Design of high-speed and low-power finite-word-length PID controllers

A. K. OUDJIDA [1†], N. CHAILLET [2], A. LIACHA [1], M. L. BERRANDJIA [1], M. HAMERLAIN [1]

1.*Centre de Développement des Technologies Avancées, Algiers, Algeria;*

2.*FEMTO-ST Institute, Besançon, France*

Abstract:

ASIC or FPGA implementation of a finite word-length PID controller requires a double expertise: in control system and hardware design. In this paper, we only focus on the hardware side of the problem. We show how to design configurable fixed-point PIDs to satisfy applications requiring minimal power consumption, or high control-rate, or both together. As multiply operation is the engine of PID, we experienced three algorithms: Booth, modified Booth, and a new recursive multi-bit multiplication algorithm. This later enables the construction of finely grained PID structures with bit-level and unit-time precision. Such a feature permits to tailor the PID to the desired performance and power budget. All PIDs are implemented at register-transfer-level (RTL) level as technology-independent reusable IP-cores. They are reconfigurable according to two compile-time constants: set-point word-length and latency. To make PID design easily reproducible, all necessary implementation details are provided and discussed.

Keywords: Design-reuse; Embedded finite-word-length (FWL) controllers; Intellectual property (IP); Linear time invariant (LTI) systems; Low-power and speed optimization; PID

1 Introduction

The PID is by far the most commonly used feedback controller due to its simple structure and robust performance [1]. An important feature of this controller is that it does not require a precise analytical model of the system that is being controlled, which makes it very attractive for a large class of dynamic systems. While PID is well adapted for linear-time-invariant (LTI) systems [2], it stands powerless for non-LTI ones. Never-theless some solutions exist, such as partitioning the non-LTI control algorithm into a linear portion and a nonlinear portion [3–5]. The linear portion represents the major control loop and is computed using an integrated PID, while the nonlinear portion that acts as dynamic compensation to the linear one is performed in software using a general-purpose-microprocessor or a DSP.

In embedded control applications, such as in small-

---

†Corresponding author.

E-mail: a_oudjida@cdta.dz. Tel.: +213-21-351018; fax: +213-21-351039.

scale mobile robot, the control-loop-cycle is very tight and the power budget is very limited. A low sample rate leads to poor and degraded control-performance. Moreover, high power consumption shortens the battery lifetime. To cope with these two severe and antagonistic constraints, the need for both a high-speed and low-power PID structure is of utmost importance.

Today, design-reuse [6] is a well-established design standard that allows grasping with rapid technology changes and increasing design complexity. It consists in the use of predesigned technology-independent, generic and reconfigurable IP-cores [7], most generally implemented at register-transfer-level (RTL).

However, at RTL abstraction level, no significant optimization results can be achieved if not undertaken at architectural and especially at algorithmic level. To achieve such a goal, a deep insight into PID arithmetic is necessary. At this stage, a choice of a numeric representation format is a crucial issue. Compared to floating-point, fixed-point format is the best candidate for optimized designs as it is much simpler to implement, faster, power-efficient and requires far much less hardware resources. However, the limited dynamic range can be source of control instability. This problem, referred to as finite-word-length (FWL) effect is an active research area that aims to shorten the floating-to-fixed point conversion time while preserving control performances [8, 9].

The digital implementation of PID controllers went through several stages of evolution, initially dominated by the use of commercial-of-the-shelf (COTS) components and DSP. However, over the past few years, FPGAs have brought a key advantage to digital control: the inherent parallelism of FPGA architecture allows many independent control loops to run at different deterministic rates without relying on shared resources that might slow down their responsiveness as in the case of COTS and DSP [10, 11].

A survey of recent PID related works can be classified into three categories. The biggest one includes works that are straightforward FPGA implementations targeting specific applications: DC-DC converter [12], temperature control [13], motor multi-axis control [14], liquid level control [15], and Xilinx versus Altera FPGA implementation for result comparison [16]. The second category proposes methodologies that analyze the FWL effect on PID controller in order to reduce the number of hardware resources [17, 18]. Moreover, finally the third category, paradoxically the smallest one despite the large popularity of PID, comprises architecture-

optimization works. In [19], low-power serial and parallel multiple-channel PID architectures are proposed for small mobile robots. In this work, the optimization was carried out at macro-level considering several PIDs, rather than at micro-level (optimization of the PID itself). Nevertheless, the whole architecture will deliver much more interesting results if combined with an optimized PID. The second work [20] proposes serial, parallel, and mixed PID architectures incorporating different number (1 – 3) of multiplication cores. High power consumption, even with the serial architecture, and complex control-part are the two major shortcomings of this proposal. Finally, in [21], an attractive optimized PID structure based on distributed arithmetic (DA) is presented. Although this latter exhibits interesting results in terms of resource utilization and power consumption, it suffers from three serious drawbacks: high latency ($n + 1$ clock-cycles for n bit set-point word-length), FPGA technology-dependent as it is essentially based upon FPGA look-up-tables (LUTs), and inability to handle time-varying PID parameters since they are precomputed and stored into LUTs. Nevertheless, it is considered as a reference design against which the obtained results are confronted into the same conditions.

The objective of this paper is to design optimized FWL-PID structures that overcome all the above-mentioned shortcomings, and which are especially dedicated to embedded control applications. The PID cores are described at RTL level. They are highly reconfigurable and technology-independent, offering the possibility to be mapped both on FPGA and ASIC.

To reach such a goal, a special focus was put on the optimization of the inner arithmetic of PID. For that, we considered two discrete forms of PID algorithm: the commercial form [22], called also the standard or ISA form, and the incremental form. These two forms went through three successive types of FPGA implementations, using: Booth multiplication algorithm (BMA) [23], modified Booth multiplication algorithm (MBMA) [24], and a new developed version called recursive multibit recoding multiplication algorithm (RMRMA) [25]. Results show gradual improvements with clear superiority over those provided in [21]. PID control-rate and energy-consumption savings are respectively as follows: 32% and 25% with BMA, 177% and 23% with MBMA, 431% and 20% with RMRMA.

Our previous paper [26] introduced a limited design-space of PID. In this paper, we extended the design-space to accommodate different application cases and

provided all necessary implementation details to make the design easily reproducible.

The paper is organized as follows. In this section we outlined the main requirement specifications for embedded PID controller. Section 2 introduces the two mostly-used discrete versions of PID algorithm. Sections 3, 4 and 5 deal with BMA, MBMA and RMRMA implementations, respectively. A discussion around the obtained results is given in Section 6. Section 7 describes the verification method, while Section 8 shows how the FWL-effect is tackled. Moreover, finally some concluding remarks in Section 9.

2 The two mostly-used discrete versions of PID

A typical closed-loop system using a PID controller is shown in Fig. 1, where $u_c(k)$, $y(k)$, and $u(k)$ are the discrete signal quantities at the kth sampling instant of the reference set-point, the process-feedback measured output, and the PID controller output, respectively.
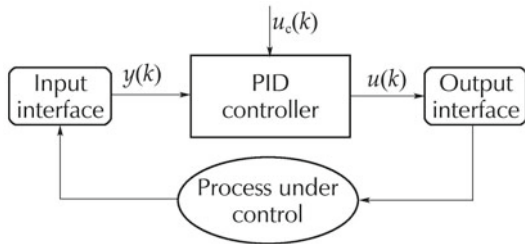

Fig. 1 Typical closed-loop control system using a PID.

In digital control, commercial and incremental forms are the two mostly-used discrete PID versions [1, 22]. They are denoted by recurrent equations (1) and (2), respectively, and their corresponding coefficients are grouped in Table 1. Equations (1) and (2) are fully detailed in the Appendix.

$$u(k) = P(k) + I(k) + D(k), \tag{1}$$

where

$$P(k) = A \cdot u_c(k) + B \cdot y(k),$$
$$I(k) = I(k-1) + C \cdot e(k-1),$$
$$D(k) = H \cdot D(k-1) + L \cdot f(k),$$

with

$$e(k-1) = u_c(k-1) - y(k-1),$$
$$f(k) = y(k) - y(k-1),$$

and

$$u(k) = u(k-1) + A \cdot e(k) + B \cdot e(k-1)$$
$$+ C \cdot e(k-2), \tag{2}$$

where

$$e(k) = u_c(k) - y(k),$$
$$e(k-1) = u_c(k-1) - y(k-1),$$
$$e(k-2) = u_c(k-2) - y(k-2).$$

Table 1 Coefficients of discrete recurrent equations.

Coefficients	Commercial PID	Incremental PID
A	$K_p b$	$K_p(1 + \dfrac{T_s}{T_i} + \dfrac{T_d}{T_s})$
B	$-K_p$	$-K_p(1 + 2\dfrac{T_d}{T_s})$
C	$-K_p\dfrac{T_s}{T_i}$	$K_p\dfrac{T_d}{T_s}$
H	$\dfrac{T_d}{T_d + NT_s}$	$-$
L	$-\dfrac{K_p T_d N}{T_d + NT_s}$	$-$

K_p is the proportional gain; T_i and T_d are the integral and derivative times, respectively; N is the maximum derivative gain; b is the fraction of set-point in proportional term; and T_s is the sampling period.

