Digital Implementation of Hierarchical Piecewise-Affine Controllers

I. Baturone[1,2], M. C. Martínez-Rodríguez[1,2], P. Brox[2], A. Gersnoviez[3], S. Sánchez-Solano[2]
[1]University of Seville. [2]Microelectronics Institute of Seville (IMSE-CNM-CSIC). [3]University of Cordoba. Spain
lumi@imse-cnm.csic.es

***Abstract*-This paper proposes the design of hierarchical piecewise-affine (PWA) controllers to alleviate the processing time or prohibitive memory requirements of large controller structures. The constituent PWA modules of the hierarchical solution have fewer inputs and/or coarser partitions, so that they can reduce considerably the hardware resources required and/or the time response of the controller. A design methodology aided by CAD tools is employed to design the parameters of the controller, implement its architecture in an FPGA, and verify the static and dynamic behavior of the digital implementation by applying hardware-in-the-loop testing.**

I. Introduction

A piecewise-affine (PWA) controller provides a linear (affine) control law, $f_{PWA}: D \rightarrow \mathbb{R}$, for each region in which the input domain, D, is partitioned ($D \subset \mathbb{R}^n$):

$$f_{PWA}(x) = f_i^T x + g_i\,,\ x \in P_i\,,\ i=1,\dots,P \qquad (1)$$

where $f_i \in \mathbb{R}^n$, $g_i \in \mathbb{R}$, and P_i are P non overlapping regions, called polytopes, that induce a polyhedral partition of the domain.

Each polytope is a closed set of points delimited by E *edges*:

$$P_i = \{x \in R^n : h_j^T x + k_j \le 0,\ j = 1,\dots,E\} \qquad (2)$$

where $h_j \in \mathbb{R}^n$, $k_j \in \mathbb{R}$, and the E edges are (n-1)-dimensional hyperplanes in the form $h^T_j\, x + k_j = 0$.

Fig. 1 illustrates an example of polyhedral partition (25 polytopes and 42 edges) of a 2-dimensional input domain. The colors filling the polytopes represent the different affine control laws associated with each polytope.

PWA control laws are being widely used in modeling and control of linear, nonlinear and hybrid dynamical systems. As a matter of fact, PWA systems are equivalent to mixed logical dynamical (MLD), linear complementarity (LC), and other hybrid systems, as shown in [1]. Attractiveness of PWA controllers is founded on their capability to approximate any nonlinear control law within any specified error (including linear threshold events and mode switching) as well as being the simplest extension to linear control laws, which engineers are familiar with. Hence, there is extensive research in synthesizing PWA controllers by using Lyapunov-based methods and, specially, model predictive control (MPC). In the latter case, the action control is obtained by solving a finite horizon open-loop optimal control problem at each sampling time. Since computing on line the constrained optimization problem is costly, it has been proposed recently to solve the optimization problem off line by obtaining an explicit MPC controller that expresses the control action as a PWA function of the plant state values (an explicit state feedback solution) [2].

In general, there is a trade-off between complexity and optimality in the design of the PWA controller. In the case of explicit MPC-based PWA controllers, the number of polytopes increases as more exactly the multiparametric programming problem is solved. This also happens when using PWA systems to approximate a given control law and the approximation error is wanted to be small. Several approaches have been recently proposed in the literature to address this trade-off. The work in [3] and [4] proposes to define suboptimal explicit MPC control based on, respectively, a hypercubic partition and extended Voronoi diagram. Other authors propose algorithms such as binary search tree [5] and lattice functions [6] to implement the optimal controllers.

The solution proposed in this paper is to reduce the complexity of a PWA controller by designing a hierarchical PWA controller (composed of PWA modules) that approximates the original one and is very adequate for digital

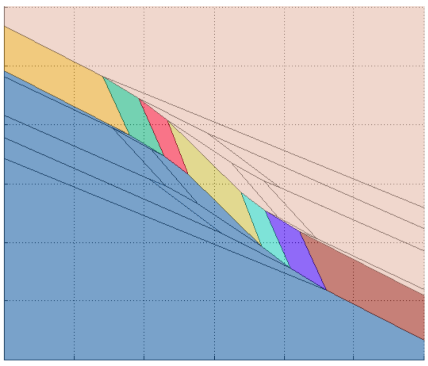


Fig. 1. Polyhedral partition of a PWA controller for the double integrator (MPC with a prediction horizon of 4).

