

# High-Speed Communication for Fault-Tolerant Systems

**Mark Jay Prizant**

© 1998 IEEE. Reprinted, with permission, from the Proceedings of the Digital Avionics System Conference (DASC), Bellevue, WA, October 31-November 6, 1998

X-38: Photo Credit - NASA

## ABSTRACT

Critical missions require ultra-high reliability systems. Byzantine-resilient quad-redundant fault-tolerant computers are one of the most rigorously proven solutions to this ultra-high reliability problem. In the past, custom-built computers in which the redundant computers are synchronized in lock step were built to provide the solution.

Custom computer architectures are no longer desired due to their expense, lack of commercial-off-the-shelf (COTS) processor boards and backplanes, and lack of easy technology insertion (such as upgrading to faster processors). The communication between redundant channels is also a bottleneck, making the performance penalty for the added fault tolerance very high.

This paper presents the design for a communicator that allows four COTS single-board computers, each in a standard backplane such as VMEbus or CompactPCI, to be networked together to facilitate a quad-redundant Byzantine-resilient fault-tolerant computer. The communicator features very high bandwidth, high utilization, and low latency, resulting in minimal performance penalty for the added rigorous fault tolerance. The communicator uses off-the-shelf gigabit link modules (designed for use in the fiber channel physical layer) for high-bandwidth synchronous interchannel data links. The data links are used to implement a time-division-multiplexed two-round data exchange, providing the fundamental source congruency operation required for Byzantine resilience.

## INTRODUCTION

Critical missions require ultra-high reliability systems. Quad-redundant fault-tolerant computers are a rigorously proven solution to this ultra-high reliability requirement. In the past, custom-built computers, in which the redundant computers are synchronized in lock step, were built to provide the solution. The quad-redundant ship control computer used in the Seawolf Submarine (Ref. [1]) is an example of such a system delivered within the last five years. This fault-tolerant computer, designed by Draper Laboratory, uses custom-designed modules (68020-based) in a nonstandard backplane. Interchannel communication is facilitated with byte-wide discrete input/output (I/O) at 1 Mbyte/s.

Custom computer architectures are increasingly undesirable due to their expense, lack of COTS processor boards and backplanes, and lack of easy technology insertion (such as upgrading to faster processors).

In any fault-tolerant computer, the communication between redundant channels necessary to achieve rigorous fault tolerance is a bottleneck, imposing a severe performance penalty for the added fault tolerance.

This paper presents the design for a communicator that allows four COTS single-board computers, each in a standard backplane such as VMEbus or CompactPCI, to be networked together to facilitate a quad-redundant fault-tolerant computer. The communicator features very high bandwidth, high utilization, and low latency, resulting in minimal performance penalty for interprocessor communication. The communicator uses off-the-shelf gigabit link modules (GLMs) (designed for use in the fiber channel physical layer) as high-bandwidth interchannel data links. The data links are used to implement a time-division-multiplexed two-round data exchange, providing the fundamental source congruency operation required for rigorous fault tolerance (Ref. [2]). The communicator is called a "carousel" because its message scheduling can be thought of as messages moving in and out of the slots of a rotating slide projector carousel.

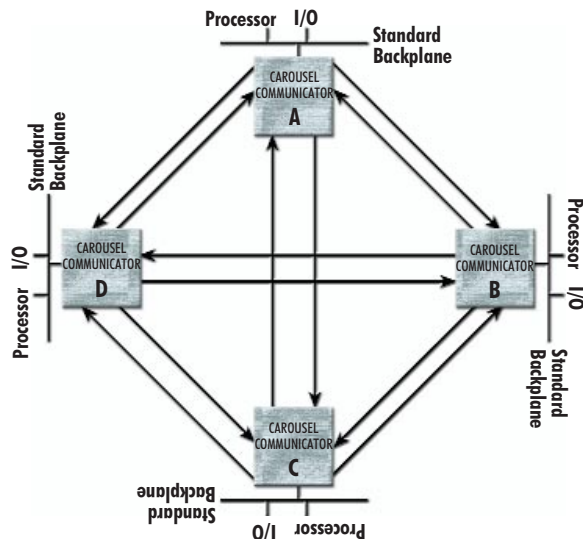
## CAROUSEL ARCHITECTURE

The Carousel Communicator is an evolutionary step with its precursor being the fault-tolerant parallel processor (FTPP) developed by Draper (Ref. [3]). Currently, the FTPP architecture is being designed into the flight-critical computer of NASA's X38 vehicle by Draper. The section on *Upper Layers* will discuss some differences between the carousel architecture and the FTPP communicator.

