Implementation of FPGA Based PSO PID Controller for Feedback IVAX SCARA Robot Manipulator

Firas Abdullah Thweny, Ahmed Ali Alhammad

Computer Engineering Department,
Al-Nahrain University
Baghdad, Iraq

Abstract **- In this paper a complete control system is proposed to control the four Degree of Freedom (DOF) Feedback Instruments IVAX SCARA robotic manipulator using Xilinx Field Programmable Gate Array (FPGA) Spartan 3AN development kit. Proportional, Integral and Derivative (PID) controllers are used to control each joint of the manipulator position. Particle Swarm Optimization algorithm (PSO) is used to tune the PID controllers' parameters. Trajectory is obtained from sequence of set point uploaded to the FPGA from a user interface. PID controllers, PSO algorithm, Inverse kinematics calculations processing are implemented on the FPGA. The system was treated as multiple input multiple output system (MIMO) where each joint works independently from the others. It was clearly noticed that PSO has several attractive features that make it an excellent candidate for the tuning of PID controllers, like fast convergence and simple computation.**

Keywords **- PID, PSO, FPGA, IVAX SCARA, Robotic manipulator.**

I. Introduction

Industrial robots are beginning now to revolutionize industry. These robots do not look or behave like human beings, but they do the work of humans. Robots are particularly useful in a wide variety of industrial applications such as material handling, painting, welding and assembly [1].

A. Introduction to robot manipulator

Robot manipulator mechanical system consists of rigid bodies (links) interconnected by means of articulations (joints) either serially or parallel. Joints derived by mechanical power driver, sensors are used to monitor the status of the joints, controller unit is used to instruct the robot and control its movements, power source and user interface are also parts of the robot manipulator system. Software programs used in the controller and in the user interface unit are considered as parts from the robot manipulator system because the manner in which the robot is programmed and controlled have an impact on the robot performance and applications.

A manipulator is characterized by an arm that ensures mobility, a wrist that confers dexterity, and an end-effector that performs the task required of the robot [2].

Industrial robots can be programmed to perform a wide variety of application and that needs a controller to provide stable movement to the desired set point and a sequencer to monitor robot's status on a programmed trajectory. User interface is required to reprogram the robot to the required new task.

With the aid of the feedback from joints' sensors, position of each joint can be controlled to assure that each joint position finally reaches its desired set point and thus controlling the end effector position and orientation.

B. Feedback Instruments IVAX SCARA robot

The SCARA robot undertaken in this paper is IVAX SCARA robot manipulator manufactured by Feedback Instruments (shown in Fig. 1). The actuation of the four joints is provided through four independent dc servo motors driving a planetary gearbox and a worm gear to move each joint. In addition to the described drive system, the vertical axis uses a rack and pinion setup to produce linear motion. The end effector of the arm is a pneumatic gripper controlled by an electronic solenoid valve [3].

Robot work space is an arc of radius between 280 mm and 108 mm, Sector angle of 270° and vertical axis of 40mm as shown in Fig. 2 [3].

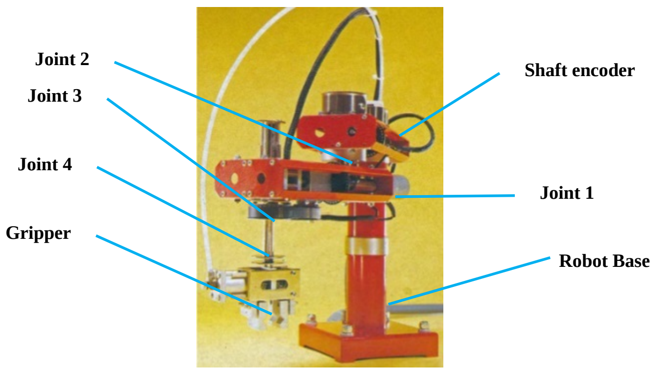

Fig. 1 Feedback Instruments IVAX SCARA robot

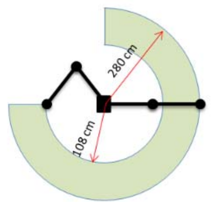

Fig. 2 Feedback Instruments IVAX SCARA robot work area

C. Introduction to PID and PSO

PID controllers are used to get each joint to its desired set point independent from the others. In this approach the manipulator is considered as a composed of *n* independent system (n joint drivers). Each system is controlled as single input single output (SISO) system disturbance may come from other joints movements or from gravity and others [4]. Figure 3 shows a typical DC motor control system used in this paper.

