A Distributed I/O Low-Level Controller for Highly-Dexterous Snake Robots

Paul Thienphrapa, Peter Kazanzides
Department of Computer Science
Johns Hopkins University
Baltimore, MD, USA

***Abstract*—A robot can be seen as a set of actuated joints, each connected to a processing unit that senses feedback signals and generates actuation commands. For robots with high locomotive sophistication, commonly seen in medical robotics, the requisite cabling and control processing can become unwieldy. Motivated by dexterous snake-like robots for minimally invasive surgery, this paper explores the use of IEEE 1394 (FireWire), attached directly to low-latency field-programmable gate arrays (FPGAs), to distribute I/O and centralize processing. This increases the viability of complex medical robots by reducing cabling and consolidating processing, thereby making systems more agile, reliable, and scalable.**

I. Introduction

Minimally invasive surgery (MIS) is often a direct benefit for patients due to reduction of trauma, leading to fewer complications and shorter hospital stays. MIS poses a number of challenges for surgeons, including constrained workspaces, limited field of view, and lack of dexterity at the distal end. These challenges remain despite the availability of manual MIS-specific instruments, which are rigid and difficult to handle through narrow insertion tubes.

A. Snake Robot

A unique design targeted for MIS of the upper airways, the Snake Robot [1] addresses these issues by introducing small, dexterous snake-like units (SLUs) that can be teleoperated. To avoid obscuring the work area, these SLUs are affixed to a 1-m narrow shaft containing its wires and appear at the distal end of a laryngoscope; the shaft can also be adjusted under remote control (see Fig. 1).

The Snake Robot is an eight degree-of-freedom (dof) manipulator. Multiple Snakes may be used for surgical tasks such as suturing and suction. The distal end of each manipulator consists of two SLUs connected in series; each SLU is constructed using four super-elastic NiTi tubes. The anatomy of an SLU is shown in Fig. 2. The center tube, which is the primary backbone, is connected to all of the discs, including the base disc, end disc, and intervening spacer discs. Surrounding the primary backbone at equally-spaced distances are the three other tubes, the secondary backbones. These are fixed to only the end disc and are free to glide through holes in the intermediate spacer discs. Two dof result from pushing and pulling the secondary backbones using three actuators located at the proximal end. The push-pull actuation modes help prevent the backbones from buckling while satisfying structural statics [2]. The second SLU is appended to the end of the first SLU. The backbones of this second SLU actually pass through the three hollow secondary backbones of the first SLU. Attached to the end of the second SLU is a gripper that is actuated via a wire passed through the hollow central backbone. This mechanical composition allows for a high payload capacity with a small size [3]. The existing prototype is 4 mm in diameter.

Figure 1. Snake Robot prototype [3]

This work was funded by the National Science Foundation (NSF) Engineering Research Center grant #EEC9731748, NSF grant #IIS9801684, NSF grant MRI 0722943, and by Johns Hopkins University internal funds.

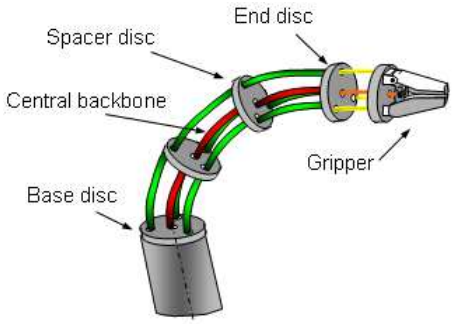

Figure 2. Anatomy of a snake-like unit [2]

The aforementioned actuators (seven in total) are encased atop the shaft in a compact cylindrical unit; the shaft-cylinder assembly can be controlled with four additional dof. A Z-Θ stage allows for rotation and translation about and along the shaft respectively. A passive universal joint mounted on a five bar mechanism supports the shaft and provides XY mobility. This stage also stabilizes the robot against lateral deflections.

B. Control Hardware for Robots of High Dimensionality

Each Snake is actuated by 11 dc motors, accounting for two three-axis SLUs, a gripper, a two-axis Z-Θ stage, and a two-axis five bar mechanism. In the prototype, two seven-axis daVinci masters are used to command two Snake slaves for bimanual control.

