Preprints of the 18th IFAC World Congress
Milano (Italy) August 28 - September 2, 2011

Explicit hybrid model predictive control: discontinuous piecewise-affine approximation and FPGA implementation

Tomaso Poggi [*] Sergio Trimboli [**] Alberto Bemporad [**]
Marco Storace [*]

[*] *Biophysical and Electronic Engineering Department, University of Genoa Via Opera Pia 11a, I-16145 Genova, Italy (e-mail: marco.storace@unige.it).*
[**] *Mechanical and Structural Engineering Department, University of Trento, Via Mesiano 77, I-38123 Trento, Italy (e-mail: {trimboli,bemporad}@ing.unitn.it)*

Abstract: In this paper we introduce a digital architecture implementing the explicit solution of a switched model predictive control problem. Given a mixed-logic dynamical system, we derive an explicit controller in the form of a possibly discontinuous piecewise-affine function. This function is then approximated by resorting to piecewise-affine simplicial functions, which can be implemented on a circuit by extending the representation capabilities of a previously proposed architecture to evaluate the control action. The architecture has been implemented on FPGA and validated on a benchmark example related to an air conditioning system.

Keywords: Piecewise linear functions; nonlinear circuits; digital architectures.

1. INTRODUCTION

The real-time implementation of linear model predictive control strategies in embedded architectures was deeply analyzed in Bleris et al. (2006); Ling et al. (2006), and an automated code generation strategy was developed in Richards et al. (2009). Starting from a model, through the definition of a model-based control strategy suitable for the implementation in an embedded architecture, the problem of implementing the control strategy was also investigated for parallel architectures in Ling et al. (2008). Often a system that integrates continuous dynamics and logical structures can be described as a mixed-logic dynamical system (MLD). A suitable strategy to control a MLD system subject to constraints is hybrid model predictive control (HMPC). In order to obtain a HMPC, one has to solve on-line a mixed-integer quadratic or a mixed-integer linear programming problem. For a high-dimensional model, solving this kind of problems may be computationally too expensive for fast real-time application Borrelli et al. (2005).

Explicit reformulations of HMPC can be carried out by solving off-line a sequence of multi-parametric quadratic or linear problems Borrelli et al. (2005). The resulting solution is a possibly discontinuous piecewise affine (PWA) function of the state. In other words, the control modes are linear affine over polytopes partitioning the state domain, thus making this approach more suitable for the embedded control implementations. Storing the gains of the explicit HMPC requires larger and larger memory blocks in the electronic implementation as the number of partitions grows. Moreover, in order to evaluate the control action, the pre-computed gains should be selected, according to the state value, from a look-up table associated to the explicit controller. As a result, since the gains selection from the look-up table can be made by a binary-tree search Tøndel et al. (2003), or by other more sophisticated algorithms, determining the correct mode can be a hard problem if the number of regions is too large.

A suitable strategy, alternative to HMPC, to reduce drastically the number of partitions is the switched MPC (SwMPC) approach, successfully applied for instance in Di Cairano et al. (2009), where a PWA system is controlled by a set of linear MPC controllers, each one defined over a different polytope of the domain. In explicit form, SwMPC is basically a set of patched PWA controllers. For each i-th *region* $\mathcal{R}_i$ of the domain, a linear MPC problem is solved, whose solution is a continuous PWA function defined over a polytopic partition of the region. Note that a MLD model can be converted in an equivalent PWA formulation Bemporad (2004).

The resulting PWA control function may be discontinuous only at the boundaries of the regions. The overall number of polytopes obtained with the explicit SwMPC approach is typically much lower than the one obtained with the explicit HMPC, especially when the number of optimization variables grows. However, the SwMPC complexity reduction with respect to HMPC is not costless, since optimal switching sequences are restricted to constant mode sequences, possibly breaking *a-priori* stability properties. However, *a-posteriori* stability analysis of the SwMPC can be performed exploiting the results in Ferrari-Trecate et al. (2002) and in Rubagotti et al. (2011).

To implement such possibly discontinuous PWA functions in approximate form on fast digital circuits, in this paper we extend to discontinuous PWA functions the results of

Parodi et al. (2005); Storace and Poggi (2010), related to the circuit implementation of continuous PWA functions. We restrict our attention to PWA control functions for which each mode is defined over a hyper-rectangular region. This limits the approach to hybrid dynamical systems where threshold conditions only depend on single components of the state vector.