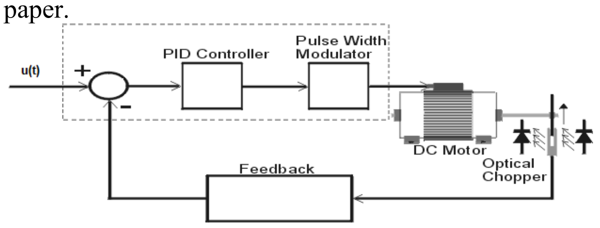

Fig. 3 DC motor control system

When all joints reach their desired set points, robot can move to the next end effector set point from the trajectory map. In this paper trajectory is performed by a series of end effector's set points uploaded to the robot system. System can work with a step by step instructions affecting single or multiple joints simultaneously.

PID controllers are widely used to control the movement of robot joints exceeding 95% of control methods for industrial due to its simplicity [5]. PID consists of three terms Proportional, Integral and Derivative. Each term has a control parameter. Tuning these parameters affect the performance of the PID controller. Thus tuning of these parameters is an important task. PSO algorithm was used to tune these parameters and the user is allowed to retune the PID parameters in order to calibrate robot performance.

D. FPGA controller

In this paper, FPGA was found to be suitable to combine user interface, trajectory sequencing, kinematics and inverse kinematics calculations, PID controllers and PSO algorithm calculation in a single controller utilizing its parallel processing nature and its reprogramability [5].

FPGAs are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) which are connected via programmable interconnects (Fig. 4 shows a typical logical view of CLB). A CLB uses a number of lockup tables (LUTs) to implement logic, each LUT can be programmed to implement any Boolean expression up to the number of inputs and outputs it has. For example, a 4x1 LUT has 4 inputs and 1 output. The interconnect uses programmable connection blocks and switch blocks to route or steer signals and connect the CLBs together and to the outside world through the I/O pads [5]. Very High Speed Integrated Circuits (VHSIC) Hardware Description Language (VHDL) was used to model the controller system into the FPGA as firmware of the controller.

FPGA controller consists of four subsystems: user interface processing unit, main sequencer processing unit, robot interface processing unit and robot monitoring system.

The user interface processing unit is used to interpret commands and data between the user interface PC and the main sequencer processing unit via RS232 serial port. Main sequencer processing unit interacts with PSO optimization unit, data storage unit, inverse kinematics unit, Liquid Crystal Display (LCD) screen interface unit and robot interface processing unit. The implemented FPGA control system diagram is shown in Fig. 5.

