Data System with Distributed Processing for a Next Generation Satellite

Cliffton M. Tsai* and Manorama Raghavender†
TRW Space & Technology Group, Redondo Beach, California

This paper describes a data system with distributed processing for a next generation satellite system that is at a conceptual design stage. It is a constellation of identical observation satellites with payloads that produce a high amount of sensor data, as well as housekeeping data. The early design had two data buses driven by two different types of data—command, control, and telemetry (C^2 T) data, and burst-type sensor data. The C^2T messages led to a low-speed 1Mbps C^2T bus; the sensor data characterized by long-length messages led to an asynchronous 50 Mbps high-speed data bus. The architecture of the buses and the computer simulation of the traffic flow on these buses is described. To support real-time operations of the spacecraft, the data system is required to support the peak load of message flows. The key factors are message frequencies, priorities, lengths, and data latency. To verify the performance, computer simulations were performed, based on the expected traffic and the selected bus protocols for the buses. The results show that the design provides an ample margin in bus bandwidth for each bus. Further simulations were then conducted to investigate if a single data bus could accommodate all the messages and meet the latency requirement, and if so, to determine the optimum data rate on the combined bus. The results indicate that the two buses can be combined into one bus and the data rate optimized, in spite of their opposite characteristics, and still provide an adequate margin for performance. We also found that the data rate can be optimized for this application.

Introduction

COMPUTERS in satellites have come a long way since the early days of the space age. We have seen evolutions from analog processing to combination logic to simple microprocessor/computer. Many of the current satellites on-orbit contain simple computers. For example, the Tracking and Data Relay Satellite (TDRS) has a custom 8-bit microprocessor (two 2901 bit-slice chips) with 4 K-words of memory. The Landsat satellite has an 18-bit computer called the NASA Standard Spaceborne Computer-1 (NSSC-1) with 64 K-words of memory. The High-Energy Astronomical Observatory (HEAO) satellite has a 16-bit computer with 32 K-words of memory called the CDC 469. Future satellites will have greater processing power, e.g., the Hubble Space Telescope will have a Litton 4516E on-board computer (OBC), as well as General Electric's IDF 224. Currently being developed is a military satellite that will have two Air Force standard 16-bit computers (1750A) made by Teledyne, with 256 K-words of memory each. With the advent of very high-speed integrated circuit (VHSIC) technology, the computer size, weight, and power have finally reached a point where distributed processing in space using high-speed interprocessor communication is achievable. Increased processing implemented on board will result in more decisions being made on board, providing an autonomous spacecraft.

The Satellite Design

For the purpose of this paper, spacecraft refers to the "bus" portion of the spacecraft, and the satellite refers to the spacecraft when integrated with the payloads. The satellite design uses a modular concept in which functional entities are made into modules that can be individually tested. Since each module can be tested independent of other modules, it makes satellite integration shorter and easier and lowers the integration cost and risk. The satellite has an advanced data system in which each module has its own processor and associated memory. The design is the result of numerous tradeoffs in data rate, technology, bus architecture, protocol, and reliability. The data rate is important in space applications because it has an impact on the power requirements and drives the technology for implementing the bus interface units. The independently functioning modules perform their own housekeeping and reconfiguration in the event of faults, thus providing a fault-tolerant system. The satellite system has a network of identical satellites. Data from on-board payloads is exchanged among the constellation satellites at high speed.

The design contains ten spacecraft modules and four payload modules, as listed below.

1) Command and Data Handling (CADH) module for handling command, telemetry and recovery from upset on the spacecraft.

2) Propulsion Module (PRP) that contains all the thrusters and the Reaction Control Subsystem (RCS).

3) Attitude Control Module (ACM) that is responsible for maintaining the spacecraft attitude; contains all the Earth and sun sensors and the star tracker.

4) Common Perigee Module (CP), to place the satellite in its designed orbit.

5) Power Module (POW) that contains the solar array, the solar array drive, and the batteries.

6) Up/downlink Module (U/DL) that has the communication equipment to maintain the RF up/downlink.

7) Communication Manager Module (CMM) that is the communication message traffic controller in charge of forwarding messages in the network.

8) Crosslink (XL) one, two, and three modules that contain communication equipment to maintain crosslink communication between satellites.

Received Sept. 11, 1987, presented as Paper 87-2819 at the AIAA Computers in Aerospace VI Conference, Wakefield, MA, Oct. 7–9, 1987; revision received Aug. 2, 1988.

*Subproject Manager, Tracking, Telemetry, and Command Subsystem, Engineering and Test Division. Member AIAA.

†Staff Engineer, Telecommunication and Data Systems Laboratory, Engineering and Test Division. Member AIAA.