This work has been partially supported by European Community under the MOBY-DIC Project FP7-IST-248858 (www.mobydic-project.eu), by Ministerio de Ciencia e Innovación under the Projects TEC2008-04920 and DPI2008-03847, and by Junta de Andalucía under the Project P08-TIC-03674 (with support from the PO FEDER-FSE).

circuit implementations. The idea is to divide the problem into simpler problems so as to provide very much small-size, low-power and high-speed realizations, which allows spreading the application of PWA controllers to more challenging embedded problems. The paper is organized as follows. Section II reviews briefly the digital solutions proposed to implement PWA controllers. Section III describes the proposal of designing hierarchical PWA controllers. The design methodology employed is summarized in Section IV and illustrated with a detailed application example in Section V. Finally, conclusions are given in Section VI.

II. Digital Circuits for PWA Controllers

Two steps are required to compute the output of a PWA controller to a given input x: (1) finding the polytope P_i that the input belongs to, and (2) evaluating the affine expression $f^T_i\, x + g_i$ associated with that polytope. The first step, called point location problem, is the key point to obtain an efficient implementation. Among the different approaches proposed to address this point (such as [3]-[6]), those implemented by digital circuits have been the generic and the simplicial approaches. They are described briefly as follows.

A. Generic PWA architecture

Checking if x belongs to one of the P polytopes by comparing x with all the edges of each polytope is computationally hard since the number of comparisons to check is high. As an alternative to this combinatory search, the authors in [5] propose to build off-line a binary search tree (with a depth as small as possible), where each non leaf node represents an edge and the leaves contain the index i of a polytope. By exploring the tree on-line, from the root to a leaf, it is possible to locate the polytope containing the input by evaluating a relatively small number of edges.

A digital architecture implementing this approach is described in [7]. It consists of a block for input acquisition (which admits the input in serial), a finite-state machine for exploring the binary search tree (with as many states as edges in the partition), a memory that stores the coefficients h_j, k_j, f_i, g_i, and a multiply-accumulate block that calculates the affine expressions for the edges and for the output in serial.

The number of coefficients to store is $(n+1)*(E+P)$. The time needed to calculate the value of the function at a given input point is $[n+(n+2)*tree\_depth]*T_{CK}$, where T_{CK} is the clock period and *tree_depth* is the depth of the tree.

The edges evaluated at each tree level usually subdivide polyhedral regions with equal affine control laws. The depth of the tree is logarithmic in the number of subdivided regions. However, this number may be significantly larger than the original number of regions. Although the scheme works very well for small partitions, it could not be applicable to large controller structures due to the prohibitive pre-processing time or on-line memory requirements.

For example, for the partition shown in Fig. 1, the depth of the tree in [7] is 6, which means 64 subdivided regions, approximately, while the number of different affine control

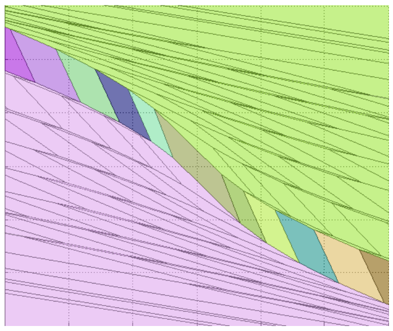


Fig. 2. Polyhedral partition of a PWA controller for the double integrator (MPC with a prediction horizon of 12).

laws (represented with different colors) is only 9. Regarding storage, 3*64=192 coefficients can be retrieved from the memory to implement the PWA function of each region. However, only 3*9=27 coefficients are strictly different (more than 85% of the memory would be redundant).

The polyhedral partition (25 polytopes) in Fig. 1 corresponds to an explicit MPC control law with a prediction horizon of 4. The situation is even worse if the explicit control law is obtained for a bigger prediction horizon. Fig. 2 illustrates the polyhedral partition for a prediction horizon of 12 (149 polytopes, 552 edges, and only 13 different affine control laws).