## Topology

The carousel architecture is based on a quad-redundant computer in which the computing channels are connected via a mesh network. Figure 1 shows the mesh network interconnecting the four channels. The four communicators are fully connected and operate in tight synchrony to perform message exchanges. The interchannel links are fiber-optic links operating at 1.06 Gb/s. Each channel

contains a standard backplane (such as VME), one or more processor boards in each channel, I/O boards as needed, and the carousel communicator. The as-built prototype supports one processor per channel. The section on *Upper Layers* discusses enhancements that could support multiprocessing, allowing more than one processor per channel. All boards are COTS, except the communicator board. The GLMs on each communicator are COTS.



**Figure 1**  
Mesh network

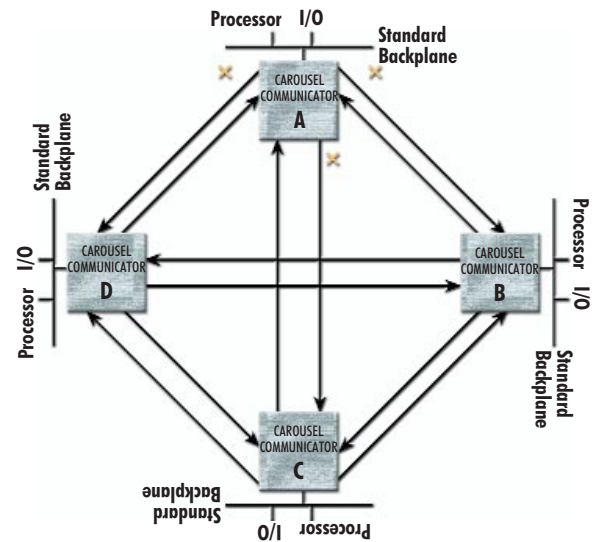
The carousel communicator is an ultra-high-speed data pipe engine distributing data from one channel to all the others, implementing a reliable fault-tolerant mechanism known as a two-round exchange. The processors in each channel are executing the same program. In order for the redundant computers to work together as a system and not diverge, all channels must receive and process the same data sets. These data may represent sensor data or the result of calculations. If these data were distributed on a bus, such as Ethernet, or with a simple point-to-point flow, then a transient error or degradation could cause one channel to receive or interpret a piece of data that is different from the other channels. Even if a cyclic redundancy code (CRC) were inserted to detect an error, a retry protocol would have to be added, which would reduce throughput and add complexity. One application requiring a very-high-bandwidth data exchange might be memory realignment to reintegrate a channel back into a quad that has had a transient failure, or to integrate a cold spare. Another application requiring a high-speed data exchange might require continuous high data rates, such as digitized video. The carousel communicator allows each channel to have a communication bandwidth approaching 1/4 (for a quad-redundant system) of the total link data rate.

### Two-Round Messages

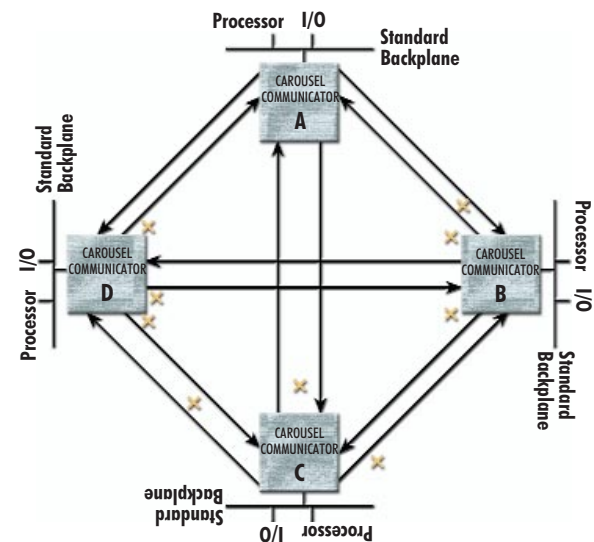
Two-round messages, also known as source congruency messages, can contain channel-specific information, such as the value of a processor clock. The communicator

performs two rounds of exchange of this message, allowing each channel to compare its own copy of the message to that received by the other channels. Figures 2 and 3 show a two-round exchange.

