High-Speed and Low-Power PID Structures for Embedded Applications.

Abdelkrim K. Oudjida, Nicolas Chaillet, Ahmed Liacha, Mustapha Hamerlain, Mohamed L. Berrandjia

High-Speed and Low-Power PID Structures for Embedded Applications

Abdelkrim K. Oudjida[1], Nicolas Chaillet[2], Ahmed Liacha[1], Mustapha Hamerlain[1], and Mohamed L. Berrandjia[1]

[1] Microelectronics and Nanotechnology Division, Centre de Développement des Technologies Avancées (CDTA), Baba-Hassen, BP. 17, 16303 Algiers, Algeria
{a_oudjida, liacha, mhamerlain, mberrandjia}@cdta.dz
[2] AS2M Department, FEMTO-ST Institute, Besançon, France
chaillet@ens2m.fr

Abstract. In embedded control applications, control-rate and energy-consumption are two critical design issues. This paper presents a series of high-speed and low-power finite-word-length PID controllers based on a new recursive multiplication algorithm. Compared to published results into the same conditions, savings of 431% and 20% are respectively obtained in terms of control-rate and dynamic power consumption. In addition, the new multiplication algorithm generates scalable PID structures that can be tailored to the desired performance and power budget. All PIDs are implemented at RTL level as technology-independent reusable IP-cores. They are reconfigurable according to two compile-time constants: set-point word-length and latency.

Keywords: Design-Reuse, Embedded Finite-Word-Length (FWL) Controllers, Intellectual Property (IP) , Linear Time Invariant (LTI) Systems, Low-Power and Speed Optimization, Proportional-Integral-Derivative (PID)

1 Background and Motivation

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 LTI dynamic systems. However, despite the large popularity of PID controller, little attention has been paid to its optimization, either for ASIC or for FPGA integration. In [2] 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 [3] proposes serial, parallel, and mixed PID architectures incorporating different number (1-3) of multiplication cores. High power

This work was supported by "Centre de Développement des Technologies Avancées" (CDTA), Algiers, Algeria, in collaboration with FEMTO-ST institute, Besançon, France.

consumption, even with the serial architecture, and complex control-part are the two major shortcomings of this proposal. Finally, in [4] 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's 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's 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 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, using a foundry standard-cell-library.

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 [5], called also the standard or ISA form, and the incremental form. These two forms went through FPGA implementations, using a new recursive multibit recoding multiplication algorithm (RMRMA). Results show clear superiority over those provided in [4]. PID control-rate and energy-consumption savings are respectively 431% and 20% . Furthermore, RMRMA algorithm generates scalable PID structures which can be customized to fit the desired speed and power budget. Its interesting feature as a low-power multiplication algorithm makes it useful for a wide range of numeric applications.

The paper is organized as follows. In this section we outlined the main requirement specifications for embedded PID controller. Section 2 presents the two most-used discrete versions of PID algorithm. Section 3 introduces the new RMRMA algorithm and its implementation. A discussion around the obtained results is given in section 4. . And finally some concluding remarks.

2 The two most-used discrete versions of PID

In digital control, commercial and incremental forms are the two most-used discrete PID versions [1][5]. They are respectively denoted by recurrent equations (1) and (2), and their corresponding coefficients are grouped in Table 1.

$u(k) = P(k) + I(k) + D(k)$ (1) ; where $P(k) = A \cdot u_c(k) + B \cdot y(k)$;

$I(k) = I(k-1) + C \cdot e(k-1)$; and $D(k) = D \cdot D(k-1) + E \cdot f(k)$.

With $e(k-1) = u_c(k-1) - y(k-1)$ and $f(k) = y(k) - y(k-1)$

And

$$u(k) = u(k-1) + A \cdot e(k) + B \cdot e(k-1) + C \cdot e(k-2) \quad (2)$$

Where $e(k) = u_c(k) - y(k)$; $e(k-1) = u_c(k-1) - y(k-1)$;

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

The translation of equation (1) and (2) into architectures is depicted by Fig. 1 and 2, respectively.

Table 1. Coefficients of discrete recurrent equations.

