A Field Programmable Gate Array Based Modular Motion Control Platform

Osman Koç[#1], Ahmet Teoman Naskali[#2], Emrah Deniz Kunt[#3], Asıf Şabanoviç[#4]

[#] *Mechatronics Engineering, Faculty of Engineering and Natural Sciences, Sabanci University Istanbul, TURKEY*

[1]kocosman,[2]teoman,[3]edkunt,[4]asif@sabanciuniv.edu

Abstract— **The expectations from motion control systems have been rising day by day. As the systems become more complex, conventional motion control systems can not achieve to meet all the specifications with optimized results. This creates the necessity of fundamental changes in the infrastructure of the system. Field programmable gate array (FPGA) technology enables the reconfiguration of the digital hardware, thus dissolving the necessity of infrastructural changes for minor manipulations in the hardware even if the system is deployed.**

An FPGA based hardware system shrinks the size of the hardware hence the cost. FPGAs also provide better power ratings for the systems as well as a more reliable system with improved performance. As a trade off, the development is rather more difficult than software based systems, which also affects the research and development time of the overall system.

In this paper a level of abstraction is introduced in order to diminish the requirement of advanced hardware description language (HDL) knowledge for implementing motion control systems thoroughly on an FPGA. The intellectual property library consists of synthesizable hardware modules specifically implemented for motion control purposes. Other parts of a motion control system, like user interface and trajectory generation, are implemented as software functions in order to protect the modularity of the system. There are also several external hardware designs for interfacing and driving various types of actuators.

Keywords— **Field Programmable Gate Array, Motion Control, hardware software co-design, modular, pantograph**

I. INTRODUCTION

As today's multi degree-of-freedom (DOF) mechatronic systems require more sophisticated control algorithms day by day, precision, speed and concurrency concepts gain more importance. This dictates the processing units of the controllers to have more capabilities than before.

Increasing the capabilities of the processors also leverages the amount of processing power per dollar. But having a more intelligent system with better control does not always satisfy the customers' demands. To reduce the costs, factories are forced to have systems that can be quickly adapted to different environments.

In comparison with application specific integrated circuits (ASIC), field programmable gate arrays (FPGA) bring more flexibility to the design with reduced production cost and lower implementation time. On the other hand FPGAs can not reach the performance and power rating of an ASIC, but the differences are becoming negligible at least for motion control purposes.

However, the microprocessors and DSPs are highly flexible as well. Also the implementation time may be even lower than FPGAs' due to easier programming. But the microprocessors and DSPs can not work fully parallel as their nature. This may create lower loop frequencies for multi DOF systems.

Apart from the performance advantages like preventing the pipeline stages in processors, FPGAs also have better classifications in terms of power consumption. One of today's high-end FPGAs, Xilinx Virtex5, claims around 3 Watts of power consumption, whereas Intel Core Extreme requires 60-70 Watts of power.

There are several different solutions developed with FPGAs commercially and in research. As Altera points out in [1], FPGAs enable reconfiguring the hardware even if it is deployed on the field, which can be used for implementing different standards for industrial ethernet. Also in home automation systems, the high operation frequency of FPGAs decreases total-harmonic distortions in the signal as well as the audible noise and power consumptions, which makes the system more reliable. Xilinx's application notes [2] and National Instruments' setup [3] also provides different solutions to the motion control problem.

Several robotic applications have been implemented on an FPGA based system experimenting different motion control algorithms [4][5][6]. The experiments presented in [7] shows the effect of sampling frequency on the control of robotic manipulators.

Apart from specific application oriented platforms , there are also projects realized to offer motion control platforms. Platforms presented in [8] and [9] are both capable of controlling multi DOF mechanisms completely.

This paper introduces a new modular platform for motion control purposes. In order to have a thorough platform, all the major parts of an industrial motion control platform is implemented, namely the control algorithm, physical interface, actuator driving hardware, user interface and the reference generation.

In section II, the IP cores for physical interfacing and control algorithm will be introduced with synthesis results. The external hardware designs will be shown in section III. The user interface and communication parts will be defined in

section IV, and the software library for reference calculation and kinematics will be described in section V. The experimental results will be shown in section VI. Finally, the paper will be concluded in section VI.

