PROGRAMMABLE CONTROL SYSTEM WITH APPLICATIONS IN DIRECT CURRENT MOTORS CONTROL

Andrei COZMA, Dan PITICA
Applied Electronics Department, Technical University of Cluj Napoca, Romania
E-mail: andrei.cozma@ael.utcluj.ro, dan.pitica@ael.utcluj.ro

Abstract - This paper presents a programmable control system with applications in Direct Current (DC) motors speed control. The system consists of a signal acquisition module, a programmable high speed Digital Signal Processor (DSP), a Microcontroller Unit (MCU), an output module and a communication module. A speed and position sensor integrated in the data acquisition module provides through an I2C bus data related to the motor's shaft speed, position and rotation direction. The DSP can perform basic arithmetic operations, finite and infinite impulse response filtering and fuzzy logic operations at high speed and with a high degree of parallelism on the data received from the signal acquisition module. The MCU is a soft processor core with a RISC instruction set and it is used for controlling the operation of all the other modules and for implementing more complicated control algorithms that cannot be performed by the DSP. The output module contains a Pulse Width Modulated (PWM) block for generating the DC motor's control signal. The communication module transfers data to/from a PC through a USB connection. By integrating all the above mentioned modules into one single chip a complete real time control core is provided.

Keywords: *automatic control, direct current motor, digital signal controller*

I. INTRODUCTION

Nowadays programmable Microcontroller Units (MCU) and Digital Signal Processors (DSP) are used to implement the control algorithms of the complex modern control systems. The MCUs are preferred for running control functions that have a great degree of complexity but do not require a very fast update rate while DSPs are preferred for computation intensive signal processing tasks. MCUs are optimized to perform a wide array of logical, diagnostic and arithmetic operations on almost any combination of input data from various sources, while DSPs are very efficient at repetitive, numerically intensive tasks [1]. This kind of control systems also provide great flexibility since they are programmable and can perform a variety of functions without modifying the hardware. Compared to analog systems, performing signal manipulation with digital systems has numerous advantages: systems provide predictable accuracy, they are not affected by component aging and operating environment, and they permit advanced operations which may be impractical or even impossible to realize with analog components [2].

Modern control systems are implemented as Systems on Chip (SoC) and combine the advantages provided by the MCUs and DSPs. This kind of chip is a high-performance multiprocessor system which incorporates various types of hardware cores: programmable processors, Application Specific Integrated Circuit (ASIC) blocks, on-chip memories, peripherals, analog components, and various interface circuits [3]. Having the complete controller on a single chip allows the hardware design to be simple and very inexpensive. By combining the MCU's control-orientated attributes with the DSP's fast calculation capability a new category of device has emerged, known as the digital signal controller (DSC).

The actuator is an indispensable part of any control system. It converts electric, pneumatic, or hydraulic energy into mechanical motion. The most common type of actuator is the electric motor. Depending on the type of power they use electric motors are classified as either Direct Current (DC) or Alternating Current (AC). AC motors tend to be smaller, more reliable, and less expensive than the DC motors but they generally run at a fixed speed that is determined by the line frequency. DC motors have speed-control capability, which means that speed, torque, and even direction of rotation can be changed at any time to meet new conditions [4]. One technique for controlling a DC motor's speed is PWM. In this system, power is supplied to the motor in the form of DC pulses of a fixed voltage. The width of the pulses is varied to control the motor's speed. If the frequency of the pulses is high enough then the motor's inductance averages them, and it runs smoothly [4].

This paper introduces a DC motor speed control application based on a programmable control system. The purpose of the control system is to adjust the speed of a permanent magnet DC motor so that it follows a reference speed trajectory. The controller consists of two parts: a speed controller implemented in the DSP and a supervisor implemented in the MCU. By combining the speed and signal processing power of a DSP with the flexibility given by a MCU the presented system provides an excellent environment for implementing control algorithms that can range from very simple ones like the classical Proportional Integral Derivative (PID) control, to fuzzy or hybrid control algorithms. Also more complex

Manuscript received November 23, 2010; revised April 3, 2011

control structures can be implemented like cascade controllers or supervisory control. Both the DSP and MCU are programmable thus making the system highly flexible. The paper is organized as follows: Section II gives a general description of the system's architecture and presents the role of each of the system's building blocks, Section III presents a particularization of the control system for a DC motor speed control application, Section IV describes in detail the design of the speed controller, Section V provides information related to the hardware used for implementing the system and presents some experimental results and Section VI presents conclusions.

