25th ABCM International Congress of Mechanical Engineering
October 20-25, 2019, Uberlândia, MG, Brazil

COB-2019-1003
Hardware Implementation of Linear Unconstrained Model Predictive Control Using FPGA

Alceu Bernardes Castanheira de Farias
André Murilo
Renato Vilela Lopes

University of Brasilia, Engineering Faculty campus Gama, DF, Brazil
alceu.castanheira@gmail.com, andremurilo@unb.br, rvlopes@unb.br

Abstract. *Due to the impact that Model Predictive Control (MPC) has caused in both industry and academia, mainly motivated by its capability of handling the constraints of a control problem in a systematic way, many studies have been held recently in order to improve and accelerate MPC controllers, making it possible to apply them to systems with faster dynamics and higher sampling periods. One of these solutions, proposed in the literature of embedded control systems, corresponds to the usage of Field Programmable Gate Arrays (FPGAs), logic devices that contain reconfigurable hardware and are capable of accelerating algorithms by implementing them in the most parallel way possible. Regarding these informations, this work proposes the implementation of a linear unconstrained MPC controller embedded in FPGA, the first step needed for the implementation of the linear constrained MPC controller, using manually developed (RTL implementation) algorithms in VHDL and floating-point arithmetic units, and thus approaching a different methodology for developing such controllers, that usually are implemented using automatic generation tools (High Level Synthesis or HLS) and fixed-point arithmetic modules.*

Keywords: *Control Systems, Embedded Systems, Model Predictive Control, FPGAs*

1. INTRODUCTION

One of the most impactful research areas in both industry and academia through the last years, contribuiting in a fundamental manner with many fields of engineering, corresponds to the field of control systems (Frank, 2012) and when it comes to control strategies, one that has gained notoriety is a type of controller known as Model Predictive Control (MPC).

The main reason for such interest in MPC is its capability of controlling a system while handling constraints imposed to it, regardless of their nature (security, operation range of the actuators, quality norms of a product, etc.), while still finding an optimal solution that satisfies the desired control objectives at each sampling period of the system along a finite number of steps, a very important parameter for MPC controllers named prediction horizon (Ekaputri and Syaichu-Rohman, 2013).

The ability of handling constraints is one of the most important aspects of MPC, but not the only one: it is also capable of handling multiple-input/multiple-output systems and it can be adapted to non linear systems, which is referred to as Non linear Model Predictive Control or NMPC (Mazen, 2013). In fact, all these characteristics make MPC the control strategy of choice for many different applications. According to a survey performed by (Domahidi *et al.*, 2016), automotive, energy, robotics, aerospace, power electronics, chemical, oil, gas, and many others are among the industrial fields that opt to use MPC as an embedded controller, which can be seen in further details in Fig. 1.

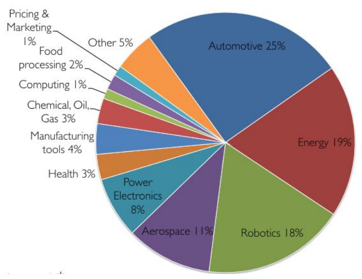

Figure 1. Embedded MPC application areas. Available from: (Domahidi *et al.*, 2016).

However, despite all the advantages presented, MPC has a major drawback: the elevated computational cost associated with its implementation. The controller must solve at each sampling period a constrained optimization problem, which is a task that increases in difficulty as the system being controlled and its dynamics increase in complexity (Wills *et al.*, 2012). This is the main reason why MPC was first applied to systems with slower dynamics and lower sampling periods, such as the ones in the chemical industry. Therefore, there is a great number of applications that could benefit from MPC, but due to this disadvantage, such approach becomes impracticable. For this reason, many studies have been developed in the last years to provide more efficient solutions for the implementation of MPC controllers, making it possible to handle its elevated computational cost and apply it to systems with faster dynamics (Dua *et al.*, 2008).

One of these solutions that has been proposed in the literature corresponds to the usage of Field Programmable Gate Arrays or FPGAs. These are programmable logic devices, composed by a technology that allows the development of embedded systems based on parallel architectures, offering solutions with high processing power that are implemented directly in hardware (Muñoz, 2012). Modern FPGAs have dedicated resources for multiple operations, allowing different operations of MPC to be performed in parallel with high frequency (Abughalieh and Alawneh, 2019), which helps mitigating the previously discussed problem related to the computational cost associated to the implementation of this type of controller in hardware platforms.

As a result of these characteristics, FPGAs have been widely used through the last years in order to implement MPC controllers for many different types of applications, such as Cessena Citation 500 (Luo *et al.*, 2011) and Boeing 747 200 (Hartley *et al.*, 2014) aircraft models, power converters (Navarro *et al.*, 2013), single link robotic manipulator (Ayala *et al.*, 2016), engine idle speed control (Xu *et al.*, 2016), lateral stability control of active chassis for intelligent vehicles (Guo *et al.*, 2019), and many others.

When analyzing most contributions to the literature regarding this topic, it is possible to see that most applications use some sort of automatic code generation tool in order to convert an algorithm available in a high-level programming language, such as C or MATLAB, to an algorithm in hardware description language, such as VHDL or Verilog, that allows the MPC controller to be embedded in a FPGA. This process is usually known as High Level Synthesis (HLS) and is considered to decrease time-to-market, allowing designers to benefit from the speed provided by the hardware solutions embedded in FPGAs while developing systems in a faster way, as opposed to manually coding the architecture of such systems using hardware description language to describe their behaviour at register transfer level (also known as RTL) logic, which demands advanced hardware expertise by the designer (Nane *et al.*, 2016). However, even the newest generation of HLS tools does not provide as good performance and resource usage as manual RTL does (Lahti *et al.*, 2019).