II. IP CORE LIBRARY

The IP cores are coded in Verilog HDL in a modular manner which can be used to build more complex modules, hence increases the re-usability. The modules can be classified in five different subsections; floating point arithmetic, Laplacian, temporal, advanced and physical interface modules. The subsections consist of modules in an ascending manner in complexity.

A. Floating Point Arithmetic Functions

For increasing the range and precision of the calculations, floating point number format is chosen. Instead of implementing the IEEE 754 single precision standard, a different floating point format has been chosen for simplicity. For disambiguation reasons, the chosen format will be referred as "Easy Floating Point" (EFP) format. In order to provide compatibility between formats, converters have also been implemented. The main differences between both formats are the implied bit and the exponent bias.

Apart from the floating point converters, two blocks for converting from and to integer format have been implemented as well. These arithmetic and conversion blocks constitute the fundamentals for implementing motion control algorithms.

The adder/subtractor block checks the sign bits of the op erands to determine the operation. The difference between the exponents results the amount of shift right operations for the mantissa. After the corresponding operation is done, the resulting number is normalized according to the EFP format.

The multiplier block adds the exponents, and uses four hardware multipliers to multiply the mantissas. The resulting mantissa is normalized depending on its MSB.

The division block is implemented as a finite state machine (FSM) with 23 states. The logical block diagram is shown in Fig. 1. The mantissas of the operands are parsed and written to dividend and divisor registers if they pass from the normalization checks. The divisor is compared to the dividend, and if it is smaller, logic-1 is written to the corresponding bit of the result register. In this case, the dividend for the next state is gathered from the subtraction of current dividend and divisor. Before the passing to the next state, the divisor is shifted right one bit.

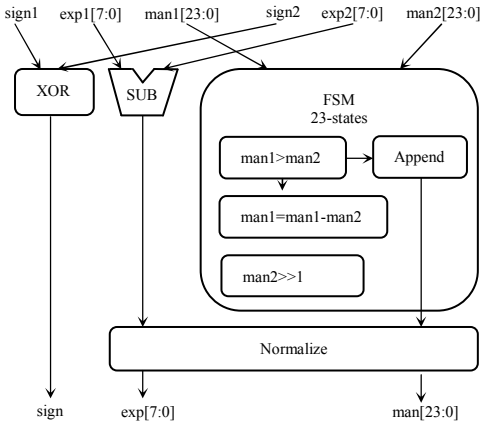

Fig. 1. The structure of floating point division unit.

B. Laplacian Functions

The mathematical models of physical structures are continuous functions or Laplacian functions. In order to use and process these functions in a digital medium, we need to find or write the digital equivalents of these functions. Before going deeper into the implemented blocks, it would be more appropriate to compare different types of approximations.

As studies show, newly introduced methods like, Al-Alaoui or implicit Adams provide superior performance than the classical methods as they are using different weighted interpolations of the Euler and Tustin methods [10]. Implicit Adams have been chosen as the Laplace approximation method; as it provides smoother response than Euler or Tustin, and easier to calculate than Al-Alaoui method.

The blocks in this subsection are implemented by using the aforementioned floating point arithmetic units. In order to consume little resources as possible, the blocks are implemented in a FSM manner. The implemented functions are derivative (s), integral (1/s), low pass filter (g/s+g) and real derivative (sg/s+g).

The derivative and integral functions consists of a 5 state FSM, which uses one EFP adder/subtractor and an EFP multiplier as resource. The low pass filter and real derivative blocks are implemented in a 34 state FSM, which also uses an EFP division block in addition to the previous blocks. The block structure of real derivative is shown in Fig. 2.