II. SYSTEM ARCHITECTURE

Fig. 1 presents the block diagram of the programmable control system with the connections and the data flow between the system's components.

A. Input Module

The input signals of the control system can be either analog or digital, depending on the type of used sensors. An *I2C Bus Controller* block is responsible for reading data from multiple sensors connected to a common I2C bus. The I2C addresses of the sensors are configured by the MCU through the system bus, and the *I2C Bus Controller* reads the configured sensors in a serial fashion and stores the data read from each sensor in a separate data buffer. A 16 bit ADC is used for digitizing the input analog signals. The analog signals are multiplexed at the ADC's input and the results of the analog to digital conversion are stored in separate data buffers by the means of a programmable demultiplexer. Both the multiplexer and demultiplexer can have two operation modes: automatic mode, when the input/output selection is done automatically based on a configuration programmed by the MCU; manual mode, when the input/output selection is controlled by the MCU.

B. Programmable DSP core

The programmable DSP core is responsible for processing the data received from the *Input module* and is able to perform a number of specialized data processing functions at high speed and with a high degree of parallelism. These functions enable the system to process the input signals and also to implement control algorithms like PID, fuzzy or hybrid algorithms. The instruction set can be extended to incorporate new basic data processing functions or to add compound instructions based on the basic functions. All the computations done by the DSP core are on 16 bit fixed point numbers having one sign bit and 15 fractional part bits. This format is known as Q15. The data coming from the input module is fetched by the *Input Controller* block from which the DSP core will extract the data when appropriate. The *Output Controller* block takes the data that was processed by the DSP core and passes it to the *Output module* or/and to the *Communication module*.

The programmable DSP core contains multiple DSP blocks which run in parallel and are able to communicate with each other through shared memories. The architecture of the DSP core is completely scalable and the number of DSP blocks that are incorporated in the DSP core, as well as the number of inputs and outputs can be selected at synthesis time depending on the system's characteristics. The programs ran by the DSP blocks are loaded by the MCU into the internal instruction and data memories of the DSP blocks either from a ROM memory or from a PC connected through USB to the system. The instruction memory of each DSP block can store 64 instructions. Each instruction is 32 bits wide. A 4 stages pipeline is implemented for instruction execution and up to 2 arithmetic instructions can be executed in parallel.

The *Output Controller* receives the data output by the programmable DSP and transfers it through the system bus to the *Output module* and to the *Communication module*. The operation of this block is programmed by the MCU.

More details related to the DSP core design can be found in [7].

D. System bus

The system bus is implemented using MicroBlaze's primary I/O bus, the CoreConnect bus. This bus is an

