2005 American Control Conference
June 8-10, 2005. Portland, OR, USA

Clustered-Architecture Motion Control System Utilizing IEEE 1394b Communication Network

Martin Hosek

Abstract— **This paper presents a scalable control system for synchronized control of complex multi-axis non-linear machines utilizing a communication network based on the IEEE 1394b high-speed serial bus standard. The system introduces a unique clustered architecture which allows for a high level of centralization of control algorithms in clusters of remote controllers in selected areas while promoting distribution of control algorithms running on substantially autonomous controllers in other areas. As a result, high performance control can be achieved where necessary without burdening the entire communication network by a heavy time-critical traffic and impairing the overall scalability of the control system. The distributed nature of the control system opens numerous challenges in the areas of implementation of model-based control, synchronization of individual controllers, and handling of events associated with inputs on multiple controllers. The paper proposes practical solutions to these challenges in the framework of the IEEE 1394b standard.**

I. Introduction

As a result of technological improvements in the area of electronics and communication, the control technology for complex motion systems has taken a clear direction toward a distributed architecture [1, 2]. As opposed to the centralized approach, where feedback signals are brought to a single, rather complicated controller that in turn drives all of the components subject to control, a distributed solution utilizes multiple simple controllers, referred to as remote controllers throughout the text, which are located close to the components subject to control, such as motors and other actuators, and coordinated by a master controller through a communication network.

The distributed architecture provides several advantages over the centralized approach. The remote controllers are typically designed as modules, which are either all identical or share as much hardware and software as possible, and can be easily added to support different levels of complexity of the machinery subject to control, thus promoting reuse of hardware and software, and making the control system scalable, flexible and configurable. The connections of the remote controllers with a central location are limited to a communication network cable and power, both of which can run relatively long distances. Since the remote controllers are located close to the respective components that they control, the length and complexity of wires and harnesses carrying feedback and action signals is minimized, thus reducing the cost of fabrication and assembly, improving noise immunity, limiting emissions and increasing the overall reliability of the system.

The distributed nature of a networked control system, however, opens numerous challenges in the areas of the distribution of the intelligence between the master and remote controllers, implementation of advanced control algorithms, utilization of the communication network, synchronization of components driven by different remote controllers, and handling of events associated with inputs on multiple remote controllers, such as capturing positions of multiple axes of motion, which either do not exist or are relatively easy to address in a centralized control system.

The control system presented in this paper provides practical answers to the above challenges in the framework of the IEEE 1394b communication network. The system has been designed as a scalable solution for synchronized control of complex multi-axis non-linear machines, such as precision robotic manipulators for automated semiconductor manufacturing applications. It utilizes a unique clustered architecture which allows for a high level of centralization of control algorithms in clusters of remote controllers in selected areas while promoting distribution of control algorithms running on substantially autonomous remote controllers in other areas. As a result, high performance control can be achieved for subsets of components where necessary without burdening the entire communication network by a heavy time-critical traffic and impairing the overall scalability of the control system.

II. IEEE 1394b Communication Network

The IEEE 1394 communication network, also known as FireWire, is based on a high-speed serial bus which was developed to provide the same services as modern parallel buses, but at a substantially lower cost. The IEEE 1394-1995 and IEEE 1394a-2000 standards [3-5] describe the initial specifications, including a standard read, write and lock operation set, asynchronous communication with data acknowledgment mechanism, isochronous data transport with guaranteed latency and bandwidth, and an accurate sub-microsecond global time-base for synchronizing events

Manuscript received September 14, 2004. M. Hosek is with Brooks Automation Inc., Chelmsford, MA 01824, USA (phone: 978-262-2987; fax: 978-262-2512; e-mail: martin.hosek@brooks.com).

and data. Two low-voltage differential signals are utilized to connect devices in a non-cyclic topology at up to 400 Mb/s data rates over single hop distances of up to 4.5 m. The arbitration system uses a self-configuring hierarchical request/grant protocol that supports hot plugging and widely varying physical topologies.