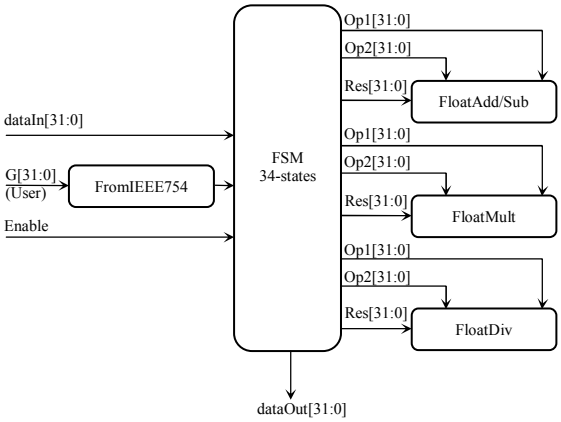

Fig. 2. The structure of real derivative block.

C. Temporal Functions

As the data is processed through time, algorithms may use the previous samples. In order to manipulate the timing of data, we need the variable delay block, which delays the input data, regardless of the number format, to its output with a given amount of cycles.

The other block is the synchronizer block, which works like a secondary clock source, enabling other blocks to input data at the same time as its name implies. This block can also be used to set the loop frequency of the overall algorithm.

D. Advanced Functions

As it has been mentioned before, in order to satisfy the concept of modularity, these advanced blocks consist of combinations of previously introduced basic blocks. The frequent use of these blocks makes them compulsory to implement. Due to the optimizations done in the formula, these combined blocks are better in performance and area compared to the cascaded implementations.

The Proportional-Integral-Derivative (PID) control is the most popular method in control theory. The chosen method uses separated gains in order to ease the trial-error based parameter tuning and the equation is given in (1).

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \quad (1)$$

And the discretized version of the formula is given in (2).

$$u[k] = K_p e[k] + K_d \frac{2g(e[k]-e[k-1])+(2+g)\dot{e}[k-1]}{2+3gT} + K_i \frac{3T\dot{e}[k]-\dot{e}[k-1]+2e[k-1]}{2} \quad (2)$$

The PID block is implemented with a 35-state FSM, which inputs the K_p, K_i and K_d gains from the user and the error signal from the system. All the calculations in the block are done in EFP format, since the block uses the one of each EFP arithmetic units as a resource.

One other implemented advanced block is disturbance observer which was introduced in [11]. The disturbance observer block is used to estimate the external disturbances acting on the system and feeds it back in order to eliminate the existing disturbance from the system, thus increasing the robustness of the system in noisy environments.

The constants of the actuator, like the inertia and the torque, and the low pass filter cut off frequency are given by the user to the system. The block is implemented in 63-states, and uses one of each EFP arithmetic units as resource.

As the last and the most advanced block, a motion controller block is implemented. The block consists of a PID controller, a disturbance observer, and a real derivative which is used for estimating the velocity from the position input. The required parameters and the constants for the system are gained from the user. The block is implemented with 72-states, in order to ease the control part of a motion control application. The structure of the block is shown in Fig. 3.

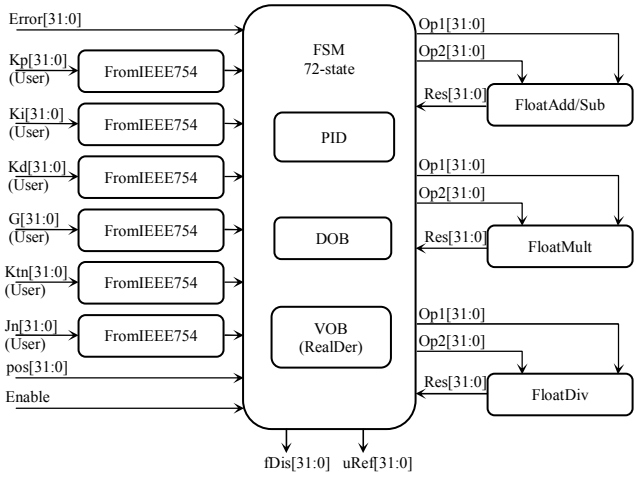

Fig. 3. The structure of motion controller block.

E. Physical Interface Blocks

Physical interface blocks are the bridge between the external hardwares and the control modules, which also makes some of them hardware specific modules.

The quadruple encoder module takes conventional A and B encoder signals, which differ 90 degrees between themselves, and takes the inverse of both signals in order to double the precision of the encoder signal. The block seeks specific signal state transitions in order to increment or decrement the counter value. The user can set the pulse-to-position ratio which can be used to receive the position of the actuator in metric scale like millimeters, degrees or radians. The module also has limit and set value parameters which can be used in hardware or software for homing purposes.