In order to circuit implement the SwMPC solution in an approximate but fast way, we resort to a modified version of the method proposed in Bemporad et al. (2011). Accordingly, each explicit solution (valid over the i-th hyper-rectangular region) is first approximated by using a PWA continuous function, defined over a regular simplicial partition of the i-th region (called *PWAS* function). Then, the obtained approximations can be merged into one PWAS discontinuous function, which can be directly mapped on programmable hardware such as a field programmable gate array (FPGA).

The architectures able to implement PWAS functions proposed so far in Echevarria et al. (2007); Rovatti et al. (2000); Storace and Poggi (2010) perform a linear interpolation of the values of the function at the vertices of the simplex the input belongs to. The main limit of such an approach is that the implementable functions are continuous. If functions with discontinuities that are not perpendicular to an axis were to be implemented, more complex and power-hungry architectures would be necessary Johansen et al. (2007); Oliveri et al. (2009).

In this contribution we first introduce the architecture for the electronic implementation of discontinuous PWAS functions and then use this architecture to implement explicit SwMPC controllers. The procedure is tested on a hybrid temperature control system. Both the maximum working frequency and the power consumption of the control FPGA implementation are estimated in the range of tens of MHz and tens of mW, respectively.

2. CIRCUIT IMPLEMENTATION OF CONTINUOUS PWAS FUNCTIONS

In this section, we briefly summarize the mathematical theory the proposed architecture is based on and we introduce some basic definitions. We deal with a continuous PWAS function $f_{PWAS} : S \to \mathbb{R}$, defined over a properly scaled n-dimensional compact domain $S = \{z \in \mathbb{R}^n : 0 \leq z_h \leq m_h,\ h = 1, \dots, n,\ m_h \in \mathbb{N}\}$. Function f_{PWAS} can be easily implemented by introducing a regular partition of the domain S Parodi et al. (2005): each dimensional component z_h of the domain S is divided into m_h subintervals of unitary length. As a consequence, the domain S is partitioned into $\prod_{h=1}^{n} m_h$ hyper-squares and contains $N = \prod_{h=1}^{n}(m_h + 1)$ vertices v_k collected in a set V. Each hyper-square can be further partitioned (*simplicial partition*) into $n!$ non-overlapping regular simplices. The coordinates of the corner of the hyper-square closest to the origin that contains a given input z can be found by extracting the integer part of z. The exact position of z within the related simplex is coded by the decimal part of z (denoted as δz) Parodi et al. (2005). The PWAS function f_{PWAS} is linear over each simplex of the partitioned domain S and can be expressed as a linear combination of N α-basis functions

$$f_{PWAS}(z) = \sum_{k=0}^{N-1} c_k \alpha_k(z). \qquad (1)$$

Once the scaled simplicial domain is defined, the basis functions (belonging to the α-basis) are directly defined as well. The k-th α-function is PWAS, holds the value 1 at the vertex corresponding to v_k and the value 0 at all the other vertices.

The shape of a given PWAS function f_{PWAS} is coded by the N coefficients c_k in Eq. (1), which are the values of f_{PWAS} at the vertices v_k of its simplicial partitions. Henceforth, we assume that the coefficients are already determined by a function approximation procedure (see, e.g., Bemporad et al. (2011)).

The coefficients c_k ($k = 1, \dots, N$) are stored by assigning a proper memory address to each vertex v_k of the simplicial partition. Define $\beta_p : \mathbb{N}^n \to \mathbb{N}_b$ as the binarizing operator that, given a column vector of n integer values and a precision p, returns a np-long string of bits, concatenating the binary values of the elements of the vector. For instance, if $v_k = [2,\ 0,\ 5]^T$ and $p = 3$, then $\beta_p(v_k) = 010\,000\,101$. Then, $\beta_p(v_k)$ is an unambiguous address for the vertex v_k. The value of $f_{PWAS}(z)$ can be calculated as a linear interpolation of the f_{PWAS} values at the vertices of the simplex containing z, i.e., as a linear interpolation of a subset of $n + 1$ coefficients c_k:

$$f_{PWAS}(z) = \sum_{j=0}^{n} \mu_j c_{\Omega_j} \qquad (2)$$

where the μ_j's are the weights that give z as a convex combination of the vertices of the simplex that contains it (i.e., $z = \sum_{j=0}^{n} \mu_j v_{\Omega_j}$, with $\sum_{j=0}^{n} \mu_j = 1$) and Ω_j is a function that maps the index j of the weight μ_j into the corresponding index k of one of the vertices surrounding z Parodi et al. (2005). Ω_j, as well as the interpolation weights μ_j, depends on z. This dependence is omitted here for ease of notation.

