On-Line Arithmetic for Real-Time Control of Microsystems

Martin Dimmler, Arnaud Tisserand, Ulf Holmberg, and Roland Longchamp

N.B.: When citing this work, cite the original article.

©1999 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.


DOI: http://dx.doi.org/10.1109/3516.769548
Copyright: IEEE

Post-Print available at: Halmstad University DiVA
http://urn.kb.se/resolve?urn=urn:nbn:se:hh:diva-3379
On-Line Arithmetic for Real-Time Control of Microsystems

Martin Dimmler, Arnaud Tisserand, Ulf Holmberg, and Roland Longchamp

Abstract—The integration of microcontrollers within mechanical systems is a current trend. However, decreasing the size of the system and satisfying higher precision requirements make it necessary to reevaluate the common signal processing techniques for controller implementations, because limited controller size, computation speed, and power consumption become major topics. In this paper, we demonstrate that serial computations with the most significant digits first, that is, on-line arithmetic, offer an important potential for real-time control. They enable a combination of traditional functions, such as analog-to-digital converters and control data computations. This leads to very efficient controller implementations with small size, high speed, and low power consumption. After a brief description of the requirements and challenges of microsystem controller design, the use of on-line arithmetic for real-time control is proposed. A short introduction to on-line arithmetic is given and control-specific implementation guidelines are presented and finally applied to a simple test system.

Index Terms—Low power consumption, microsystem control, miniaturization, on-line arithmetic, real-time control.

I. INTRODUCTION

The design and manufacturing of mechanical components and systems have reached a very high standard. Combined with the low-cost integration of microelectronics, this offers new possibilities for compact high-precision mechanisms. Several applications have already appeared on the market, e.g., drives, robots, or fine-positioning devices. They are mostly controlled by digital controllers, such as microcontrollers, digital signal processors (DSP’s), or application-specific integrated circuits (ASIC’s) with fixed parameters. These circuits are generally based on digit-parallel arithmetic operators which are sequentially scheduled by an instruction set in the memory [see Fig. 1(a)]. However, in most mechatronic systems, these general purpose solutions are only necessary during controller development. Afterwards, at run time, the controller repeats a certain number of operations cyclically with very few user interactions. The whole control algorithm could be implemented using a complex operator. This avoids communication delays between memory and arithmetic logic unit (ALU) and offers, in particular, for multiple input multiple output (MIMO) systems, a potential for efficient parallel computation of independent terms [see Fig. 1(b) and (c)]. The inherent disadvantage of using digit-parallel arithmetic for these special operators is the high gate number, resulting in increased space and power consumption. However, in particular, in microsystem technology, small dimensions, low power consumption, and high controller speed are the most important controller requirements. For example, in many mobile and aerospace applications, battery lifetime and system dimensions play a major role.

In the past, digit-serial least significant digits first (LSDF) arithmetic has often been suggested to reduce the computation complexity [1], [2]. Its main advantages are as follows:

• simplicity of the basic operators (digit level);
• serial communication (few I/O pins);
• potential overlapping of several operations.

However, two main disadvantages make the LSDF approach too slow for real-time control of microsystems. First, sequential A/D converters work in the most significant digits first (MSDF) direction. Consequently, large delays are necessary to transform their outputs to LSDF form. Secondly, multiplications in the LSDF mode produce the unused least significant half of the result first. In particular, for control algorithms with many multiplications, computation time or necessary clock speed increase significantly.

Herein, the new concept of using a known MSDF serial arithmetic, so called on-line arithmetic, in real-time control systems will be

![Fig. 1. Timing and size aspects of the computation $a x_1 + b x_2 + c x_3$ with different operational schemes. (a) Sequential processing with simple operators.
(b) Complex digit parallel operator. (c) On-line arithmetic operators (MSDF).](image-url)
developments are presented in Section V. Compared to an optimized parallel design, a conclusion and future

Section IV, this library is used for the PID controller. The result is

specific issues are introduced which make the given concept useful

implementation concepts have been omitted. For example, the need

for controller implementations. These guidelines are demonstrated

low power consumption. In previous studies, Ercegovac [3] and

In on-line arithmetic, the operands, as well as the results, flow

MSDF. Further details can be found in [3], [6], and [7].