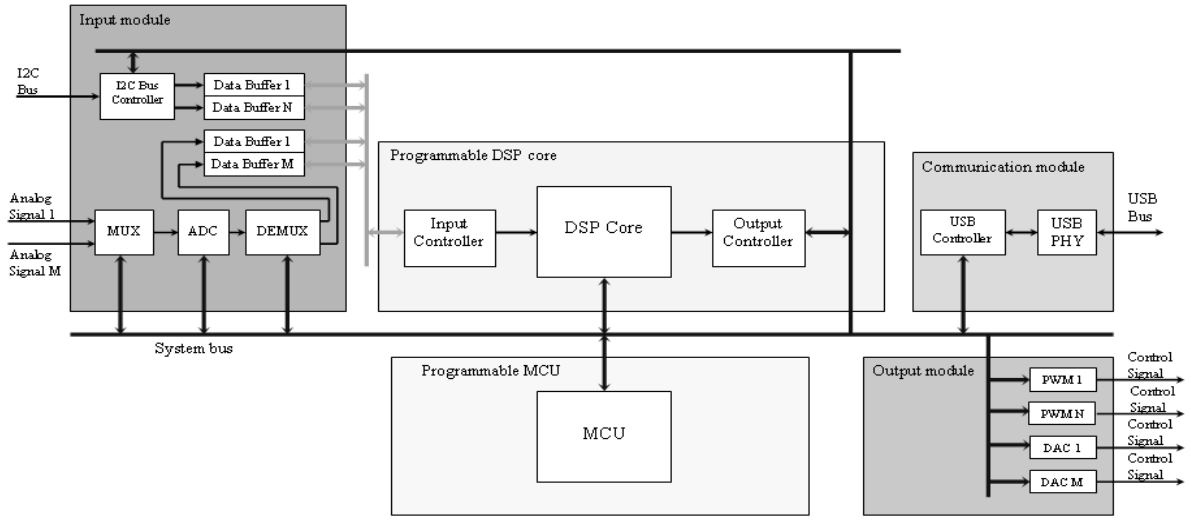

Figure 1. Programmable control system block diagram

IBM-developed on-chip communications link that enables chip cores from multiple sources to be interconnected to create entire new chips. CoreConnect technology eases the integration and reuse of processor, system and

peripheral cores within standard product platform designs to achieve overall greater system performance. The CoreConnect bus architecture includes the Processor Local Bus (PLB), the On-chip Peripheral Bus (OPB), a bus bridge, two arbiters, and a Device Control Register (DCR) bus [8]. High-performance peripherals connect to the high-bandwidth, low-latency PLB. Slower peripheral cores connect to the OPB, which reduces traffic on the PLB. There are bridging capabilities to the competing AMBA bus architecture allowing reuse of existing SoC-components.

E. Output module

This module is responsible for converting the digital control signals received from either the MCU or the DSP core into analog control signals. It contains multiple PWM generators and a high speed DACs. The PWM generators have a 16 bit resolution and the frequencies of the output waves are programmable.

F. Communication module

The Communication module is able to send/receive data to/from a PC through an USB connection. The system sends to the outside world data related to the system's operation and receives configuration parameters and programs for the DSP core.

III. DC MOTOR CONTROLLER ARCHITECTURE

Fig. 2 presents a block diagram of the DC motor controller with the connections and the data flow between the system's components.

A. I2C Speed and Position Sensor

The I2C Speed and Position Sensor computes the speed and position of the motor's shaft based on the signals received from two Hall sensors situated next to the motor's shaft, at a 90° angle from each other. On the motor's shaft there is a disc with three magnets, which pass in front of the Hall sensors as the shaft is turning.
The magnets are placed 120° apart from each other. When a magnet passes in front of a Hall sensor an electrical pulse is generated. Four 1MHz counters are used to count the number of clock cycles between two consecutive pulses generated by a Hall sensor. For each sensor two counters are used: one for counting the cycles between two consecutive rising edges and one for counting the cycles between two consecutive falling edges. Once all the four counters have generated a result, the results are averaged and the final speed value is ready to be read by the other blocks through the I2C bus.

The position of the motor's shaft is determined by counting the number of rotations that the motor is performing in a direction since the last position data was read from the sensor. The accuracy of the position reading is of 1/3 of a full shaft rotation.

The sensor contains an I2C slave module which allows it to communicate with the rest of the system through and I2C bus. The most significant 3 bits of the module's I2C address can be externally configured by connecting the I2C address pins 1:3 to either ground or VCC.

B. DSP

The DSP processes the data related to the motor's speed and position and computes the command for controlling the motor's armature voltage.
Two DSP programmable cores are used to implement the data processing and motor speed control algorithms. The first DSP core computes the speed of the motor in rotations per second from the data receiver from the I2C Speed and Position Sensor block. The computed speed is passed to the second DSP core and is also sent to the module's output to be used by the communication module and the MCU. Using a Proportional and Integral (PI) control algorithm the second DSP core computes the motor command so that the motor's speed follows the reference speed received from the MCU. The computed command represents the fill factor of the PWM signal used to control the motor's armature voltage. A detailed description of the control algorithm is presented in Section V.