As a consequence, the circuit realization of a PWAS function proposed in Storace and Poggi (2010) requires three functional elements:

(1) a memory where the N c_k coefficients are stored;
(2) a block that finds, for any given input z, the indices Ω_j and the coefficients μ_j;
(3) a block performing the weighted sum (2).

Since the $\{c_k\}$'s are stored in a memory, Ω_j corresponds uniquely to the address Ω_j^b of the j-th coefficient in Eq. (2), through the binarizing operator β_p

$$\Omega_j^b = \beta_p(\lfloor z \rfloor + a_j), \qquad j = 0, \dots, n \qquad (3)$$

where a_j's are vectors whose components are calculated from the decimal parts of the input z Parodi et al. (2005).

3. GENERALIZATION TO A CLASS OF DISCONTINUOUS FUNCTIONS

The algorithm presented in Sec. 2 can be generalized to include a particular class of discontinuous functions, namely the functions composed of continuous PWAS functions separated by discontinuities that lie perpendicular to a coordinate axis. In this case, we can define an index labeling the subregion a continuous PWAS function is defined over and use this index to solve the point location problem, i.e. to address correctly the memory containing the coefficients.

Let us suppose that there are D_h discontinuities orthogonal to each axis z_h, with $h = 1, \ldots, n$. The discontinuities are hyperplanes (straight lines for $n = 2$) in the form $z_h = d_{h,t}$ $(h = 1, \ldots, n;\ t = 1, \ldots, D_h;\ d_{h,t}$ constant) that further partition the domain S into $P = \prod_{h=1}^{n}(D_h + 1)$ hyper-rectangular regions $\mathcal{R}_i$ (discontinuity partition, $i = 1, \ldots, P$). Figure 1 shows an example of a two-dimensional domain of a discontinuous function with $m_1 = 4$, $m_2 = 3$, $D_1 = 2$ and $D_2 = 1$. Both the regular simplicial partition and the six regions $\mathcal{R}_i$ are highlighted.

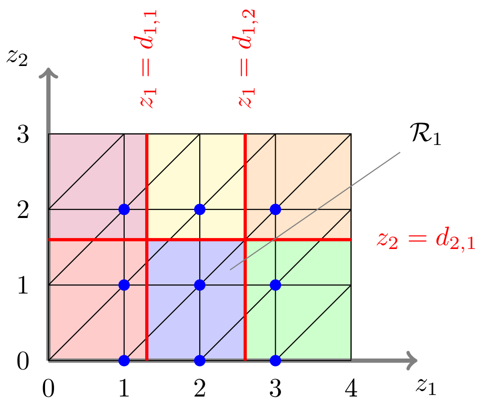

Fig. 1. Two-dimensional domain with discontinuities.

The discontinuous function f_{PWAS} can be defined as follows:

$$f_{PWAS}(z) = f_{PWAS_i}(z) = \sum_{k=0}^{N-1} c_k^i \alpha_k(z), \quad \forall z \in \mathcal{R}_i \tag{4}$$

where f_{PWAS_i} are continuous functions that can be implemented using the technique proposed in Sec. 2. They are defined over the whole domain S, since the set of α-functions is unique for both continuous and discontinuous PWAS functions (see Eqs. (1) and (4)). The shape of a particular function f_{PWAS_i} is coded by the coefficients related to the vertices that lie inside $\mathcal{R}_i$ and immediately outside of the boundary of $\mathcal{R}_i$. Thus, most of the coefficients c_k^i related to vertices that fall outside $\mathcal{R}_i$ can be discarded. Indeed, we need to consider only the set V_i of the vertices that lie inside the smallest hyper-rectangle containing $\mathcal{R}_i$ defined over the vertices of the simplicial partition. For instance, in Fig. 1 the vertices V_1 that define the shape of f_{PWAS_1} over $\mathcal{R}_1$ are marked by blue dots. They are all contained inside the rectangle $[1, 3] \times [0, 2]$. Then, f_{PWAS_i} is completely characterized by the coefficients corresponding to V_i:

$$f_{PWAS_i}(z) = \sum_{k \in \mathcal{K}_i} c_k^i \alpha_k(z), \qquad z \in \mathcal{R}_i,$$

where $\mathcal{K}_i = \{k : v_k \in V_i\}$.

