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

Fast explicit MPC with multiway trees

M. Mönnigmann * **and M. Kastsian** *

* *Automatic Control and Systems Theory, Ruhr-Universität Bochum, 44801 Bochum, Germany*

Abstract: We show that explicit model predictive control (EMPC) laws, or more generally continuous piecewise affine control (PWA) laws on polyhedra, can be represented by multiway trees with two important features: (i) Their height can be reduced arbitrarily by increasing their order m (i.e. the number of binary tree nodes hidden in each multiway node). (ii) A multiway node can be evaluated as fast as a binary node with a simple but massively concurrent (or "parallel") procedure for $m \gg 1$. As a result the control law evaluation can be evaluated considerably faster than with a binary tree. Furthermore, we show that a multiway tree representation of an EMPC control law can be derived from the corresponding binary tree representation with a simple algorithm. Finally, we demonstrate that programmable logic devices are ideally suited for an implementation. First tests show that EMPC control laws with several thousand hyperplanes can be evaluated in a few clock cycles on compact, low-cost, low-power programmable logic devices.

Keywords: predictive control, explicit, field programmable gate array, piecewise affine

1. INTRODUCTION

Model predictive control (MPC) has been acclaimed for its ability to handle constrained multivariable systems. MPC generally is computationally demanding, however. The number of potential applications of MPC has increased ever since it has been shown that MPC control laws can be computed explicitly for large classes of problems (Bemporad et al., 2002a,b). Once available, an EMPC control law can be evaluated in two simple steps: (1) searching for the polyhedron that contains the current state of the system, and (2) evaluating the affine function that applies on this polyhedron. While algorithmically simpler than online MPC, step (1) still turns out to be computationally demanding whenever the number of polyhedra is large.

Tøndel and Johansen (2002) and Tøndel et al. (2003) proposed to arrange the hyperplanes that define the polyhedra in a binary search tree. Since the height h of a height-balanced binary tree T with N_{int} internal nodes is bounded above by $h \leq 1.4404 \log_2(N_{\text{int}}+1)+1$ (Adelson-Velskii and Landis, 1962), the time needed to search the valid polyhedron grows logarithmically in the number of hyperplanes. Notice that the binary tree needs to be height-balanced for this bound to hold. Other tree-based approaches have been proposed for approximate EMPC (Johansen, 2004) and recently for the more general problem of PWA functions on regular grids (Oliveri et al., 2009; Storace and Poggi, 2009). These tree-based representations of EMPC control laws and PWA functions have successfully been used in implementations on programmable logic devices (Oliveri et al., 2009; Storace and Poggi, 2009; Johansen et al., 2007).

⋆ Funding by Innovationspool der Fakultät gratefully acknowledged. We thank S.M. Mross and D. Vey for help with implementing the toolchain.

The purpose of the present paper is twofold. We show that EMPC control laws, or more generally PWA functions on polytopes, can be represented by a multiway tree T_M that has a smaller height h_M than the binary tree. More specifically, $h_M \leq 1.4404 \log_{m+1}(N_{\text{int}}+1)+1$, where m is the order of the multiway tree, i.e. $(m+1)$ is the maximum number of children of a node in the new tree, which is assumed to be a natural power of 2 for simplicity (see Sect. 3). As a consequence of the smaller height, the search for the valid polyhedron requires fewer steps. Secondly, we show that the nodes of the multiway tree T_M can be evaluated as fast as a node of the binary tree T even for $m \gg 1$ with a simple, massively concurrent circuit. We claim that EMPC control laws with a dozen to several thousand hyperplanes can be evaluated in a few (< 10) clock cycles on compact, low-cost, low-power programmable logic devices. Since current field programmable gate arrays (FPGA) have clock frequencies of about 1GHz, EMPC on the nanosecond time scale will soon be feasible.

2. BACKGROUND AND NOTATION

We consider LTI systems of the form

$$\begin{aligned} x(t+1) &= Ax(t)+Bu(t), \\ y(t) &= Cx(t), \end{aligned} \tag{1}$$

where the constraints $y_{\min} \leq y(t) \leq y_{\max}$, $u_{\min} \leq u(t) \leq u_{\max}$ must be obeyed for $t \geq 0$. In Eq. 1 $x(t) \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$, $u(t) \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ and $y(t) \in \mathbb{R}^{n_y}$ are state, input and output variables, and $\mathcal{X}$ and $\mathcal{U}$ are polyhedra. Assume (A, B) is stabilizable, and assume we intend to regulate the system (1) to the origin.

