

A Coprocessor Accelerator for Model Predictive Control

Panagiotis Vouzis & Mark Arnold
Computer Science and Engineering Dept.
Lehigh University
Bethlehem, PA 18015, USA
{vouzis, maab}@lehigh.edu

Leonidas Bleris
Bauer Laboratory
Harvard University
Cambridge, MA 02138, USA
lbleris@cgr.harvard.edu

Mayuresh Kothare
Chemical Engineering Dept.
Lehigh University
Bethlehem, PA 18015, USA
mayuresh.kothare@lehigh.edu

Yongho Cha
Advanced Digital Chips
Seoul, 135-270, Korea
cygx1@adc.co.kr

Model Predictive Control (MPC) [1] is an established control technique that has found widespread application mainly in the chemical-process industry. There is increased interest for its introduction into a wide range of nonindustrial applications due to its ability to handle Multiple-Input-Multiple-Output (MIMO) systems and to take into account constraints and disturbances explicitly. Notably, the explicit handling of constraints by MPC makes it the algorithm of choice for safety-critical control systems found in small physical size applications, such as drug delivery [2], and microchemical systems that encompass a chemical system with a number of actuators and sensors [3]. The efficient implementation of a Digital-Signal-Processing (DSP) system for MPC requires a custom-designed architecture that is tailored to the particular algorithm in order to exhibit reduced area and consequently reduced power consumption, while meeting the real-time operation requirements.

The limiting factor for the adoption of MPC by small physical size applications is its high computational requirements. On each time-step MPC requires the solution of an optimization problem. When MPC is used for industrial applications, which are systems with slow dynamics, a general-purpose high-end workstation can be utilized to carry out the necessary computations. For applications, whose physical size is small, the limited area and power constraints have to be addressed carefully by a custom-designed hardware architecture.

In this work we propose an architecture that consists of a general-purpose microprocessor and an auxiliary unit, tailored to accelerate computationally-demanding MPC operations. For example, the basic update step of the Newton optimization algorithm is given by the equation

$$u(t+1) = u(t) - H(f(u))^{-1} \cdot \nabla(f(u)), \quad (1)$$

where $u(t+1)$ is the new control move, $u(t)$ the previous move, $H(f(u))^{-1}$ is the inverse of the Hessian of the objective function, $f(u)$, and $\nabla(f(u))$ the Gradient of the objective function. We can see there are matrix inversions, matrix-by-vector multiplications, vector subtractions and also a number of matrix operations required by the Hessian and Gradient

The design objective for implementing MPC is an architecture that occupies small area, is efficient enough to meet the real-time requirements of the target application, and it can be combined with a general-purpose microprocessor, such

as the ones encountered in embedded systems. Towards this direction, a co-design methodology is used to develop an architecture that is efficient both in power consumption and performance, while it is sufficiently flexible to be embedded in bigger systems that need to include MPC in their functionality. Co-design combines software's flexibility, which is executed on the general-purpose microprocessor, with the high performance offered by hardware.

The computationally intensive parts of the algorithm are implemented in hardware by a matrix coprocessor in order to reduce the computational delay and free the microprocessor from the burden, while software is used to carry out algorithmic-control tasks and high-level operations. The total number of arithmetic operations is a substantial load for an embedded general-purpose microprocessor which may have to carry out a number of other tasks required by the embedded application. The decision for the partitioning of the MPC between software (microprocessor) and hardware (coprocessor) is the most critical one in the whole process, and it is based on a profiling study of the algorithm which helps to identify its bottlenecks. The analysis led to the transfer of the calculation of the Gradient, the Hessian and the inverse Hessian to the coprocessor, while the rest of the algorithm is executed on the general-purpose microprocessor.

The path of the co-design process begins with the setting of the specifications, and the software-hardware partitioning follows. After the communication protocol between the two parts is specified, these are implemented by using a Hardware Description Algorithm (HDL) for the hardware and a high-level programming language for the software. The next step is to co-simulate the two parts in order to verify the correct functionality and the performance of the complete design. If the verification process fails, then a backward jump is made to the appropriate design step, e.g., if the performance of the system does not meet the specifications, then a new partitioning decision is made and the design process is repeated.

The matrix coprocessor encompasses a one-hot Finite State Machine (FSM) and an Arithmetic Logic Unit (ALU) that can carry out one multiply-accumulate operation per clock cycle, i.e., it operates in an iterative fashion on the matrix elements. The implementation of the FSM was developed by using a tool called VITO, which is a preprocessor for the Verilog HDL.

VITO simplifies the description of one-hot FSMs to software-like statements (while loops, if-then-else branches, etc.) which speed-up the development process [4].

The software part is developed in the C programming language which has an extended instruction set consisting of the commands that are executed by the matrix coprocessor. There are conventional commands, such as loading a matrix (LOADC), storing a matrix (STOREC), outputting a matrix (OUTC), and novel custom-designed ones such as calculating $1/a_i^2$ and $1/a_i^3$ for a vector a (POW2A, POW3A), multiplying a matrix C by vector b (MULV), finding the pivot line for Gauss-Jordan inversion (PIVOT). The general-purpose microprocessor acts as the master in the system; i.e., it carries out the tasks of Input/Output (I/O), initializes and sends the appropriate commands to the auxiliary unit and receives back the optimal control moves. The auxiliary unit acts as a matrix coprocessor by carrying out matrix operations, such as addition, multiplication, etc. While the coprocessor executes a command, the microprocessor can run any other task.