Figure 2 shows that in the first round, channel A transmits its single-source data (x) on all its links to the other three channels. On the second round, each channel, except the original channel that sourced the message, reflects the message out on the links. This is shown in Figure 3. The received second round is then voted at each channel. As shown in the diagram, channel D, for example, receives reflections from channels C and B. The three pieces of data that are voted in channel D are the reflections from B and C along with the saved value that D originally received from A.



**Figure 2**  
First round of two-round exchange



**Figure 3**  
Second round of two-round exchange

These exchanges and votes ensure that all channels will receive and interpret the same data, even if any one link is broken.

## Data Flow

The data exchange is composed entirely of source-congruent two-round exchanges. At each clock tick, a 32-bit word is exchanged on every link. The clock ticks are broken up into repeating 4-tick frames. Each frame consists of an A-tick, a B-tick, a C-tick, and a D-tick. Each channel is only allowed to send a word during its own tick. This is true whether the word is being sent in the first round or as a reflection in the second round. This means that all data occupying an A-tick, for example, were sourced by channel A, either representing the first round being sent by channel A or reflecting the data that originally came from channel A.

Four redundant channels, A, B, C, and D, are shown in Figure 1, along with the communications links between them. Table 1 shows the data flow for nine successive ticks. In Table 1, there is a row for every link. For example, AB means the link that sends data from A to B. The columns from left to right represent time, with each column representing a time tick in which a word is exchanged. The data contents are in the form of a letter and a number, i.e., A0. The letter represents the channel that sourced the word, and the number is the sequential word number. For example, word 1 from channel A is sent from channel A to all the other channels during the leftmost tick in the table. This is the first round of the exchange. Four ticks later,\* this word is reflected from channels B, C, and D. This is the second round of the exchange. During the second round, channel A sends out its next word, word 2. The data exchanges in the table are continuous and are independent of data exchange requests by the processors in each channel. If a processor wants to source a message, then it will write the word to its communicator, and the communicator will put the data out onto the links when its slot is next available in the time-division-multiplexed pattern shown.

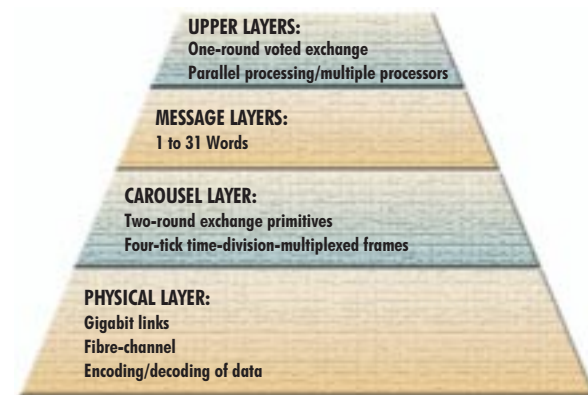
**Table 1**  
Data flow

LINK	TICK	TICK	TICK	TICK	TICK	TICK	TICK	TICK	TICK
AB	A1	B0	C0	D0	A2	B1	C1	D1	A3
AC	A1	B0	C0	D0	A2	B1	C1	D1	A3
AD	A1	B0	C0	D0	A2	B1	C1	D1	A3
BA	A0	B1	C0	D0	A1	B2	C1	D1	A2
BC	A0	B1	C0	D0	A1	B2	C1	D1	A2
BD	A0	B1	C0	D0	A1	B2	C1	D1	A2
CA	A0	B0	C1	D0	A1	B1	C2	D1	A2
CB	A0	B0	C1	D0	A1	B1	C2	D1	A2
CD	A0	B0	C1	D0	A1	B1	C2	D1	A2
DA	A0	B0	C0	D1	A1	B1	C1	D2	A2
DB	A0	B0	C0	D1	A1	B1	C1	D2	A2
DC	A0	B0	C0	D1	A1	B1	C1	D2	A2

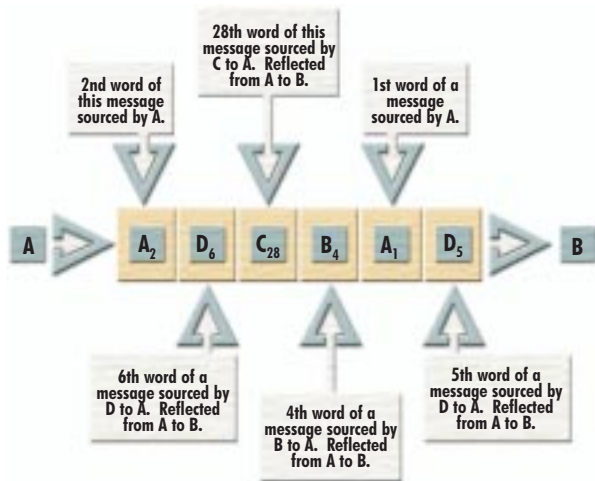
\* Actually, the number of ticks later that the word is reflected is 4n, where n is around 4. This is due to the pipelining of the ENDEC and the logic.

