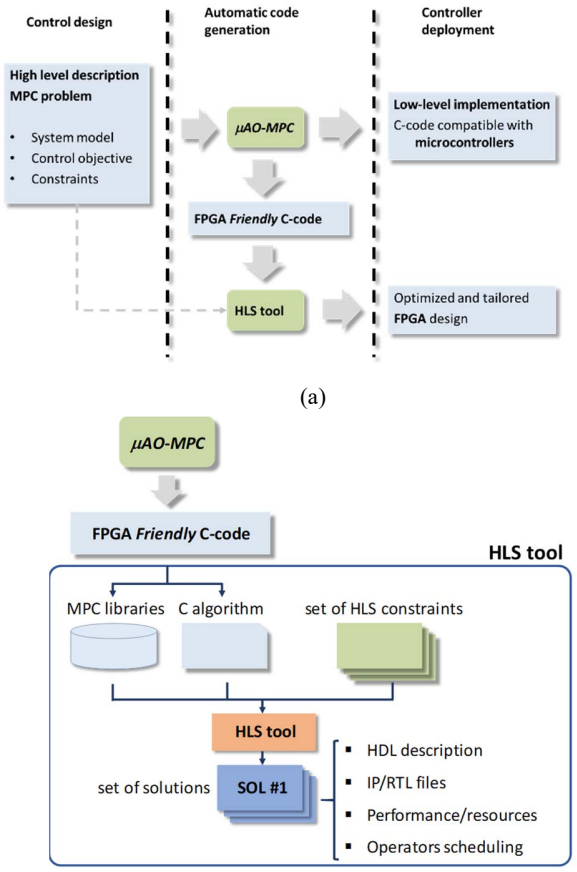


Fig. 2. Design workflow for FPGA implementation of MPC using HLS: global overview (a) and HLS tool workflow (b).

implementation by unrolling and pipelining the digital design, as it will be explained later.

This approach offers several advantages for computations on embedded platforms. The HLS tool processes the auto-generated code taking into account explicitly the particularities of the optimization algorithms presented in Table I and Table II. The result of the proposed approach is an automatic and optimized FPGA design for MPC, which depends on the problem data and designer requirements. The proposed approach offers several design alternatives to achieve different trade-offs between FPGA size and computation time. Alternatively, µAOMPC can be used to generate C-code that can be directly used on microcontrollers, without any additional libraries.

The high level synthesis tool workflow to implement the MPC controller is summarized in Fig. 2(b). Inputs are the C-code containing the MPC algorithm as well as the required libraries and additional constrains to be considered by the HLS tool. As discussed later, these constraints must be designed carefully taking the MPC problem into account in order to optimize both the performance and the needed digital resources. The output of

Table III. Arithmetic operations

Line Algorithm 1	Operation type	Dimension
3	A * b	(nu * N, nx) × nx
for j = 1 : n_iter	-	-
3	A * b	(nu * N) × nu *N
3	a + b	nu * N
3	a - b	nu * N
3	saturate(a)	nu * N
4	a - b	nu * N
4	scale(a)	nu * N
4	a + b	nu * N

the HLS process is an optimized register transfer level (RTL) implementation using a hardware description language ready for both simulation and/or synthesis. One of the main benefits of this design flow is the fast design process, which enables to analyze a large set of implementations in a time-effective manner, which is not possible to address manually. The main novel contributions of the proposed design flow include the extension of the tool µAOMPC to generate C-code that can be directly used by HLS tools, as well as the optimized use of HLS constraints to take advantage of the arithmetic operations that are necessary to solve the typical optimization problems that arise within MPC.

B. Optimization of the FPGA design

The optimal implementation of model predictive control with respect to FPGA resources and computational speed depends strongly on the matrix-vector operations that are necessary to solve the optimization problem at each sampling time. Table III summarizes all the necessary operations, as a function of the number of states (nx), the number of control inputs (nu), and the prediction horizon (N). These operations and their relation with the FPGA implementation are key during the design process in order to optimize the implementation performance and digital resource usage. The *saturate* operation is the projection operator defined in the FGM algorithm for the case of box-constrained inputs and the *scale* operation denotes the multiplication of all elements of a vector by a scalar. A similar analysis can be performed for the FGM+ALM algorithm. It is omitted here for simplicity in the presentation.

