Embedding Approximate Nonlinear Model Predictive Control at Ultrahigh Speed and Extremely Low Power

Arnab Raha, Ankush Chakrabarty, Vijay Raghunathan, and Gregery T. Buzzard

***Abstract*— Embedded systems require control algorithms that are safe and able to operate in embedded platforms with extreme limitations on energy, memory, and area footprint. Nonlinear model predictive control (NMPC) algorithms respect operational constraints to ensure safety but are typically challenging to implement on resource-constrained embedded systems at high speeds. This brief introduces a formalism for deploying an approximate NMPC control law on severely resource-constrained hardware by systematically leveraging approximate computing tools. The resulting field-programmable gate array (FPGA) implementation operates at extremely low power, is ultrafast, requires very small on-chip area, and consumes lower memory than cutting-edge implementations of embedded NMPC for systems of similar state-space dimension. Feasibility and stability guarantees are provided for the embedded controller by preemptively bounding the allowable approximation error in the hardware design phase. An FPGA-in-the-loop implementation exhibits speeds in nanosecond range with power consumption in <1 mW for 2-D and 3-D nonlinear systems.**

***Index Terms*— Approximate computing, embedded systems, field-programmable gate array (FPGA), finite-precision, internet-of-things (IoT), model predictive control (MPC), real-time systems.**

I. Introduction

APPLICATIONS for pervasive computing platforms such as the emerging Internet-of-Things (IoT) require high-speed real-time implementable control algorithms that operate with consistent reliability and can be deployed from low-power embedded systems. In the presence of practical constraints and nonlinear dynamics, model predictive control (MPC) makes for a strong contender in various fields, including biomedical, automotive, and factory automation systems [1], [2].

High-speed and low-power/memory implementations of nonlinear MPC (NMPC) are a relatively unexplored but critical problem. Since the underlying optimal control problem in NMPC is typically not convex, local solutions via Newton-type optimization algorithms such as interior-point method (IPM) and sequential quadratic programming (SQP) are effective [3], [4]. One of the initial Newton-type algorithms that enabled real-time NMPC was described in [5], which was extended from an SQP to an IPM formalism using continuation methods in [6]. The real-time iteration scheme and its variants [7]–[9] are widely used SQP-type online algorithms that exploit the fact that NMPC requires the solution of similar optimal control problems at each iteration to enable very fast online implementation. Other numerically efficient methods for embedded NMPC deployment can be found in the literature. For example, Liniger *et al.* [10] propose an iterative linearization-based NMPC implementation and solve the corresponding quadratic programs in an embedded platform, reporting computational speeds of milliseconds. An investigation into the tractability of Nesterov's fast gradient in embedded MPC with successive linearization of the nonlinear model is made in [11] with complexity studies. Other implementations of fast-gradient methods in NMPC are found in [12] and [13]. Splitting methods with complexity certificates are provided in [14] and [15]. Modifications and approximations of SQP approaches can be found in [16] and [17] and careful removal of constraints to enable unconstrained minimization is proposed for field-programmable gate array (FPGA) implementation in [18]. This latter paper, as well as [19], considers optimization algorithms such as fast gradients or particle swarms that can be parallelized, therefore being amenable to distributed implementation. In spite of these advances, not only the maintaining feasibility is an extremely challenging problem [20], even the fastest implementations cannot go below microseconds in spite of employing highly sophisticated tools [21] (see a comparative study made in [22] on a 2-D nonlinear model).

To the best of authour knowledge, very few embedded NMPC implementations have exhibited the potential for operations under severe resource constraints, e.g., requirement of ultrafast (in the order of nanoseconds) control updates or very low-memory implementation. Notable exceptions include [23]–[25], where the authors use function approximators to incorporate speed and scalability. For example, Ayala *et al.* [23] demonstrated speeds in tens of microseconds using radial basis function networks, while Summers *et al.* [24] and Chakrabarty *et al.* [25] use ideas from interpolation and regression (respectively) to eliminate the computational complexity involved in successive linearization. These latter works theorize controller updates in the order of nanoseconds with current on-chip technology, but neither validate this claim on hardware.

Approximate computing is an emerging design paradigm that leverages the inherent error tolerance of error-resilient

Manuscript received December 5, 2018; accepted February 5, 2019. Date of publication March 1, 2019; date of current version April 13, 2020. Manuscript received in final form February 6, 2019. Recommended by Associate Editor L. Fagiano. The work of G. T. Buzzard was supported in part by NSF under Award CCF-1763896. *(Corresponding author: Ankush Chakrabarty.)*

A. Raha is with the Microarchitecture Research Laboratory, Intel Laboratories, Santa Clara, CA 95054 USA (e-mail: araha@purdue.edu).

A. Chakrabarty is with the Mitsubishi Electric Research Laboratories, Cambridge, MA 02139 USA (e-mail: chakrabarty@merl.com).

V. Raghunathan is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: vr@purdue.edu).

G. T. Buzzard is with the Department of Mathematics, Purdue University, West Lafayette, IN 47907 USA (e-mail: buzzard@purdue.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCST.2019.2898835

applications to improve performance on-chip [26], [27]. In this brief, approximate computing is used to deploy a sampling-based explicit nonlinear model predictive control (ENMPC) [25] for fast, low-energy implementation to demonstrate the potential of this approach in resource-constrained embedded systems via an FPGA-in-the-loop (FIL) verification.

This brief is an extension of preliminary results presented in [28]. Herein, we make the following additional contributions: 1) we introduce a simple design procedure using approximate computing that allows embedding an NMPC under severe resource limitations; a detailed description of the steps involved in the design is presented in Section III; 2) we demonstrate our method's potential via experimental validation using an FIL in Section V; 3) we investigate the effect of regularization and basis function selection on controller performance in Section V-A3; and 4) we study theoretical properties of the proposed approximate NMPC in Section IV.

II. Sampling-Based NMPC

We consider systems of the form

$$x^+ = f(x,u) \tag{1}$$

where $x \in \mathbb{X} \subset \mathbb{R}^{n_x}$ denotes the state of the system, $u \in \mathbb{U} \subset \mathbb{R}^{n_u}$ denotes the control action, f is a continuously differentiable nonlinearity that models the system dynamics, and x^+ represents the state update (that is, $\dot{x}$ for continuous-time systems, and x_{k+1} for discrete-time systems). Standard assumptions such as $f(0,0) = 0$ and compactness, convexity of $\mathbb{X}$ and $\mathbb{U}$ are made (see [29]). To expedite embedded implementation and without loss, we assume as in [14] that the set $\mathbb{U}$ contains the origin in its interior and can be represented as a set of box constraints.

