TELEMETRY TIME INTEGRATION

Edward L. Davis

Loral Instrumentation

8401 Aero Drive

San Diego, California 92123

ABSTRACT

A new systems hardware and software architecture is presented. Time-tagging is accomplished on multiple packets of information flowing concurrently at extremely high data rates. In the telemetry field to date, there has been a serious problem with the inability to properly time-tag multiple incoming data elements simultaneously. It is crucial when flight testing missiles or other aircraft to determine precisely the sequence of events.

The hierarchical modified dataflow architecture contains a supervisor module which manages an Inter-Range Instrumentation Group (IRIG) Time Acquisition Module and several acquisition and processing modules within a system. The high speed serial bussing techniques utilized allow the time module to source time information to any one or all of the other modules simultaneously. Each module operates autonomously and time-tagging is allowed at any intermediate stage of processing.

Archiving of data with the time tag allows high speed retrieval and processing of time "slices" of data. Temporal inter-relationships between many thousands of data types is determinable in real-time or post-processing.

INTRODUCTION

With the rapid advent of high-speed embedded computers into the telemetry fields, conventional real-time processing techniques are becoming inadequate. High data rates and complex process algorithms inundate general purpose computers. Coupling front end pre-processors to these computers has historically offered an acceptable solution. This brings with it the complexity of programming multiple computers interactively as well as problems of executing real-time software on conventional hardware. Problems are exacerbated by sequential processing architectures, wherein each process is iterative and dependent upon processes sharing the same processor and storage area.

In addition to the acquisition/processing/control and interactive graphics displays, the system must effectively time-tag many elements or sets of data very quickly and accurately. This added burden is sometimes slighted to a secondary priority due to lack of a solid approach at resolving it. High speed, parallel, pipelined processing systems are becoming very prevalent, but to date, no clear architectural vehicle for orchestrating these processors has emerged. The task of developing or modifying real-time software is arduous and frequently encumbered by process interdependencies that consume critical processing bandwidth.

The Telemetry Processing System (TPS) is a pragmatic approach to systems architecture targeted at solving bus bandwidth and process bandwidth of telemetry systems which highlights high-speed serial bus structures. Conventional dataflow techniques are utilized with a continuously variable packet size having file headers which are attached on-the-fly. Multiple packets flow through the system buses concurrently.

An evolutionary distributed operating system is utilized in which a supervisor manages modular program execution, dynamic process/processor allocation and data flow control. A Unix operating system runs parallel with the distributed operating system. The distributed operating system maintains a scoreboard of current and pending tasks to be executed.

Each node maintains an ID and load level indicator. Real-time process and processor control allows various levels of fault tolerance to be implemented. The Supervisor monitors status on all other nodes, senses potential problems and immediately reallocates resources as necessary. The Supervisor may spawn, suspend, or kill any process or task block on any node. Multiple high-speed array processor nodes may be plugged in as desired to select performance from 20 Million Floating Point Operations per Second (MFLOPS) to as high as 5 GIGAFLOPS.

A stimulus response type storage/retrieval technique allows tremendous programming flexibility. Responsibility for directory and file control is modularly distributed and delegated. Data may be transferred from peripheral to peripheral with no degradation in performance of other CPUs. Bus bandwidth in a single chassis is several hundred Mbytes/sec, and the architecture allows for multiple expansion chassis.

An artificial intelligent (AI) human interfaced is utilized for real-time parameter interaction, analysis and definition. Programs may be developed by operator interaction while actually executing the program. Off-line concurrent software development is equally well supported. A software automation technique is presented whereby the operator enters the specification for a program then an AI routine generates source code syntax, listings, and compiles. This relieves the programmer of cryptic, sematic, syntax data entry and associated errors and provides optimized macro kernels which are expanded via an AI module to fit the programmmer's specifications.

2.0 SUPERVISOR/FRAGMENTED OPERATING SYSTEM

The TPS system contains a variety of node types. The Supervisor node (SUP) executes the main kernel of the Fragmented Operated System (FOS). The FOS is a lean real-time distributed operating system. Unlike conventional architectures, the SUP never executes any application software, its responsibility is to execute only the operating system. The purpose for distributing the FOS is to gain a higher degree of specialization in function; thereby greatly reducing the breadth of software development requirements overall. The architecture of the TPS also capitalizes on VSLI in the form of Reduced Instruction Set Computer (RISC) processors on each mode to reduce software complexity, development time and volume.

