A Portable Implementation on Industrial Devices of a Predictive Controller using Graphical Programming

Silviu Folea *Member IEEE*, George Mois *Member IEEE*, Cristina I. Muresan, Liviu Miclea *Member IEEE*, Robin De Keyser, Marcian Cirstea *Senior Member IEEE*

***Abstract*— This paper presents an approach for developing an Extended Prediction Self-Adaptive Controller employing graphical programming of industrial standard devices, for controlling fast processes. For comparison purposes, the algorithm has been implemented on three different FPGA (Field Programmable Gate Arrays) chips. The paper presents research aspects regarding graphical programming controller design, showing that a single advanced control application can run on different targets without requiring significant program modifications. Based on the time needed for processing the control signal and on the application, one can efficiently an easily select the most appropriate device. To exemplify the procedure, a conclusive case study is presented.**

***Index Terms*— Field programmable gate arrays, Predictive control, Benchmark testing, Real-time systems.**

I. Introduction

PREDICTIVE control has been used successfully in control applications in all fields of industrial activity, a fact that has triggered an increasing interest in the methodology during the last decade. The choice for predictive control, rather than other modern control concepts, is based on some series of important benefits such as: its intuitive principles, performance oriented design parameters, the ability to handle nonlinearities and its capability of taking into account various constraints (such as actuator constraints, safety constraints, quality constraints). Typically, predictive control has been used in the control of slow dynamics processes, such as thermal and chemical plants [1]. However, more and more model predictive control applications are directed towards dynamical systems with fast response times [2], [3], [4]. Until recently, the main model predictive control (MPC) limitation resulted from the long computational time needed for performing the optimization. The use of DSPs and FPGAs led to the reduction of the time needed for solving the constrained optimization problem with a period of tens or hundreds of microseconds [5], [6]. The large real-time computational complexity was managed until now by the use industrial computers and the papers present especially results obtained from personal computer implementations.

The purpose of this paper is to present an efficient and robust control solution for fast dynamic systems, using FPGA devices and the LabVIEW™ graphical programming environment. The motivation for using this solution is based on the fact that compared to hardware description languages such as VHDL, graphical programming is a more user-friendly configuration environment and offers a very short project development time [7], [8]. An application for a DC motor was chosen for validation and testing purposes. The DC motor supports a wide range of command rates and of execution time variations, without being damaged or broken. The particular application here refers to the control of the DC motor, as a part of the vacuum pumps used to maintain an efficient thermal isolation in the vacuum jacket of a train of three carbon isotopes separation columns. The efficiency of the isotope separation process, occurring at very low temperatures, is strongly dependent upon a strict operation of these vacuum pumps. These need to be carefully controlled since a failure of the vacuum leads to the compromise of the entire separation process [9]. The efficiency of the proposed solution was highlighted by comparing it to several different implementations, a PC based control system, one implemented on a real-time target and one on an ARM microcontroller, all of them running the controller. Finally, the same predictive control algorithm was implemented on three different FPGA chips: a Virtex-II, a Spartan-6 and a Zynq. The comparison between the three systems has shown that this type of complex algorithms can operate on cheaper FPGA chips, such as the Spartan-6 and Zynq, achieving not only the same levels of computational performance as their more complex and more expensive counterparts, but also important power savings. This comparison regarding the implementation on various FPGA targets and microcontrollers represents one of the main contributions and original elements in comparison to previous research [10], including also comparative tables and benchmarks for resource allocation in FPGAs. The paper also

This work was supported in part by the Romanian National Authority for Scientific Research CNDI-UEFISCDI under project number PCCA 155/2012.

Silviu Folea, George Moiș, Cristina I. Muresan and Liviu Miclea are with Technical University of Cluj-Napoca, Department of Automation, 26-28 Gh. Baritiu st., Cluj-Napoca, Romania (e-mail: {Silviu.Folea, George.Mois, Cristina.Pop, Liviu.Miclea} @aut.utcluj.ro).

Robin De Keyser is with Ghent University, Department of Electrical Energy, Systems and Automation, Technologiepark, 913, 9052 Gent, Belgium (e-mail: Robain.DeKeyser@UGent.be).

Marcian Cirstea is with Anglia Ruskin University, Department of Computing and Technology, Cambridge, CB1 1PT, England, UK. (e-mail: Marcian@ieee.org).

presents the diagrams implemented using LabVIEW™.

The choice of using an FPGA instead of a processor-based solution was motivated by several advantages. FPGAs have already been used in industrial control systems, being capable of providing an increased level of performance, while at the same time reducing the cost, size and power consumption of the actual implementation and improving reliability [11], [12], [13], [14], [15]. The ever-increasing sophisticated control algorithms can take advantage of the natural parallelism and increased resource density of the FPGA chips [16]. Thus, complex architectures, fully dedicated to the control algorithm to implement, can be developed [17]. The design and real-time implementation of control loops running at frequencies above 1 MHz is now possible with the use of these System-on-Chip digital reconfigurable platforms. Although still more expensive than DSPs and microcontrollers, they compensate through their compactness, all the building parts of the competitor solutions (CPU, RAM, bus) being placed inside a single capsule. While DSPs are aimed at implementing signal processing applications and can perform large amounts of computations, FPGA chips offer higher flexibility levels and transfer the printed circuit board complexity inside the device, on-chip. The work in [18] presents a systematic comparison between these two technologies along with their main advantages and drawbacks when used in control applications. FPGAs also provide the possibility of in-the-field programming, which allows the addition of other features to the controller, the implementation of further data post-processing algorithms, etc. They can also be dynamically reconfigured, enabling the controller to adapt to the needs of the plant. Thus, adaptation to changes in environmental conditions becomes possible.