The low-level controller for the Snake Robot must be scalable to handle at least 36 axes (i.e. two 11-axis Snakes and two seven-axis masters), and possibly more in the future. A significant reduction in dimensionality and hardware complexity was achieved via the push-pull actuation of flexible wires combined with derived kinematics [1], as opposed to actuation of several precision joints. Nevertheless, the number of control axes for the Snake Robot remains considerable, as does its associated cabling. One can envision the increases in both as robots are devised for more sophisticated surgical tasks. By distributing I/O over an IEEE 1394a (henceforth without the 'a') bus to nodes with low-latency field-programmable gate arrays (FPGAs), and by centralizing processing, the control system hardware proposed here aims to mitigate the potential problems in robot mobility and reliability that arise with increasing structural complexity.

II. Low-Level Control System

Fig. 3 provides an overview of the low-level controller, with I/O distributed away from the computer. Each node contains multiple channels, corresponding to the axes handled by the node. Nodes can be added by daisy-chaining them on a single 1394 bus or by connecting them to the computer via an additional 1394 bus. The bus is attached to a real-time computer that reads feedback signals from the channels, generates actuation commands, and writes them to their respective channels.

A. Nodes

A node, detailed in Fig. 4, contains circuitry that allows it to access I/O channels and handle 1394 packets. IEEE 1394 allows up to 63 nodes per bus. Multi-bus configurations can be used for yet larger numbers of axes or for heterogeneous control environments.

B. Channels

A channel, or axis, contains the I/O components (e.g., DAC, ADC, quadrature encoder counter) and power amplification required to control one dc motor. The channel under development (in Fig. 4) also includes a digital pot that allows software configuration for different motors and a digital switch to select between speed and torque control. This makes this node-channel set applicable to other robots.

The number of channels per node depends on the physical distribution of joints, limited by the memory and I/O capacity of the resident FPGA.

C. Field-Programmable Gate Array

Most of the functionality of each node is implemented as firmware on the FPGA, which serves as a low-latency interface between the channel I/Os and the bus. The FPGA receives packets from the 1394 bus, responds to them, and communicates with the I/O devices. The computer can access the channels through control and data registers. Fig. 5 depicts the FPGA operation.

D. Other Components

The Texas Instruments TSB41AB2 is an IEEE 1394 physical layer IC that can handle standard bus speeds up to 400 Mbps. It generates the 49.152 MHz clock signal used to synchronize data and clock the FPGA.

For noise isolation, and to facilitate emergency shutdowns, the motor and digital voltages are drawn from separate regulated supplies. Power from the 1394 bus is not used for these reasons, as well as to simplify the development effort.