The initial version of the IEEE 1394 bus is constrained to operate over fairly short distance and cannot handle higher data rates, which limitations have been addressed in the IEEE 1394b-2002 amendment [6]. This standard defines features and mechanisms that provide gigabit speed extensions in a backwards compatible fashion, the ability to signal over single hop distances of up to 100 m, a large variety of interconnect media including Category 5 (CAT-5) unshielded twisted pair (UTP), plastic optical fiber (POF) and glass optical fiber (GOF), and detection of physical loops in the bus topology and their subsequent resolution by selective disabling of physical layer (PHY) port(s). These enhancements make the IEEE 1394b bus attractive as a communication network for motion control applications in the industrial environment.

The features that separate this bus from many other network solutions from control perspective include the following: determinism, synchronization features, high bandwidth (100Mb/s over CAT-5 UTP, 800 Mb/s over FireWire-specific cable and 3.2 Gb/s over POF or GOF), flexible utilization (different data rates can be used for individual network nodes, allowing for lower rates on remote branches which may be routed through slip-rings within a robot arm, and for higher rates on heavily utilized main branches of the network), generic tree topology, redundancy (physical loops in the bus topology can be used to provide redundant links; the network automatically resolves the redundancy, and reconfigures itself in the case of a failure of a link currently in use), plug-and-play, relatively long distance (up to 100 m over the CAT-5 UTP, POF and GOF physical media), noise immunity, wireless support, and attractive performance-to-cost ratio.

While the initial version of the IEEE 1394 bus has already been adopted in several commercially available control systems, the IEEE 1394b technology is relatively new with the first chipsets recently being introduced to the market. The control system described in this paper belongs to the very first adopters of the IEEE 1394b technology.

III. Clustered Architecture

Generally, in the distributed control architecture, remote controllers, which are located close to the components that they control, need to be able to acquire data, including feedback signals, produce control actions, such as driving motors, and communicate with the master controller. The role of the master controller is to coordinate the remote controllers and facilitate communication interface with an operator and/or a higher-level host computer, both of which responsibilities need to be handled centrally. The level of distribution of algorithms that run between the high-level functions of the master controller and low-level responsibilities of the remote controllers, such as trajectory generation and closure of feedback control loops, varies in the state-of-the-art control systems.

The highest level of centralization is seen in the motion control systems which close the feedback control loops over the communication network. In this case, the master controller is responsible for trajectory generation and runs all or most of the control algorithms. The functionality of the remote controllers is limited to reading feedback signals, such as actual positions of the motors subject to control, sending them over the communication network to the master controller, and producing control actions, such as applying voltage to the motors that are being controlled, based directly on the commands received from the master controller. In some cases, the remote controllers may perform simple control operations, such as closing subordinated current control loops. Since the current loops typically execute a torque command received from the master controller, this approach is frequently referred to as a torque mode of operation.

The benefits of highly centralized intelligence can be found in the following areas: Advanced control algorithms, which often require real-time information from multiple axes, such as multi-input-multi-output (MIMO) control, can be conveniently implemented on the master controller. The remote controllers require comparatively low computational power and small amount of on-board memory, which allows for a relatively simple and inexpensive design. These advantages are achieved at the cost of a relatively high utilization of the communication network due to intensive time-critical data exchange between the remote controllers and the master controller which closes the control loops, and limited scalability of the system with the maximum number of axes being constrained by the computational power of the master controller as well as the bandwidth of the communication network.

Control systems with distributed algorithms, on the other hand, utilize more intelligent remote controllers which close the control loops locally and execute on the trajectory information received from the master controller in the form of periodic data frames. The data frames typically include position, velocity and time (PVT) information for each axis of motion, and are sent by the master controller at a rate substantially lower than that of the control loops. Since the PVT frames can be viewed as set points for the remote controllers, this approach is often referred to as a set-point mode of operation. Another step in increasing the autonomy of the remote controllers is moving the trajectory generation calculations from the master controller to the remote