We can think of this time-division-multiplexed data flow just described as the second lowest layer of a layered model (see Figure 4), analogous but not identical to the open-system interconnect reference model (ISO) seven-layer model. The bottom layer is the physical access layer. This includes the electrical characteristics of the links, and the 8b/10b encoding and decoding of data. This is described further in the *Link Discussion* section. The second lowest layer, which can be called the carousel layer, comprises the two-round exchange primitives within the 4-tick time-division-multiplexed data flow described. The carousel communicator that is currently built at Draper implements the bottom three layers. The third-lowest layer, called the message layer, groups words into messages from 1 to 31 words long. Each word is sent through a two-round exchange by the carousel layer, and then the message layer assembles the received words into messages and puts each assembled message into dual-port RAM (DPRAM) to be read in VMESpace by the processor. The message layer also puts an info word into each message buffer. This info word contains the word count for the message, user fields that the processor can use for control or message number info, and vote error information called syndromes. Figure 5 shows a possible snapshot of a data flow pattern on a link. Each channel can source a 32-bit word every 4 ticks, giving it 1/4 of the total link bandwidth. A different channel can source a message every tick, making very efficient use of the network. The last word of every message is a control word signifying to the message layer that the end of the message has arrived. This word is called the "footer," and can also contain control information.

The voted assembled messages are placed into one of four input buffer areas in the memory map of the processor. Each buffer represents one of four possible sources for the messages. These buffers are in a relative format, and are "myself," "left," "middle," and "right." The "myself" buffer contains messages that originated in this channel. The "left" buffer contains messages that originated in the channel to the left of this one, etc. For each of these buffer



**Figure 4**  
Time-division-multiplexed data flow in a layered model



**Figure 5**  
Link snapshot

types, the as-built carousel communicator has a ring of 32 input buffers, allowing 32 input messages to arrive before the processor reads them. If the ring is full and a new message arrives, then the new message is lost and an overflow bit is set.

### Clock Synchronization

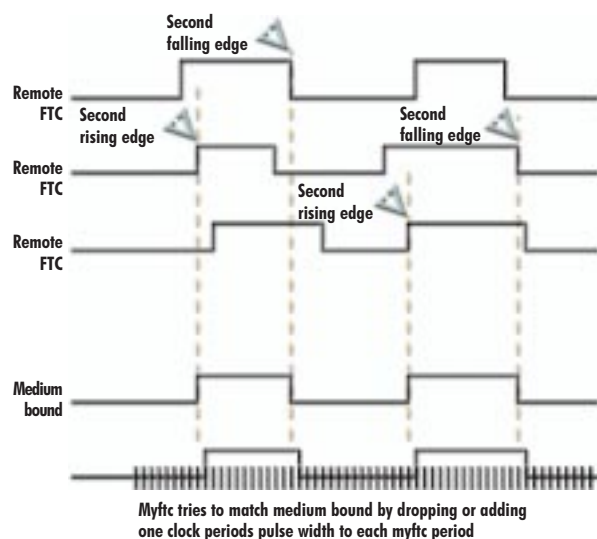
The COTS processors in each channel are not clock synchronized; however, the four communicators must be clock synchronized in order to ensure that the data word sets being voted together arrive within a bounded skew (with about one tick allowed for skew). For every 26-MHz clock tick, new 32-bit data words arrive at the communicators' three input channels. At every new tick, these data represent a different channel's sourced data and must be routed to the appropriate voter. The system complexity would blow up quickly if data had to be stored waiting for the appropriate fellow words of data with the same tick and the same message word count number to arrive. If the reflected data words are to be sent on an A-tick, then each channel's A-tick must occur at the same time. If not, then the received two-round words to be voted would not arrive together. A common clock cannot be routed to all the communicators, as that would violate fault containment, another requirement for rigorous fault tolerance. Fault containment ensures that a single fault does not propagate to more than one channel. This requirement leads to each channel having separate power supplies and electrical isolation. The solution to this problem is as follows: each channel has its own oscillator and generates its own copy of a fault-tolerant clock (FTC), which is a pulse that occurs every 64 frames. Remember that a frame consists of a 4-tick A, B, C, D sequence. After power-up, each channel's FTC is at an arbitrary phase with respect to each other. For the data exchange engine to work correctly, the FTC edges in each channel must line up with each other within a bounded skew. In addition, once they do line up, there must be some adjustment mechanism to compensate for the drift of each channel's oscillator with respect to each other. The mechanism for synchronizing FTC edges to