A wide range of applications in the field of electrical systems employ FPGAs [19], [20], [21]. The authors in [19] developed a reliable low-complexity reusable digital controller, by using an FPGA implementation. The work in [20] presents an FPGA-based adaptive digital PI controller and emphasizes the advantages provided by FPGAs in the control of complex industrial processes. MPC was addressed for the control of power converters [22] and electric drives [23], FPGA-based solutions showing good control performance [24], [25].

The rest of the paper is structured as follows. The second section describes the EPSAC control principles, while Section III shows the steps completed for finalizing the EPSAC design and the methodology used for the FPGA implementation. Then, section IV provides information regarding the details of the hardware and of the software setup. The testing and validation of the proposed solution along its performance evaluation are synthesized in the section V and, finally, the concluding remarks are outlined in section VI.

II. EPSAC Control Principles

The EPSAC methodology is a typical member of the Model Based Predictive Control (MBPC) family. MBPC is a type of control which uses an on-line process model (in the control computer) for calculating predictions of the future plant output and for optimizing future control actions. The two key principles of model based predictive control consist in the explicit on-line use of the process model for forecasting the process output at future time instants and in the calculation of an optimal control action based on the minimization of a cost function [26]. The principle of the EPSAC control, presented in [26], is based on the minimization of the error between the specified reference trajectory and a future predicted process output. A cost function having the form:

$$\sum_{k=N_1}^{N_2}\left[r(t+k/t)-y(t+k/t)\right]^2+\lambda\sum_{k=0}^{N_u-1}\left[\Delta u(t+k/t)\right]^2 \quad (1)$$

will be minimized. The design parameters of the cost function are: N_2 – the maximum prediction horizon, N_u – the control horizon, N_1 – the minimum prediction horizon, λ – the weight parameter, $y(t)$ – the (measured) process output, $u(t)$ – the process input, $r(t)$ – the reference trajectory. The control signal in (1) is given by:

$$\Delta u(t+k/t)=u(t+k/t)-u(t+k-1/t)\text{, with (2)}$$

$$\Delta u(t+k/t)\equiv 0\text{, for } k\geq N_u\text{. (3)}$$

For minimizing (1), the choice of N_2 and N_1 plays an important role, as well as the estimation of the process output, y(t), over the prediction horizon N_1 to N_2. In the EPSAC approach, the prediction of the process output is done based on previous measurements of the process output and input signal, as well as some future values of the input signal.

For predicting the output, the generic model in (4) can be used:

$$y(t+k/t)=x(t+k/t)+n(t+k/t)\text{, (4)}$$

where $x(t)$ represents the process model output, while $n(t)$ is the process/model disturbance.

To predict the process output $y(t)$, $x(t+k|t)$ is computed based on an existing model of the process, while $n(t+k|t)$ is predicted using filtering techniques.

Assuming the process model for a single-input-single-output system is given by:

$$x(t)=\frac{B(q^{-1})}{A(q^{-1})}u(t)\text{, (5)}$$

the output model x may be predicted k samples ahead using previous values of the process model and of the control input u, considering that polynomials $B(q^{-1})$ and $A(q^{-1})$ in (5) are fully known.

The algorithm for computing the control signal required to minimize (1), uses also the concepts of free and forced response:

$$y(t+k/t)=y_{free}(t+k/t)+y_{forced}(t+k/t) \quad (6)$$

The component $y_{free}(t+k/t)$ can be easily computed using (4), by simply putting $u(t/t)=...=u(t+N_2-1/t)=u(t-1)$. The component $y_{forced}(t+k/t)$, however, is the effect of a sequence of step inputs. In matrix notation, y_{forced} may be computed as:

$$\begin{bmatrix} y_{forced}(t+N_1/t) \\ y_{forced}(t+N_1+1/t) \\ ..... \\ ..... \\ y_{forced}(t+N_2/t) \end{bmatrix}=\mathbf{G}\begin{bmatrix} \Delta u(t/t) \\ \Delta u(t+1/t) \\ ... \\ \Delta u(t+N_u-1/t) \end{bmatrix}=\mathbf{G}\cdot\mathbf{U} \quad (7)$$

where $G = \begin{bmatrix} g_{N_1} & g_{N_1-1} & \cdots & \cdots \\ g_{N_1+1} & g_{N_1} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ g_{N_2} & g_{N_2-1} & \cdots & g_{N_2-N_u+1} \end{bmatrix}$ and the parameters g_k are the coefficients of the unit step response. Using matrix notation, replacing the result in (7) into (6), gives:

$$\mathbf{Y} = \mathbf{Y}_{\text{free}} + \mathbf{Y}_{\text{forced}} = \mathbf{Y}_{\text{free}} + \mathbf{G} \cdot \mathbf{U} \qquad (8)$$

The cost function in (1) can be written in matrix notation as:

$$\begin{aligned} &(\mathbf{R}-\mathbf{Y})^{\mathbf{T}}(\mathbf{R}-\mathbf{Y}) + \lambda \mathbf{U}^{\mathbf{T}}\mathbf{U} = \\ &[(\mathbf{R}-\mathbf{Y}_{\mathbf{free}}) - \mathbf{G}\mathbf{U}]^{\mathbf{T}}[(\mathbf{R}-\mathbf{Y}_{\mathbf{free}}) - \mathbf{G}\mathbf{U}] + \lambda \mathbf{U}^{\mathbf{T}}\mathbf{U} \end{aligned} \qquad (9)$$

Minimizing (9) with respect to U leads to the optimal solution:

$$\mathbf{U}^{*} = (\mathbf{G}^{\mathbf{T}}\mathbf{G} + \lambda \mathbf{I})^{-1}\mathbf{G}^{\mathbf{T}}(\mathbf{R} - \mathbf{Y}_{\text{free}}) \qquad (10)$$

The first element, $\Delta u(t/t)$, in U* is then used to update the control signal:

$$u(t) = u(t-1) + \Delta u(t/t) \qquad (11)$$

The procedure is then repeated at the next sampling instant, when u(t+1) is computed based on the new measurement y(t+1). A pseudo-code of the EPSAC algorithm is given in Fig. 1.