Coefficients	Commercial PID	Incremental PID
A	$K_p b$	$K_p\left(1+\frac{T_s}{T_i}+\frac{T_d}{T_s}\right)$
B	$-K_p$	$-K_p\left(1+2\frac{T_d}{T_s}\right)$
C	$K_p\frac{T_s}{T_i}$	$K_p\frac{T_d}{T_s}$
D	$\frac{T_d}{T_d+NT_s}$	–
E	$-\frac{K_p T_d N}{T_d+NT_s}$	–

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

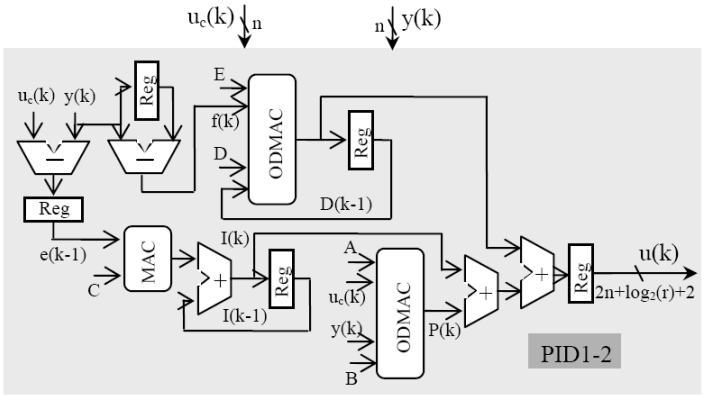

Fig. 1. Commercial PID architecture.

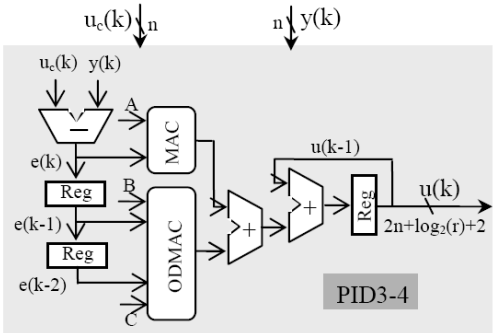

Fig. 2. Incremental PID Architecture.

To satisfy different application cases, two IP versions are developed for each equation: with constant coefficients (PID1) and with varying coefficients (PID2). This latter requires a host side interface (HIS) to handle the runtime change of the 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 HIS) for maximum energy conservation.

3 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 modified Booth algorithm accompanied with its proof [6]. Most of the proposals aimed to reduce the number of partial products either by employing digital optimization techniques [7][8][9] or by using larger slices (higher radices) [10]. However, experience showed [11] that beyond 4-bit slices (radix 8), the complexity to generate hard partial products can not be managed in a realistic way. In [11], three metrics are provided for comparing the tradoffs 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 [12] consists in applying 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).

Based on the multibit recoding algorithm presented in [6], the equation (2.1.2) of [6] is rewritten in a simpler hardware-friendly form as follows:

$$Y = \sum_{j=0}^{(n/r)-1} \left(y_{rj-1} + 2^0 y_{rj} + 2^1 . y_{rj+1} + 2^2 y_{rj+2} + \cdots + 2^{r-2} y_{rj+r-2} - 2^{r-1} y_{rj+r-1} \right) 2^{rj} = \sum_{j=0}^{(n/r)-1} Q_j 2^{rj} \quad (3)$$

Where $y_{-1} = 0$; $r \in \mathrm{N}^*$; and $Q_j \in \left\{ -2^{r-1}, \ldots, 0, \ldots, 2^{r-1} \right\}$

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, Q_j terms are split into 3-bit slices (r=2) with one overlapping bit. Thus, equation (3) takes the new simpler recursive form:

$$Y = \sum_{j=0}^{(n/r)-1} \left[\left(y_{rj-1} + y_{rj} - 2 . y_{rj+1} \right) 2^0 + \left(y_{rj+1} + y_{rj+2} - 2 . y_{rj+3} \right) 2^2 + \ldots \right.$$

$$+\left(y_{rj+r-5}+y_{rj+r-4}-2.y_{rj+r-3}\right)2^{2(\frac{r}{2}-2)}+\left(y_{rj+r-3}+y_{rj+r-2}-2.y_{rj+r-1}\right)2^{2(\frac{r}{2}-1)}\Bigg]\ 2^{rj} \quad (4)$$

$$Y=\sum_{j=0}^{(n/r)-1}\left[\sum_{i=0}^{(r/2)-1}\left(y_{rj-1+2i}+y_{rj+2i}-2.y_{rj+1+2i}\right)2^{2i}\right]\ 2^{rj} \quad (5)$$

$$Y=\sum_{j=0}^{(n/r)-1}\left[\sum_{i=0}^{(r/2)-1}Q_{ji}\,2^{2i}\right]2^{rj} \quad (6)$$

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

There is no need to prove equation (4) since it is a combination of equations (3) and modified Booth algorithm (MBA) which were both already proven in [6] and [13], respectively.

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:

$$X.Y+T.Z=\sum_{j=0}^{(n/r)-1}\left[\sum_{i=0}^{(r/2)-1}\left(Q_{ji}.X+P_{ji}.T\right)2^{2i}\right]2^{rj} \quad (7)$$

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 (6) 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 [14].