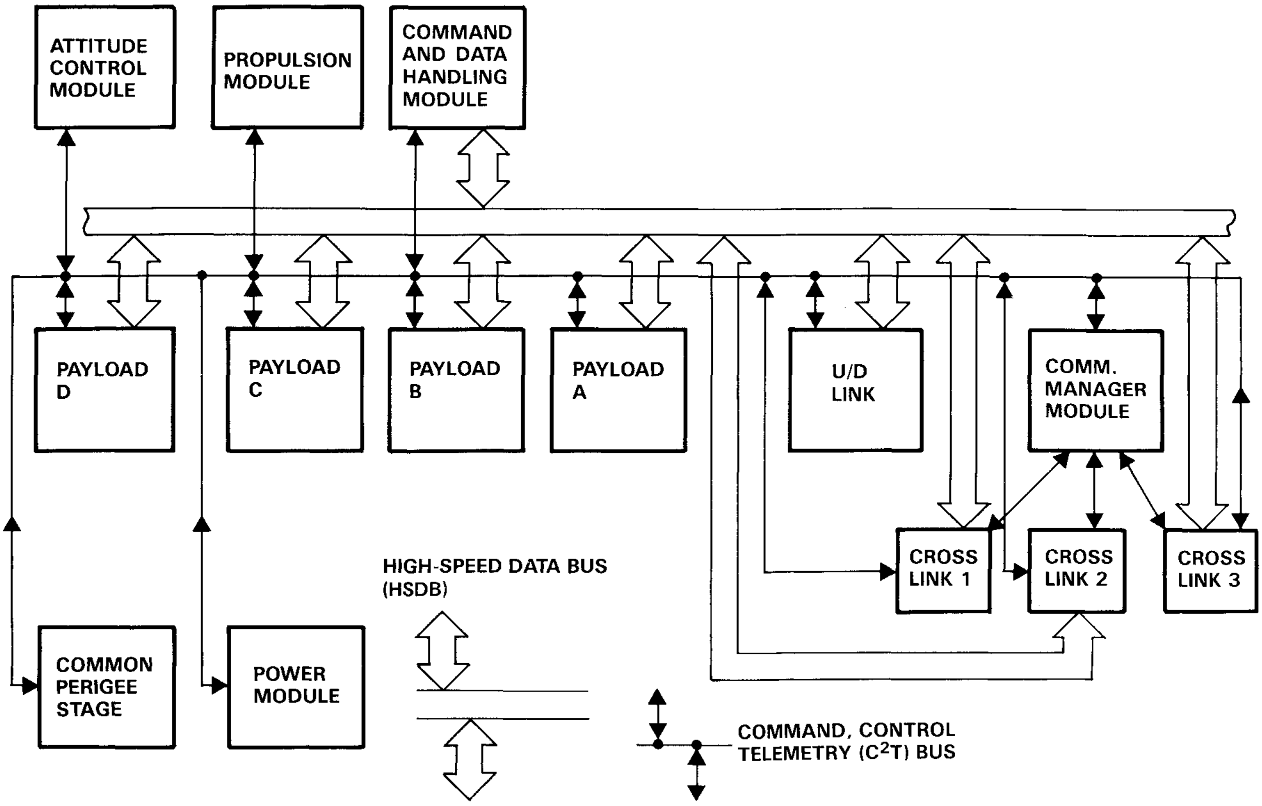

Fig. 1 Satellite block diagram showing interconnection among modules.

9) Payloads (PL) one, two, three, and four for observation purposes.

The satellite block diagram is shown in Fig. 1. Each module has its own computer and associated memory and typically functions independently of other modules. However, real-time spacecraft operations necessitate data exchanges between modules. For example, the ACM issues thruster commands to the PRP module and needs sun sensor data from the POW module where the sun sensor is mounted, and for control loop pre-emphasis purposes, the ACM needs solar array movement command information. The modules generate health and status messages that are transmitted to the CADH module for packetization. The payload modules generate a great amount of sensor data for other payload modules within the satellite and also for payloads of other satellites. The two types of data generated by the modules are quite opposite in characteristics. The C^2T data has the following attributes: short messages, mixed priorities, low repetition rate, short and fixed message latency, and mostly scheduled events.

The sensor data has opposite characteristics: variable message lengths, high priority, high burst rates and low average rates, variable (within bound) message latency requirements, and mostly unscheduled events.

The early conceptual design of the satellite involved two distinct data buses. The bus used for communicating the C^2T data was called the C^2T bus and is connected to all 14 modules. The bus for sensor data was called a HSDB and was connected to all the payload modules and a communication manager module on the spacecraft. To properly size the data buses, we analyzed the traffic expected on these buses. The satellite design being mature enough, the message lengths and frequencies could be reasonably estimated or calculated. We used four levels of priority, and individual messages were assigned to a priority level according to their importance to the mission. The estimated bus traffic is tabulated in Tables 1 and 2 for the C^2T bus and the HSDB, respectively. To support spacecraft real-time operations, e.g., when the control loops are closed between more than two modules, the data system is required to support the peak load of messages. Hence, from the system point of view, the tolerable message delay for each priority group is required to be within certain bounds. For priority 1 (highest priority), the maximum delay (or latency) was calculated to be 10 ms.

Table 1 Traffic on C^2T bus

Source terminal	Message size, words	Message period, s	Number of inputs	Message priority[a]
CADHM	4, 10, 1	0.1, 10, 10	1, 1, 20	1, 1, 1
CADHM	168, 5, 25	1, 10, 10	1, 1, 1	2, 2, 2
CADHM	1, 12	1, 0.1	1, 1	3, 3
CMM	4, 12, 10	1, 0.1, 1	1, 1, 1	1, 3, 4
POW	2, 4	10, 1	1, 1	1, 1
POW	12, 3	0.1, 1	1, 1	3, 4
CP	12	0.1	1	2
PRP	4, 1, 12	1, 1, 0.1	1, 1, 1	1, 4, 3
ACM	6, 4, 4	0.02, 1, 1	1, 1, 1	1, 1, 1
ACM	25, 12	1, 0.1	1, 1	4, 3
P1	4, 12, 25	1, 0.1, 1	1, 1, 1	1, 3, 4
P2	5, 4	10, 1	1, 1	1, 1
P2	12, 5	0.1, 1	1, 1	3, 4
P3	8, 12	1, 0.1	1, 1	4, 3
P4	4, 12, 25	1, 0.1, 1	1, 1, 1	1, 3, 4
UDL	4, 4	1, 1	1, 1	1, 1
UDL	5, 12	1, 0.1	1, 1	4, 3
XL1	113, 4	1, 1	1, 1	1, 1
XL1	5, 12, 2	1, 0.1, 1	1, 1, 1	4, 3, 2
XL2	4, 2	1, 1	1, 1	1, 1
XL2	5, 12	1, 0.1	1, 1	4, 3
XL3	4, 4, 12	1, 1, 0.1	1, 1, 1	1, 4, 3

[a]Priority 1 = the highest priority; priority 4 = the lowest priority.