The SUP is a hierarchical data/control flow manager. It is responsible for initialization, configuration, task allocation, monitoring and managing all nodes via a master schedule scoreboard. The SUP initializes the scoreboard by polling each node by its geographic location and receiving the node type identification from each one. The tasks to be executed are then parceled out to each node according to the nodes predefined capabilities after the SUP configures the FOS.

If a nodes load balance counter is less than 25% or greater than 75%, it will report status to the SUP of underburdened or overburdened respectively. When a nodes load balance counter reaches zero, the nodes Input Output Director (IOD) will begin running background diagnostics and as necessary, fault isolation.

The SUP delegates scoreboard items in the form of Task Control Blocks (TCB) to all other nodes. In this manner, the SUP may interrogate, spawn, suspend or kill any processes on any node(s) at any time.

The SUP also handles fault tolerance mechanisms by passing semaphores for acknowledgement in the form of messages within a TCB. The SUP could not easily exercise this magnitude of control over the constituent nodes unless it had nearly instantaneous direct access and communication with all nodes continually.

3.0 HIGH SPEED SERIAL BUS

The quest for an architecture to virtually eleminate all bus bandwidth restraints while simultaneously reducing the number of I/O connections required led to a two-fold concept. The foundation of the TPS system is the multiple, independent High Speed Serial Bus (HSSB) structure. The protocol and transmission techniques utilized on these buses is referred to as Impulse Dataflo (ID). (See Figure 1).

An Application Specific Integrated Circuit (ASIC) is being utilized to implement the physical hardware layer. This ASIC chip, the Input Output Director (IOD) ASIC, contains sixteen buses. A data compression technique is implemented in the IOD ASIC which further extends the bus bandwidth. Peak data rates of over one Gigabyte per second are achievable on the HSSB with a single IOD ASIC chip on each node.

