A New Motion Control Hardware Architecture with FPGA-Based IC Design for Robotic Manipulators

Xiaoyin Shao and Dong Sun
Department of Manufacturing Engineering and Engineering Management
City University of Hong Kong, Hong Kong
shao.xy@student.cityu.edu.hk, medsun@cityu.edu.hk

James K Mills
Department of Mechanical and Industrial Engineering,
University of Toronto
5 King’s College Rd.
Toronto, Canada M5S 3G8
mills@mie.utoronto.ca

Abstract— In this paper, a new motion control hardware architecture is proposed for improved motion performance of robotic manipulators during high speed motion. The main idea is to remove the servo control loop from the DSP (Digital Signal Processor) to a FPGA (Field Programmable Gate Array), and utilize the high speed hardwired logic of the FPGA to enhance the computation capability and relieve the computing load on the DSP. The control algorithm is partitioned into a linear portion and a nonlinear portion. The linear portion with position/velocity feedback represents the major control loop and is implemented in the FPGA. The nonlinear portion acts as dynamic compensation to the linear portion to perform complex modeling related calculations, and is implemented in the DSP. A new FPGA-based motion control IC is designed to realize this new control hardware structure. Experiments were conducted on a Yamaha robot manipulator to compare new control architecture and the existing one, when the same control algorithm was used. Experimental results demonstrate that the proposed new control architecture exhibits much improved motion performance especially during high-speed motions.

Index Terms— FPGA, motion control, IC design, robotic manipulator.

I. INTRODUCTION

Motion control system represents a major subsystem that is responsible for actuation of all sorts of devices in industrial automation such as robotics, wire bonding machines, CNC machining, and high speed assembly machines. In these tasks, higher speed with the same error performance means higher productivity. Over the past years, there have been proposed many different control algorithms. Most of these algorithms can be considered as special cases of the class of computed torque controllers [1]. Computed torque control is a special application of feedback linearization of nonlinear systems, which has gained popularity in modern systems. One way to classify robot control schemes is to classify them as computed torque like or non-computed torque like. As specified in [1], computed torque-like controls appear in robust control [2], adaptive control [3], learning control [4], hybrid control [5], and impedance control [6] etc. The non-computed torque like control appears in PID feedback [7] [8], neural network and fuzzy logic controls [9] etc.

Success of a motion control systems depends not only on the control algorithm but also on the control hardware structure. The control hardware structures used by existing controllers include i) purely relying on a motion control DSP [10]; ii) utilizing both a DSP and motion control ASIC (Application Specific Integrated Circuit) [11]; iii) using general-purpose DSP and external control resources such as a FPGA [12]; iv) using a modular structure [13]; and v) using multiple DSPs [14]. A combination of a DSP and an FPGA is the most popular structure at the present time, where FPGA implements I/O functions according to specific requirements.

Increase in productivity i.e. throughput, usually not higher accuracy, is the central driving force behind advances in the design of motion control hardware. What designers strive for is higher speed with the same error performance, providing higher productivity – all at low cost. It has been theoretically demonstrated that many advanced control algorithms (i.e. model-based controls) exhibit far superior performance over PID especially for high-speed systems with nonlinear dynamic characteristics. The relatively low update rate leads to degraded performance especially at high robot speeds. Further, due to the limited computational power of existing low cost control architectures, many advanced control algorithms are impractical since the complexity of these algorithms result in a low sampling frequency, leading poor control performance. A multi-DSP system may be a solution to realize a complex model based control algorithm, but its structure incurs limitation with regard to the data exchange between different DSPs [15]. A new controller architecture is therefore required, which can implement advanced controls resulting in substantially improved performance especially during high-speed motions.

FPGA have historically been used to create custom hardware logic circuit (such as I/O functions) that could not be performed by microcontroller or DSP. In recent years, FPGAs have been greatly improved due to enhancement of submicron process technology. Some researchers incorporated control algorithms into FPGA chip to enhance the performance of servo control systems. The system developed by Takahashi and Goetz [16] could run a current control algorithm with a Xilinx FPGA to increase the bandwidth of the current loop control. Tzou and Kuo [17] performed the vector and velocity controls of a PMAC servo motor by using FPGA technology successfully. Other works on FPGA based motion controls...
include Paramasivam [18], Bielewicz [19], and Dubey [20] PID control was used as the control algorithm in these works.