Also, most of these applications utilize fixed-point arithmetic units in order to perform the necessary calculations for MPC, which helps reducing the FPGA hardware consumption, but limits the range of numeric values that can be represented within the developed architecture, which also limits the applicability of the developed solution for different types of systems being controlled. Modern FPGAs, with more hardware and processing capability, allow the development of applications using floating-point arithmetic units, that consume more hardware resources for their implementation, but allow a greater range of numeric values to be represented, resulting in architectures less dependant to a specific situation (Urriza *et al.*, 2010), applicable to different types of systems.

Regarding this scenario, this work proposes the hardware implementation of a linear unconstrained MPC using FPGA, which is the first step necessary for achieving a full hardware implementation of a linear constrained MPC controller, using both a RTL implementation (which means no HLS tool is used for developing the controller) and floating-point arithmetic units. It may seem a bit contradictory to implement an unconstrained version of MPC after highlighting all the advantages regarding the controller ability to handle constraints, but it is important to start with the unconstrained scenario, since it creates a simpler control problem that is ideal for testing the hardware modules developed, that will be used for the implementation of the linear MPC controller in the future.

This methodology can be considered an important contribution for the literature of embedded MPC controllers in FPGA, as most works present fixed-point architectures developed using HLS.

The kit used for the implementation of the unconstrained MPC controller is a KC-705 Evaluation Kit by Xilinx and in order to simulate and validate the developed architecture, HDL Verifier (a MATLAB tool capable of simulating embedded applications in FPGA in conjunction with Simulink) is used. The application chosen for validating the developed design is a linearized Boeing 747 aircraft model, as aerospace systems correspond to an important field of utilization of embedded MPC, as discussed previously in this section.

This work is divided as it follows: Section 2 describes the mathematical modelling of both the unconstrained MPC controller and the Boeing 747 aircraft model, Section 3 presents the methodology used in this work and how the controller was implemented manually using floating-point arithmetic units in FPGA, as well as how tests were setup in order to validate the developed architecture. Section 4 presents results and discussions regarding the experiments performed and finally, Section 5 concludes this work and also presents future work propositions.

2. MATHEMATICAL MODELLING OF THE CONTROL SYSTEM AND THE CONTROLLED SYSTEM

2.1 Mathematical modelling of the unconstrained MPC controller

MPC is a control strategy in which the current control actions are obtained through the online solution of an open-loop optimal control problem within a prediction horizon (that shall be refered as N from now on) at each sampling period, using the current state of the plant/controlled system as initial state. This optimization problem produces an optimal control sequence, from which only the first element is applied to the controlled system (Mayne *et al.*, 2000).

Therefore, MPC also requires the definition of a cost function that expresses the control objective, i.e., the most relevant variables to the control problem, that are minimized in order to achieve the optimal results described in the previous paragraph.

The steps needed for obtaining the optimal control sequences using MPC are given by (Mazen, 2013):

1. Definition of the cost function that expresses the control objective
2. At each sampling period, measure the plant/controlled system current states
3. Compute the sequence of future actions using the defined cost function
4. Apply only the first element of the optimal control sequence obtained in the previous step
5. Repeat steps 2, 3 and 4 at each new sampling period

One interesting characteristic of MPC is that only the first element of the optimal control sequence calculated is applied to the control system, which is important to help maintaining a certain level of robustness, since at each new sampling period the controlled system might be affected by disturbances that were not taken into consideration when performing calculations for the previous optimal control sequence.

In general, MPC aims to solve a quadratic programming optimization problem, commonly known as QP, that minimizes a cost function that represents the controller objectives at each sampling period. If the control problem is a linear one, which is the case of the problem presented in this work, only one QP is solved at each sampling period (Ferreau *et al.*, 2017). Solutions for QPs are well established in the literature and can be used for solving the optimal control problem within MPC.

All the mathematical modelling and equations regarding the unconstrained MPC controller are based in (Mazen, 2013). For further details about this topic, the reader is invited to check this work.

The first step in order to obtain the mathematical modelling of the controller is to represent the system being controlled (the Boeing 747 aircraft model that will be explained in the next Subsection of this work) in state-space form, allowing us to express the systems dynamics in a matrix-vectorial equation format, as it can be seen in Eq. (1).

$$\left\{ \begin{array}{lcl} \mathbf{x}(k+1) & = & \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \\ \mathbf{y_r}(k) & = & \mathbf{C_r}x(k) \end{array} \right. . \tag{1}$$

In Eq. (1), $\mathbf{x}$, $\mathbf{u}$ and $\mathbf{y_r}$ indicate the state, control and regulated outputs vectors respectively, while $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C_r}$ correspond to the state, inputs and regulated output matrices. The index k represents the current sampling period and, similarly, $k+1$ represents the next sampling period.

When the controlled system is linear time invariant (LTI) in state-space form, as it is the case of many applications including the one presented in this work, the solution of the previously mentioned QP that is intrinsic to the optimization of the cost function related to linear MPC is given by Eq. (2):

$$\frac{1}{2}\tilde{\mathbf{u}}^T\mathbf{H}\tilde{\mathbf{u}} + \mathbf{F}^T\tilde{\mathbf{u}} + \text{constant}. \tag{2}$$

The vector $\tilde{\mathbf{u}}$ represents the possible optimal control sequence that minimizes the cost function associated with the QP, while matrices $\mathbf{H}$ and $\mathbf{F}$ represented in Eq. (2) are associated to the quadratic cost function of the control problem and each one depends on others matrices, defined in (Mazen, 2013), including not only the state-space matrices that describe the model to be controlled, but also weighting matrices. Equation (3) shows how both $\mathbf{H}$ and $\mathbf{F}$ are calculated.

