Proceedings of the 25th International Conference *"Mixed Design of Integrated Circuits and Systems"*
June 21-23, 2018, Gdynia, Poland

Low Power, Low Chip Area, Programmable PID Controller Realized in the CMOS Technology

Tomasz Talaśka*, Rafał Długosz*,†
* UTP University of Science and Technology
Faculty of Telecommunication, Computer Science and Electrical Engineering
ul. Kaliskiego 7, 85-796, Bydgoszcz, Poland
e-mail: talaska@utp.edu.pl, markol@utp.edu.pl, rafal.dlugosz@gmail.com
† Aptive Poland S.A., ul. Podgórki Tynieckie 2, 30-399, Kraków, Poland

***Abstract*—The subject of the proposed paper is a novel, transistor level implementation (in the CMOS technology) of a programmable PID (proportional–integral–derivative) controller. In our work we focus on a discrete-time digital approach, as it facilities realization of a programmable structure which is more flexible. The novelty of the proposed solution relies on implementing the PID controller as a parallel and asynchronous structure, controlled by a simple 2-phases clock. Each of the P, I and D parts is realized as a separate channel with an own multi-bit multiplier, a summing circuit and a delay line (in the I and the D parts). The multiplier is realized as a binary tree circuit that works fully asynchronously. The implementation in the CMOS technology allows to obtain a small structure. For the input signals, and the coefficients of the PID controller encoded on 8-bits the total number of transistors does not exceed 13000. In the CMOS 180 nm technology the chip area approximately equals 0.15 mm2. Data rate is in this case even as high as 200-330 MHz, depending on the temperature and supply voltage, at very low power dissipation not exceeding 1 mW. Such a solution is suitable for various microsystems and embedded systems (used for example in automotive applications) in which small sizes and high data rate become very important features.**

***Keywords*—PID controller, ASIC, Programmable Circuits, Transistor level design, Parallel and Asynchronous circuits**

I. Introduction

Proportional-integral-derivative (PID) controllers are known since more than hundred years and widely used in the control engineering. It can be said that it is very difficult to propose something new on the system level, as the PID controllers are well described mathematically and investigated. For this reason, most recent works concerning this type of controllers focus on their efficient hardware implementation, as there is still a room for improvement in this regard. In our work we also focus on this aspect, looking for such hardware realization in the CMOS technology that will enable very high performance at low power dissipation and low chip area. In our work we do not focus on the optimization of the values of particular coefficients of the PID controller for a specific task. Our assumption is to obtain a flexible architecture with strong programming abilities that could handle the values of particular coefficients varying in a relatively wide range.

In the literature there are not too many positions related to the implementation of PID controllers in the form of application specific integrated circuits (ASIC). One the presented solutions of this type is a fuzzy PID controller [1], that has been used to control some subsystems of the aircraft (for example, responsible for the control of the roll angle). In this work it has been demonstrated, by means of simulations, that using fuzzy PID to control such systems in the aircraft provides to better results than in case of using the conventional PID controller. Additionally it has been compared the calculation time of the controller realized in an field programmable gate array (FPGA) (the result of 29.83 MHz) and in the form of ASIC (the result of 56.05 MHz).

The [2] work presents a digital implementations of a flexible, easily controlled, PID controller realized in the CMOS 2 μm technology. This solution is not fully comparable with our solution, as the circuit has been realized in the standard-cells techniques. It is not clear from the paper whether the circuit was programmable.

The most popular realizations of the PID controller are those based on microcontrollers [3], [4], in FPGA [5], and in digital signal processors (DSP) [6]. In case of the implementation on microconrollers one can distinguish both the 8-bit and 16-bit solutions, performing classical PID control scheme, as well as modified fuzzy PID controllers.

The PID controllers realized in hardware are traditionally implemented as analog devices, especially in the situations in which they are used to control continuous-time processes (heat control). This due to the ease of obtaining such operations as integration and differentiation using active devices (operational amplifiers). In one of the state-of-the art solutions an analog controller has been converted to a digital form by the use of pulse-width-modulators (PWM). Such a solution, called All-Digital PID (ADPID) has been described in [7].

The paper is organized as follows. The next Section is devoted to details of the proposed circuit. We present a general structure as well as details of particular components of the circuit. In the following Section selected simulation results are presented. The conclusions are drawn in Section IV.