In this paper, a new motion control architecture with FPGA based motion control IC design is proposed to implement various robot control algorithms for improved performance in high speed motions. The main idea is to remove the servo control loop from the DSP to a FPGA, and utilize the high speed hardwire logic of FPGA to enhance the computation capability and relieve the load of DSP. In the proposed control structure, the control algorithms are partitioned into a linear portion and a nonlinear portion. The linear portion with position/velocity feedback represents the major control loop and is implemented in FPGA. The nonlinear portion acts as dynamic compensation to the linear portion to perform complex modeling related calculations, and is implemented in DSP. The core of this motion control IC design is a flexible motion control engine, designed with Verilog HDL (Hardware Description Language) and implemented in an industry standard FPGA provided by Xilinx. Experiments were performed on a Yamaha SCARA service robotic manipulator to demonstrate that under the same control algorithm, the motion performance with the new control architecture can be much improved compared to the conventional control architecture, especially in high-speed motions.

II. A NEW MOTION CONTROL ARCHITECTURE

The control structure with combination of DSP and FPGA, as shown in Fig. 1, is presently widely used. In this structure, the FPGA implements encoder logic, I/O interface, and other glue logic functions. The DSP performs inverse kinematics, trajectory generation, servo loop controls, and communication functions, etc. Since the burden on the DSP with this structure is high, it is usually infeasible to employ high frequency control updates when complex control algorithms are used.

With development of the IC technology, FPGAs have become the main stream in complex logic circuit design due to its flexibility, ease of use, and short time to market. The programmable hardwire feature of the FPGA provides a solution to the conflict between demanding computation requirements and cost. Therefore, it is beneficial to employ a FPGA as part of a digital controller to relieve the microprocessor/DSP from time consuming computations. If the control input can be done in a FPGA chip, custom parallel processing architectures can be embedded in a single device, and computations of several control signals will not impact each other and the load of microprocessor or DSP will be reduced greatly.

Fig. 2 illustrates the proposed control architecture. Compared to Fig. 1, the majority of the servo control loop has been moved from the DSP to FPGA. Due to the high speed nature of FPGA, the sampling frequency of the resultant motion control loop can be raised up to a new range that can never be reached by a conventional digital controller based on microcontrollers or DSPs. The DSP is now used only for dynamic compensation and other computations such as inverse kinematics and trajectory generation, which will be done more efficiently.

III. CONTROL ALGORITHM PARTITIONING

An important issue to successfully implement the above control structure lays in the realization of existing control algorithms in the FPGA. Through investigation of the existing control algorithms, it is found that many of these control algorithms can be treated as a computed torque-like control [1], which, without less of generality, can be partitioned into a linear portion and a nonlinear portion. The linear portion is a PD-like or PD plus feedforward controller, with actual and/or desired position/velocity, shown as follows:

\[
\tau = K_A(t)\ddot{q} + K_B(t)\dot{e} + K_C(t)e + K_D(t)\int e dt + K_E(t)
\]  

(1)

where, \(e\) and \(\dot{e}\) are errors of position and velocity, and \(e = q^d - q\); \(q\) and \(q^d\) denote the actual and desired position coordinates; \(K_A(t)\) is a feedforward control gain; \(K_B(t)\), \(K_C(t)\) and \(K_D(t)\) are feedback control gains; \(K_E(t)\) denotes the Coriolis and centrifugal control term. All of these control gains/parameters are time-varying, and will be updated by the nonlinear portion.

The nonlinear portion acts as the dynamic compensation to calculate the control gains/parameters used in the linear portion. It is with a nonlinear formulation containing time-varying dynamic modeling parameters, i.e.,

\[
[K_A, K_B, K_C, K_D, K_E] = F(q, \dot{q}, \ddot{q}, \dddot{q})
\]  

(2)

Note that PD and/or PID control can be treated as a special case without nonlinear portion.

As shown in Fig. 3, the linear portion can be treated as a special case without nonlinear portion.
simple structure with a linear relationship between position/velocity feedback and the updated control gains. The nonlinear portion calculates control gains/parameters used in the linear portion, and is realized in a DSP running at a lower sampling rate.

Table I lists examples of partitioning several existing control algorithms into the two portions. The listed control algorithms are PID control, PID plus acceleration feedforward control, the computed torque control [21], the adaptive control [3], the learning control [22], and the robust control [23].