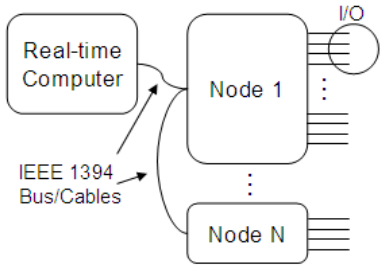

Figure 3. Conceptual overview of the low-level controller

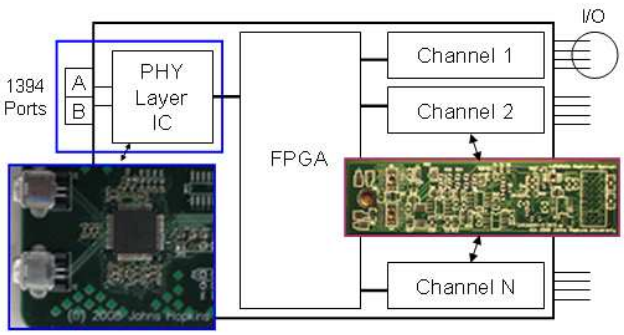

Figure 4. Block diagram of a node

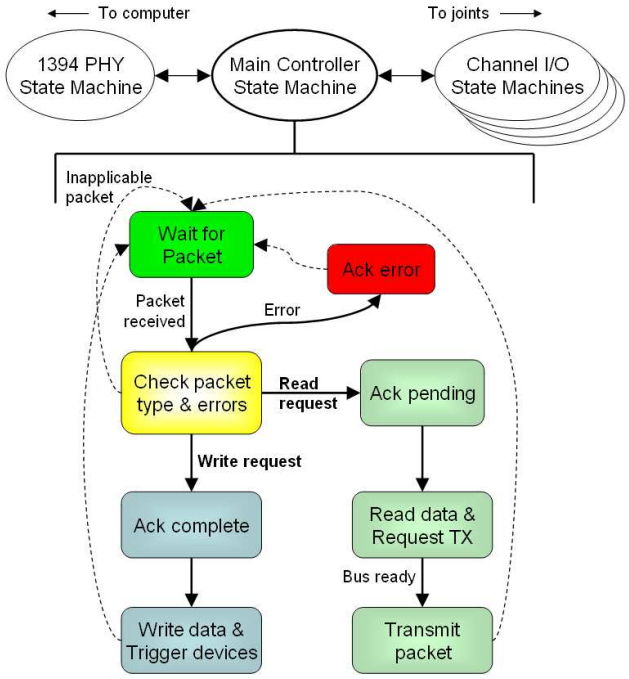

Figure 5. FPGA structure and operation

Linux on a conventional workstation (with a 1394 port) is being used for development purposes, with the intent of migrating to a real-time version of Linux (e.g. RTAI) to run the control software and robot applications. Programs use the libraw1394 library to communicate with nodes; RT-FireWire [12] is being considered as an alternative.

III. Advantages

The advantages of using IEEE 1394 versus traditional parallel buses are multifold. IEEE 1394 can multiplex data from many channels, reducing the cabling complexity as the number of channels grows. This leads to improvements in hardware reliability, mobility, packaging, and scalability.

Parallel buses limit the number of I/O channels that can be connected to one computer. For example, an industrial-grade computer can reliably accommodate four ISA cards, and the number of channels per card is constrained to a modest number by physical size. As a result, a single robot may need multiple computers to run. In contrast, 1394 allows a large number of channels to be handled by a single computer. Low-latency inter-task communication can be used instead of network communication. Because the processing can be done on one computer, the architecture is better suited to harness the power of high performance computing. The integration allows for a familiar software development environment, and it alleviates the need for one to learn the idiosyncrasies of various embedded microprocessors, allowing researchers and developers to focus on higher level tasks.

The generality of the channels and the need for many such boards on a dexterous robot help drive down the per-axis cost. The design encourages modularity, making the assembly compact and the boards conveniently swappable.

IV. Alternatives to IEEE 1394

IEEE 1394 was selected because the protocol supports real-time communication with guaranteed 8 kHz (125 µs) bus cycles in isochronous mode, with faster access rates possible in asynchronous mode, and because it allows for daisy-chaining of nodes. It is an effective solution for real-time control, as shown in [6, 12], and by its use in fly-by-wire systems [7], but it is not necessarily the single best choice.

The works of [6] and [12] focus on real-time control bandwidth, but not the physical benefits of distributed I/O and centralized processing emphasized here. The differences manifest in their use of PCI cards and onboard computers, contrasting with our use of compact custom electronics.

Fair bus access is incorporated into IEEE 1394 hardware; bus arbitration in Ethernet is nondeterministic, but kilohertz-range motor control is achievable on isolated networks with software modifications [14, 15]. Several Ethernet variations have been developed that make the medium very promising. Powerlink (ethernet-powerlink.org) employs a bus manager that schedules 200-µs cycles of isochronous and asynchronous phases. In EtherCAT (ethercat.org), nodes forward and append data packets as they are received with the aid of dedicated hardware and software, resulting in the ability to communicate with 100 axes in 100 µs. EtherCAT is a relative newcomer; [8] is an example showing its potential. SERCOS approached a communication bottleneck in [9] with increasing axes and cycle rates, but its recent combination with Ethernet (SERCOS-III) has endowed it with the ability to update 70 axes every 250 µs. Similarly, the Controller Area Network (CAN, can-cia.org) bus is well-suited for real-time control, but its bandwidth is limited to 1 Mbps.