An example of discontinuous PWAS function is shown in Fig. 2. In this case, f_{PWAS} is defined over a one-dimensional domain $S = [1, 5]$, with one discontinuity $(z = d_{1,1} = 2.7)$ and

$$f_{PWAS}(z) = \begin{cases} f_{PWAS_0}(z), & z \in \mathcal{R}_0 = [1, 2.7) \\ f_{PWAS_1}(z), & z \in \mathcal{R}_1 = [2.7, 5] \end{cases}$$

To calculate f_{PWAS_0}, it is necessary to know the value of its coefficients at the vertices v_0, v_1, v_2 (all $\in \mathcal{R}_0$) and $v_3 (\notin \mathcal{R}_0)$, then $V_0 = \{v_0, v_1, v_2, v_3\}$ and $\mathcal{K}_0 = \{0, 1, 2, 3\}$. On the other hand, the set of vertices needed to evaluate f_{PWAS_1} is v_3, v_4, v_5 (all $\in \mathcal{R}_1$) and $v_2 (\notin \mathcal{R}_1)$, then $V_1 = \{v_2, v_3, v_4, v_5\}$ and $\mathcal{K}_1 = \{2, 3, 4, 5\}$. We notice that the correspondence between vertices of the simplicial partition and coefficients is no longer one-to-one, as both c_2^0 and c_2^1 are related to v_2 and both c_3^0 and c_3^1 are related to v_3.

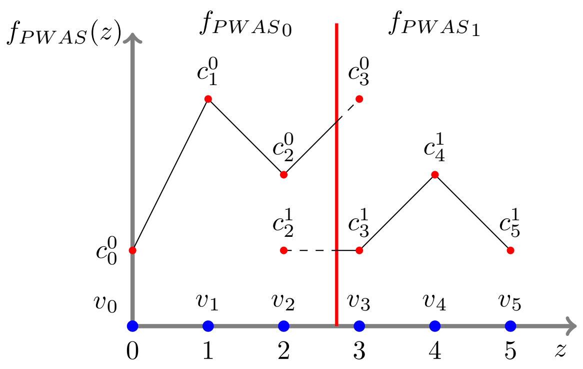

Fig. 2. One-dimensional discontinuous PWAS function

Since some coefficients c_k^i of each function f_{PWAS_i} are in a relation many-to-one with the vertices of the simplicial partition and since they are stored in the same memory, we need to refine the way they are addressed. For any given $z = [z_1, z_2, \ldots, z_n]^T \in \mathcal{R}_i$, we can define

$$r(z) = \begin{bmatrix} \sum_{t=1}^{D_1} u(z_1 - d_{1,t}) \\ \sum_{t=1}^{D_2} u(z_2 - d_{2,t}) \\ \vdots \\ \sum_{t=n}^{D_1} u(z_n - d_{n,t}) \end{bmatrix} \tag{5}$$

where $u(\cdot)$ denotes the unitary step function. Then each rectangle can be uniquely identified by the binary string (with $n \times (p-1)$ bits) $\beta_{p-1}(r(z))$, $z \in \mathcal{R}_i$. We choose p as the lowest integer such that $D_h \le 2^{p-1} - 1$ $(h = 1, \ldots, n)$.

Given a point z, we need to find the rectangle $\mathcal{R}_i$ such that $z \in \mathcal{R}_i$ and the related set of coefficients c_k^i. Thus, the index map Ω_j is redefined so that it corresponds uniquely (for any z) to the memory address

$$\Omega_j^b = \beta_{p+1}(2r(z) + \lfloor z \rfloor + a_j), \qquad j = 0, \ldots, n \tag{6}$$

Finally, a discontinuous PWAS function can be evaluated using the method provided in Sec. 2 by substituting Eq. (3) with Eq. (6). The binary vector $\beta_{p-1}(r(z))$ can be easily obtained by using comparators to process the input z and find the region it belongs to.

To evaluate each function f_{PWAS_i} it is possible to use the architecture A proposed in Storace and Poggi (2010), that provides a correct output every $p+q+n+5$ clock cycles, where p and q are the number of bits used to code the integer part and the decimal part, respectively, and n is the input dimension. As stated before, the coefficients c_k^i defining the shape of f_{PWAS} are stored in a memory and they can be addressed by calculating the strings Ω_j^b. Then, we need to modify the way the address is calculated in Storace and Poggi (2010), according to Eq. (6).

The number of elementary devices (comparators, adders, multipliers, etc.) required to evaluate a discontinuous PWAS function is reported in Tab. 1. The items of the part added to evaluate discontinuous functions are kept separated and described in italic text.

Item	Bits	# Devices
Comparator	q	n
Multiplexer	n	n
ROM	$2^{np} \times b$	1
Adder/Subtractor	$n+1$	n
Adder/Subtractor	q	n
Multiplier	$b \times q$	1
Comparator	$p+q$	$\sum_{i=1}^{n} D_i$
Adder	1	n
Adder	$p+1$	n
Shift Register	p	1

Table 1.