IV. THE FPGA BASED MOTION CONTROL IC DESIGN

A. FPGA Based Motion Control IC

The FPGA based motion control IC to be designed will be used as a programmable ASIC (Application Specification Integrated Circuit). The functions include UART (Universal Asynchronous Receiver/Transmitter), Delta-Sigma analog to digital converter, velocity estimator, current sensing and closed loop current control, closed position and velocity control, PWM modulation, incremental encoder logic and brake/fault functions. Fig. 4 illustrates the proposed FPGA based motion control IC design, where all function blocks are modular and independently tested, and the entire system is formed by connecting these blocks together.

Although the FPGA design is written in Verilog HDL, the method used to configure the control system is via a configurable register interface. The user can thus configure these registers through DSP without changing Verilog code.

Several main function blocks are described below:

Position/velocity control loop (linear portion)

The linear portion of the control algorithm is designed using fixed-point computation. The signal of position is a 32 bits fixed point integer number. The velocity and acceleration signals are 18 bits fixed point integer number. The control gains are 18 bits fixed point number with 8.10 format (8 bits integer, 10 bits fraction). Velocity feedback is provided by a velocity estimator. With fast parallel hardware implementation, the update rate of the position/velocity control can be 20 kHz. In addition, a 32-bit and a 18-bit integer, 10 bits fraction). Velocity and acceleration gains are computed in DSP without changing Verilog code.

Current loop control

The inner current control loop employs a digital PI compensator, which is tuneable for various motors. The output of the inner current loop is PWM duty cycle given by

\[
PWM_{\text{cycle}} = K_{pc}i_e + K_{ic} \int i_e dt
\]

where, \(i_e = i_{set} - i_{act}\) denotes the current error, \(i_{set}\) is the set-point current, and \(i_{act}\) is the measured current; \(K_{pc}\) and \(K_{ic}\) denote proportional and integral gains, respectively.

Table I

<table>
<thead>
<tr>
<th>Control Type</th>
<th>(K_A(t))</th>
<th>(K_B(t))</th>
<th>(K_C(t))</th>
<th>(K_D(t))</th>
<th>(K_E(t))</th>
</tr>
</thead>
<tbody>
<tr>
<td>PID</td>
<td>0</td>
<td>(K_v)</td>
<td>(K_p)</td>
<td>(K_i)</td>
<td>0</td>
</tr>
<tr>
<td>PID plus feedforward</td>
<td>(J)</td>
<td>(K_v)</td>
<td>(K_p)</td>
<td>(K_i)</td>
<td>0</td>
</tr>
<tr>
<td>Computed torque</td>
<td>(M(q))</td>
<td>(M(q)K_v)</td>
<td>(M(q)K_p)</td>
<td>0</td>
<td>(N(q,\dot{q}))</td>
</tr>
<tr>
<td>Adaptive</td>
<td>(\dot{M}(q))</td>
<td>(\dot{M}(q)K_v)</td>
<td>(\dot{M}(q)K_p)</td>
<td>0</td>
<td>(\dot{N}(q,\dot{q}))</td>
</tr>
<tr>
<td>Adaptive control</td>
<td>(\Phi)</td>
<td>(\Phi K_v)</td>
<td>(\Phi K_p)</td>
<td>0</td>
<td>(\Phi v + \dot{N}(q,\dot{q}))</td>
</tr>
<tr>
<td>Learning control</td>
<td>(\dot{M}(q))</td>
<td>(\dot{M}(q)K_v)</td>
<td>(\dot{M}(q)K_p)</td>
<td>0</td>
<td>(F_k + \dot{N}(q,\dot{q}))</td>
</tr>
</tbody>
</table>

* Note: \(\Phi = (2d + d_z)^{-1}\) (the details can be found in [22]).
Trajectory generation

In the proposed architecture, the load of trajectory generation is shared by both DSP and PFGA. The desired trajectory is generated in two phases. In phase one, a reference trajectory is generated by the DSP, with the desired position, velocity, and acceleration, then buffered at the FPGA registers. In phase two, the trajectory signal generated in phase one is divided into a series of smaller increments to generate a fine trajectory in the using interpolation in the FPGA, described below:

\[ v(kT_m + iT_x) = v(kT_m) + \frac{i}{n}a(kT_m), \quad i = 0 \cdots (n-1) \]  

\[ p(kT_m + iT_x) = p(kT_m) + \frac{i}{n}v(kT_m), \quad i = 0 \cdots (n-1) \]  

where \( v(kT_m) \), \( v(kT_m) \), \( a(kT_m) \) are signals related to position, velocity and acceleration generated in phase one at the time \( kT_m \), \( T_m \) and \( T_s \) denote the sampling period of the DSP and FPGA, respectively, \( k \) is the time sequence number in DSP, \( n \) denotes the interpolated segments between the time \( kT_m \) and \((k+1)T_m\), and \( n = T_m / T_s \).