This special digit-serial arithmetic was introduced by Ercegovac

II. INTRODUCTION TO ON-LINE ARITHMETIC

For the reason of size restrictions, only a short introduction is given

In on-line arithmetic, the operands, as well as the results, flow through arithmetic units in a digit serial fashion starting with the

important characteristics of on-line operators are their delay ($\delta$) and period ($\tau$), as illustrated in Fig. 2. Serial computations with the

The used redundant number systems [8] permits negative digit values, i.e., the number $0.a_1a_2 = \sum_{i=1}^{2} a_i r^{-i}$ of radix $r = 2$ can have negative $a_i$. This leads to several representations for some numbers ($\frac{1}{4} = 0.a_1a_2$ with $a_1 = 0$ and $a_2 = 1$ or $a_1 = 1$ and $a_2 = -1$). With

radix 2, usually a 2-bit representation ($a^+_1$, $a^-_1$), called borrow–save, of the digits ($a_i$) is defined as follows: $a^-_1 = a^+_1 - a^-_1$. Thus, digit 1 is represented by $(1, 0)$, digit $-1$ by $(0, 1)$, while digit 0 has two possible representations, namely $(0, 0)$ and $(1, 1)$.

This special digit-serial arithmetic was introduced by Ercegovac and Trivedi in 1977 [9]. Nowadays, on-line algorithms are available for all common arithmetic operations, in the fixed-point representation, as well as in the floating-point representation, but they have been rarely used in hardware applications. This is mainly due to the different original motivation (high-precision computation) and the lack of a convenient formulation for an efficient hardware implemen-
tation. In recent years, more effort has been spent on implementation issues (e.g., [6] and [10]). Two different approaches have been chosen. One follows the recursive formulation of Ercegovac [10] and the other is based on Avizienis’ parallel adder (Fig. 8). The latter leads to much smaller implementations, but is limited to a few operations (addition, multiplication). The application presented here is mainly concerned with size and power consumption requirements, and additions/multiplications represent the majority of the operations. Therefore, the second approach has been focused on. The functional schemes of the multiplier and adder, as well as their building blocks, are shown in Figs. 7, 4, 8, and 9, respectively.

III. IMPLEMENTATION OF ON-LINE ARITHMETIC FOR CONTROL ALGORITHMS

Previous literature focused rather on single on-line operations than on their interconnection to implement complex algorithms. Consequently, no uniform framework existed and, usually, arithmetic experts were needed for the implementation of specific algorithms.

In this section, we supply the implementation guidelines necessary for the realization of a modular library of basic on-line operations. Each operator is composed of four main building blocks: initialization, mathematical algorithm, normalization, and output switch (see Fig. 3). The different mathematical algorithms can be found in the literature (e.g., [6]). The other three blocks are explained in more detail below. The common interface of these modular on-line arithmetic operators enables system designers to construct mechatronic controllers in on-line arithmetic without advanced knowledge in computer arithmetic.

A. Initialization of On-Line Operators

In digit-serial arithmetic, the operands are distributed over several subsequent operations (operators work wise), and there is an internal state update in the operators at each clock period. Therefore, a clear indication of every operation start is necessary for initialization. A simple way to achieve this is by a distributed control scheme in the form of an additional control line synchronized to the operands. The line is kept high if significant operand digits are present and is otherwise low. Internal state and status values (e.g., intermediate results in multiplications) are thereby reset as soon as an operator is unused. In Fig. 4, the initialization (init) is shown for an on-line adder. The two registers in the init block are necessary to compensate for the operator delay ($\delta_{adder} = 2$). As soon as $ctr_{in} = ctr_{out} = 0$,
the three registers of the adder are reset. The initialization takes at least one clock cycle (see Fig. 5).