A brief overview of the sample-driven NMPC design methodology proposed in [25] is provided here: the objective of this brief is to implement this sample-driven NMPC control law on an embedded system at high-speed while consuming very low power and requiring low area footprint. We assume that an inner approximation $\mathcal{F}$ of the feasible region for the NMPC is known, and begin by extracting N samples within $\mathcal{F}$. At each sample $x_i \in \mathcal{F}$, we solve the following finite-horizon optimal control problem

$$\begin{aligned} u^\star(x_i) &\\ = \arg\min_{u\in\mathbb{U}}\ & V(x,u)\\ \text{s.t. } & x(\tau)\in\mathbb{X},\quad u(\tau)\in\mathbb{U},\ \forall\,\tau\in[0,T_f]\\ & \dot{x} = f(x,u),\quad x(0)=x_i,\ x(T_f)\in\mathbb{X}_T,\\ & V(x,u) = V_T(x(T_f)) + \int_0^{T_f}\ell(x(\tau),u(\tau))\,d\tau. \end{aligned} \tag{2}$$

Here, the functions V_T and ℓ are the terminal penalty and stage cost, respectively, the positive scalar T_f is the predictive horizon of the controller, and $\mathbb{X}_T$ is a predetermined terminal set. The solution of (2) yields a feasible NMPC control law

$$u^\star(x_i) = \left[u_1^\star(x_i)\cdots u_j^\star(x_i)\cdots u_{n_u}^\star(x_i)\right]$$

where $u_j^\star(x_i)$ represents the jth component of the optimal NMPC control law with x_i as the initial state.

Using the control actions obtained at each sample within $\mathcal{F}$, we can construct an NMPC control law over $\mathcal{F}$ using function approximators. Concretely, for the jth component of the control vector, let $\{\phi_{jr}\}_{r=1}^{M}$ denote user-defined polynomial basis functions used for performing regression; M denotes the number of basis functions used for the controller approximation. Let

$$\mathcal{X}_j := \begin{bmatrix} \phi_{j1}(x_1) & \dots & \phi_{jM}(x_1)\\ \vdots & \ddots & \vdots\\ \phi_{j1}(x_N) & \dots & \phi_{jM}(x_N)\end{bmatrix} \text{ and } \mathcal{U}_j := \begin{bmatrix} u_j^\star(x_1)[0]\\ \vdots\\ u_j^\star(x_N)[0]\end{bmatrix}$$

where $u_j^\star(x_1)[0]$ denotes the first element of the control sequence $u_j^\star(x_1)$. Using these matrices, we pose the control law approximation as the (regularized) regression problem

$$\textbf{Basic:}\quad \min_{\|\bar{\alpha}_j\|_\infty\le a} \|\mathcal{U}_j - \mathcal{X}_j\bar{\alpha}_j\| \tag{3a}$$

$$\textbf{Ridge:}\quad \min \|\mathcal{U}_j - \mathcal{X}_j\bar{\alpha}_j\| + \lambda_j\|\bar{\alpha}_j\|_2^2 \tag{3b}$$

$$\textbf{Lasso:}\quad \min \|\mathcal{U}_j - \mathcal{X}_j\bar{\alpha}_j\| + \lambda_j\|\bar{\alpha}_j\|_1 \tag{3c}$$

where $\bar{\alpha}_j$ is a vector of regression coefficients with element α_{jk} with $1\le k\le M$, $\lambda_j > 0$ is a user-defined regularization parameter, and the tuning parameter $a > 0$ ensures that the regression coefficients in the unregularized formulation are bounded. Thus, the jth ENMPC control law is

$$\hat{u}_j(x) = \Gamma\left(\sum_{r=1}^{M}\alpha_{jr}\phi_{jr}(x)\right) \tag{4}$$

where Γ is a projection operator to the set $\mathbb{U}$ and $j = 1,2,\dots,n_u$. Note that we sample $\mathcal{F}$ and solve (2)–(3) offline. For online implementation, we use the state x to compute a control action using (4), which usually requires few computations even if the state-space dimension is large (see [25]).

Although the method in [25] is designed to stabilize continuous-time systems, it has also been modified for the tracking problem in discrete-time systems [30].

III. Embedded Design via Approximate Computing

The controller designed in Section II may require a large number of basis functions (i.e., $M \gg 1$) in order to get a good approximation of the control law. Assuming that the ENMPC designed using (3) is fixed (i.e., its basis representation cannot be altered), we describe approximate computing methods in this section that optimize hardware for acceleration on a resource-constrained embedded system by sharing common basis units [factor sharing (FS)] to reduce arithmetical operations and by modifying the bit-precision of regression coefficients while bounding the allowable approximation error (precision scaling).

A high-level schematic of the application-specified integrated circuit (ASIC) architecture when $n_u = 1$ is shown in Fig. 1. Given an input state x, the controller evaluates the basis functions $\{\phi_r(x_k)\}$ at the register transfer level (RTL). Subsequently, the evaluated basis functions are multiplied with the corresponding coefficients $\{\alpha_r\}$, and the generated products

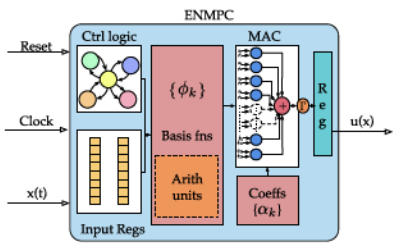

Fig. 1. Schematic of the ASIC architecture with $n_u = 1$. Arith units = Arithmetic Units. Regs = Registers. MAC = Multiplier-Accumulator.

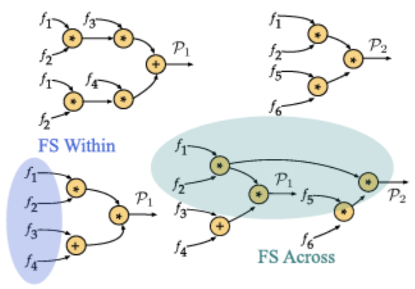

Fig. 2. Efficient hardware implementation using FS. P_1 naive implementation (top). FS within a polynomial (bottom left). FS across two polynomials (bottom right).

are all added together in the multiplier-accumulator (MAC) unit. The MAC unit also contains a combination of a logic gate and a multiplexer for implementing the projection operator Γ. This is possible because the admissible control set $\mathbb{U}$ is a product of intervals, hence, the projection operator (Γ) maps the final output $\hat{u}(x)$ elementwise to $\mathbb{U}$.

A. Embedding Multivariate Polynomials via Factor Sharing

FS is an efficient design method for reducing the complexity of the overall hardware when evaluating polynomials on a chip. The premise of FS is to prevent recomputing of common basis functions, monomials, or combination of monomials (referred to herein as "factors"), by sharing the arithmetic units on the embedded system dedicated to evaluating those factors. This leads to a reduction in the overall chip area and energy footprint while enhancing the computation speed. Furthermore, FS enables an efficient evaluation of common basis functions across multiple polynomials; this becomes critical when the number of control actions, n_u, is large.