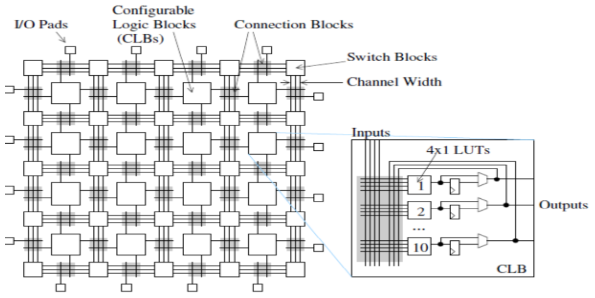

Fig. 4 Typical logical view of CLB

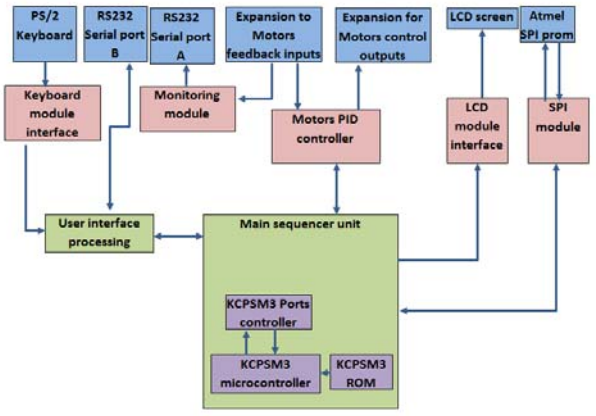

Fig. 5 Implemented FPGA control system diagram

E. Literature survey

P. Xu and D. Wai [6] presented a graphical user interface and kinematics calculation for IVAX robotic manipulator using IVAX controller from Feedback including its three Z80 processors and MIC 926 parallel interface on Visual basic as a replacement of the system program which is written by BASICA old language.

N. Ravari et al. [7] introduced a hybrid Fuzzy-PID controller based on learning automatic, the goal was the optimal tracking of robot systems including motor dynamics. In the proposed controller, the learning automatic is used at the supervisory level for adjustment of the parameters of hybrid Fuzzy-PID controller during the system operation. The proposed controller was tested using simulation on PUMA560 manipulator which gave satisfactory results.

S. Sonoli et al. [8] used VHDL for Xilinx FPGA (XC3S400) based PID controller for DC motor speed. The tools used for building and testing the software modules are Xilinx ISE 9.2i and ModelSim XE III 6.3c. Before verifying the design on FPGA, the complete design is simulated using Modelsim simulation tool.

O. Inwelegbu et al. [9] used DE2 FPGA development board from Altera to provide PID based DC motor torque controller for dough mixing machine.

In this paper, PSO algorithm, which was introduced by Kennedy and Eberhart [10], is used to optimize the coefficients of the PID controllers of the IVAX manipulator joints to overcome the main shortcoming of the PID controller and the lack of efficient tuning method to tune the controller parameters.

II. PID CONTROLLER

PID controller has a long history in automation control starting from the last century as a three term controller. Thus it was has been demonstrated to be effective for DC servo motor position control. The PID controller is used to reduce or eliminate the steady-state error between the measured motor position and the reference position to be reached**.**

Conventional feedback control systems has a basic structure as shown in Fig. 6, where *P* is the process or plant to be controlled, *C* is the controller (PID controller in this case), *F* is feed forward filter, *r* is the reference signal (set point), *e* is the error between the output signal *y* (the controlled signal) and the reference signal affected by sensor noise *n, u* is the control signal which is by nature merged with load disturbance signal ***d*.**

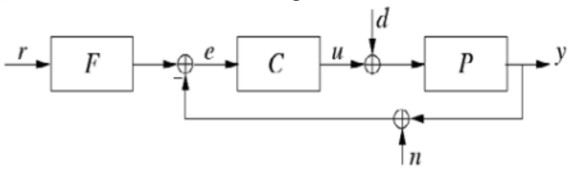

Fig. 6 Conventional feedback system

The PID controller calculates the error between the reference signal and the output signal and provides the control signal due to this error. As PID controller has three components and three parameters proportional, integral and derivative, these as in the mathematical formula for calculating continuous time controller in Eq. (1).

$$u = K_p \left(e + K_i \int_0^t e \, d\tau + K_d \frac{de}{d\tau} \right) \tag{1}$$

Discrete time implementation of Eq. (1) which is used in the implemented FPGA controller as a digital controller can be derived into three modules as in Eqs. (2), (3) and (4). The final control signal is the sum of outputs of the three modules as in Eq. (5) [11].

$$u_p(k) = K_p(e(k)) \tag{2}$$

$$u_i(k) = u_i(k-1) + K_i(e(k))\Delta\tau \tag{3}$$

$$u_d(k) = K_d \frac{e(k) - e(k-1)}{\Delta\tau} \tag{4}$$

$$u(k) = u_p(k) + u_i(k) + u_d(k) \tag{5}$$

Where *k* is the current sample, $u_p(k)$, $u_i(k)$ *and* $u_d(k)$ are the output of proportional, integral and derivative modules respectively, τ is the sample time and *e(k)* is the error at the k^{th} sample.

