Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 50-1 (2017) 11877–11882

# Nonlinear predictive control on a heterogeneous computing platform

**Bulat Khusainov** * **Eric C. Kerrigan** ** **Andrea Suardi** * **George A. Constantinides** *

* *Department of Electrical & Electronic Engineering, Imperial College London, London, SW7 2AZ, UK(e-mail: b.khusainov,a.suardi,g.constantinides@imperial.ac.uk.)*

** *Department of Electrical & Electronic Engineering and Department of Aeronautics, Imperial College London, London, SW7 2AZ, UK(e-mail: e.kerrigan@imperial.ac.uk).*

**Abstract:**
Nonlinear Model Predictive Control (NMPC) is an advanced control technique that often relies on computationally demanding optimization and integration algorithms. This paper proposes and investigates a heterogeneous hardware implementation of an NMPC controller based on an interior point algorithm. The proposed implementation provides flexibility of splitting the workload between a general-purpose CPU with a fixed architecture and a field-programmable gate array (FPGA) to trade off contradicting design objectives, namely performance and computational resource usage. A new way of exploiting the structure of the Karush-Kuhn-Tucker (KKT) matrix yields significant memory savings, which is crucial for reconfigurable hardware. For the considered case study, a 10x memory savings compared to existing approaches and a 10x speedup over a software implementation are reported. The proposed implementation can be tested from Matlab using a new release of the Protoip software tool, which is another contribution of the paper. Protoip abstracts many low-level details of heterogeneous hardware programming and allows quick prototyping and processor-in-the-loop verification of heterogeneous hardware implementations.



*Keywords:* Nonlinear predictive control; Hardware-software co-design; Scheduling

## 1. INTRODUCTION

Model predictive control (MPC) is an advanced control technique that allows systematic performance optimization, constraint handling and tackling multiple input and multiple output systems (Rawlings and Mayne, 2009). MPC relies on solving optimization problems at every sampling instant, which applies tight constraints on the algorithm execution time and hence limits the application scope to slow plants.

However, recent developments in algorithms for solving online optimization problems allows applying predictive control to systems with relatively fast dynamics (Debrouwere et al., 2014). In addition to improvements on the software side, employing reconfigurable computing platforms, such as as field-programmable gate arrays (FPGAs), resulted in a further speedup of linear MPC algorithms (Jerez et al., 2012; Hartley et al., 2014). Extending a hardware acceleration approach to nonlinear model predictive control (NMPC) is not straightforward, because NMPC involves both optimization and integration, while only the former is known to be efficiently mapped onto reconfigurable platforms. For this reason existing FPGA implementations of NMPC mostly rely on stochastic algorithms (Ayala et al., 2016; Xu et al., 2016), which cannot guarantee local optimality, feasibility and closed-loop stability. Accelerating deterministic algorithms on hardware might be achieved by employing heterogeneous computing platforms that involve both a general-purpose processor with a fixed architecture and FPGA logic. For example, Peyrl et al. (2015) present a heterogeneous implementation of a multiple-shooting based NMPC algorithm. The authors propose implementing integration on software while accelerating a fast gradient-based quadratic programming (QP) solver on an FPGA. The reported speedup of the heterogeneous implementation over a software realization is 1.6x and further improvement is limited, since integration and optimization algorithms have comparable computational complexity. This is a consequence of Amdahl's law (Amdahl, 1967), which states that an algorithm's speedup is limited by the part of the workload that cannot benefit from acceleration.

This paper proposes and investigates a heterogeneous implementation of a nonlinear interior point algorithm for predictive control. The main features of the proposed implementation are:

- Flexible splitting of the algorithmic workload between software and hardware for trading off the computational resource usage against performance. A 10x speedup of a heterogeneous implementation compared to a pure software implementation is reported.
- A new way of exploiting the structure of the Karush-Kuhn Tucker (KKT) matrix that allows a significant memory reduction compared to the existing approach


10.1016/j.ifacol.2017.08.1413

of Boland and Constantinides (2011). For the considered example, a 10x memory saving is reported.

Another contribution of the paper is a new release of the Protoip software tool (Suardi et al., 2015). The tool allows quick prototyping and processor-in-the-loop verification of optimization algorithms on a Xilinx Zynq system-on-a-chip (SoC), which contains an ARM processor and FPGA fabric. In contrast to the previous releases, which were focused on pure FPGA implementations, the new version of Protoip allows incorporating both an ARM processor and FPGA. Protoip can be used both for quick testing of the proposed implementation from Matlab and for design and verification of other heterogeneous implementations.