In the initialization scheme, the digits of the result must have left the operator entirely before the reset was achieved. Otherwise, the last \( \delta \) (on-line delay) digits of the result would be wrong. Therefore, at least \( \delta_{\text{max}} + 1 \) intermediate zeros between the operands must be inserted at algorithm entry, where \( \delta_{\text{max}} \) is the largest delay of all of the operators in the entire algorithm. In Fig. 5, an algorithm is supposed to have two operators and \( \delta_{\text{op1}} > \delta_{\text{op2}} \). The additional delay is introduced for the initialization. The zeros increase the sampling period. However, in order to ensure the correct timing of the controller (sampling instant and termination of the D/A conversion at the same time), many more intermediate zeros are needed anyway. In the case where converter and arithmetic have the same resolution, the number of zeros is determined by

\[
\delta_{\text{zeros}} = \delta_{\text{D/A}} + \delta_{\text{sampler}} + \delta_{\text{Arithmetic}},
\]

where \( \delta_{\text{D/A}}, \delta_{\text{sampler}}, \delta_{\text{Arithmetic}} \) are the delays of the D/A converter, the sampler of the A/D converter, and the controller arithmetic, respectively.

In order to avoid an interference from subsequent numbers, these intermediate zeros have to be maintained, even after several operations in the algorithm. This output switching can be realized with the additional control line (see Fig. 4, out-switch block).

This distributed control scheme improves the modularity of the design. Controller execution can be stopped easily by resetting the registers in the initialization block.

### B. Synchronization

Implementations of dynamic systems give rise to loops in the signal flows due to the states. These states need to be synchronized to the signals in the forward flows. In Fig. 5, the output of the second operator (Op2) is a state that is used as input to Op2 at the next sampling instant. For synchronization, shift registers of size \( n \) (number resolution) are included in the backward branch. The shift operation is controlled by simple glue logic based on the control line \( \text{ctr}_3 \).

### C. Normalization

In redundant number systems, some numbers can have several representations (e.g., \( 1 - 1/4 = 1.0(-1) = 0.11 = 1/2 + 1/4 \) in radix 2, notations as in Section II). This property allows that in additions the sum can be represented by \( n + 1 \) valuable digits whereas the operands and the theoretical result, using a nonredundant number system, only need \( n \) digits. In multiadders, even several digits in front of the point are possible. In order to avoid a continuously growing number of digits after additions, especially in state loops (growing number of additions), a conversion to a limited representation is necessary. In the literature, two approaches have been given. One is the complete on-the-fly normalization algorithm by Ercegovac and Lang [11] which converts the redundant numbers into conventional digital representations. Another is Merrheim’s normalization algorithm [5] which generates a redundant fractional number with zero unit part (i.e., \( 0, s_1, s_2, s_3, \ldots \)). However, the former causes a delay of \( n \) clock cycles and the latter is only appropriate for additions of two numbers (but without on-line delay). Herein, an extension of Merrheim’s algorithm also suitable for multiadditions is proposed.

**Proposition:** With an appropriate scaling of the operands \( |s| < 1 \) only two types of result need conversion (notations as in the numerical examples above)

\[
1(1 - r) \cdots (1 - r), 0 \cdots 0(-a) s_{m+1} \cdots s_n
\]

\[
\rightarrow 0, (r - 1) \cdots (r - 1)(r - a) s_{m+1} \cdots s_n
\]

\[
(-1)(r - 1) \cdots (r - 1)0 \cdots 0a s_{m+1} \cdots s_n
\]

\[
\rightarrow 0, (1 - r) \cdots (1 - r)(a - r) s_{m+1} \cdots s_n
\]

where \( r \) is the radix and \( a, (-a) \) satisfying \( 0 < a < r \), are the first nonzero digits after the decimal point.

**Proof:** Consider \( s_i \) and \( s'_i \) to be the digits before and after the conversion, respectively. Suppose there are \( k + 1 \) nonzero digits before the decimal point in (1). Then, up to the \( m \)th digit

\[
\sum_{i=0}^{m-1} s_i r^{-i} + s_m r^{-m} = r^k + (1 - r) \sum_{i=0}^{m-1} a_i r^{-i} - a r^{-m} = (r - 1) \sum_{i=0}^{m-1} r^{-i} + (r - a) r^{-m} = \sum_{i=0}^{m-1} s_i' r^{-i}
\]

Conversion of (2) is shown similarly \( \square \)

**Remark:** In case of overflow, the closest possible value appears on the output \( (0, (r - 1) \cdots (1 - r) = 0) \) or \( 0, (1 - r) \cdots (1 - r) \), respectively.