As Table 1 illustrates, every module is expected to produce messages. These include status messages and, during fail-over

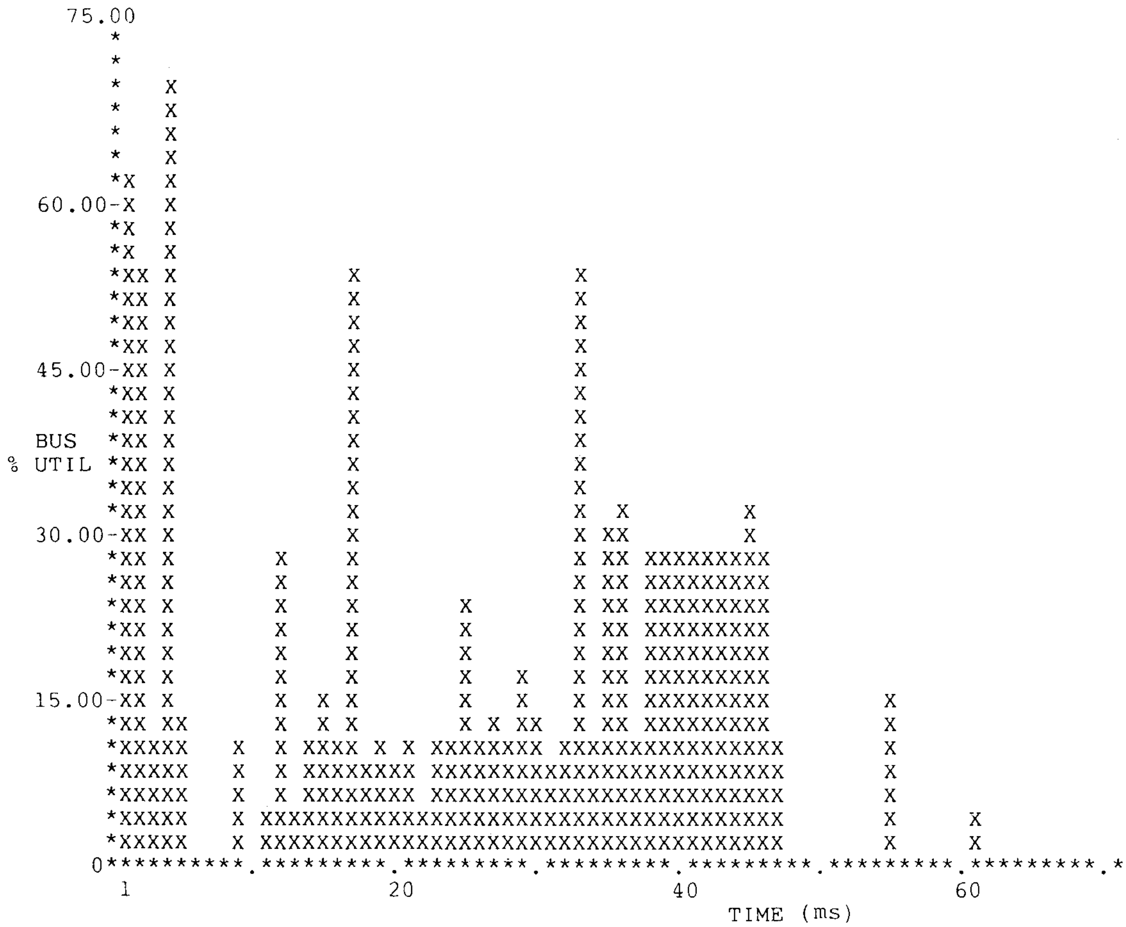

Fig. 2 Bus use vs time over a simulation run of 60 ms.

Table 2 Traffic on HSDB

Source terminal	Message size, words	Message period, s	Number of inputs	Message priority[a]
CADHM	125, 63	1, 1	24, 60	2, 2
P1	1000	9	16	2
P1	1000, 500, 1000	3, 3, 60	25, 44, 18	1, 1, 1
P1	1000, 78, 20	3, 5, 5	20, 8, 1	1, 1, 1
P2	31, 13, 6	0.5, 10, 10	20, 20, 1	1, 1, 3
P2	25, 31, 6	1, 10, 10	1, 1, 1	1, 1, 2
P3	4	300	20	1
P4	500, 500, 1000	3, 9, 120	28, 32, 8	1, 1, 1
P4	250, 500	9, 3	36, 28	1, 1
XL1	155, 12, 1000	0.5, 300, 9	4, 1, 19	1, 1, 1
XL1	2000, 1000, 42	9, 3, 10	24, 40, 1	3, 2, 3
XL1	2000, 8, 75	60, 10, 1	27, 1, 1	2, 3, 2
XL1	1000, 93	3, 5	14, 1	2, 2
XL2	1000, 1000, 6	9, 60, 10	36, 36, 2	3, 2, 3
XL2	25, 31, 31	1, 5, 0.5	2, 2, 50	1, 1, 1
XL2	4, 250, 500	300, 9, 3	2, 12, 28	1, 1, 2
XL3	125, 125	9, 60	24, 24	1, 1
U/DL	31, 1	10, 60	20, 2	1, 2

[a]Priority 1 = the highest priority; priority 4 = the lowest priority.

