FPGA-implementation of PID-controller by differential evolution optimization

Research Article **Open Access**

Mika Hanhila, Timo Mantere*, and Jarmo T. Alander

FPGA–implementation of PID-controller by differential evolution optimization

https://doi.org/10.1515/eng-2018-0038
Received August 21, 2017; accepted October 4, 2018

Abstract: We will describe an FPGA implementation of PID-controller that uses differential evolution to optimize the coefficients of the PID controller, which has been implemented in VHDL. The original differential evolution algorithm was improved by ranking based mutation operation and self-adaptation of mutation and crossover parameters. Ranking-based mutation operation improves the quality of solution, convergence rate and success of optimization. Due to the self-adaptive control parameters, the user does not have to estimate the mutation and crossover rates. Optimization have been performed by calculating for each generation fitness value by means of trial parameters. The final optimal parameters are selected based on the minimum fitness.

Keywords: Adaptive PID Controllers, Differential evolution, Field Programmable Gate Array (FPGA), Optimization, VHDL, Ranking-based mutation operation, Self-adaptive control parameters

1 Introduction

Intelligent PID controllers are nowadays implemented also with various optimization methods. The PID controller has progressively evolved from original mechanical pneumatic systems to microprocessor-based systems. These days it is also become common to use digital Field Programmable Gate Array (FPGA) based control systems. The PID controller is used in many different kind of control systems in almost all fields of life and technology, from process control, product manufacturing, robotics, automotive systems, to the space systems.

Mika Hanhila: CM Tools Oy, Muottitie 1A, FI-65320, Finland, Tel: +358 50 4089 227, E-mail: hanhila.mika@gmail.com
***Corresponding Author: Timo Mantere:** University of Vaasa, P.O. Box 700, FI-65101, Finland, Tel: + 358 29 4498 298, E-mail: timo.mantere@uva.fi
Jarmo T. Alander: University of Vaasa, P.O. Box 700, FI-65101, Finland, Tel: + 358 50 5534 006, E-mail: jarmo.alander@uva.fi

Programming of FPGAs is considered challenging. This is because of several reasons: FPGAs are digital hardware; basic knowledge of digital circuits is needed to understand and use them. FPGAs are used by programming; basic knowledge of programming is needed but is not enough simply because of the parallel nature of circuits and their modelling languages. This parallelism is usually totally absent from the general purpose programming languages like Java. The efficient use of parallel processing by FPGAs presupposes understanding of pipelining, which is usually not known by programmers.

In this study we have implemented evolutionary computation based intelligent PID-controller.

Differential evolution algorithm (DE) is a relatively new optimization method, which was selected due to its simple presentation. It is ideally suited for numerical optimization for the PID controller parameters due to its simplicity and real-time nature. PID controller optimization was implemented in FPGA, for which there is only marginal experience in reported controller optimizations in literature.

DE is suitable for accurate minimization of numerical parameters. FPGA implementation increases efficiency of PID-controller, is real-time, and diversify functionality, stability and minimize power consumption.

FPGA is desirable technique to use due its massive parallelism. It is programmable hardware that enables e.g. hardware reconfiguration and massive parallel calculations and operations.

It is quite difficult and challenging implementing of Differential Evolution (DE) optimization algorithm to the FPGA. DE algorithm by nature always uses floating point coding and needs several random number generations. Therefore, FPGA implementation of DE required quite a lot of work with implementing several random number generators to the FPGA. Floating point coding was mostly handled with fixed point coding in order to avoid heavy floating-point arithmetic calculations with FPGA.

In this work the cost function of the delta value, sampling value and derivation term of PID controller was optimized with differential evolution algorithm according the fitness value with FPGA. Fitness value was calculated in

each generation from the trial parameters. According to the fitness value, best control parameters were selected for PID-controller. PID-controller was tested in the ModelSim program first and then the test results was analyzed with Matlab.

Our main hardware description language (HDL) has been Very High–Speed Integrated Circuit Hardware Description Language (VHDL) since mid-90s. The reason for selecting VHDL was its modelling power. Unfortunately, the full power of VHDL also requires quite much programming skills and work. In real-life FPGA projects, also other HDL alternatives such as Verilog need to be used.