4. SWITCHED MPC

The architecture described in the previous sections can be used to implement an explicit SwMPC controller in approximate form. In this section we summarize the main elements of the SwMPC control strategy. As explained in Sec. 1, a MLD system subject to constraints can be controlled through an implicit HMPC strategy. The explicit HMPC strategy can be applied to a MLD model, after recasting it to an equivalent PWA form. A time-invariant PWA discrete-time model is defined as follows

$$x(k+1) = A_i x(k) + B_i u(k) + f_i \tag{7}$$

$$i \;:\; H_i x(k) \le K_i \;,\; i \in \mathcal{I} \tag{8}$$

where $x \in \mathbb{R}^{n\times 1}$, $u \in \mathbb{R}^{m\times 1}$, $A_i \in \mathbb{R}^{n\times n}$, $B_i \in \mathbb{R}^{n\times m}$, $f_i \in \mathbb{R}^{n\times 1}$ characterizes the i-th mode, H_i, K_i are matrices of suitable dimensions defining the i-th region $\mathcal{R}_i$, $\mathcal{I} = \{1,\dots,P\}$ and P is the number of regions. A suitable strategy to control (7), (8) in state feedback, subject to state and input constraints, is the explicit HMPC Bemporad (2004). This approach requires enumerating all the feasible switch sequences between the dynamics i and solving a multi-parametric quadratic problem for each sequence. Storing all the control gains leads to a large use of memory blocks in the FPGA implementation with respect to a simpler controller such as the SwMPC. Considering only the sequences for which the region i is the same during the prediction steps, since the constraints that define the PWA regions are ignored after the first prediction step, the number of multi-parametric quadratic problems to be solved is equal to the number of PWA regions, leading to a suboptimal solution to the control problem.

In order to formulate (7), (8) as standard linear system, fixing the mode i, we merge the affine term f_i in the input matrix B_i. The resulting set of linear systems is a suitable formulation for a set of linear MPCs. Let $v(k)$ be a measured input disturbance such that $v(k) = 1, \forall k \ge 0$, then (7) can be rewritten as follows

$$x(k+1) = A_i x(k) + B_i u(k) + f_i v(k) \tag{9}$$

or, in a more compact way,

$$x(k+1) = A_i x(k) + \bar{B}_i \bar{u}(k) \tag{10}$$

where $\bar{B}_i = \begin{bmatrix} B_i & f_i \end{bmatrix}$ and $\bar{u} = \begin{bmatrix} u'(k) & v(k) \end{bmatrix}'$. Exploiting model (10), (8) we define a set of linear MPCs based on the following quadratic problem:

$$\begin{aligned} \min_{U=[u_0,\,\dots,\,u_M]} J(x,U) = \sum_{k=0}^{M-1} & x(k)'Qx(k) + \\ & + u'(k)Ru(k) + \rho\epsilon^2 \\ \text{s.t.} \quad x_{min} - \epsilon \le & x(k) \le x_{max} + \epsilon, \\ u_{min} \le & u(k) \le u_{max}, \\ x(k+1) = & A_i x(k) + \bar{B}_i \bar{u}(k) \end{aligned} \tag{11}$$

where M is the prediction horizon; the quantities x_{min}, x_{max}, u_{min}, u_{max} are state and input bounds, respectively; ϵ is a slack variable, weighted by ρ; R, Q are weight matrices of suitable dimensions; A_i, B_i are the i-th model matrices. At time k, only the first component u_0 of the optimal sequence is applied, in a receding horizon fashion. A SwMPC is a set of linear MPCs based on (11) each one defined over its corresponding region $\mathcal{X}_i$. For each control step, one has to evaluate the active mode i and compute the i-th control action.
Problem (11) is stated as a regulation of the states to the origin. A reference tracking problem can be recast as partial state regulation problem by extending the state vector and exploiting the same formulation, as follows. Let $y(k) = Cx(k)$ be the output of model (10), (8), where $C \in \mathbb{R}^{o\times n}$ is the output matrix, then consider the extended state vector $x_e = \begin{bmatrix} x' & r_y' \end{bmatrix}$, where $r_y \in \mathbb{R}$. The reference tracking formulation is obtained by substituting in (11) $Q = \begin{bmatrix} C & -\mathbf{I} \end{bmatrix}' Q_y \begin{bmatrix} C & -\mathbf{I} \end{bmatrix}$, where Q_y is a weight matrix of suitable dimension and $\mathbf{I}$ is the identity matrix of order o. This leads to a reference tracking problem with cost function $J(x,U) = \sum_{k=0}^{M-1} (Cx(k) - r_y)' Q_y (Cx(k) - r_y) + u'(k)Ru(k) + \rho\epsilon^2$.