$$\left\{ \begin{array}{lcl} \mathbf{H} & = & 2\sum_{i=1}^{N}\left[\mathbf{\Psi_i}^T\mathbf{C_r}^T\mathbf{Q_y}\mathbf{C_r}\mathbf{\Psi_i} + \left(\mathbf{\Pi_i^{(n_u,N)}}\right)^T\mathbf{Q_u}\left(\mathbf{\Pi_i^{(n_u,N)}}\right)\right] \\ \mathbf{F} & = & \mathbf{F_1}\mathbf{x}(k) + \mathbf{F_2}\tilde{\mathbf{y}}_\mathbf{r}^\mathbf{d}(k) + \mathbf{F_3}\mathbf{u^d} \\ \mathbf{F_1} & = & 2\sum_{i=1}^{N}\left[\mathbf{\Psi_i}^T\mathbf{C_r}^T\mathbf{Q_y}\mathbf{C_r}\mathbf{\Phi_i}\right] \\ \mathbf{F_2} & = & -2\sum_{i=1}^{N}\left[\mathbf{\Psi_i}^T\mathbf{C_r}^T\mathbf{Q_y}\mathbf{\Pi_i^{(n_r,N)}}\right] \\ \mathbf{F_3} & = & 2\sum_{i=1}^{N}\left[\left(\mathbf{\Pi_i^{(n_u,N)}}\right)^T\mathbf{Q_u}\right] \end{array} \right. . \tag{3}$$

Equation (3) presents many different matrices and vectors that must be defined. $\mathbf{Q_y}$ and $\mathbf{Q_u}$ are weighing matrices for the outputs and control variables of the system, respectively. These are parameters that are defined by the designer in order to tune the controller and achieve better responses. Constant matrices $\mathbf{\Psi_i}$ and $\mathbf{\Phi_i}$ come from the prediction scheme used by MPC and depend only on system matrices $\mathbf{A}$ and $\mathbf{B}$. Matrix $\mathbf{\Pi_i}$ corresponds to a selection matrix, that from a total of N elements select only the first n_u(number of control variables) or n_r (number of regulated outputs) from the matrix that it multiplies. Vectors $\tilde{\mathbf{y}}_\mathbf{r}^\mathbf{d}$ and $\mathbf{u^d}$ represent the desired trajectory values for the regulated outputs, i.e., the reference values for the regulated outputs of the system in steady-state, and the desired values for the control variables in steady-state, respectively.

One important detail to take in consideration is that matrices $\mathbf{F_1}$, $\mathbf{F_2}$, $\mathbf{F_3}$ and $\mathbf{H}$ are constant, which means these matrices can be computed offline. This is important because it means that these matrices do not need to be calculated at each sampling period of the system: they can simply be stored in memory so that they are used for the calculation of $\mathbf{F}$. In fact, only matrix $\mathbf{F}$ must be computed online, i.e., must be calculated at each sampling period, because it depends on the value of $\mathbf{x}(k)$ and $\tilde{\mathbf{y}}_\mathbf{r}^\mathbf{d}(k)$, which are updated at each new sampling period.

Since the control formulation used in this work corresponds to the unconstrained scenario of MPC, these are the only matrices needed for expressing the desired control law. When constraints are applied to the system, new matrices are introduced to the modelling of the controller in order to incorporate them into the control problem formulation.

The desired control law for unconstrained MPC can be expressed in the format presented by Eq. (4), in which the control variables $\mathbf{u}$ at a current sampling period k can be calculated by a negative feedback gain $\mathbf{K_N}$ multiplied by the states vector x at that same sampling period.

$$\mathbf{u}(k) = -\mathbf{K_N}\mathbf{x}(k). \tag{4}$$

The optimal sequence of future control actions $\tilde{\mathbf{u}}^{opt}(\mathbf{x}(k))$ is expressed by vector $\tilde{\mathbf{u}}$, that minimizes the cost function of the control problem given by Eq. (5), which is obtained by the substitution of $\mathbf{F}$ in Eq. (2) by the terms necessary for its calculation that were presented in Eq. (3).

$$\frac{1}{2}\tilde{\mathbf{u}}^T\mathbf{H}\tilde{\mathbf{u}} + [\mathbf{F_1}\mathbf{x}(k) + \mathbf{F_2}\tilde{\mathbf{y}}_\mathbf{r}^\mathbf{d}(k) + \mathbf{F_3}\mathbf{u^d}]^T\tilde{\mathbf{u}} + \text{constant}. \tag{5}$$

In the unscontrained scenario, the optimal control sequence $\tilde{\mathbf{u}}^{opt}(\mathbf{x}(k))$ can be obtained by making the gradient of the cost function defined in Eq. (5) equal to zero, resulting in Eq. (6):

$$\mathbf{H}\tilde{\mathbf{u}} + \mathbf{F_1}\mathbf{x}(k) + \mathbf{F_2}\tilde{\mathbf{y}}_\mathbf{r}^\mathbf{d}(k) + \mathbf{F_3}\mathbf{u^d} = 0. \tag{6}$$

Therefore, the optimal control sequence can be defined by the resulting Eq. (7):

$$\tilde{\mathbf{u}}^\mathbf{opt}(\mathbf{x}(k)) = -\mathbf{H}^{-1}[\mathbf{F_1}\mathbf{x}(k) + \mathbf{F_2}\tilde{\mathbf{y}}_\mathbf{r}^\mathbf{d}(k) + \mathbf{F_3}\mathbf{u^d}]. \tag{7}$$