II. Proposed Implementation of the PID Controller

The architecture of the PID controller proposed in this paper has been optimized for the hardware implementation at the transistor level in the CMOS technology. The controllers as well as many different circuits realized as ASICs are usually

very flexible. The structure of such circuits can be well fitted to the performed tasks. In the proposed solution each of the P, I and D channels operates in parallel and mostly asynchronously, which allows to reach high data rates. An additional advantage is that it requires only a simple 2-phases controlling clock generator.

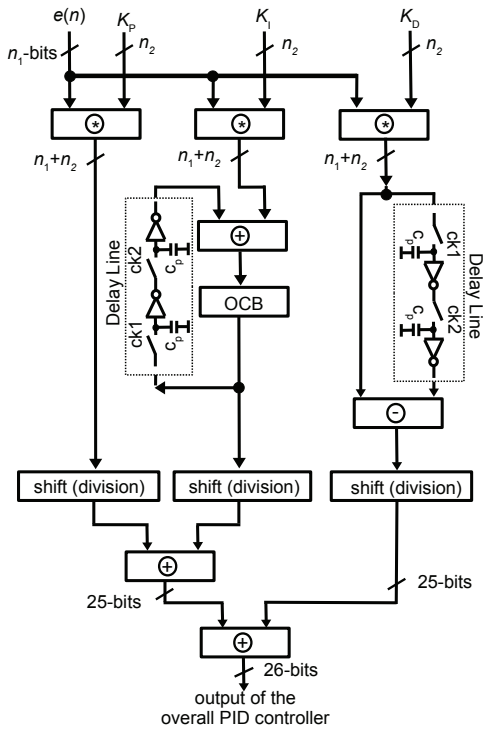

Fig. 1. General structure of the proposed PID controller.

The proposed circuit is shown in Figure 1. Each of the P, I and D channels is equipped with its own multiplier that multiplies the error signal, $e(n)$, by one of the coefficients K_{P}, K_{D} and K_{I}. The values of these coefficients are usually fractions and thus a simple multiplication is insufficient. To obtain multiplication by a fractional number we assume that the values of the K_{P}, K_{D} and K_{I} coefficients are always vulgar fractions, in which the denominator is always one of the powers of 2. This allows to obtain the division operation simply by shifting the bits to the right by a given number of positions.

The calculation scheme in the P channel relies on multiplying the error signal by the fixed point number, E_{P} in the range from 0 to 255, followed by shifting the bits to the right by 0 to 8 positions (division by a D_{P} constant). If signal is not shifted (shift by 0 positions), $D_{\mathrm{P}} = 1$ that means that there is no division. Several examples: For $K_{\mathrm{P}} = 1/2$ we multiply the $e(n)$ signal by 1 and then we shift the resultant number by one position to the right. For $K_{\mathrm{P}} = 17/32$ we multiply the $e(n)$ signal by 17 and shift the resultant signal by five positions to the right. For $K_{\mathrm{P}} = 7/256$ we multiply the $e(n)$ signal by 7 and shift the result by eight positions to the right. The signal at the output of the P channel is equal to:

$$Y_{\mathrm{P}}(n) = K_{\mathrm{P}} \cdot e(n) = e(n) \cdot E_{\mathrm{P}}/D_{\mathrm{P}} \tag{1}$$

where $K_{\mathrm{P}} = E_{\mathrm{P}}/D_{\mathrm{P}}$, D_{P} is one of the numbers of 1, 2, 4, 8, 16, 32, 64, 128, 256.

The calculation scheme in the D channel to some degree is similar to the one of the P channel. We also multiply the error signal by the fixed point number E_{D} in the range from 0 to 255. In this channel we additionally store the obtained result in the delay line, so that in the next cycle (for the next $e(n)$ sample) it is still available in the system. In practice, the D channel is composed of two sub-channels. The delayed signal $(e(n-1) \cdot E_{\mathrm{D}})$ stored in the delay line is subtracted from the new calculated signal $(e(n) \cdot E_{\mathrm{D}})$, thus realizing a differentiation. The resultant value is provided to the shifting block as in the P channel. The signal at the output of the D channel is equal to:

$$Y_{\mathrm{D}}(n) = K_{\mathrm{D}} \cdot [e(n) - e(n-1)] = [e(n) - e(n-1)] \cdot E_{\mathrm{D}}/D_{\mathrm{D}} \tag{2}$$

where $K_{\mathrm{D}} = E_{\mathrm{D}}/D_{\mathrm{D}}$, D_{D} is a similar parameter to the D_{P} one.