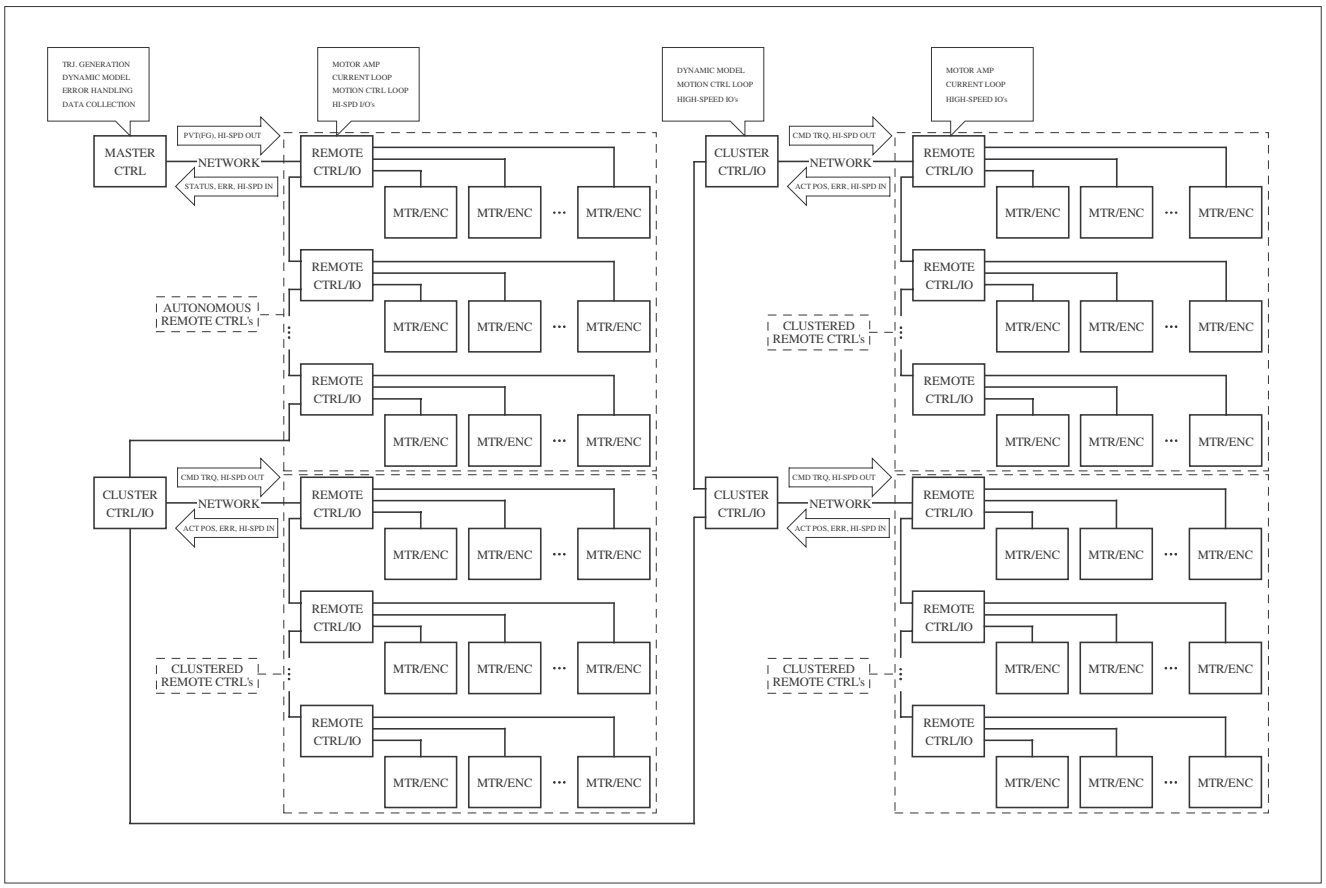

Fig. 1. Clustered-architecture control system.

controllers. This approach, which is also referred to as a point-to-point mode of operation, further reduces the traffic over the communication network. However, it makes it difficult to support applications with kinematically coupled axes controlled by multiple remote controllers.

Control systems with distributed control algorithms typically utilize a comparatively low network traffic which involves substantially less time-critical data. This is because the frequency of PVT frames can be substantially lower than that of the control loops, and the PVT frames can be conveniently buffered at the remote controllers to compensate for jitter and to improve the overall safety margin of the communication network. This approach also provides a desirable level of scalability of the control system since an increased number of remote controllers needed for additional axes automatically scale the overall computational power available for execution of the control loops. Furthermore, the remote controllers, being capable of generating simple trajectories and running control algorithms, have enough intelligence to support operations required by safety regulations, such as a controlled stop in the case of a failure of the communication network. As a drawback, this additional intelligence requires more computational power and memory, which may make the remote controllers more expensive. Also, implementation of full MIMO control algorithms is problematic since the control law running on one remote controller generally does not have real-time access to the states of the axes driven by other remote controllers.