B. Simplicial PWA architecture

A subclass of PWA functions, called piecewise-affine simplicial (PWAS), was proposed from the circuit designer community as a good trade-off between approximation capability and circuit complexity [8]. In this approach, every dimensional component of D is divided into m subintervals, this resulting in a partition of D into m^n hyper-rectangles. Besides, each hyper-rectangle is further partitioned into $n!$ non overlapping simplexes. The value of the PWA function at a point x is calculated as a linear combination of the values of the function at the $(n+1)$ vertices of the simplex that contains x. Fig. 3 illustrates a portion of a PWAS function defined over a 2-dimensional space (the simplexes are triangles).

Several digital architectures have been reported in the literature implementing this approach [9]-[11]. They consist of a memory that stores the values of the function at the simplex vertices, a block to find the simplex within the hyper-

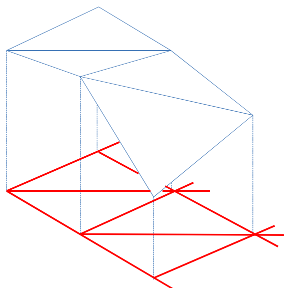


Fig. 3. Simplicial partition of a 2-dimensional space.

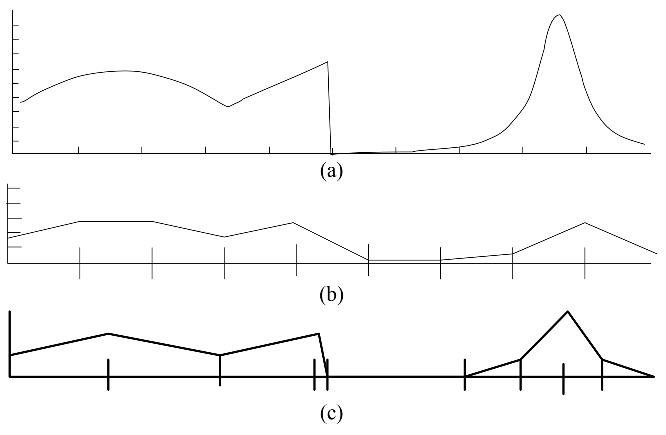


Fig. 4. (a) Unidimensional example of approximation with: (b) PWAS uniform and (c) PWAS non uniform.

rectangular that the point belongs to (finding the hyper-rectangular is an easy task), and a multiply-accumulate or a weighted sum block that calculates the affine expression for the output (in serial or parallel, respectively).

The coefficients to store are $(m+1)^n$ simplex vertices. The time needed to calculate the value of the function at a given input point can be 1 clock period (in the case of a fully parallel architecture) or grows as $[\log_2 m+n]*T_{CK}$ (in the case of a serial architecture) [11].

This approach approximates a given (generic) PWA controller. In general, the approximation error decreases as the number m increases. However, the PWA controller may require finer partitions for several regions and coarser for others. If the m subintervals per input are uniformly distributed, the number m should be imposed by the finer partition. Hence, non-uniform PWAS solutions would be more adequate in these cases, as proposed in [12]. This is illustrated in Fig. 4 for a unidimensional example. In this example, the root mean square error of the approximation is 13.4% when considering 9 uniform subintervals. This error is reduced to 2.1% if the subintervals are not uniform.

In any case, the main problem of the simplicial approach is known as "the curse of dimensionality", which means that complexity grows exponentially with the number n of inputs (since information to store is $(m+1)^n$). Hence, this approach is adequate for PWA controllers with a small number of inputs (and, preferably, a small number of m subintervals per input).

III. Hierarchical PWA Controllers

Hierarchical PWA controllers can reduce considerably the complexity of the solution because a complex PWA module can be obtained by interconnecting simple PWA modules. The constituent modules are simpler because of dealing with fewer inputs and/or smaller partitions. If the modules are simple, any of the architectures described above can be employed without problems.

Connections between PWA modules should fulfill the following requirements to form a PWA controller:

1.- The output of a PWA module is the input of another PWA module. PWA property of the global output is preserved because a PWA function of a PWA function is another PWA function. Such kind of connections is illustrated in Fig. 5a and b for the case of 2 inputs.