From Table III, it is clear that the matrix-vector multiplication inside the loop is the most resource consuming operation. The optimization analysis will focus on this aspect.

In order to explore and optimize the design space using high level synthesis, several design constraints must be carefully chosen:

i) unrolling permits unrolling a certain loop to compute arithmetic operations using parallel processing. This significantly improves the implementation performance at the cost of additional digital resources.

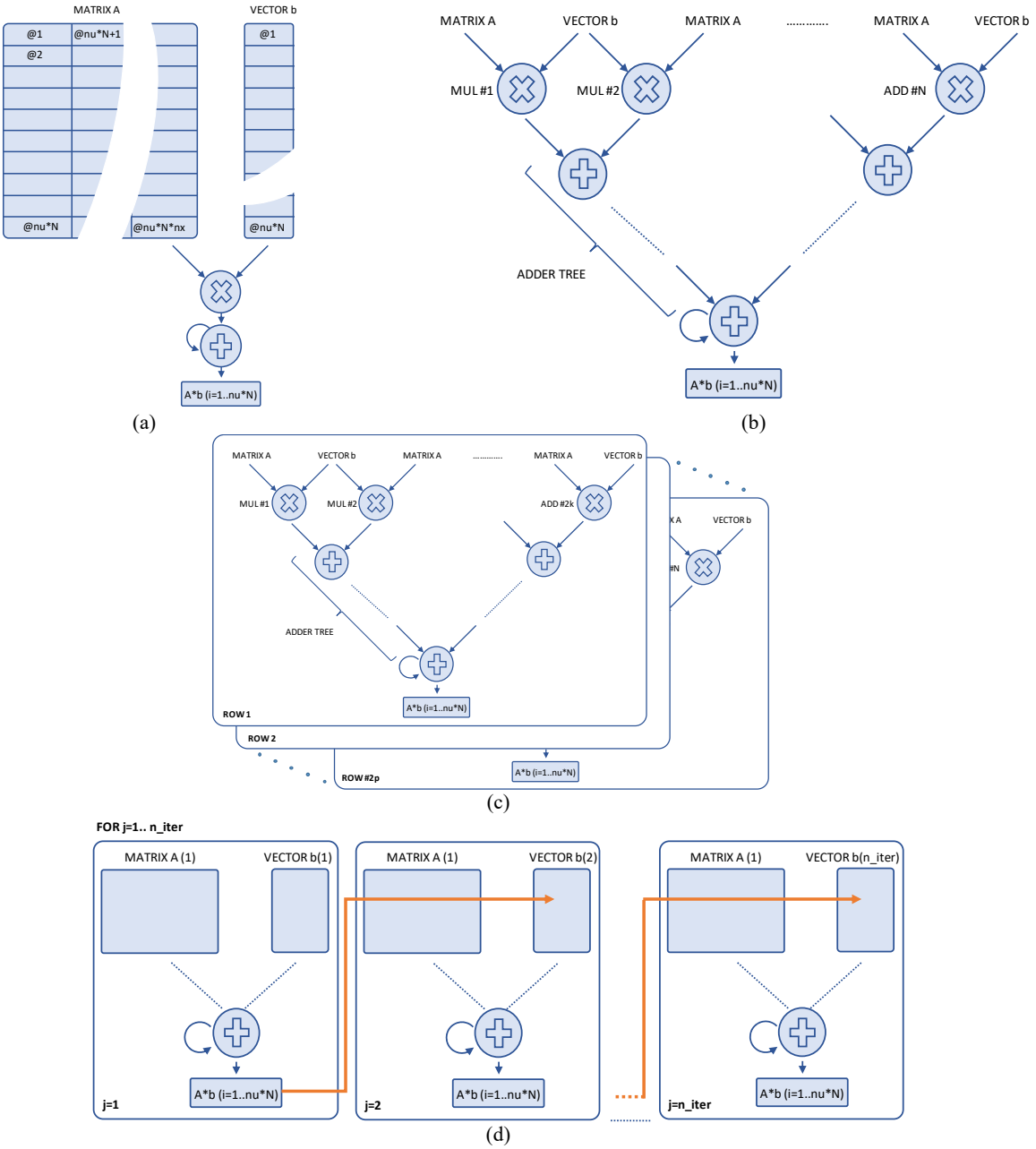


Fig. 3. Arithmetic operation implementation: standard implementation (a), unroll inner loop (b), unroll inner loop plus row parallelization (c), and unroll outer loop (d).