MPC solves, at each sampling time instance t, the following optimization problem in $U = (u_{t|t}^T, \ldots, u_{t+N-1|t}^T)^T$.

$$\min_U \quad J(U, x(t)) \tag{2}$$

$$\begin{aligned} \text{s.t. } & x_{t+k+1|t} = Ax_{t+k|t} + Bu_{t+k|t},\ k \geq 0, \\ & y_{t+k|t} = Cx_{t+k|t},\ k \geq 0, \\ & x_{t|t} = x(t), \\ & u_{t+k|t} = Kx_{t+k|t}, h_u \leq k < h_y \\ & y_{\min} \leq y_{t+k|t} \leq y_{\max},\ k = 1, \ldots, h_c, \\ & u_{\min} \leq u_{t+k|t} \leq u_{\max},\ k = 1, \ldots, h_c, \end{aligned} \tag{3}$$

where

$$J(U, x(t)) = x_{t+h_y|t}^T P\, x_{t+h_y|t} + \sum_{k=0}^{h_y-1} \left(x_{t+k|t}^T Q\, x_{t+k|t} + u_{t+k}^T R\, u_{t+k} \right)$$

with $Q = Q^T \succeq 0$, $R = R^T \succ 0$, and $P = P^T \succeq 0$. Furthermore, K is a gain matrix, $x_{t+k|t}$ denotes the predicted state vector at time $t+k$, and h_y, $h_u \leq h_y$, and h_c are the output, input and constraint horizon, respectively.

Bemporad et al. (2002b) show that the solution of the MPC optimization problem (2), (3) has the following structure. There exist a finite number n_P of nonempty polytopes P_i of the form

$$P_i = \left\{ x \in \mathcal{X} \,\middle|\, a_{i_1}^T x + b_{i_1} \leq 0, a_{i_2}^T x + b_{i_2} \leq 0, \ldots \right\}, \tag{4}$$

where a finite number of at least $n_x + 1$ inequalities occur for each polytope P_i, and where this number depends on i. These polytopes have pairwise disjoint interiors, and their union is $\mathcal{X}$. We refer to equations of the form

$$h_i(x) = a_i^T x + b_i, \tag{5}$$

where $a_i \in \mathbb{R}^{n_x}$, $a_i \neq 0$, and $b_i \in \mathbb{R}$, as hyperplane equations, since $h_i(x) = 0$ defines a hyperplane in $\mathbb{R}^{n_x}$. The control law $u : \mathcal{X} \to \mathcal{U}$ that results from solving (2) and (3) is continuous and can be stated in the form

$$u(x) = \begin{cases} F_1 x + g_1 & \text{if } x \in P_1 \\ \vdots & \vdots \\ F_{n_P} x + g_{n_P} & \text{if } x \in P_{n_P}, \end{cases} \tag{6}$$

where $F_i \in \mathbb{R}^{n_u \times n_x}$ and $g_i \in \mathbb{R}^{n_u}$, $i = 1, \ldots, n_P$ (Bemporad et al., 2002a).

Polytopes as defined in Eq. (4) can conveniently be rewritten by using index sets (Tøndel and Johansen, 2002). Specifically, there exist index sets $I_1^-, \ldots, I_{n_P}^-$ with $I_j^- \subseteq \{1, \ldots, n_h\}$ and $I_1^+, \ldots, I_{n_P}^+$ with $I_j^+ \subseteq \{1, \ldots, n_h\}$ for all $j = 1, \ldots, n_P$ such that

$$P_i = \left\{ x \in \mathcal{X} \,\middle|\, h_j(x) \leq 0\, \forall j \in I_i^-, h_j(x) > 0\, \forall j \in I_i^+ \right\}, \tag{7}$$

where n_h denotes the number of hyperplanes. This representation suggests to evaluate each hyperplane equation only once for a given x, and to record the sign of the result. More precisely, let T be a height-balanced binary tree of the type shown in Fig. 1(a), where each internal node represented by a filled circle corresponds to a hyperplane $h_i(x)$. Define the mapping $\pi_T(x) : \mathcal{X} \to \{0,1\}^{n_h}$ by its components $\pi_{T,1}(x), \ldots, \pi_{T,n_h}(x)$ where

$$\pi_{T,i}(x) = \begin{cases} 1, \text{ if } a_i^T x + b_i \leq 0 \\ 0, \text{ otherwise.} \end{cases} \tag{8}$$

Then $x \in P_i$ if and only if

$$\pi_{T,j}(x) = \begin{cases} 1, \text{ if } j \in I_i^- \\ 0, \text{ if } j \in I_i^+ \end{cases} \text{ for all } j \in I_i^- \cup I_i^+. \tag{9}$$

Note that $\pi_{T,j}(x)$ is defined for all hyperplanes $j = 1, \ldots, n_h$, while Eq. (9) involves only $j \in I_i^- \cup I_i^+$, which is in general only a small subset of the indices of all hyperplanes.

3. MULTIWAY TREES FOR CONCURRENT EVALUATION OF EMPC LAWS

Consider a rooted binary tree T with N nodes. The length of the path from the root node to a node n_i is called the *depth* of the node n_i, where the depth of the root node is zero by convention. The *height* of the tree T refers to the largest depth that can be found in the tree. The set of all nodes of a binary tree that share the same depth d is called the dth level of the tree. A node without children is called *leaf* or *external node*. All other nodes are called *internal nodes*. By *internal height* of T we refer to the height of the tree $\tilde{T}$ that results by omitting all external nodes from T. A binary tree is called *full* if all nodes except for the leaf nodes have two children. A binary tree is called *perfect* if it is full and all leaves are on the same level. A binary tree is called *complete* if all its levels except for possibly its deepest one contain the maximum number of nodes and all nodes are as far left as possible. A binary tree is called *height-balanced* if, for any node, the heights of the left and right subtree of this node differ by at most one. A binary tree is called *leftist* if, for any node, the height of the left subtree at this node is not smaller than the height of the right subtree at this node.