III. PID PARAMETERS TUNING

Parameter tuning is the optimization process that involves some performance criterion related to the controller form response and the error between the set point and the system output [12].

Tuning PID parameters will tune the performance of the controller, thus it should be done based on a required performance which is usually related to set point following or system robustness or some time both are used. The trade-off here is between accuracy and system complexity.

Tuning methods can be classified into three classes: classical tuning, evolutionary algorithms search tuning and adaptive tuning.

Classical tuning methods do some process tests on a specific strategy on the system once and calculate the PID parameters from it. In the 1940s, Ziegler and Nichols developed two methods for controller tuning based on simple characterization of process dynamics in the time and frequency domains. The tests are based on open loop step response measurements by applying unit step signal on the system. The rules were simple to use and gave initial conditions for manual tuning. The ideas were adopted by manufacturers of controllers for routine use. The Ziegler–Nichols tuning rules unfortunately have two severe drawbacks: too little process information is used, and the closed loop systems that are obtained lack robustness. Unfortunately such classical tuning method can be applied only on linear systems [13].

Evolutionary algorithms search methods such as Genetic Algorithm (GA), PSO and ant colony optimization are stochastic search methods that mimic the metaphor of natural biological evolution and/or the social behaviour of species. Examples include how ants find the shortest route to a source of food and how birds find their destination during migration. The behaviour of such species is guided by learning, adaptation, and evolution [14].

In general, Evolutionary Algorithms (EAs) share a common approach for their application to a given problem. The problem first requires some representation to suit each method. Then, the evolutionary search algorithm is applied iteratively to arrive at a near-optimum solution.

EAs can be deployed on nonlinear system due to its search nature, but it cannot be used on adaptive systems that change their behaviour by time.

Adaptive Control covers a set of techniques which provide a systematic approach for automatic adjustment of controllers in real time, in order to achieve or to maintain a desired level of control system performance when the parameters of the plant dynamic model are unknown and/or change in time. The tuning of the controller will be done in real time from data collected in real time on the system. Since the system in this paper is not adaptive nor changes in time, adaptive control will not be discussed further.

IV. IMPLEMENTED PSO PID OPTIMIZATION

Kennedy and Eberhart developed PSO inspired by the social behaviour of a flock of migrating birds trying to reach an unknown destination. In PSO, each solution is a 'bird' in the flock and is referred to as a 'particle', the particle in the population evolve their social behaviour and accordingly their movement towards a destination.

Fig. 7 shows the proposed PSO algorithm for PID optimization used in this paper. Where random solutions are initialized as particles in the swarm community, the three dimension space used in this paper as community space is the PID controller parameters K_p, K_i *and* K_d. Each particle in the swarm monitors its best position, swarm's best particle position and itself velocity. Determining how best particle's position is based on some fitness criteria which result on a specific value that can be tested to check whether it is better than the others or not. After every particle in the swarm gets its fitness on the system, each particle sets its local best fitness and the best particle's fitness is sets as the global best particle.