B. System Evaluation

The FPGA design described above has been realized in a FPGA chip provided by Xilinx with module No. XC3S400. An evaluation system was developed to test the performance of the proposed motion control IC, as shown in Fig.5. The entire system is constructed on a single PCB board with a compact size of 140mm × 100mm, and contains a MOSFET module, low voltage DC-DC, and high voltage supply. The control frequencies of the position/velocity loop and the current loop are 20 kHz and 120 kHz, respectively.

In the current loop test, a square wave current signal with amplitude switched between ±0.35A was generated. Fig. 6 illustrates the test results which are satisfactory. The position control loop was then examined by using a PID control algorithm. A DC brushless motor with an encoder of 4000 counts/revolution provided by Parker with module No. SM2328E-N10N, was used in the test. The motor was driven to move 47.1(rad) along a cubic polynomial trajectory, with the desired maximum velocity, acceleration and jerk of 157(rad/second) (i.e. 1500rpm), 3140(rad/second\(^2\)), and 31400 (rad/second\(^3\)), respectively. Figs. 7 and 8 illustrate the time responses. It can be seen that the performance of these time responses is good, which indicates that the proposed FPGA based motion control IC design works efficiently.

V. EXPERIMENTS

To verify effectiveness of the proposed motion control architecture, experiments were performed on a 2-dof Yamaha SCARA robot manipulator, as shown in Fig.9. Each arm of the manipulator uses an AC servo motor with an encoder of 8000 counts/revolution. Two reducer gears are mounted on the shoulder and elbow, with reduction ratio of 1:25 and 1:5.
In the experiment, the manipulator was required to move from an initial position \( q_{\text{initial}} = [0 \ 0]^T \text{(rad)} \) to a final position \( q_{\text{final}} = [0.5 \ 1.25]^T \text{(rad)} \) along a cubic polynomial trajectory, with high and low speeds respectively. In high speed motion, the desired maximum velocity, acceleration and jerk are \( 785 \text{(rad/second)} \) (i.e. \( 7500 \text{rpm} \)), \( 31400 \text{(rad/second} ^2 \) ) and \( 314000 \text{(rad/second} ^3 \) ). In low speed motion, the desired maximum velocity, acceleration and jerk are \( 78.5 \text{(rad/second)} \) (i.e. \( 750 \text{rpm} \)), \( 314 \text{(rad/second} ^2 \) ) and \( 3140 \text{(rad/second} ^3 \) ), respectively.

Figs. 11–13 illustrate experimental results with the two control architectures. Fig. 11(a), 12(a), and 13(a) and Fig. 11(b), 12(b), and 13(b) give the experimental results during high speed and low speed motions, respectively. It can be seen that under the same control algorithm, the new control architecture exhibits much improved motion performance (i.e., smaller position errors) compared to the conventional control architecture, especially in high-speed motions. It is also seen that with the new control architecture, the motion performance of the computed torque control and the adaptive control is improved substantially during high-speed motions compared to PID control. Such performance improvement implies higher productivity can be achieved, a very significant result.

VI. CONCLUSION

A new motion control hardware architecture is proposed by moving the servo control loop from the DSP to an FPGA. This is motivated by the fact that high speed hardwired logic of FPGA can greatly enhance controller computational capability while relieving the load of DSP. The control algorithm is partitioned into a linear portion and a nonlinear portion, implemented in an FPGA and DSP, respectively. The linear portion with position/velocity feedback represents the major control loop running at a fast sampling rate. The nonlinear portion calculates control gains/parameters used for the linear portion and acts as dynamic compensation. A FPGA-based motion control integrated circuit is designed to realize control functions in FPGA. Experiments conducted on a Yamaha SCARA manipulator demonstrate the effectiveness of the proposed approach, especially in high-speed motions.
ACKNOWLEDGEMENTS

This work was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CityU 119705), and a grant from City University of Hong Kong (Project no. 7001755).

REFERENCES