2.- Addition, substraction, maximum or minimum are the only operators between modules, as they maintain PWA nature. This kind of connections is illustrated in Fig. 5c.

Before describing how to design and implement hierarchical PWA controllers in the next section, let us illustrate the advantages of using them (in particular, the structure in Fig. 5c) with the following examples.

A. Hierarchical versus non-hierarchical generic PWA

Consider the problem of regulating to the origin the discrete-time unstable multivariable system:

$$x_{t+1} = \begin{bmatrix} 1.3 & 1 \\ 0 & 1.1 \end{bmatrix} x_t + \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} u_t \qquad (3)$$

An explicit MPC state feedback control vector $u=(u_1, u_2)$ has been obtained as a solution of the finite-time optimal control problem. The non-hierarchical generic PWA controller is defined over a partition with 88 polytopes (88 affine functions of 2 inputs for each control output) separated by 341 edges. It is shown in Fig. 6a for the case of u_2. To approximate this control output u_2, a hierarchical PWA controller has been designed with 2 PWA modules of one input with 8 subintervals per input, which means 8+8=16 affine functions of one input. The surface provided by the hierarchical solution is shown in Fig. 6b.

B. Hierarchical versus non-hierarchical simplicial PWA

The ideal control surface to park in diagonal an autonomous robot by driving backward was obtained after a kinematic and dynamic analysis of the robot features. It is shown in Fig. 6c. A hierarchical PWA controller has been designed with 2 PWA modules of one input with 5 subintervals per input, which means to store 20 parameters for defining the 5+5=10 affine functions of one input. The surface provided by the hierarchical solution is shown in Fig.

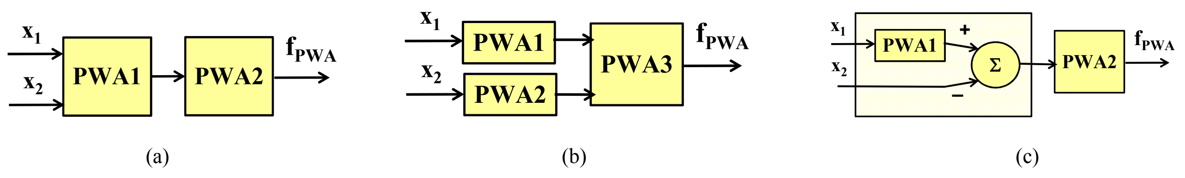


Fig. 5. Examples of hierarchical PWA controllers with two inputs.

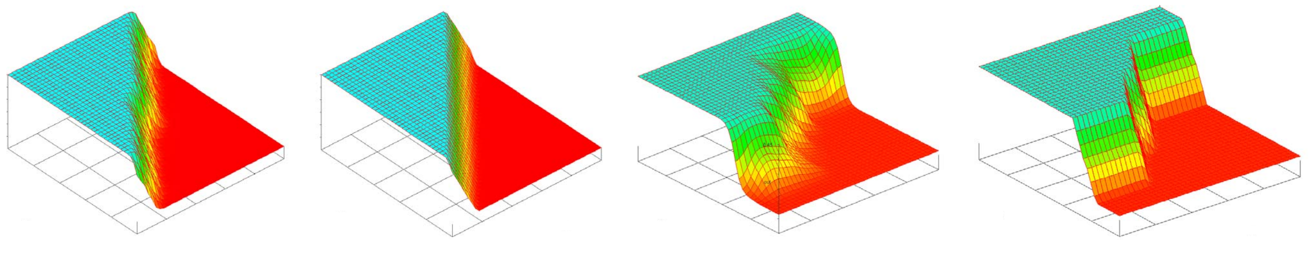


Fig. 6. Control surfaces in (a) and (c) approximated by hierarchical PWA controllers in (b) and (d), respectively.

6d. The approximation error is similar to that obtained with a non-hierarchical PWAS module of 2 inputs with 9 and 7 subintervals per input, which means to store 80 parameters of the simplex vertices.

Another example is described in detail in Section V.