To satisfy different application cases, two IP versions are developed for each equation: with constant coefficients and with varying coefficients (Fig. 2). This latter requires a host side interface (HSI) to handle the runtime change of the coefficients.
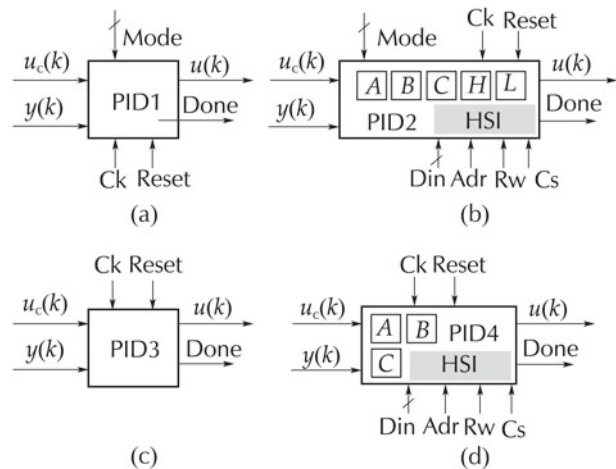

Fig. 2 Various PID IP-cores. (a) Commercial PID with constant coefficients; (b) commercial PID with time varying coefficients; (c) incremental PID with constant coefficients; and (d) incremental PID with time varying coefficients.

The commercial version allows the three standard PID functioning modes (P, PI, PID) according to Mode input value. At the end of $u(k)$ computation, the Done output signal toggles during one clock cycle, and the PID enters into sleep mode (whole internal activity stopped except for clocking and HSI) for maximum energy conservation.

3 MBA-based PID

A straightforward parallel implementation of PID requires an amount of 7 adders/substractors and 5 multiplication cores for equation (1), and 4 adders/substractors and 3 multiplication cores for equation (2). In digital hardware, the total gate count scales linearly with word length for an adder core, while it scales quadratically for a multiplier core. Thus, any effort for a low-power optimization of PID must be focused on the implementation of the multiply-and-accumulate (MAC) function $(X \cdot Y)$ [27]. In this work, the optimization effort is rather concentrated on the double MAC function $(X \cdot Y + T \cdot Z)$ called DMAC, considered as the main building block of our PID structures. Equations (1) and (2) are partitioned accordingly.

For FWL-PID, two's complement fixed-point representation is used, which is habitually expressed in Q notation as $Q_{n_i.n_f}$. The values are coded in n_i bits before the point (integer word length including 1 sign bit), and n_f bits after the point (fractional word length). The total word length is $n = n_i + n_f$.

Booth multiplication algorithm [23] belongs to the class of recoding algorithm, i.e., algorithms that recode one of the two operands to cope with signed two's complement multiplication. Let Y be the multiplier:

$$Y = -y_{n-1}2^{n-1} + \sum_{j=0}^{n-1} y_j 2^j. \tag{3}$$

Equation (3) can also be expressed as follows:

$$Y = \sum_{j=0}^{n-1} (y_{j-1} - y_j)2^j = \sum_{j=0}^{n-1} Q_j 2^j, \tag{4}$$

where $y_{-1} = 0$ and $Q_j \in \{-1, 0, 1\}$.

Consequently, the multiplier Y is divided into n slices, each of 2 bits. Each pair of two contiguous slices has one

bit in common. Thus, the DMAC becomes

$$X.Y + T.Z = \sum_{j=0}^{n-1} (Q_j.X)2^j + \sum_{j=0}^{n-1} (P_j.T)2^j \tag{5}$$

$$= \sum_{j=0}^{n-1} (Q_j.X + P_j.T)2^j. \tag{6}$$

According to (5), Booth algorithm consists in recoding the multiplier Y into a set of ternary numbers $\{-1, 0, 1\}$ in order to generate n simple partial products which are summed subsequently. Table 2 summarizes the 4 possibilities that may occur. The $-X$ can be easily formed by adding 1 to the complement of X. A direct translation of DMAC equation (5) into architecture (Fig. 3) requires one extra adder and two registers in comparison with the optimized version (Fig. 4) based on (6), called ODMAC. Additionally, one clock cycle latency is also needed in Fig. 3. The control-part responsible of producing the successive couples (y_{j-1}, y_j) is insignificant: just one multiplexer driven by a counter.

Table 2 Booth algorithm.

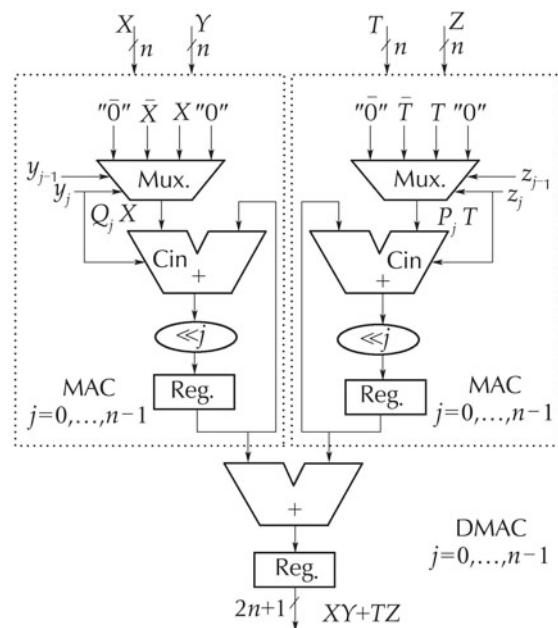| Y_j	Y_{j-1}	Operation
0	0	0
0	1	$+X$
1	0	$-X$
1	1	0

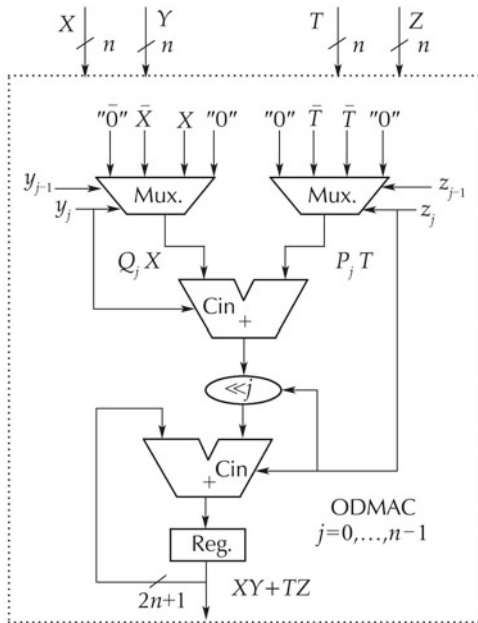Fig. 3 Straightforward DMAC implementation.

Fig. 4　Optimized DMAC implementation.

Based upon ODMAC as the main building block, PID architectures are constructed for both incremental (Fig. 5) and commercial (Fig. 6) forms, and their implementation results (Table 3) are respectively compared to those of [21]. Comparison was made into identical conditions using the same FPGA device (Spartan XC2S50E-7FT256), although relatively old, as well as the same synthesis-tool version (Xilinx ISE 9.1i). In [21], only a 16-bit word-length commercial version with constant coefficients (without HSI) is implemented. PID1 and PID3 exhibits interesting results: 44%, 25%, and 32% savings and 62%, 35%, and 38% savings in terms of gate count, power, and speed, respectively. PID3 exhibits higher savings but at the expense of control-quality. Latency is rather the same (17), which is $n + 1$ clock cycles for all designs (PIDX).

Optimizing latency without sacrificing the three other issues is the main objective of the next two sections.
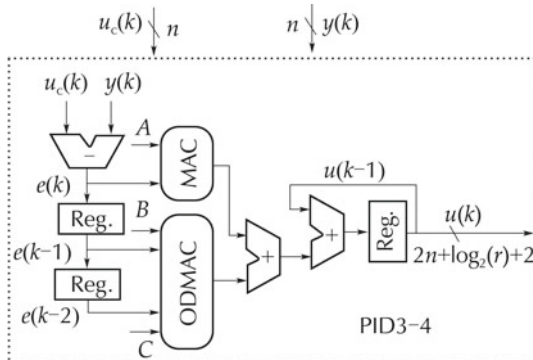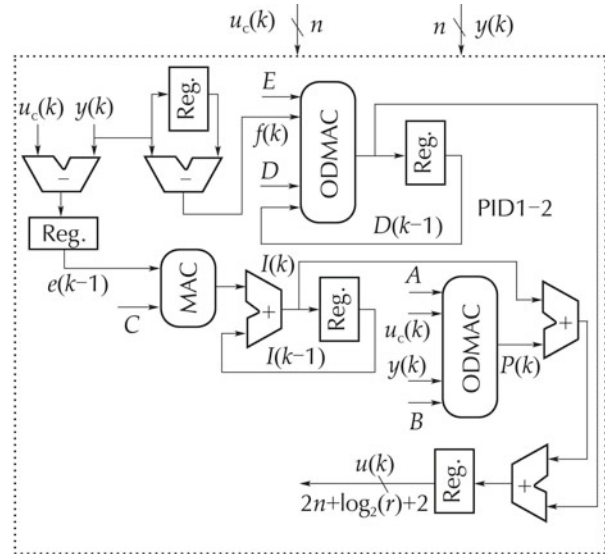

Fig. 5　Incremental PID architecture.

Fig. 6　Commercial PID architecture.

Table 3　Implementation result comparision of MBA-based PID.

PID core	Total gate count	Power* (mW)	Max. clock freq. (MHz)	Latency
PID [21]	16728	456	47	
PID1	9286 (44%)	342 (25%)	62 (32%)	
PID2	10661 (36%)	359 (21%)	61 (30%)	17
PID3	6337 (62%)	297 (35%)	65 (38%)	
PID4	7168 (57%)	308 (32%)	62 (32%)	

* Dynamic power consumption at 47 MHz; (XX%): saving.

4　MBMA-based PID

Equation (3) can also be rewritten as follows [24]:

$$Y = \sum_{j=0}^{\frac{n}{2}-1} (y_{2j-1} + y_{2j} - 2y_{2j+1})2^{2j} = \sum_{j=0}^{\frac{n}{2}-1} Q_j 2^{2j}, \quad (7)$$

where $y_{-1} = 0$ and $Q_j \in \{-2, -1, 0, 1, 2\}$.