Fig. 3 (a) shows the standard arithmetic operation implementation, which requires a single multiplier plus an accumulator which sweeps all matrix elements. This implementation, however, is the slowest and it can limit real-time computing with complex problems or large prediction horizons. In order to improve the performance, the inner loop can be unrolled (Fig. 3(b)), which requires additional multipliers plus an adder tree. Depending on the unroll level, a balance between performance and digital resource usage must be met. Additionally, several rows of the matrix multiplication can be computed at the same time. This implies the replication of the aforementioned structure as many times as the number of rows to be computed simultaneously (Fig. 3(c)).

Usually, unrolling is chosen to be an even number to take the most of dual port memories. Finally, if the performance needs to be further improved, the outer loop can be also unrolled (Fig. 3(d)). This means that partial data set from an iteration can be used to start computing subsequent iterations, enabling a

significant gain in computing time. It is important to note that this enhancement, possible using HLS, is in general not feasible by hand-coding due to the complex arithmetic data paths.

ii)*pipelining* is a common practice in digital design used to increase the maximum clock frequency when several arithmetic operations are performed sequentially. In this case, the constraint is used to set the target initialization interval as parameters, enabling an increased throughput and clock frequency at the cost of additional digital resources. Considering modern control problems and FPGA technology, this is a strongly recommended strategy due to the high number of flip-flops commonly available even in inexpensive devices.

iii) *inlining* enables cross optimization among different C functions organized in a hierarchy. Unlike straight-forward optimization, which considers each function as separate black boxes, this directive enables further optimization and resource sharing to increase the performance and decrease the digital resource usage.

IV. Design Examples

In order to prove the advantages of the proposed HLS scheme and to discuss implementation details, this section presents several representative design examples using HLS for MPC implementation in FPGA. We discuss two examples, a small dc-motor problem and a larger chain of masses problem to highlight the differences in arithmetic complexity and optimization.

A. Considered example problems

i) Dc-Motor: A simple dc-motor can be represented by the following discrete-time linear system:

$$A = \begin{bmatrix} 1 & t_s \\ 0 & 1 - t_s/T \end{bmatrix}, B = \begin{bmatrix} 0 \\ t_S \cdot K/T \end{bmatrix}, \tag{8}$$

where the time constant $T = 0.06$, the amplification factor $K = 0.15$ and the sampling period is equal to $t_s = 4$ ms. The states of the system represent the rotor position and the angular speed. The input is the PWM voltage, which is constrained to be between $\pm 100\%$ of its maximum amplitude. The considered MPC controller needs to solve at each sampling time the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{u}}{\text{minimize}} \quad & \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k + x_N^T P x_N \\ \text{subject to:} \quad & \mathrm{x}_{k+1} = A x_k + B u_k \\ & -100 \le u_k \le 100 \end{aligned} \tag{9}$$

where the tuning matrices in the cost function are:

$$Q = \begin{bmatrix} 1.1\times10^4 & 0 \\ 0 & 2.9\times10^1 \end{bmatrix}, R = 2.4\times10^{-1}. \\ P = Q. \tag{10}$$

We consider a prediction horizon of $N = 40$. The same solution is obtained regardless of the hardware platform used. The closed-loop performance of the system is shown in Fig. 4. The sampling time of the controller is the same as the sampling

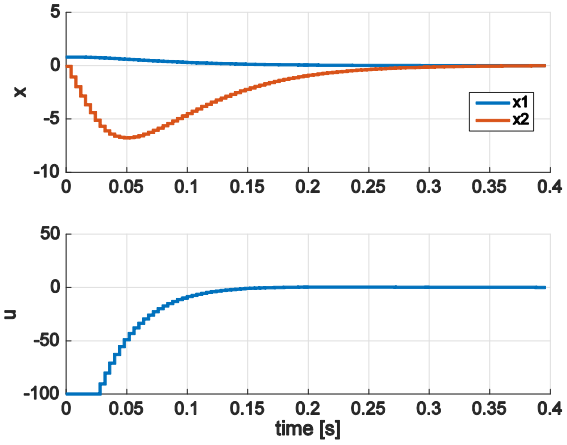


Fig. 4. Time response of the dc-motor system with the implemented control.