Pulse width modulation (PWM) is a method used for outputting analog values by digital signals. The PWM signal can be used to source drivers' reference inputs. Two different versions of this block are implemented in order to cover most of commercial drivers which works with PWM signals. The only difference between the modules is the direction signal being extracted as an external signal or not. The frequency and the precision of the PWM signal are linearly dependent

and can be adjusted according to the driver circuitry by manipulating the parameter in the hardware code.

These two are the fundamental modules for physical interfacing, hence another module is implemented which combines the two, namely actuator interface module.

The amount of FPGA resource accommodation of each block implemented in the library is given in Table 1.

TABLE I
FPGA UTILIZATION OF THE IMPLEMENTED BLOCKS

Name	# of Slices	# of FF's	# of LUT's	# of Mult	Speed
XC2VP30	13696	27392	27392	136	100MHz
FromInteger	165	-	298	-	Comb.
ToInteger	117	-	215	-	Comb.
FromIEEE754	5	-	9	-	Comb.
ToIEEE754	4	-	8	-	Comb.
FloatAddSub	272	-	493	-	Comb.
FloatMult	50	-	92	4	Comb.
FloatDividor	184	151	339	-	23-states
Derivative	545	241	1057	4	5-states
Low Pass Filter	892	606	1703	4	34-states
Real Derivative	894	639	1688	4	34-states
Integral	516	209	1000	4	5-states
Variable Delay	51	81	34	-	Seq.
Synchronizer	22	33	42	-	Seq.
PID controller	1199	894	2200	4	35-states
Dist. Observer	1040	676	1988	4	63-states
M. Controller	1599	1046	3044	4	72-states
QuadEncoder	244	36	444	4	Comb.
PWMBias	44	31	78	-	Seq.
PWMDirection	33	12	60	-	Seq.
ActuatorInterface	292	71	538	4	Seq.

III. EXTERNAL HARDWARE DESIGNS

Most of the commercial FPGA boards do not include power stages, which are required for driving actuators. There are various types of actuators with different ways to drive. Commercial drivers are referenced by PWM or analog signal. The PWM signal is already present in the system; therefore a PWM-to-analog converter has been designed in order for the system to be compatible with any kind of commercial actuator. The input stage consists of two cascaded RC low-pass filters, which is followed by an offset and an amplification stage. The hardware's output range is ±10V, 0V being the output of a PWM signal with %50 duty cycle. There is also an external direction signal included in the hardware which may be required for some drivers. The encoder interface on the hardware is standardized with the designed adapters for various types of encoder connectors. The schematic of the design is shown in Fig. 3.

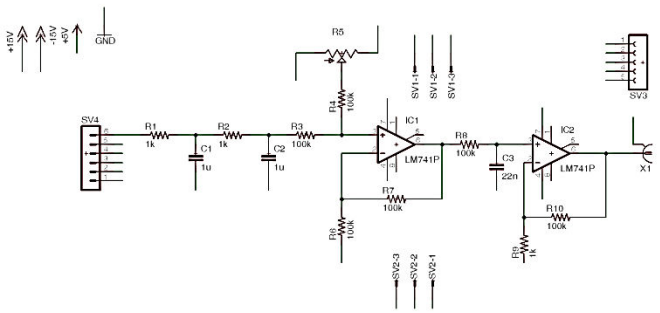

Fig. 3. Schematic of PWM to analog converter

Two different boards, for different DC motor sizes, are designed. The one for low power DC motors is based on L298 chip, which limits the circuit up to 46V and 2Amps. The high power board is based on HIP4082 gate driver IC; therefore the power rating of this board is dependent on the driven MOSFET's.

For driving solenoid based drivers, Darlington array based drivers are designed and can be activated by the logic level output pins of the FPGA.

In order to read industrial switches like Hall Effect, a basic two cascaded resistor voltage divider is sufficient.