As MPC utilizes only the first component of the optimal control sequence calculated previously, the control vector at a given sampling period for unconstrained MPC is given by Eq. (10):

$$\mathbf{u}(k) = \mathbf{K_{MPC}}(\mathbf{x}(k)) = -\mathbf{\Pi_1^{(n_u,N)}}\mathbf{H}^{-1}[\mathbf{F_1}\mathbf{x}(k) + \mathbf{F_2}\tilde{\mathbf{y}}_\mathbf{r}^\mathbf{d}(k) + \mathbf{F_3}\mathbf{u^d}], \tag{8}$$

$$\mathbf{u}(k) = -\underbrace{\mathbf{\Pi_1^{(n_u,N)}}\mathbf{H}^{-1}\mathbf{F_1}}_{\mathbf{K_N}}\mathbf{x}(k)\underbrace{-\mathbf{\Pi_1^{(n_u,N)}}\mathbf{H}^{-1}\mathbf{F_2}}_{\mathbf{G_N}}\tilde{\mathbf{y}}_\mathbf{r}^\mathbf{d}(k)\underbrace{-\mathbf{\Pi_1^{(n_u,N)}}\mathbf{H}^{-1}\mathbf{F_3}}_{\mathbf{L_N}}\mathbf{u^d}, \tag{9}$$

$$\mathbf{u}(k) = -\mathbf{K_N}\mathbf{x}(k) + \mathbf{G_N}\tilde{\mathbf{y}}_\mathbf{r}^\mathbf{d}(k) + \mathbf{L_N}\mathbf{u^d}. \tag{10}$$

Since $\mathbf{F_1}$, $\mathbf{F_2}$ and $\mathbf{F_3}$ are computed offline, so are the matrices $\mathbf{K_N}$, $\mathbf{G_N}$ and $\mathbf{L_N}$ introduced in Eq. (10). Thus, these matrices can be computed previously once the application and the control parameters are defined and stored in memory so that they can be used at the proper time for the calculation of the unconstrained MPC control law.

Therefore, Eq. (10) corresponds to the mathematical equation that is implemented in RTL using floating-point arithmetic units. Further details regarding the implementation of the controller in VHDL are given in Section 3.

2.2 Mathematical modelling of the Boeing 747 aircraft

Just like the mathematical modelling of the unconstrained MPC, the controlled system was selected from one of the case studies that is presented in (Mazen, 2013): the Boeing 747 aircraft model. Aerospace applications using MPC controllers embedded in FPGAs are fairly common in the literature, as stated in Section 1, and due to the relative simplicity of this linear model, it is considered a good choice for the first tests to be performed with the developed controller architecture.

The Boeing 747 model is represented in state-space form, as explained in section 2, and the continuous matrices that represent it in such format are given in (Mazen, 2013). However, in order to implement the controller in hardware, it is necessary to use discrete matrices, that can be obtained using the continuous matrices presented in (Mazen, 2013) with the MATLAB routine *c2d*. The sampling time adopted corresponds to $\tau = 200$ ms, and the discrete matrices $\mathbf{A_d}$ and $\mathbf{B_d}$ that represent the Boeing 747 model in state-space are given by Eq. (11) and Eq. (12).

$$\mathbf{A_d} = \begin{bmatrix} 0.97720 & -0.19420 & 0.01650 & 0.00820 \\ 0.11880 & 0.96530 & -0.00500 & 0.00050 \\ -0.57140 & 0.13090 & 0.90610 & -0.00240 \\ -0.05750 & 0.02720 & 0.19060 & 0.99980 \end{bmatrix}, \quad \mathbf{B_d} = \begin{bmatrix} 0.10780 & 0.00830 \\ -0.93460 & 0.14480 \\ -0.02910 & 2.73500 \\ -0.00820 & 0.27880 \end{bmatrix} \tag{11}$$

$$\mathbf{C_r} = \begin{bmatrix} 0.00000 & 1.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & 0.00000 & 1.00000 \end{bmatrix} \tag{12}$$

From the matrices presented in Eq. (11) and Eq. (12), it is possible to notice that the system has four states, two control variables and two regulated outputs, that correspond to the roll and tilt angles with respect to the aircraft reference frame. These are the matrices that describe the system dynamics and that used for obtaining matrices $\mathbf{K_N}$, $\mathbf{G_N}$ and $\mathbf{L_N}$ for the unconstrained MPC control law, as stated in the previous subsection.

3. METHODOLOGY

The main methodology for this work is to perform computational simulations for the unconstrained MPC controller using MATLAB and then comparing the response of the unconstrained MPC embedded in FPGA to the results obtained from the MATLAB simulations.

At this work, MATLAB 2016a is used for computational simulations of the unconstrained MPC applied to the control of the Boeing 747 system, using two different parameters of prediction horizon: N = 20 and 30.

In addition to the matrices A_d and B_d presented in Subsection 2.2, the weighing matrices used for both the computational simulations and hardware implementation of the unconstrained MPC controller, $\mathbf{Q_y}$ and $\mathbf{Q_u}$, are given by Eq. (13).

$$\mathbf{Q_y} = \begin{bmatrix} 1000 & 0 \\ 0 & 100 \end{bmatrix}, \quad \mathbf{Q_u} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{13}$$

The reference values for each output (y_1 and y_2) correspond to -0.4 and 0.5 respectively (until 5s of simulation), 0 and 0.8 (until 15s of simulation), 0.3 and 0.25 (until 25s of simulation) and finally, 0 and 0 (for the remaining of the simulation).

The same algorithms were implemented in FPGA using VHDL as the hardware description language of choice. Most algebraic operations for the MPC controller (for both the unconstrained and constrained cases) involve matrix multiplication and matrix addition, thus developing hardware modules for these operations in VHDL is mandatory. This is where using the unconstrained scenario of MPC becomes important as a first step of validation: since there are less matrices involved in the unconstrained MPC control problem formulation (as well as matrices with lower dimensions) than the constrained MPC, testing the matrix multiplication and matrix addition modules becomes easier. Once these modules are functioning properly in the unconstrained scenario, they are ready to be used for the implementation of linear constrained MPC in future applications.