CompactPCI is an industrial backplane interface capable of 132 MB/s throughputs, used notably in space systems by NASA in transitioning from VMEbus [10]. PCI Express is a new serial interface designed to replace computer expansion buses; a cable-based standard was not fully established at the time of the designs presented in this paper. PCI Express supports real-time applications such as the industrial control example in [11].

High data rates are readily available with USB, but its reliance on the host processor for bus level tasks compromises its scalability in real-time control. Conversely, IEEE 1394 self-manages the bus at the physical layer.

V. Timing Results

As part of a control loop, the computer reads the ADCs and encoder feedbacks and writes the DACs. The reads are implemented on the FPGA such that there is no protocol delay (i.e. no busy wait) on the FPGA between receiving a read request and generating a response. Similarly, there is no delay between the receipt of a write request and the start of the write. The I/O device access times are negligible (~2.5 µs, deterministic) relative to the system bandwidth. Contention is not expected on the bus because the computer is implemented as the bus master and the nodes as slaves.

A read transaction entails a request from the computer, an acknowledgment from the node, and a data response from the

node. A write transaction is the same, save for the response. Fig. 6 shows the timing results of 9,000 iterations of quadlet reads and writes. The tests were run at 400 Mbps with one node connected to a 2 GHz Pentium 4 PC by a 6' cable.

The read times appear in three bands (30, 42, 49 µs) and the write times in two (27, 42 µs), possibly due to variability in discrete transaction sequences (e.g. request-ack-response); the average read and write times are 34.5 and 30.2 µs respectively—each respective low band is most common. We will investigate the regular distribution patterns as well as outliers should they arise under a real-time operating system.

High-valued outliers are sporadic and are likely artifacts of scheduling or memory accesses. Overall the times are well above theoretical maxima—e.g. a quadlet read, 296 total bits, should only take a fraction of a microsecond to complete at 400 Mbps. Operating system overheads dominate (as in [13]) but should not increase appreciably with data size.

VI. Conclusions and Future Work

Though parallel buses such as ISA, Q-Bus, Multibus, and VME have become tried-and-true interfaces for robot control, they are increasingly deprecated with the emergence of IEEE 1394, PCI Express, and Ethernet-based protocols, which feature greatly simplified cabling. These high-speed serial networks provide higher performance than traditional field buses, such as CAN, SERCOS, and RS-485, which have also been used for real-time control.

This paper proposes a promising controller design based on IEEE 1394 for communication between the computer and the actuated joints. The results confirm the desired real-time performance. The advantages of distributing I/O to less obtrusive sites are discussed, particularly for medical robots.

The IEEE 1394 bus helps reduce wiring complexity, making systems more robust. The consolidation of processing tasks eases intra-robot communication (e.g. master-slave) and allows systems to utilize ever-advancing computing power.

The design encourages modular, general electronics. With the need for many controllers in agile medical robots, this factor helps reduce the average cost. Modularity makes the boards easier to package, debug, and replace.

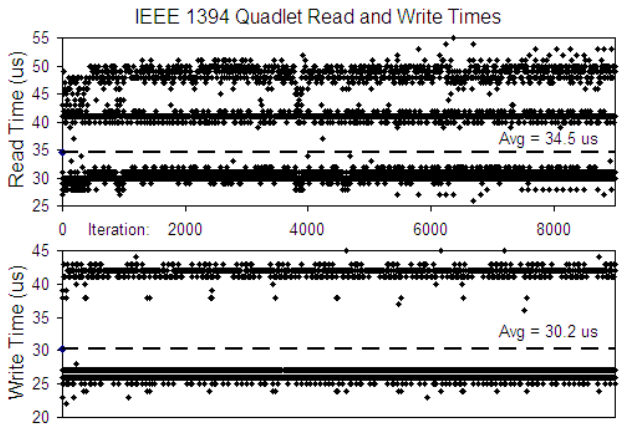

Figure 6. Quadlet read and write times over 9,000 iterations

Block transfers are being integrated into node firmware to amortize software overhead. The maximum packet size at 400 Mbps, 2048 bytes, can carry data for hundreds of channels. Isochronous packets could be used to send real-time feedback data without the overhead of asynchronous read requests.