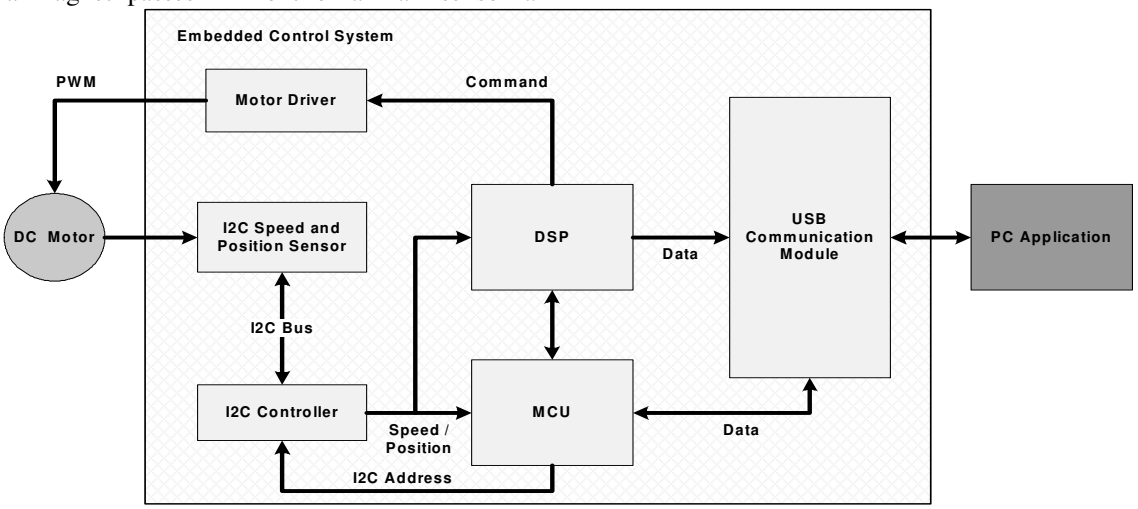

Figure 2. Control system block diagram

Both DSP cores have the same internal structure, the only difference between them is the program that is executed. The DSP core structure was simplified in respect to what was presented in Section II: the unused Fuzzy Processing Unit was removed from the design to reduce the size. The heart of a DSP core is the control

unit, which controls the operation of all the other internal blocks based on the program stored in the instructions memories. The fetch and decode unit fetches the instructions from the instructions memory, performs some preliminary decoding and passes the decoded instructions to the control unit. The ALU executes all the arithmetic and logical operations using data from the registers, data and coefficients memories. In case of logical operations that are executed as part of a conditional jump instruction the ALU also signals to the fetch and decode unit the result of the logical operation in order to properly compute the address of the next instruction to be fetched from the instructions memory. An IO controller manages the data transfer with the outside world.

The input data of the DSP is stored in four circular buffers from where it is retrieved by the DSP cores for further processing. The four inputs of the DSP are the actual speed and position of the motor and the reference speed and position to be used by the control algorithm. An input controller manages the interaction between the DSP cores and the input buffers. The data output by the DSP cores is stored in two output circular buffers from where it is retrieved by the output controller block to be dispatched to other components in the system.

C. MCU

The MCU controls the operation of all the other modules and implements supervisory control. The following features were selected for the processor's implementation:

- Clock frequency: 50MHz
- Local memory size: 32KB
- Standard peripherals : Interrupt controller, 32 bit timer, RS232 controller, GPIO controller

Besides the standard peripherals two custom peripherals were implemented:

- An I2C controller for controlling the operation of the I2C speed and position sensor.
- A peripheral with four 32 bit registers that can be used for reading the speed, position and command values from the system and for providing the reference speed to the DSP.

The advantage of using an open source soft core like the Xilinx MicroBlaze comes from the fact that it is provided as part of a embedded development kit that includes compilers and other libraries [9].