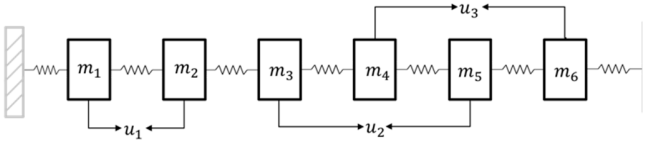


Fig. 5. Chain of oscillating masses with control inputs.

period of the system (4 ms). As it can be seen, the MPC controller drives the system to the desired equilibrium point while respecting the input constraints.

ii) Chain of masses: The second example is a chain of masses that are linked by a spring [32] representing, e.g., an oscillating system with no damping. We consider 6 masses ($m_i = 1$ kg) and spring constants of $k = 1$ N/m (Fig. 5). The system can be represented by 12 dynamic states, the first six states describe the position of the masses and the last 6 its velocities. The forces between the masses are the three control inputs of the system. The control task is to drive the system to the origin, where all masses are at the original position and with no velocity, starting from a disturbed state. The dynamics of the system with a sampling period of $t_s = 0.5$ s can be described by the following discrete-time linear system:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0.5 \\ -1 & 0.5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & -1 & 0.5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & -1 & 0.5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & -1 & 0.5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & -1 & 0.5 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$, B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.5 & 0 & 0 \\ -0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \\ 0 & -0.5 & 0 \\ 0 & 0 & -0.5 \end{bmatrix},$$

We consider constraints on the position and velocity of each mass and on the inputs of the system. The optimization problem to be solved at each sampling time is:

$$\begin{aligned} &\underset{u}{\text{minimize}} \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k + x_N^T P x_N \\ &\text{subject to:} \quad x_{k+1} = A x_k + B u_k \\ &\qquad -4 \le x_k \le 4 \\ &\qquad -0.5 \le u_k \le 0.5 \end{aligned} \tag{12}$$

where the tuning matrices in the cost function are chosen as identity matrices of suitable dimensions, $Q = P = \text{eye}(12)$ and $R = eye(3)$ with a prediction horizon $N = 10$. The MPC algorithm is triggered at each sampling time of the controller, which is chosen to be 10 ms. Fig. 6 shows that the MPC controller is also able to drive the system of oscillating mases to the origin.

B. *FPGA implementation and optimization*

The MPC controllers for both the dc-motor and chain of masses examples have been designed and implemented. This subsection summarizes the main results in the optimal implementation exploration for the floating-point implementations, highlighting the main optimization aspects. The implementations have been made using the VIVADO HLS tool from Xilinx. The target FPGA is a cost-effective XC7A200. Tables IV and V summarize the main implementation results for floating-point implementations, where the microprocessor (µP) implementation, using a standard STM32F407V µP, has been included for comparison. Data are obtained from the actual routed design. All implementations have been made with a target clock period of 10 ns and power consumption is evaluated with a vector-less activity propagation methodology [37], which is an industry-standard probabilistic methodology for power consumption estimation. The proposed implementations use the three optimization methods explained in section III-B, i.e.

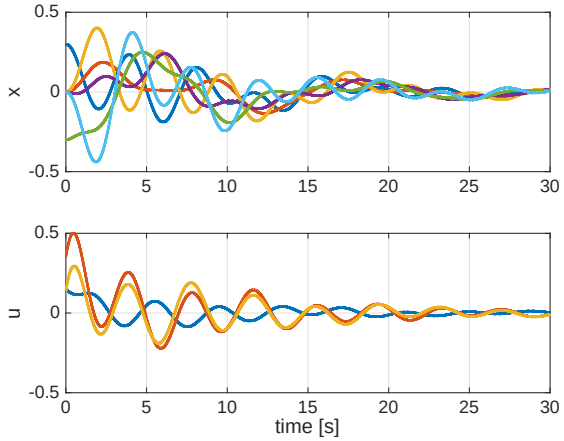


Fig. 6. Time response of the chain of oscillating masses system with the implemented control.

unrolling, pipelining and inlining. Solutions 1, 2 and 3 implement different unrolling strategies at different loop levels, either internal or external. All the solutions implement pipelining, which can be seen in the loop factor or initialization intervals. Finally, all the solutions implement also inlining to optimize cross-function optimization.