The calculation scheme in the I channel also starts with the multiplication of the error signal by a fixed point number E_{I} from the range in-between 0 to 255. In the next stage we add the product of this multiplication to the value stored in the accumulator, whose structure is based on a similar delay line as in the D channel. The delay line provides a positive feedback loop.

The signal from the summing circuit is provided to an additional block, whose role is to prevent the overflow. The output of the overflow control block (OCB) is then provided to the delay line in the feedback loop of the accumulator, as well as to the shifting block. As a result, the signal at the output of the I channel is given as:

$$Y_{\mathrm{I}}(n) = Y_{\mathrm{I}}(n-1) + K_{\mathrm{I}} \cdot e(n) \tag{3}$$

where $K_{\mathrm{I}} = E_{\mathrm{I}}/D_{\mathrm{I}}$, and D_{I} is similar to D_{P}.

Finally the outputs of particular channels are summed together providing the output signal:

$$Y_{\mathrm{PID}}(n) = Y_{\mathrm{P}}(n) + Y_{\mathrm{I}}(n) + Y_{\mathrm{D}}(n) \tag{4}$$

The values of the E and D parameters allow to tune the K_{P}, K_{D} and K_{I} coefficients in the range from $1/256$ to 255. An additional shift of the bits at the output of the overall PID controller allows to normalize the values of these coefficients to 1, so that their value vary in this case in the range from $1/65536$ to 1.

A. Particular components of the controller

To obtain a high data rate of the controller we perform as many operations in a parallel and asynchronous fashion as possible. All components have been optimized for the operation in this way.

1) The multiplier: One of the main blocks is the multiplier. We realized the multiplier as an asynchronous binary tree (BT) which does not require the clock generator. Such solutions are less popular than typical shift-and-add circuts, due to larger number of transistors, however they are substantially faster. In case of relatively small numbers of bits in the signals, the hardware complexity becomes comparable.

The used multiplier allows to multiple both positive and negative values of the input signal, which is codded in the Two's Complement (U2) code. The E parameter is always positive. To obtain a correct result the most significant bits in particular adders are set to the sign value of the error signal.

An important aspect here is the hardware complexity. The number of transistors depends on the number of 1-bit full adders (1BFA) that are used in multi-bit full adder (MBFA), which are main building bblocks of the binary tree multiplier. A single 1BFA requires 28 transistors, while the total number of transistors in the overall multiplier equals about 2300. In the overall PID controller three multipliers are used, so the total number of transistors equals 6900, in total.

2) The dividing block: The division by the D constants is performed in the shift block, which is a commutation field composed of switches. Each of such blocks has 16 inputs (i) (signals from particular channels of the controller) and 24 outputs (o). In case if there is no shift operation ($d_0 = 1$), the 16 inputs become the most significant outputs of the circuit, i.e. o23=i15, o22=i14, ..., o08=i00; For an example case of division by 256 the connection sequence is as follows: o15=i15, o14=i14, ..., o00=i00.

The switches in the circuit are realized as transmission gates, controlled by complementary signals d_k and $\overline{d_k}$.

Since the number of the outputs of the circuit is larger than the number of their inputs, some outputs can be floating. To avoid such a situation the unused outputs are connected either to V_{SS} or to V_{DD} terminals, depending on the sign of the input value. Independently on the sign, the floating outputs "below" the 16-bits input number are always connected to ground. The outputs "above" the input number (in case if there is any shift) are connected to most significant bit (MSB) that is the sign of the input number. For negative numbers all floating terminals "above" are connected to logical '1', while for the positive number they are connected to logical '0'.

The number of transistors in a single shift block does not exceed 500 ($24 \cdot 9 \cdot 2 = 432$ in the switches + 16 in NOT gates used to generate complementary $\overline{d_k}$ signals).

3) The subtracting and summing circuits: Both summing and subtracting operations are performed in the MBFAs. In the D channel of the controller the subtraction operation is required to calculate a difference between the current and the previous samples of the input error signal. The $E_{\mathrm{D}} \cdot e(n-1)$ sample, stored in the delay line, is negated and supplemented to 2 by the addition of '1'. The resultant value is added to the $E_{\mathrm{D}} \cdot e(n)$ signal in a typical MBFA. To obtain the supplementing to 2 operation, the carry-in bit in the least significant 1BFA in the adder is connected to V_{DD} supply.