In chapter 1 we introduce the topic, differential evolution, PID and FPGA. Chapter 2 introduces our implementation and FPGA architecture. In chapter 3 we discuss the testing of the system, and in chapter 4 there are some results. In chapter 5 there are some discussion about our results, and also what other people have found out in their related publications, and system and chapter 6 conclusions.

1.1 Related work

Optimization of differential evolution, structure and activities of PID-controllers have been studied in many publications. The most commonly used sources of Survey of differential evolution algorithm were examined; the original differential evolution is was introduced [1]. The basis of the optimization used in this work was [2] and the speed and operation and of differential evolution was improved [3]. The basis of studying the PID-control was [4], fixed-point representation have been studied [5], and PID-controller tuning based on experimental tuning [6]. Differential evolution based PID optimization is also studied in [7], they used steady-state error, peak overshoot, rise time and settling time as fitness function. They found out that DE based controller improves system responses compared to the open loop system, and also reduces the percentage overshoot to zero.

Adaptive controllers, like adaptive PID (APID) [8–11] have been under grooving interests in recent years. Usually APID utilizes e.g. fuzzy logic based compensator [8–10, 12], evolutionary computation based parameter optimization [8], or Sliding Mode Control (SMC) scheme [10, 11], or something that combines several techniques [8]. Quite often APID is realized with FPGA technique due its speed and ability to real time response.

Chan et al. [13] have investigated the module-embedded PID controller implemented with FPGA. It was applied to the temperature control system. The controller functions were divided into modules that can be modified if needed. The work has focused on ADC converter, PID controller and PWM generator module, which can be reused in future applications.

Saad et al. [4] have explored and compared the optimization and design of the PID controller with differential evolution and genetic algorithms. The PID controller was tuned both by the Ziegler-Nichols method and by the MSE and IAE method. In their study MSE and IAE methods were remarkably more efficient compared to the Ziegler-Nichols method. The optimization time difference can vary from several seconds to dozens of seconds depending on the application you want to adjust.

Lima et al. [14] have studied the operation of the PID controller when using a fixed-point presentation in an FPGA program. The performance is compared to a floating-point presentation, which compares the effect of a fixed-point presentation with different bit amounts to system stability. The fixed-point presentation has been evaluated using word-length analysis techniques. The aim of their study was to find a compromise on the right bit accuracy between the floating and the fixed-point presentations. They found that with a fixed point presentation could save significant resources in the FPGA circuit such as power losses and energy consumption and reduce planning time.

There are several other techniques that can be used for PID design and optimization, but due our history with applying evolutionary algorithms to several different types of problems, we decided to use DE method in our FPGA implemented PID experiments. There are some more related work at chapter 5, where we analyze our results and some more related work.

In practice we were more interested to implement differential evolution to the FPGA than PID. PID was chosen mainly just to have something that we can optimize with our DE. Therefore we did not consider whether of not PID controller system is best to control something, or would the other controller types better. There are a lot of consideration of how controlling should be done these days, what new issues and challenges should be considered, and review of many different type new controller technigues in [26].

1.2 Differential evolution

Original (DE/rand/bin) method includes to traditional exploration methods [1]. DE-algorithm is operative and simple population based numerical optimization method. The objective function is sampled in several randomly selected initialization points in DE optimization. Population struc-

ture is needed population size (NP), crossover rate (CR), mutation parameters (F) and number of chromosomes to match control parameters.

Population is initialized by generating randomly individuals with 1 to NP chromosomes evenly between 1 to D. Initialization values are given to individuals manually in optimization. Difference vector is randomly generated from individual selection. The weighted difference vector is formed from the difference vector multiplying by mutation parameter F. Mutation constant is affected step length of mutation, where in difference vector decreasing also step length of mutation is decreased in population dwindle. Mutation vector is born from sum of parameter of difference vector and individual parameter. Finally target vector is crossed with mutation vector. The crossing was performed between crossover parameter CR and random number to each chromosome by generation.

Hereafter, target and trial vector was selected based on the objective function value to place of target vector chromosomes by generation. Selection is done the according the objective function. From mutation vector is selected always at least one chromosome. If target vector is equal to the trial vector then trial vector is selected to continue in the genus.

Figure 1 shows the differential evolution algorithm flow. We select four parents randomly, then we ccalculate the difference vector between two of the parents, and add it to the third parent vector multiplied with the mutation coefficient weight F. Crossover between this new vector and the third parent is done with the gene selection probability CR. Then we will calculate target function values and compare the new trial with fourth parent, the more fit will reach the next generation.