each other involves each channel receiving the FTC clocks from the other channels and voting the edges with an algorithm that determines whether its own FTC period needs to slow down or speed up. The FTC rising edge is embedded as a special command word in the data stream. Each receiver reconstructs the incoming FTC pulses from the other channels on the existing links. Figure 6 shows a data stream with the FTC clock embedded. The adjustment that a channel uses to move its FTC clock involves deciding whether to have 0, 1, or 2 optional clock idle ticks during the adjustment phase. The FTC clock voting algorithm involves detecting the second incoming rising and falling edges and creating a medium bound signal to which the channel tries align itself by adjusting its own FTC (myftc) one clock cycle at a time. Although the algorithm seems complicated, it is a better solution than using dedicated links for the FTCs, which double the number of fiber links. Figure 7 shows a representation of the FTC algorithm.

The FTC rising edge is also used as a reference to determine whether an incoming data word is an A-tick, B-tick, C-tick, or D-tick word, and to which frame number the word belongs. The first tick after the FTC edge is always an A-tick of frame number 0. Frame numbers are used to determine when to reflect a given word to ensure that the reflections to be voted will all arrive at a channel together.



**Figure 6**  
Data stream



**Figure 7**  
Fault-tolerant clock voting/synchronization

## Link Discussion

The interchannel links use fiber-optic GLMs. These modules conform to GLM standard FCSI-301-Revision 1.0. They provide serial transmission at 1.06 Gb/s. These modules are normally used as the physical layer devices for the fiber channel. A separate IC, called an ENDEC, is used to provide 8b/10b optical encoding and decoding. The encoding translates each byte into a 10-bit pattern. The 10-bit pattern always has multiple data transitions and ensures clock recovery at the receiver by providing dc balance. The ENDEC also can encode various command words into the data stream. This is possible because codes not needed for data representation are available as control/command characters. These commands are used to embed an FTC that allows all channels to vote clock edges and synchronize to a common clock, even though all channels have independent oscillators. Idle words, when there is no data, and clock idle words, used for FTC adjustment, are also embedded as fiber channel-like command words. Command characters are also used in word framing. The full fiber channel FC-1 layer protocol is not used. The lower section of the fiber channel FC-1 layer that is being used is the 8b/10b encoding, and the control/command character usage. The upper part of the fiber channel layer used for link management and message framing is not used, but is replaced with the protocol described.

## Physical Implementation

Figure 8 shows a representation of the 6U VME board implementation. At the front are the three GLMs. These provide the fiber-optic links to the other three channels. In a relative representation, these are referred to as the *right*, *middle*, and *left* links. Each GLM has an ENDEC associated with it. A DPRAM is shown that provides the buffer storage. There is a transmit buffer and the four receive buffers described above in the *Data Flow* section. To give the reader an idea of the amount of logic required, the field programmable gate array (FPGA) used is an Altera 10k100 SRAM-based device with 100,000 equivalent gates. About 75% of the device is used.