The matrix multiplication and matrix addition modules were developed using floating-point arithmetic units, that might consume more hardware resources for its implementation, but it will allow a greater range of applications to be tested with the developed architecture, as stated in Section 1. The basis for developing each of these modules are the floating-point multipliers and floating-point adders presented in (Muñoz, 2012), originally developed in VHDL, which is why this was the selected hardware description language for this work. These floating-point arithmetic units have dynamic range, which means that the designer selects the desired data width as a parameter of the project. Data width used in this work corresponds to 27 bits, which is stated in (Ayala *et al.*, 2016) to be an optimal value for hardware resource usage when working with these floating-point modules.

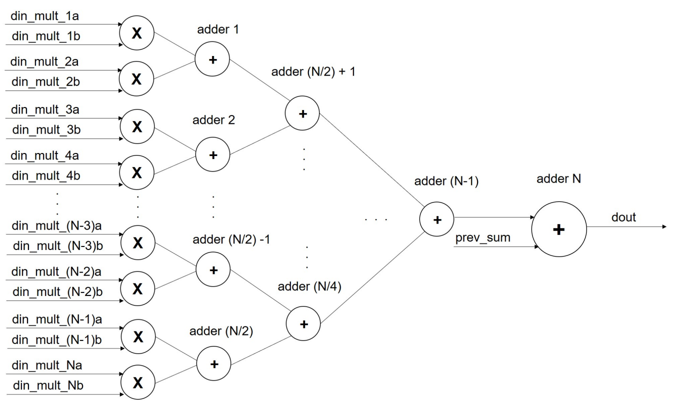

Figure 2. Architecture of the developed floating-point matrix multiplication module for the unconstrained MPC controller implemented in VHDL.

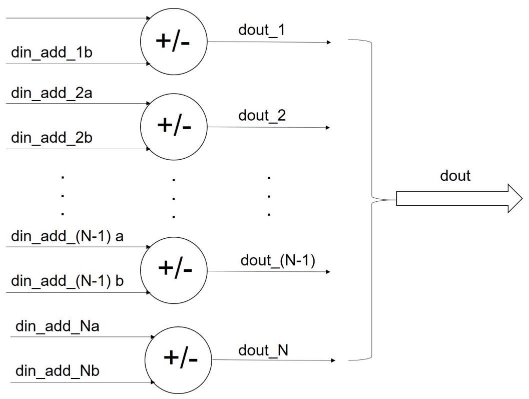

Figure 3. Architecture of the developed floating-point matrix addition module for the unconstrained MPC controller implemented in VHDL.

The architecture for each of the matrix multiplication and matrix addition modules can be seem in Fig. 2 and Fig. 3 respectively. Both these modules implement floating-point multipliers and/or adders in parallel in order to perform their respective algebraic operations in a faster way, as multiple elements from matrices being multiplied or added can be processed at the same time.

The matrix multiplication module has N multipliers and N adders in parallel for performing accumulation operations needed for matrix multiplication, where N corresponds to the prediction horizon adopted for the MPC controller, while the matrix adder has N adders in parallel. That means that N elements from two matrices being added or multiplied can be processed in parallel, at the same time, allowing the unconstrained MPC control law to be calculated faster. Since all matrices involved in the MPC control problem formulation (both for the unconstrained and constrained scenarios) have dimensions that are multiple of N, this parameter was adopted as the number of parallel elements in each of these modules.

The accumulation process in the matrix multiplication module is used to obtain one element of the resulting matrix, which is why it has only one output. For the matrix addition module, since N elements being added from two different matrices result in N final elements of the resulting matrix, the output of each adder is grouped into a vector containing all new elements. This is an useful feature for saving multiple information in memory blocks using only one write cycle, which is not necessary for the unconstrained scenario, but it is very important for the future constrained MPC applications.

All the logic of these modules was developed manually, but their inputs, outputs, signals and the number of floating-point point adders and multipliers in parallel are parameterized using MATLAB scripts, according to the system being controlled and the prediction horizon adopted. This does not mean that the VHDL modules developed are automatically generated in the same way that HLS tools work: only parameterization is automatic, the logic of each module was manually developed and remains unchanged, making this a RTL implementation nonetheless. The usage of such scripts allows the testing of different parameters (such as floating-point data width or different weighting matrices) in a faster way, as well as allowing new applications to be tested in the future.

Figure 4 presents a block diagram with the main modules contained in the linear unconstrained MPC controller top module implemented in hardware.

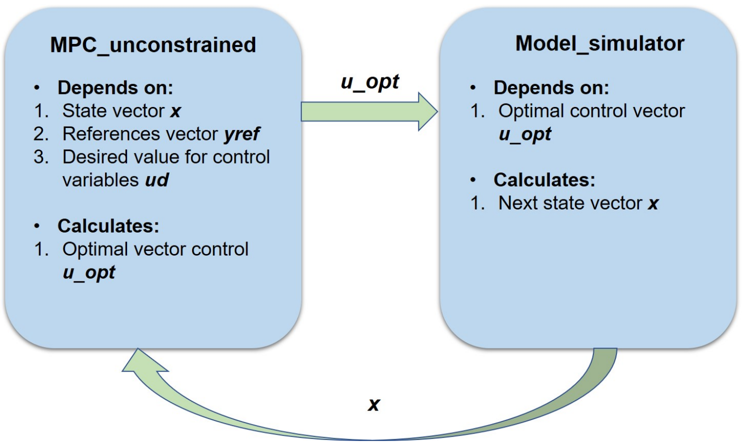

Figure 4. Block diagram presenting the main modules used for the implementation of the linear unconstrained MPC controller top module in hardware.