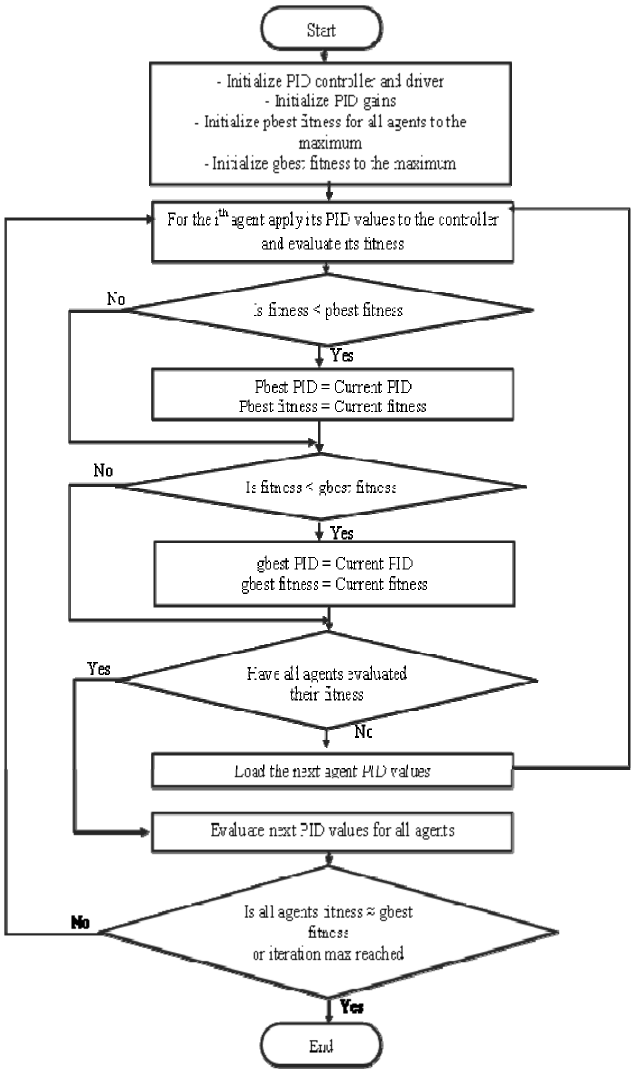
Start
- Initialize PID controller and driver
- Initialize PID gains
- Initialize pbest fitness for all agents to the maximum
- Initialize gbest fitness to the maximum
For the i^th agent apply its PID values to the controller and evaluate its fitness
No
Is fitness < pbest fitness
Yes
Pbest PID = Current PID
Pbest fitness = Current fitness
No
Is fitness < gbest fitness
Yes
gbest PID = Current PID
gbest fitness = Current fitness
Yes
Have all agents evaluated their fitness
No
Load the next agent PID values
Evaluate next PID values for all agents
No
Is all agents fitness ≈ gbest fitness or iteration max reached
Yes
End

Fig. 7 Proposed PSO algorithm for PID optimization

Based on Eqs. (6), (7) and (8), each particle updates its velocity and position then rechecks its fitness on the system and updating local best and global best values.

The algorithm stops if all agents reach almost the same solution or system iterations exceeds the maximum iteration set.

$$v_i^{k+1} = \chi[v_i^k + c_1 rand_1 \times (pbest_i - s_i^k) + c_2 rand_2 \times (gbest - s_i^k)] \qquad (6)$$

$$s_i^{k+1} = s_i^k + v_i^{k+1} \qquad (7)$$

Where

$$\chi = \frac{2k}{\varphi - 2 + \sqrt{\varphi^2 - 4\varphi}}\ ,\ \varphi = c_1 + c_2 > 4\ , \qquad (8)$$

v^k_i and s^k_i are velocity and position of the i^{th} particle in the k^{th} iteration respectively , χ is the constriction factor, *rand* is a random number between 0 and 1, c_j is weighting coefficients, $pbest_i$ is the local best position of the i^{th} particle. *gbest* is the global best position in the swarm community.

Typical performance criterion have been used to describe the close loop system performance such as Integral Square Error (ISE) index, Integral of time multiplied by Squared Error (ITSE) index, Integral of Absolute Error (IAE) index, and Integral of Time multiplied by Absolute Error (ITAE) index. Each of them has its own characteristic performance.

For instance, the ISE index penalizes large errors heavily and small errors lightly. A system designed by this criterion tends to show a rapid decrease in a large initial error. Hence, the response is fast and oscillatory; leading to a system that has poor relative stability, while ITSE places little emphasis on initial errors and heavily penalizes errors occurring late in the transient response to a step input. Therefore a system optimized based on the IAE index penalizes the control error where as if designed using ITAE criterion produces a small overshoot and a well damped oscillation [15].