The distributed control system described in this paper combines the benefits of the two approaches. It is based on a unique architecture which allows for a high level of centralization of control algorithms in clusters of remote controllers in selected areas while promoting distribution of control algorithms running on substantially autonomous remote controllers in other areas. As a result, high performance control can be achieved for subsets of components where necessary without burdening the entire communication network by a heavy time-critical traffic and impairing the overall scalability of the control system.

The clustered-architecture control system consists of a master controller, multiple cluster controllers and a plurality of remote controllers which are connected through a communication network, as depicted diagrammatically in Fig. 1, each of the cluster controllers supervising multiple remote controllers, and each of the remote controllers, subordinated either to the master controller or to a cluster controller, being utilized to drive one or more axes, each

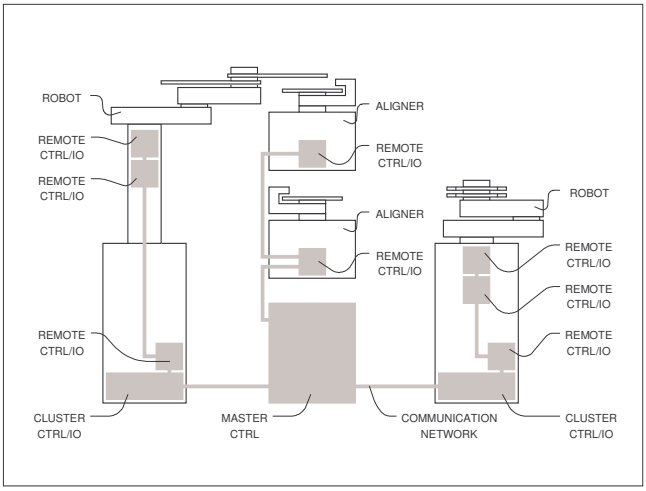

Fig. 2. Work-cell with two robots and two aligners.

axis being represented by a motor-encoder block in Fig. 1.

The master controller performs high-level supervisory and control functions, including a communication interface with an operator and/or a higher level host computer, configuration data management, coordination of cluster and remote controllers, motion sequencing and trajectory generation, dynamic modeling (if required), error handling and data logging. The cluster controllers run real-time control algorithms, including local dynamic models (if necessary), and provide torque command for remote controllers subordinated to the cluster controllers. The remote controllers subordinated to a cluster controller operate in a torque mode, performing control functions necessary to execute the torque command received from the cluster controller, and communicating to the cluster controller information necessary to determine torque command, such as actual positions of the axes controlled by the remote controllers. The remote controllers subordinated directly to the master controller may operate in a set-point mode, in which case they execute real-time algorithms to determine torque command for axes subject to control using trajectory frames obtained from the master controller. In another mode of operation, referred to as a point-to-point mode, the remote controllers subordinated directly to the master controller receive a motion command from the master controller, and generate trajectories and run real-time control algorithms to execute the motion command.

The IEEE 1394b bus is particularly suitable as a communication network for a clustered-architecture control system. It supports a tree topology, allows for different data rates for different network nodes, and provides asynchronous and isochronous modes of communication with a data acknowledgment mechanism and guaranteed latency, respectively. The asynchronous mode can be used for less time-critical data, such as motion commands (which require acknowledgments) and PVT frames, while the isochronous data transport can be utilized for real-time torque commands. The IEEE 1394b bus can support up to 63 cluster and remote controllers, which limit is dictated by the maximum of 64 nodes on a single IEEE 1394 network.

An example application of a clustered-architecture control system to a work-cell including a pair of robotic manipulators cooperating with a pair of semiconductor substrate aligners is illustrated in Fig. 2. While the aligners are simple devices with two axes of motion, the robots are five-axis machines with complex nonlinear dynamics [7]. In this particular example, the robots pick substrates from designated locations, place them on the aligners, which scan each substrate for eccentricity and alignment features, pick the substrates from the aligners, and place them to other designated locations. In order to maximize the throughput of the work-cell and to guarantee collision-free operation, the motion of the robots and aligners needs to be synchronized accurately. Since there is no dynamic coupling among the four devices, they can use independent control algorithms without real-time data sharing. On the other hand, the mechanical configuration of the robotic manipulators introduces strong dynamic coupling of individual axes within each robot, which can be addressed by implementing a separate cluster controller for each one.