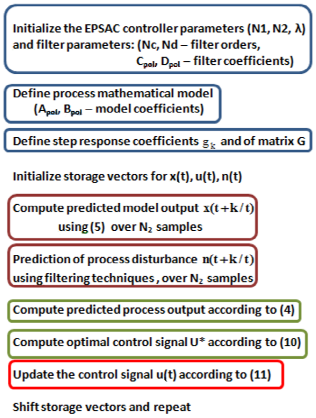

Fig. 1. Pseudo-code of the EPSAC control algorithm

III. EPSAC DC Motor Controller

The EPSAC predictive algorithm can be used for controlling various types of electrical systems [22], [27]. The DC motor provided the possibility of building a flexible stand for running the tests and of achieving rapid performance comparisons. This is just a case study for testing and for validating the FPGA-based implementation, which clearly demonstrates that the EPSAC predictive controller can be used in a wide range of applications. The block diagram of the system, including the controller, the driver, the signal processing module, the DC motor and the load implemented using a generator and a controlled resistive load, is presented in Fig. 2.

The CompactRIO™ embedded system used for implementation is a reconfigurable control and acquisition system providing high performance and reliability, and is programmable with LabVIEW™. The device includes a PowerPC real time controller running at 400 MHz and an extension module with digital input-output lines. Three different systems were used, one having a chassis with a Virtex-II FPGA, one having a chassis with a Spartan-6 device and another one with a Zynq programmable system on chip, including a real time dual core processor running at 667 MHz.

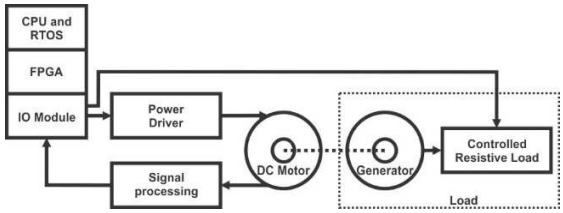

Fig. 2. System block diagram

The special architecture of the embedded system is built around two chips: the first one, on which a real-time operating system runs, and the second, the FPGA.

A. *Extraction of DC motor parameters for finalizing EPSAC design*

The EPSAC control strategy implemented in the FPGA has been tested in the closed loop trajectory control of a DC motor. The first step in the FPGA implementation of the EPSAC consists in determining a mathematical model of the process, that is the polynomials $A(q^{-1})$ and $B(q^{-1})$ in (5). To determine these polynomials, experimental identification techniques were employed.

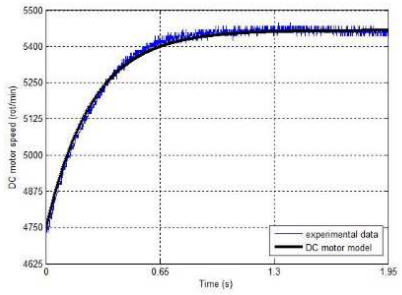

Fig. 3. Experimental data for process identification – speed rises

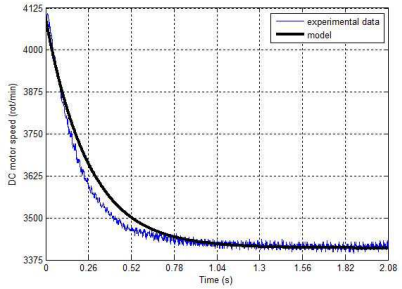

Fig. 4. Experimental data for process identification – speed decreases

Figures 3 and 4 present the experimental data used for identification of the DC motor model and the output of the identified process model compared to experimental data, when

the speed increases and decreases, respectively.

The DC motor output is its rotation speed, represented as experimental data in these figures, while the control input is the DC voltage supplied to the rotor. Prior to the experiment, the input voltage supplied was 70%. A step input of +10% was then applied to the rotor. For the speed decreasing, a -10% step was applied to the input.

Based on the shape of the step response, a transfer function was selected to model the process. The gain, as well as the time constants, are determined through identification techniques.

Using the determined transfer function, the polynomials $A(q^{-1})$ and $B(q^{-1})$ are computed based on a zero-order hold discretization, considering the sampling time Ts = 0.015sec, chosen according to Shannon theorem:

$$A(q^{-1}) = 1 - 0.94q^{-1} \ , \ B(q^{-1}) = 1.54q^{-1}. \quad (12)$$

In the EPSAC controller design, the maximum prediction horizon is chosen in order for the predicted signal to capture around 60% of the process dynamics [26]. Since there is no process time delay, the minimum prediction horizon may be chosen N_1 =1 sample, while N_2 = 10 samples, λ =0 and N_u = 1.With this choice of the prediction horizons, the controller designed was firstly tested in the MATLAB® simulation environment, using the transfer function of the model as the mathematical representation of the DC motor.

The controller designed and tested in the simulation environment was further implemented in a FPGA module, using the guidelines given in Sections IV and V, and employed in the closed loop control of the DC motor previously described.

B. FPGA implementation of EPSAC

This subsection shows the methodology that can be used for achieving the FPGA implementation of different types of control algorithms through graphical programming. The steps that have to be followed for reaching an optimal implementation method on the FPGA of the various control methods, realized using specific analysis and simulation environments, are briefly described below:

1) Rewriting the code used for simulation in the LabVIEW™ environment on the PC or on the real-time target;
2) Program testing using control vectors that were generated during simulation, for the control and for the controlled unit;
3) Data conversion from floating-point format to fixed-point format (FXP) or integer (INT);
4) The comparative testing of the implementations using the control vectors and the data available in the second step;
5) Go through steps 3 and 4 again until the stationary errors are acceptable; in the case of this paper it is assumed that a small error is acceptable.

The use of MATLAB® sequences of code using MathScript was avoided because it is not supported on the FPGA target.