The algorithm was implemented for radix 2 in a field programmable gate array (FPGA) and requires approximately the space of 20 Actel cells. This is not significant if used only a few times in a design (operator combinations reduce occurrence, see Section III-E).

Two additional properties should be mentioned in order to use most advantageously the introduced modular on-line arithmetic operators. First, appropriate controller representations, and secondly, simplifications by using multiplications are presented.

### D. Appropriate Controller Representation

The controller representation has a significant influence on calculation speed and implementation complexity. A state-space representation based on the Jordan form offers many advantages for the implementation in on-line arithmetic. The state updates and the output equations can be calculated in parallel and the constants in the dynamic matrix are scaled \( |a, i| < 1 \) for stable controllers. In particular, the latter is important because it keeps the delay of the multipliers small (for fixed-point operands, the range of multiplicative constants has a significant influence on the operator delay). The controller deadline for fast sampling designs is exactly one sampling period. Consequently, the controller has no direct term. The deadline of one period has to be taken into account during controller design.

This transformation is highlighted by an implementation example. The aim herein is to introduce the concept of on-line arithmetic for real-time control, rather than to describe a specific application. Therefore, the simple PID algorithm with filtered differential part is treated

\[
u_{k} = K \left( e_{k-1} + \frac{h}{T_{i}} x_{i, k-1} + T_{d} x_{d, k-1} \right)
\]

with

\[
x_{i, k-1} = e_{k-1} + x_{i, k-2}
\]

\[
x_{d, k-1} = \gamma \left( e_{k-1} - e_{k-2} + \tau x_{d, k-2} \right)
\]

(3)

where \( e_{k} = s_{k} - y_{k} \) is the difference between set point \( s_{k} \) and measurement \( y_{k} \), \( x_{i, k} \) and \( x_{d, k} \) the controller states, \( u_{k} \) the controller output, \( h \) the sampling period, \( \gamma \) the time constant of the differential part filter, \( \gamma = 1/(\gamma + \tau) \) and \( K, T_{i}, T_{d} \) are the proportional, integral, and differential gain, respectively.
Transformation to Jordan form gives

\[ z_{k+1} = A z_k + B u_{k-1} \]

\[ u_k = C z_k + D v_{k-1} \]

where

\[ A = \begin{pmatrix} \tau \gamma & 0 \\ 0 & 1 \end{pmatrix} \]

\[ B = \begin{pmatrix} -TD\gamma (1 - \tau \gamma) \\ \frac{h}{T_f} \end{pmatrix} K \]

\[ C = \begin{pmatrix} 1 & 1 \\ \frac{h}{T_f} + TD\gamma \end{pmatrix} K. \]

E. Simplification by Using Multioperations

For Matrix x Vector computations, on-line arithmetic offers an efficient simplification. The static logic part of all multiplications can be executed in parallel without on-line delay, and the final addition can be performed by a simplified final adder with a much smaller delay than a binary tree of adders. The algorithm is given in [6]. This simplification keeps the overall delay of the controller, as well as the number of individual operators, very small. The consequence is a reduced calculation time and a small number of necessary normalization units (scaling is limited to a few intermediate results). In our example, every row of (4) is calculated by a simplified multiadder (see Fig. 6).

F. Hardware and Software Support

With programmable logic, such as FPGA’s, efficient implementations of algorithms in hardware can be made with software-like design principles (for examples, see [12]). These gate arrays can be used as testbeds for later implementations in ASIC’s or directly as mechatronic controllers. The controller algorithms are herein composed in a graphical or formal description language (e.g., VHDL1). A formal description is generally preferred because of its greater possibilities for validation and implementation on different hardware platforms. Once the basic arithmetic modules are available, implementing a controller using on-line arithmetic is very similar to standard arithmetic approaches.