The resulting configuration of the control system is depicted in Fig. 2. The time-critical traffic associated with closing the control loops over the network is contained within each of the two clusters without burdening the rest of the network, particularly the main branch from/to the master controller, which would be subject to a heavy real-time traffic in the conventional architecture with centralized control algorithms.

IV. Distributed Model-Based Control

High-performance motion control of complex mechanical systems, such as articulated robotic manipulators, needs to take into account non-linear properties and dynamic coupling between individual axes. This may be achieved by utilizing various MIMO control algorithms, such as the computed torque technique [9, 10]. However, truly MIMO control algorithms are problematic to implement in a fully distributed control system, as explained in the previous section. This section presents a simplified approach which may be used to take into account non-linear properties of the system subject to control and compensate partially for dynamic coupling between axes.

In the proposed approach, the master controller generates trajectory for each axis in terms of the commanded position, velocity and acceleration, calculates inverse dynamic model of the system subject to control to determine a gain and feedforward term, and groups the resulting information into frames, each frame being specific to a particular axis. The communication network distributes the frames periodically

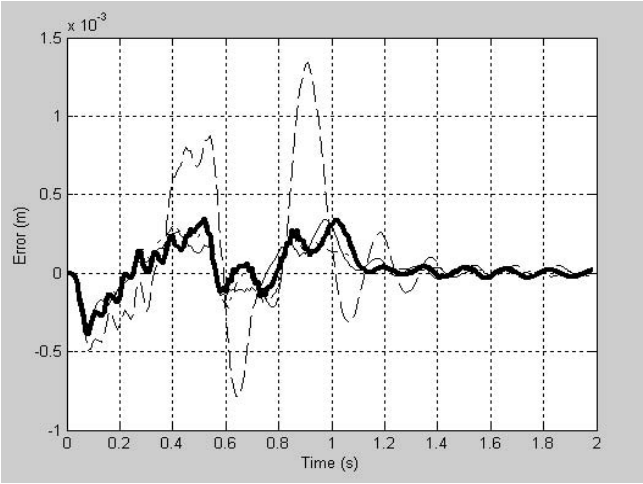

Fig. 3. Tracking error for MIMO control (thin), distributed model-based approach (bold), SISO feedback (dashed) and distributed approach with disturbance observer (dash-dotted).

to remote controllers subordinated directly to the master. The remote controllers which receive the data interpolate between two consecutive frames to obtain the instantaneous position, velocity, feedforward term and gain value, and utilize this information in the control loop.

Consider a dynamic model in the form of:

$$\begin{aligned}\tau_i &= \sum_{j=1}^{n} M_{ij}(\{\theta_{act}\})[\ddot{\theta}_{cmdj} + u_j(\theta_{cmdj}, \dot{\theta}_{cmdj}, \theta_{actj}, \dot{\theta}_{actj})] + h_i(\{\theta_{act}\}, \{\dot{\theta}_{act}\}) \\ &= \sum_{j=1}^{n} M_{ij}(\{\theta_{act}\}) u_j(\theta_{cmdj}, \dot{\theta}_{cmdj}, \theta_{actj}, \dot{\theta}_{actj}) + \sum_{j=1}^{n} M_{ij}(\{\theta_{act}\})\ddot{\theta}_{cmdj} + h_i(\{\theta_{act}\}, \{\dot{\theta}_{act}\})\end{aligned} \quad (1)$$

where τ_i denotes torque applied to axis i, θ_{acti} and θ_{cmdi} are actual and commanded positions of axis i, respectively, $\{\theta_{act}\}$ and $\{\theta_{cmd}\}$ stand for position vectors defined as $\{\theta_{act}\} = \{\theta_{act1}, \theta_{act2}, ..., \theta_{actn}\}^T$ and $\{\theta_{cmd}\} = \{\theta_{cmd1}, \theta_{cmd2}, ..., \theta_{cmdn}\}^T$, M_{ij} is an inertia matrix element indicating inertial cross-coupling between axes i and j, h_i is a function including other position- and velocity-dependent dynamic effects for axis i, such as centrifugal, Coriolis and gravity terms, u_i stands for feedback control output for a unit-inertia system corresponding to axis i, i varies from 1 to n, n is the total number of axes or degrees of freedom, and dot and double-dot indicate first and second time derivatives, respectively.