As an extension of the binary tree we introduce a multiway tree with certain properties. A tree is called *tree of order* m if its root node has more than one and at most $(m+1)$ outgoing edges, all of its internal nodes except the root node have exactly one ingoing edge and at least one and up to $(m+1)$ outgoing edges, and its external nodes have exactly one ingoing and no outgoing edges. A multiway node of the form of an internal node is called an m-node for short. A tree of order m is called *height-balanced*, if, for any of its internal nodes, the heights of any two of its subtrees at this node differ by at most one. The notions of a *leaf*, an *internal node*, and an *external node* are understood as in the binary case. We refer to a height-balanced tree of order m as an *M-tree* for short.

An M-tree can be constructed from a binary tree by merging binary nodes into m-nodes. We describe how to construct an m-node in Alg. 1 and Def. 1. An M-tree can then be constructed according to Alg. 2.

Algorithm 1. Let T be a height-balanced full leftist binary tree that represents an EMPC control law, i.e. each internal node of T represents a hyperplane as defined in Eq. (5) and each external node of T represents a polytope as defined in Eq. (4). Assume that T has $N_{\text{int}} \geq 1$ internal nodes. Let $m \in \mathbb{N}$, $m > 0$ be arbitrary. Construct a subtree S of T by traversing across the internal nodes of T as in a breadth first search starting at the root node of T until $\min(m, N_{\text{int}})$ internal nodes of T have been visited. By $\lambda_1, \ldots, \lambda_k$, $k \leq m+1$ refer to the leaves of S, where the numbering is arbitrary.

Algorithm 1 requires the tree T to be leftist. This condition is merely added for convenience. Without proof we claim that for any binary tree T that fulfills the conditions of Alg. 2 there exists a leftist binary tree $\tilde{T}$ that represents the same EMPC control law as T. $\tilde{T}$ can be constructed

from T by inverting the signs of some of its hyperplanes and interchanging left and right children accordingly.

The subtree S that results from Algorithm 1 is absorbed into a new type of node.

Definition 1. (m-node and m-subtree) Let T and m be as in Alg. 1. Apply Alg. 1 to T. We call the resulting subtree S *m-subtree*. By *m-node* we refer to the node of order m that contains the internal nodes of S and that has as its outgoing edges the edges that lead to the nodes $\lambda_1, \ldots, \lambda_k$ that result from Alg. 1.

Note that by construction the internal nodes of S are internal nodes of T, while the leaves of S may be internal or external nodes of T. Algorithm 1 and Def. 1 are illustrated in Fig. 1.

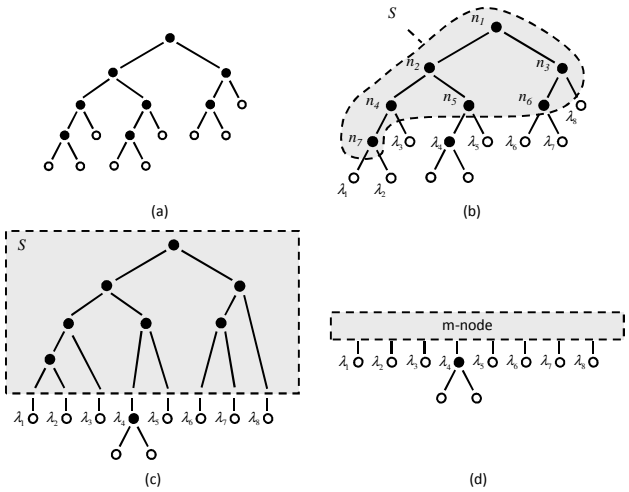

Fig. 1. (a) Height-balanced leftist binary tree. (b) Subtree S which defines an m-node for $m = 7$ acc. to Def. 1. (c) Subtree S defines an m-node. (d) Shorthand notation with m-node. Internal and external nodes are represented by full and open circles, respectively.

An m-node resp. m-subtree of a height-balanced leftist binary tree T need not be located at the root node of T. Since any subtree of T is a heigh-balanced leftist binary tree, Alg. 1 may be applied to any internal node of T. This is apparent from Fig. 2.

Based on the notion of an m-node we can now introduce a simple algorithm for the construction of an M-tree.

Algorithm 2. Let T and m be as in Alg. 1. The following steps provide an M-tree denoted by T_M that represents the same EMPC control law as T.