conditions, reconfiguration messages. As can be seen, all C^2T messages, excepting one, have only one input over the message period indicated. The C^2T messages are generally shorter in length and higher in frequency relative to the HSDB messages. These messages occupy all priority groups but have the highest concentration in priority group 1 (77%). This is imposed by the real-time operation requirement for the spacecraft. In this context, the term "message" includes any commands from either the ground or any module. For instance, because of the large mass of the solar arrays (considered part of the power module), the ACM module needs feed forward of the array move commands originating in the power module for applying pre-emphasis in the control loops. Such messages need the highest priority to bound the delay as required by the control loop. Commands to fire the thrusters (a part of the propulsion module) as actuation of the control algorithm are other examples of the highest priority messages. The lowest priority messages are dominated by functions such as self-test and status reporting. The middle priorities are assigned to the remaining messages. The HSDB messages are, however, dominated by burst-type messages of long lengths, as depicted in Table 2. They have low-average frequency compared to the C^2T messages. The most frequent messages in the HSDB occur every ½ s, compared with every 16 ms in the C^2T bus. Notice that many C^2T messages appear every 3, 6, and 9 s. These messages result in peak loads on the data bus, as seen in the simulation results described in the simulation section.

Data Bus Design

The early design used two separate data buses. The bus selections were based on the projected traffic on each bus shown in Tables 1 and 2. In satellite applications, it is important to consider the space environment, radiation requirements, technology maturity, hardware availability, size, weight, power, and so forth. For the HSDB, preliminary calculations show that a 10–50 Mbps data bus is needed to meet the requirements. However, the current generation spacecraft buses do not provide such data rates. The state of the art for the data rate is in the 1–3 Mbps range. In the avionics world, however, an HSDB is being developed that meets our needs. It is the HSDB being developed for the Air Force under the Pave Pillar Program. This program is designed to produce a high-speed serial data bus to accommodate more stringent requirements in data transfer between the avionic subsystems, better and more flexible resource sharing, and finally, improved fault tolerance. It is designed as the next generation MIL-STD-1553B data bus. Fortunately, the same attributes are important to next generation spacecraft data system as well. The requirements driving the Pave Pillar HSDB design are as follows: 1) 20 Mbps throughput rate; 2) Up to 64 terminals; 3) Up to 300 ft terminal separation; 4) One to 4096 word message length; 5) 16-bit words; and 6) Compatibility with both fiber-optic and wire media.

At 50 Mbps, the Pave Pillar HSDB under development far exceeds our requirement. Therefore, the Pave Pillar HSDB is selected as the baseline for the HSDB. Ludvigson and Milton[1] describe this bus in detail. We recognize that the hardware to be produced for Pave Pillar application will not be suitable for space environment. Furthermore, in the avionics world, minimizing power requirement on the Bus Interface Units (BIU) is not as critical as it is for space applications. Weight and size of the hardware produced on the Pave Pillar program may be acceptable for our use, because of their use of VHSIC and the VLSI technologies necessitated by the data rate requirement. Hence, sufficient heritage can be derived from the Pave Pillar HSDB development, such that we can use their bus with appropriate modifications. For example, the radiation requirement can be met by transposing the design to the standard 1 μ rad-hard CMOS technology used in this satellite design. In any case, short of any other high-speed bus under development, this provides the best hardware to meet our requirements. The characteristics of the HSDB are summarized in the simulation section.

For the C^2T bus, there are more choices. Four well-known spacecraft data buses are studied to determine the best candidate to meet our requirements. These are SCI's Data Acquisition and Control System (DACS) bus used in Talon Gold and MX; Fairchild Space Company's Standard Telemetry and Command Component (STACC) bus used in Landsat and the Solar Max Mission; the military avionics MIL-STD-1553B bus, used primarily on airplanes; and Gulton Data System's Telemetry, Timing, Command, and Control (T^2C^2) bus, which is based on GPS and is currently under development. The features of these buses and the remote interface units are tabulated in Table 3. For the C^2T bus, we select the 1553B bus. The characteristic of the bus, whereby bus control can be transferred between the modules, supports the modular design concept better than the other buses; all other buses will require substantial modification to achieve the same. The existing bus interface and remote units are also either too heavy and power hungry, or low in performance. Because of the number of modules per satellite and the number of satellites in the constellation, in-house development of a low-power and low-weight BIU is justified. (The BIU is defined as the bus coupler together with the remote unit, with command and telemetry capability.)

Data Bus Simulations

Two sets of data bus simulations were performed to aid in the conceptual design of the satellite. The simulations were used to analyze the performance of the C^2T bus and the HSDB at the data rates selected, using the traffic data compiled. The simulations also helped resolve such issues as the minimum required data rate for the two buses to satisfy the latency constraint for specific messages; and whether the data bus system be optimized with a single set of physical wiring being used for communicating both types of messages; and, if so, what is the optimum data rate for the combined bus. The first set of simulations described below pertains to the C^2T bus analysis, and the second to that of the HSDB bus alone, as well as a combined bus using both C^2T and HSDB messages. The results of the simulations will be described along with a brief description of the simulation tools used.