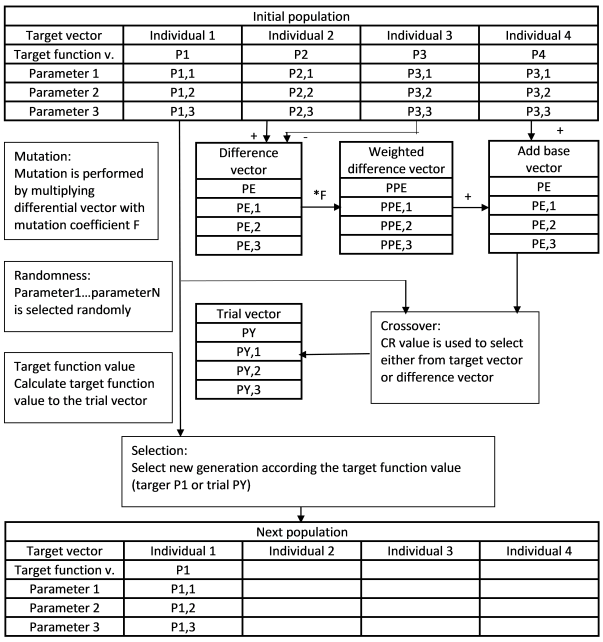

Figure 1: The differential evolution algorithm (DE/rand/1/bin)

1.3 FPGA

Digital control systems of Field Programmable Gate Arrays (FPGA) have been more popular nowadays. Fixed-point numbers can be implemented scaling the floating-point numbers to integer without fractions in FPGA. The scaling is simple and fast upon addition, reduction, multiply and divide. Multiply and division is done by bit shifting. The integer arithmetic and fixed-point arithmetic are included and calculations of decimal numbers was done by scaling. Reduction of power losses and energy consumption and saving hardware resources can be achieved by using analysis and word length optimization techniques, and fixed-point numbers for operationally large systems. Implementation of PID-controller for FPGA is improved execution of the real-time controller. Several PID-controllers could easy programming to a single FPGA-circuit when using fixed point number. Fixed-point numbers decrease also design and programming time.

Optimization is also great for minimizing the numerical parameters of the PID controller. The FPGA program is able to increase the efficiency of the PID controller, diversify functionality, enhance real-time response, stability and minimize power consumption [13]. However, the use of fixed-point numbers and floating point numbers in the FPGA makes it difficult to design, as many applications require different FPGA hardware constructions, their use require a lot of experience [15]. Fixed-point architecture is cheaper and faster than floating-point execution, but calculating integers and decimal parts requires time-consuming functions in the FPGA [14]. In addition, in FPGA, fixed-point calculation can be replaced by scaling the floating-point numbers into integers. Scaling calculation is simple and fast in addition and subtraction calculations, multiplication and division calculations when multiplication and division occurs by transferring bits as two potencies [16].

1.4 PID

Digital PID-controller structure can be implemented fixed-point number for FPGA. Structure of the PID-controller is implemented on module level. Its digital action based on the addition, reduction, multiplying and to the delay

of signal z(-1). PID-controller can be tuned various tuning method as step response and the oscillation boundary of Ziegler-Nichols and numerical (IAE, ISE) methods. Step response tuning method can be used in delay applications such as industrial processes usual are. Numerical tuning method can be used although in the real-time applications such as fuel feed and ignition control optimization.

Unsigned optimized parameters of 16 bits are imported from DE-optimization architecture to PID-control system. Unsigned values are adapted signed values for PID-controller at first, which simplify to digital optimization of the controller. Proportional, integral and derivative terms are calculated separately in different program modules. Output value of PID-controller is got by summing up all terms in the end.

Optimization of PID-controller generally based on reduction of reference and measuring value or quantity of generations. Parameters of PID-controller are selected by comparing the fitness values of different generations. The cost function is calculated of delta value, sampling value and derivation term of PID-controller. According to fitness value the best control parameters is selected for PID-controller.

PID controller tuning can use application-specific tuning methods. Generally, a discrete PID controller requires a short sampling interval, which corresponds to the operation of a continuous-time PID controller. In our case, when we want to create dynamic and real-time optimized PID controller, we need several different types of model signals that used for seeing that the PID tuning is acting correctly in different kind of control situations.