Solution 0 is the standard implementation with no optimization, which is a sequential one close to a microprocessor implementation. Solution 1 consists on unrolling the inner loop plus pipelining accordingly using different unrolling factor from 1 (only pipelining) to N (completely unrolled). Solution 2 consists on replicating the hardware to compute simultaneously several rows. For this solution, full unroll and unroll factor 2 have been considered assuming dual port memory since higher unrolling factor would lead to unacceptable memory resources usage. The straightforward full unroll option (2.N on tables) has been included to show that an over-constrained solution leads to a suboptimal result that does not fit into the selected device. Finally, Solution 3 consists on pipelining the outer loop. It is important to note that this optimization cannot be hand-coded in a feasible way and it provides further optimization at the cost of additional resources. Moreover, the design space exploration is mandatory, since the optimum solution cannot be found without performing the actual hardware implementation. Consequently, some implementations achieve high performance with a balanced resources consumption whereas other implementations achieve reduced performance even with unfeasible resource usage.

C. *Discussion*

From the previous results, it is clear that the design space exploration using HLS for MPC problems enables optimization of the FPGA design in terms of performance, digital resources and power consumption. The optimal solution, however, is not trivial. Too aggressive design constraints result in a solution

Table IV. Floating-Point FPGA implementations for the chain of masses MPC

Solution	Optimization	Power (W)	Sampling Frequency (KHz)	BRAMs	DSP48Es	FFs	LUTs
μP	Standard μP implementation	-	1.11	-	-	-	-
0	Standard	0.19	0.95	9	5	1764	1913
1.1	Unroll inner loop factor 1	0.19	9.92	9	5	3261	2893
1.2	Unroll inner loop factor N/5	0.27	22	11	25	7553	6621
1.3	Unroll inner loop factor N/10	0.33	28	18	50	13075	13680
1.4	Unroll inner loop factor N	0.55	34.6	46	148	17166	15864
2	Unroll inner loop factor N + parallel row factor 2	0.96	55.3	96	296	33618	31390
2.N	Full Unroll	-	139	4	2250	180531	190503
3.1	Initiation interval outer loop 200	0.34	48.8	0	58	14046	18932
3.2	Initiation interval outer loop 100	0.51	93.4	0	69	22380	24796
3.3	Initiation interval outer loop 50	1.3	175	0	315	50334	47008

Table V. Floating-Point FPGA implementations for the dc-motor system MPC

Solution	Optimization	Power (W)	Sampling Frequency (KHz)	BRAMs	DSP48Es	FFs	LUTs
μP	Standard μP implementation	-	0.67	-	-	-	-
0	Standard	0.19	0.565	19	5	1314	1842
1.1	Unroll inner loop factor 1	0.21	4.58	19	5	3197	2973
1.2	Unroll inner loop factor N/5	0.34	11.5	41	25	6378	4943
1.3	Unroll inner loop factor N/10	0.37	14.3	33	50	12402	12275
1.4	Unroll inner loop factor N	0.54	17.7	93	198	21269	18631
2	Unroll inner loop factor N + parallel row factor 2	1.1	40.4	181	396	43358	36922
2.N	Full Unroll	-	69.9	9	3232	256931	266127
3.1	Initialization interval outer loop 200	0.29	48.1	9	54	29950	48875
3.2	Initialization interval outer loop 100	0.83	92.9	9	134	39242	64593
3.3	Initialization interval outer loop 80	0.79	114	9	191	48531	75073

with less performance and more usage of FPGA area. Whereas some strategies, such as inner loop unroll directly increases performance and resource usage, other more complex strategies such as outer loop unrolling cannot be easily predicted.

Also, it is important to note that the optimal solution depends highly on the problem and on the digital platform used for implementation. The proposed combination of automatic code generation and the use of HLS tools enables a fast optimization procedure that can be specifically performed for each problem at a reasonable cost. Once the set of possible results has been generated, the designer can choose the most suitable for each specific application.

It is important to note that with the proposed methodology optimized FPGA implementations of MPC for fast systems are now possible, opening the design space to new applications.