In this case, the multiplier Y is divided into $n/2$ slices, each of 3 bits, with one bit overlapping between adjacent slices. The proof of equation (7) is given in [28]. Thus, the DMAC equation becomes:

$$X.Y + T.Z = \sum_{j=0}^{\frac{n}{2}-1} (Q_j.X + P_j.T)2^{2j}. \quad (8)$$

Likewise, $n/2$ simple partial products are generated (Table 4). Since ODMAC is a reconfigurable RTL block, it is parameterized to suit equation (8). The new adapted ODMAC architecture is depicted in Fig. 7. The only dif-

ference is that Mux(8:1) are used instead of Mux(4:1), and ($\ll 2.j$) hardwired shifter instead ($\ll 1.j$). Compared to BMA-based PID (Table 5), MBMA-based one (PID1) shows much more interesting results, since latency is divided by 2 while maintaining stable power consumption and speed. Control rate is drastically improved as its equal to maximum clock frequency divided by latency. As the discrete commercial form (equation (1)) can accommodate the three functioning modes, implementation of PID2 produced the following power consumption values at 47 MHz: 268 mW, 313 mW, and 366 mW for P, PI, and PID functioning modes, respectively.

With regard of these improvements, one is encouraged to pursue farther [24] in reducing latency by considering larger slices, such as

$$Y = \sum_{j=0}^{\frac{n}{3}-1} (y_{3j-1} + y_{3j} + 2.y_{3j+1} - 2^2 y_{3j+2}) 2^{3j} = \sum_{j=0}^{\frac{n}{3}-1} Q_j 2^{3j}, \quad (9)$$

where $y_{-1} = 0$ and $Q_j \in \{-4,\ldots,0,\ldots,4\}$.

However, in this case, some hard partial products are required such as $3X$ and $-3X$ which are not easy to generate. How to circumvent this obstacle is the purpose of the next section.

Table 4 Modified booth algorithm.

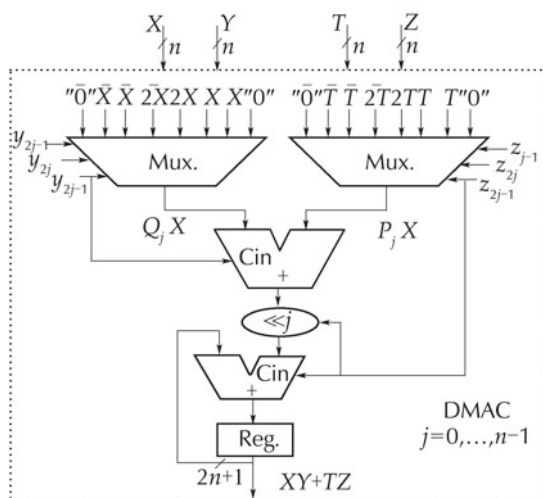| Y_{2j+1}	Y_{2j}	Y_{2j-1}	Operation
0	0	0	0
0	0	1	$+X$
0	1	0	$+X$
0	1	1	$+2X$
1	0	0	$-2X$
1	0	1	$-X$
1	1	0	$-X$
1	1	1	0

Fig. 7 Optimized DMAC architecture for $r = 2$.

Table 5 Implementation result comparison of MBMA-based PID.

PID core	Total gate count	Power* (mW)	Max. clock freq. (MHz)	Latency
PID [21]	16728	456	47	17
PID1	10642 (36%)	350 (23%)	62 (32%)	
PID2	11923 (29%)	366 (20%)	61 (30%)	
PID3	7042 (58%)	303 (33%)	64 (38%)	9 (47%)
PID4	7795 (53%)	315 (31%)	62 (32%)	

* Dynamic power consumption at 47 MHz; (XX%): saving.

5 RMRMA-based PID

Multiplication is a fundamental operation in digital design. Its speed and power requirements are two critical factors limiting the whole system performances (PID in our case). Since the publication of Booth's algorithm in 1951, a huge number of improvement attempts were proposed, especially after the publication of a generalized version of MBA algorithm accompanied with its proof [29]. Most of the proposals aimed to reduce the number of partial products either by employing digital optimization techniques [30–32] or by using larger slices (higher radices) [33]. However, experience showed [34] that beyond 4-bit slices (radix 8), the complexity to generate hard partial products cannot be managed in a realistic way. In [34], three metrics are provided for comparing the tradeoffs when employing higher radix Booth recodings: partial product compression factor (gain), the number of hard multiples that must be precomputed (computation complexity), and partial product generation fanin (routing complexity).

To circumvent the problem of hard partial products in higher radices, the idea proposed in [35] is to apply a recursive Booth recoding on the r-bit slice. While the idea is interesting, it relies upon a complicated mathematical formulation, leading to a complex control circuitry and especially to an exaggerated latency ($2n/r$).

According to the multibit recoding algorithm presented in [29], a n-bit two's complement operand Y can be written as

$$Y = \sum_{j=0}^{\frac{n}{r}-1} (y_{rj-1} + 2^0 y_{rj} + 2^1.y_{rj+1} + 2^2 y_{rj+2} + \ldots$$
$$+ 2^{r-2} y_{rj+r-2} - 2^{r-1} y_{rj+r-1}) 2^{rj}$$

$$= \sum_{j=0}^{\frac{n}{r}-1} Q_j 2^{rj}, \tag{10}$$

where $y_{-1} = 0$, $r \in N^*$ and $Q_j \in \{-2^{r-1}, \ldots, 0, \ldots, 2^{r-1}\}$.

In this general case, the multiplier Y is divided into n/r slices, each of $r+1$ bits. Each pair of two contiguous slices has one overlapping bit. To bypass the problem of hard partial products, MBMA (equation (7)) is applied to the Q_j terms. Thus, equation (10) takes the new simpler recursive form:

$$Y = \sum_{j=0}^{\frac{n}{r}-1} [(y_{rj-1} + y_{rj} - 2.y_{rj+1})2^0$$
$$+ (y_{rj+1} + y_{rj+2} - 2.y_{rj+3})2^2 + \ldots$$
$$+ (y_{rj+r-5} + y_{rj+r-4} - 2.y_{rj+r-3})2^{2(\frac{r}{2}-2)}$$
$$+ (y_{rj+r-3} + y_{rj+r-2} - 2.y_{rj+r-1})2^{2(\frac{r}{2}-1)}]2^{rj} \tag{11}$$

$$= \sum_{j=0}^{\frac{n}{r}-1} [\sum_{i=0}^{\frac{r}{2}-1} (y_{rj-1+2i} + y_{rj+2i} - 2.y_{rj+1+2i})2^{2i}]2^{rj} \tag{12}$$

$$= \sum_{j=0}^{\frac{n}{r}-1} [\sum_{i=0}^{\frac{r}{2}-1} Q_{ji} 2^{2i}]2^{rj} \tag{13}$$

with $Q_{ji} \in \{-2, -1, 0, 1, 2\}$.

There is no need to prove equation (11) since it is a combination of equations (10) and (7) which were already proven in [29] and [28], respectively. The partitioning of operand Y according to equation (13) is illustrated by Fig. 8.
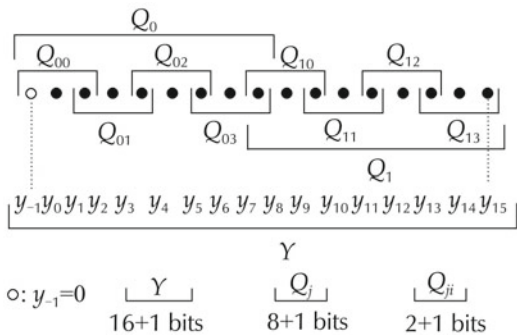

Fig. 8　Partitioning of a 16-bit Y operand with $r = 8$.

To avoid dealing with special cases, n and r must be chosen as even numbers, with r as a divider of n. Thus, the DMAC equation becomes

$$X.Y + T.Z = \sum_{j=0}^{\frac{n}{r}-1} [\sum_{i=0}^{\frac{r}{2}-1} (Q_{ji}.X + P_{ji}.T)2^{2i}]2^{rj}. \tag{14}$$

Depending on r value ranging from 2 to n, PIDs

with various levels of parallelism and latencies ($n/r + 1$) can be automatically generated with slight control complexity. The special cases of $r = n$ and $r = 2$ correspond to fully-parallel and fully-sequential PID, respectively. In between ($r = 4, n/2$), partially-parallel PIDs are obtained. The outstanding advantage of this algorithm (equation (13)) is that hard partial products are generated using simple ones ($2X, X$) only. For a simplified hardware and lower power consumption, the step-by-step sign-propagate technique is employed [36].

Obviously, equation (13) does not reduce the number of partial products, but allows a modulable space-time partitioning of the multibit recoding algorithm (equation (10)), where n/r sets comprising each $r/2$ partial products can be generated and summed either simultaneously or iteratively. Whilst the parallel implementation of equation (13) allows an important reduction of the critical path (using a carry-save adder CSA), it requires too much space. Therefore, only the serial implementation is retained. In this case, latency drops from ($n/2+1$) to ($n/r + 1$), whereas the overhead on the total critical path, which goes through $\log_2(r/2)$ adder levels and which is equal to D in the case of MBMA, is slightly increased $D + \log_2(r/2)$. Note that we are using a logarithmic summation tree and not a linear one (CSA like).