D. Motor Driver

The Motor Driver generates a 50KHz PWM signal to control the motor's armature voltage.

IV. SPEED CONTROLLER DESIGN

Fig. 3 presents the block diagram of the DC motor speed and position controller. The aim of the controller is to control the speed of the DC motor and the number of rotations the DC motor performs so that at the end of a control sequence the motor would have rotated with a specified average speed following a known speed trajectory and would have performed a specified number of rotations.

The controller is divided in two components: a speed controller implemented in the DSP and a position controller implemented in the MCU. The reference speed trajectory is generated by the MCU.

A. DSP speed controller

The speed and position sensor provides to the system speed information that specifies the number of clock cycles generated by a 1MHz clock in the period of time necessary for the motor's shaft to perform 1/3 of a complete rotation. In order for the speed controller to use this information it has to be first converted into rotations per second. The conversion is done using the following equation:

$$S_{RPS} = \frac{10^6}{3 * C}[rps] \tag{1}$$

where: - S_{RPS} – speed in rotations per second (*rps*)
- C – 1 MHz clock cycle counts received from the speed sensor

The job of the Speed Processor block is to perform this conversion. Since all the operations inside the DSP are done using the Q15 fixed point format equation (1) cannot be directly implemented so the following equation is used:

$$S = \frac{2^{11}}{C} = 0.005 * S_{RPS} \tag{2}$$

where: - S – speed
- C – 1 MHz clock cycle counts received from the speed sensor

The Speed Processor block is implemented by one of the two DSP cores present in the DSP. The computed speed is passed to the Speed Controller block which is implemented by the second DSP core. Based on the current speed of the motor and the reference speed received from the MCU the speed controller block computes the motor's command using a PI control algorithm according to equation (3).

$$C(kT) = k_P * (S_{\mathrm{Re}f}(kT) - S_{meas}(kT)) + k_I * \sum_{k=1}^{N} (S_{\mathrm{Re}f}(kT) - S_{meas}(kT)) \tag{3}$$

where: - T – sampling period
- $S_{Ref}(kT)$ – reference speed at time kT
- $S_{meas}(kT)$ – measured speed at time kT
- $C(kT)$ – command at time kT
- k_P – proportional gain
- k_I – integral gain

The total execution time of the control algorithm is given by the execution time of the slowest DSP core, which is the Speed Controller which requires 44 clock cycles to execute the program. On a Spartan 3e FPGA the maximum frequency at which the entire system can operate is around 52MHz, thus giving a controller maximum controller update rate of around 1.2MHz.