V. Conclusions

An optimized, software-supported FPGA implementation of model predictive control has been proposed. In order to take the most of modern FPGA technology and provide control engineers with powerful and simple tools, a design methodology based on high level synthesis has been presented. Special attention has been paid to the code and FPGA optimization, providing several guidelines for designers. Finally, the proposed methodology has been applied to two design examples, proving the feasibility and possibilities of the proposed approach. As a conclusion, the

combination of FPGA technology and high level synthesis is a promising technique for modern MPC controllers with high performance, low cost and energy-effective implementations. Furthermore, it opens the door to strictly verifiable MPC implementations, as for example required in aerospace industries.

Future work will include the consideration of model uncertainty in MPC, as done e.g. in [38]. Additionally, combinations of software plus ad-hoc digital hardware via hard-core or soft-core microprocessors opens new possibilities. In this sense, the so-called software defined system on chip provides valuable tools for control designers.

REFERENCES

[1] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*: Nob Hill Pub., 2009.

[2] J. Rodriguez, P. Cortes, R. Kennel, and M. P. Kazmierkowski, "Model predictive control -- a simple and powerful method to control power converters," in *Power Electronics and Motion Control Conference, 2009. IPEMC '09. IEEE 6th International*, 2009, pp. 41-49.

[3] T. Geyer, G. Papafotiou, and M. Morari, "Model Predictive Direct Torque Control. Part I: Concept, Algorithm, and Analysis," *IEEE Transactions on Industrial Electronics,* vol. 56, pp. 1894-1905, 2009.

[4] C. Xia, M. Wang, Z. Song, and T. Liu, "Robust Model Predictive Current Control of Three-Phase Voltage Source PWM Rectifier With Online Disturbance Observation," *IEEE Transactions on Industrial Informatics,* vol. 8, pp. 459-471, 2012.

[5] H. Guzman, M. J. Duran, F. Barrero, L. Zarri, B. Bogado, I. G. Prieto, and M. R. Arahal, "Comparative Study of Predictive and Resonant Controllers in Fault-Tolerant Five-Phase Induction Motor Drives," *IEEE Transactions on Industrial Electronics,* vol. 63, pp. 606-617, 2016.

[6] F. Wang, S. Li, X. Mei, W. Xie, J. Rodr, x00Ed, guez, and R. M. Kennel, "Model-Based Predictive Direct Control Strategies for Electrical Drives: An Experimental Evaluation of PTC and PCC Methods," *IEEE Transactions on Industrial Informatics,* vol. 11, pp. 671-681, 2015.

[7] Y. Zhang and C. Qu, "Model Predictive Direct Power Control of PWM Rectifiers Under Unbalanced Network Conditions," *IEEE Transactions on Industrial Electronics,* vol. 62, pp. 4011-4022, 2015.

[8] M. Rivera, A. Wilson, C. A. Rojas, J. Rodriguez, J. R. Espinoza, P. W. Wheeler, and L. Empringham, "A Comparative Assessment of Model Predictive Current Control and Space Vector Modulation in a Direct Matrix Converter," *IEEE Transactions on Industrial Electronics,* vol. 60, pp. 578-588, 2013.

[9] F. Xu, H. Chen, X. Gong, and Q. Mei, "Fast Nonlinear Model Predictive Control on FPGA Using Particle Swarm Optimization," *IEEE Transactions on Industrial Electronics,* vol. 63, pp. 310-321, 2016.

[10] E. C. Kerrigan, G. A. Constantinides, A. Suardi, A. Picciau, and B. Khusainov, "Computer architectures to close the loop in real-time optimization," in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 4597-4611.

[11] G. Valencia-Palomo and J. Rossiter, "Efficient suboptimal parametric solutions to predictive control for PLC applications," *Control Engineering Practice,* vol. 19, pp. 732-743, 2011.

[12] S. Richter, C. N. Jones, and M. Morari, "Computational complexity certification for real-time MPC with input constraints based on the fast gradient method," *Automatic Control, IEEE Transactions on,* vol. 57, pp. 1391-1403, 2012.

[13] P. Zometa, M. Kögel, T. Faulwasser, and R. Findeisen, "Implementation aspects of model predictive control for embedded systems," in *Proc. American Control Conf.*, 2012, pp. 1205-1210.

[14] J. Sawma, F. Khatounian, E. Monmasson, L. Idkhajine, and R. Ghosn, "Cascaded Dual Model Predictive Control of an Active Front-End Rectifier," *IEEE Transactions on Industrial Electronics,* vol. PP, pp. 1-1, 2016.