Table 3 Characteristics of C^2T bus candidates

Attribute	DACS Bus	STACC bus	1553B bus	T^2C^2 bus
Architecture	TDM, synchronous full-duplex	TDM, synchronous full-duplex	TDM, asynchronous half-duplex	TDM, synchronous full-duplex
Access control	Command/response	Command/response	semi-distributed	Command/response
Real-time distributed control	Modification required	Near real time control via bus controller	Modification required	
Data rate	3.4 Mbps	1.024 Mbps	1.024 Mbps	2.048 Mbps
Word length (command)	32 bit (16 bit command message)	32 bit (16 bit command message)	20 bit (16 bit comand message)	48 bit (16 bit command message
Word length (telementry)	32 bit (16 bit telemetry data)	9 bit (8 bit telemetry data)	20 bit (16 bit telementry data)	28 bit (8 bit telemetry data)
Physical characteristics	Command bus/reply bus	Comand bus/reply bus	Single bus	Command bus/telemetry request/reply bus
Termination	Not required	Yes	Yes	Yes
Remotes	Up to 32	Up to 64	Up to 32	Up to 32
Remote unit size (in^3)	40	140	N/A	N/A
Remote unit weight (lb)	12	6.1	N/A	N/A
Remote unit power (watt)	22	3	N/A	N/A
Remote comand channels	64 discrete 16 digital	64 discrete 8 digital	N/A N/A	640 discrete 32 digital
Remote telemetry channels	64 bilevel 32 dif. analog 16 digital	64 bilevel up to 64 analog 16 digital	N/A N/A N/A	512 discrete 640 analog 64 digital
Development status	Existing	Existing	Needs additional development	Under development

Simulation Set 1

The C^2T messages are simulated for a multiplexed, 1.0 Mbps half-duplex bus, using the MIL-STD-1553B protocol. This protocol represents a centralized data bus with a command/response protocol, a fixed intermessage gap (4 μs minimum) and a variable (4–12 μs) response delay between a command and its response, imposed by the half-duplex nature of the bus. We assumed a 12 μs delay to simulate the worst case. In the 1553B protocol, one module (the CADHM in our case) is designated as the data bus controller, which would issue commands and data to the other modules and request telemetry. There is also a provision for dynamic mode control where any other module can be designated as the next bus controller. The half-duplex nature essentially reduces the bus bandwidth to half of the bus capacity. The messages were assigned priorities to the best of our knowledge.

Simulation Tool

The simulation tool is called ASSIST (ADPE system sizing and simulation tool). Developed by TRW, it is a FORTRAN-based program with some subroutines in C language. It is geared specially for simulating data processing systems using networks of processors, operating systems, software tasks, operators, and so forth, and allows easy integration of the results into the concept development process. The tool is adaptive, so that any changes in the system as the design evolves may easily be incorporated into the existing simulations.

Results

We find that at 1.0 Mbps, even with the half-duplex nature, the bus provides big margins in average bus use and low latency for messages, i.e., the wait time before gaining access to the bus. Figure 2 shows a plot of the percentage of bus use as a function of the simulation time for 60 ms. We find that in the traffic data compiled most of the messages are assumed to be initiated during the first 50 ms into the simulations. This coincides with the duration during which the bus use grows momentarily. During the next 50 ms, the bus is mostly idle. The same pattern is repeated for a 1.0 s simulation run. We also find that by staggering the start times of the messages, the bus use goes down even further, caused by the lower queue count (number of messages pending access to the bus due to a transmission in progress). Table 4 summarizes the statistics for bus use and queue wait times. The main conclusion of the simulations is that 1.0 Mbps data rate is adequate to handle the C^2T message. A detailed timing analysis of the C^2T bus (not reported here), using the 1533B protocol, also supports the preceding conclusion.

Simulation Set 2

In the second set of simulations, we simulate the HSDB traffic that consists of all the longer length, burst-type sensor messages. We use data rates of 50 and 10 Mbps. We then combine the HSDB and the C^2T traffic and simulate the combined bus to determine the feasibility of using a single set of physical wiring for communicating both types of messages. To gain added insight into the bus performance, we simulate the traffic both with and without message priorities. A description of the results, simulation tool, and the HSDB protocol follows.

Simulation Tool

The simulation tool used is one developed by Rockwell Collins in Cedar Rapids, Iowa. Collins' simulator is a special-purpose simulator, written in turbo Pascal, that simulates a SAE-STD-AE9BL$_2$, standard for a linear token passing mul-tiplex data bus. It is designed for supporting the high-speed data bus application on the Air Force sponsored Pave Pillar Program. Hence, it simulates a 50 Mbps bus. Later, we modified the program to stimulate a 10 Mbps bus also, in order to establish a lower bound on the required data rate for the bus. Collins' simulator assumes four levels of message priorities. We accordingly assign priorities to the messages as appropriate.

HSDB Protocol

The HSDB protocol is a linear token passing multiplex bus. Meyer[2] describes the characteristics of the high-speed token passing bus. The linear token passing bus consists of a set of stations connected by a broadcast transmission medium. Bus lengths of up to 1000 m can be accommodated. The data protocol is distributed and gives fair bus access to all users. Access to the media is controlled by the token. The token is continually passed around a logical ring superimposed on the linear bus. Receiving the token entitles a station to use the medium for a predetermined maximum amount of time, as determined by the token holding timer (THT) for transmitting the data. When the timer expires, or when the station has sent all of its messages, it forwards the token to the next member of the logical ring.

Low latency for high-priority messages is assured by the use of message priorities and token rotation times (TRT's). There is a predetermined TRT value for each priority level between 2 and 4. The same value is used by all the terminals. There is no TRT for the highest priority. Every time a station receives a token, all its TRT's are reset. (The timers count up until the token returns after going around the ring once.) A station that has a message at the highest priority can always send it upon receiving the token. However, a low-priority message can only be sent if the TRT associated with that priority has not expired. It the TRT has not expired, it indicates that the bus is not heavily loaded. Otherwise, the terminal must defer to the higher priority traffic by forwarding the token to the successor and awaiting the next token for transmitting the message. Thus, the bus users always defer to the higher priority traffic when the load becomes heavy.