An illustrative serial example with $r = 4$ is described as follows:

$$Y = \sum_{j=0}^{\frac{n}{4}-1} (y_{4j-1} + y_{4j} + 2y_{4j+1} + 2^2 y_{4j+2} - 2^3 y_{4j+3})2^{4j} \tag{15}$$

$$= \sum_{j=0}^{\frac{n}{4}-1} [\sum_{i=0}^{1} (y_{4j-1+2i} + y_{4j+2i} - 2.y_{4j+1+2i})2^{2i}]2^{4j} \tag{16}$$

$$= \sum_{j=0}^{\frac{n}{4}-1} [Q_{j0} + Q_{j1} 2^2]2^{4j}, \tag{17}$$

$$X.Y + T.Z = \sum_{j=0}^{\frac{n}{4}-1} [(Q_{j0}X + P_{j0}T) + (Q_{j1}X + P_{j1}T)2^2]2^{4j}. \tag{18}$$

The mapping of equation (18) into a serial architecture is shown in Fig. 9. Such a case ($r = 4$) would have required the computation of hard partial products ($7X, 5X, 3X$) if the simple form of equation (15) was used. Notice that MBMA is a special case of RMRMA for $r = 2$. For $r = 1$, equation (10) corresponds to BMA (equation (4)).
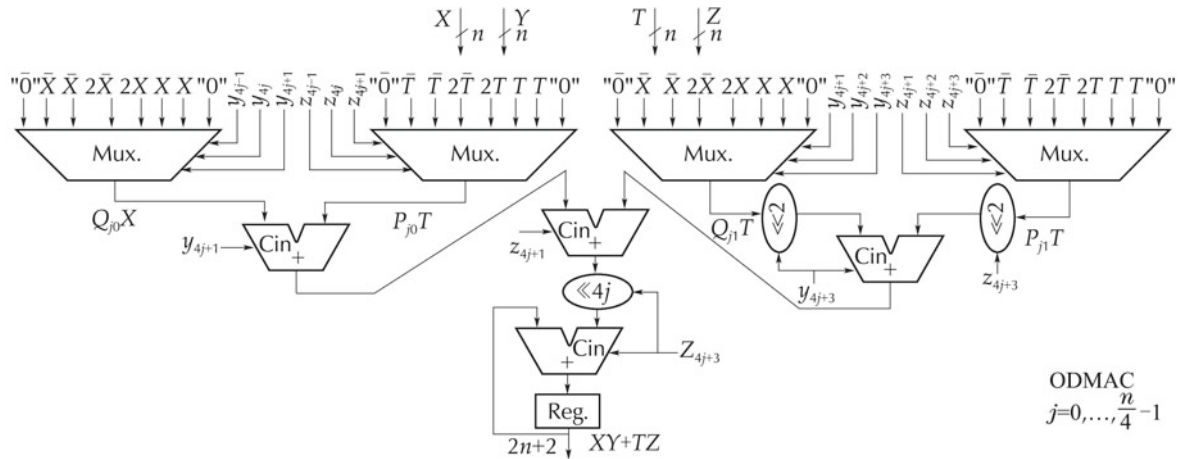
Fig. 9 Optimized DMAC architecture for $r = 4$.

Table 6 comprises the implementation results of PIDs with $n = 16$ and $r = 4, 8, 16$. For instance, PID1 with $r = 4$ not only achieves high improvement in latency (71%), but also maintains positive savings in power (14%) and speed (13%). These important achievements are partially due to logic-trimming performed by the synthesis tool on the constant coefficients. Such an operation is impossible in the case of PID [21] since the coefficients are stored into LUTs.

Table 6 Implementation result comparison of RMRMA-based PID.

PID core	Total gate count	Power* (mW)	Max. clock freq. (MHz)	Latency
PID [21]	16728	223	47	17
PID1_4	12443 (+26%)	191 (+14%)	53 (+13%)	5 (+71%)
PID1_8	15688 (+06%)	194 (+13%)	44 (−06%)	3 (+82%)
PID1_16	23545 (−41%)	217 (+03%)	26 (−45%)	2 (+88%)
PID2_4	22962 (−37%)	256 (−15%)	43 (−08%)	5 (+71%)
PID2_8	26073 (−56%)	204 (+08%)	37 (−21%)	3 (+82%)
PID2_16	40327 (−141%)	488 (−119%)	23 (−51%)	2 (+88%)

*: Dynamic power consumption at 23 MHz; PID$Y\_X : X = r$; and (+AB%): saving; (−AB%): overhead.

At this stage, a key question arises: among this panoply of PIDs, which one fits the best one's application case? The answer to this question is given in the next section.

6 Discussion

In embedded control, satisfactory control-rate (without performance degradation) at minimum power consumption is the main requirement. To select the most adequate PID for a given application, it is necessary to investigate how speed, power and hardware resources scales versus r factor for a fixed word length n. Referring to equation (14) and aided by Fig. 9, the ODMAC architecture scales as a binary tree with one stage of r mux(8:1) followed by $\log_2(r) + 1$ stages of adders with a total of r adders too. Thus, the total delay cumulated by the critical path which goes through $\log_2(r) + 2$ stages increases with $O(\log(r))$ complexity, whilst latency ($n/r + 1$) decreases linearly $O(r)$, which makes the maximum control-rate increases as r increases. This is confirmed by implementation results shown in Tables 7 and 8 corresponding to PID1 and PID2, respectively. The sole exception to this general rule is PID$X\_n/2$ which always yields to the highest control-rate compared to PIDX_n despite the numerous tests with various n values. This is justified since they exhibit very close latencies (3 and 2, respectively) and one stage difference in the critical path ($n − 1$ and n, respectively), but an important multiplexer fanin difference ($n/4$ and $n/2$, respectively).

Table 7 Maximum power-consumption and control-loop-cycle of PID1.

PID core	Power* (mW)	Max. clock freq. (MHz)	Latency	Max. control loop cycle (MHz)
PID [21]	456	47	17	2.76
PID1_1	342 (+25%)	62	17	3.65 (+32%)
PID1_2	350 (+23%)	62	9	7.66 (+177%)
PID1_4	431 (+05%)	53	5	10.60 (+284%)
PID1_8	365 (+20%)	44	3	14.67 (+431%)
PID1_16	244 (+46%)	26	2	13.00 (+371%)

* Dynamic power consumption at maximum clock frequency; PID1$\_X : X = r$; and Max. control loop cycle = Max. clock frequency/Latency.

Table 8 Maximum power-consumption and
control-loop-cycle of PID2.

PID core	Power* (mW)	Max. clock freq. (MHz)	Latency	Max. control loop cycle (MHz)
PID [21]	456	47	17	2.76
PID2_1	466 (−02%)	61	17	3.59 (+30%)
PID2_2	475 (−04%)	61	9	6.78 (+146%)
PID2_4	479 (−05%)	43	5	8.60 (+211%)
PID2_8	328 (+28%)	37	3	12.33 (+347%)
PID2_16	488 (−07%)	23	2	11.50 (+317%)

* Dynamic power consumption at maximum clock frequency; PID2_X : $X = r$; and Max. control loop cycle = Max. clock frequency/Latency.

In terms of resource occupation, the total complexity grows linearly O(r) as r multiplexers and r adders are required by ODMAC which is the most resource consuming block of PID architecture. This is also confirmed by the implementation results shown in Table 6. Note that each adder of each level of MAC and ODMAC as well as the two ones at the output of the PID (Figs. 5 and 6) are successively extended by one bit so that the total bit size of the control output $u(k)$ becomes $2n + \log_2(r) + 2$. It's necessary to do so to prevent the apparition of a possible overflow in the data-path which can cause signal clipping and instabilities in the closed loop response [37].

As for power consumption, intuitively, one would expect to see PID1_16 of Table 7 as being the most rapid and the most power consumer too, for the reason that it exhibits the smallest latency and the biggest total gate count! While it is almost true for the latter (13 MHz, before the first), it is quite the opposite for the former (244 mW, the smallest one). The explanation is that power consumption ($P = 0.5V_{dd}^2 C_{sw} F_{clk}$) depends linearly on the frequency (F_{clk}), which is in this case 26 MHz (the smallest one) and also on the switched capacitance (C_{sw}) which describes the average capacitance charged during each clock period ($1/F_{clk}$). In fact, C_{sw} depends on a number of parameter (circuit structure, logic function, input pattern dependence, ...) and not only on the total gate count (more precisely, not only on the total physical capacitance of the circuit). Furthermore, a study [38] that analyzed the dynamic power consumption in Xilinx's FPGA revealed the following share: 60% by routing, 16% by logic, and 14% by clocking. The reason is that routing is intensively segmented, using pass logic and buffers.

When both high control-rate close to 13 MHz and low power are required, PID1_16 (244 mW at 13 MHz) stands as the best candidate compared to PID1_8

(323 mW at 13 MHz). However, it is noteworthy to mention that this comparison stands valid only for the special case of 16-bit word-length PID, for a given set of coefficients, mapped on XC2S150E-7FT256 FPGA circuit and using Xilinx's XST synthesis tool, version 9.2. Results could significantly change under other conditions, especially when considering the logic trimming process which is essentially dependant on the bit-arrangement of the coefficients. For a minimum influence of the trimming operation on the synthesized results, appropriate coefficients were used such as all Q_j terms are represented except the null one to avoid generating null partial products that greatly simplify the circuit logic. In fact, constant coefficients PIDs (PID1) are somehow unpredictable with regard to r. They are coefficient dependant. Adversely, PID2 is not involved with the trimming process since coefficients are time varying. Implementation results comprised in Table 8 show that PID2_8 is the best at all aspects for the same reasons cited above. In sum, when high control-rate is the ultimate objective, PIDX_n/2 is the best candidate whatever n value. However, in the case where both high speed and low power are required, timing and power evaluations are necessary to decide which PID to select: either PIDX_n/2 or PIDX_n.

Finally, when only low power is targeted, PIDX_1 is the best candidate. We dealt here with extreme situations only, but for a given couple (cr, pc) of control-rate and power consumption, several candidates are possible. Yet, the best PID is the one which requires the smallest gate count.

So far, speed and power have been considered in isolation to area which becomes critical, and sometimes prohibitive, for large word-length n due to the fact that PID is basically built of a set of multipliers (three or five) that scale quadratically with word length. The bigger is the area, the higher is the cost. Consequently, another advantage of RMRMA algorithm is to cope also with the cost issue as an additional constraint to speed and power.

