HIGH PERFORMANCE, REAL-TIME, PARALLEL PROCESSING TELEMETRY SYSTEM

Richard L. Powell **Farhad Razavian**
Gale L. Williamson **Paul J. Friedman**

Loral Instrumentation
8401 Aero Drive
San Diego, California 92123

ABSTRACT

Flight test and signal and image processing systems have shown an increasingly voracious appetite for computer resources. Previous solutions employed special-purpose, bit-sliced technology to supplant costly general purpose computers. Although the hardware is less expensive and the throughput greater, the expense to develop or modify applications is very high. Recent parallel processor technology has increased capabilities, but the high applications development cost remains. Input/output (I/O) such as intermediate mass storage and display has been limited to transfer to general purpose or attached I/O computers.

The PRO 550 Processing and Storage Subsystem of the System 500 was developed to provide linearly expandable, programmable real-time processing and an interface to distributed data acquisition subsystems. Each data acquisition subsystem can acquire data from multiple telemetry and other real-time sources. Processing resources are provided by one or more 8 MIPS (20 MFLOPS peak) processor modules, which utilize an array of predefined algorithms, algorithms specified by algebraic notation, or developed via high level languages (C and Fortran). Setup and program development occur on an external, general purpose color graphics workstation that is connected to the subsystem via an Ethernet network for command, control, and resultant data display.

High-performance peripherals and processors communicate with each other via a 16-MHz broadcast bus, the MUXbus II, where any or all devices can acquire data elements called tokens. A token is a single MUXbus II word of 32 bits of data and a 16-bit tag to identify the word uniquely to the acquiring modules. The output of each device to the bus can be one or more tokens, but each device captures the bus to insert a single token. This ensures all devices receive equal priority and the MUXbus II is maximally utilized. This multiple

instruction, multiple data (MIMD) architecture automatically schedules and routes data to processors or to I/O modules without control processor overhead.

Traditional peripherals and administrative functions utilize the second subsystem bus, which is a traditional VMEbus. It controls the high performance devices while permitting the utilization of standard off-the-shelf controllers (e.g., magnetic tape, Ethernet, and bus controllers) for less demanding I/O tasks. A dedicated Bridge Module is the gateway for moving data between bus domains.

INTRODUCTION

The goal of employing many relatively inexpensive processors in parallel to achieve the yield of a main-frame computer has existed for years. To date, parallel processing has seen limited application beyond the research environment. This is due to the difficult task of decomposing or fractionating programs onto multiple processors. Telemetry data processing applications, on the other hand, are prime candidates to utilize parallel processing architectures, because the data stream emanates from multiple, usually independent, sensors.

Although there is often the requirement to keep results synchronized, processing independent signals can occur independently (i.e., asynchronously). Individual telemetry application processes tend toward small to medium granularity (i.e., each sample requires only a few to a few hundred machine instructions before data is transferred to output or storage peripherals). Therefore, a very large portion of the computing resources is spent on input/output (I/O). The overhead associated with communications on traditional proprietary and standard computer bus technology (e.g., VME, Multibus) is severely throughput limited. This situation is aggravated when employing multiple processors. However, standard bus architectures offer a large array of commercial-off-the-shelf hardware and software products to build systems. Perhaps the best solution would incorporate the architectures of both a standard bus for low performance tasks and a proprietary bus designed to meet the requirements of the application for high performance.

ARCHITECTURE

The System 500 belongs to a family of products designed specifically to meet the requirements of telemetry applications. These products incorporate real-time data acquisition, processing, data distribution, and display graphics through the optimum utilization of real-time resources in a distributed architecture. Data transfer from point of capture, to decommutation, to real-time processing by traditional or user-created algorithms, and finally to storage on very high performance media occurs via a single high-speed logical bus that is synergistic with flight test telemetry data. This is the

MUXbus, a parallel data flow bus, that transmits each parameter as a single token consisting of a 16- or 32-bit data word and a 16-bit tag uniquely defining its source. The overwhelming majority of acquisition devices produce single word values, thus a perfect match of bus structure to data. In those applications requiring multiple data words, multiple tokens are employed. Receiving devices on the MUXbus simply acquire multiple unique tagged tokens to construct the complete measurement.