The above mentioned performance criteria can be calculated by means of the following equations:

$$ISE = \int_0^T e^2(t)\,dt \qquad (9)$$

$$IAE = \int_0^T |e(t)|\,dt \qquad (10)$$

$$ITAE = \int_0^T t|e(t)|\,dt \qquad (11)$$

$$ITSE = \int_0^T te^2(t)dt \qquad (12)$$

In this paper, PSO is configured to use ten particles in a maximum of 128 iterations in the swarm configured to tune all joints PID controllers simultaneously.

PSO particle's local best positions are saved into FPGA flash memory and the user can trigger PSO tuning process whenever it's required and the previous local best position will be the initial values for the system.

V. INVERSE KINEMATIC

Inverse Kinematics unit is used to convert the series of end effector's position and orientation set point obtained from the sequencing unit into each joint position value.

Inverse kinematics calculations for IVAX SCARA manipulator are based on Eqs. (13), (14), (15) and (16).

$$d = a_0 - Z \qquad \textit{(0<Z<40mm)} \qquad (13)$$

$$\theta_2 = atan2\,(M\,, \pm\sqrt{1 - M^2}) \qquad (14)$$

where $M = \left(\frac{X^2 + Y^2 - a_1{}^2 - a_2{}^2}{2a_1\,a_2}\right)$

$$\theta_1 = atan(Y^2, X^2) - atan2\,(a_1 + a_2 cos\,\theta_2, a_2 sin\,\theta_2) \qquad (15)$$

$$\theta_3 = atan(Y^2, X^2) - \theta_1 - \theta_2 \qquad (16)$$

where d is the distance of the prismatic joint, θ_1, θ_2 and θ_3 are the angles of the revolute joints 1,2 and 3 respectively, *X,Y and Z* represent the end effector desired position in the three dimensional space. a_0 , a_1 and a_2 are the robot links lengths which are 200 mm, 140 mm, 140 mm respectively.

VI. FPGA SYSTEM CONTROLLER

In this paper, Spartan 3AN development kit has been used to contain the control unit of the system. As shown in Fig. 5, the control system was modelled into several parts, some of them are interfaces to the hardware resources such as LCD screen, PS/2 keyboard, RS232 serial port and SPI flash. Others have specific task such as PID controllers, PSO module and kinematics calculation module. Each module can work independent from the others and all of them are instructed by the main sequencer unit that contain KPCMS3 Pico Blaze Xilinx micro controller which fetches its instructions program from the specific location in the SPI flash memory.

Several programs were installed in the SPI flash such as: PSO calibration, immediate end-effector movement and work demos. User can upload a User File Program (UFP) that contains the Hex code of KCPSM3 assembly language to perform a specific task by the robot manipulator.

User interface and monitoring panels allow the user to interact with the robot manipulator using graphical user interface or command line interface from attached terminal PC or using PS/2 keyboard and LCD screen available on Spartan 3AN development kit. Fig. 8 shows the proposed system design diagram.

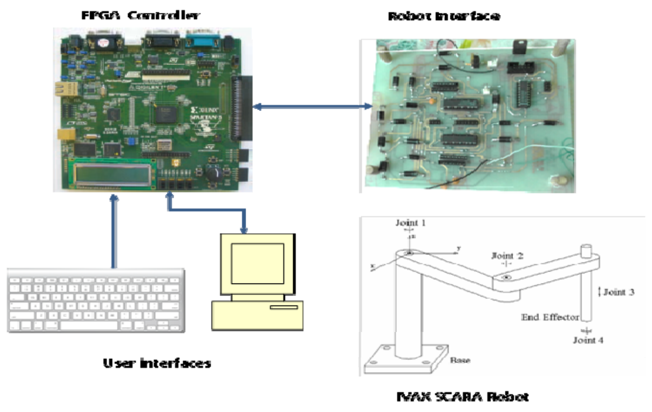

Fig. 8 Proposed system design diagram

VII. RESULTS

ISE performance criteria index has been used with PID PSO tuning for 10 particles in a maximum of 128 iterations configured to tune all joints' PID controllers simultaneously. Set point was configured to 200 step counted from the shaft encoder (i.e. 20 degree on the rotational joints and 20 mm in the prismatic joint).