[15] L. Gomes, E. Monmasson, M. Cirstea, and J. J. Rodriguez-Andina, "Industrial electronic control: FPGAs and embedded systems solutions," in *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, 2013, pp. 60-65.

[16] K.-V. Ling, B. F. F. Wu, and J. M. Maciejowski, "Embedded model predictive control (MPC) using a FPGA," in *Proc. IFAC World Congress*, 2008, pp. 15250-15255.

[17] L. G. Bleris, J. Garcia, M. V. Kothare, and M. G. Arnold, "Towards embedded model predictive control for system-on-a-chip applications," *Journal of Process Control,* vol. 16, pp. 255-264, 2006.

[18] M. Ricco, P. Manganiello, E. Monmasson, G. Petrone, and G. Spagnuolo, "FPGA-Based Implementation of Dual Kalman Filter for PV MPPT Applications," *IEEE Transactions on Industrial Informatics,* vol. PP, pp. 1-1, 2015.

[19] I. Bahri, L. Idkhajine, E. Monmasson, and M. E. A. Benkhelifa, "Hardware/Software Codesign Guidelines for System on Chip FPGA-Based Sensorless AC Drive Applications," *IEEE Transactions on Industrial Informatics,* vol. 9, pp. 2165-2176, 2013.

[20] E. Monmasson and M. N. Cirstea, "FPGA design methodology for industrial control systems: a review," *IEEE Trans. Ind. Electron.,* vol. 54, pp. 1824-1842, August 2007.

[21] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica,* vol. 36, pp. 789-814, 6// 2000.

[22] G. Martin and G. Smith, "High-level synthesis: Past, present, and future," *IEEE Design & Test of Computers,* vol. 26, pp. 18-25, 2009.

[23] J. Cong, L. Bin, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhiru, "High-level synthesis for FPGAs: From prototyping to deployment," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 30, pp. 473-491, 2011.

[24] D. Navarro, O. Lucía, L. A. Barragán, I. Urriza, and O. Jiménez, "High-level synthesis for accelerating the FPGA implementation of computationally-demanding control algorithms for power converters," *IEEE Trans. Ind. Informat.,* vol. 9, pp. 1371-1379, August 2013.

[25] O. Jimenez, O. Lucia, I. Urriza Parroque, L. A. Barragan, D. Navarro, and V. Dinavahi, "Implementation of an FPGA-based on-line hardware-in-the-loop emulator using high-level synthesis tools for resonant power converters applied to induction heating appliances," *IEEE Transactions on Industrial Electronics,* vol. 62, pp. 2206-2214, April 2015.

[26] A. Suardi, E. C. Kerrigan, G. A. Constantinides, and R. Findeisen, "Fast FPGA prototyping toolbox for embedded optimization.," in *Proc. of the European Control Conference*, 2015, pp. 2589-2594.

[27] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optimization and Engineering,* vol. 13, pp. 1-27, 2012.

[28] E. Chu, N. Parikh, A. Domahidi, and S. Boyd, "Code generation for embedded second-order cone programming," in *Proc. European Control Conf.*, 2013, pp. 1547-1552.

[29] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded online optimization for model predictive control at megahertz rates," *IEEE Transactions on Automatic Control,* pp. 3238-3251, 2014.

[30] S. Boyd and L. Vandenberghe, *Convex Optimization*: Cambridge University Press, 2004.

[31] M. Kögel and R. Findeisen, "Fast predictive control of linear, time-invariant systems using an algorithm based on the fast gradient method and augmented Lagrange multipliers," in *Proc. IEEE Conf. Control Applications*, 2011, pp. 780-785.

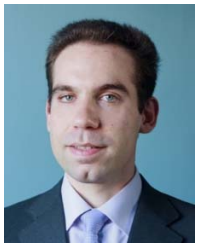
[32] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *Control Systems Technology, IEEE Transactions on,* vol. 18, pp. 267-278, 2010.

[33] P. Zometa, H. Heinemann, S. Lucia, M. Kögel, and R. Findeisen, "Efficient Implementation of Stochastic Model Predictive Control for Embedded Systems Based on Second-Order Cone Programs," in *Submitted to the European Control Conf.*, 2016.