0 Let $\tilde{T} := T$.
1 Apply Alg. 1 to the root node of $\tilde{T}$. Replace the resulting subtree S of $\tilde{T}$ by the m-node that is defined by S according to Def. 1. Let $\lambda_1, \ldots, \lambda_k$ be as in Alg. 1.
3 For $\nu = \lambda_1, \ldots, \lambda_k$, carry out the following steps.
  3.1 If ν is an external node of T, skip it and continue with the next leaf at 3.
  3.2 Apply algorithm 2 to the subtree of $\tilde{T}$ that emanates at root node ν.
4 Set $T_M := \tilde{T}$ and return T_M.

Algorithm 2 and the M-tree T_M are illustrated in Fig. 2. We claim without proof that T_M and T represent the same control law. This is apparent, since the m-nodes of T_M merely absorb and hide some of the binary nodes of T.

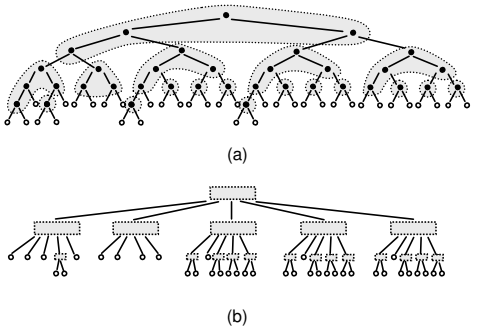

Fig. 2. (a) Example for a height-balanced leftist binary tree with m-nodes for $m = 4$. (b) M-tree that results for (a) and $m = 4$.

3.1 Bound on M-tree height

The height of the binary tree T and the height of the M-tree that results from applying Alg. 2 to T are related as stated in Prop. 1 and Cor. 1.

Proposition 1. (upper bound on height of M-tree) Let T be a height-balanced full binary tree that represents an EMPC control law, i.e. each internal node of T represents a hyperplane as defined in Eq. (5) and each external node of T represents a polytope as defined in Eq. (4). Let $m \in \mathbb{N}$, $m > 0$ be arbitrary. Assume that T has $N_{\text{int}} \geq 1$ internal nodes. By h_{int} denote the internal height of T. By T_M denote the M-tree that results from applying Alg. 2 to T. Then the height h_M of T_M is bounded above according to

$$h_M \leq \frac{h_{\text{int}}}{\text{floor}\left(\log_2\left(m+1\right)\right)} + 1 =: h_M^{\max}. \tag{10}$$

A sketch of the proof of Prop. 1 is given in the appendix.

Since the height h of any height-balanced binary tree with N nodes is bounded above according to Adelson-Velskii and Landis (1962) by

$$h \leq h_{\text{Fib}}(N) := \log_2\left(N+1\right) / \log_2\left(\frac{1+\sqrt{5}}{2}\right) \tag{11}$$

we have the following corollary, where we assume $m + 1 = 2^k$, for some $k \in \mathbb{N}$ for simplicity.

Corollary 1. Let the assumptions and the notation be as in Prop. 1. By $h_{\text{Fib}}(N)$ denote the height of the Fibonacci tree with N nodes as defined in Eq. (11). Assume $m + 1$ is a natural power of 2. Then

$$h_M \leq \frac{h_{\text{Fib}}(N_{\text{int}})}{\log_2\left(m+1\right)} + 1 \tag{12}$$

$$\leq 1.4404 \log_{m+1}\left(N_{\text{int}}+1\right) + 1. \tag{13}$$

In Eq. (13) we used $1/\log_2(\frac{1+\sqrt{5}}{2}) \leq 1.4404$.

The bound on h_M stated in Prop. 1 is tighter than the bound stated in Cor. 1. Corollary 1 is stated, however, because it makes the logarithmic dependency with a basis $m + 1$ explicit.

We note that the M-tree and its construction according to Alg. 2 are reminiscent of the B-tree and its variants

and their construction by bulk loading. It can be shown by example, however, that bulk loading does in general not result in a multiway tree that represents the same piecewise affine function as the original binary tree.

3.2 Fast evaluation of m-nodes

As anticipated at the beginning of Sect. 3 an m-node is the key to a speedup of the evaluation of the EMPC control law. The speedup is obtained by a *concurrent* (or "parallel") evaluation of the m binary nodes contained in each m-node. In order to introduce the concurrent approach to evaluating an M-tree T_M, we need to relate the paths from the root node of T_M to its leaves to the mapping π introduced in Sect. 2. This is done in the following definition.

Definition 2. (path p and mapping σ_S) Let T be an arbitrary full binary tree with $N_{\text{int}} \geq m$ internal nodes $n_1, \ldots, n_{N_{\text{int}}}$ and corresponding hyperplanes $h_1, \ldots, h_{N_{\text{int}}}$ of the form (5), where $m \in \mathbb{N}$, $m > 0$ is arbitrary. Let S be an m-subtree that results from applying Alg. 1 to T. Denote the nodes of S and the corresponding hyperplanes by $n_{i_1}, \ldots, n_{i_m}$ and $h_{i_1}, \ldots, h_{i_m}$, resp., and assume without restriction that n_{i_1} is the root of S. By $\lambda_1, \ldots, \lambda_{m+1}$ denote the leaves of S that result from Alg. 1. Identify any leaf λ_k of S with the unique path p from n_{i_1} to λ_k, or equivalently, with the mapping $\sigma_S(k)$ defined by its m components

$$\sigma_{S,j}(k) = \begin{cases} 1 \text{ if } n_{i_j} \text{ and its left child lie on } p \\ 0 \text{ if } n_{i_j} \text{ and its right child lie on } p \\ -1 \text{ otherwise,} \end{cases} \quad (14)$$

where $j = 1, \ldots, m$.