Provided that position and velocity tracking errors are small, the actual positions and velocities can be approximated with commanded quantities. Furthermore, neglecting the $M_{ij}u_j$ terms for $i \neq j$, the dynamics of (1) can be rewritten and approximated as:

$$\begin{aligned}\tau_i &= M_{ii}(\{\theta_{cmd}\}) u_i(\theta_{cmdi}, \dot{\theta}_{cmdi}, \theta_{acti}, \dot{\theta}_{acti}) + \sum_{j=1}^{n} M_{ij}(\{\theta_{cmd}\})\ddot{\theta}_{cmdj} + h_i(\{\theta_{cmd}\}, \{\dot{\theta}_{cmd}\}) \\ &= G_i u_i + F_i\end{aligned} \quad (2)$$

where:

$$G_i = M_{ii}(\{\theta_{cmd}\}),\ F_i = \sum_{j=1}^{n} M_{ij}(\{\theta_{cmd}\})\ddot{\theta}_{cmdj} + h_i(\{\theta_{cmd}\}, \{\dot{\theta}_{cmd}\}) \quad (3)$$

The above expressions for G_i and F_i define the gain and feedforward terms, respectively. The index i indicates which axis the two terms will be applied to.

To verify the proposed approach and to quantify potential performance losses, the distributed model-based control algorithm was applied to a 5-axis robotic manipulator for automated pick-place operations in semiconductor manufacturing applications, such as the work-cell described in the example of the previous section. The robot consists of an articulated arm with two end-effectors operating in parallel horizontal planes complemented by a vertical lift drive. The arm represents a challenging test bed since the dynamics is highly non-linear with strong cross-coupling among axes. A detailed description of the robot, including the mechanical configuration, major dimensions and dynamic properties, can be found in [11].

Among other operations, the robot was commanded to perform a straight-line extension move of one of the end-effectors in the radial direction. The resulting tracking error of the end-effector, in particular its normal component which indicates the deviation of the end-effector from the desired straight-line path, is presented in Fig. 3. The MIMO control of (1) is shown in solid thin line, the simplified model-based approach according to (2) is represented by the solid bold graph, and a conventional single-input-single-output (SISO) feedback is depicted in dashed line. For the MIMO and simplified model-based cases, the feedback portions of the control laws, i.e., u_i in (1) and (2), were selected as conventional PID loops with identical sets of gains. Similarly, the SISO feedback was designed as a PID loop with gains taken from the MIMO case so that at least the same stability margin was achieved in any position of the robot. It is observed that the tracking performance of the simplified model-based control closely approaches the performance of the MIMO case, and clearly outperforms the conventional SISO feedback.

The performance of the distributed model-based approach can be further improved by utilizing a disturbance observer for each axis. Conveniently, the disturbance observer can replace the integrator in the feedback portion of the control law. This case is represented by the dash-dot graph in Fig. 3.

Considering 100 Hz frequency of PVT(FG) frames and 2 kHz position and velocity sampling for event capture purposes (Section V), the IEEE 1394b communication network can handle 126 axes of motion at 100 Mb/s data rate with 30% utilization, or 512 axes at 400 Mb/s data rate with 50% utilization. In both cases, there is sufficient capacity available to transfer additional data, such as video streaming in applications with integrated vision systems. The practical limiting factor for the number of axes thus becomes the computational power of the master controller and the practicality of such a large number of axes (and

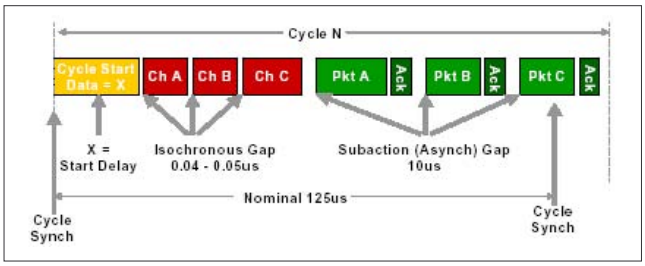

Fig. 4. Typical FireWire cycle.

machines) being dependent on a single master controller, in particular from the reliability perspective.