Obviously, equation (6) does not reduce the number of partial products, but allows a modulable space-time partitioning of the multibit recoding algorithm (equation 3), 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 (6) allows an important reduction of the critical path (using a carry-save adder CSA), it requires too much power. 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 MBA, is slightly increased $D+d.\log_2(r/2)$, where d is a unit delay of 1-bit adder. 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:

$$Y=\sum_{j=0}^{(n/4)-1}\left(y_{4j-1}+y_{4j}+2\,y_{4j+1}+2^2\,y_{4j+2}-2^3\,y_{4j+3}\right)2^{4j} \quad (8)$$

$$Y=\sum_{j=0}^{(n/4)-1}\left[\sum_{i=0}^{1}\left(y_{4j-1+2i}+y_{4j+2i}-2.y_{4j+1+2i}\right)2^{2i}\right]\ 2^{4j} \quad (9)$$

$$Y=\sum_{j=0}^{(n/4)-1}\left[Q_{j0}+Q_{j1}\,2^2\right]2^{4j} \quad (10)$$

$$X.Y+T.Z=\sum_{j=0}^{(n/4)-1}\left[\left(Q_{j0}X+P_{j0}T\right)+\left(Q_{j1}X+P_{j1}T\right)2^2\right]2^{4j} \quad (11)$$

The mapping of equation (11) into a serial architecture is shown by Fig. 3. Such a case (r=4) would have required the computation of hard partial products (7X, 6X, 5X, 3X) if the simple form of equation (8) was used. Notice that MBA is a special case of RMRMA for r=2. For r=1, equation (10) corresponds to Booth algorithm (BA).

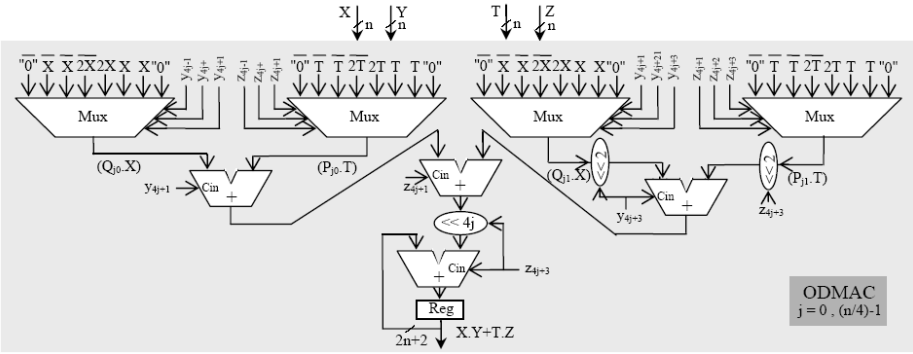

Fig. 3. Optimized double multiply-and-accumulate (ODMAC) architecture for r = 4

Table 2 comprises the implementation results of PIDs with n=16 and r=1,2,4,8,16. 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 [4] since the coefficients are stored into LUTs.

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.

Table 2. Implementation result comparison of RMRMA-based PID.

PID Core	Total Gate Count	Power* (mW)	Max. Clock Freq. (MHz)	Latency
PID [4]	16728	223	47	17
PID1_1	9286 (+44%)	167 (+25%)	62 (+32%)	17 (+00%)
PID1_2	10642 (+36%)	171 (+23%)	62 (+32%)	9 (+47%)
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_1	10661 (36%)	176 (+21%)	61 (+30%)	17 (+00%)
PID2_2	11923 (29%)	179 (+19%)	61 (+30%)	9 (+47%)
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 23MHz; PIDY_X: X = r
(+XY%): saving; (-XY%): overhead

4 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's necessary to investigate how speed, power and hardware resources scales versus r factor for a fixed word length n. Referring to equation (7) and aided by Fig. 3, 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 Table 3 and 4 corresponding to PID1 and PID2, respectively. The sole exception to this general rule is PIDX_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 3. Maximum power-consumption and control-loop-cycle of PID1

PID Core	Power* (mW)	Max. Clock Freq. (MHz)	Latency	Max. Control Loop Cycle (MHz)
PID [4]	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
Maximum control loop cycle = Maximum clock frequency / Latency

Table 4. Maximum power-consumption and control-loop-cycle of PID2

PID Core	Power* (mW)	Max. Clock Freq. (MHz)	Latency	Max. Control Loop Cycle (MHz)
PID [4]	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
Maximum control loop cycle = Maximum 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 2. Note that each adder of each level of MAC and ODMAC as well as the two ones at the output of the PID (Fig. 1 and 2) 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, limit cycles, and instabilities in the closed loop response [15].