If $\sigma_{S,j}(k) = -1$ for a $j \in \{1, \ldots, m\}$, then node n_{i_j} and hyperplane h_{i_j} do not play a role for the path p from the the root n_{i_1} to the leaf λ_k. It is therefore sufficient to consider the subsequence of $\sigma_{S,1}, \ldots, \sigma_{S,m}$ that contains only those entries that are either 1 or 0, which can formally be defined as follows.

Definition 3. (condensed mapping $\bar{\sigma}$) Let the assumptions and σ_S be as in Def. 2. Let $k \in \{1, \ldots, m\}$ be arbitrary. By $j_1, \ldots, j_{\bar{m}}$ denote the largest subsequence of $1, \ldots, m$ such that $\sigma_{S,j_l}(k) \neq -1$ for all $l = 1, \ldots, \bar{m}$. Define $\bar{\sigma}_S(k)$ by its $\bar{m}$ components

$$\bar{\sigma}_{S,l}(k) = \sigma_{S,j_l}(k), \quad l = 1, \ldots, \bar{m}. \quad (15)$$

As discussed in Sect. 2 the binary search on T can be represented by $\pi_T(x)$ for any binary tree T that fulfills the conditions stated in Prop. 1. For the subtree S from Def. 2 this mapping $\pi_S(x)$ is defined on $\mathcal{X}$ by the components

$$\pi_{S,j}(x) = \begin{cases} 1 \text{ if } h_{i_j}(x) \leq 0 \\ 0 \text{ if } h_{i_j}(x) > 0, \end{cases} \quad j = 1, \ldots, m. \quad (16)$$

In analogy to $\bar{\sigma}_S$ we define $\bar{\pi}_S$ by its $\bar{m}$ components

$$\bar{\pi}_{S,l}(x) = \pi_{S,j_l}(x), \quad l = 1, \ldots, \bar{m}, \quad (17)$$

where $j_1, \ldots, j_{\bar{m}}$ and $\bar{m}$ are as in Def. 2. Then carrying out a binary search for $x \in \mathcal{X}$ on S results in the leaf λ_k if and only if, for all $j = 1, \ldots, m$ with $\sigma_{S,j}(k) \neq -1$,

$$\sigma_{S,j}(k) = \pi_{S,i_j}(x), \quad (18)$$

or equivalently, if, for all $l = 1, \ldots, \bar{m}$,

$$\bar{\sigma}_{S,l}(k) = \bar{\pi}_{S,l}(x). \quad (19)$$

Since the l.h.s. of Eqs. (18) and (19) are independent of x, these terms can be evaluated *before* the runtime of the EMPC controller. On the other hand, the components of π_{S,i_j} that constitute the r.h.s. of Eqs. (18) and (19) are independent of one another. Therefore, the outcome of the comparison in Eq. (18) or Eq. (19) does not change if the m hyperplane equations are evaluated simultaneously instead of sequentially. This suggests the following algorithm for the concurrent (or "parallel") evaluation of an m-node.

Algorithm 3. Let S be an m-subtree of an M-tree T_M. Let the internal nodes $n_{i_1}, \ldots, n_{i_m}$ of S with corresponding hyperplanes $h_{i_1}, \ldots, h_{i_m}$, the leaves λ_k, $k = 1, \ldots, m+1$, and the mapping σ_S be as in Def. 2. Furthermore, assume an $x \in \mathcal{X}$ is given. The following algorithm returns a λ_k such that a polytope P_l with $x \in P_l$ is a leaf of the subtree of T with root node λ_k.

1 For $j = 1, \ldots, m$ calculate $\pi_{S,j}(x)$, simultaneously for all j, and collect the result in the sequence of m bits $\Pi = (\pi_{S,1}(x), \ldots, \pi_{S,m}(x))$.
2 For $k = 1, \ldots, m+1$, carry out the following comparisons simultaneously for all k:
  2.1 Obtain the subsequence
  $$\bar{\sigma}_S(k) = (\bar{\sigma}_{S,1}(k), \ldots, \bar{\sigma}_{S,\bar{m}}(k)) \quad (20)$$
  from σ_S according to Def. 3 and the subsequence
  $$\bar{\pi}_S(x) = (\bar{\pi}_{S,1}(x), \ldots, \bar{\pi}_{S,\bar{m}}(x)) \quad (21)$$
  from $\pi_S(x)$ according to Eq. (17).
  2.2 If $\bar{\sigma}_S(k) = \bar{\pi}_S(x)$ return k.