The low-level control architecture presented here can pave the way for medical robots of greater agility, allowing more robot-assisted surgical tasks to be conceived. For the Snake Robot, an additional arm can be more conveniently integrated and used for tasks such as camera placement or tissue manipulation. New applications being considered include dexterous ultrasound imaging and ablation. The software interface will be compatible with a standard medical robotics framework, the Surgical Assistant Workstation [5].

Acknowledgment

We thank Hamid Wasti of Regan Designs, Inc. (Coeur d'Alene, Idaho) and Fran Wu for their extensive help with the design and layout of the controller boards, and Mitch Williams for his help with software setup.

References

[1] Simaan, N., R. Taylor, and P. Flint, "A dexterous system for laryngeal surgery," *IEEE Robotics & Automation*, vol. 1, pp. 351-357, Apr 2004.

[2] Kapoor, A., N. Simaan, and P. Kazanzides, "A system for speed and torque control of DC motors with application to small snake robots," *IEEE Mechatronics and Robotics*, Aachen, Germany, Sep 2004.

[3] Kapoor, A., "Motion constrained control of robots for dexterous surgical tasks," Ph.D. dissertation, Johns Hopkins Univ., Sep 2007.

[4] Simaan, N., R. Taylor, and P. Flint, "High dexterity snake-like robotic slaves for minimally invasive telesurgery of the upper airway," *MICCAI*, Rennes-Saint-Malo, France, Sep 2004.

[5] Vagvolgyi, B., S. DiMaio, A. Deguet, P. Kazanzides, R. Kumar, C. Hasser, R. Taylor, "The Surgical Assistant Workstation," *MICCAI Workshop on Systems and Arch. for Computer Assisted Interventions* (online at http://hdl.handle.net/10380/1466), Sep 2008.

[6] Sarker, M., C. Kim, S. Baek, and B. You, "An IEEE-1394 based real-time robot control system for efficient controlling of humanoids," *IEEE Intelligent Robots and Systems*, Beijing, China, Oct 2006.

[7] Baltazar, G. and G. Chapelle, "Firewire in modern integrated military avionics," *IEEE Aerospace and Electronic Systems Magazine*, vol. 16, no. 11, pp.12-16, Nov 2001.

[8] Robertz, S., K. Nilsson, R. Henriksson, and A. Blomdell, "Industrial robot motion control with real-time Java and EtherCAT," *IEEE Emerging Technologies & Factory Automation*, pp. 1453-1456, 2007.

[9] Lin, S., C. Ho, and Y. Tzou, "Distributed motion control using real-time network communication techniques," *International Power Electronics and Motion Control*, vol. 2, pp. 843-847, Aug 2000.

[10] Walls, B., M. McClelland, S. Persyn, and D. Werner, "Leveraging flight heritage to new CompactPCI space systems: a fusion of architectures," *Digital Avionics Systems*, vol. 2, pp. 8C41-8C47, 2001.

[11] Szydlowski, C., "Implementing PCI Express for industrial control," *RTC Magazine*, vol. 13, Sep 2004.

[12] Zhang, Y., B. Orlic, P. Visser, and J. Broenink, "Hard real-time networking on FireWire," *RT Linux Workshop*, Lille, FR, Nov 2005.

[13] Sarker, M., C. Kim, J. Cho, B. You, "Development of a network-based real-time robot control system over IEEE 1394: using open source software platform," *IEEE Mechatronics*, pp. 563-568, July 2006.

[14] Schneider, S., "Making Ethernet work in real time," *Sensors Magazine*, vol. 17, no. 11, Nov 2000.

[15] Kerkes, J., "Real-time Ethernet," *Embedded Systems Design*, vol. 14, no. 1, Jan 2001.