IV. Design Methodology. FPGA Implementations

The methodology proposed to design and implement hierarchical PWA controllers combines the use of several CAD tools from three design environments: Matlab-Simulink, Xfuzzy (a design environment developed at Microelectronics Institute of Seville and University of Seville) [13], and ISE by Xilinx. The steps to carry out are the following.

A. Design of the optimal non-hierarchical controller

The first step is to design the non-hierarchical controller whose performance will be approximated by the hierarchical approach. The designed controller should be explicit so as to generate a set of numerical data for applying supervised learning techniques in the following step.

In the case of explicit MPC-based PWA controllers, the Hybrid Toolbox for Matlab allows obtaining an optimal non-hierarchical controller [14].

B. Structure and parameters of the hierarchical controller

Hierarchy may be induced from the analysis of the control problem (i.e. multiple-input multiple-output controllers are designed as several multiple-input single-output controllers) as well as from the heuristics involved. Once the structure is selected, the parameters of each constituent module should be also selected. The latter selection can be formulated as an optimization problem where different algorithms and CAD tools can be applied. For example, the tool *xfsl* from Xfuzzy allows applying gradient-descent, second-order, Gauss-Newton, and statistical supervised learning algorithms through a hierarchy of modules [15].

Since the hierarchical solution approximates the optimal controller, the hierarchical controller requires a posteriori analysis of its stability. A constraint that should be imposed to ensure local optimality and stability is that the polytope containing the equilibrium point in the hierarchical approach is included within the corresponding polytope in the optimal controller. In addition, there should be a feasible backup gain outside the ranges of the input variables [16].

In any case, a closed-loop simulation should be carried out with a model of the plant to verify the behavior of the controller. This can be done with Simulink or with the tool *xfsim* of Xfuzzy.

C. Digital Implementation on FPGAs

The last step is the digital implementation of the designed controller. This step also involves the approximation of the controller already verified in the previous step, mainly due to the number of bits used in the different blocks of the implementation. Hence, a closed-loop simulation including hardware details should be carried out again. Xilinx System Generator tool in Matlab is very useful for this purpose. Once validated the design at this level, System Generator allows generating the HDL code, the bitstream file to program the FPGA, or the environment configuration to perform hardware/software co-simulation. In the latter case, the controller implemented in the FPGA interacts with the plant model described in Matlab, which is known as hardware-in-the-loop verification.

V. Application Example

Let us illustrate in detail how the above described methodology has been applied to design the controller of the double integrator plant, which is one of the most fundamental systems in control engineering. The double integrator is described as follows:

$$\ddot{y}(t) = u(t) \tag{4}$$

The goal of the control action, *u(t)*, is to regulate the system position, *y(t),* and velocity, $\dot{y}(t)$, to the origin, with the hard constraints $-1 \le u(t) \le 1$. The design of the optimal PWA controller for this problem is performed automatically by the Hybrid Toolbox for Matlab. Depending on the prediction horizon, the resulting control law, *u(t)*, is defined over more or less polytopes in the two-dimensional space *y(t)*, $\dot{y}(t)$, as was illustrated in Fig. 1 and 2. The numerical data obtained from optimal controllers with different prediction horizons are employed to learn the parameters of the hierarchical structure shown in Fig. 5c. The tool *xfsl* of Xfuzzy allows automating this process. A relevant result is

TABLE I
APPROXIMATING THE OPTIMAL MPC CONTROLLER WITH A HIERARCHICAL CONTROLLER

Horizon	Exact regions (optimal)	Approx. regions (hierarchical)	RMSE
4	25	7	1.17%
6	57	7	1.21 %
8	87	7	1.21 %
10	119	7	1.17 %
12	149	7	1.17%

that the hierarchical solution with the same number of regions is able to approximate different optimal controllers with regions that increase rather rapidly. This is shown in Table I. Fig. 7a and c illustrates, respectively, the optimal control surface and polyhedral partition for a horizon of 4. The control surface and polyhedral partition of the hierarchical solution are shown in Fig. 7b and d. It can be seen how the region containing the equilibrium point in the hierarchical approach is included within the corresponding region in the optimal controller, as required to ensure local optimality. Stable behavior has been confirmed with closed-loop simulations with a model of the plant, using the simulation tool *xfsim* of Xfuzzy.