Lastly, an extension board is designed, which utilizes all of the user GPIO pins present on the FPGA board. The extension board has 10 input stages for industrial switches, 10 Darlington array based output stages, and 10 sockets for PAC circuits. This extension board is specific for XUPV2P board.

IV. COMMUNICATION

Communication can be separated into two parts; hardware and software. Since a commercial FPGA board is used, the onboard communication hardwares are used. RS232 is used for low-speed and ethernet is used for high-speed communication.

The onboard MAX3388 chip converts the logic level signals to RS232 level. The serial communication is very easy to use, as it is set as the standard input output communication medium in the software; therefore it is preferred for debugging purposes.

The LXT972A chip installed onboard supports IEEE802.3 standard for 10/100 Mbit/sec communication. The chip provides standard Media Independent Interface (MII) for easy attachment to 10/100 Media Access Controllers (MAC). The full-duplex operation mode can be selected between auto-negotiation, parallel detection or manual control. The physical layer transceiver (PHY) also provides all functions of the physical coding sub-layer (PCS), the physical media attachment (PMA) sub-layer, and the physical media dependent (PMD) sub-layer for 100 Mb/s connections. The board also includes a Dallas Semiconductor DS2401 Silicon Serial Number. This provides a unique identity for each board. The chip includes a 64-bit ROM with a 48-bit serial number, an 8-bit CRC, and an 8-bit device family code. The label on the board has the registered Ethernet MAC address.

V. SOFTWARE LIBRARY

The overall system does not only consist of hardware part, but also some elements which have been implemented in software. The written software runs on the embedded IBM PowerPC 405 processor inside the FPGA. The PowerPC exists as hardware inside the FPGA; therefore it does not consume any of the resources present in FPGA, but it creates convenience for some parts of the implementation.

Aforementioned communication mediums are handled in software. While the software for RS232 communication is pretty straightforward, Ethernet communication is a bit more challenging. For Ethernet communication, an external open source library named Light Weight IP [12] is implemented. Using the features provided by the library, a communication grammar has been built upon. The library intends to ease the usage of the system for accessing the hardware modules.

A graphical user interface (GUI) has been developed in C# for Ethernet communication. Via this GUI a user can command and monitor the system for specific data.

The inverse kinematics of the mechanism which will be controlled is implemented in software also. The main reason for implementing the kinematics in the software is, the overhead brought by implementing all the functions in hardware. Kinematics consists of several different mathematical functions like trigonometric, logarithmic, exponential and polynomial functions. Some of the basic functions can be approximated with different methods like Taylor expansion, but logarithmic and exponential functions are more complicated to approximate.

Each function in the software has been written in a library form. This enables the future users to understand the code easier and minimizing the amount of code changes for different implementations. All the codes have been written in C, and built by using Xilinx Embedded Development Kit tool.

VI. EXPERIMENTS

From a wider perspective, motion control systems consist of theory and functionality. In order to test both, two experiments were made for each perspective.

In order to prove the robustness that the system gains by using a disturbance observer, the motion controller module has been compared to the PID module. PID control may need very fine parameter tuning in several cases. The trial-and-error period consumes a lot of time. There have been proposed methods for parameter tuning, like Zeigler-Nichols, but unfortunately it only provides shallow tuning without considering the system dynamics. The error that the controller fails to handle, like overshoot or steady-state error, are treated as disturbance and compensated by the observer. In order to test and validate these blocks a simple setup has been constructed. The setup consists of Maxon motors with graphite brushes. As gears damp the actuators and increases stability, in order to observe the differences explicitly, gearless motors have been chosen. Two motors are connected by a belt from their shafts, where one motor is controlled, and the other is supplied with a constant current in order to create disturbance. The step response of the system with PID and motion controllers with external disturbances can be found in Fig. 4 and Fig. 5.