We deliberately chose Spartan2e FPGA to compare our results with those provided in [21]. A mapping on a recent FPGA circuit (Virtex6) using XST 12.1 version of extreme PID2 delivered state-of-the-art results grouped in Table 9. Note that control-rate scaled with an average factor of 2, while power dissipation scaled with an average factor of 45.

This is not surprising, since Spartan2e and Virtex6 were fabricated with two differently scaled technology processes: 150 nm and 40 nm, respectively. Therefore, the physical capacitances of the circuit in Virtex6 are relatively too much smaller. Additionally, the supply-

voltages (V_{dd}) used for internal core (V_{ccint}) and for output blocks (V_{cco}) are respectively 1.8 V and 3.3 V for Spartan2e, 1 V and 2.5 V for Virtex6. Furthermore, the efficient advances made in CAD tools (from Xilinx ISE 9.1 to 12.1 versions) as well as in FPGA architecture, such as advanced segmented-routing, much contributed to lower the power consumption [39]. Power consumption evaluation studies [38, 39] based on simulation and measurements, targeting Virtex2 and Virtex6 families revealed the following results: 5.9μW per CLB per MHz, and 1.09 mW per 100 MHz at 38% toggle rate, respectively. These studies roughly confirm our power results as proximate values are obtained.

Timing and power evaluations were performed in the following conditions. Delays were calculated for two types of paths: Clock-To-Setup and all paths together (Pad-To-Setup, Clock-To-Pad and Pad-To-Pad.) The Clock-To-Setup gives more precise information on the delays than other remaining paths, which depend in fact on I/O Block (IOB) configuration (low/high fanout, CMOS, TTL, LVDS, …). Thus, all delays (frequencies) presented so far are clock-to-setup delays with the highest speed grade of the FPGA circuit. As for power, we chose the highest Vcco voltage value (3.3 for Spartan2e and 2.5 for Virex6) with a maximum toggle activity of 50%, which means that Flip-Flops (FFs) toggle one time during each clock cycle. The reason is that only simple-edge triggered FFs are used for synthesis (no double-edge FFs).

Table 9 Maximum power-consumption and control-loop-cycle of PID2 mapped on Virtex6.

PID core	Number of slices	Power* (mW)	Max. clock freq. (MHz)	Latency	Max. control loop cycle (MHz)
PID_1	231	23	122	17	7.17
PID2_8	1060	4	90.5	3	30.16
PID2_16	1963	13	50.4	2	25.19

* Dynamic power consumption at maximum clock frequency; PID2_X : $X = r$; and Max. control loop cycle = Max. clock frequency/Latency.

7 Verification method

The PID design verification process went through several steps. First, equations (12) and (14) were tested with a random C-program. Then, a severe cycle-accurate functional verification procedure using Modelsim simulator was applied to MAC and ODMAC as they are the main building blocks of PID architecture. They were challenged against a set of special test cases (visual

simulation), and then submitted to a random test for a very large number of vectors. Once tested successfully, the RTL PID module written in Verilog-2001 (IEEE 1364) was integrated into Modelsim/Simulink environment for a co-simulation. At this stage, a ZOH discrete time invariant model of a third order continuous process ($G(s) = 1/(s + 1)^3$) was chosen from the test set used by Åström and Hägglund [1] as examples of representative plants for the dynamics of typical industrial processes. To derive the PID parameters, a theoretical PID taken from MATLAB component-library was tuned using floating-point numerical representation (IEEE 754 double format). The sampling period T_s was chosen based on the magnitude of the pole time constants. For this case $T_s = 10$ ms. The following parameters were obtained:

K_p = 0.5913; T_i = 0.0523; T_d = 0.0225 for N = 10 and b = 1. Calculations give the following floating-point values for the coefficients of commercial PID:

A = 0.5913; B = −0.5913; C = 0.1130; D = 0.1836; E = −1.0860.

To co-simulate the RTL PID, a conversion of the coefficients to 16-bit ($Q_{4.12}$) fixed-point representation was necessary. Variations were obtained:

A = 0.5911; B = −0.5911; C = 0.1130; D = 0.1836; E = −1.0860.

Note that to represent the original parameters with full-precision, 44 bits are needed for the fractional part. Varied simulations were performed to verify the correctness of the PID RTL code. First, to explore the precision effect on control quality, the control output of PIDs with various fractional-part sizes ($Q_{4.4}$, $Q_{4.12}$, $Q_{4.20}$) were compared to that of the MATLAB floating-point PID component (Fig. 10). Simulation shows different rise-times for different precisions. The higher is the precision; the closer is the control output from the ideal model. The second simulation tests the behavior of the PID after having reached the steady state (Fig.11). For that, two perturbations are successively exerted on control output and on the plant measure. Each time the system recovers as expected. Moreover, finally, the third simulation investigates the PID capabilities to track set-points of arbitrary amplitudes and durations (Fig. 12).

After a successful functional verification, the RTL code of PID was synthesized, placed, and routed on Xilinx's FPGA (Virtex-2). The three preceding co-simulations but with timing backannotation were performed again as a last necessary software verification step before hardware integration of the PID into an FPGA evaluation board (MEMEC V2MB1000).

Finally, as an ultimate validation step, a physical test

of our PIDs is performed. We built up a classical temperature control setup (Figs. 13 and 14), which consists in a tube comprising a halogen lamp (12 V, 21 W), a temperature sensor (LM35), and a DC Fan (12 V, 1.68 W). Temperature regulation inside the tube is achieved by controlling either the intensity of the lamp, or the rotation speed of the fan. This is carried out by the use of two PWMs, whose duty-cycle ratios represent the PID controller output ($u(k)$). These two PWMs do not act directly on the fan or on the lamp but rather on transistors (IRF540) that control the power consumed by the lamp and fan.
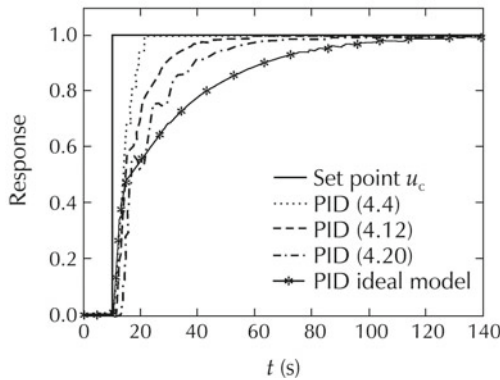

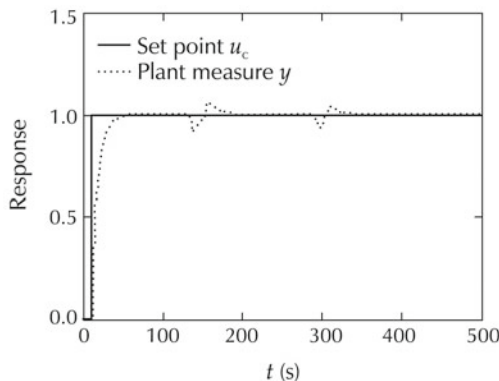Fig. 10 Fixed-point versus floating-point.

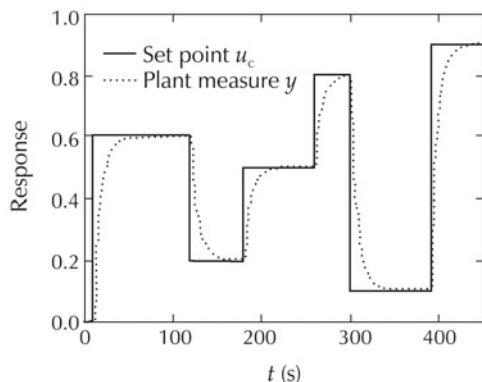Fig. 11 Perturbations after steady-state on control-output and on plant measure, successively.

Fig. 12 Set-point tracking of arbitrary amplitudes and durations.

The sensing of the actual temperature of the tube is assured by LM35 component which delivers a voltage value that grows linearly with temperature (1.5 V corresponds to 150°C). As the maximum voltage allowed by FPGA evaluation board (V2MB1000) is 3.3 V, the calculation of the real temperature (T) is done as follows: $T = [(\text{val\_opb\_ADC} \times 3.3)/1023] \times 100$. This allows a temperature control with a minimum step of 0.32°C.

The V2MB1000 board is connected through RS232 port to a PC running a .net application which allows a real-time display of the temperature as well as an instantaneous tuning of the set-point.
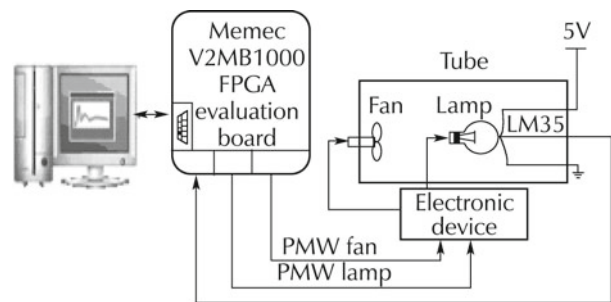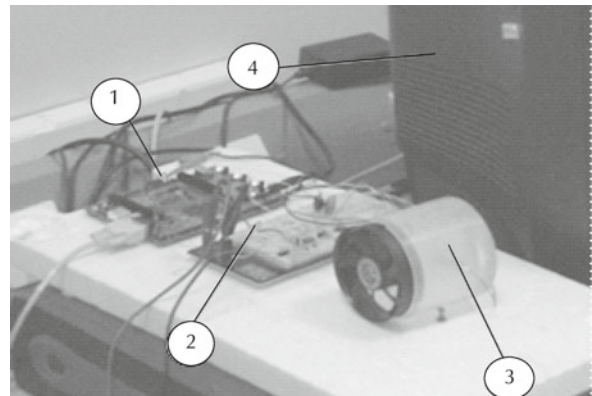

Fig. 13 Synoptic scheme of the setup.