Once selected the structure and parameters of the hierarchical approach, it has been described by a Simulink model with blocks taken from the Xilinx Blockset for System Generator. The constituent PWA modules are very simple. They have one input and one output and very few pieces, which can be further reduced if exploiting that they implement odd functions (i. e., PWA(x)=-PWA(-x)). Hence, a fully parallel architecture, as shown in Fig. 8, has been developed, with 12 bits for the input and output variables. Fig. 9a shows the control surface provided by the hardware implementation. Global errors of this surface compared with the optimal control surface are shown in Fig. 9b-c.

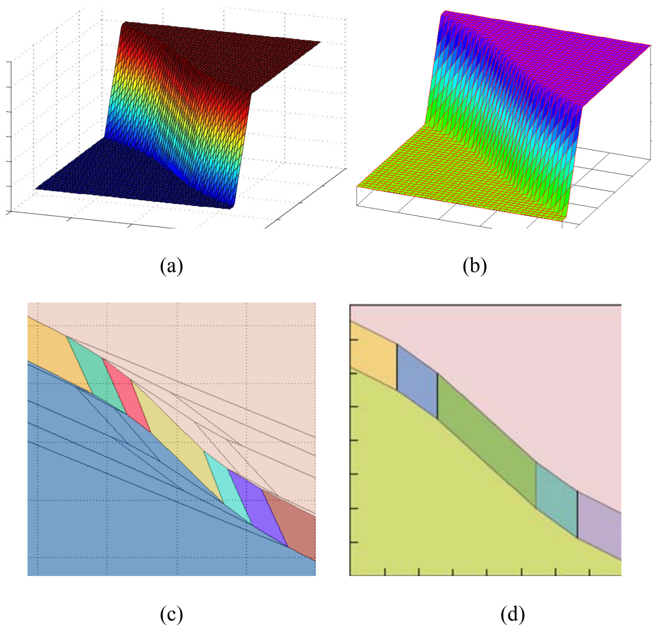


Fig. 7. (a) and (c) Optimal control. (b) and (d) Hierarchical control.

Again the closed-loop behavior of the designed controller (now including all the hardware details) has been analyzed with Simulink. Fig. 10a-d illustrate some of these results. They are quite similar to the results obtained with *xfsim*, thus confirming that hardware implementation performs correctly.

Using System Generator and Xilinx ISE tools, the hierarchical controller has been implemented on a Xilinx Spartan 3 (xc3s200-5ftp256). The occupation and timing results are shown in Table II. This table shows the advantages of the hierarchical versus non-hierarchical (generic and simplicial) solutions. Finally, performance of the designed controller has been also verified by combining the controller implemented in the FPGA with a double integrator model implemented in software (hardware-in-the-loop testing).