The MUXbus is also a broadcast bus; any device on the bus may be set to acquire specific tokens. Multiple devices acquire data simultaneously (Figure 1). This can be compared to traditional computer buses moving data through memory from or to a single device (Figure 2). Separate transfers are required to move the same data to multiple locations. The System 500 philosophy is to include the entire device as one or more functionally organized cards in its chassis. Bit syncs, decoms, quantizers, analog and digital ports, MIL-STD-1553 bus interfaces, and many other devices are on the MUXbus and housed in the same chassis. Only large mass storage and display devices are relegated to the outside.

Expansion can occur as required to meet new requirements, simply by adding new or more of the same resources. A hub and spoke configuration is employed for large systems of many input streams and output devices (Figure 3). The hub chassis houses the processing and storage resources. Its MUXbus was increased threefold to accommodate the concatenation of data. Although each chassis contains its own independent MUXbus, the network is considered one logical bus. Tokens may be kept within a chassis or automatically moved to or from the hub at full bus speed.

The hub chassis employs a VMEbus for administrative tasks including system setup, loading algorithms, and monitoring performance. It also provides a mechanism to utilize relatively low speed (when compared to the MUXbus) commercial-off-the-shelf I/O controllers. Why the VMEbus? The definition of bus standards led to the development of literally thousands of board-level processing, support, and I/O controller products for each accepted architecture. A major attraction to standards is that system developers could easily create unique solutions for a given application. If an entire solution could not be fabricated from off-the-shelf products, then one would at least have a tremendous head start.

To accommodate unique applications such as those demonstrated by high performance telemetry systems, bus architects left room for a proprietary data bus. Unfortunately, there is not one standard but several. New standards emerged as microprocessor technology advanced from 8 to 16 to 32 bits. Competition evolved because of the conflicting goals of the bus architects. By June 1987, the IEEE recognized four standard 32 bit buses: VMEbus (IEEE 1014), Multibus II (IEEE 1296), Futurebus (896), and NuBus (IEEE 1196). The system developer is left in a quandary over bus selection. In reality, the selection for the System 500 was determined by performance and the breadth of processing, support, and I/O products available for the VMEbus.

Employing twin buses required the development of a conduit to move data between each of the bus environments. Thus, the Bridge Module was created to collect defined tokens (parameters) from the MUXbus and place them in the VMEbus System Controller's environment. Each module can acquire three streams with a unique set of parameters to be acquired for/from each of three peripherals. Data can also be moved in the opposite direction. Additional Bridge Modules can be added as dictated by the application. The System Controller is a based on a Motorola 68020 microprocessor with a real-time executive for optimum performance.

Data manipulation and reduction is accomplished through one or more Field Programmable Processors (FPP) in a parallel processing environment. Each FPP incorporates a high performance Weitek XL RISC architecture numerics microprocessor with floating point coprocessor. This combination produces a LIN-PACK rating of 3.4 MFLOPS (Million Floating Point Operations per Second) and 8.7 million Whetstones per second. This exceeds the performance of a Digital Equipment Corporation VAX 780 FPA by over 8 and 25 times, respectively. But perhaps the most significant benchmarks for telemetry applications are the maximum throughput of 740,000 parameters per second and over 200K fifth-order polynomial expansion engineering unit (EU) conversions per second. Memory (768K bytes) is segmented into areas for code and data. Programs are loaded at system setup to retrieve specific tokens (parameters) from the MUXbus, perform algorithms, then place results on the MUXbus for retrieval by other processors or output devices.

Should the power of a single FPP be insufficient to complete processing algorithms before the receipt of the next tokens, parallel processing environments can be employed to achieve results.

(1) Pipeline parallelism by splitting a large algorithm into several serial components and routing intermediate results to another FPP as in Figure 4. This scheme functions without bottlenecks if the process duration for each subtask is less than the sample rates. Further division of the subprocess to reach goals may be required or another tack employed. Pipelining becomes inefficient as the algorithm processing time approaches communication overhead.

(2) Telemetry data typically emanates from multiple independent sources. Therefore processing tasks may be distributed to multiple FPPs. Again one must insure process durations do not exceed sample rate.

Combinations of these techniques and/ or the addition of more FPPs can be utilized to achieve the desired throughput for almost any telemetry application.

Three programming alternatives are available to develop application suites: (1) selecting one or more standard algorithms from a large repertoire; (2) defining algorithms through algebraic notation, including use of transcendental functions and parameters names; and (3) high-level languages of C and Fortran plus assembler for the utmost single processor throughput.