The mappings σ_S, $\bar{\sigma}_S$, π_S, $\bar{\pi}_S$ and Alg. 3 can be defined accordingly if the requirement $N_{\text{int}} \geq m$ is dropped in Def. 2. We state only the case $N_{\text{int}} \geq m$ in order to avoid a too tedious notation.

4. IMPLEMENTATION AND EXAMPLES

Algorithm 3 can efficiently be carried out with the circuit sketched in Fig. 3. Consider Fig. 3(a) first. Each `MULT` block calculates one product $a_{i_j,l}\, x_l$, $l \in \{1, \ldots, n\}$, where $x \in \mathbb{R}^n$ is the current state space vector. These n independent multiplications can be carried out with n parallel `MULT` blocks as sketched in Fig. 3(a). Subsequently, the `SUM` block carries out $n-1$ additions to calculate $a_{i_j}^T x$, and the `LEQ` block compares $a_{i_j}^T x$ to b_{i_j} to give $\pi_{S,j}$ as defined in Eq. (16). The components of π_S can be calculated independently as indicated in Fig. 3(b), where the circuit from Fig. 3(a) is repeated m times. This concludes step 1 of Alg. 3. The `COMP` block in Fig. 3(b) implements step 2 of Alg. 3 as a bitwise comparison of the binary numbers $\bar{\pi}_S(k)$ and $\bar{\sigma}_S(k)$.

Two levels of concurrency (or "parallelism") are apparent from Fig. 3. For one, the multiplications necessary to evaluate a *single hyperplane* equation (5) can be carried out simultaneously. Secondly, *all m hyperplanes* of an m-node can be treated simultaneously. As a result, an m-node with $\bar{m}$ binary nodes and an m-node with $\hat{m} > \bar{m}$ nodes can be evaluated in the same period of time, as long as the hardware resources necessary to implement the $\hat{m}$-fold concurrency sketched in Fig. 3(b) are available. In particular this statement holds for $\bar{m} = 1$ and $\hat{m} > 1$, i.e. an m-node can be evaluated as fast as a binary node. It is stressed, however, that the number of necessary `MULT` blocks grows bilinearly in the concurrency m and the state space dimension n_x.

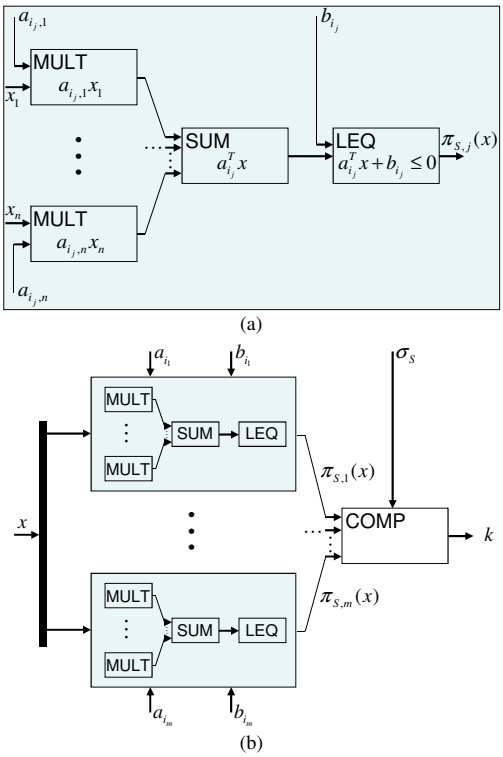

Fig. 3. Sketch of a simple circuit that implements Alg. 3. Diagram (a) details the shaded parts of diagram (b).

Examples

The circuit sketched in Fig. 3 can conveniently be implemented on a programmable logic device such as an field programmable gate array (FPGA) using a hardware description language, e.g. VHDL or Verilog. Many FPGAs provide dedicated circuit blocks for fast integer multiplications, which are ideally suited to implement fast `MULT` blocks, where we assume that real numbers are represented by an appropriate fixed point format. The `SUM` and `LEQ` block correspond to simple VHDL or Verilog language elements. The `COMP` block can be implemented as a comparison of the binary numbers $\bar{\pi}_S(k)$ and $\bar{\sigma}_S(k)$ with a bitwise `NXOR` operation.

We considered the following three systems, which we refer to as Example 1, 2, and 3, respectively. By I we denote the identity of the appropriate dimension.

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

where $Q = I$, $R = 1$, $-1 \leq u \leq 1$;

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 1 \\ 0.5 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

where $Q = I$, $R = 1$, $-1 \leq u \leq 1$; and

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}, C = I,$$

where $Q = I$, $R = 0.01 \cdot I$, $(-1, -1, -1, -1)^T \leq u \leq (1, 1, 1, 1)^T$. The first example is a double integrator system that is frequently considered in the EMPC literature. Examples 2 and 3 are not related to actual systems but have merely been chosen to generate EMPC control laws with a useful number of polytopes. In all calculations the prediction and control horizon are set equal and denoted by H.