Results

Tables 5–7 summarize the results of the simulations for various simulation lengths,using 10 and 50 Mbps buses for the HSDB traffic as well as the combined HSDB and C^2T traffic. The simulation run times were between 1.0 and 13.3 s. The average program run time being approximately 1.5 h per 1 s of simulation time for a 10 Mbps bus, we could afford only a limited length of simulation time.

The results, as seen in Figs. 3 and 4, indicate that there are significant increases in the message delays around 3, 6, 9, and 12 s. Such increases relate to those messages with corresponding periods that have long lengths and many inputs. The simulation assumes that a module presents all messages of the same type to the bus queue at the same time. In reality, however, the message arrival times will follow some sort of a time-multiplexed timeline such that the maximum queue sizes would be much smaller, making the delays smaller as a consequence.

Token rotation time is the time between a terminal relinquishing a token to its successor and the time it arrives back at

Table 4 C^2T bus performance

Parameter	Simulation time		
	Over 100 ms	Over 0.5 s	Over 1.0 s
Idle time, ms	89.54	470.38	946.32
Busy time, ms	10.46	29.62	53.68
Idle time, %	89.54	94.08	94.63
Busy time, % use (average use)	10.46	5.92	5.37
Maximum use, %	62	43.04	25.2
Average wait time, μs	5.6	3.9	3.8
Maximum wait time, ms	0.724	0.724	0.724

Table 5 Performance of a 50 Mbps bus with combined HSDB and C^2T traffic

No.	Result	1.3 s	2.0 s	3.7 s	8.0 s	9.0 s	13.3 s
1	Total traffic offered, Mops	1.38	1.38	1.38	1.38	1.38	1.38
2	Average token rotation time, μs	24.4	24.4	24.8	24.75	25.0	24.9
3	Maximum token rotation time, μs	127	127	3099	3099	3301	3301
4	Minimum token rotation time, μs	24.3	24.3	24.3	24.3	24.3	24.3
5	Mean words transmitted, words/s	10,977	13,545	56,244	54,199	83,795	73,431
6	Mean transmitted data rate, Mbps	0.18	0.22	0.90	0.87	1.34	1.17
7	Token picked up one message from a terminal	514	884	1690	3881	4538	6625
8	Token picked up two messages from a terminal	4	8	36	77	147	184
9	Token picked up three or more messages from a terminal	0	0	5	11	33	39
10	Total token passes without any terminal transmitting	52,832 (99.2%)	81,132 (99.1%)	147,340 (99.1%)	321,38 (99.1%)	358,386 (99.1%)	529,411 (99.1%)
11	Mean P1 message delay, μs	18.48	20.0	98.2	89.75	187.16	154.1
12	Maximum P1 message delay, μs	91.8	91.85	2452	2452	2931	2931
3	Mean P2 message delay, μs	14.25	14.23	108.9	93.0	181.68	154.8
14	Maximum P2 message delay, μs	34.0	34.3	3322	3322	4457	4457
15	Mean P3 message delay, μs	13.7	14.0	18.4	18.3	61.9	48.0
16	Maximum P3 message delay, μs	32.0	32.5	592.6	626	5175	5175
17	Mean P4 message delay, μs	37.0	32.7	166.7	123.5	270.6	224.6
18	Maximum P4 message delay, μs	87.0	87.2	2061	2071	7923	7923

M* = 1 million. Message distribution in priority groups: P1 – 55.5%, P2 – 33.4%, P3 – 9.4%, and P4 – 1.7%.

Table 6 Performance of a 10 Mbps bus

		HSDB + C^2T with priority		HSDB + C^2T no priorities		HSDB with priority
No.	Result	1.3 s	8.0 s	1.3 s	8.0 s	6.8 s
1	Total traffic offered, Mbps	1.38	1.38	1.38	1.38	1.38
2	Average token rotation time, μs	97.8	105.3	97.8	105.3	59.5
3	Maximum token rotation time, μs	611	22,003	611	2264	15,518
4	Minimum token rotation time, μs	96	96	96	96	54
5	Mean words transmitted, words/s	10,977	54,394	10,977	54,394	57,622
6	Mean transmitted data rate, Mbps	0.18	0.87	0.18	0.87	0.92
7	Token picked up one message from a terminal	512	3379	512	3683	948
8	Token picked up two messages from a terminal	5	132	5	105	123
9	Token picked up three or more messages from a terminal	0	83	0	47	82
10	Total token passes without any terminal transmitting	12,872 (96.8%)	73,291 (96.5%)	12,872 (96.8%)	72,992 (96.0%)	114,129 (99.2%)
11	Mean P1 message delay, μs	77	1134	71.5	745.8	1698
12	Maximm P1 message delay, μs	537	19,227	537.4	274,472	12,569
13	Mean P2 message delay, μs	75.2	20,858			26,582
14	Maximum P2 message delay, μs	263	247,133			241,864
15	Mean P3 message delay, μs	57.7	4443	There is one priority group here		
16	Maximum P3 message delay, μs	155.6	249,640			
17	Mean P4 message delay, μs	173.4	24,318			
18	Maximm P4 message delay, μs	397	205,340			