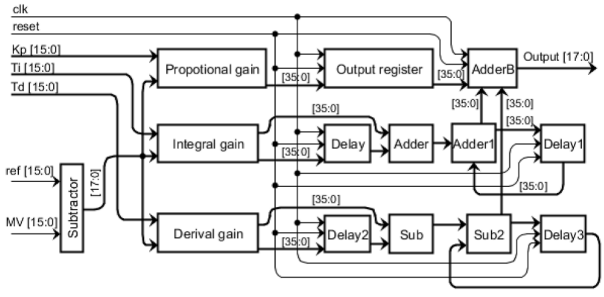

Figure 2: PID-controller in module level

Figure 2 shows module level of the PID-controller. P-, I- and D- terms are calculated in the program modules by adding, subtracting, and delaying values within clock cycles. The final output value of the controller comes out from output [17:0]. Bit accuracy of our PID-controller was 36 bits.

2 Implementation

The system was designed with the Development and Education board DE2 of Altera Corporation. The system consisted of Nios II processor, audio codec, SDRAM memory, switches and buttons. Nios II is a soft processor that can be instantiated on an Altera FPGA board. Nios processor was programmed using SOPC Builder of the Quartus II 10.1sp1 software and Nios Software Build Tools (SBT) 10.1sp1, which is an Eclipse-based integrated development environment (IDE) for Nios-specific C/C++ code. The latter is an Eclipse-based integrated development environment (IDE) for Nios-specific C/C++ code. The former is the built-in SOPC tool of Quartus, but is nowadays considered a legacy tool because in newer Quartus versions it is replaced with more high-performance tool, Qsys.

2.1 The system

The FPGA system was built around Altera's Nios II processor. The Nios II system was connected to one user peripheral - the audio codec Wolfson WM8731 controller - and to some pre-designed peripherals: parallel I/O (PIO) modules for the buttons, switches and light-emitting diodes (LEDs), Liquid Crystal Display (LCD) controller, on-chip memory controller and an SDRAM controller. All these modules used the Avalon Memory-Mapped (MM) interface to communicate with the processor. PIO modules had also the Interrupt Sender interface and signals from the audio codec were exported to the top level using Avalon conduit interface.

2.2 Memories

The application used two memories: the on-chip memory of the FGPA device and an external SDRAM memory module. The on-chip memory of the EP2C35 FPGA is organized in 105 M4K blocks. Each block consists of 4608 bits: 4096 data bits (4Kbits) and 512 parity bits. However, in this work, we needed only a small on-chip memory, and thus the size of the on-chip memory was set to be 16 Kbytes. Communication was performed using 32-bit words. It is recommended to use this word length when connecting the on-chip memory to the data master port of the processor (Altera 2010).

The SDRAM memory module located on the DE2 board can store 8 MiB of data. It is meant for storing large blocks of data, for example audio data (Altera 2010). The communication between the system and memory module was

performed using the pre-designed SDRAM controller provided by the SOPC Builder. In addition to this, a phase-locked loop (PLL) was needed to clock the SDRAM module. The purpose of the PLL circuit is to generate a clock signal, which is suitable for the SDRAM. That clock must be phase-shifted by -3ns from the original clock source, as described in Altera's (2008b) tutorial. The ALTPLL module of the SOPC Builder software did not work, but the problem was solved by using an ALTPLL megafunction instead of ALTPLL Nios module. The megafunction was inserted to the top-level schematic diagram and the inputs and outputs were connected to the appropriate pins.

2.3 LCD display

The LCD display was configured to use two lines and 5x7 pixels sized characters. Placing characters to the display was performed by writing the coordinates of the character to the command register and writing ASCII code of the character to the data register of the LCD device.

2.4 Architecture

The implementation of our PID controller program was created with the structure model of FPGA, so that the DE algorithm architecture was created, and PID was placed to its substructure. The system resources consumed a total of 11821 logic cells. The resources of the FPGA program is calculated according to the number of logic elements and LUT-input use, registers, I/O inputs and outputs and the multipliers.