We consider arithmetic operators that operate on two operands at a time, which is common in hardware implementation. Suppose the two polynomials to be evaluated are given by $\mathcal{P}_1 = f_1\ f_2\ f_3 + f_1\ f_2\ f_4$ and $\mathcal{P}_2 = f_1\ f_2\ f_5\ f_6$, where $f_1, f_2, \ldots, f_6$ are factors. Sharing factors within a polynomial can reduce the total number of arithmetic operators (four multipliers and one adder to two multipliers and one adder), decreasing the area as well as the power consumption of the ASIC. Furthermore, although the implementation of $\mathcal{P}_2$ in "FS Within" shown in Fig. 2 seems efficient, since both $\mathcal{P}_1$ and $\mathcal{P}_2$ are to be evaluated, sharing the common factor $f_1 f_2$ results in a reduction of a multiplier unit, as demonstrated in "FS Across" in Fig. 2.

B. Hardware Optimization Using Precision Scaling

Precision scaling or bit-width reduction of input and intermediate operands has proven to be an efficient technique for tradingoff quality for energy [27], [31]. Contrary to FS, precision scaling not only leads to additional area and power savings but also reduces the latency of operation to a significant extent.

To accelerate the ENMPC, precision scaling involves purposefully introducing roundoff errors in the coefficients $\{\alpha_{jr}\}_{r=1}^{M}$ of the each of the n_u polynomials, as well as basis function evaluation steps, in order to lower hardware complexity. The notion of average pointwise error (APE) is the cornerstone of investigating the tradeoff in complexity reduction versus controller law approximation error. For the jth polynomial ENMPC control law (4), the APE is estimated by extracting N_τ low-discrepancy samples within $\mathcal{F}$ and computing an error metric between the exact ENMPC $\hat{u}_j(x)$ and the embedded ENMPC (by "embedded ENMPC," we refer to the approximate precision scaled ENMPC), denoted $\hat{u}_j^{\text{test}}$. Mathematically, the APE is written as

$$\varepsilon_{\text{APE}} = \frac{1}{N_\tau}\sum_{k=1}^{N_\tau}\left(\hat{u}_j(x_r) - \hat{u}_j^{\text{test}}(x_r)\right)^2 \tag{5}$$

where $r = 1, 2, \ldots, N_\tau$.

Remark 1: Although we consider approximate control laws with basis functions as tensored polynomials for which we can provide control-theoretic guarantees, the approximation computing methods described in this section generalize to popular approximators such as neural networks (see [32]).

IV. Performance Guarantees

In this section, conditions are provided that if satisfied during hardware design, guarantee feasibility, and stability for the closed loop under the embedded control law $\Gamma(\hat{u}^{\text{test}})$. Concretely, these conditions are: 1) the N samples drawn for controller construction are extracted from a low-discrepancy sequence (see [33] and [34] for the definition and implementation details of low-discrepancy sequences); 2) the regression polynomials are tensored Chebyshev polynomials; and 3) the N_τ samples used to compute the average precision error are drawn from a low-discrepancy sequence.

We add two definitions to facilitate the ensuing discussion.

Definition 2: Let $\mathcal{F}$ be a topological space and $\mathfrak{T}(\mathcal{F})$ denote the open sets of $\mathcal{F}$. Elements of the σ-algebra generated by $\mathfrak{T}(\mathcal{F})$ that are subsets of $\mathcal{F}$ are called Borel subsets of $\mathcal{F}$

Definition 3 (c.f. [33]): The discrepancy of a sequence $\{x_j\}_{j=1}^{N} \subset \mathbb{X}$ is defined as

$$D_N\left(\{x_j\}_{j=1}^{N}\right) \triangleq \sup_{X\subset J}\left|\frac{\#X_N}{N} - \frac{\text{Vol}(X)}{\text{Vol}(\mathbb{X})}\right|$$

where J is a set of n-dimensional product intervals and $\#X_N \triangleq \text{card}\{j \in \{1,\ldots,N\} : x_j \in X\}$.

We begin with the following lemma.

Lemma 4: Let Ω be a Borel subset of the feasible set $[0, 1]^{n_x}$, and $\mathcal{P}$ be any polynomial with finite coefficients.

Let $\mathcal{P}_\Omega$ denote the restriction of $\mathcal{P}$ on Ω. Then there exists a scalar $\beta \in (0, \infty)$ such that

$$\left| \int_\Omega \mathcal{P}(x)\, dx - \frac{1}{N} \sum_{j=1}^{N} \mathcal{P}_\Omega(x_j) \right| \le \beta\, \mathcal{D}(\{x_j\}_{j=1}^N) \tag{6}$$

where $\mathcal{D}(\{x_j\}_{j=1}^N)$ is the discrepancy of the sequence $\{x_j\}$.

Proof: Since $\mathcal{P}$ is smooth, and $\mathcal{P}_\Omega$ is piecewise smooth, we know from [35] that the inequality (6) is satisfied with

$$\beta = \sum_{\alpha \in \{0,1\}^{n_x}} 2^{(d-|\alpha|)} \int_{[0,1]^{n_x}} \left\| \left(\frac{\partial}{\partial x}\right)^\alpha \mathcal{P}(x) \right\| dx. \tag{7}$$

Since $\mathcal{P}(x)$ has a finite number of coefficients, the integrand in (7) is bounded. Thus, β is a finite weighted sum of bounded integrals, which implies that $0 < \beta < \infty$. ■

The next theorem ensures that we can select an admissible ε_{APE} small enough and a number of low-discrepancy samples N_τ large enough such that the deviation between the ENMPC $\hat{u}_j$ and the embedded ENMPC $\hat{u}_j^{\text{test}}$ is as small as we desire. Since a maximal N_τ and minimal ε_{APE} can be chosen for all $j = 1, \ldots, n_u$, one can assume $n_u = 1$ without loss of generality.

Theorem 5: Let Ω denote a Borel subset in the interior of the feasible set $\mathcal{F}$, and $\{x_r\}$ is the low-discrepancy samples within Ω used to check that upon precision scaling, the embedded ENMPC satisfies (5). Then for each $\varepsilon > 0$, there exists $\varepsilon^\star > 0$ and $N_\tau^\star \in \mathbb{N}$ so that if $\varepsilon_{APE} < \varepsilon^\star$ and $N_\tau \ge N_\tau^\star$, then $\|\hat{u} - \hat{u}^{\text{test}}\|_{\mathcal{L}_\infty(\Omega)} \le \varepsilon$ for any $N_\tau \ge N_\tau^\star$.

Proof: Note that $\phi_1, \ldots, \phi_M$ is a basis for the polynomial $\mathcal{P} := \hat{u} - \hat{u}^{\text{test}}$, since both these polynomials have the same basis. By Gram–Schmidt orthogonalization, one can convert $\phi_1, \ldots, \phi_M$ to a basis $\zeta_1, \ldots, \zeta_M$ that is orthonormal with respect to the inner product $\langle g, h\rangle_\Omega \triangleq \int_\Omega g(x)h(x)dx$. Since each ζ_i is a polynomial, it is bounded on Ω, i.e., $\|\zeta_i\|_{\mathcal{L}_\infty(\Omega)} \le \rho_i$. In the new basis, $\mathcal{P}$ has an expansion $\mathcal{P} = \sum_{i=1}^M c_i \zeta_i$, and using Parseval's identity, we obtain $\int_\Omega \mathcal{P}^2(x)\, dx = \sum_{i=1}^M c_i^2$. Using the definition of ρ_i and invoking the triangle and Cauchy–Schwarz inequalities on the expansion of $\mathcal{P}$, the following result is obtained:

$$\|\mathcal{P}\|_{\Omega,\infty} \le \alpha \left(\textstyle\int_\Omega \mathcal{P}^2(x)\, dx\right)^{\frac{1}{2}} \tag{8}$$

where $\alpha \triangleq (\sum_{i=1}^M \rho_i^2)^{\frac{1}{2}}$ depends only on the basis $\zeta_1, \ldots, \zeta_M$; thus, only on $\phi_1, \ldots, \phi_M$ and Ω.