Table 7 Effect of data rate on the bus with combined HSDB and C^2T traffic

		10 Mbps		50 Mbps	
No.	Result	1.3 s	8.0 s	1.3 s	8.0 s
1	Total traffic offered, Mbps	1.38	1.38	1.38	1.38
2	Average token rotation time, μs	97.8	105.3	24.4	24.75
3	Maximum token rotation time, μs	611	22,003	127	3099
4	Minimum token rotation time, μs	96	96	24.3	24.3
5	Mean words transmitted, words/s	10,977	54,394	10,977	54,199
6	Mean transmitted data rate, Mbps	0.18	0.87	0.18	0.87
7	Token picked up one message from a terminal	512	3379	514	3881
8	Token picked up two messages from a terminal	5	132	4	77
9	Token picked up three or more messages from a terminal	0	83	0	11
10	Total token passes without any terminal transmitting transmitting	12,872 (96.8%)	73,291 (96.4%)	52,832 (99.2%)	321,381 (99.1%)
11	Mean P1 message delay, μs	77	1134	18.48	89.75
12	Maximum P1 message delay, μs	537	19,227	91.8	2452
13	Mean P2 message delay, μs	75.2	20,858	14.25	93.0
14	Maximum P2 message delay, μs	263	247,133	34	3322
15	Mean P3 message delay, μs	57.7	4443	13.7	18.3
16	Maximum P3 message delay, μs	155.6	249,640	32	626
17	Mean P4 message delay, μs	173.4	24,318	37	123.5
18	Maximum P4 message delay, μs	397	205,340	87	2071

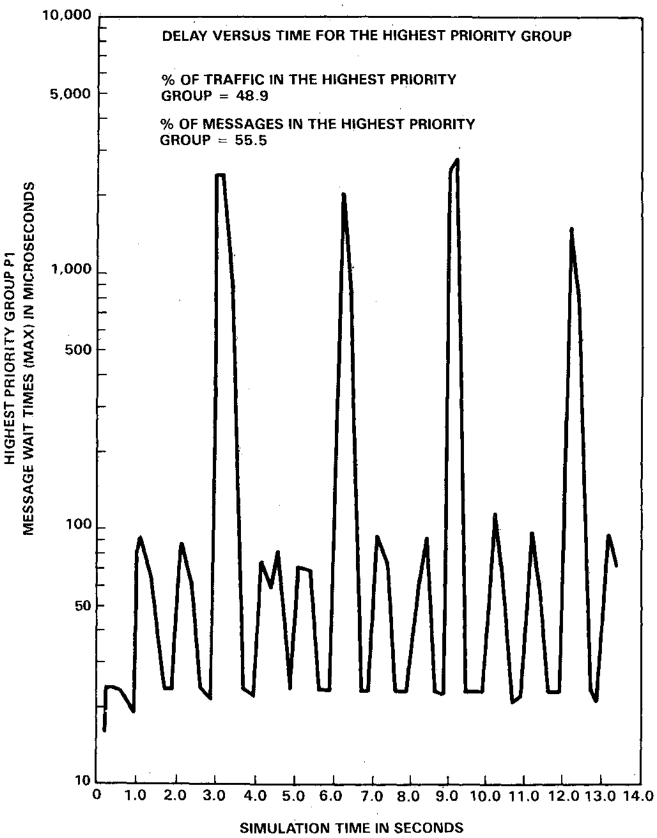

Fig. 3 Performance of a 50 Mbps bus for command and telemetry and communications messages, showing the highest priority group.

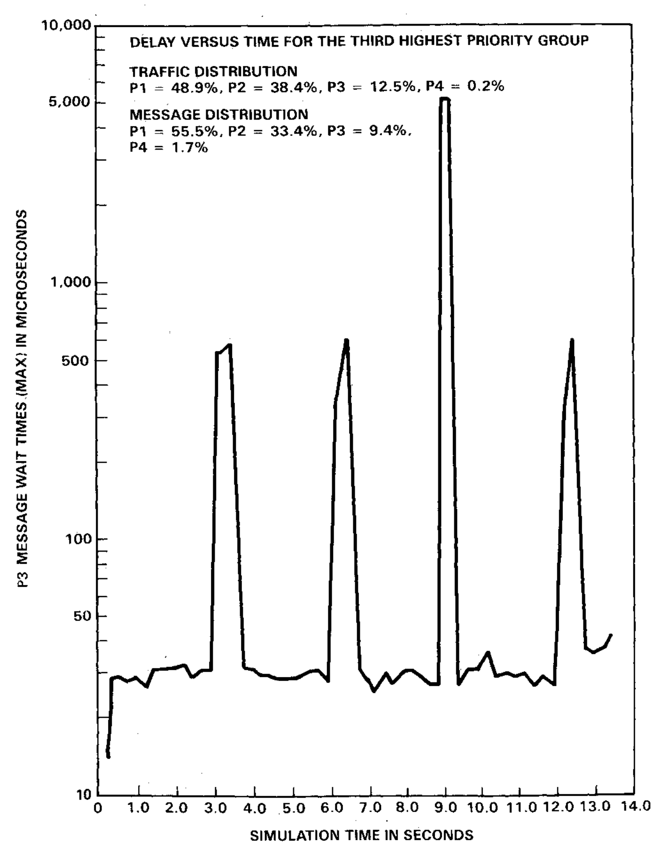

Fig. 4 Performance of a 50 Mbps bus for command and telemetry, and communications messages, showing the third highest priority group.

the terminal after going around the ring once. The token rotation time (entries 2, 3, and 4 of Tables 5, 6 and 7) gives an indication of the bus loading, since the heavier the traffic the longer the rotation time.

Table 5 shows the performance of a 50 Mbps for the combined traffic. The worst-case delays are a few milliseconds, whereas the mean delays are of the order of 0.1 ms. This is for the worst case message arrival scenario, with all the inputs arriving simultaneously. Given that the bus queues are going to be much smaller in real life because of the staggered arrival times, we can expect even smaller delays. In any case, the bus use is such that over 99% of the token passes had no transmission at all (entry 10). This means that the bus loading is in bursts, at specific intervals, and at all other times the bus is idle. Figures 3 and 4 show the plots of delay vs time. Hence, it can be concluded that a 50 Mbps bus meets the bus requirements with big margins. We can infer from Tables 6 and 7 that the current traffic can be handled by a data rate less than 50 Mbps bus because the delays are well within the 10 ms limit. If the delay requirement can be relaxed to 0.2 s, then even the 10 Mbps rate can satisfy the requirement in the worst case.