IV. IMPLEMENTATION

A prototype board holding 2 Actel FPGA’s (1 x 1240A, 1 x 1280A), some logic for a PC bus interface, as well as 2 A/D and 2 D/A converters was built. It enables the implementation of different controllers and the monitoring of run-time values via a PC bus interface. In order to verify our design rules, we implemented the PID scheme for resolutions of 12 bits and 24 bits in the Actel 1280A device (a low-cost FPGA in antifuse technology; for specification details see [13]). The resulting operational scheme for our PID example is shown in Fig. 6 for a resolution of n bits. It includes two inner loops with appropriate registers and three Vector x Vector operators with simplified final adders. In order to reduce the length of the critical combinatorial paths between the entry of the subtraction and the output of the multipliers, the inputs of the multipliers are delayed by one clock cycle. This reduces the period \( \tau \). Due to the choice of reasonable controller constants and an analog output gain, the on-line delay of the controller is \( \delta_{\text{arithmetic}} = 6 \). For the chosen converters, the times \( \tau \times \delta_{\text{convert}} = 1 \mu s \) and \( \tau \times \delta_{D/A} = 0.4 \mu s \) are given. The sampling time was fixed to a reasonable value of 25 \( \mu s \). Resulting clock frequencies and number of intermediate zeros are listed in Table I.

We compared our design to a sequential realization in digit-parallel arithmetic (see Table I). For the digit-parallel operators, a carry-look-ahead adder and a Wooley [14] multiplier were used. Only \( u \) is calculated immediately after the A/D conversion. The state updates of \( z1 \) and \( z2 \) are made in parallel to the following conversion. Owing to the simplicity of the PID controller, the clock frequencies are comparable for the two approaches. However, the on-line solutions are much smaller and fit on single one single FPGA, even for 24-bit resolution. The power consumption could not be evaluated explicitly (FPGA implementation), but it would certainly be higher in the parallel case (more gates and bus traffic).

Note that more drastic savings in size and power consumption can be expected for a higher complexity of the controller, e.g., multivariable controllers with coordinate transforms and sensor preconditioning. In particular, nonlinear operations, as for example trigonometric functions in control of rotational magnetic bearings or divisions in advanced friction compensation based on the new LuGre model [15], are very costly for digit-parallel arithmetic (in size and speed). In on-line arithmetic, these operations still grow only linearly with resolution and can be put in parallel with other computations.

V. CONCLUSIONS AND FUTURE DEVELOPMENTS

After a discussion of the main challenges in microsystems controller design, it has been shown that a digit-serial processing scheme based on on-line arithmetic is particularly well suited for real-time control. Some guidelines for their control-specific implementation have been presented and verified for a simple implementation example. Their small size, high speed, and low power consumption make them well suited to microsystems.

In parallel to this work, the use of radix 4 was investigated. In [16], it has been shown that radix 4 offers a lower maximum delay and calculation time due to the much shorter representation on the digit level. However, operator complexity is significantly higher than in radix 2. Therefore, more attention was given to radix 2.

With the PID example, the concept was verified. The next challenge will be a controller implementation for a complex mechatronic system, including sensor signal calibration and transformation. Demands

1VHDL stands for VHSIC (very-high-speed integrated circuit) hardware description language.
have arisen from two application areas. One is an optical terminal for satellite communication where a triangular pointing mirror, driven by three actuators, one in each mirror corner, has to follow a fast reference signal. The other is a magnetic bearing for the actuation and guidance of a computer hard disk. In both cases, very fast controllers with strict size and power consumption constraints are required. An on-line implementation for these systems has, in addition to the arguments presented above, the important advantage that a large number of sensor interfaces can be realized in a serial manner. This reduces significantly interconnection constraints and communication delays.

APPENDIX

The functional scheme of the on-line arithmetic multiplication with a constant number and its necessary building blocks are displayed in Figs. 7–9. The on-line addition can be found in Fig. 4. For the construction of multiadders and more details, see [6].

ACKNOWLEDGMENT

The authors would like to thank the members of the Centre Suisse d’Electronique et de Microtechnique, in particular, Dr. J. Moerschell, for many fruitful discussions, as well as Prof. J.-M. Muller and Prof. J.-D. Nicoud for their comments.

REFERENCES