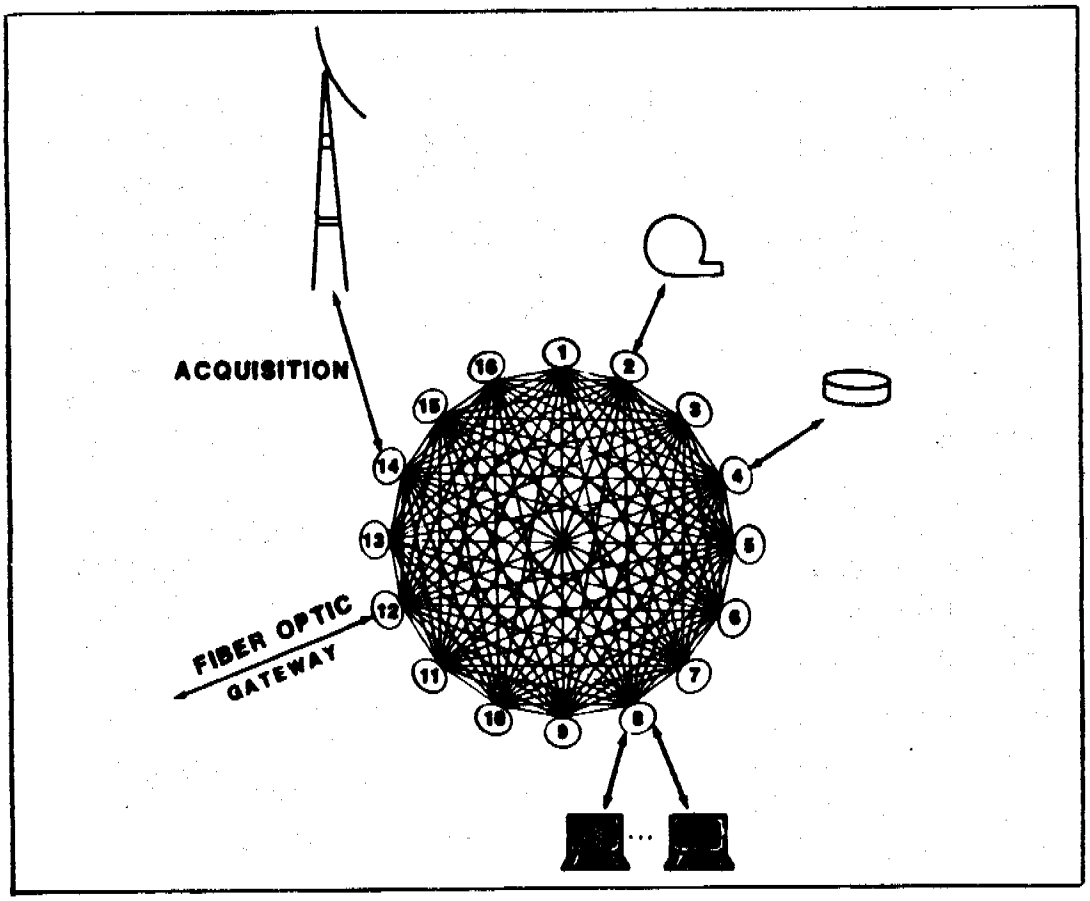


Figure 1. TPS LOGICAL/PHYSICAL BUS CONSTRUCTION

The HSSB buses do not require arbitration by the module upon which they reside. The SUP assigns buses for transmission to specific nodes as necessary to execute that nodes TCBs. Any node may receive on any bus at any time; consequently, only the ability to transmit is allocated by the SUP. The transmission protocol is simplified by having the SUP allocate these buses as quasi-dedicated to particular nodes obviating the requirement for collision detection and arbitration by nodes. The technique employed utilizes both packet switching and circuit switching and optimizes advantages of both. The SUP also reserves bus zero for a private control flow channel to all nodes. In this manner, the SUP oversees all activities while maintaining a tight coupling to constituent nodes. Allowing multiple independent simultaneous communications between nodes greatly reduces I/O bandwidth constraints. Also, peripheral to peripheral transfers may occur between nodes without tying up any other nodes or their memories or communications.

4.0 IMPULSE DATAFLO

Unlike conventional data flow architectures which pass a tag and data on each cycle (to allow multiplexed access to the bus), the Impulse Dataflo takes advantage of packet switching techniques transferring large numbers of data items with a single tag header. In conventional data flow machines, a large portion of the bus bandwidth is consumed by transmitting these tags on each cycle. Utilizing circuit switching techniques on the transmit side only allows each node the luxury to transmit at any time without arbitration. In addition, the data compression on the packet (applied in hardware) essentially adds a new dimension to the phrase data flow.

There are several types of tags employed in the TPS system to allow clear concise message handling, semaphore passing and Impulse Dataflo. The aforementioned TCBs are sent to the constituent nodes utilizing a TCB tag followed by the list of tasks to be performed. A status tag is sent by a node at the completion of a task (if requested by the SUP). Following that status tag is the current status of the node and its tasks. A data tag describes the packet of information to follow including: Number of bits per word, number of words per packet, and the identification number. A multitag defines multiple sets of interlaced data to be sent in a single packet. This tag explains the time division multiplexing of how the data is packaged. A header tag is utilized to name a new file or retrieve an existing one. This file name corresponds to a long directory listing on conventional computer systems.

The SUP may send a TCB tag containing an interrogate bit which requires a node to respond immediately. The SUP uses HSSB 0 for all transmissions and addresses nodes by their geographical location; (i.e., the TCB tag contains a field which designates which node it is addressed to. The SUP, therefore, has high speed unencumbered communication to the nodes. Control flow is easily accomplished.

5.0 STIMULUS-RESPONSE FILE STRUCTURE

File storage and retrieval I/O on conventional systems require that an operating system maintain records and track all files throughout the system and on all attached peripherals. This burden of handling I/O interrupt service routines can easily overwhelm a real-time operating system and drive it lethargically to a standstill.

A key feature in the TPS system which again optimizes information flow and reduces the burden on the SUP and the FOS is the Stimulus-Response (SR) File Structures utilized for file management and handling. This structure allows all data to be stored with a Universal Time Constant (UTC) header which uniquely identifies it for current and future processing.

This file management technique distributes the burden of responsibility to each node rather than forcing the SUP and the main kernel of the FOS to be globally responsible for executing every IO interrupt service routine.

6.0 NODES

The nodes within a chassis are heterogeneous although multiple nodes of a particular type are accommodated, (e.g., ten 40 MFLOP array processor nodes may be installed to achieve 400 MFLOP performance). All node types contain two Reduced Instruction Set Computer (RISC) Bit Slice Processors; one in the Input Output Director (IOD) section; and another in the Processing Element (PE) section. The Metacompiler "C" software architecture supports both. Each RISC processor contains writable control store and comprehensive diagnostic testware.

6.1 Input Output Director (IOD)

The Input Output Director section of a node contains sixteen channels of high speed serial bus on an ASIC chip and a RISC bit slice IOD processor which communicates to the SUP, other nodes and its on-board Processing Element (PE). The IOD executes a portion of the Fragmented Operating System. The IOD runs diagnostics on its associated PE and reports discrepancies to the Supervisor.

Each node has a geographic location and an on-board type identification and history log in EEROM. Each node handles file management for all files under its current jurisdiction. The IOD is the node controller and DMA interface also. During diagnostics, fault isolation and symbolic debug, the operator communicates through the SUP to the IOD and the IOD physically exercises the node and returns responses to the SUP.