Fig. 14 Setup of temperature regulation. ①: FPGA evaluation board; ②: Electronic device; ③: Tube containing a fun and a lamp; and ④: PC display screen.

8 The FWL effect

Fixed-point arithmetic is employed as an approximation of real numbers (floating-point), with a fixed bit-length of the word used to represent data (finite word-length). This limitation leads to performance degradation (FWL effect) mainly due to quantization of coefficients (parametric errors) and roundoff errors subsequently cumulated during the computation process (numeric noise). In fact, the FWL effect is more-or-less exaggerated depending on the control algorithm used (I/O

relationship, levels of parallelism, etc.) as well as on the way the computations are performed (number of bits, different/unique fixed point position, round/truncation, etc.). Compared to the reference floating-point implementation, the FWL effect can be assessed using some indicators such as transfer function sensitivity, or pole sensitivity [40–42].

In fact, the objective is twofold: we need to provide an optimal ASIC/FPGA implementation of FWL PID without degrading control performances. To achieve such a goal, a double expertise is required in hardware design and control system. However, usually, hardware designers do not master control system design, and control system experts do not have the required skills to implement and evaluate the controllers using ASIC/FPGAs [17, 43]. This is why we propose, as hardware designers, a highly reconfigurable (n, r) and technology-independent FWL PID that can systematically respond to control-engineer demands after having modelled, simulated, and evaluated the performances provided by different bit-width fixed-point representations using MATLAB/Simulink environment, and finally opted for an appropriate word-length (n) of the setpoint. As for latency value (r), it depends on the application domain and intended objectives. Precise guidelines on how to choose r value were given in Section 6.

Now that (n, r) couple is known, the FWL problem is tackled from hardware side by simply adjusting in the RTL code the two compile-time constants: setpoint bit-size (n) and latency (r). The synthesis of such a PID generates an optimal structure that not only meets the performances specified by control-engineers, but also consumes minimum power and hardware resources. This would not have been possible without the use of the new highly serialisable multi-bit multiplication algorithm (equation (13)). The incorporation of equation (13) [25] into equations (1) and (2) as an efficient PID engine, allows the generation of PID architectures classified as regular iterative architectures (RIA) [44], known for their high conformity with the principles of regularity and locality. In addition to equation (13), we propose in [25] several new highly serialisable multiplication algorithms, offering different features in power, space and delay, depending on the operand size (n). Reader is encouraged to explore these algorithms [25] to select the appropriate one that leads to best performances of its controller with regard to the size (n) of the setpoint.

Regularity and locality are two important features, highly sought in hardware design as they lead to an important gain in space and delay. Regularity is a general space feature, where the repetitiveness of just one or few elementary building-blocks (mux, adders and shifters of ODMAC, Fig. 9) and their interconnection scheme (predefined netlist) suffice to draw the whole architecture (MAC/ODMAC and then PID). In the other hand, locality is both space and time feature, in the sense where each building-block can only interact with its nearest surrounding neighbours, and any transaction from one building-block to the next is completed in one and only one unit time delay (clock period). Because of these two important features, our PIDs can be finely grained at bit level in space (setpoint bit-size n, latency r) and unit delay in time (latency r).

Experimental results depicted in Fig. 15 illustrate the FWL effects on temperature regulation. Reducing the fractional-part size of the set-point beyond a certain limit (4 bits) yields to a continuous fluctuation of the temperature inside the tube (Fig. 15 (d)). The best compromise is a 6-bit fractional-part (Fig. 15 (c)) which ensures a correct regulation while consuming less power and hardware resources. As temperature regulation system has a very slow dynamic, speed is not a concern. Therefore, the most appropriate PID in this case is PIDX_1 as it is the least power consumer. Adversely, for very fast dynamic systems, such as MEMS [45] or microrobotics applications [46], PID$X\_n/2$ is the most adequate option as it leads to the highest control rate.
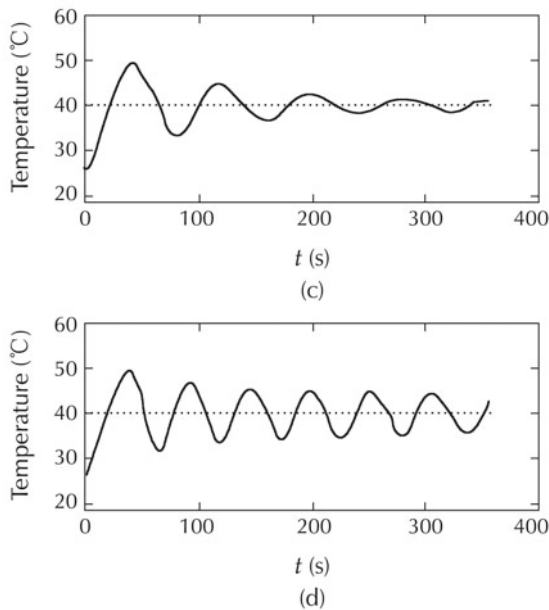

(a)

(b)
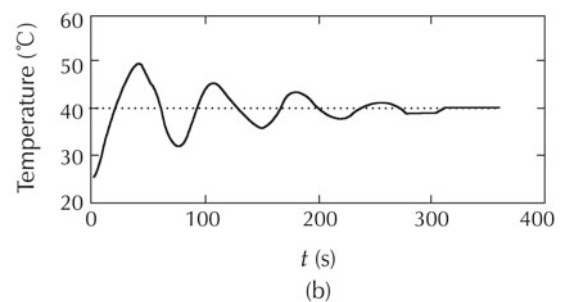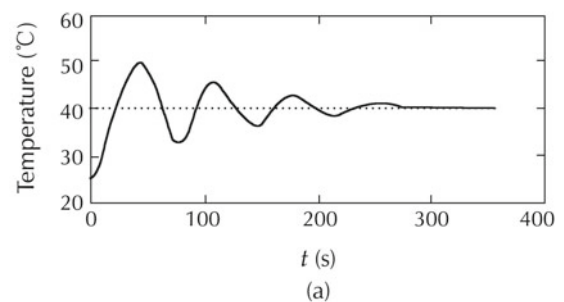
Fig. 15   Effect of the setpoint fractional length on temperature regulation. (a) Floating point PID; (b) our PID with $Q_{n_i.n_f} = Q_{8.8}$; (c) our PID with $Q_{n_i.n_f} = Q_{8.6}$; and (d) our PID with $Q_{n_i.n_f} = Q_{8.4}$.

9   Conclusions

Despite the large popularity of PID controller, little attention has been paid to its optimization, either for ASIC or for FPGA integration. To break down this paradoxical situation, a series of high-speed and low-power PIDs, especially dedicated to embedded applications was proposed. They are based on two discrete forms of PID algorithm: the incremental form and the commercial form, both with constant and time-varying coefficients. The work focused more particularly on the commercial form with varying coefficients as it is the most used in industry due to the higher control-quality provided. Two types of optimizations were carried out: architectural and algorithmic optimizations. The former is a macro-level optimization, based on an efficient partitioning of PID discrete-equations, considering the double MAC (DMAC= $XY + ZT$) as the main building block of PID architecture. An optimized version of DMAC was developed (ODMAC) for less hardware resource occupation. As for the micro-level optimization (inner optimization of ODMAC), three multiplication algorithms were experienced: BMA, MBMA, and a new general and recursive version of MBMA called RMRMA. In addition, some low-power design techniques were incorporated, such as sleep mode, and step-by-step sign-propagation technique.

The implementation results of PID based upon these three algorithms yielded to gradual improvements with a clear superiority over results presented in [21]. For instance, concerning PID1_2 and PID1_4, savings of 177%, 23%, and 36%, and savings of 284%, 14%, and 26% are obtained in control-rate, power consumption, and total gate count, respectively. Additionally, analytical scaling-complexity evaluations with respect to the couple (n, r), confirmed also by software simulations, revealed useful information which is summarized as follows:

. PIDX_n/2 is the fastest PID that yields to the highest control-rate (30 MHz for PID2_8 mapped on Virtex6, with $(n, r) = (16, 8)$);

. PIDX_1 is the most power efficient PID when speed is not a concern;

. PIDX_n and PIDX_n/2 are the most efficient PIDs when both high control-rate and low-power dissipation are required.

Further extension to the present work is to apply the same (or appropriate) partitioning in conjunction with RMRMA algorithm to the set of recurrent equations of an arbitrary number of multi-loop PID controllers taken as a whole.

Finally, the new recursive multiplication algorithm (RMRMA), well adapted to large word-lengths, and which was behind the drastic optimization of PID, can be efficiently applied to a variety of advanced control algorithms such as to linear-quadratic-gaussian (LQG) or sliding-mode controllers, etc.

References

[1] K. Åström, T. Hägglund. *PID Controllers: Theory, Design, and Tuning.* 2nd ed. by the Research Triangle Park: Instrument Society of America, 1995.

[2] D. Xue, Y. Chen, D. P. Atherton. *Linear Feedback Control.* Philadelphia: Society for Industrial and Applied Mathematics (SIAM), 2007.

[3] X. Shao, D. Sun, J. K. Mills. A new motion control hardware architecture with FPGA based IC-design for robotic manipulators. *IEEE International Conference on Robotics and Automation (ICRA).* New York: IEEE, 2006: 3520 – 3525.

[4] J. S. Kim, H. W. Jeon, S. Jeung. Hardware implementation of nonlinear PID controller with FPGA based on floating point operation for 6-DOF manipulator robot arm. *International Conference on Control, Automation and Systems (ICCAS).* Piscataway: IEEE, 2007: 1066 – 1071.