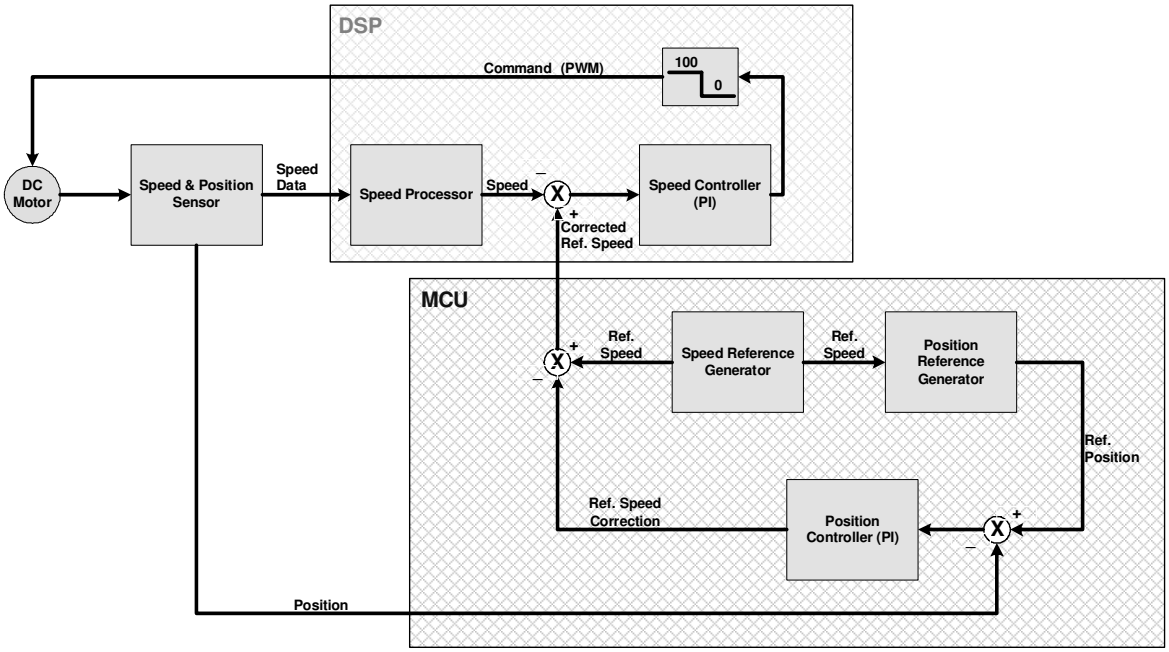

Figure 3. Speed controller block diagram

B. MCU position controller

In the current design the job of the MCU is to generate the speed trajectory that the motor must follow, to supervise the number of rotations that the motor's shaft has performed and to correct the speed trajectory in such a way that at the end of a control cycle the number of rotations gets as close as possible to the desired number. The speed trajectory is generated based on two parameters: average speed and number of rotations that must be performed with the specified average speed. The speed trajectory is computed so that it has a trapezoidal form, with equal rise and fall times, which expand for 15% (7.5% rise and 7.5% fall) of the total time needed to complete the specified number of rotations with the specified average speed. Fig. 4 presents the general shape of the speed trajectory, and the main parameters that characterize the trajectory. The key parameter for computing the speed trajectory is the slope K_v, because it is used to determine the speed value at every moment of time during the rise and fall stages, and also the maximum speed. The slope is determined using the following equation:

$$v_{med} = \frac{1}{T}\left[\int_0^{T_r} K_v \cdot t\,dt + \int_{T_r}^{T-T_f} K_v \cdot T_r\,dt + \int_{T-T_f}^{T} K_v \cdot (t-(T-T_f))\,dt\right] \quad (4)$$

$$\Rightarrow K_v = \frac{v_{med}}{c \cdot (1-c) \cdot T}$$

where v_{med} represents the average speed and

$$c = \frac{T_r}{T} = \frac{T_f}{T} \quad (5)$$

The speed reference is updated with a periodicity T of 10ms. When a new speed value is computed also an ideal position value is computed using the following equation:

$$p(kT) = p((k-1)T) + S((k-1)T) * T \quad (6)$$

where: - T – sampling period
- $p(kT)$ – position value at time kT
- $S(kT)$ – reference speed at time kT

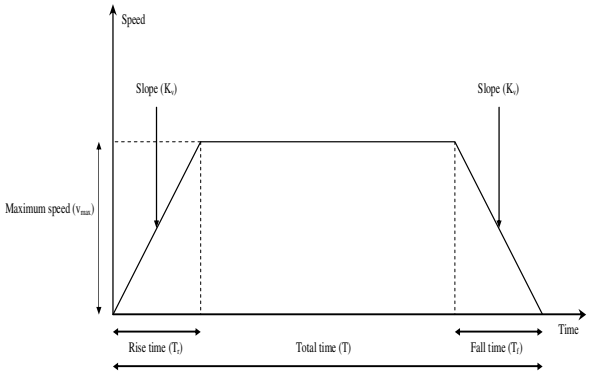

Figure 4. Speed trajectory

The error between the ideal position and the real position received from the position sensor is used to correct the speed reference provided to the DSP speed controller according to equation (7).

$$S_C(kT) = S_I(kT) + k_P * (p_I(kT) - p_R(kT)) + k_I * \sum_{k=1}^{N} (p_I(kT) - p_R(kT)) \quad (7)$$

where: - T – sampling period

- $S_c(kT)$ – corrected speed reference at time kT
- $S_I(kT)$ – ideal speed reference at time kT
- $p_R(kT)$ – real position at time kT
- $p_I(kT)$ – ideal position at time kT
- k_P – proportional gain
- k_I – integral gain