Test results are summarized in Tab. 1. The symbols h_{int}, h_M, $h_M^{\max}$, n_{cyc}, φ, and e_{mr} denote the internal height of the binary tree that represents the EMPC law, the height of the M-tree, the upper bound on the height of the M-tree from Eq. (10), the maximum number of cycles on the FPGA needed to evaluate the EMPC control law with the M-tree, the fraction of the FPGA hardware resources (slices) used, and a measure of the error explained below. For each of the three examples we increased m until either $m+1$ would have exceeded the hardware resources of the FPGA (rows 8, 12, 17, 21), or until the tree height cannot be reduced any further, because an M-tree resulted that consisted of only a root node ($h_M = 0$, rows 4, 8, 26).

All circuits were defined in VHDL with the tool chain sketched in Fig. 4 and tested with 10^4 randomly generated points $x \in \mathcal{X}$. The error e_{mr} given in Tab. 1 for example

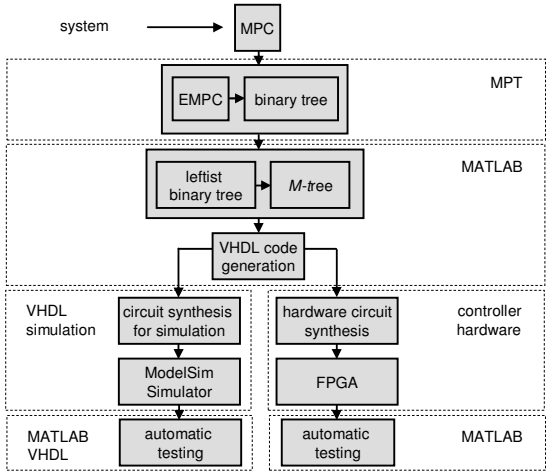

Fig. 4. Toolchain. Circuits are synthesized with Xilinx ISE Release 12.2 (NT). ModelSim refers to the ModelSim XE III 6.5c simulator.

1 was calculated by determining the absolute relative error for each random x between $u(x)$ as calculated by the circuit proposed here and the result provided by the MPT toolbox (Kvasnica et al., 2004) with single precision floating point numbers, and by averaging over all 10^4 absolute relative errors. Errors reported in Tab. 1 are within the expected rounding errors that arise due to the fixed point representation. For all reported results, the polyhedron determined by the FPGA implementation was equal to the polydron determined by matlab with the same fixed point representation.

The circuits for examples 2 and 3 have been tested in the ModelSim simulator. For 10^4 random points $x \in \mathcal{X}$ $u(x)$ as evaluated in ModelSim were equal to $u(x)$ evaluated with the MPT toolbox, where the same fixed point representation was used in ModelSim and matlab.

Our results demonstrate that an m-node can be evaluated in one clock cycle with the circuit architecture sketched in Fig. 3. Evaluating the r.h.s. of $u(x)$ as defined in Eq. 6 requires another clock cycle. Note the described timing is consistent with $n_{\text{cyc}} = h_M + 2$ in Tab. 1. While the FPGA used for testing here is not particularly fast

(Xilinx Spartan 3-E, 50MHz, i.e. clock cycle time 20ns), the control action $u(x)$ can be calculated in no more than 100ns for all the examples treated here with the respective highest level of concurrency m[1]. We stress that the number of cycles depends on details such as the availability of resources such as dedicated multipliers and memory on the particular hardware.

The examples corroborate the theoretical result on the tree height reduction and the resulting speedup in EMPC control law evaluation stated in Prop. 1.

Table 1. Summary of test results.[2]

#	H	n_P	n_h	h_{int}	m	h_M	$h_M^{\max}$	n_{cyc}	φ	e_{mr}
Example 1										
1	2	15	19	4	5	2	3	4	11%	0.39%
2	2	15	19	4	10	1	$2.\bar{3}$	3	14%	0.37%
4	2	15	19	4	20	0	2	2	25%	0.79%
5	3	19	27	4	5	2	3	4	12%	0.43%
6	3	19	27	4	10	1	$2.\bar{3}$	3	15%	0.46%
8	3	19	27	4	27	0	2	2	48%	0.50%
9	4	23	30	5	5	2	3.5	4	12%	0.33%
10	4	23	30	5	10	1	$2.\bar{6}$	3	15%	0.66%
12	4	23	30	5	27	1	2.25	3	92%	0.38%
13	5	25	32	5	5	2	3.5	4	12%	0.31%
14	5	25	32	5	10	1	$2.\bar{6}$	3	15%	0.41%
17	5	25	32	5	27	1	2.25	3	92%	0.28%
18	10	35	35	6	5	2	4	4	13%	0.37%
19	10	35	35	6	10	1	3	3	16%	0.66%
21	10	35	35	6	27	1	2.5	3	95%	0.23%
Example 2										
22	5	129	275	9	5	3	5.5	5	ModelSim	
25	5	129	275	9	100	1	2.5	3	ModelSim	
26	5	129	275	9	500	0	2.125	2	ModelSim	
Example 3										
27	5	1657	6667	14	6	5	8	7	ModelSim	
28	5	1657	6667	14	100	2	3.333	4	ModelSim	

5. CONCLUSIONS

We introduced a multiway tree for the representation of explicit solutions of MPC problems, and an algorithm for the construction of such a multiway tree from a EMPC law binary tree. The height of the proposed multiway tree of order m is of order $\log_{m+1}(n_h+1)$, where n_h denotes the number of hyperplanes in the EMPC control law.