[5] L. Qu, Y. Huang, L. Ling. Design and implementation of intelligent PID controller based on FPGA. *Proceedings of the 4th International Conference on Natural Computation (ICNC).* Piscataway: IEEE, 2008: 511 – 515.

[6] M. Keating, P. Bricaud. *Reuse Methodology Manual for System on a Chip Designs*. 3rd ed. New York: The Kluwer Academic Publishers, 2002.

[7] International Technology Roadmap for Semiconductors. *ITRS Reports*, 2007/2008: http://www.itrs.net/reports.html.

[8] T. Hilaire, P. Chevrel, J. F. Whidborne. A unifying framework for finite wordlength realizations. *IEEE Transactions on Circuits and Systems*, 2007, 54(8): 1765 – 1774.

[9] T. Hilaire, D. Ménard, O. Sentieys. Bit accurate roundoff noise analysis of fixed-point linear controllers. *Proceedings of the IEEE International Conference on Computer-Aided Control Systems (CACSD)*. New York: IEEE, 2008: 607 – 612.

[10] S. Gretlein, G. Garcia, J. Sumner. DSPs, microprocessors and FPGAs in control. *Magazine of Record for the Embedded Computing Industry (RTC Magazine)*, 2006: http://www.rtcmagazine.com/articles/view/100495.

[11] E. Manmasson, L. Idkhajine, M. N. Cirstea, et al. FPGA in industrial control applications. *IEEE Transactions on Industrial Informatics*, 2011, 7(2): 224 – 243.

[12] S. Chander, P. Agarwal, I. Gupta. FPGA-based PID controller for DC-DC converter. *Proceedings of the IEEE Joint International Conference on Power Electronics, Drives and Energy Systems (PEDES)*. Piscataway: IEEE, 2010.

[13] S. Yang, M. Gao, J. Lin, et al. The IP core design of PID controller based on SOPC. *Proceedings of the IEEE International Conference on Intelligent Control and Information Processing*. Piscataway: IEEE, 2010: 363 – 366.

[14] J. Lazaro, A. Astarloa, J. Arias, et al. Simulink/Modelsim simulable VHDL PID core for industrial SoPC multiaxis controllers. *Proceedings of the IEEE 32nd Annual Conference on Industrial Electronics (IECON)*. Piscataway: IEEE, 2006: 3007 – 3011.

[15] F. Fons, M. Fons, E. Canto. Custom-made design of a digital PID control system. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Piscataway: IEEE, 2006: 1020 – 1023.

[16] B. V. Sreenivasappa, R. Y. Udaykumar. Design and implementation of FPGA based low power digital PID controllers. *Proceedings of the IEEE International Conference on Industrial and Information Systems (ICIIS)*. New York: IEEE, 2009: 568 – 573.

[17] J. Lima, R. Menotti, J. M. P. Cardoso, et al. A methodology to design FPGA-based PID controllers. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*. Piscataway: IEEE, 2006: 2577 – 2583.

[18] I. Urriza, L. A. Barragan, J. I. Artigas, et al. Word Length Selection Method based on Mixed Simulation for Digital PID Controllers Implemented in FPGA. *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE)*. Piscataway: IEEE, 2008: 1965 – 1970.

[19] W. Zhao, B. H. Kim; A. C. Larson, et al. FPGA implementation of closed-loop control system for small-scale robot. *Proceedings of the IEEE 12th International Conference on Advanced Robotics (ICAR)*. Piscataway: IEEE, 2005: 70 – 77.

[20] L. Samet, N. Masmoudi, M. W. Kharrat, et al. A digital PID controller for real-time and multi-loop control: a comparative study. *Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. Piscataway: IEEE, 1998: 291 – 296.

[21] Y. Fong, M. Moallem, W. Wang. Design and implementation of modular FPGA-based PID controllers. *IEEE Transactions on Industrial Electronics*, 2007, 54(4): 1898 – 1906.

[22] B. Wittenmark, K. J. Astrom, K. E. Arzenin. *Computer Control: An Overview*. Lund, Sweden: Lund Institute of Technology, 2003: http://www.control.lth.se/kursdr/ifac.pdf.

[23] A. D. Booth. A signed binary multiplication technique. *The Quarterly Journal of Mechanics and Applied Mathematics*, 1951, 4(2): 236 – 240.

[24] O. L. MacSorley. High-speed arithmetic in binary computers. *Proceedings of the IRE*, 1961, 49(1): 67 – 91.

[25] A. K. Oudjida, N. Chaillet, A. Liacha, et al. A New recursive multibit recoding algorithm for high-speed and low-power multiplier. *Journal of Low Power Electronics (JOLPE)*, 2012, 8(5): 1 –16.

[26] A. K. Oudjida, N. Chaillet, A. Liacha, et al. High-speed and low-power PID structures for embedded applications. *Proceedings of the 21st Edition of the International Workshop on Power and Timing Modeling, Optimization and Simulation PATMOS*. Berlin: Springer-Verlag, 2011: 257 – 266.

[27] Y. H. Seo, D. W. Kim. A new VLSI architecture of parallel multiplier-accumulator based on radix-2 modified Booth algorithm. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2010, 18(2): 201 – 208.

[28] L. P. Rubinfield. A proof of the modified booth algorithm for multiplication. *IEEE Transactions On Computers*, 1975, 24(10): 1014 – 1015.

[29] H. Sam, A. Gupta. A generalized multibit recoding of two's complement binary numbers and its proof with application in multiplier implementations. *IEEE Transactions on Computers*, 1990, 39(8): 1006 – 1015.

[30] F. Lamberti. Reducing the computation time in (short bit-width) two's complement multiplier. *IEEE Transactions on Computers*, 2011, 60(2): 148 – 156.

[31] S. R. Kuang, J. Wang, C. Guo. Modified booth multipliers with a regular partial product array. *IEEE Transactions on Circuit and Systems*, 2009, 56(5): 404 – 408.

[32] J. Kang, J. L. Gaudiot. A simple high-speed multiplier design. *IEEE Transactions on Computers*, 2006, 55(10): 1253 – 1258.

[33] D. Crookes, M. Jiang. Using signed digit arithmetic for low-power multiplication. *Electronics Letters*, 2007, 43(11): 613 – 614.

[34] P. M. Seidel, L. D. McFearin, D. W. Matula. Secondary radix recodings for higher radix multipliers. *IEEE Transactions on Computers*, 2005, 54(2): 111 – 123.

[35] R. C. North, W. Ku. β-bit serial/parallel multipliers. *Journal of VLSI Signal Processing*, 1991, 2(4): 219 – 233.

[36] D. A. Henlin, M. T. Fertsch, M. Mazin, et al. A 16 bit × 16 bit pipelined multiplier macrocell. *IEEE Journal of Solid-State Circuits*, 1985, 20(2): 542 – 547.

[37] J. S. Kelly, V. S. Rao, H. J. Pottinger, et al. Design and implementation of digital controllers for smart structures using field programmable gate arrays. *Smart Material Structure Journal*, 1997, 6(5): 559 – 572.

[38] L. Shang, A. S. Kaviani, K. Bathala. Dynamic power consumption in Virtex-II FPGA family. *Proceedings of the ACM/SIGDA 10th International Symposium on Field-programmable Gate Arrays*. Monterey: ACM, 2002: 157 – 164.

[39] Xilinx Inc. *Virtex6 FPGA: Satisfying the Insatiable Demand for Higher Bandwidth*, 2009. http://www.xilinx.com/publications/prod_mktg/Virtex6_Product_Brief.pdf.

[40] M. Gevers, G. Li, *Parametrizations in Control, Estimation and Filtering Probems*. Berlin: Springer-Verlag, 1993.

[41] T. Hilaire, P. Chevrel. Sensitivity-based pole and input-output errors of linear filters as indicators of the implementation deterioration in fixed-point context. *EURASIP Journal on Advances in Signal Processing*. Special Issue on Quantization of VLSI Digital Signal Processing Systems, 2011: DOI 10.1155/2011/893760.

[42] B. Lopez, T. Hilaire, L. S. Didier. Sum-of-products evaluation schemes with fixed-point arithmetic, and their application to IIR filter implementation. *Proceedings of the International Conference on Design and Architecture for Signal and Image Processing* (*DASIP*). New York: IEEE, 2012: 1 – 8.

[43] M. Petko, G. Karpiel. Semi-automatic implementation of control algorithms in ASIC/FPGA. *Proceedings of Emerging Technologies and Factory Automation Conference* (*ETFA '03*). New York: IEEE, 2003: 427 – 433.

[44] S. K. Rao, T. Kailath. Regular iterative algorithms and their implementation on processor arrays. *Proceeding of the IEEE*, 1988, 76(3): 259 – 269.

[45] G. Hoover, F. Brewer, T. Sherwood, et al. Towards understanding architectural tradeoffs in MEMS closed-loop feedback control. *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems* (*CASES'07*). Salzburg: ACM, 2007: 95 – 102.

[46] R. Casanova et al. Integartion of the control electronics for a mm^3-sized autonomous microrobot into a single chip. *Proceedings of the IEEE International Conference on Robotics and Automation* (*ICRA*). Kobe: IEEE, 2009: 3007 – 3012.

Appendix

1) Incremental form.