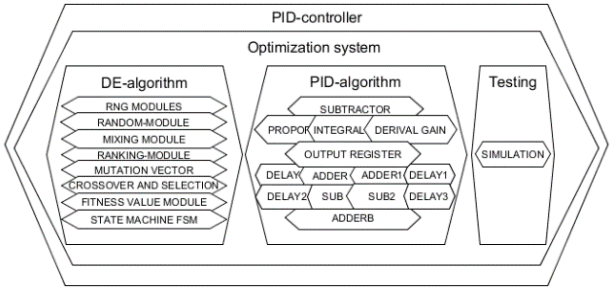

Figure 3: The Architecture of the PID-controller

The number of logic elements needed was 5383, needed LUTs 4713, 1622 registers, 79 I/O's and total of 20049 internal connections. Total power consumption of the used resources Altera's PowerPlay Power Analyzer gave 119 mW, which is relatively low. Figure 4 shows what modules there are inside of FPGA in this implementation. The system for which controlling the FPGA based PID controller is used is outside of the FPGA.

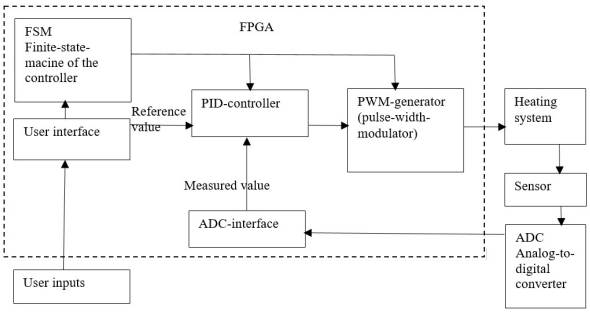

Figure 4: PID-controller in closed loop and what are inside FPGA

In figure 4 FSM module responses to the user commands like controllers stand-by mode, start, shutdown, optimization etc. User interface module handles the commands user gives, like temperature setup values, system starting and shotdown. User interface controls FSM state-machine. ADC-interface sends the measured signals to PID calculation. PID-module calculates from the measurement values the P, I, and D terms. Stability of PID controller is controlled with the coefficients of P, I, and D terms. Controller output is connected to input for PWM module. PWM generator module uses comparator and counter in order to Convert input signal to regular square wave. Counter value increases in every clock cycle and it is compared to the value that comes from PID controller. When value is greater than counter PWM outputs are on, otherwise closed. Counter is zeroed according the setup when there is enough clock cycles. The system is used to control heating.

3 Testing

Testing is an important part of the implementation of the program. It is also important to divide the program in suitably sized modules, so that the testing can be successful as easily as possible with the testing program. Finally, when the modules has been tested, they are connected to each other and the overall system is tested as a whole. The following results has been obtained by collecting snapshots of the most optimal generations from ModelSim.

The test program created four random individuals, for which each was given three parameters. The number of test runs was 24, so in each test run there were twelve ran-

dom parameters. Some of the random parameters were selected based on the mutation and the fitness of preliminary test runs in order to match the random values ??and the best fitness values with each other in the testing process. The rest of the parameters are taken from the simulation table in corresponding order. The program tried to variate the individuals' random parameters, in which it succeeded quite well.

In the start-up situation of the optimization, before random mixer module starts, the random variation in values ??was obtained by starting the random number generators different time by the help of counters. The actual random values ??mixer began to function when optimization program has started. The early test runs have some problems, but eventually the random number mixing started to work when clock pulse timings where set correctly.

We conclude that the random number mixer affect the frequency of random numbers to reoccur, and different timing setting increase or reduce the variation of random number values in the different test runs.

Randomly created individuals were fed to the ranking module, in which the first calculated fitness value of individuals by summing the individual's parameters individually. After this comparison is performed on individuals, selecting the largest and second largest specimen per test run. The ordering of the population by fitness value ads randomness and identical random values did not occur among the parameters.

4 Results

The execution time of the PID program was tested by measuring with the ModelSim. When operated at 500 Hz sampling rate optimizing 50 generations it took 6.562 s. When testing by using 10 GHz sampling rate (maximum speed) it took only 0.3281 microsecond. Based on testing the speed was affected by to the sampling rate and the number of generations. A test optimization took 3821 clock cycles during the fifty-generation, which remained constant in different test runs if the generations remains the same.

We also tested 25, 50, 100 generations, the number of clock cycles needed with different optimization run lengths were respectively 1398, 3281 and 6010. The execution time varied linearly between [0.14; 0.60] µs, with 0.0001 µs sampling time, and with 2 s sampling time [2.80; 12.02] s.