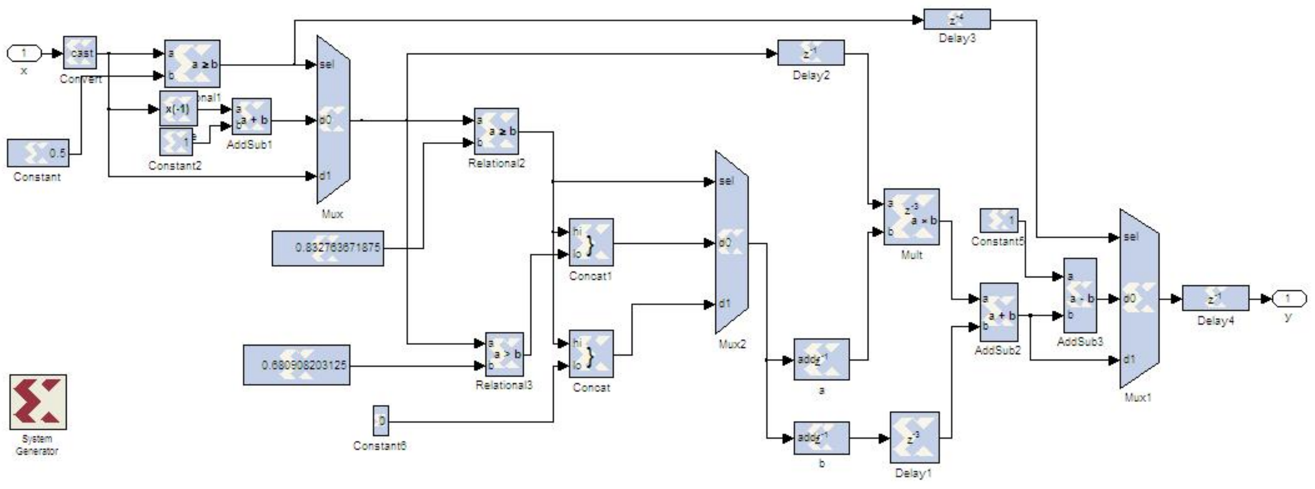


Fig. 8. Fully parallel architecture for the PWA modules of the hierarchical controller.

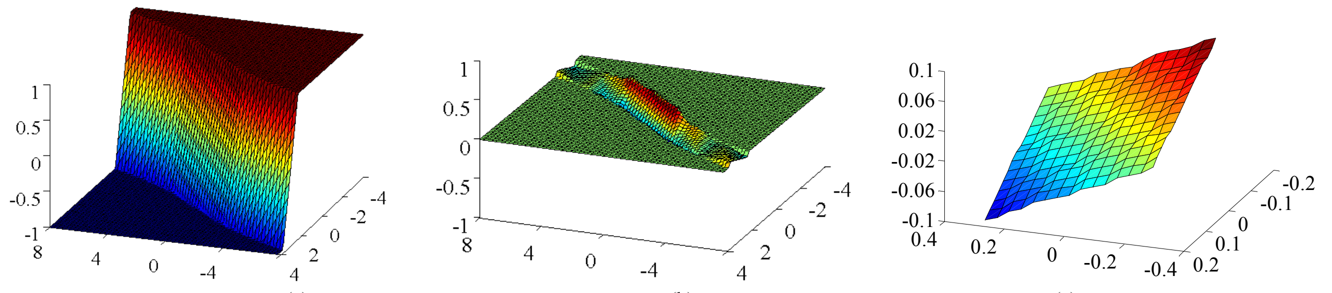

(a) (b) (c)

Fig. 9. (a) Hierarchical control surface provided by the hardware implementation. (b) Difference between the hierarchical control surface and the optimal control surface. (c) Zoom of error surface in (b) around the equilibrium point.

TABLE II
IMPLEMENTATION RESULTS ON A SPARTAN 3 FPGA

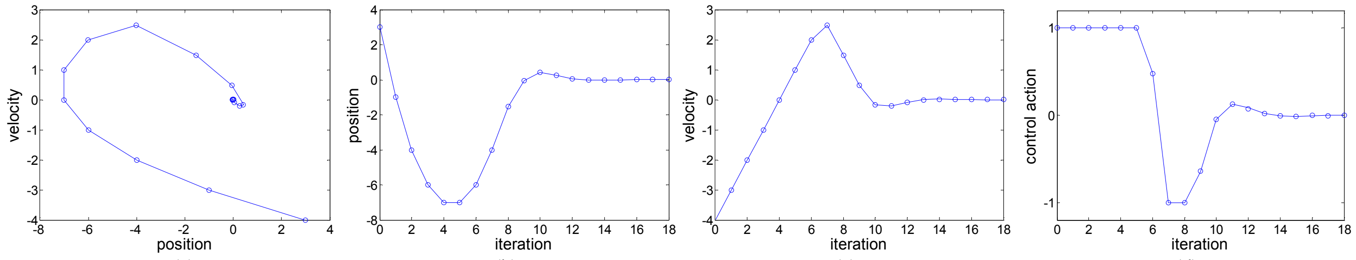
	Generic PWA [7]	PWAS (serial) [11]	PWAS (parallel) [11]	Hierarchical PWA
Slices	11.8%	11%	16%	7%
Clock Period	14.7 ns	14 ns	48.7 ns	8.2 ns
Multipliers	1	1	3	2
Time response (throughput)	26 cycles (381.9 ns)	14 cycles (196 ns)	1 cycle (48.7 ns)	1 cycle (8.2 ns)

VI. CONCLUSIONS

Many control surfaces can be approximated by hierarchical PWA controllers composed of PWA modules. Since the constituent modules are simple, the digital implementation of the hierarchical controller offers smaller size, lower power and/or higher speed than the implementation of the non-hierarchical solutions. This has been illustrated with several examples. A design methodology that facilitates description and verification of the controllers has been detailed with an application example implemented on a FPGA.