IV. Hardware and Software Setup

Setting up the hardware and the software for implementing real-life control systems can be a troublesome task. On one hand, the hardware part requires taking into account various parameters including component compatibility, signal conditioning, placing and routing problems, while, on the other hand, the software part must consider the architecture of the equipment. However, in this case, the software application takes advantage of the facilities provided by graphical programming [28]. The following two subsections will present how the hardware and software had been developed in the case of the example application.

A. Hardware setup

For interfacing the DC motor, two printed circuit boards (PCBs) have been developed. The first PCB performs the processing of the signal received from the speed transducer, implying the amplification of the signal from the encoder (speed transducer) and signal filtering and its formatting for obtaining rectangular pulses. The board also includes the power driver for commanding the motor. The second PCB represents the load of the DC motor and consists of a digitally controlled resistive load. It is, in fact, a motor acting as a generator, with the same characteristics as the DC motor used, connected to a controlled resistive load. The stand used for verification consists in the embedded system, a power supply, the DC motor and the components described above (Fig. 5).

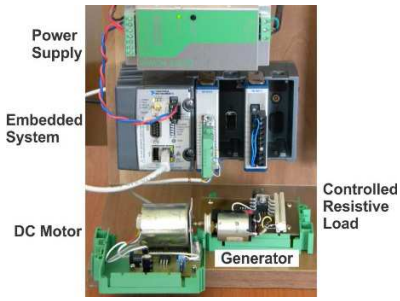

Fig. 5. The experimental stand

B. Software Setup

The FPGA implementation of EPSAC consists of three different *while* loops: the first loop is used for measuring the speed of the motor; the second loop is used for generating the PWM that changes the speed of the motor using a digital output line; the third loop represents the main loop, where the control algorithm is implemented.

The first loop measures the speed of the motor using a digital input line and can run at different speeds depending on the sensors that are used. This feature is made possible by including a time delay function, the implementation being based on a sequence of functions, which forces the execution order: two rising edges determine the time period.

The while loop is specific to LabVIEW™ FPGA implementations and is used for representing the continuous operation mode of the circuit to be realized.

The application running on the real-time target implies the opening of a connection to the FPGA program. The values of the parameters are set using property and method nodes, while the measured values are read inside a loop. Data are not transferred between the real-time target and the FPGA through DMA FIFO because only a small amount is transmitted, only for the graphical representation of the involved values, the entire control algorithm being implemented in the FPGA.

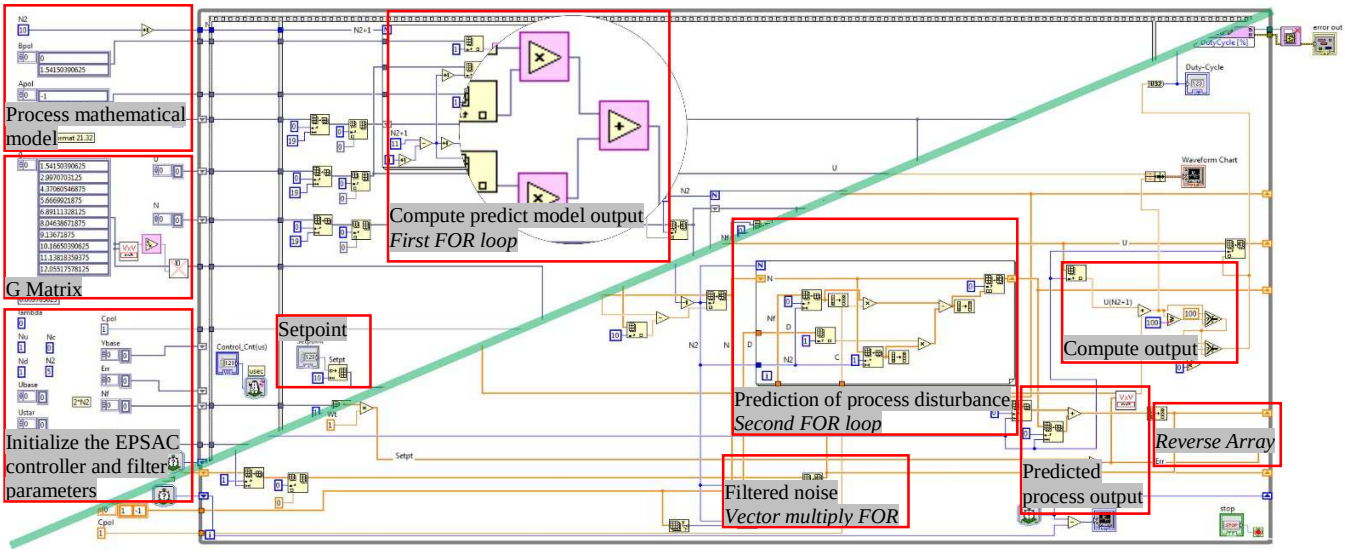

Fig. 6. The third loop -EPSAC control with fixed point data (upper part) and floating-point data (lower part)

The loop that implements the EPSAC method can be seen in Fig. 6. The part below the diagonal of the picture shows the virtual instrument using floating-point values, while the part above the diagonal shows the program using fixed-point data. The scope presents some of the special fixed point functions, high throughput multiplication and addition. Fig. 6 includes the blocks from Fig. 1, where the pseudo-code for the EPSAC control algorithm is presented, and the occupied FPGA resources, listed in Table II.

V. Testing and Validation

The testing and validation of the design represent an important step in the development of FPGA-based systems and are usually performed using simulator-specific environments. In the case of this paper, several benchmark programs were developed and run on various targets for comparing the computation performances achieved.

LabVIEW™ provides functions that access the real-time timers of the systems that were tested, offering resolutions in the order of milliseconds, microseconds or tens of nanoseconds for execution time or jitter measurement. Special frameworks, inside which the application could be tested, were developed. In the end, histograms including the execution time and jitter were realized for analysis.