## 2. OPTIMAL CONTROL PROBLEM FORMULATION

We consider the nonlinear optimal control problem (OCP) with initial state $\hat{x}$ and prediction horizon $T$:

$$\min_{u,x} \quad \int_0^T \Big(\frac{1}{2}x^T(t)Q_c x(t) + \frac{1}{2}u^T(t)R_c u(t) + x^T(t)S_c u(t)\Big)dt + \frac{1}{2}x^T(T)P_c x(T) \tag{1a}$$

$$\text{subject to:} \quad x(0) = \hat{x}, \tag{1b}$$

$$\dot{x}(t) = f_c(x(t), u(t)), \quad \forall t \in [0, T] \tag{1c}$$

$$Jx(t) + Eu(t) \leq h, \quad \forall t \in [0, T] \tag{1d}$$

$$J_T x(T) \leq h_T, \tag{1e}$$

where $f_c : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$; $Q_c \in \mathbb{S}^n_+$, $R_c \in \mathbb{S}^m_{++}$ and $S_c \in \mathbb{R}^{n\times m}$ are state, input and cross penalty matrices, respectively. $\mathbb{S}^n_{++}(\mathbb{S}^n_+)$ denotes the set of positive (semi-)definite matrices. Path constraints are defined by $J \in \mathbb{R}^{v\times n}, E \in \mathbb{R}^{v\times m}, h \in \mathbb{R}^v$, where $v$ is the number of inequality constraints; the terminal constraint (1e) is defined by $J_T$ and $h_T$ with compatible dimensions. The presented formulation can be generalized for time-varying reference tracking, which, as will be shown later, requires only changing the software part of the algorithm.

## 3. TARGET COMPUTING HARDWARE

The target hardware for the proposed implementation is a Xilinx Zynq-7000 XC7Z020 SoC with dual-core ARM Cortex-A9 and FPGA logic that contains 53200 lookup tables (LUTs), 106400 flip-flops (FFs), 220 DSP blocks and 140 block RAMs with total capacity 4.9 Mb. Communication between software and hardware is performed via an AXI interface.

Programming heterogeneous platforms is not trivial , since the subsystems are often configured using different development environments, which creates problems with communication and library reusage. Modelling languages for programming SoCs, e.g. the Mathworks HDL coder, provide a high level of abstraction, ease of building prototypes and flexibility of moving the workload between subsystems. Unfortunately, these benefits come at the price of efficiency (in terms of resource usage and computation time) of the generated code. An alternative is using C-based tools, e.g. Xilinx SDSoC, that provide a good compromise between code efficiency and design effort. However, in this case, user has to take responsibility for creating testbench files, which often slows down the design process.

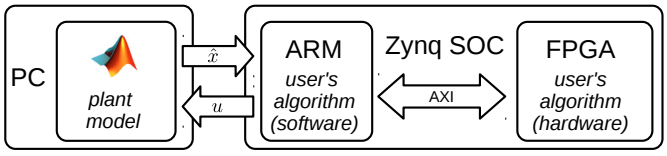


Fig. 1. Processor-in-the-loop test with Protoip.

The new release of Protoip provides infrastructure for quick prototyping and verification of online optimization algorithms using a C-based approach. Using the vendor's software on the underlying levels (Xilinx Vivado, Vivado HLS, SDK) Protoip creates template projects, software verification testbench files and processor-in-the-loop (PIL) testing facilities. A PIL test implies running the optimization algorithm on a hardware platform while simulating the controlled system on a desktop machine, which allows rapid closed-loop performance verification (see Figure 1). The tool is implemented in the Tcl scripting language and is available on the Xilinx Tcl store. A Matlab interface is provided for compatibility with state-of-the-art optimization and control toolboxes.

## 4. NONLINEAR PREDICTIVE CONTROL ALGORITHMS

Direct solution of the continuous-time optimal control problem (1) involves two main stages: integration, i.e. solving the ordinary differential equation (ODE), and optimization. Implementing integration on an FPGA is not desirable because of the following reasons:

- The ODE (1c) may involve mathematical expressions (e.g. sine and square root) that require significant amounts of computational resources compared to standard addition and multiplication operations.
- Due to the non-regular structure in $f_c$, reusing computational logic becomes non-trivial.

Optimization algorithms, on the other hand, can benefit from hardware acceleration due to (i) their iterative nature, which is beneficial for reusing computational logic, and (ii) the fact that linear algebra algorithms can be efficiently mapped onto hardware (Jerez et al., 2012).

Taking the above into account we consider two classes of algorithms for solving (1): shooting-based and direct transcription algorithms (Betts, 2010). The common feature of shooting methods is decoupling the ODE and optimization solvers. Accelerating only the latter does not result in significant improvements, due to Amdahl's law, since the workloads of the two operations are comparable. In contrast, direct transcription algorithms transform (1) directly to a discrete OCP by approximating the ODE with algebraic equations based on numerical integration equations, i.e.

$$\min_{\substack{u_0\dots u_{N-1}\\ x_0\dots x_N\\ r_0\dots r_{N-1}}} \sum_{k=0}^{N-1}\Big(\frac{1}{2}x_k^T Q_d x_k + \frac{1}{2}u_k^T R_d u_k + x_k^T S_d u_k\Big) + \frac{1}{2}x_N^T P_d x_N \tag{2a}$$