6.2 Processing Element (PE)

The PE portion of the node is also a RISC bit slice processor. Since the nodes are heterogeneous, they may contain a wide array of additional hardware and software. The RISC processor controls everything on the node, except the IOD, whether the node is an array processor, quantizer, peripheral controller, graphics driver, etc.

6.3 Some Types of Nodes

(1) 40M Flop Array Processor (16 bit integer to 64 bit IEEE floating point).

(2) IRIG Format G Time Acquisition and Generation node.

(3) 120 channel Analog/Digital/Analog Signal channel I/O.

(4) SMD interface module (20 M bits/sec) real-time storage/retrieval.

(5) Real-time interactive window graphics work-station controller node.

(6) Supervisor node.

(7) Fiber Optic Gateway and Bus Expansion (up to 200 M bits/sec).

7.0 SOFTWARE

The use of RISC architectures consistently on all nodes reduces the breadth of the software requirement. The Metacomplier for these RISC processors is fairly straight-forward. Utilization of Artificial Intelligence (AI) in implementing software automation is expected to greatly enhance user friendliness. Semiconductor memory is becoming so inexpensive that it is now economically feasible to implement a large knowledge base to handle parsing syntax and semantics in real-time. Also, real-time expansion and integration of macro kernels is expected to greatly reduce programming time.

The software automation relies on stored examples of particular types of routines (a library) such as an eight point FFT, a four by four matrix multiply, an augmented matrix solution, matrix inversion example, etc. When the operator utilizes these functions in a program they are simply specified along with the dimensions and the AI runner performs the expansion of these kernels to the proper size and integrates them into the user's program.

Interactive real-time development is permitted by use of parameter blocks. The graphics work-station screen supports windowing where several parameters and graphs may be viewed and/or modified. A program that was previously developed may be modified by adding, deleting, or otherwise changing these parameter blocks; i.e., the types of processing and the funtions invoked on various streams of parameters are adaptable by the operator/programmer. At the end of the run, the new program may be stored on disk or other media in readable source code form.

8.0 FAULT TOLERANCE

The TPS system architecture implements a graceful degradation in performance fault tolerant system. Duplicate hardware at the node level is not required; instead the less burdened PE nodes already running will be allocated more processing to do if one of their neighbors fails.

A novel implementation of the rollback and recovery scheme presented by Gaudiot [1985] is exploited by the hierarchial control structure of the TPS.

The acknowledgement of correct data receipt is handled through the Supervisor node as is the subsequent eradication of used data after acknowledgement. The supervisor assigns a task control block (TCB) to node n-1, (see Figure 2) The TCB indicates that node n-1 should retain the resultant data until receipt of a TCB which permits eradication.

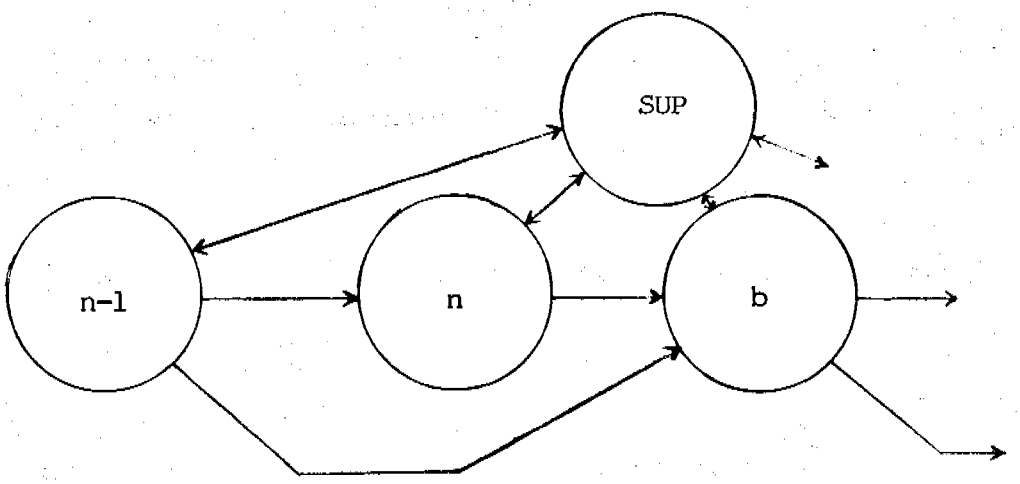
The supervisor sends a TCB to node n, containing the type of process to perform, tag header identification and acknowledge request.