Similar blocks are used in the I channel (in this case without changing the sign as in the D channel) and at the output of the overall PID controller. The total number of 1BFAs equals 16 (D channel) + 16 (I channel) + 49 (output stage of the PID) = 81. The number of transistors used in these blocks (including the negation in the D channel) does not exceed 2400.

4) The delay line: In the D and the I channels are used 16-bit delay lines, composed of chains of switches and NOT gates. Particular bits are stored on parasitic capacitances of the logic NOT gates. Each of the 16 channels in the delay line is composed of two NOT gates and two switches realized as transmission gates. As a result, a single 16-bits delay line contains 128 transistors (256 in both delay lines). The delay line is controlled by a 2-phases clock generator. The 1$^{\mathrm{st}}$ phase (ck1) starts in the moment in which a new sample of the error $e(n)$ is provided to the input of the PID controller. During this time period the controller performs all calculations that include: multiplications, summations in the P and the D channels, shifting the bits, and summation of the outcomes of particular channels of the controller. The two clock phases are not symmetrical. The clk2 phase is much shorter than the clk1 one, as this time is required only to update the information stored in the delay lines.

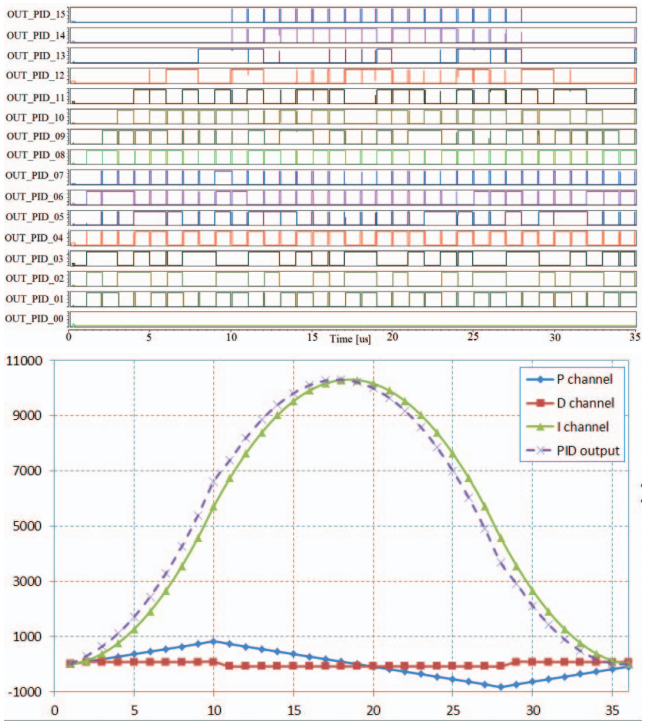

Fig. 2. Simulations results in case of the proposed implementation of the PID controller

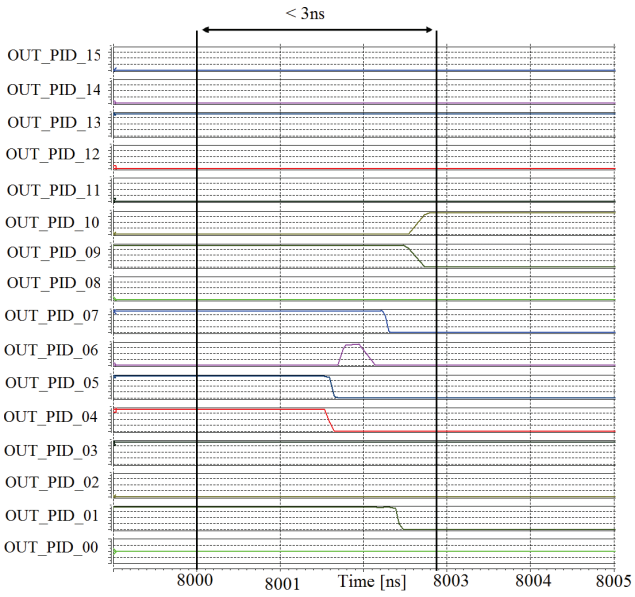

Fig. 3. Time required to perform calculations for a single input sample

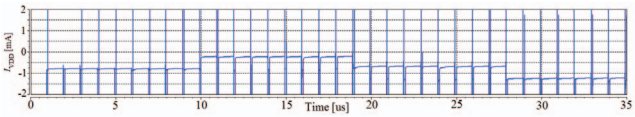

Fig. 4. Current consumption of the overall PID controller

III. Verification of the Proposed PID Controller

The circuit has been verified by means of the transistor-level simulations performed in the Hspice environment in the TSMC CMOS 180 nm technology. In this Section we present selected results to demonstrate the performance of the realized controller.