REFERENCES

[1] W. P. M. H. Heemels, B. De Schutter, A. Bemporad, “Equivalence of hybrid dynamical models,” *Automatica*, vol. 37, no. 7, pp. 1085–1091, July 2001.
[2] A. Bemporad, M. Morari, V. Dua, E. N. Pistikopoulos, “The explicit linear quadratic regulator for constrained systems”, *Automatica*, vol. 38, no. 1, pp. 3-20, 2002.
[3] T. A. Johansen, and A. Grancharova, “Approximate explicit constrained linear model predictive control via orthogonal search tree”, *IEEE Trans. on Automatic Control*, 48(5), pp. 810-815, 2003.
[4] C. N. Jones, P. Grieder, S. Rakovic, “A logarithmic-time solution to the point location problem for parametric linear programming”, *Automatica*, vol. 42, no. 12, pp. 2215-2218, 2006.
[5] P. Tondel, T. A. Johansen, A. Bemporad, “Evaluation of piecewise affine control via binary search tree”, *Automatica*, vol. 39, pp. 743-749, 2003.
[6] C. Wen, X. Ma, B. E. Ydstie, “Analytical expression of explicit MPC solution via lattice piecewise-affine function”, *Automatica*, vol. 45, pp. 910-917, 2009.
[7] A. Oliveri, T. Poggi, M. Storace, “Circuit implementation of piecewise-affine functions based on a binary search tree”, in Proc. ECCTD 2009, Antalya, Turkey, Aug. 2009, pp. 145-148.
[8] P. Julián, A. Desages, O. Agamennoni, “High-level canonical piecewise linear representation using a simplicial partition”, *IEEE Trans. Circ. Syst. I*, 46, pp. 463–480, 1999.
[9] R. Rovatti, C. Fantuzzi, S. Simani, “High-speed DSP-based implementation of piecewise-affine and piecewise-quadratic fuzzy systems,” *Signal Processing*, 80, pp. 951–963, 2000.
[10] P. Echevarria, M. Martínez, J. Echanobe, I. del Campo, J. Tarela, “Digital hardware implementation of high dimensional fuzzy systems,” in *Applications of Fuzzy Sets Theory*, Springer, 2007, pp. 245–252.
[11] M. Storace, T. Poggi, “Digital architectures realizing piecewise-linear multi-variate functions: two FPGA implementations,” Int. J. Circ. Th. Appl., 37, 2009.
[12] T. Poggi, F. Comaschi, and M. Storace, “Digital circuit realization of piecewise affine functions with non-uniform resolution: theory and FPGA implementation,” *IEEE Trans. on Circuits and Systems II*, vol. 52, no. 2, pp. 131–135, 2010.
[13] Xfuzzy, http://www.imse-cnm.csic.es/Xfuzzy
[14] Hybrid Toolbox: http://www.dii.unisi.it/hybrid/toolbox
[15] F. J. Moreno-Velo, I. Baturone, A. Barriga, S. Sánchez-Solano, “Automatic tuning of complex fuzzy systems with Xfuzzy”, *Fuzzy Sets and Systems*, vol. 158, no. 18, pp. 2026-2038, 2007.
[16] A. Bemporad, A. Oliveri, T. Poggi, M. Storace, “Synthesis of stabilizing model predictive controllers via canonical piecewise affine approximations”, in Proc. IEEE Conf. on Decision and Control 2010, Atlanta, Georgia, USA, Dec. 2010.



(a) (b) (c) (d)

Fig. 10. (a)-(d) Closed-loop simulation results considering hardware details.