As for power consumption, intuitively, one would expect to see PID1_16 of Table 3 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 ($P = 0.5\, V_{dd}^2\, C_{sw}\, F_{clk}$) 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 [16] 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 13MHz and low power are required, PID1_16 (244 mW at 13MHz) stands as the best candidate compared to PID1_8 (323 mW at 13MHz). However, it's 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 *Qj* 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 4 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. But 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 [4]. 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 5.

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 (Vccint) and for output blocks (Vcco) are respectively 1.8V and 3.3V for Spartan2e, 1V and 2.5V 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 [17]. Power consumption evaluation studies [16][17] 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.

Table 5. 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)
PID2_1	231	23	122	17	07.17
PID2_8	1060	04	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
Maximum control loop cycle = Maximum clock frequency / Latency

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).

5. Conclusion

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.

References

1. Åström, K., Hägglund, T.: PID Controllers: Theory, Design, and Tuning. by the Instrument Society of America, Research Triangle Park, NC, USA, 2nd Edition, ISBN: 1-55617-516-7, Copyright (1995)
2. Zhao, W., et al: FPGA Implementation of Closed-Loop Control Systems for Small-Scale Robot. Proceedings of the IEEE 12th International , on Advanced Robotics (ICAR), pp. 70-77, (2005)
3. Samet, L., 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), vol. 1, pp. 291-296, (1998)
4. Fong, Y., Moallem, M., Wang, W.: Design and Implementation of Modular FPGA-Based PID Controllers. IEEE Trans. on Industrial Electronics, Vol. 54, N° 4, pp. 1898-1906, August (2007)
5. Wittenmark, B., Astrom, K. J., Arzenin, K.E.: Computer control: An overview. Technical Report of Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden, Apr. (2003). Available: www.control.lth.se/kursdr/ifac.pdf
6. Sam, H., Gupta, A.: A Generalized Multibit Recoding of Two's Complement Binary Numbers and its Proof with Application in Multiplier Implementation. IEEE Trans. on Computers, vol. 39, N° 8, August (1990)
7. Lamberti, F.: Reducing the Computation Time in (Short Bit-Width) Two's Complement Multiplier. IEEE Trans. on Computers, vol. 60, N° 2, pp. 148-156, February (2011).
8. Kuang, S.R., Wang, J.P., Guo, C.Y.: Modified Booth Multipliers with a Regular Partial Product Array. IEEE Trans. on Circuit and Systems II, Express Brief, vol. 56, N° 5, May (2009)
9. Kang, J.Y., Gaudiot, J.L.: A Simple High-Speed Multiplier Design," IEEE Trans. on Computers, vol. 55, N° 10, Oct. (2006)
10. Crookes, D., Jiang, M.: Using Signed Digit Arithmetic for Low-Power Multiplication. Electronics Letters, vol. 43, N° 11, may (2007)
11. Seidel, P.M., McFearin, L. D., Matula, D.W.: Secondary Radix Recodings for Higher Radix Multipliers. IEEE Trans. on Computers, vol. 54, N°2, February (2005).
12. North, R.C., Ku, W.H.: β-Bit Serial/Parallel Multipliers. Journal of VLSI Signal Processing, Kluwer Academic Publishers, Boston, vol. 2, pp. 219-233, (1991)
13. Rubinfield,L.P.: A Proof of the Modified Booth Algorithm for Multiplication. IEEE Trans. On Computers, C-24, (10), pp. 1014-1015, (1975)
14. Henlin, D.A., Fertsch, M.T., Mazin, M., Lewis, E.T.: A 16 bit x 16 bit Pipelined Multiplier Marcrocell. IEEE Journal of Solid-State Circuits, vol. SC-20, no. 2, pp. 542-547, (1985)
15. Kelly, J.S., et al: Design and Implementation of Digital Controllers for Smart Structures Using Field Programmable Gate Arrays. Smart Material Structure Journal, PII: S0964-1726 (97) 87085-1, pp. 559-572, Printed in the UK, (1997)
16. Shang, L., Kaviani, A.S., Bathala, K.: Dynamic Power Consumption in Virtex-II FPGA Family. Proceedings of FPGA Conference, pp. 157-164, Monterey, California, USA, February (2002)
17. Xilinx Inc.: Virtex6 FPGA: Satisfying the Insatiable Demand for Higher Bandwidth. PN 2403, Printed in the USA, Copyright (2009)
    Available: www.xilinx.com/publications/prod_mktg/Virtex6_Product_Brief.pdf