First, there is a Finite State Machine (FSM) named unconstrained_ MPC, which controls the calculation steps needed for implementing the unconstrained MPC control law, using the values of states vector ($\mathbf{x}$), references vector $\mathbf{y_{ref}}$ and the desired value of the control variables ($\mathbf{u_d}$) for obtaining the optimal control vector $\mathbf{u_{opt}}$ expressed in Eq. (10), as stated in Subsection 2.1. Using this optimal control vector, the model_ simulator FSM implements the controlled system equations that describe its behaviour using a state-space formulation, expressed in Eq. (1), calculating the next state vector that will be used by the controller for calculating the next optimal control vector at the next sampling period.

It is important to notice that the model_ simulator is not actually part of the controller itself, but is necessary for simulating and validating the controller. Also, both these modules are still present in the constrained scenario of MPC, proving once again that the testing of the unconstrained scenario is an important step of validation before implementing the constrained MPC controller.

As stated in the previous section, in order to validate MPC control strategies embedded in FPGA, HDL Verifier was used. This tool from MATLAB allows the validation of hardware description based modules using a technique known as FPGA-in-the-Loop (FIL), which is shown in Fig. 5.

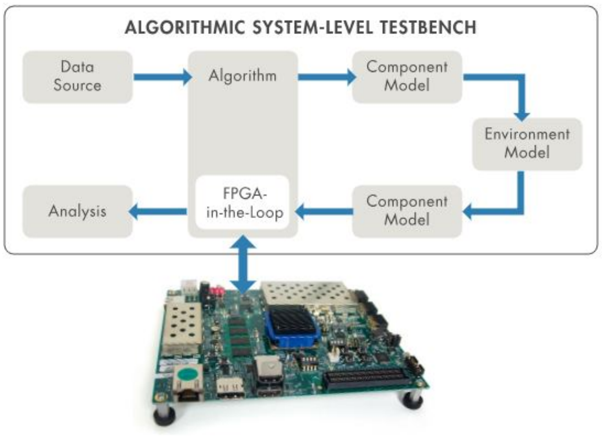

Figure 5. Performing FPGA-in-the-Loop (FIL) verification with FPGA boards. Available from: (Mathworks, 2019).

As it can be seen in Fig. 5, FIL is a technique that is used for testing algorithms developed using hardware description language at a system level, making it possible to create stimulus, signals and testbenches at MATLAB or Simulink and sending them to embedded algorithms in FPGA. In the same way, data can be sent from the algorithm embedded in FPGA to MATLAB or Simulink, which makes it easier to gather data and plot graphs using the information collected during tests. It is also very helpful when data needs to be compared to previous computational simulations using MATLAB.

Using HDL Verifier, the unconstrained MPC was simulated in FPGA, using the same parameters as the MATLAB computational simulations, for two different prediction horizon values: N = 20 and 30. The kit used for implementing unconstrained MPC controller in FPGA is the Xilinx KC705 Evaluation Kit, chosen because of its high quantity of hardware resources, as it can be seen in Tab. 1.

Table 1. Overview of the characteristics and hardware resources of the KC-705 Evaluation Kit by Xlinx (Xilinx, 2019).

Maximum clock frequency	200 MHz
Clock type	Differential
FPGA family	Xilinx Kintex-7
Lookup Tables (LUTs)	203800
Block RAMs (BRAMs)	445
Digital Signal Processors (DSPs)	840
Input/Output (I/O) Pins	500

After all simulations were completed, they were compared to the previous MATLAB computational simulations, which is presented in the next section.

4. RESULTS AND DISCUSSION

Figure 6 and Fig. 7 present the results of both computational simulations of the unconstrained MPC applied to the Boeing 747 system and the results of the same embedded control system in FPGA, using prediction horizons of N = 20 and N = 30 respectively. It is important to notice that the Boeing 747 system is linearized so that no units are used. Following these results, Tab. 2 shows the hardware consumption according to the prediction horizon adopted.

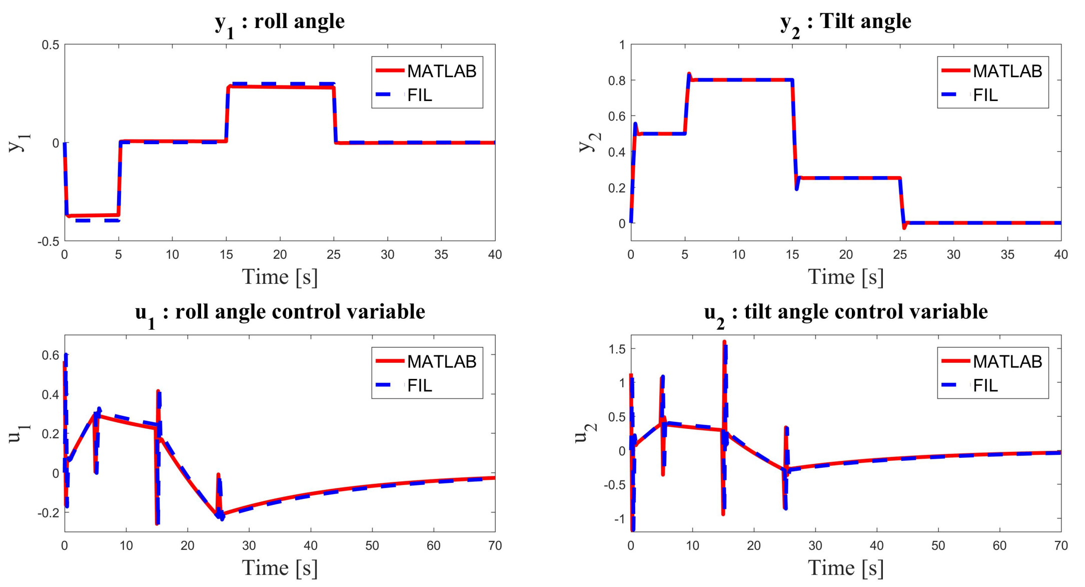

Figure 6. Comparison between simulations of unconstrained MPC in MATLAB and embedded in FPGA (FIL) using a prediction horizon of N = 20.