Note that $\mathcal{P}^2$ is a polynomial with finite coefficients (by design). Furthermore, even though $\mathcal{F}$ is not the unit hypercube $[0, 1]^{n_x}$, it can be bounded by a box in $\mathcal{B} \subset \mathbb{R}^{n_x}$, where $\mathcal{P}^2$ is defined to be zero in $\mathcal{B} \setminus \mathcal{F}$. Without loss of generality, the box $\mathcal{B}$ can be transformed into the unit hypercube. Thus, the preconditions for Lemma 4 are satisfied. We now combine (8) with the inequality (5), (6) to obtain $\|\mathcal{P}\|_{\mathcal{L}_\infty(\Omega)} \le \alpha/N_\tau|\varepsilon_{\text{APE}}| + \beta\mathcal{D}(\{x_r\})$, where β is a finite scalar and $\varepsilon_{\text{APE}} := \sum_{r=1}^{N_\tau} \mathcal{P}_\Omega^2(x_r)$. Since $\varepsilon_{\text{APE}} < \varepsilon^\star$, this implies

$$\|\mathcal{P}\|_{\mathcal{L}_\infty(\Omega)} \le \alpha\, \varepsilon^\star + \beta\, \mathcal{D}(\{x_r\}). \tag{9}$$

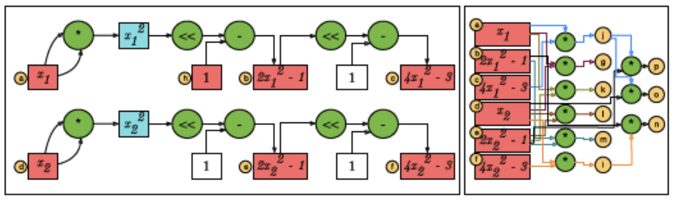

Fig. 3. FS in implementation of polynomials. Alphabets (*a–p*) within the smaller circles represent factors. The doubling of leading coefficients in the basis polynomials is exploited for swift implementation using 8-bit exponent adders/bit-shift operators (denoted by $\ll$).

By construction, the samples $\{x_r\}_{r=1}^{N_\tau}$ are drawn from a low-discrepancy sequence. Thus, there exists an $N_\tau^\star$ sufficiently large such that for any $N_\tau \ge N_\tau^\star$, we get $\mathcal{D}(\{x_r\}) \le \varepsilon/2\beta$. For the given ε, one can then select $\varepsilon^\star \le \varepsilon/2\alpha$. Thus, the inequality (9) yields $\|\hat{u} - \hat{u}^{\text{test}}\|_{\mathcal{L}_\infty(\Omega)} \le \alpha(\varepsilon/2\alpha) + \beta(\varepsilon/2\beta) = \varepsilon$. ■

This important result guarantees closed-loop feasibility and stability for the nonlinear system under embedded ENMPC control. This is stated in the following theorem. The set Ω and samples $\{x_r\}_{r=1}^{N_\tau}$ are the same as in Theorem 5.

Theorem 6: There exist positive scalars $N^\star, M^\star, N_\tau^\star$, and $\varepsilon^\star$ such that if $M \ge M^\star$, $N \ge N^\star$, $N_\tau \ge N_\tau^\star$, and $\varepsilon_{\text{APE}} < \varepsilon^\star$, then the closed-loop trajectories $x(t)$ of the nonlinear system (1) with control $u = \Gamma(\hat{u}^{\text{test}})$ and initial condition $x_0 \in \Omega$ satisfy the following conditions.

1) *Feasibility:* $x(t) \in \mathcal{F}$ for all $t \ge t_0$.
2) *Asymptotic Stability:* $\|x(t)\| \to 0$ as $t \to \infty$.

Proof: For a given $\varepsilon > 0$, Theorem 5 ensures that there exist scalars $\varepsilon^\star > 0$ and $N_\tau^\star$ such that if $\varepsilon_{\text{APE}} < \varepsilon^\star$ and $N_\tau \ge N_\tau^\star$, then $\|\hat{u} - \hat{u}^{\text{test}}\|_{\mathcal{L}_\infty(\Omega)} \le \varepsilon/2$. Furthermore, from [25, Th. 3], we know that there exist scalars $M \ge M^\star, N \ge N^\star$ that ensure $\|u - \hat{u}\|_{\mathcal{L}_\infty(\Omega)} \le \varepsilon/2$, where u_j denotes the actual NMPC. From the triangle inequality, it is clear that $\|u - \hat{u}^{\text{test}}\| \le \varepsilon/2 + \varepsilon/2 = \varepsilon$. The result of [25, Th. 4] can be directly applied to yield properties 1) and 2). ■

V. Hardware-in-the-Loop Experiments

We offer two examples to illustrate the potential of our proposed method. The first example is a two-state academic nonlinear system that has been studied in the literature; the rationale behind choosing this unstable system is to help visualize the design procedure step-by-step while illustrating feasibility and stability of the embedded ENMPC. We use knowledge gained from this academic example to test the proposed method on a more sophisticated model that has multiple control inputs and more involved nonlinearities. All experiments are performed with an FPGA in the control loop.

A. Example 1 (Illustrative Example)

We use the unstable nonlinear system

$$\dot{x} = \begin{bmatrix} -1 & 2 \\ -3 & 4 \end{bmatrix} x + \begin{bmatrix} 0.5 \\ -2 \end{bmatrix} u + \begin{bmatrix} 0 \\ -0.25x_2^3 \end{bmatrix}$$

described in [36] to test the efficacy of our embedded ENMPC implementation. The constraint sets are given by

$\mathbb{X} = \{x : \|x\|_\infty \leq 4\}$ and $\mathbb{U} = \{u : |u| \leq 2\}$; $N = 2000$ samples are drawn from a low-discrepancy Halton sequence and the NMPC control actions are obtained at these samples by solving (4) with $V_f = x^\top P x$ where $P = [\begin{smallmatrix} 2.3476 & 0.7360 \\ 0.7360 & 0.8296 \end{smallmatrix}]$, and $\ell(x, u) = x^\top Q x + u^\top R u$ with $Q = 10 I_2$, $R = 0.1$, and prediction horizon $T_f = 0.5$ s. An inner approximation of the feasible region is constructed using support vector machines, using the approach provided in [25]. Next, $M = 13$ Chebyshev polynomial basis functions are chosen to construct the ENMPC regression surface using (3a) with $a = 1000$. Although we see that M is not large, we use this as an example to demonstrate the design steps.