The System 500 architecture supports a variety of output devices including analog and digital modules, DMA channels to external mini- and superminicomputers, and SMD (disk drive) and magnetic tape controllers for data rates to 1.2 MB/second and 220 kB/second, respectively. Even higher performance mass storage is available through the use of parallel disk drives (also called disk stripping). The current mass storage data transfer bottleneck is mechanical, with an instantaneous throughput of less than 3 million bytes per second (MB/s). Adding overhead for head movement and revolution latency reduces this further.

In disk stripping, the controller acquires data at many times the speed that a single drive can absorb. It then ships a buffer to the first drive. As this drive is writing the buffer contents, the next drive is shipped the next buffer, and so on. The bottleneck now becomes the controller, as opposed to the electromechanics of the drive. Continuous storage transfer rates above 10 MB/s are achievable with over 16 GB (16 x 10^9 bytes) of on-line storage. In addition to continuous storage, multiple wrap files can be employed to reduce data retention. Here the last “x” minutes of data is continuously stored, new data writes or “wraps” over old. Upon command or a predefined condition, this file (along with “y” minutes of new data) is storage for later retrieval and new wrap files continued.

CONCLUSION

The System 500, uses a parallel processing architecture with integral standard and proprietary buses to meet throughput requirements of data intensive telemetry applications. The standard VMEbus offers a tremendous range of low performance, off-the-shelf-peripherals, while MUXbus II provides a data flow highway for multiple parallel processors, FPPs, and a very high performance parallel disk drive subsystem.

BIBLIOGRAPHY

Friedman, Paul J., System 500 Flight Test System (Real-Time Analysis, Reporting, and Decision Support). International Telemetering Conference Proceedings, Vol. XXIII, 1987.

Kaplan, Ian, ACM SIGPLAN, Programming the Loral LDF 100 Dataflow Machine, Vol. 22 #5, May 1987 Pg 47-57

Querido, Robert and Friedman, Paul J., Distributed, Real-Time, High-Resolution Color Graphics Display System for Telemetry, this volume.

White, George P., Update: Battle of the Buses, Unix World, March 1988, pages 85-86.