Exploiting the results in Bemporad et al. (2002), each linear MPC of the SwMPC formulation could be explicitly solved through a multi-parametric quadratic problem, leading to a set of linear explicit MPCs. Moreover, in each region the explicit controller is a continuous PWA function of the state. The overall explicit SwMPC controller is defined as follows.

$$u(k) = F_j^i x(k) + G_j^i \tag{12}$$

$$\text{if } \; H_j^i x(k) \le K_j^i \tag{13}$$

where j indexes the polytopes of the i-th region $\mathcal{R}_i$ in the explicit linear MPC. In the framework described in the previous sections, these polytopes reduce to identical simplices and the regions are hyper-rectangles.

In the next section, a benchmark for the SwMPC implemented with PWAS in a FPGA reveals the capabilities of the proposed approach, suggesting that the SwMPC performances can get very close to the HMPC ones, at least for functions belonging to the class described in Sec. 3.

5. EXAMPLE

In order to test the circuit implementation on FPGA, we propose a revised case study of the hybrid temperature control problem described in the Hybrid Toolbox Bemporad (2004a). The model is a MLD description of an air conditioning system and in closed loop with a HMPC. A discontinuous PWA control of the MLD system is found by using the SwMPC approach described in Sec. 4 using the same HMPC tuning parameters for each linear MPC. Then, each PWA continuous function defining the controller is approximated by a PWAS function. We obtain a discontinuous PWAS controller, which is implemented on a FPGA by using the architecture introduced in Sec. 3.

5.1 Model and control description

The state vector x represents two different temperatures, while the input u is the ambient temperature to be regulated:

$$x(k) \triangleq \begin{bmatrix} T_1(k) \\ T_2(k) \end{bmatrix}, \; u \triangleq T_{amb}$$

The auxiliary variables associated with threshold events u_{hot}, u_{cold} are such that

$$\begin{aligned} &\text{IF } x_1 \le T_{c1} \text{ OR } (x_2 \le T_{c2} \text{ AND } x_1 < T_{h1}) \\ &\qquad \text{THEN } u_{hot} = U_h \text{ ,ELSE } u_{hot} = 0 \\ &\text{IF } x_1 \le T_{h1} \text{ OR } (x_2 \le T_{h2} \text{ AND } x_1 < T_{c1}) \\ &\qquad \text{THEN } u_{cold} = U_c \text{ ,ELSE } u_{cold} = 0 \end{aligned} \tag{14}$$

where $T_{\{c1,c2,h1,h2\}}$ are constant temperatures, U_c represents the air conditioning power flow, U_h represents the heater flow.

The hybrid model is stated as follows.

$$\begin{aligned} x_1(k+1) &= x_1(k) + T_s[-\alpha_1(x_1(k) - u(k)) + \\ &\qquad + K_1(u_{hot}(k) - u_{cold}(k))] \\ x_2(k+1) &= x_2(k) + T_s[-\alpha_2(x_2(k) - u(k)) + \\ &\qquad + K_2(u_{hot}(k) - u_{cold}(k))] \end{aligned} \tag{15}$$

where $T_s = 0.5\ s$ is the sampling time and $\alpha_{\{1,2\}}$, $K_{\{1,2\}}$ $[s^{-1}]$ are constant coefficients. The output of the system is the state x.
The state and input constraints for the hybrid model are the following.

$$\begin{aligned} -10 \le & u(k) \le 50 \\ -10 \le & x(k) \le 50 \end{aligned} \tag{16}$$

By exploiting the results of Bemporad (2004), the hybrid model (14),(15) is translated into an equivalent PWA model, which is defined over a three-dimensional domain ($n = 3$, with 2 state dimensions and 1 input dimension) partitioned into 5 polyhedral regions. As shown in Fig. 3, where a section of the PWA model for $T_{amb} = 25°C$ is shown, the polyhedral partition has boundaries parallel with respect to the state axes T_1 and T_2. Since the open-loop dynamics in regions 2, 3 and in 1, 4 are the same, the MPC calculated over the partition $\mathcal{P}$ associated to region 2 is equivalent to the one calculated in region 3, as well as region 1 shares the same MPC with region 4, although on different sets of states.