First, a comparison between different platforms running the controller was done: a PC, a real-time controller, an FPGA, an FPGA including DSP blocks, a dual-core ARM and an ARM microcontroller. After this, the different implementation options offered by the graphical programming environment in the case of FPGA devices were studied. In the end, a parallel between the performances offered by three different FPGA technologies used for implementing the controller was made: a more expensive, but relatively old Virtex-II device and cheaper and newer Spartan-6 and Zynq chips. For the first benchmark, the EPSAC algorithm code was compiled on a PC and run locally.

In the second test, the benchmark was transferred and run on the real-time target. For the third set of tests, the benchmarks ran on the FPGA and finally, the same program was downloaded to a microcontroller. The first limitation of this solution consists of the data representation. In the case of the simulation and for the implementations on the PC, on the real-time target or on the microcontroller, the data are represented on floating-point, double type. When the FPGA is used, the data representation is fixed-point, using different formats, such as 14 bits for the integer word length and 32 bits for the entire word length.

The computational performance differs depending on the tested platform and the jitter is different than the data sheet value, depending on the implementation. The results, presented in Fig. 7, are the maximum reachable values and lead to the conclusion that the FPGA-based EPSAC controller can be used for fast dynamic processes.

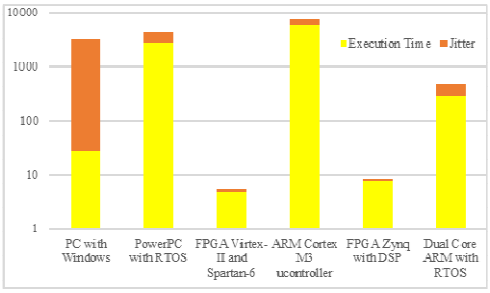

Fig. 7. Execution time and jitter for all targets (µs)

The FPGA target is more than 5 times faster than the PC, for the same control performance parameters. Another important parameter tested here is the variation on the loop execution time (jitter). In the case of the PC, the jitter varies depending on the tasks that run in parallel with the application at a specific point of time. The minimum resolution of the function used to measure the execution time in the case of the FPGA is 25 ns. For achieving maximum execution speed, the FPGA program includes mathematical operations specially designed for the FPGA target, allowing the specification of the data representation and its configuration, for both the input and output. The difficulty here consists in the computations involving arrays and in choosing the proper format for the data in fixed-point representation (integer word length and entire word length).

The use of reentrant or non-reentrant virtual instruments (VIs) in the control loop leads to different percentages in the FPGA resource utilization, in the case of the multiplier blocks, as it can be seen in Table I.

TABLE I. FPGA, DEVICE UTILIZATION

Virtex-II	Total Slices **(14336)**	Slice Regs **(28672)**	Slice LUTs **(28672)**	MULT 18X18 **(96)**	Exec. Time	Clock **(MHz)**
R VIs, **FXP 14.32**	49.8%	29.2%	41.0%	86%	5 µs	40.0
N-R VIs, **FXP 14.32**	49.5%	37.2%	41.0%	57%	9 µs	40.0
Spartan-6	Total Slices **(6822)**	Slice Regs **(54576)**	Slice LUTs **(27288)**	MULT 18X18 **(58)**	Exec. Time	Clock **(MHz)**
R VIs, **FXP 14.32**	Could not compile because not enough multipliers available					
N-R VIs, **FXP 14.32**	42.2%	18.9%	33.9%	89.7%	8.975µs	40.0
Zynq 7010 Artix-7	Total Slices **(4400)**	Slice Regs **(35200)**	Slice LUTs **(17600)**	DSP48 s **(80)**	Exec. Time	Clock **(MHz)**
only R VIs, **FXP 14.32**	Could not compile because not enough multipliers available					
only N-R VIs, **FXP14.32**	Could not compile because not enough multipliers available					
R & N-R VIs, **FXP14.32**	79.0%	46.8%	79.5%	80.0%	7.775µs	40.0

The conclusion that can be drawn from Table I is that it is difficult to predict which of the methods will always occupy less hardware resources, but, in general, as can be deduced from the presented cases, non-reentrant VIs lead to overall implementations requiring less area, that could be compiled successfully. Certainly, reentrant VIs lead to FPGA implementations offering faster execution speeds, as can be seen in this table.

In the case of the Zynq 7010 Artix-7 FPGA, which belongs to a more recent generation and for which a compiler from the year 2014 was used, a new situation emerged: the VI could be compiled only after a part of the VIs were configured experimentally as non-reentrant and the others as reentrant. Here, the occupied resources are far from the limit, and many possible configuration cases were obtained. The execution time represents a median value when compared with the one in the other cases, where the other 2 types of FPGAs were used and where all the VIs were set up to be either reentrant or non-reentrant. Other types of implementations on Artix-7 FPGA were not possible to be compiled due to an increase of the number of slices or DSP48s multipliers above the maximum limit.

The execution time can be improved by choosing reentrant VIs and preallocating clones for each instance of the blocks, in this way instantiating each one of them. In Table I, R stands for reentrant and N-R for non-reentrant VIs. The use of non-reentrant subVIs (subroutines) requires less multipliers, but more other FPGA resources are needed in this case, leading to an increase in the overall program execution time. The amount of occupied resources in the FPGA are specific to the LabVIEW™ implementation and can be different than that of a VHDL implementation.

Furthermore, the area occupied by the controller can differ depending on the device and software version. However, the advantage of the approach used in this paper lies in the short project completion time [29]. Experiments indicate that the system used for algorithm implementation allows clock speeds between 3 and 40 MHz. Therefore, if the process dynamics permits it, the clock frequency can be decreased, so that power savings can be achieved. This is also the case of the example application, where the execution time can be extended without affecting the control.

Table II presents the resource requirements and the execution time of some of the functions used in the control algorithm written in the FPGA. Based on data presented in this table, optimization can be performed regarding the resources in the FPGA that are used and regarding the execution time.