1) FPGA-in-the-Loop Implementation Details: The ENMPC controller is implemented at the RTL, synthesized using the Synposys Design Compiler, and mapped to the 45-nm-based open-cell Nangate technology library. For verification, the controller was programmed on an Altera Cyclone IV FPGA-based DE2-115 development board that is interfaced with Simulink using FIL which is contained in MATLAB's Hardware Description Language Verifier. Note that the FPGA is an emulator for an ASIC that will eventually implement the control law in hardware.

2) Modes Arising From Approximate Computing Tools: All inputs, intermediate scalars, and outputs use the IEEE-754 standard for floating point arithmetic. The baseline hardware implementations use 32-bit (1 sign, 8 exponent, 23 mantissa) single-precision floating point representation.

In the subsequent discussion, the hardware implementation of the ENMPC prior to approximation is referred to as Mode 0, denoted $\mathcal{M}_0$, and the mode induced by FS is denoted $\mathcal{M}_1$. Note that both $\mathcal{M}_0$ and $\mathcal{M}_1$ use the full 32-bit width, and therefore compute identical control actions.

The factors themselves are implemented with low-complexity arithmetic units. For example, we leverage the shift operator for multiplication with exponents of two for fixed-point representation. Note that the shift operator is just an internal manipulation of wire positions and almost completely eliminates the logic overhead required for implementing a full-fledged multiplier. Similarly, for floating point numbers, the resource usage of a floating-point multiplier can still be greatly reduced when multiplied using exponents of two. In this case, when a floating-point number is multiplied by exponents of two, the mantissa and sign (total 24 bits) of the result remain unchanged, whereas only the exponent of the number changes. The change in exponent can be obtained by just using an 8-bit adder only for the exponent part that has significantly/considerable lower resource usage compared to a full 32-bit multiplier. This enables us to eliminate area and power-hungry multipliers to a large extent. Leading coefficients being multiples of two offer a particular advantage to employing the Chebyshev polynomials from a hardware implementation perspective (see Fig. 3). Note that we do not perform any approximations to the control circuitry of the controller that is responsible for completing the application execution.

For precision scaling, an APE upper bound $\varepsilon^\star = 15 \times 10^{-3}$ is selected, computed over $N_\tau^\star = 10^4$ samples within $\mathcal{F}$. This introduces eight new modes $\mathcal{M}_2 - \mathcal{M}_9$ (see [28, Table I])

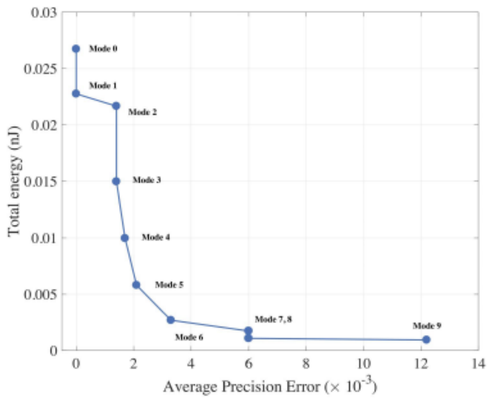

Fig. 4. Error-power tradeoff for various operating modes obtained FIL using Chebyshev polynomials with basic regression.

with varying degrees of approximation and varying output quality. Henceforth, the term "output quality" refers to the approximation accuracy of the embedded ENMPC control action compared to the exact ENMPC control action. Out of these eight approximation modes, the first six modes $\mathcal{M}_2$–$\mathcal{M}_7$ have lower operand precision with only the mantissa bit-width reduced. It is also possible to truncate the exponent; however, for this example, truncating the exponent results in a worse error-energy tradeoff since the error incurred is much more than mantissa truncation, while the energy savings are less (as there are only eight exponent bits compared to 23 mantissa bits). Gradually reducing bit-widths in the mantissa results in the last two approximation modes $\mathcal{M}_8$ and $\mathcal{M}_9$ where some of the coefficients reduce to zero. For example, in $\mathcal{M}_8$, the basis functions $\phi_9, \phi_{10}, \phi_{12}$, and ϕ_{13} are all eliminated. In $\mathcal{M}_9$, in addition to the previous basis functions, ϕ_7 and ϕ_{11} are also omitted. It is important to mention that we preserve multiple modes of approximation to demonstrate the tradeoff between computational efficacy and controller performance; in practice, only one mode (at the designer's discretion) would be decided upon and extensive formal verification on the selected mode would be required to validate the implementation on an ASIC.

As described previously, the effect of precision scaling on the controller implementation is twofold. It can result in a lower power (and lower area) implementation while producing acceptable outputs with reduced accuracy. This leads to the generation of a well-defined quality (or error) versus energy (or power) tradeoff (Q–E) plot as shown in Fig. 4. An essential trait in any approximate computing work is the Q–E tradeoff. This relationship is extremely useful to determine the appropriate operational mode for a particular quality/error bound. In practice, one would select the mode on this tradeoff curve closest to the application-specific needs. If a high accuracy is required with significant energy savings, a designer would likely select modes $\mathcal{M}_4$ or $\mathcal{M}_5$, whereas if the primary concern is energy, $\mathcal{M}_8$ would be the best choice. In case, the error bound $\varepsilon^\star$ is set to a different value, for example: lowered to 5×10^{-3}, then the designer can use this Q–E relationship to select $\mathcal{M}_5$.

Offline verification of the hardware approximation infrastructure involves a training phase and a validation phase [27], [37], [38]. In the training phase, we explore different configurations of the approximation knobs that will

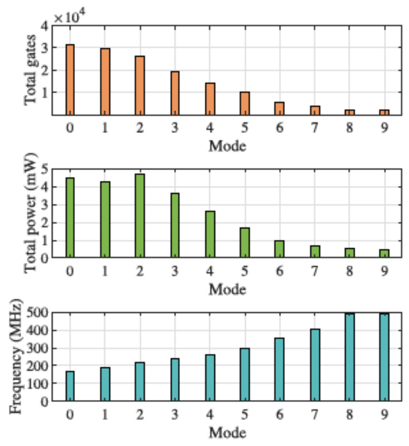

Fig. 5. Characteristics of different operating modes when Chebyshev-based ENMPC controller is synthesized using IBM 45-nm Nangate library.

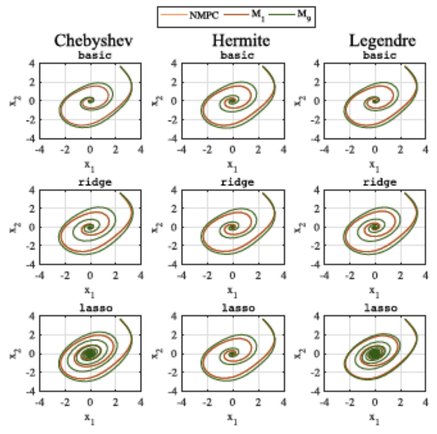

Fig. 6. Closed-loop state trajectories from a fixed-initial condition $[2.21, 3.83]^\top$ for multiple implementations of the embedded ENMPC.

satisfy different error bounds/constraints using representative samples. Once a satisfactory mode is fixed by the designer based on these samples, the validation phase involves computing control actions at samples outside the training set and ensuring that the APE bound is not violated.