Altera's Cyclone II PowerPlay Power Analyser EP2C35 gave our optimizing PID controller the key permanent power consumption 79.95 mW, inputs and outputs' power consumption was 39.18 mW and the total power consumption was 119.13 mW at 10 GHz clock speed. The power consumption slightly exceeds the information promised by the Actel for Cyclone III family EP3C16 power consumption, where central ten thousand logic cells permanent power consumption were claimed to be 42 mW and total power consumption of 161 mW with 100 MHz clock frequency [17].

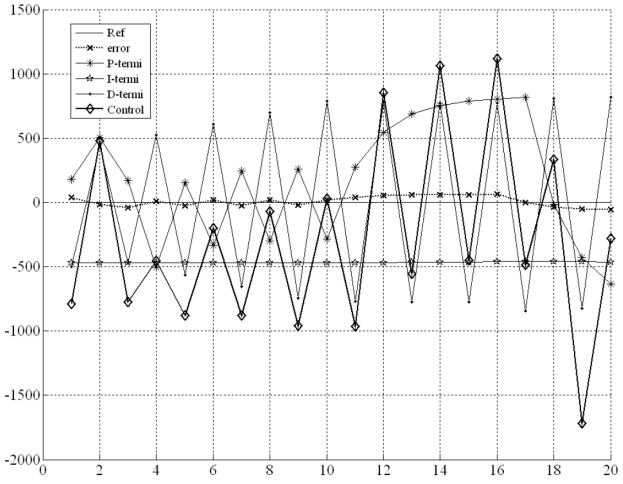

Figure 5: The testing of PID controller. Y-axle shows optimized PID-values in 20 different optimization runs (X-axle)

Figure 5 shows that the best-found PID values vary widely between different test runs. The system is quite nondeterministic and it is possible that very different PID values will work well depending on the situation.

5 Discussion

The variation of the self-adaptive parameters needed for mutation and a crossover module was relatively small in the test runs. This was due to clock rate of pseudo random generator, which led the random values ??being relatively similar. This caused mutation coefficient to get low values ??and crossover coefficient to get variation of low and high values. This problem can be corrected with better random number mixer and by expanding the random generator value range. Despite the aforementioned problem, mutation and crossover succeeded as expected.

The results are difficult to compare with any of the other FPGA based PID controllers, since the application people use these are usually unique. There are papers by Chang and Juang who have created Fuzzy-PID controller with FPGA and optimized parameters by real-valued ge-

Table 1: FPGA DE/PID execution performance numbers

Number of generations	Clock cycles	Sample time	Execution time
25	1398	0.0001 µs	0.1398 µs
50	3281		0.3281 µs
100	6010		0.6010 µs
25	1398	2 ms	2.796 s
50	3281		6.562 s
100	6010		12.020 s

netic algorithms [18]. In addition, [19–24] have implemented PID controller and genetic algorithm (GA) to optimize its parameters with FPGA. Also there are several other PID FPGA Implementations [22–24]. All these implementations are interesting but we cannot really compare our solution with them.

Aforementioned papers usually state that there are relatively few FPGA based PID controller research papers. They also state that the benefits of using FPGA are reliability, flexibility, re-programmability, high speed due massive parallelism, and low power comsumption [8–11, 23**?**]. Also fast time-to-market, and shorter design cycle [9] are mentioned. Papers [19–22] stated that real-valued GA consumes many FPGA resources. The advantage of FPGA implementation are obviously the speed, design flexibility, high reliability and short cycle of technical development.

We did not use any other system to run similar tests as we did with FPGA implementation, therefore we do not have results of how much faster FPGA based intelligent and optimizing PID controller was compared to e.g. PC implementation. In paper [23] there is a comparison between PC and FPGA implementation, but basically they state that with FPGA it was possible to achieve much higher sampling rate. With PC they used 10ms sampling time, while with FPGA implementation it could be shorten to 100 µs. The error rate was reduced to half due faster sampling rate. Therefore, FPGA was only real-time platform in that test. In paper [12] they stated that in their FPGA PID implementation control process needed just 19 clock cycles, with 57.4 MHz system frequency that corresponds process speed of 0.33 µs.