Timing was found to be about 2-3 seconds to reach the desired set point. Additional 1.5 second is required to return the joint to zero location. Over all process took about 40 second for a single iteration and about 85 minutes for the 128 iterations.

For each PID control system, global best PID values were used in the system after PSO had reached its steady state.

Results for percentage error have been recorded for system iterations for the four joints as shown in figures. 9, 10, 11 and 12.

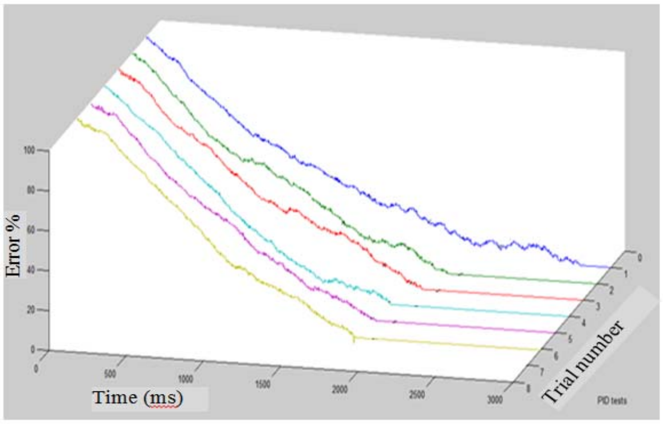

Fig. 9 Cascaded error curves for the best responses of the joint 1

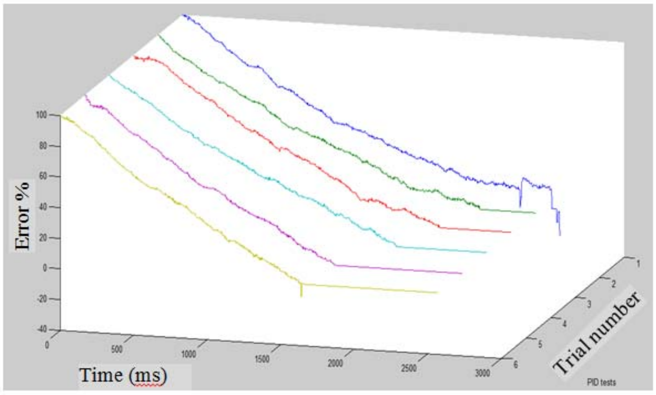

Fig. 10 Cascaded error curves for the best responses of the joint 2

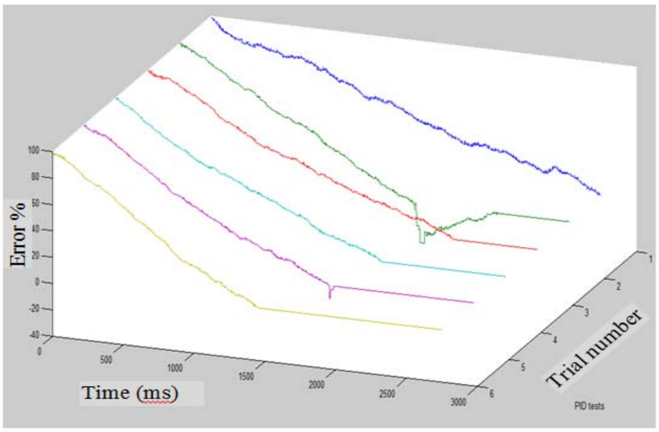

Fig. 8 Cascaded error curves for the best responses of the joint 3

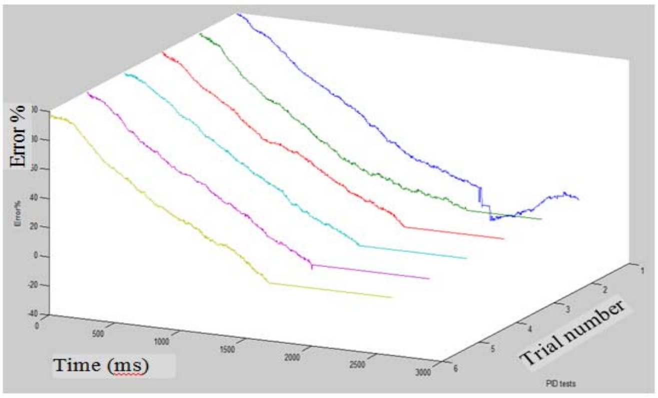

Fig. 9 Cascaded error curves for the best responses of the joint 4

Table 1 shows the best acquired PID parameters along with their transient response characteristics.

TABLE 1
BEST ACQUIRED PID PARAMETERS

Joint No.	K_p	K_i	K_d	**Rise time (sec.)**	**Settling time (sec.)**
1	10	1.00	0.060	1.7	1.9
2	11	1.50	0.125	1.4	1.5
3	9	0.50	0.250	1.3	1.3
4	9	1.25	0.250	1.4	1.4

VIII. CONCLUSIONS

PID was proved to be an effective control algorithm for complex nonlinear systems like robot manipulators, in addition to its simple implementation and computational efficiency.

The PSO has several attractive features that make it an excellent candidate for the tuning of PID controllers, like fast convergence and simple computation.

FPGA systems are very effective to be used as robot interface for its high speed of calculation, independent and simultaneous modules work and the ease of accessory components access like flash prom and RS232 included in the FPGA kit itself.

REFERENCES

[1] Y. Koren, *Robotics for engineers*, Technon Israel Institute of technology, McGraw-Hill, 1985.