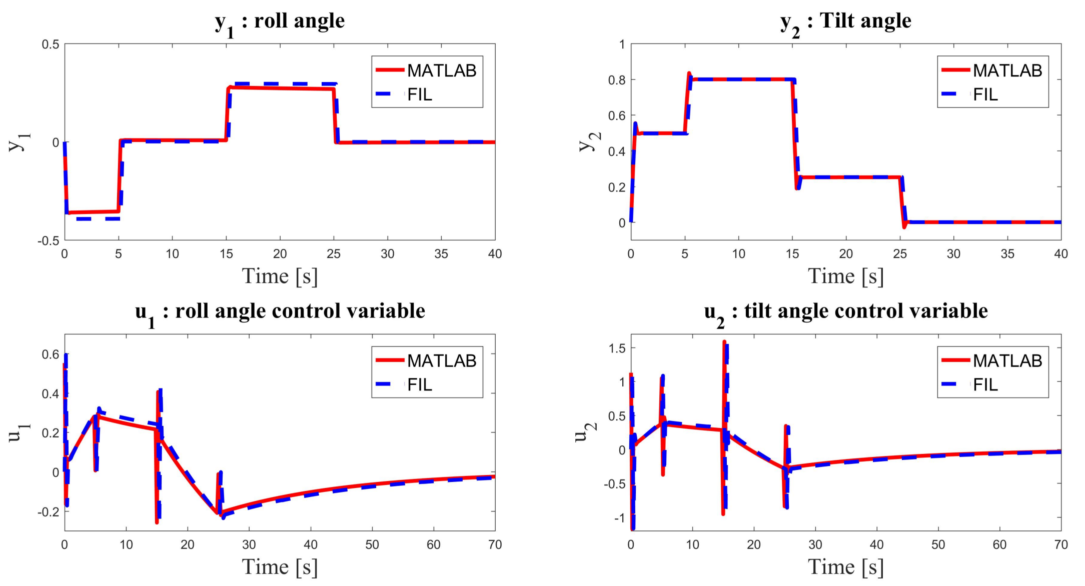

Figure 7. Comparison between simulations of unconstrained MPC in MATLAB and embedded in FPGA (FIL) using prediction horizon of N = 30.

Table 2. Hardware resources consumption necessary for the implementation of the unconstrained MPC controller according to the value of prediction horizon used.

Prediction Horizon	LUTs	Flip Flops	DSPs
20	58982 (28.94%)	12003 (2.94%)	80 (9.52%)
30	88120 (43.24%)	17208 (4.22%)	120 (14.29%)

The results show that embedded unconstrained MPC in FPGA works very similar to MATLAB computational simulations, validating the matrix multiplication, matrix adder and FSM modules developed so far. Also, both outputs (y_1 and y_2) present appropriate behaviour as they both achieve their reference values throughout both the computational and FIL simulations.

The amount of time required by the calculation of the unconstrained MPC control law is of approximately 3.88 μs or 0.00388 ms, which is way lower than the sampling period adopted for the control system ($\tau = 200$ ms). Such fast calculations are obtained using a fully pipelined architecture, as discussed in Section 3. For comparison, MATLAB computation simulations took an average time of 72.91 μs / 0.7291 ms and 91.26 μs / 0.9126 ms for calculating the unconstrained MPC control law using N = 20 and N = 30, respectively. Both times are much lower than the sampling period adopted (200 ms), but also slower than the controller embedded in FPGA.

One important aspect of the proposed architecture is that the calculation time obtained (3.88 μs) is obtained at both tests, using N = 20 and N = 30. It is logical to think that higher values of N would yield higher calculations times, but since the architecture is fully pipelined, the developed MATLAB scripts always make sure that the amount of parallel floating-point multipliers and adders used for unconstrained MPC is capable of performing each matrix operation at one clock cycle, which guarantees that calculation time is the same, regardless of the adopted value of N.

This impacts on the hardware resources usage of these solutions given by Tab. 2, that can be considered quite high for a relatively simple application such as the unconstrained MPC scenario of a linearized Boeing 747 model. However, this decision was made taking in consideration that future applications will consider constrained MPC scenarios, with more complex control problems. For these situations, the additional speed provided by this type of architecture will be necessary, as matrix dimensions and quantity of operations increase, as well as the complexity of the QP, that shall be resolved using quadratic optimization algorithms and not an analytic solution as it is the case of the unconstrained scenario.

Since both results using N = 20 and N = 30 were satisfactory, the first implementation is preferred. Lower values of N result in a lower hardware resources consumption by the FPGA kit for the solution to be implemented, which can also be seen in Tab. 2. This can be explained as the dimension of the matrices that are needed for implementing MPC unconstrained control law vary according to the prediction horizon adopted: the higher the value of N is, the higher the dimensions of these matrices become and thus, the lower prediction horizon that is able to control the system is considered a better choice for implementing it as embedded hardware.

Therefore, analyzing all the informations presented and the results obtained, it is possible to confirm that the developed RTL implementation of linear unconstrained MPC using floating-point arithmetic units is successful.