The wide dynamic range required by MPC does not favor the use of a fixed-point number system, thus the viable alternatives are between the Floating-Point (FP) number system and the Logarithmic Number System (LNS) [5], which has been adopted for the arithmetic operations on the coprocessor. The utilization of LNS allows the reduction of the required word-length to 16 bits, and consequently a general-purpose microprocessor of the same word-length is used. The alternative of an equivalent FP unit, although exhibiting similar delay as the LNS, is proven to occupy 40% more area [6], and thus consumes more power. The adoption of LNS in this work, instead of FP, is based on the study by Garcia et al. [6] where it is shown that a reduced-precision LNS is capable of solving an MPC problem efficiently in terms of performance and area, which are very critical aspects when low-power consumption and embeddability are sought. The same authors in [7] propose an LNS-based Application-Specific-Instruction Processor (ASIP) of reduced precision suitable for MPC algorithms. Further proof of the applicability of LNS for MPC problems is given in [8]. This work presents a different approach in the utilization of LNS in terms of architecture, since we propose a design that consists of two parts, the microprocessor and the coprocessor that incorporates an LNS unit with a algorithm-specific instruction set, and not a single ASIP that uses LNS to carry out the arithmetic operations as in [7].

Both the microprocessor and the coprocessor are described in Verilog and were simulated by using ModelSim XEIII/Starter 6.0a in order to verify the functionality and measure the performance of the system. For implementation purposes we selected the 16-bit Extensible Instruction Set Computer (EISC) of ADCUS to act as the host. The target technology for synthesis is a Field Programmable Gate Array (FPGA) of Xilinx; thus the development environment ISE 7.1 of the same vendor is used. The program running on the microprocessor is developed using the EISC Studio of ADCUS. A prototype is developed using the Xilinx ML401

board which hosts a XC4VLX25-FF668-10C Virtex-IV FPGA.

The functionality of the system was tested with hardware-in-the-loop simulation for two control problems—a linear antenna-control problem [9] and the nonlinear glucose-regulation problem for diabetic patients [10]. The antenna-control problem can be solved by using Motorola's 32-bit MPC 555 processor, running at 40MHz and incorporating a 64-bit FP unit (double precision), in 15ms [11], while the microprocessor-coprocessor proposed architecture can solve the same problem in 0.89ms running at 5MHz. We can see, for the particular problem, there is a 27 times speed-up (normalized to the same clock speed) while the 64-bit FP unit of the Motorola MPC 555 occupies 17 times more area than the 16-bit LNS arithmetic unit [7]. This is an example of the amount of speed-up and area reduction that can be achieved by using the custom-designed microprocessor-coprocessor architecture, instead of an off-the-shelf general-purpose microprocessor.

The microprocessor of an embedded systems should be chosen to be the most economical one in terms of performance, wordlength, peripherals, and memory size—all of which aim to a system with reduced power consumption and cost. When such an embedded system needs to incorporate MPC functionality, the proposed matrix coprocessor offers a cost-efficient solution capable of bearing the computational effort of the algorithm in real time. The interested reader is referred to [12] for more details on the architecture and the design process.

REFERENCES

- [1] E. F. Camacho and C. Bordons, *Model Predictive Control*. Springer-Verlag, London, 2nd edition, 2004.
- [2] R. S. Parker, F. J. Doyle III, and N. A. Peppas, "A Model-Based Algorithm for Blood Glucose Control in Type I Diabetic Patients," *IEEE Trans. on Biomedical Engineering*, vol. 46, pp. 148–156, Feb. 1999.
- [3] K. F. Jensen, "Microchemical Systems: Status, Challenges, and Opportunities," *AIChE Journal*, vol. 45, pp. 2051–2054, Oct. 1999.
- [4] M. G. Arnold and J. Shuler, "A Processor that Converts Implicit Style Verilog into One-hot Designs," in *Proceedings of the 6th International Verilog HDL Conference*, (Santa Clara, CA), pp. 38–45, 1997. www.verilog.vito.com.
- [5] E. E. Swartzlander and A. G. Alexopoulos, "The Sign/Logarithm Number System," *IEEE Trans. on Comp.*, vol. 24, pp. 1238–1242, Dec. 1975.
- [6] J. G. Garcia, M. G. Arnold, L. G. Bleris, and M. V. Kothare, "LNS Architectures for Embedded Model Predictive Control Processors," in *Proceedings of the 2004 CASES International Conference*, (Washington, DC), pp. 79–84, Sept. 2004.
- [7] L. G. Bleris, J. G. Garcia, M. V. Kothare, and M. G. Arnold, "Towards Embedded Model Predictive Control for System-on-a-Chip Applications," *Journal of Process Control*, vol. 16, pp. 255–264, March 2006.
- [8] L. G. Bleris, J. G. Garcia, M. G. Arnold, and M. V. Kothare, "Model Predictive Hydrodynamic Regulation of Microflows," *Journal of Micromechanics and Microengineering*, vol. 16, pp. 1792–1799, Sept. 2006.
- [9] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust Constrained Model Predictive Control Using Linear Matrix Inequalities," *Automatica*, vol. 32, pp. 1361–1379, Oct. 1996.
- [10] R. S. Parker, F. J. Doyle III, and N. A. Peppas, "The Intravenous Route to Blood Glucose Control," *IEEE Engineering in Medicine and Biology*, vol. 20, pp. 65–73, Jan./Feb. 2001.
- [11] L. G. Bleris and M. V. Kothare, "Real-Time Implementation of Model Predictive Control," in *Proceedings of the American Control Conference*, (Portland, OR), pp. 4166–4171, June 2005.
- [12] P. Vouzis, L. Bleris, M. Kothare, and M. Arnold, "A System-on-a-Chip Implementation for Embedded Real-Time Model Predictive Control," submitted to *IEEE Trans. on Control Systems Technology*, 2006.