As described in Sec. 4, the affine terms in the PWA formulation are considered as constant measured input disturbances in the SwMPC formulation. The target of the controller is to track a reference r_y for $y = x_2$, while enforcing the constraint $x_1 \ge 25$ in addition to the constraints in (14), (16). The controllers (HMPC and SwMPC) share the same tuning parameters: $M = 4$, $Q_y = 1$, $R = 0$, $\rho = +\infty$ (corresponding to hard constraints). For a constant reference tracking $r_y = 30$, the difference Δ_u between the manipulated variables in HMPC and SwMPC is negligible. The controllers characteristics are summarized in Table 2.

Item	HMPC	SwMPC (for each controller)
# params	3 (2 states, 1 reference)	3
# variables	36 (12 cont., 24 binary)	6
# inequalities	96 (mixed-integer)	26
# regions	N/A	5
# subregions explicit	1385 *	10 (maximum), (35 total)

* Possible overlapping regions that are never optimal are not removed

Table 2.

5.2 FPGA implementation

The PWAS approximation of the SwMPC controller has been implemented on a Xilinx Spartan 3 FPGA, using the VHDL language to define the circuit.

The first step towards the FPGA implementation is the PWAS approximation of each explicit linear MPC by applying the method proposed in Bemporad et al. (2011). The result of the approximations are five continuous PWAS functions defined all over the domain, partitioned into simplices using $m_h = 7$ divisions along each dimensional component.

The second step is to merge the five continuous PWAS functions into one discontinuous PWAS function f_{PWAS}. Since there are two discontinuities along the first and the second dimensions, the discontinuity partition is composed by nine hyper-rectangular subregions $\mathcal{R}_i$, $i = 1, \ldots, 9$.

Fig. 4 shows the input and state signals obtained using the PWAS approximated control and the HMPC approach, where a sinusoidal reference r_y for x_2 is imposed.

The estimated maximum working frequency is 40 MHz, that corresponds to a throughput of one sample every 550 ns, with a power consumption of 85 mW. The approximated discontinuous PWAS control occupies 69% of the hardware resources of the chosen FPGA.

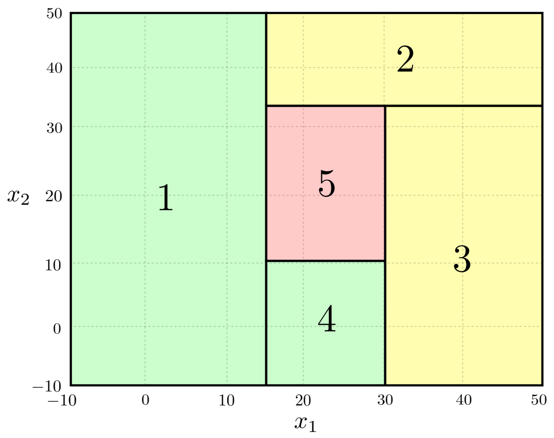

Fig. 3. PWA state partitions for $u = 25°C$.

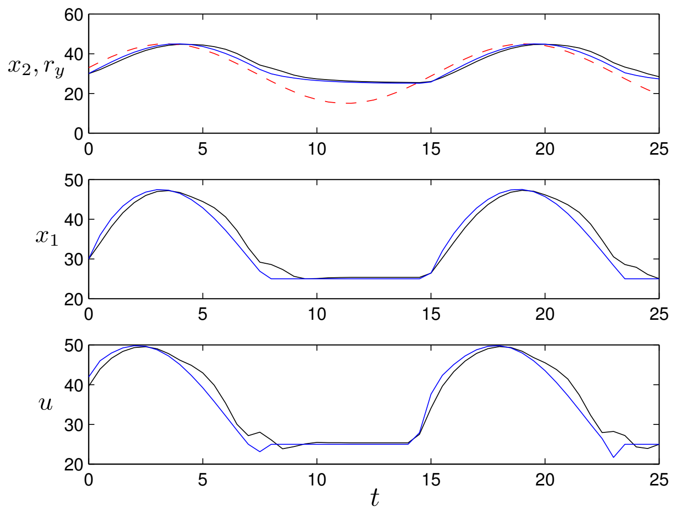

Fig. 4. HMPC (blue) vs. PWAS (black).

6. CONCLUSIONS

For hybrid MPC controllers approximated as switched linear MPC controllers we have proposed an architecture to evaluate discontinuous PWAS functions by extending the architecture proposed in Storace and Poggi (2010). The implementation requires the introduction of comparators and 1-bit adders which are very simple and fast devices and so is remarkably efficient. This fact is a direct consequence of the chosen class of discontinuous functions. Thus, our circuit represents a good trade-off between model complexity and circuit performances in terms of area occupation and power consumption.