The time needed to evaluate a multiway tree node was shown not to depend on m. Therefore, an m-node can be evaluated as fast as a binary node ($m = 1$), if computational resources suffice to evaluate $m > 1$ binary tree nodes simultaneously. We proposed a simple circuit for the concurrent evaluation of an m-node with $m > 1$, which can conveniently be implemented on a programmable logic device. The concurrency m is only limited by the resources that are available on the programmable logic device.

The multiway tree, the algorithm for its construction, and the circuit for its evaluation were successfully tested for values of m up to $m = 27$ in actual hardware circuits, and up to $m = 500$ in a commercial VHDL simulator. We note that the example size is currently only limited due to memory limitations on the 32bit operating system used for circuit synthesis. In all examples we were able to decrease the number of clock cycles needed to evaluate the EMPC control law by increasing m as anticipated theoretically. For the particular systems treated here, for example, this upper bound was $n_{\text{cyc}} \leq 5$. At a typical clock frequency of 500MHz of a current FPGA, this corresponds to an upper bound of 10ns on the EMPC control law evaluation time for the example size treated here.

The circuits used here are considerably simpler than even the simplest microprocessor. It is therefore reasonable to expect that a much higher concurrency can be achieved with a programmable logic device than with a microprocessor, if the same hardware technology is used for both. Clearly, the low power consumption of programmable logic devices and their low cost are additional benefits.

[1] At time of submission (10/2010), the fastest FPGA available were > 10 times faster and provided a larger number of MULT blocks than the one used here.

[2] all results obtained with Xilinx Spartan 3-E Speed Grade-4 (xc3s500e-4fg320), 50MHz, 18bit fixed point, fraction length 10

6. APPENDIX

We give a sketch of the proof of Prop. 1. An m-node is called *full* if it contains m binary nodes. It is called *terminal* if none of its children are m-nodes and called *nonterminal* otherwise. Without restriction we assume that T is leftist; see the comment below Alg. 1. It can be shown that every nonterminal node of an M-tree that results from Alg. 2 is full by construction. Furthermore, it can be shown that for any m-subtree S that corresponds to a nonterminal m-node that results from applying Alg. 2 to T, the length l of any path from the root node of S to any of its leaves that is an internal node of T is bounded according to

$$l \geq \text{floor}(\log_2(m+1)). \tag{22}$$

For a path p from the root node of T to any of its leaves, there exists a sequence of m-nodes $q_1, \ldots, q_k, q_{k+1}$ through T_M that p passes through. Since each of them has at least one child that is an m-node, the nodes $q_1, \ldots, q_k$ are nonterminal, thus full, consequently (22) applies. This implies $l_p \geq k\ (\text{floor}(\log_2(m+1))) + 1$, for the length l_p of p where the trailing 1 accounts for q_{k+1}, which contains at least one internal node of T by construction. Substituting $l_p \leq h_{\text{int}}$ and solving for k results in $k+1 \leq h_M^{\max}$. Since the path p is arbitrary, this proves the desired result.

REFERENCES

Adelson-Velskii, G. and Landis, E.M. (1962). An algorithm for the organization of information. *Soviet Math. Doklady*, 3, 1259-1263.

Bemporad, A., Borrelli, F., and Morari, M. (2002a). Model predictive control based on linear programming – the explicit solution. *IEEE Trans. Autom. Contr.*, 47(12), 1974–1985.

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

Johansen, T. (2004). Approximate explicit receding horizon control of constrained nonlinear systems. *Automatica*, 40, 293–300.

Johansen, T.A., Jackson, W., Schreiber, R., and Tøndel, P. (2007). Hardware synthesis of explicit model predictive controllers. *IEEE Trans. Contr. Sys. Tech.*, 15(1), 191–197.

Kvasnica, M., Grieder, P., and Baotić, M. (2004). Multi-Parametric Toolbox (MPT). URL `http://control.ee.ethz.ch/ mpt/`.

Oliveri, A., Oliveri, A., Poggi, T., and Storace, M. (2009). Circuit implementation of piecewise-affine functions based on a binary search tree. In *2009 European Conference on Circuit Theory and Design, Vols 1 and 2*, 145–148. IEEE.

Storace, M. and Poggi, T. (2009). Digital architectures realizing piecewise-linear multivariate functions: Two FPGA implementations. *Int. J. Circ. Theor. Appl.* In print.

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

Tøndel, P. and Johansen, T.A. (2002). Complexity reduction in explicit linear model predictive control. In *Proc. of 15-th IFAC World Congress*. Barcelona, Spain.