5. CONCLUSION

With all the results presented so far, it is possible to conclude that the modules developed for the linear unconstrained MPC are working properly. Due to the methodology applied, the modules are capable of being parameterized for different test scenarios and applications, making it possible to control different models using the developed controller.

Up until this point, only linear unconstrained MPC results are presented, which is the first step of validation of the embedded MPC controllers. Implementation of the constrained MPC in FPGA is proposed as a future work alternative. In fact, simulations with the constrained MPC controller are currently being held, using a PGE (Projected Gradient Expansion) algorithm as the MPC optimization solver, for the Boeing-747 model and others models as well.

Thus, this work presents the initial results of the implementation of MPC controllers using FPGA in order to accelerate such control algorithms. This is a field of study in the literature that is very relevant and has space for contributions. Even though only the unconstrained MPC scenario is approached by this work, important contributions are made by making a RTL implementation (parametrization scripts were developed for implementing the controller with appropriate data width and resource management, but the logic of the architecture is not automatically generated as it is the case of HLS) and by using floating-point arithmetic units in order to perform all matrix calculations needed, which guarantee a wide range of numeric values to be represented, allowing different types of systems to be tested in the future.

6. ACKNOWLEDGEMENTS

The authors acknowledge CAPES (Coordenation of Improvement of Undergraduate and Graduate Personnel) for the financial support of this research.

7. REFERENCES

Abughalieh, K.M. and Alawneh, S.G., 2019. "A survey of parallel implementations for model predictive control". *IEEE Access*, Vol. 7, pp. 34348–34360.

Ayala, H., Sampaio, R., Muñoz, D.M., Llanos, C., Coelho, L., Leandro and Jacobi, R., 2016. "Nonlinear Model Predictive Control Hardware Implementation with Custom-precision Floating Point Operations". In *IEEE 24th Mediterranean Conference on Control and Automation (MED)*. pp. 135–140.

Domahidi, A., Ferreau, J., Almér, S., Jerez, J. and Hovgaard, T.G., 2016. "Survey of Industrial Applications of Embedded Model Predictive Control". In *European Control Conference (ECC)*.

Dua, P., Kouramas, K., Dua, V. and Pistikopoulos, E.N., 2008. "MPC on a chip-Recent advances on the application of multi-parametric model-based control". *Computers and Chemical Engineering*, Vol. 32, No. 4-5, pp. 754–765.

Ekaputri, C. and Syaichu-Rohman, A., 2013. "Model predictive control (MPC) design and implementation using algorithm-3 on board SPARTAN 6 FPGA SP605 evaluation kit". *Proceedings of 2013 3rd International Conference on Instrumentation, Control and Automation, ICA 2013*, pp. 115–120.

Ferreau, H.J. *et al.*, 2017. *qpOASES User's Manual*, 3rd edition.

Frank, P.M., 2012. "Advances in Control: Highlights of ECC'99".

Guo, H., Liu, F., Xu, F., Chen, H., Cao, D. and Ji, Y., 2019. "Nonlinear model predictive lateral stability control of active chassis for intelligent vehicles and its fpga implementation". *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 49, No. 1, pp. 2–13.

Hartley, E.N., Jerez, J.L., Suardi, A., Maciejowski, J.M., Kerrigan, E.C. and Constantinides, G.A., 2014. "Predictive control using an fpga with application to aircraft control". *IEEE Transactions on Control Systems Technology*, Vol. 22, No. 3, pp. 1006–1017.

Lahti, S., Sjövall, P., Vanne, J. and Hämäläinen, T.D., 2019. "Are we there yet? a study on the state of high-level synthesis". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 38, No. 5, pp. 898–911.

Luo, B., Shao, Z., Xu, Z., Zhao, J. and Zhou, L., 2011. "A new model predictive controller with swarm intelligence implemented on fpga". In *2011 International Symposium on Advanced Control of Industrial Processes (ADCONIP)*. IEEE, pp. 427–432.

Mathworks, 2019. "Hdl verifier". The Mathworks Inc. 21 Mar. 2019 <https://www.mathworks.com/products/hdl-verifier.html>.

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

Mazen, A., 2013. *A Pragmatic Story of Model Predictive Control: Self-Contained Algorithms and Case-Studies*. CreateSpace Independent Publishing Platform.

Muñoz, D.M., 2012. *Otimização por inteligência de enxames usando arquiteturas paralelas para aplicações embarcadas*. Ph.D. thesis, Universidade de Brasília.

Nane, R., Sima, V.M., Pilato, C., Choi, J., Fort, B., Canis, A., Chen, Y.T., Hsiao, H., Brown, S., Ferrandi, F. *et al.*, 2016. "A survey and evaluation of fpga high-level synthesis tools". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 35, No. 10, pp. 1591–1604.

Navarro, D., Lucı, Ó., Barragán, L.A., Urriza, I., Jimenez, O. *et al.*, 2013. "High-level synthesis for accelerating the fpga implementation of computationally demanding control algorithms for power converters". *IEEE Transactions on Industrial Informatics*, Vol. 9, No. 3, pp. 1371–1379.

Urriza, I., Barragan, L.A., Navarro, D., Artigas, J.I. and Lucia, O., 2010. "Word length selection method for controller implementation on fpgas using the vhdl-2008 fixed-point and floating-point packages". *EURASIP Journal on Embedded Systems*, Vol. 2010, p. 6.

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

Xilinx, 2019. *KC705 Evaluation Board for the Kintex-7 FPGA User Guide UG810*, 1st edition.

Xu, F., Chen, H., Gong, X. and Mei, Q., 2016. "Fast nonlinear model predictive control on FPGA using particle swarm optimization". *IEEE Transactions on Industrial Electronics*, Vol. 63, No. 1, pp. 310–321.

8. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.