3) Area, Power, Energy, and Frequency of Operation for Different Modes: Fig. 5 shows the total area, power, and frequency for different modes of operation for a Chebyshev polynomial-based ENMPC controller with basic regression described in (3a). Chebyshev polynomials are considered primarily because these basis functions possess theoretical properties that enable the proof of Theorem 6. The most interesting modes are the ENMPC $\mathcal{M}_0$, the factor shared ENMPC $\mathcal{M}_1$, and the three most aggressive approximation modes, $\mathcal{M}_{7,8,9}$. In comparison with the ENMPC implementation $\mathcal{M}_0$, the total area and power are shown to decrease by factors of $1.07\times$ and $1.06\times$, respectively, when using FS ($\mathcal{M}_1$). As expected, the total number of gates, which includes the combinational and noncombinational gates, decreases with an increase in the degree of approximation via precision scaling. For example, modes $\mathcal{M}_8$ and $\mathcal{M}_9$ have area reductions of $14.5\times$ and $17.1\times$, respectively, compared to the baseline ENMPC. Similarly, both the dynamic and leakage power (and therefore, total power) decrease for higher modes. For example, $\mathcal{M}_8$ and $\mathcal{M}_9$ reduce the total power by $8.4\times$ and $9.7\times$ compared to the baseline.

If the primary resource constraint is in terms of processor speed, the output latency value becomes critical. Note that approximations not only reduce the area and power consumption but also the overall processing time due to the lesser bit-width of different operands. To this end, the lowest plot in Fig. 5 also illustrates the maximum frequency of operation, which is the inverse of the latency incurred in computing a controller action via (4). The frequency varies in the range [165, 500] MHz for the ASIC implementations using Open Nangate technology. As expected, at higher modes of approximation, the circuits have smaller critical paths due to fewer logic components. This results in a higher frequency of operation. For example, compared to the baseline $\mathcal{M}_0$, modes $\mathcal{M}_8$ and $\mathcal{M}_9$ cause operational frequencies to increase by $2.94\times$. This implies that the controller update is in the order of nanoseconds.

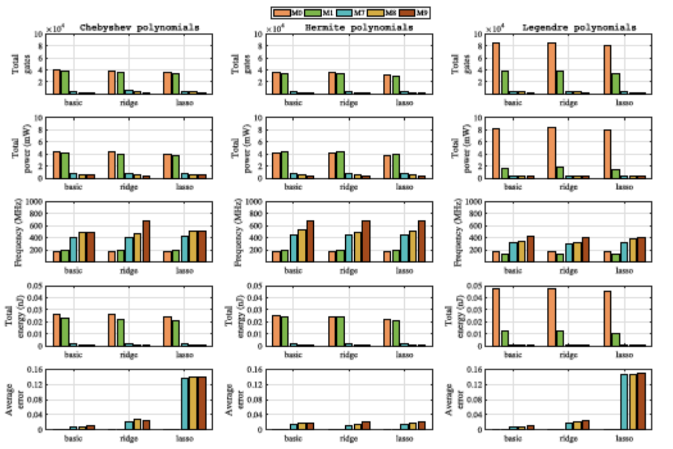

Fig. 7. Characteristics of various ENMPC controller implementations using different basis functions under different constraints.

To compare the effect of basis function selection and regularization of polynomial coefficients, nine ENMPC controllers are constructed with combinations made of three basis functions that exhibit good practical performance for continuous function approximation (Legendre, Hermite, and Chebyshev polynomials) and three regularized regression methods [described in (3)] with $\lambda = 1$. Fig. 6 shows the comparison of closed-loop state trajectories of an optimal NMPC against an ENMPC $\mathcal{M}_0/\mathcal{M}_1$ and the corresponding embedded ENMPC $\mathcal{M}_9$. Due to the increased aggressiveness of the approximation modes, it is expected that the closed-loop trajectory of mode $\mathcal{M}_1$ will be more similar to the NMPC than $\mathcal{M}_9$. Although each approximation mode ($\mathcal{M}_2$–$\mathcal{M}_9$) converged from the initial condition to the origin, only the worst approximation mode $\mathcal{M}_9$ is depicted for clarity. A detailed comparison is provided in Fig. 7, with exact numbers reported in Table I. With basic regression using (3a), the area reduction obtained for the most aggressive approximation mode $\mathcal{M}_9$ is roughly $17\times$, $40\times$, and $35\times$ for Chebyshev, Hermite, and Legendre polynomial-based implementations, respectively. Thus, the Hermite polynomial implementation results in the fastest operation, with frequencies as high as 676 MHz corresponding to mode $\mathcal{M}_9$. However, the lowest energy is required by mode $\mathcal{M}_9$ of the Legendre bases, which is

TABLE I

POWER, FREQUENCY, AND APE FOR $\mathcal{M}_0$, $\mathcal{M}_1$, AND $\mathcal{M}_9$ WITH DIFFERENT IMPLEMENTATIONS

Mode	**Cheby.** - basic			**Herm.** - basic			**Legend.** - basic			**Cheby.** - ridge			**Herm.** - ridge			**Legend.** - ridge			**Cheby.** - lasso			**Herm.** - lasso			**Legend.** - lasso		
Parameter	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_9$	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_9$	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_9$	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_9$	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_9$	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_9$	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_9$	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_9$	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_9$
Power (mW)	4.5	4.2	0.5	4.1	4.4	0.4	8.2	1.7	0.3	4.4	4.0	0.4	4.2	4.5	0.4	8.4	1.8	0.3	3.9	3.7	0.5	3.7	3.9	0.4	8.0	1.4	0.3
Freq. (MHz)	167	186	490	165	182	676	172	130	413	164	183	685	172	187	676	177	135	407	160	183	510	172	181	676	177	130	407
$\log_{10}$ APE	-6	-6	-2	-6	-6	-2	-6	-6	-6	-6	-6	-2	-6	-6	-6	-6	-6	-2	-6	-6	-1	-6	-6	-6	-6	-6	-1

approximately 0.7 pJ: about 72$\times$ lower than the baseline $\mathcal{M}_0$. For ridge regression as described in (3b), the maximum reduction in area and energy occurs for mode $\mathcal{M}_9$ of Hermite and Legendre polynomials, respectively. The number of gates is 939, the energy consumption is around 0.6 pJ, and the corresponding reduction factors are approximately 39$\times$ and 72$\times$, respectively. The maximum frequency of operation occurs for Chebyshev bases at 685 MHz. Similarly, for lasso regression, the maximum reduction in area and energy occurs for mode $\mathcal{M}_9$ of Hermite and Legendre polynomials, respectively. These reduction factors are approximately 35$\times$ and 72$\times$, respectively. The maximum frequency of operation occurs for the Hermite polynomial implementation at 676 MHz. As expected, the lasso regression results in a sparse coefficient vector that enables an implementation with fewer gates overall. It was found that using the Legendre polynomials as the set of basis functions results in higher power consumption and lower frequency values compared to Hermite and Chebyshev polynomials (see Table I). One reason for this is because it has many more factors that are not multiples of two in comparison with the other polynomial families considered, and therefore cannot be bit-shifted.