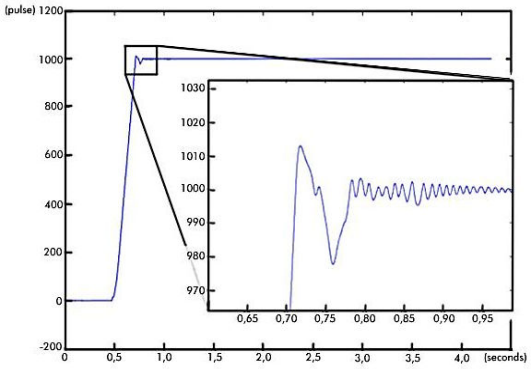

Fig. 4. *Step response of PID controller with external disturbance*

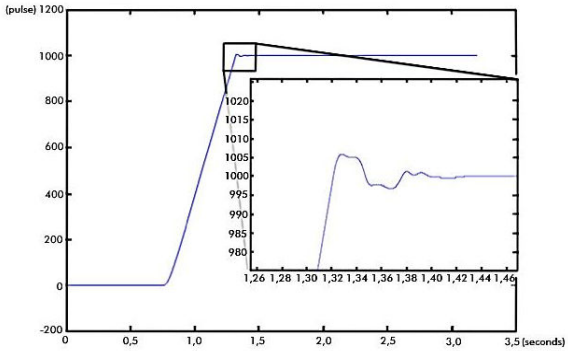

Fig. 5. *Step response of motion controller with external disturbance.*

The oscillations in Fig. 4 are caused by the lack of tension in the belt. As it can be seen clearly, the observer enhances the controller's ability to compensate the external disturbance, as it is in this case the elasticity of the belt. Given a step reference of 1000 pulses, PID controller results 15 pulse overshoot, and the system oscillates in steady state with amplitude of ±2 pulses, while the motion controller results 6 pulses overshoot with no steady-state error.

While this experiment proves the functionality of the IP cores, in order to measure the precision of the overall system a pantograph mechanism has been chosen. Pantograph was first developed to enlarge or shrink the mimicking motion between each end. The first design had two end effectors. The design was manipulated over the years and converges to the mechanism that we know as pantograph and started to be used in many different application fields. Today it is one of the most popular planar parallel mechanisms.

The pantograph works in the XY Cartesian plane, transforming the rotation of the actuators to linear motion; therefore the trajectory of the end effecter would be given in XY coordinates, by the user or generated trajectory. This creates the necessity for the derivation of the inverse kinematics of the mechanism. Using the notations in the configuration space, the forward and inverse kinematics of the mechanism is derived from [13].

In the software, pantograph is defined as a structure in order to change the constants, like link lengths, easily. This also makes the software structure modular and scalable. The GUI for the pantograph shows the end effecter positions. The parameters for the motion controller hardware block are also tuned via GUI. A motion controller block is used for each actuator. The synchronizer block is set to interrupt each motion controller block in every hundred cycles, which determines the frequency of the control loop as 1MHz. The limit switches for the XY actuators are used for calibration and homing purposes. After the limit switch has been reached, the positions are set to zero, and ±60 degrees have been given as initial reference for the actuators. The actuators used in the mechanism are Faulhaber brushless DC motors, with zero-backlash gearboxes (ratio 76:1) and embedded incremental encoder. Mid-Power H-Bridge circuit is used to drive the actuators. Link lengths of the utilized pantograph are 40mm, and the distance between the actuators are 30mm.

A square is given as a sample trajectory with 1cm edge dimension. The trajectory versus position is shown in Fig. 6;