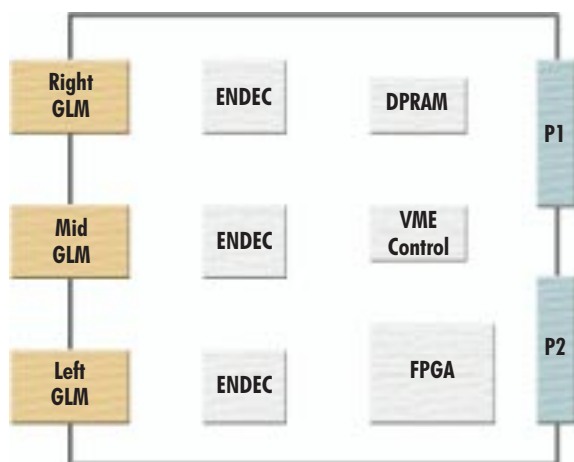


Figure 8

6U VME board

## UPPER LAYERS

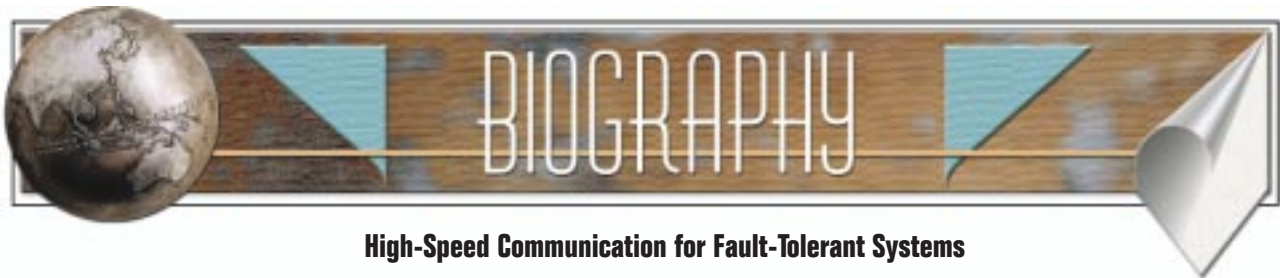
The carousel communicator as currently constituted implements the bottom three layers shown in Figure 4. A layer above these three would provide various services. One service could be a voted one-round exchange. If all redundant members of the quad have identical data that they want to compare, they could exchange and vote the data in one round of data exchange if this were supported. The Seawolf ship control computer and the FTTP have a one-round voted exchange primitive. The carousel communicator does not have a one-round exchange primitive, at least in its lowest three layers. This capability was given up in order to allow the high network use and low overhead of the elegant and simple carousel scheduling described. In the FTTP, additional overhead is incurred in order to accommodate one-round exchanges in addition to two-round exchanges. Each data exchange round must be preceded by a round of exchanges for the purpose of all communicators reaching consensus on what type of exchange is to be performed next. In the carousel, there is no need for such a consensus, since only two-round exchanges are used and the A-, B-, C-, and D-tick assignments provide consensus. However, a one-round voted exchange is possible with the carousel by implementing it with a layer-four function. This could be accomplished by having each channel source a two-round exchange. The four copies of the data resulting from the two-round exchanges can then be voted in software or by an additional hardware layer added to the communicator for this purpose. Another possible upper-layer function might be providing communication between multiple computers, with each computer being of arbitrary redundancy. There would still be four communicators in a mesh configuration, but there could be multiple processor boards in each channel on each backplane. This is the configuration of the Draper FTTP. For a carousel implementation, the footer at the end of each message would contain fields specifying which processor sourced the message. The "which processor" specified here would not specify which processor in a redundant group, but rather, on an orthogonal axis, which parallel computer, which in itself can be composed of redundant processors. The control fields do not have to specify which member or channel of a redundant group sourced the message because the receiver knows the tick that the message arrived in, which specifies the channel that sourced the message.

## CONCLUSION

A layered architecture was presented for a communicator that allows four COTS processor boards to be networked together to produce a quad-redundant fault-tolerant computer. The communicator features a high-bandwidth data exchange, making data reliably congruent across the four channels. Upper protocol layers can be added to the communicator hardware, reducing the software overhead, which would be assessed by performing these functions in software.

## REFERENCES

- [1] "Seawolf Submarine Ship Control System: A Case Study of a Fault-Tolerant Design," *Naval Engineers Journal*, January 1994.
- [2] "Reaching Agreement in the Presence of Faults," *Journal of the Association for Computing Machinery*, Vol. 27, No. 2, April 1980, pp. 228-234.
- [3] Harper, R.E. and J.H. Lala, "Fault-Tolerant Parallel Processor," *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 3, 1991, pp. 554-563.



## BIOGRAPHY

### High-Speed Communication for Fault-Tolerant Systems



Jay Prizant has worked at Draper since 1980. He is a Digital Hardware designer who has worked in the fault-tolerant area for a number of years. Jay was a principal designer of the fault-tolerant computer used as the Ship Control Computer on the Seawolf Submarine and is currently the hardware designer for a communicator used in the fault-tolerant mission control computer on NASA's X38 re-entry vehicle prototype. He graduated from Cornell University with a BS in electrical engineering in 1978.

[prizant@draper.com](mailto:prizant@draper.com)