The standard version of PID controller id described in a differential equation as

$$u(t) = K_p(e(t) + \frac{1}{T_i} \int_0^t e(\tau)d\tau + T_d \frac{de(t)}{dt}),$$

where e is the system error ($e(t) = u_c(t) - y(t)$), u_c is the command signal (setpoint), y is the process variable (measured variable). K_p is the proportional gain, T_i the integration time constant, and T_d the derivative time constant of the controller. Using Laplace transform, $u(t)$ is expressed in s-domain by

$$U(s) = K_p(E(s) + \frac{E(s)}{s \cdot T_i} + s \cdot T_d \cdot E(s)).$$

For a small sample interval T_s, the continuous time variable $u(t)$ can be discretized using the following approximations:

$$\int_0^{k \cdot T_s} e(t)dt \approx \sum_{j=0}^k e(j) \cdot T_s; \quad \frac{de(t)}{dt} \approx \frac{e(k) - e(k-1)}{T_s}.$$

k denotes the kth sampling instant ($k.T_s$). Thus, $u(t)$ can be rewritten as

$$u(k) = K_p(e(k) + \frac{1}{T_i} \sum_{j=0}^k e(j) \cdot T_s + T_d \cdot \frac{e(k) - e(k-1)}{T_s})$$

with

$$e(k) = u_c(k) - y(k),$$

$$u(k-1) = K_p(e(k-1) + \frac{1}{T_i} \sum_{j=0}^{k-1} e(j) \cdot T_s + T_d \frac{e(k-1) - e(k-2)}{T_s}).$$

We calculate the difference: $u(k) - u(k-1) = K_p(e(k) - e(k-1)) + \frac{K_p}{T_i}(\sum_{j=0}^k e(j) \cdot T_s - \sum_{j=0}^{k-1} e(j) \cdot T_s) + K_p \cdot T_d \cdot (\frac{e(k) - e(k-1)}{T_s} - \frac{e(k-1) - e(k-2)}{T_s})$.

Developing separately each term of $u(k) - u(k-1)$, we obtain

$$K_p(e(k) - e(k-1)) = K_p \cdot e(k) - K_p \cdot e(k-1),$$

$$\frac{K_p}{T_i}(\sum_{j=0}^k e(j) \cdot T_s - \sum_{j=0}^{k-1} e(j) \cdot T_s) = K_p \cdot \frac{T_s}{T_i} \cdot e(k),$$

$$K_p \cdot T_d \cdot (\frac{e(k) - e(k-1)}{T_s} - \frac{e(k-1) - e(k-2)}{T_s})$$
$$= K_p \cdot \frac{T_d}{T_s} \cdot e(k) - K_p \cdot \frac{2 \cdot T_d}{T_s} \cdot e(k-1) + K_p \cdot \frac{T_d}{T_s} \cdot e(k-2).$$

After simplifications, we get the following recurrent equation:

$$u(k) = u(k-1) + K_p \cdot (1 + \frac{T_s}{T_i} + \frac{T_d}{T_s}) \cdot e(k)$$
$$- K_p \cdot (1 + 2\frac{T_d}{T_s}) \cdot e(k-1) + K_p \cdot \frac{T_d}{T_s} \cdot e(k-2)$$
$$= u(k-1) + A \cdot e(k) + B \cdot e(k-1) + C \cdot e(k-2).$$

This latter equation is called the incremental form of the controller. A drawback with the incremental algorithm is that it cannot be used for P or PD controllers.

2) Commercial form.

For better performances of PID, two corrections are performed: limitation of the derivative gain and setpoint weighting. A pure derivative action will induce a very large amplification of measurement noise. The gain of the derivative must thus be limited. This can be done by approximating the transfer function $s \cdot T_d$ as follows: $s \cdot T_d \approx \frac{s \cdot T_d}{1 + s \cdot T_d / N}$, where N is typically in the range of 3 to 20. In addition, to avoid sudden overshoots due to high variations of the setpoint, only a fraction b of u_c acts on the proportional part ($b.u_c - y$). Hence, the

improved PID algorithm becomes

$$U(s) = K_{\mathrm{p}} \cdot ((b \cdot U(s) - Y(s)) + \frac{1}{s \cdot T_{\mathrm{i}}} \cdot (U_{\mathrm{c}}(s) - Y(s))$$
$$- \frac{s \cdot T_{\mathrm{d}}}{1 + s \cdot T_{\mathrm{d}}/N} \cdot Y(s)).$$

$U(s)$ expression is discretized such that the proportional, integral and derivative terms are separately obtained, as follows: $u(k) = P(k) + I(k) + D(k)$, where $P(k) = K_{\mathrm{p}} \cdot b \cdot U_{\mathrm{c}}(k) - K_{\mathrm{p}} \cdot Y(k)$ and $I(k) = I(k-1) + K_{\mathrm{p}} \cdot \dfrac{T_{\mathrm{s}}}{T_{\mathrm{i}}} \cdot (U_{\mathrm{c}}(k-1) - Y(k-1))$.

To determine the derivative term $D(k)$, we use the differential equation representing the transfer function of $G_{\mathrm{d}}(s)$: $G_{\mathrm{d}}(s) = \dfrac{U_{\mathrm{d}}(s)}{Y(s)} = -K_{\mathrm{P}} \dfrac{s \cdot T_{\mathrm{d}}}{1 + s \cdot T_{\mathrm{d}}/N}$. By performing cross products, we obtain

$$U_{\mathrm{d}}(s) \cdot (1 + \frac{s \cdot T_{\mathrm{d}}}{N}) = -K_{\mathrm{p}} \cdot Y(s) \cdot s \cdot T_{\mathrm{d}}.$$

Applying the inverse Laplace Transform to this latter equation, we obtain $u_{\mathrm{d}}(t) = -\dfrac{T_{\mathrm{d}}}{N} \cdot \dfrac{\mathrm{d}u_{\mathrm{d}}(t)}{\mathrm{d}t} - K_{\mathrm{p}} \cdot T_{\mathrm{d}} \cdot \dfrac{\mathrm{d}y_{\mathrm{d}}(t)}{\mathrm{d}t}$.

Consequently, the discretized form of $u_{\mathrm{d}}(t)$ is

$$D(k) = -\frac{T_{\mathrm{d}}}{N} \cdot \frac{D(k) - D(k-1)}{T_{\mathrm{s}}} - K_{\mathrm{p}}T_{\mathrm{d}} \frac{Y(k) - Y(k-1)}{T_{\mathrm{s}}}.$$

After simplification, we obtain

$$D(k) = \frac{T_{\mathrm{d}}}{T_{\mathrm{d}} + N \cdot T_{\mathrm{s}}} D(k-1) - \frac{K \cdot N \cdot T_{\mathrm{d}}}{T_{\mathrm{d}} + N \cdot T_{\mathrm{s}}} (Y(k) - Y(k-1)).$$

Finally, we can write

$$u(k) = P(k) + I(k) + D(k)$$

with

$$P(k) = A \cdot u_{\mathrm{c}}(k) + B \cdot y(k), \quad I(k) = I(k-1) + C \cdot e(k-1),$$
$$D(k) = H \cdot D(k-1) + L \cdot f(k), \quad A = K_{\mathrm{p}} \cdot b, \quad B = -K_{\mathrm{p}},$$
$$C = -K_{P} \cdot \frac{T_{\mathrm{s}}}{T_{\mathrm{i}}}, \quad H = \frac{T_{\mathrm{d}}}{T_{\mathrm{d}} + N \cdot T_{\mathrm{s}}}, \quad L = -\frac{K_{\mathrm{p}} \cdot N \cdot T_{\mathrm{d}}}{T_{\mathrm{d}} + N \cdot T_{\mathrm{s}}}.$$

Abdelkrim Kamel OUDJIDA received his B.S. degree in Computer Engineering from the Institut National d-Informatique, INI, Algiers, Algeria, in 1989. He joined the Centre de Developpement des Technologie Avancées, CDTA, Algiers in 1990. Since then, he has been working on the design of high-speed and low-power digital circuits. He is team leader specialized in digital IP design and verification. He is currently preparing a Ph.D. at Franche-Comté University, UFC, Besançon, France, under the supervision of Prof. Nicolas Chaillet. His current work focuses on optimization of arithmetic algorithms for the development of high-speed and low-power finite-word-length controllers, dedicated to micromanipulation applications. E-mail: a_oudjida@cdta.dz.

Nicolas CHAILLET received his B.S. degree in Electrical Engineering from the Ecole Nationale Supérieure de Physique, Strasbourg, France, in 1990, and the Ph.D. degree in Robotics and Automation from the University Louis Pasteur, Strasbourg, France, in 1993. In 1995 he became associate professor at the University of Franche-Comté in Besançon, France. Since 2001, he is a professor at the University of Franche-Comté and is since 2012 the Director of the FEMTO-ST Institute. His research interests are in microrobotics and more generally in micromechatronics fields, especially in micromanipulation, microgrippers, smart materials, microactuators, smart microstructures and the design and implementation of their control. E-mail: nicolas.chaillet@femto-st.fr.

Ahmed LIACHA received his B.S. degree in Electronics from the Université des Sciences et de la Technologie, Houari Boumediène, Algiers, Algeria, in 2002. In 2004 he joined the department of microelectronics at the Centre for Development of Advanced Technologies, CDTA, where he is currently a digital IP designer for industrial control applications. E-mail: liacha@cdta.dz.

Mohamed Lamine BERRANDJIA received his B.S. degree in Computer Science Engineering from the University of Mantouri, Constantine, Algeria, in 2003. Since 2005, he is research assistant at Centre de Développement des Technologies avancées, CDTA, Algiers, Algeria, where he specialized in digital IP design and embedded systems for FGPA. E-mail: mberrandjia@cdta.dz.

Mustapha HAMERLAIN received the Doctorate degree in 1993 from National Institute of Applied Sciences of Toulouse (France). He is currently the head of the robotics department of the Advanced Technologies and Development Center (CDTA) of Algiers, and professor at the school (EMP). From 1988 to 1993 he was involved in a research project in the field of robotics PAM (Pneumatic Artificial Muscles) as a researcher in the National Institute of Applied Sciences (INSA) of Toulouse. His current research interests include robust control of nonlinear systems, robot motion control, visual control, robots manipulators and pneumatic artificial muscles actuators. E-mail: mhamerlain@cdta.dz.