After completion of the initial TCB, node n-1 exports the output to node n but node n-1 retains a copy of that data until otherwise advised by the supervisor. The supervisor then issues a TCB which advises node n-1 to eradicate the data. Had an error occurred at node n, the supervisor would check the scoreboard for load levels on other nodes of the same type and send the TCB which was initially sent to n over to an underburdened node (node b).

This scheme may be selectively implemented only on certain data and nodes at specific times or it may be applied to the entire system continuously.

For optimum processing speed at critical times, the supervisor may send the TCB which went to node n to node b simultaneously and node b becomes a "warm back-up". Should node n fail, it would be deleted from the scheme and the results automatically taken from node b upon completion.

When node n is deleted, the SUP will immediately assign another node to take its place. Concurrently, the SUP will issue a diagnostic TCB to node n. The diagnostic TCB cause the node n IOD to run fault isolation diagnostics on that node to determine the nature and extent of damage and report back to the SUP. The SUP may then utilize this node in a degraded capacity to the extent allowable by the damage; i.e., if a memory error occurred, the section of memory where it occurred would be blocked out of that nodes resource table. In any case, the operator is alerted and the malfunction is stored in the nodes onboard history log.



FAULT TOLERANT FLOW

Figure 2

9.0 SUMMARY

The TPS system is a complete packaged solution to:

(1) Real-time interactive telemetry stream acquisition and processing of signals from missiles, aircraft, and satellites to analyze data, monitor and report errors, perform vibration analysis on the airframe and compute partial derivatives of parameters with respect to others

(2) Real-time interactive simulation of entire avionics systems and subsystems prior to their development to analyze their interaction with other existing systems. Also to monitor and report errors and variations as well as inject errors into the system to analyze responses.

The TPS supports several simultaneous users with high level interactive graphics with imperceptible performance loss. Interactive monitoring, and analysis of real-time processes operating at high speeds has been a difficult task to date. At best, systems barely manage to stay abreast of the events occurring. The TPS is designed to unleash I/O bandwidth and compute power which will not only allow the operator to keep pace with events on the screen but also the TPS may be checking, comparing, and modeling the entire system in the background. This additional processing power allows the TPS to perform concurrent background processing and:

(1) Forewarn the operator(s) of impending or actual problems by reporting to a reserved window on the screen, (e.g., if one parameter is changing too fast with respect to another);

(2) Take action to rectify problems in a closed loop system while informing the operator(s) via the reserved window area;

regardless of what parameters the operator is currently viewing when a background problem occurs.

REFERENCES

1. Moller-Nielsen, P., and Staunstrup, J., "Experiments With A Multiprocessor". IEEE Computer Society Computer Architecture Technical Committee Newsletter, December 1984.

2. Vedder, R., Campbell, M., Tucker, G., "The Hughes Data Flow Multiprocessor", Proceedings of the Fifth International Conference on Distributed Computing Systems", May, 1985, Pp. 1-9.

3. Davis, E. L., "Multibus Array Processor Uses 29116 To Speed Computations", Electronic Imaging, May 1983, Pp 44-47.

4. Patterson, D. A., "Reduced Instruction Set Computers", Communications of the ACM, Volume 28, Number 1, January 1985, Pp. 8-21.

5. Patterson, D. A., and Sequin, C. H., "A VLSI RISC", Computer, Volume 15, Number 9, September 1982, Pp. 8-21.

6. More, B. R., "More Hardware Means Less Software", IEEE Spectrum, Volume 18, Number 12, December 1981, Pp. 30-37.

7. Gaudiot, J. L., Raghavendra, C. S., "Fault- Tolerance and Data-Flow Systems", Proceedings of the Fifth International Conference on Distributed Computing, May, 1985, Pp. 16-23.

QUESTIONS AND ANSWERS

UNIDENTIFIED QUESTIONER:
You show the use of a certain frequency, 10 MHz, I think. Are there other frequencies that can be used?

MR.DAVIS:
Yes, the bandwidth of the system is from 100 KHz to 200 MHz, so within that bandwidth you can use any signal that you want. The important thing about the pulse through the system is that, since we allow for a skew, and since it is linear, it treats rising edges and falling edges the same. This is my assumption, and it has been verified to within about eleven picoseconds.

UNIDENTIFIED VOICE:
It is worth mentioning that, given the wide bandwidth of the system, you are probably dealing, from pulse to pulse, with white phase noise processes. This means that you can improve the time stability by averaging. Hence, the actual time stability of the instrument may be of the order of picoseconds.

MR. DAVIS:
Yes, the numbers that you saw are the average of one thousand measurements.