TABLE II. FPGA FUNCTIONS, RESOURCES USED AND EXECUTION TIMES

Virtex-II	Total Slices **(14336)**	Slice Regs **(28672)**	Slice LUTs **(28672)**	MULT 18X18 **(96)**	Exec. Time
Vect. Scalar Multiply	11.9%	7.7%	7.0%	41%	0.125µs
Vect. Multiply *For*	12.9%	10.2%	5.8%	4%	1.125µs
Add Vectors	14.4%	9.7%	7.6%	33-bit adder 10	0.125µs
Subtract Vectors	14.8%	9.7%	7.9%	33-bit sub. 10	0.125µs
Reverse Array	9.3%	6.3%	5.7%	–	0.1µs
1st *For* Loop	19.2%	13.9%	14.4%	8%	2.6µs
2nd *For* Loop	17.0%	12.8%	11.4%	41%	3.375µs
Spartan-6	Total Slices **(6822)**	Slice Regs **(54576)**	Slice LUTs **(27288)**	MULT 18X18 **(58)**	Exec. Time
Vect. Scalar Multiply	9.6%	4.2%	4.1%	69%	0.125µs
Vect. Multiply *For*	12.2%	5.6%	5.9%	6.9%	1.125µs
Add Vectors	11.5%	5.3%	7.9%	33-bit adder 10	0.125µs
Subtract Vectors	11.2%	5.3%	8.1%	33-bit sub. 10	0.125µs
Reverse Array	7.7%	3.5%	4.9%	–	0.1µs
1st *For* Loop	16.3%	7.6%	10.1%	13.8%	2.6µs
2nd *For* Loop	14.7%	7.0%	10.3%	69%	3.375µs
Zynq 7010 Artix-7	Total Slices **(4400)**	Slice Regs **(35200)**	Slice LUTs **(17600)**	DSP48s **(80)**	Exec. Time
Vect. Scalar Multiply	52.5%	29.2%	50.7%	50%	0.125µs
Vect. Multiply *For*	24.4%	31.2%	52.7%	5.0%	1.125µs
Add Vectors	20.5%	30.8%	53.2%	–	0.125µs
Subtract Vectors	20.5%	30.8%	53.3%	–	0.125µs
Reverse Array	19.4%	26.0%	45.8%	–	0.1µs
1st *For* Loop	29.0%	34.4%	62.7%	10.0%	2.6µs
2nd *For* Loop	53.2 %	33.5%	59.9%	50%	3.375µs

The blocks presented in Table II can be seen in Figure 6, and can be found in the EPSAC implementation and in the algorithm presented as pseud-code in Fig. 1.

The operations performed on vectors occupy more FPGA resources as the ones performed on scalars, and the loops, in the current case *for* loops, significantly increase the execution time. The information in this table allows the user to perform optimization actions in case the achievement of an application that requires less FPGA resources is desired, which compiles faster or which provides shorter execution times. Although different technologies, with release dates separated by several years, are compared, the differences between the results are relatively small. The reasons for choosing the newer technology, Spartan-6 or Zynq, consist in the reduced cost and power consumption of the FPGA, but the industrial equipment embedding these state of the art devices is still expensive.

Two data vectors, one for command and one for speed, generated through simulations, were used for evaluating the correctness of the proposed solution. The closed loop experimental results are presented in Figs. 8 and 9 together with the simulation results.

The simulated EPSAC controller reaches the new prescribed setpoint within 0.4 seconds with no overshoot, while the experimental results with the FPGA based EPSAC controller show that a similar performance is obtained with a settling time of 0.5 seconds and zero overshoot. The DC motor

rotation speed is given in Fig. 8, while the corresponding control input, required to drive the DC motor to its new prescribed position, is presented in Fig. 9.

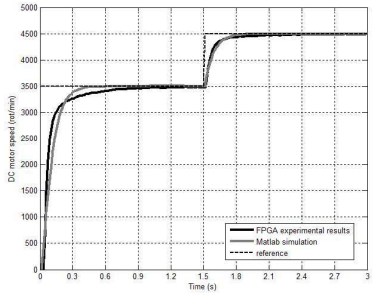

Fig. 8. Comparison between simulation and experimental data - Output amplitude (rot/min)

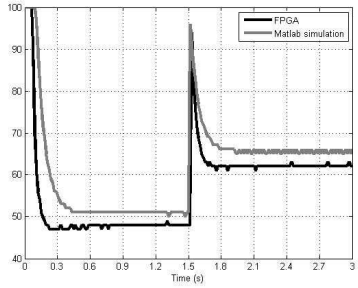

Fig. 9. Comparison between simulation and experimental data - Input (%)

The validation of the proposed implementation, first on the PC, then on the real-time target and, finally, on the FPGA, was made possible by using the data vectors generated through simulation. The validation step was important, especially for the FPGA implementation, because additional changes in the behaviour of the controller, caused by the translation from DBL to FXP representation, occurred. Taking into account the fact that the program compilation time lasts for approximatively 10 minutes, the simulation of the FPGA program was also an important action.

The behaviour of the FPGA-based solution and the robustness of the controller are emphasized through using three different DC motors from the same power class (Fig. 10). Fig. 11 shows that the command varies in different ways, because of the differences between the motors' parameters.

The execution time and jitter on the targets that were used are presented in Fig. 7. The execution time of the control loop (sampling time) in the FPGA or running on the real-time target has a constant value, 15 ms, while a variation of ±200 µs appears on the real time target. The PC implementation has loop execution time variations which can reach up to tens of milliseconds. The performance is higher when the execution time is shorter, but in the same time the jitter should be as low as possible.

In the vast majority of cases, for a numerical control system or for a "time critical" process, a better control system is obtained when the jitter is at its minimum. The PC does not belong to this category, having a short execution time, but a rather large jitter value. A jitter value which is hundreds of times smaller than the value of the execution time does not affect the control system, but a jitter having the same magnitude as the execution time negatively affects the entire system.