[2] B. Siciliano, L. Sciavicco, L. Villani and G. Oriolo, Robotics: modeling, plannig and control, Springer, 2009
[3] Cover story, IVAX joins the arms, Practical Robotics Magazine, May 1985.
[4] http://www.xilinx.com/training/fpga/fpga-field-programmable-gate-array.htm
[5] K. J. Astrom and T. Hagglund, PID Controllers, Theory, Design and Tuning, 2nd Edition, Instrument Society of America, 1995.
[6] D. Wai and P. Xu, Graphical User Interface Development for PC Based Control of a SCARA Robotic Arm, Massey University, 2001.
[7] N. Ravari and H. Taghirad, A novel hybrid Fuzzy-PID controller for tracking control of robot manipulators, IEEE International Conference on Robotics and Biomimetics, Bangkok, 2008.
[8] S. Sonoli and K.Nagabhushan, Implementation of FPGA based PID Controller for DC Motor Speed Control System, WCECS , 2010.
[9] O. Inwelegbu and A. Nwodoh, FPGA controller design and simulation of a portable dough mixing machine, Nigerian Journal of Technology, 2011.
[10] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in IEEE Proceedings of the 4th International Conference on Neural Networks, 1995.
[11] J. Ledin, Embedded Control Systems in C/C++: An Introduction for Software Developers Using MATLAB, CMP Books, 2004.
[12] W. Y. Svrcek, D. P. Mahoney and B. R. Young, A Real-Time Approach to Process Control,2nd edition, John Wiley & Sons, 2006.
[13] K. J. A. ström and R. M. Murray, Feedback Systems: An Introduction for Scientists and Engineers, Princeton University Press, 2008.
[14] E. Elbeltag , T. Hegazy and D. Grierson, Comparison among five evolutionary-based optimization algorithms, Mansoura University, Advanced Engineering Informatics, 2005.
[15] A. Marzoughi, H. Selamat and F. Marzoughi, " Application of Particle Swarm Optimization Approach to Improve PID," in Proceedings of 2010 IEEE Student Conference on Research and Development, Putrajaya, 2010.