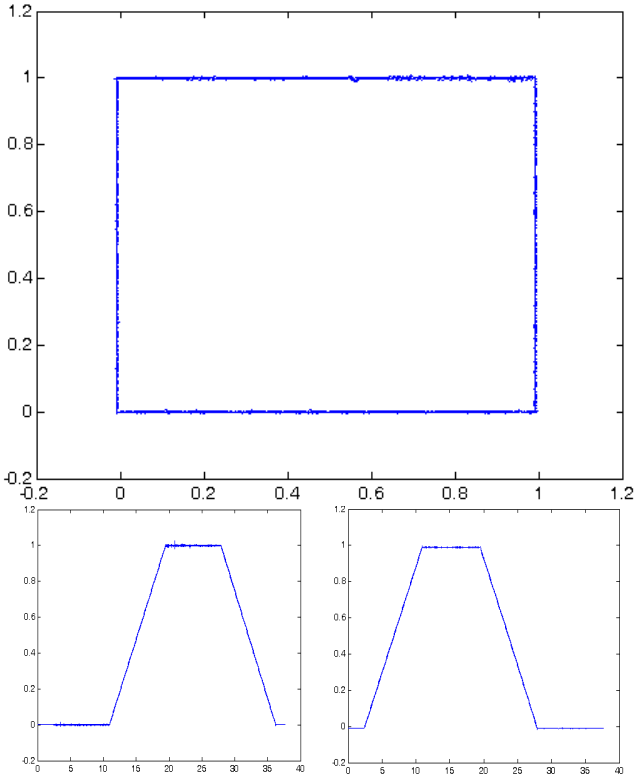

Fig. 6.*Tracking results for a square trajectory.*

The experimental setup accommodates 68% of the FPGA resources, consuming 3 Watts excluding the supply for the actuators. The error of the mechanism is measured to be 40 microns in average, and 70 microns maximum. The overall setup is shown in Fig 7.

VII. CONCLUSION

The main aim of this paper was to propose an FPGA based motion control platform which offers an easy-to-build system, due to the given hardware and software libraries. The libraries

Fig. 7.*Overview of the system.*

also allow creating custom functionalities without advanced hardware description language knowledge. The experiments are setup by using both hardware and software libraries, thus the results validate the platforms functionality. The functionalities are designed to cover the fundamentals of motion control theory.

The performance comparison of the proposed system can be seen more clearly as the implemented applications require high sampling rates, and high loop frequencies due to the physical hardware structure of the system and parallel processing capabilities. The energy efficiency of the system is higher than other similar platforms, as the system does not supply any redundant parts, and works at a lower clock frequency. The hardware libraries can be synthesized and compiled for various FPGA's with different technologies, thus they are technology independent.

ACKNOWLEDGEMENTS

This work was supported by the Ministry of Industry of Turkey, under SanTez project 00183.STZ.2007-2.

REFERENCES

[1]Altera, *Is Motion control technology moving from controllers to FPGA's?*, Embedded Control Europe Magazine, pp. 24-26, Octorber, 2008
[2]National Instruments, *Creating Custom Motion Control and Drive electronics with FPGA based system*, 2010
[3] Xilinx, *FPGA Motor Control Reference Design*, 2005
[4]J. S. Kim, *Hardware Implementation of Nonlinear PID Controller with FPGA Based on Floating Point for 6-DOF Manipulator Robot Arm*, International Conference on Control, Automation and Systems, Oct. 2007
[5] J. S. Kim, *Hardware Implementation of a Neural Network Controller on FPGA for a Humanoid Robot Arm*, Proc. IEEE/ASMEInternational Conference on Advanced Intelligent Mechatronics, July 2008
[6] B. S. Kariyappa, *FPGA Based Speed Control of AC Servomotor Using Sinusoidal PWM*, IJCSNS International Journal of Computer Science and Network Security, vol.8 no.10, October 2008
[7] E. Ishii, *Improvement of Performance in Bilateral Teleoperation by Using FPGA*, AMC, 2006
[8] X. Shao, Development *of an FPGA-Based Motion Control ASIC for Robotic Manipulators*, Proc. 6th World Congress on Intelligent Control and Automation, June 2006
[9]J. U. Cho, *An FPGA-Based Multiple-Axis Motion Control Chip*, IEEE Transactions on Industrial Electronics, vol. 56, no. 3, March 2009
[10] R. S. Barbosa, *Time domain design of fractional differintegrators using least-squares*, Signal Processing 86, pp. 2567–2581, 2006
[11] I.Godler, H.Honda and K.Ohnishi, *Design Guidelines for Disturbance observer's Filter in Discrete Time*, AMC2002, The 7th IEEE International Workshop on Advanced Motion Control, pp. 390-395, 2002.
[12] Xilinx Inc., *LightWeight IP (lwIP) Application Examples*, June 2009
[13] G. Campion, *The Pantograph Mk-II: A Haptic Instrument*, Proc. IROS 2005, IEEE/RSJ Int. Conf. Intelligent Robots and Systems, pp. 723-728