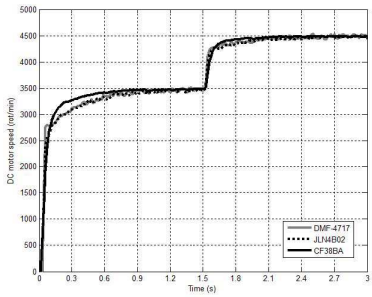

Fig. 10. Closed loop experimental results obtained using three different motors - Output amplitude (rot/min)

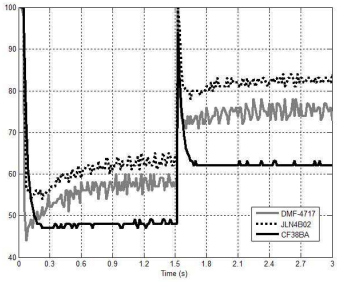

Fig. 11. Closed loop experimental results obtained using three different motors - Input (%)

VI. CONCLUSIONS

For the case of the EPSAC control strategy, the paper demonstrates the feasibility of the graphical programming controller design methodology as a fairly elegant, effective and user friendly method. Different implementations were compared against each-other regarding speed, hardware resources, real-time performance and programming aspects, under the following circumstances: graphical programs portability on as many industrial standard devices as possible, program scalability providing the possibility of running on resource limited and relatively cheap devices or on high performance systems. The results show that the FPGA solution offers a good compromise considering computational speed, hardware resource usage, power consumption and real-time performance. These advantages provide the possibility of using predictive control for fast dynamic processes. The results obtained justify the use of a graphical programming environment in industry for realizing fast synthesis of control algorithms and for shortening time to market of dedicated solutions.

REFERENCES

[1] Ridong Zhang; Anke Xue; Furong Gao, "Temperature Control of Industrial Coke Furnace Using Novel State Space Model Predictive Control," in *Industrial Informatics, IEEE Transactions on* , vol.10, no.4, pp.2084-2092, Nov. 2014.

[2] Xu, F.; Chen, H.; Gong, X.; Mei, Q., "Fast Nonlinear Model Predictive Control on FPGA Using Particle Swarm Optimization," in *Industrial Electronics, IEEE Transactions on*, vol.63, no.1, pp.310-321, Jan. 2016.

[3] Guzman, H.; Duran, M.J.; Barrero, F.; Zarri, L.; Bogado, B.; Gonzalez Prieto, I.; Arahal, M.R., "Comparative Study of Predictive and Resonant Controllers in Fault-Tolerant Five-Phase Induction Motor Drives," in *Industrial Electronics, IEEE Transactions on* , vol.63, no.1, pp.606-617, Jan. 2016.