In paper [8] there was a comparison between GA optimization and Ziegler-Nichols tuning of PIDs. They also compared GA optimized PID and fuzzy logic controller. They find out that it is difficult to determine which controller was best, since it depends on uncertainty (noise). With student test they come conclusion that GA optimized FLC has the lower error rate.

6 Conclusions

The most valuable contribution of this work was to implement both differential evolution and PID controller to FPGA. There were not so many DE FPGA implementations in the literature, so it was a challenge. For doing so, we needed solve several problems concerning the random number generators, as well as crossover and mutation operators. Many experimental work and test runs were done in order to improve model. The eventual implementation worked relatively well, and we were able to obtain some results of the how much FPGA resources and power consumption our model and test runs needed. However, there is not a direct other implementation to compare our results with. Now that we have DE FPGA implementation we can also use it for other optimization purposes. In this work PID controller was used as an optimization subject, but it can be relatively easily changed since it was realized as substructure of the DE program.

By testing the PID controller, we obtain graph from where we can be compare the different aspects of the controller. The difference value was generated randomly by imitating the real control situation of where max error was set to 5% from the reference value, which was set at 2000. In the graph, the controller terms can be compared with the magnitude of the difference value. In the test runs the difference between the values were increased towards the end of the optimization. Ratio (P-term) strongly influencing to the rapid reaction of the controller (control). The integration part (I-term) remained constant, and the derivative (D-term) increased with the increasing difference value. There was a programming error that the controller derivative lacked the factor of two. Increasing in the factor coefficient would increase the magnitude of derivative component of the controller and the control, but at the same time, the control value would become more stable.

Based on the results of different aspects of the program can be developed in the future as well as expanding

and complicated by randomness that increasing the number of individuals and generations.

This was the first attempt to implementing DE-optimized PID to the FPGA. There were still many problems and limitations in the system. The new version is planned to be implemented with SOC-FPGA, where parts of the system can be run on a processor and FPGA is used to only those operations that have highest speed demand.

Acknowledgement: This paper is a short English summary of Mika Hanhila's Master's thesis [25]. Those who are interested of the topic are advised to read original work (in Finnish) in order to get more detailed information of the work done and results.

References

[1] Price, Kenneth V, Rainer M. Storn (1997). Differential Evolution - A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* **4**, 341-359.

[2] Brest, Janez, Aleš Zamuda, Borko Bošković, Sašo Greiner & Viljem Žumer (2008). An analysis of the control parameters adaptation in DE. *Advances in Differential Evolution, Springer Berlin Heiderberg* 143, 89-110. doi:10.1007/978-3-540-68830-3_3.

[3] Rönkkönen, Jani (2009). *Continuous multimodal global optimization with differential evolution-based method*. Lappeenranta University of Tehnology. ISBN: 978-952-214-852-0.

[4] Saad, Mohd Sadli, Hishamuddin Jamaluddin & Intal Zaurah Mat Darus (2012). PID controller tuning using evolutionary algorithms. *WSEAS Transactions on Systems and Control* **4**, 139-149.

[5] Penttinen, Aki (2005). *FPGA:lle sulautetulla mikroprosessorilla toteutettu sähkökäytön säätöjärjestelmä*. Lappeenrannan teknillinen yliopisto, Sähkötekniikan osasto Teollisuuselektroniikan laitos.

[6] Aalto University, *Analogisen säädön verkkokurssi (2011b)*. PID-säätimen kokeellinen virittäminen. [PID controller tuning with experiments] ftp://autsys.aalto.fi/pub/control.tkk.fi/Kurssit/Verkkokurssit/AS-74.2111/kehittyneet/oppitunti12/kokeellinen.html.

[7] Bharti, Om Prakash, R.K. Saket, S.K. Nagar (2018). In proc. *2018 SICE Symposium of Control Systems*, Tokyo, Japan, March 9-11, 2018, 128-135.

[8] Maldonado, Yazmin, Oscar Castillo, Patricia Melin (2014). A multi-objective optimization of type-2 fuzzy control speed in FPGAs. *Applied Soft Computing*, 24, 2014, 1164-1174.

[9] Ramadan, E.A. M. El-bardini, M.A. Fkirin (2014), Design and FPGA-implementation of an improved adaptive fuzzy logic controller for DC motor speed control, *Ain Shams Engineering Journal* 5(3). 2014, 803-816.