B. Example 2 (Cement Milling System)

This example is a more realistic and challenging system compared to Example 1. Consider the cement milling circuit example described in [39]

$$\begin{aligned} 0.3\dot{y}_f &= -y_f + (1-\alpha(l, v_s, d))\varphi(l, d) \\ \dot{l} &= -\varphi(l, d) + u_f + y_r \\ 0.01\dot{y}_r &= -y_r + \alpha(l, v_s, d)\varphi(l, d) \end{aligned}$$

with $\alpha(l, v_s, d) = \varphi^{0.8}v_s^4/(3.56 \times 10^{10} + \varphi^{0.8}v_s^4)$, $\varphi(l, d) = \max\{0, -0.1116dl^2 + 16.5l\}$, where y_f is the product flow rate in tons/hr, l is the load in the mill in tons, y_r is the tailing flow rate in tons/hr, u_f is the feed flow rate in tons/hr, v_s is the classifier speed in rpm, and d is the hardness of the material inside the mill. The system is discretized with a sampling time of 120 s using a fourth-order Runge–Kutta scheme. We assume that the full state information is available. The control actions are $v_s \in [165, 180]$ and $u_f \in [80, 150]$, and the outputs that are to be regulated to the reference $r = [110, 425]^\top$ are y_f and y_r. The states are bounded as follows: $y_f \in [80, 150]$, $l \in [0, 100]$, and $y_r \in [400, 500]$. MPC-related matrices, costs, and horizons are chosen as in [39]. To reiterate, this example has three states, two inputs, one output, and each of these is constrained, resulting in 18 constraints and 6 control optimization variables for a predictive horizon of three time samples.

For the ENMPC construction, we use $N = 10^4$ Halton samples (50% validation) with $\varepsilon^\star = 10^{-4}$ and tensored

TABLE II

HARDWARE ACCELERATION PERFORMANCE (EXAMPLE 2)

	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_3$	$\mathcal{M}_4$
# Gates	72K	21K	6K	3K
Power (mW)	4.29	3.06	0.75	0.79
Freq. (MHz)	152	242	365	538
$\log_{10}$ APE	-6	-6	-5	-4

Hermite polynomials as basis functions, resulting in $M = 21$ basis functions per control component. Results obtained by hardware acceleration are shown in Table II; we do not show the evolution of system states, but report that they converged to the desired equilibrium in closed loop for 1000 randomly selected initial conditions within the admissible state space. The mode $\mathcal{M}_0$ denotes the case when the approximate NMPC is implemented with the controller components in isolation, that is, each polynomial representing $\hat{u}_1 := [1 \;\; 0]\hat{u}$ and $\hat{u}_2 := [0 \;\; 1]\hat{u}$ is implemented without sharing factors between $\hat{u}_1$ and $\hat{u}_2$. The mode $\mathcal{M}_1$ represents an implementation that shares factors both within and across the polynomials $\hat{u}_1$ and $\hat{u}_2$: this is, especially advantageous because the basis functions are the same in both controller components, and only the coefficients vary. From Table II, the benefits of $\mathcal{M}_1$ are apparent: we get a 3.5$\times$ and 1.4$\times$ reduction in area and power consumption (respectively). As in Example 1, precision scaling with allowable APE set to 10^{-5} (mode $\mathcal{M}_3$) and 10^{-4} (mode $\mathcal{M}_4$) further enhances savings during acceleration. Reducing bit-width results in reduction of coefficients required to $M = 10$, which, along with FS, enables operations at 365 MHz (2.74 ns) and 538 MHz (1.86 ns), which is more than twice the rate obtained by the baseline while incurring <1 mW of power, enabling low-power, low-area, and ultrafast implementation of a sophisticated and safe control technology.

VI. CONCLUSION

In this brief, techniques from embedded system design and the emerging field of approximate computing are leveraged to provide a design methodology for ultrafast, extreme low-power, NMPC. Sampling-based methods for controller construction enable scalability in state spaces [30]; unlike state space partitioning methods that are useful for systems with $\leq$5 states [40]. This brief demonstrates the potential of careful and systematic approximation of sophisticated functions such as the NMPC control law in order to make them embeddable under severe resource constraints. Our embedded ENMPC takes a few nanoseconds and operates at below the milliwatt range without significant compromise in regulatory performance for systems of comparable size. For applications where suboptimality is tolerable, or under extreme resource constraints (extremely small sampling times, low-cost microcontrollers, or applications where reducing power consumption is critical) that render solving constrained nonconvex problems practically infeasible (such as for control of the IoT or human-powered implantables), our proposed method yields a feasible implementation of the complex NMPC framework via approximate computing. A weakness of polynomial approximation is in reconstructing highly fragmented (piecewise continuous with many partitions) control laws; recent work has demonstrated the effectiveness of neural network approximation in such scenarios [41]. Since the tools discussed in this brief are data-driven, it is agnostic to the basis function selected for approximation, enabling implementation of neural networks cheaply and with high efficacy [32].

References

[1] T. A. Johansen, "Toward dependable embedded model predictive control," *IEEE Syst. J.*, vol. 11, no. 2, pp. 1208–1219, Jun. 2017.

[2] H. J. Ferreau *et al.*, "Embedded optimization methods for industrial automatic control," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 13194–13209, Jul. 2017.

[3] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," in *Nonlinear Model Predictive Control*. Berlin, Germany: Springer, 2009, pp. 391–417.

[4] C. Kirches, L. Wirsching, S. Sager, and H. G. Bock, "Efficient numerics for nonlinear model predictive control," in *Recent Advances in Optimization and its Applications in Engineering*. Berlin, Germany: Springer, 2010, pp. 339–357.

[5] W. C. Li and L. T. Biegler, "A multistep, Newton-type control strategy for constrained, nonlinear processes," in *Proc. Amer. Control Conf.*, Jun. 1989, pp. 1526–1527.

[6] T. Ohtsuka, "A continuation/GMRES method for fast computation of nonlinear receding horizon control," *Automatica*, vol. 40, no. 4, pp. 563–574, Apr. 2004.

[7] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM J. Control Optim.*, vol. 43, no. 5, pp. 1714–1736, Jul. 2005.

[8] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: Bridging the gap via the real-time iteration," *Int. J. Control*, 2016. doi: 10.1080/00207179.2016.1222553.

[9] H. G. Bock, M. Diehl, E. Kostina, and J. P. Schlöder, "Constrained optimal feedback control of systems governed by large differential algebraic equations," in *Real-Time PDE-Constrained Optimization*. 2007, pp. 3–24.