V. Communication Network Synchronization

The operation of a distributed control system depends on synchronous execution of commands and trajectories by a set of remote controllers connected through a communication network. The approach described in this section enables synchronization of motion control nodes using the time management features available in the IEEE 1394 standards.

The synchronization of the communication network is achieved by maintaining a common time reference (a common clock) on all network nodes, including the master, cluster and remote controllers, combined with time-stamping of the time-critical network traffic (PVT-frames, I/O events, status messages, etc). The IEEE 1394 network provides a mechanism to keep the clocks at the nodes in sync with the so called "bus-time". A typical bus cycle takes about 125 μs and consists of a "cycle start" message, followed by an interval for isochronous traffic and an interval for asynchronous traffic, as shown diagrammatically in Fig. 4. The nodes can maintain synchronized clocks by utilizing the following FireWire features:

- A "cycle start" message, which is issued by the cycle master node at the beginning of each 125 μs cycle. The message contains a timestamp of the time when the message went out on the bus with resolution of ~40 ns.
- A built-in hi-resolution clock in each (isochronous-capable) node.
- "Cycle-time" (hi-resolution) and "Bus-time" (low-resolution) registers on each node. Combination of those two registers provides a clock with resolution of 40 ns that will overflow every 136 years.
- A mechanism for auto-update of the "Cycle-time" register based on the "cycle start" message.
- An interrupt mechanism providing indication to the application layer on sync events, including "cycleSynch" interrupt when a new isochronous cycle has started.

One of the nodes in the communication network, typically the master controller, plays a central role. This node is referred to as the "supervisor node" in this section. The nodes that are coordinated by the supervisor node, typically the cluster and remote controllers described in Section III, are called "slave nodes".

The proposed mechanism for synchronization of the communication network in the FireWire-based distributed control system operates as follows:

- After a bus reset, the supervisor node enforces to be a root node and becomes a cycle master for the network.
- The supervisor node maintains a bus-time register.
- Each slave node maintains a bus-time register and updates it based on the cycle-time register available to the node. FireWire features provide synchronization of cycle-time registers across the network.
- The supervisor node sends PVT frames to the slave nodes stamped with the time which indicates when in the future the PVT frame should be executed.
- Each slave node maintains software PLL of its current and position loops analyzing CycleSynch interrupts from the FireWire link layer controller and its cycle-time and bus-time registers.

The accuracy of the synchronization mechanism is affected by several sources of error, including the following factors:

- Bus-time resolution. The resolution of the bus-time message and registers introduces jitter of about 40 ns.
- Cycle-start message set-up time on master node. The cycle-start packet is formed by the network link layer controller (LINK), and contains the time when the physical layer controller (PHY) communicated to the LINK that it had won arbitration of the bus. The first bit of the message starts immediately after that. The resulting delay, assumed to be less than 10 ns jitter, can be considered negligible in most applications.
- Propagation delay. The propagation delay of the cycle-start message from the master node to the recipient slave node depends on the length of the cable segments and the number of repeater nodes between the master and recipient nodes. These two components can be quantified as approximately 5 ns/m and less than 144 ns/repeater.
- Transmission time. The delay coming from the transmission time of the cycle-start message totals about 2.2 μs.
- Register update time on slave nodes. The slave nodes set their cycle timer register immediately on receipt of the cycle-start message. The resulting delay, assumed to be less than 10 ns jitter, can be neglected in most applications.

The two major contributors to the synchronization error, the propagation delay and transmission time, are deterministic and can be compensated for. The transmission time is known and constant. The propagation delay between two nodes can be conveniently estimated by measuring the

time interval between a PING message issued by the master node and the receipt of a Self ID response from each slave node. The error resulting from the propagation delay and transmission time then can be accounted for by adding/subtracting the corresponding time offsets to/from the "bus-time" on the slave nodes.

The latency introduced by the transmission time is irrelevant in many applications since it does not affect synchronization of the remote controllers relative to each other. The synchronization error due to the propagation delay is often acceptable because the resulting position error may be below the required accuracy. As an example, the synchronization error for the 5-axis robot of Sections III and IV is around 0.5 μs, the maximum synchronization error in a system with 16 hops and a hop distance of 4.5 m is less than 2.5 μs.