To facilitate the analysis of the obtained results, the input signal is a triangular wave. A single period of this signal is composed of 36 samples. The signal rises up from the value of 0 to 9 with a step of 1, then it drops linearly to the value of -9 with a step of -1 and finally it increases to 0 again. The input signal is coded on 8 bits, so it can theoretically vary in between -128 and 127 (Two's Complement code). These values can easily be extended in the hardware implementation through increasing lengths of particular used MBFAs.

The results are presented for particular channels separately, as well as for the overall PID controller. The values of the K_{P}, K_{D} and K_{I} have been set to $91/256$, $80/256$ and $127/256$, respectively. To make it possible, the values of particular E parameters are set to 91, 80 and 127, while all dividing blocks shift the resultant signals by 8 bits to the right.

A. Verification of the overall PID controller

The output of the overall PID controller is shown in Fig. 2. The obtained results are in the agreement with the theoretical values for the assumed parameters of the controller, and the input signal. The important aspect is the power dissipation of the circuit and attainable data rate. The results are on previous diagrams presented are for input data rate of 1 MSamples/s. Such data rate allows to distinguish steady-state values at the output of the circuit and transient states in which power dissipation is the largest. We can also observe in this way how much time the controller needs to reach the steady-state for each new input signal sample. It is shown in Fig. 3. The results depend on the value of the input signal, as well as the values of the K coefficients. They have especially an influence on the multiplication time. For the worst scenario the circuit needs 3-7 ns to calculate the output signal. During this time it dissipates an average power of 1 mW, so the energy consumed during the calculation of a single sample equals 3-7 pJ. This value is for the worst case scenario. A typical energy consumption does not exceed 5 pJ per calculation of a single sample. The supply current is shown in Fig. 4.

IV. Conclusions

In this paper we proposed a novel transistor level implementation of the PID controller in the CMOS technology. We selected the 180 nm technology due to practical reasons (price and availability). For newer CMOS processes the data rate will be higher by a factor $r = \lambda_{180}^2/\lambda_l^2$, where λ_{180} and λ_l are minimal transistor lengths in the 180 nm and the newer processes, respectively.

The circuit is programmable. It enables modification of the values of particular parameters of the controller in a relatively wide range. Due to parallel and to high degree the asynchronous operation the circuit is very fast.

References

[1] Subash John; Abdul Imran Rasheed; Viswanath K. Redd; "ASIC Implementation of Fuzzy-PID Controller for Aircraft Roll Control", *International conference on Circuits, Controls and Communications* (CCUBE), Bengaluru, 2013, pp.1-6

[2] K. Prasad, A. E. Raj, "VLSI implementation of digital compensators and predictive PID controllers", *International Conference Combined Volumes* (Electro/94), May 1994, pp.670-688

[3] Sereyvatha Sarin, Hilwadi Hindersah, Ary Setijadi Prihatmanto, "Fuzzy PID Controllers Using 8-Bit Microcontroller for U-Board Speed Control", *International Conference on System Engineering and Technology*, Bandung, Indonesia, 2012, pp.1-6

[4] Adik S. Wardoyo, Hendi S., Darwin Sebayang, Imam Hidayat and Andi Adriansyah, "An Investigation on the Application of Fuzzy and PID Algorithm in the Two Wheeled Robot with Self Balancing System Using Microcontroller", *International Conference on Control, Automation and Robotics*, Copenhagen, Denmark, June 2015, pp.64-68

[5] Yuen Fong Chan; M. Moallem; Wei Wang, "Design and Implementation of Modular FPGA-Based PID Controllers", *IEEE Transactions on Industrial Electronics*, Vol. 54, Iss. 4, 2007, pp.1898-1906

[6] Jingmeng Liu, Shangfeng Li, Xingming Wu and Baicheng Chen, "Design of self-adaptive fuzzy-pid controller based on DSP and FPGA for rapid thermal processing"; *IEEE Conference on Industrial Electronics and Applications*, Beijing, China, June 2011, pp.1649-1653

[7] Hui Hui Chin, *All Digital Design and Implementation of Proportional-Integral-Derivative (PID) Controller*; University of Kentucky Master's Theses, 2006