[10] Chun-Fei Hsu, Bore-Kuen Lee (2011), FPGA-based adaptive PID control of a DC motor driver via sliding-mode approach. Expert Systems with Applications 38 (9) 2011, 11866-11872.

[11] Gürsoy, Handan, Mehmet Önder Efe (2016). Control System Implementation on an FPGA Platform. *IFAC-PapersOnLine* 49(25) 2016, 425-430.

[12] Yao, Minglin (2010), Realization of Fuzzy PID controller used in turbine speed control system with FPGA. *2010 International Conference on Future Information Technology and Management Engineering*, 261-264.

[13] Chan, Yuen Fong, M. Moallem & Wei Wang (2007). Design and implementation of modular FPGA-based PID controllers. *Industrial Electronics, IEEE Transactions on* 4, 1898-1906.

[14] Lima, João, Ricardo Menotti, João M. P. Cardoso & Eduardo Marques (2006). A methodology to design FPGA-based PID controllers. *Systems, Man and Cybernetics, IEEE Transactions on,* 2577-2583.

[15] Anumandla, Kiran Kumar, Rangababu Peesapati, Samrath L. Sabat, Siba K. Udgata & Ajith Abraham (2013). Field programmable gate arrays-based differential evolution coprocessor: a case study of spectrum allocation in cognitive radio network. *IET Computers & Digital Techniques 7*:5, 221-234.

[16] Alander, Jarmo, Digitaalitekniikan perusteet (2015). *Lukujärjestelmät ja koodit* [In Finnish, Digital electronics, number systems and codes]. Available: http://lipas.uwasa.fi/~TAU/AUTO1010/.

[17] Actel Corporation (2008). Zero-Power or NOT Zero-Power: That Is the Question. *Actel eZone*. http://www.actel.com/eZone/Q108/p3.html

[18] Chang, Chi-Ming, Jih-Gau Juang (2014). Real Time TRMS Control using FPGA and Hybrid PID controller. In *11th IEEE International Conference on Control & Automation (ICCA)*, June 18-20, 2014, Taichung, Taiwan, 983-988.

[19] Chang, Changyan, Yubo Yuan, Tianlin Jiang, Zhongjie Zhou (2016). Field programmable gate array implementation of a single-input fuzzy proportional-integral.derivative controller for DC-DC buck converters. *IET Power Electronics* **9**(6) 1259-1266.

[20] Lucas, Ricardo, R.M. Oliveira, C. B. Nascimento, M. S. Kaster (2015). Performance Analysis of an Adaptive Gaussian Nonlinear PID Control Applied to a Step-down CC-CC Converter. *24th IEEE International Symposium on Industrial Electronics (ISIE)*, 3-5 June 2015, 3-5 June 2015, 1022-1026.

[21] Chen, Yajuan, Qinghai Wu (2011). Design and Implementation of PID Controller Based on FPGA and Genetic Algorithm. *2011 IEEE International Conference on Electronics and Optoelectronics (ICEOE)*, v4-308-3011.

[22] Chen, Yanni, Bin Xie, Enrong Mao (2016), Electric Tractor Motor Drive Control Based on FPGA, *IFAC-PapersOnLine 49*(16) 2016: 271-276.

[23] Ponce, Pedro, Arturo Molina, Guillermo Tello, Luis Ibarra, Brian MacCleery, Miguel Ramirez (2015), Experimental study for FPGA PID position controller in CNC micro-machines, *IFAC-PapersOnLine* 48(3) 2015, 2203-2207.

[24] Aguirre, Adriana A., Leonardo D. Muñoz, César A. Martín, María J. Ramírez, Carlos A. Salazar (2018), Design of Digital PID Controllers Relying on FPGA-based Techniques, IFAC-PapersOnLine 51(4) 2018, 936-941.

[25] Hanhila, Mika (2015). *PID-säätimen optimointi differentiaalievoluutiolla* [Optimization of PID-controller by differential evolution], M.Sc. Thesis. University of Vaasa. https://www.tritonia.fi/download/gradu/6472

[26] Jiang, Yuchen, Shen Yin, Okyay Kaynak (2018). Data-Driven Monitoring and Safety Control of industrial Cyber-Physical Systems: Basics and Beyond. *IEEE Access* 6, 2018, 47374-47384.