A further generalization of the proposed architecture can be obtained by using a binary-tree search Johansen et al. (2007); Oliveri et al. (2009) instead of the bank of comparators. In this case, we could implement discontinuous functions that are continuous over non hyper-rectangular regions, at the cost of a more complex circuit.

ACKNOWLEDGEMENTS

This work was partially supported by the European Commission through project MOBY-DIC "Model-based synthesis of digital electronic circuits for embedded control" (FP7-INFSO-ICT-248858),
`http://www.mobydic-project.eu`.

REFERENCES

A. Bemporad. Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form. *IEEE Trans. Autom. Control*, 49(5):832 – 838, May 2004.

A. Bemporad. Hybrid Toolbox - User's Guide, 2004a. `http://www.ing.unitn.it/~bemporad/hybrid/toolbox`.

A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

A. Bemporad, A. Oliveri, T. Poggi, and M. Storace. Ultra-fast stabilizing model predictive control via canonical piecewise affine approximations. *IEEE Trans. Autom. Control.*, 2011. In press.

L. G. Bleris, P. D. Vouzis, M. G. Arnold, and M. V. Kothare. A co-processor FPGA platform for the implementation of real-time model predictive control. In *Amer. Control Conf.*, Minneapolis, MN, June 2006.

F. Borrelli, M. Baotić, A. Bemporad, and M. Morari. Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. *Automatica*, 41 (10):1709–1721, 2005.

S. Di Cairano, H. E. Tseng, D. Bernardini, and A. Bemporad. Steering vehicle control by switched model predictive control. In *6th IFAC Symp. Adv. Automotive Control*, Munich, Germany, July 2009.

P. Echevarria, M. V. Martínez, J. Echanobe, I. del Campo, and J. M. Tarela. Digital hardware implementation of high dimensional fuzzy systems. In *Applications of Fuzzy Sets Theory*, Lecture Notes in Computer Science, pages 245–252. Springer, Berlin, 2007.

G. Ferrari-Trecate, F. A. Cuzzola, D. Mignone, and M. Morari. Analysis of discrete-time piecewise affine and hybrid systems. *Automatica*, 38(12):2139–2146, December 2002.

T. A. Johansen, W. Jackson, R. Schreiber, and P. Tøndel. Hardware synthesis of explicit model predictive controllers. *IEEE Trans. Control Syst. Techn.*, 15(1):191–197, Jan. 2007.

K. Ling, B. Wu, and J. Maciejowski. Embedded model predictive control (MPC) using a FPGA. In *Proc. 17th IFAC World Congress*, pages 15250–15255, Seoul, Korea, July 2008.

K. V. Ling, S. P. Yue, and J. Maciejowski. A FPGA implementation of model predictive control. In *Amer. Control Conf.*, Minneapolis, MN, June 2006.

A. Oliveri, A. Oliveri, T. Poggi, and M. Storace. Circuit implementation of piecewise-affine functions based on a binary search tree. In *Proc. Europ. Conf. Circ. Th. Design (ECCTD'09)*, pages 145–148, Antalya, Turkey, August 23-27 2009.

M. Parodi, M. Storace, and P. Julián. Synthesis of multiport resistors with piecewise-linear characteristics: a mixed-signal architecture. *Int. J. Circ. Th. Appl.*, 33 (4):307–319, 2005.

A. Richards, W. Stewart, and A. Wilkinson. Auto-coding implementation of model predictive control with application to flight control. In *Proc. Europ. Control Conf. 2009*, Budapest, Hungary, August 2009.

R. Rovatti, C. Fantuzzi, and S. Simani. High-speed DSP-based implementation of piecewise-affine and piecewise-quadratic fuzzy systems. *Signal Processing*, 80:951–963, 2000.

M. Rubagotti, S. Trimboli, D. Bernardini, and A. Bemporad. Stability and invariance analysis of approximate explicit MPC based on PWA Lyapunov functions. In *18th IFAC World Congress*, Milan, Italy, August 2011.

M. Storace and T. Poggi. Digital architectures realizing piecewise-linear multi-variate functions: two FPGA implementations. *Int. J. Circ. Th. Appl.*, 37, 2010. in press, doi: 10.1002/cta.610.

P. Tøndel, T. A. Johansen, and A. Bemporad. Evaluation of piecewise affine control via binary search tree. *Automatica*, 39(5):945–950, 2003.