V. EXPERIMENTAL RESULTS

The DC motor control system was implemented on a Digilent Nexys2 1200 FPGA development board. For this FPGA the maximum frequency at which the system can operate is 52MHz and the used gate count is 67% of the total gate count of the FPGA. Below is presented a list of the used hardware components:

- Digilent Nexys2 1200 FPGA board
- Digilent Pmod HB5 board (H bridge)
- 12V power source
- USB cable
- 12V permanent magnet DC motor with integrated Hall sensors

Table 1 presents a set of experimental results that were obtained by using the control system with this hardware.

Ref. Speed [rps]	Ref. Rotations	Actual Speed [rps]	Actual Rotation No
40	800	39.52	799.67
50	1500	50.24	1501.00
60	1800	60.57	1801.67
70	2100	69.91	2098.33

Table 1. Position control experiments results

Fig. 5 presents the ideal and real speed trajectories for a control experiment having the reference speed set to 50[rps] and the number of reference rotations set to 1500.

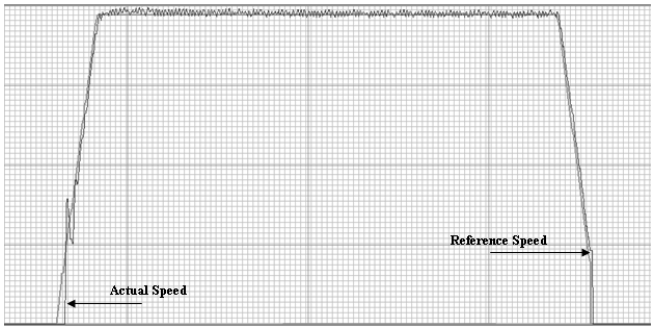

Figure 5. Position control experiment plot

VI. CONCLUSIONS

In this paper was presented a complete real time control solution that can be used in a wide range of automation applications. Also a particularization of the solution for a permanent magnet DC control application was introduced. By combining the advantages of a programmable DSP with those of a MCU high performance control algorithms can be implemented. A programmable DSP core was introduced, which is to be used for real time processing of the input signals and also for real time execution of control algorithms. The architecture of the DSP core is completely scalable and the number of DSP blocks that are incorporated in the DSP core can be selected at synthesis time. The design of the system is oriented towards flexibility and scalability. Processing blocks as well as inputs and outputs can be easily added or removed from the system to suit the particular needs of various applications. As shown by the DC motor control application the generic control system can be very easily adapted to the needs of a particular application by adding or removing features for both the DSP and the MCU and by adding custom input and output blocks. The presented system has many similarities with the existing Digital Signal Controllers (DSC) in terms of operation and targeted applications, but it also brings a set of new features that are useful for implementing complex control algorithms.

REFERENCES

[1] Ross Bannatyne, "The evolution of the digital signal controller", Motorola Semiconductor Products, 2009
[2] E. C. Ifeachor and B. W. Jervis, "Digital Signal Processing: A Practical Approach", Addison Wesley Longman, Inc., Menlo Par k, CA, U.S.A., 1993
[3] Mika Kuulsa, "DSP Processor Core-Based Wireless System Design", PhD Thesis, 18th of August 2000
[4] "Modern Control Technologies: Components and systems, Second edition", Thomson Delmar Learning, 2001
[5] David A. Patterson,John L. Hennessy, "Computer organization and design: the hardware/software interface", Morgan Kaufmann Publishers, 2005
[6] "Microblaze Processor Reference Guide", Xilinx, 2008
[7] Andrei Cozma, Dan Pitica, "Design of a Programmable Control System", ACTA Technica Napocensis, Volume 51/1, 2010
[8] http://en.wikipedia.org/wiki/MicroBlaze, 2009
[9] Peter Wilson, "Design Recipes for FPGAs", Newnes, 2007