VI. Distributed Event Capture

In many applications, the control system is required to capture instantaneous values of specified variables on multiple remote controllers, such as actual positions and velocities of the axes subject to control, based on an internal or external event, e.g., a change of a digital input residing on one of the network nodes [12]. The distributed event capture method described below provides the above functionality utilizing the following steps:

- Controller nodes buffer position data and/or other quantities of interest at a lower sampling rate than that of the servo loop.
- When a digital input or other event triggers event capture on a controller node, current time at the respective node is recorded.
- A message with the time information is send from the controller node where event capture has been triggered to all other or a specified subset of controller nodes.
- The specified nodes interpolate buffered positions and/or other quantities of interest to determine their values at the given time.
- The specified nodes send the positions and/or other quantities of interest determined as a result of interpolation to the master controller.

In some cases, direct communication among remote nodes may not be desirable, e.g., due to a large number of communication plugs required. In order to address this concern, the master controller can receive the time when the event occurred and distribute it to the specified nodes. As another alternative, if the remote nodes do not posses sufficient resources for data buffering and interpolation, in particular due to limited memory and computational power, these operations can be performed by the master controller. However, the cost of this approach is increased traffic over the communication network.

It should be noted that the accuracy of the event capture mechanism does not depend on the data transmission speed of the communication network, which only affects how fast the event capture results can be made available. However, several other factors contribute to the overall accuracy of the event capture mechanism, including the following:

- I/O hardware latency and interrupt service routine (SIR) execution time - can be typically kept well below 2 μs.
- Synchronization error - a detailed discussion of this error component can be found in the previous section.
- Interpolation error - depends on the dynamics of the system subject to control, the sampling interval and the degree of the interpolation polynomial.

For instance, the latency of the position capture mechanism is less than 3 μs for the example 5-axis robot of Sections III and IV, or less than <5 μs for a system with 16 hops and a hop distance of 4.5 m. This corresponds to a position error less than 5 μm at 1 m/s speed, which is acceptable for the targeted motion control applications.

VII. Conclusion

The distributed control system described in this paper provides a means of accurate and synchronized control of complex multi-axis non-linear machines, which are exemplified by precision robotic manipulators for semiconductor manufacturing applications, with a desirable combination of performance, flexibility and scalability. Selected features of the system disclosed in this paper are subject to a patent application process.

References

[1] Rawlings E., Trends in Precision Motion Control, Design News, September 2001, available at http://www.designnews.com.

[2] Scheiber S., Decentralized Control, April 2004, available at http://www.controleng.com.

[3] IEEE Std 1394-1995, IEEE Standard for a High Performance Serial Bus, available at http://standards.ieee.org.

[4] IEEE Std 1394a-2000, IEEE Standard for a High-Performance Serial Bus - Amendment 1, available at http://standards.ieee.org.

[5] Anderson D., FireWire System Architecture, Second Edition, Addison-Wesley, Boston, 1999.

[6] IEEE Std 1394b-2002, IEEE Standard for a High-Performance Serial Bus - Amendment 2, available at http://standards.ieee.org.

[7] Hosek M., and Bleigh T., Brooks Automation Expands Control Capability of Precision Industrial Robots, Industrial Robot: An International Journal, No. 4, July 2002, pp. 334-348.

[8] Hosek M., Observer-Corrector Control Design for Robots with Destabilizing Unmodeled Dynamics, IEEE/ASME Transactions on Mechatronics, Vol. 8, No. 2, June 2003, pp. 151-164.

[9] Craig J. J., Introduction to Robotics - Mechanics and Control, Addison-Wesley, Reading, Massachusetts, 1986

[10] Fu K. S., Gonzales R. C., Lee C. S. G., Robotics: Control, Sensing, Vision, and Intelligence, McGraw-Hill, New York, N.Y., 1987

[11] Moura J. T., and Hosek M., Neural Network Based Perturbation Identification Approach for High Accuracy Tracking Control of Robotic Manipulators, Proceedings of 2003 ASME IMECE, Paper No. IMECE2003-41179, November 16-21, 2003, Washington, DC.

[12] Hosek M., and Prochazka J., On-The-Fly Substrate Eccentricity Recognition for Robotized Manufacturing Systems, ASME Journal of Manufacturing Science and Engr., Vol. 127, February 2005.