[4] S. Chai, L. Wang, and E. Rogers, "A cascade MPC control structure for a

PMSM with speed ripple minimization," *IEEE Transactions on Industrial Electronics*, vol. 60, pp. 2978 – 2987, Aug. 2013.
[5] M. A. Stephens, C. Manzie, and M. C. Good, "Model predictive control for reference tracking on an industrial machine tool servo drive," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 2, 2013.
[6] Can Wang; Ming Yang; Weilong Zheng; Jiang Long; Dianguo Xu, "Vibration Suppression with Shaft Torque Limitation Using Explicit MPC-PI Switching Control in Elastic Drive Systems," in *Industrial Electronics, IEEE Transactions on*, vol.62, no.11, pp.6855-6867, 2015.
[7] M. Kaminski and T. Orlowska-Kowalska, "FPGA implementation of ADALINE-Based speed controller for the drive system with elastic joint,"*IEEE Transactions on Industrial Informatics*, vol.PP, no.99, 2012.
[8] L. Gomes, E. Monmasson, M. Cirstea, and J. J. Rodriguez-Andina, "Industrial electronic control: FPGAs and embedded systems solutions," in *IECON 2013 Conference Proceedings*, pp. 60 – 65, 2013.
[9] C.I. Muresan, E.H. Dulf, R. Both, R., "Comparative analysis of different control strategies for a train of cryogenic ^{13}C separation columns", *Chemical Engineering and Technology*, vol. 38, pp. 619-631, 2015.
[10] Folea, S.; Mois, G.; Muresan, C.I.; Miclea, L.; De Keyser, R.; Cirstea, M., "Implementation of an extended prediction self-adaptive controller using LabVIEW™," in *Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on* , vol., no., pp.883-888, 22-24 July 2015.
[11] Ricco, M.; Manganiello, P.; Monmasson, E.; Petrone, G.; Spagnuolo, G., "FPGA-Based Implementation of Dual Kalman Filter for PV MPPT Applications," in *Industrial Informatics, IEEE Transactions on* , vol.PP, no.99, pp.1-1, 2015.
[12] E. Monmasson, L. Idkhajine, and M. Naouar, "FPGA-based controllers," *IEEE Industrial Electronics Magazine*, vol. 5, pp. 14-26, 2011.
[13] Jamshidpour, E.; Poure, P.; Saadate, S., "Photovoltaic Systems Reliability Improvement by Real-Time FPGA-Based Switch Failure Diagnosis and Fault-Tolerant DC–DC Converter," in *Industrial Electronics, IEEE Transactions on* , vol.62, no.11, pp.7247-7255, Nov. 2015.
[14] M. Shahbazi, P. Poure, S. Saadate, and M. R. Zolghadri, "FPGA-based reconfigurable control for fault-tolerant back-to-back converter without redundancy," *IEEE Transactions on Industrial Electronics*, vol. 60, pp. 3360 – 3371, Aug. 2013.
[15] E. Monmasson and M. Cirstea, "Guest editorial special section on industrial control applications of FPGAs," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1250 – 1252, 2013.
[16] L. Idkhajine, E. Monmasson, and A. Maalouf, "Fully FPGA-based sensorless control for synchronous AC drive using an extended Kalmanfilter," *IEEE Transactions on Industrial Electronics*, vol. 59, pp. 3908 – 3918, Oct. 2012.
[17] E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan, and M. W. Naouar, "FPGAs in industrial control applications," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 2, pp. 224–243, 2011.
[18] C. Sepulveda, J. Munoz, J. Espinoza, M. Figueroa, and F. C. Baier, "FPGA v/s DSP performance comparison for a VSC-based STATCOM control application," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, 2012.
[19] A. Dinu, M. N. Cirstea, and S. E. Cirstea, "Direct neural-network hardware-implementation algorithm," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 5, pp. 1845 – 1848, 2010.
[20] J. Rodriguez-Araujo, J. Rodriguez-Andina, J. Farina, F. Vidal, J. Mato, and M. A. Montealegre, "Industrial laser cladding systems: FPGA-based adaptive control," *IEEE Industrial Electronics Magazine*, vol. 6, pp. 35 – 46, Dec. 2012.
[21] Oliveri, A.; Cassottana, L.; Laudani, A.; Riganti Fulginei, F.; Lozito, G.; Salvini, A.; Storace, M., "Two FPGA-Oriented High Speed Irradiance Virtual Sensors for Photovoltaic Plants," in *Industrial Informatics, IEEE Transactions on* , vol.PP, no.99, pp.1-1, 2015.
[22] Amin; Bambang, R.T.; Rohman, A.S.; Dronkers, C.J.; Ortega, R.; Sasongko, A., "Energy Management of Fuel Cell/Battery/Supercapacitor Hybrid Power Sources Using Model Predictive Control," in *Industrial Informatics, IEEE Transactions on*, vol.10, no.4, pp.1992-2002, 2014.
[23] Vazquez, S.; Leon, J.I.; Franquelo, L.G.; Rodriguez, J.; Young, H.A.; Marquez, A.; Zanchetta, P., "Model Predictive Control: A Review of Its Applications in Power Electronics," in *Industrial Electronics Magazine, IEEE* , vol.8, no.1, pp.16-31, March 2014.
[24] Damiano, A.; Gatto, G.; Marongiu, I.; Perfetto, A.; Serpi, A., "Operating Constraints Management of a Surface-Mounted PM Synchronous Machine by Means of an FPGA-Based Model Predictive Control Algorithm," in *Industrial Informatics, IEEE Transactions on* , vol.10, no.1, pp.243-255, Feb. 2014.
[25] Zhixun Ma; Saeidi, S.; Kennel, R., "FPGA Implementation of Model Predictive Control With Constant Switching Frequency for PMSM Drives," in *Industrial Informatics, IEEE Transactions on* , vol.10, no.4, pp.2055-2063, Nov. 2014.
[26] R. De Keyser, "Model based predictive control," *UNESCO Encyclopaedia of Life Support Systems (EoLSS)*, vol. 83, 2003. article 6.43.16.1, Eolss Publishers Co Ltd, Oxford, ISBN 0 9542 989 18-26-34.
[27] C. S. Lim, N. A. Rahim, W. P. Hew, and E. Levi, "Model predictive control of a two-motor drive with five-leg-inverter supply," *IEEE Transactions on Industrial Electronics*, vol. 60, pp. 54 – 65, Jan. 2013.
[28] T. Orlowska-Kowalska and M. Kaminski, "FPGA implementation of the multilayer neural network for the speed estimation of the two-mass drive system," *IEEE Transactions on Industrial Informatics*, vol. 7, pp. 436 –445, Aug. 2011.
[29] A. Hace and M. Franc, "FPGA implementation of sliding mode control algorithm for scaled bilateral teleoperation," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, 2012.

Silviu C. Folea received the degree in Control Systems in 1995, and the Ph.D in 2005 from Technical University of Cluj-Napoca, Romania. He is currently associated professor within the Automation Department of the same university. His research interests include: embedded and reconfigurable systems, data acquisition systems, wireless sensor networks, and graphical programming.

George Mois received the degree in Control Systems in 2008, and the Ph.D. in 2011 from Technical University of Cluj-Napoca, Cluj-Napoca, Romania. He is currently lecturer within the Automation Department of the same university. His research interests include embedded system design, digital design, and wireless sensor networks.

Cristina I. Muresan received the degree in Control Systems in 2007, and the Ph.D. in 2011 from Technical University of Cluj-Napoca, Romania. She is currently lecturer within the Automation Department of the same university. Her research interests include modern control strategies, such as predictive algorithms, fractional order control, time delay compensation methods and multivariable systems.

Liviu Miclea received the Ph.D. in Automatic Systems in 1995 from Technical University of Cluj-Napoca, Cluj-Napoca, Romania. He holds a currently professor position in the Department of Automation within the same university. His research interests include: design for testability, automatic testing, computer aided design, distributed systems, agent systems, CPSs, cloud computing.

Robin De Keyser received the M.Sc. degree in electro-mechanical engineering in 1974 and the Ph.D. in control engineering in 1980 from Ghent University, Belgium. He is currently senior professor of Control Engineering at the Faculty of Engineering, Ghent University. His research activities include model predictive control, autotuning and adaptive control, modeling and simulation, and system identification.

Marcian N. Cirstea (M'97-SM'04) received the degree in electrical engineering in 1990 from Transilvania University of Brasov, Romania, and the Ph.D. (1996) from Nottingham Trent University, Nottingham, UK. He is currently Professor of Industrial Electronics and Head of the Computing and Technology Department at Anglia Ruskin University, Cambridge, UK, after having worked previously for De Montfort University, UK. His research is focused on digital controllers for power electronics. He has published over 135 works in this field and has delivered a number of international tutorials/presentations. In January 2016 his achievements were celebrated through the prestigious award of the Doctor Honoris Causa title by Transilvania University of Brasov, Romania.