Table 6 shows the effect of adding the C^2T traffic to the HSDB bus at 10 Mbps data rate and also that of priorities vs no priorities. It is interesting to note that the HSDB data adds to the traffic only in priority groups 1 and 2. As we can see, addition of the C^2T messages makes the P1 and P2 delays longer by approximately 7 ms (for an 8.0 s run). Also, there is new traffic in P3 and P4 groups, with mean delays on the order of milliseconds. From the table, it is evident that at 10 Mbps, even the mean delays exceed the 10 ms end-to-end delay requirement. Hence, a higher data rate than 10 Mbps is required.

Table 7 illustrates the effect of increase in the data rate for the combined traffic. Again, it is seen that in some cases, the mean delays at 10 Mbps exceed the requirement of 10 ms delay; but at 50 Mbps, even the maximum delays are within that value. Hence, it can be concluded that the required data rate is between 10 and 50 Mbps.

It is apparent that although the maximum delays within any given priority group may be very long, the mean delays are much smaller than the maximum (less than 1:10 for combined traffic at 10 Mbps over 8 s of simulation time). Also, the maximum delays do not last for long periods. In most cases, it appears that the delays shoot up only around 3, 6, and 9 s (and multiples of the periods thereafter) and even those big delays go down within 0.2 s. The mean delays at all other times are almost always less than 50 and 500 μs for 50 and 10 Mbps buses, respectively.

The delay values in the tables represent the time between when a terminal enters its message in the queue and when the message starts on the bus. It does not include the transmission time. However, the transmission time for the longest message (168 words) would be 53.8 μs for a 50 Mbps data rate. Thus, the end-to-end delays would be well within 10 ms for the given traffic at 50 Mbps rate.

Another feature of the traffic is the message distribution in the priority groups. The chosen priority assignment is such that the percent of messages in priority 1 through 4 (the highest to lowest) is 55.54, 33.37, 9.4, and 1.69, respectively. A reassignment of the priorities to achieve a more even distribution (e.g., 15, 20, 30, and 35%, respectively) would make the delays even smaller within each priority group. Another significant result obtained from the simulations is that the addition of the C^2T traffic on the HSDB accounts for most of the delay in the lower priority groups P3 and P4 and also adds to the maximum delays in the other two groups. This is as expected, due to the nature of the messages. However, the delays are still within the acceptable range for a 50 Mbps bus. This indicates that a single bus can handle both the short mes- sages and the typically longer HSDB messages.

Conclusion

This paper describes the data system of a complex next-generation observation satellite in the design stage, and the tools and methods used to analyze its performance and optimize the design. To minimize satellite assembly and test costs, the satellite is partitioned into individually testable modules. The modules are independently functioning, processor-driven entities that provide the spacecraft with a certain degree of autonomy. The projected message flow between the modules to perform the C^2T functions and between the payloads to perform the observation functions led to the initial design with two data buses. The simulation tools provide good insight into the data bus operation and their handling of the messages and peak loads under the simulated environment. Based on that, we are able to ascertain that the selected data buses are adequate to handle the traffic envisioned. The bus simulation shows the margins in bus throughput, the average message latencies, and most important of all, the maximum delays. We can optimize the C^2T bus performance based on either reassignment of the message priorities or through management of how the control of the bus is slotted and passed around. The message priority assignment is such that priority group 1 contains the highest percentage of messages. It is, therefore, a conservative assignment. In reality, the traffic distribution would be more uniform over the priority groups. For the HSDB, similar insight is obtained through simulations. The most important result of the simulations is the conclusion that the HSDB alone can handle all the traffic on both buses.

We then proceeded to combine the two data buses into a single bus. This was motivated by the fact that there is a total of 14 modules, each with a BIU connected to the C^2T bus, out of which nine modules are also connected to the HSDB through nine more BIU's. A combined bus will not only eliminate nine BIU's but will also greatly improve system reliability.

The simulations show that the combined bus can handle all the traffic and meet the message latency requirement with a data rate of 10-50 Mbps. Further simulations at the intermediate data rates can identify the optimum data rate. We did not have the resources to continue the simulations to determine the optimum rate. The biggest payoff of the data rate optimization will be in reducing the HSDB clock rate. For a 50 Mbps, Oxide Aligned Technology (OAT) is needed at the front end of the BIU to perform protocol and conversion of serial data into parallel. OAT technology is power hungry compared with CMOS. In the case of the combined bus, if the bus clock rate can be brought down to the 10-20 Mbps range, CMOS can be used, resulting in savings of 8-10 W per BIU. With 14 BIU's, the savings can be significant. In summary, computer simulations allow us to simplify and optimize our satellite design, resulting in a simpler and more reliable design. Substantial cost savings are also realized.

Acknowledgment

The authors would like to acknowledge Rockwell Collins—Government Avionics Division for the use of their simulator, the work performed on which formed the bulk of the simulation results presented in this paper.

References

[1]Ludvigson, M. T. and Milton, K. L., "High-Speed Bus for Pave Pillar Applications," *Proceedings of the AIAA/ACM/NASA/IEEE Computers in Aerospace V Conference,* AIAA, New York, Oct. 1985, pp. 75-83.

[2]Meyer, S. W., "SAE-9B Draft Standard High-Speed Token Passing Data Bus for Avionics Applications," *Proceedings of the IEEE/AIAA 7th Digital Avionics Systems Conference,* AIAA, New York, Oct. 1986, pp. 234-241.