[34] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*: Kluwer Academic Publishers, 2004.
[35] P. Zometa, M. Kögel, and R. Findeisen, "muAO-MPC: A free code generation tool for embedded real-time linear model predictive control," in *Proc. American Control Conf.*, 2013, pp. 5320-5325.
[36] Xilinx, *Introduction to high-level synthesis with Vivado HLS Standalone*, 2016.
[37] F. N. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 2, pp. 446-455, 1994.
[38] S. Lucia, T. Finkler, and S. Engell, "Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty," *Journal of Process Control,* vol. 23, pp. 1306-1319, 10// 2013.

Sergio Lucia (M'16) obtained his M.Sc. in Electrical Engineering from the University of Zaragoza in 2010 and the Dr.-Ing. degree from TU Dortmund in the field of optimization and automatic control in 2014. He then joined the Otto-von-Guericke Universität Magdeburg, and visited the Massachusetts Institute of Technology as a Postdoctoral Fellow.
Since May 2017, he is Assistant Professor and holds the chair "Internet of Things for Smart Buildings" at the TU Berlin and the Einstein Center Digital Future. His research efforts focus on decision-making under uncertainty, distributed control, and embedded optimization using micro-controllers and FPGAs in the framework of the Internet of Things. Applications of interest include smart buildings and Li-ion battery systems.

Denis Navarro received the M.Sc. degree in Microelectronics from the University of Montpellier, France, and the Ph.D. degree from the University of Zaragoza in 1987 and 1992, respectively.
Since September 1988, he has been with the Department of Electronic Engineering and Communications at the University of Zaragoza, where he is an Associate Professor. His current research interests include CAD for VLSI, low power ASIC design, and modulation techniques for power converters. He is involved in the implementation of new applications of integrated circuits. In 1993 Dr. Navarro designed the first SPARC® microprocessor in Europe.
Dr. Navarro is a member of the Aragon Institute for Engineering Research (I3A).



Óscar Lucía (S'04, M'11, SM'14) received the M.Sc. and Ph.D. degrees (with honors) in Electrical Engineering from the University of Zaragoza, Spain, in 2006 and 2010, respectively.
During 2006 and 2007 he held a research internship at the Bosch and Siemens Home Appliances Group. Since 2008, he has been with the Department of Electronic Engineering and Communications at the University of Zaragoza, Spain, where he is currently an Associate Professor. During part of 2009 and 2012, he was a visiting scholar at the Center of Power Electronics Systems (CPES), Virginia Tech. His main research interests include resonant power conversion, wide-bandgap devices, and digital control, mainly applied to contactless energy transfer, induction heating, electric vehicles, and biomedical applications. In these topics, he has published more than 50 international journal papers and 125 conference papers, and he has filed more than 25 patents.
Dr. Lucía is a Senior Member of the IEEE and an active member of the Power Electronics (PELS) and Industrial Electronics (IES) societies. He was a Guest Associate Editor of the IEEE Transactions on Industrial Electronics and the IEEE Journal of Emerging and Selected Topics in Power Electronics in 2013 and 2015, respectively. Currently, he is an Associate Editor of the IEEE Transactions on Industrial Electronics and IEEE Transactions on Power Electronics. Dr. Lucía is a member of the Aragon Institute for Engineering Research (I3A).

Pablo Zometa received the M.Sc. in mechatronic systems from the University of Siegen, Germany in 2007. He received his Ph.D. from the Otto-von-Guericke University Magdeburg, Germany in 2017. The topic of his dissertation was automatic code generation for model predictive control of embedded systems. Currently he leads the technical development at his own startup.

Rolf Findeisen is full professor at the Department of Electrical Engineering and Information Technology at the Otto-von-Guericke-Universität Magdeburg. He studied Engineering Cybernetics (Diploma '97) at the University of Stuttgart, chemical engineering (M.Sc. '97) at the University of Wisconsin-Madison, and received his Ph.D. from the University of Stuttgart ('05). Rolf is associated editor of the IEEE Transaction on Control of Networked System, editor of the IEEE Control Systems Magazin, and heads the workgroup on Internet of Things of the Network and Communication Systems Technical Commity of the IEEE Control Systems Society. He has been organiser and chair of several conferences and is the Co-chair of the International Program Committee of the IFAC World Congress 2020 in Berlin. Rolfs main research interests are optimisation based / predictive control and estimation for uncertain systems, cyberphysical systems and the interplay of control and IoT. Applications span from mechatronic systems, batteries, electrical systems to biomedicine and systems biology.