[10] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Appl. Methods*, vol. 36, no. 5, pp. 628–647, Sep./Oct. 2015.

[11] D. Kouzoupis, H. J. Ferreau, H. Peyrl, and M. Diehl, "First-order methods in embedded nonlinear model predictive control," in *Proc. Eur. Control Conf. (ECC)*, Jul. 2015, pp. 2617–2622.

[12] B. Käpernick and K. Graichen, "The gradient based nonlinear model predictive control software GRAMPC," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2014, pp. 1170–1175.

[13] J. Kalmari, J. Backman, and A. Visala, "A toolkit for nonlinear model predictive control using gradient projection and code generation," *Control Eng. Pract.*, vol. 39, pp. 56–66, Jun. 2015.

[14] Y. Pu, M. N. Zeilinger, and C. N. Jones, "Complexity certification of the fast alternating minimization algorithm for linear MPC," *IEEE Trans. Autom. Control*, vol. 62, no. 2, pp. 888–893, Feb. 2016.

[15] J.-H. Hours and C. N. Jones, "A parametric multi-convex splitting technique with application to real-time NMPC," in *Proc. 53rd IEEE Conf. Decis. Control (CDC)*, Dec. 2014, pp. 5052–5057.

[16] G. Knagge, A. Wills, A. Mills, and B. Ninness, "ASIC and FPGA implementation strategies for model predictive control," in *Proc. Eur. Control Conf. (ECC)*, Aug. 2009, pp. 144–149.

[17] E. N. Hartley, J. L. Jerez, A. Suardi, J. M. Maciejowski, E. C. Kerrigan, and G. A. Constantinides, "Predictive control using an FPGA with application to aircraft control," *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 3, pp. 1006–1017, May 2014.

[18] B. Käpernick and K. Graichen, "Nonlinear model predictive control based on constraint transformation," *Optim. Control Appl. Methods*, vol. 37, no. 4, pp. 807–828, Jul./Aug. 2016.

[19] F. Xu, H. Chen, X. Gong, and Q. Mei, "Fast nonlinear model predictive control on FPGA using particle swarm optimization," *IEEE Trans. Ind. Electron.*, vol. 63, no. 1, pp. 310–321, Jan. 2016.

[20] T. Bächle, S. Hentzelt, and K. Graichen, "Nonlinear model predictive control of a magnetic levitation system," *Control Eng. Pract.*, vol. 21, no. 9, pp. 1250–1258, Sep. 2013.

[21] R. Quirynen, M. Vukov, M. Zanon, and M. Diehl, "Autogenerating microsecond solvers for nonlinear MPC: A tutorial using ACADO integrators," *Optim. Control Appl. Methods*, vol. 36, no. 5, pp. 685–704, Sep./Oct. 2015.

[22] M. Klaučo, M. Kalúz, and M. Kvasnica, "Real-time implementation of an explicit MPC-based reference governor for control of a magnetic levitation system," *Control Eng. Pract.*, vol. 60, pp. 99–105, Mar. 2017.

[23] H. Ayala, R. Sampaio, D. M. Muñoz, C. Llanos, L. Coelho, and R. Jacobi, "Nonlinear model predictive control hardware implementation with custom-precision floating point operations," in *Proc. 24th Medit. Conf. Control Automat. (MED)*, Jun. 2016, pp. 135–140.

[24] S. Summers, D. Raimondo, C. Jones, J. Lygeros, and M. Morari, "Fast explicit nonlinear model predictive control via multiresolution function approximation with guaranteed stability," in *Proc. 8th IFAC Symp. Nonlinear Control Syst. (NOLCOS)*, Sep. 2010.

[25] A. Chakrabarty, V. Dinh, M. J. Corless, A. E. Rundell, S. H. Żak, and G. T. Buzzard, "Support vector machine informed explicit nonlinear model predictive control using low-discrepancy sequences," *IEEE Trans. Autom. Control*, vol. 62, no. 1, pp. 135–148, Jan. 2017.

[26] V. K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. T. Chakradhar, "Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency," in *Proc. Design Automat. Conf.*, New York, NY, USA, Jun. 2016, pp. 555–560.

[27] A. Raha, S. Venkataramani, V. Raghunathan, and A. Raghunathan, "Energy-efficient reduce-and-rank using input-adaptive approximations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 462–475, Feb. 2017.

[28] A. Raha, A. Chakrabarty, V. Raghunathan, and G. T. Buzzard, "Ultrafast embedded explicit model predictive control for nonlinear systems," in *Proc. Amer. Control Conf. (ACC)*, May 2017, pp. 4398–4403.

[29] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205–1217, Oct. 1998.

[30] H. Zhang, A. Chakrabarty, R. Ayoub, G. T. Buzzard, and S. Sundaram, "Sampling-based explicit nonlinear model predictive control for output tracking," in *Proc. IEEE 55th Conf. Decis. Control (CDC)*, Dec. 2016, pp. 4722–4727.

[31] A. Raha, H. Jayakumar, and V. Raghunathan, "Input-based dynamic reconfiguration of approximate arithmetic units for video encoding," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 3, pp. 846–857, Mar. 2016.

[32] A. Raha and V. Raghunathan, "qLUT: Input-aware quantized table lookup for energy-efficient approximate accelerators," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5s, p. 130, Sep. 2017.

[33] H. Niederreiter, "Low-discrepancy and low-dispersion sequences," *J. Number Theory*, vol. 30, no. 1, pp. 51–70, Sep. 1988.

[34] A. Chakrabarty, G. T. Buzzard, and S. H. Żak, "Output-tracking quantized explicit nonlinear model predictive control using multiclass support vector machines," *IEEE Trans. Ind. Electron.*, vol. 64, no. 5, pp. 4130–4138, May 2017.

[35] L. Brandolini, L. Colzani, G. Gigante, and G. Travaglini, "On the Koksma–Hlawka inequality," *J. Complex.*, vol. 29, no. 2, pp. 158–172, Apr. 2013.

[36] S. Yu, C. Maier, H. Chen, and F. Allgöwer, "Tube MPC scheme based on robust control invariant set with application to Lipschitz nonlinear systems," *Syst. Control Lett.*, vol. 62, no. 2, pp. 194–200, Feb. 2013.

[37] M. Samadi, J. Lee, D. A. Jamshidi, A. Hormati, and S. Mahlke, "SAGE: Self-tuning approximation for graphics engines," in *Proc. 46th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2013, pp. 13–24.

[38] A. Ranjan, A. Raha, V. Raghunathan, and A. Raghunathan, "Approximate memory compression for energy-efficiency," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2017, pp. 1–6.

[39] L. Magni, G. De Nicolao, and R. Scattolini, "Output feedback and tracking of nonlinear systems with model predictive control," *Automatica*, vol. 37, no. 10, pp. 1601–1607, Oct. 2001.

[40] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, Mar. 2010.

[41] S. Chen *et al.*, "Approximating explicit model predictive control using constrained neural networks," in *Proc. Annu. Amer. Control Conf. (ACC)*, Jun. 2018, pp. 1520–1527.