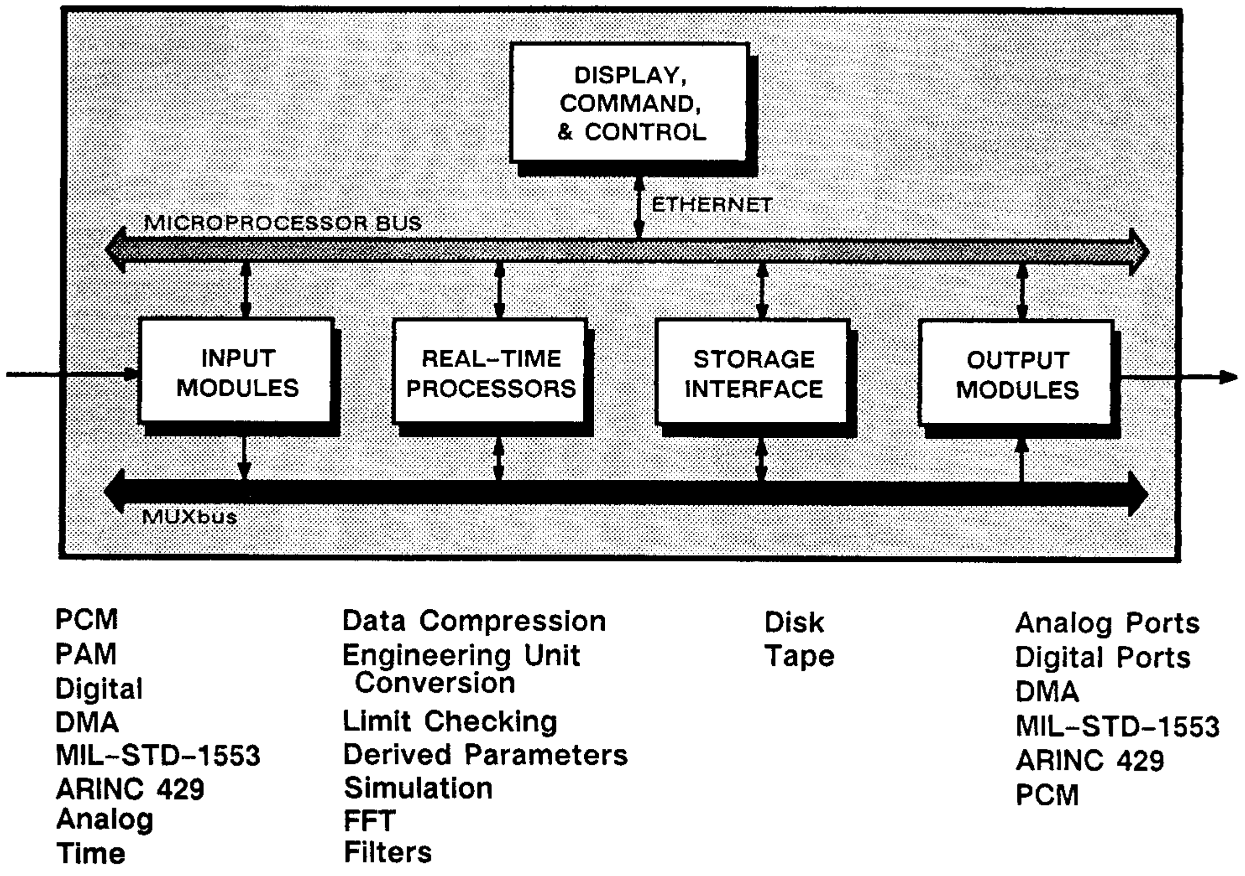

Figure 1. System 500 Data Flow Bus Architecture With Separate Control and Display Bus

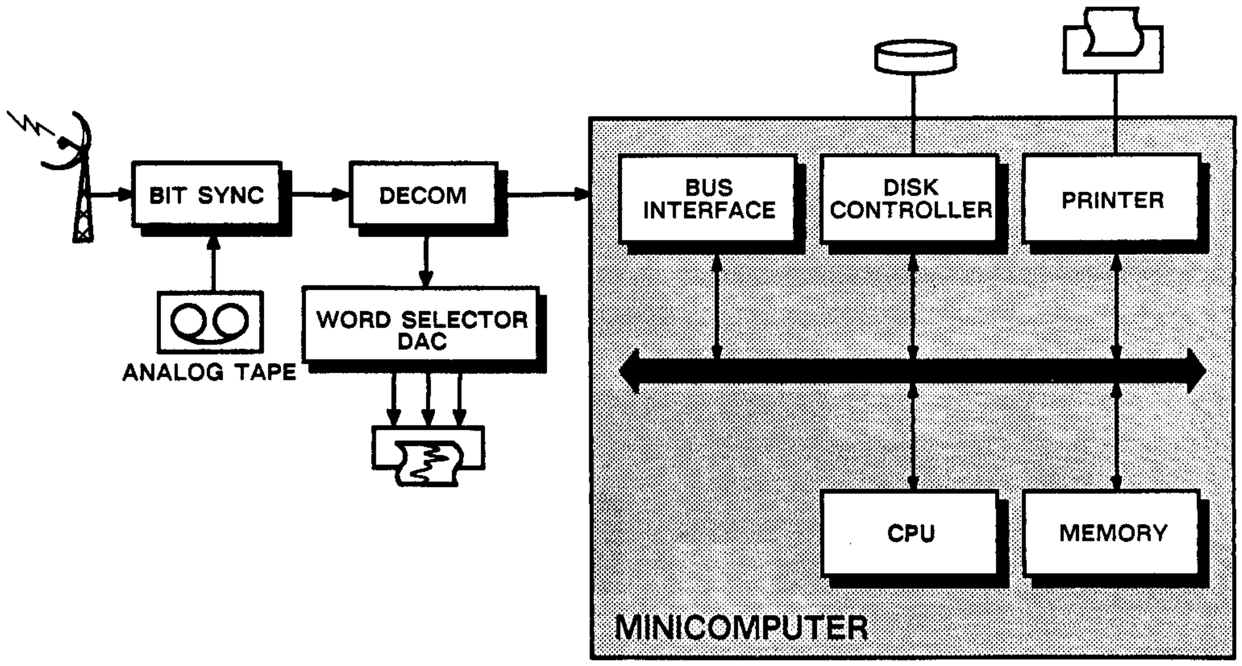

Figure 2. Traditional Flight Test System Architecture

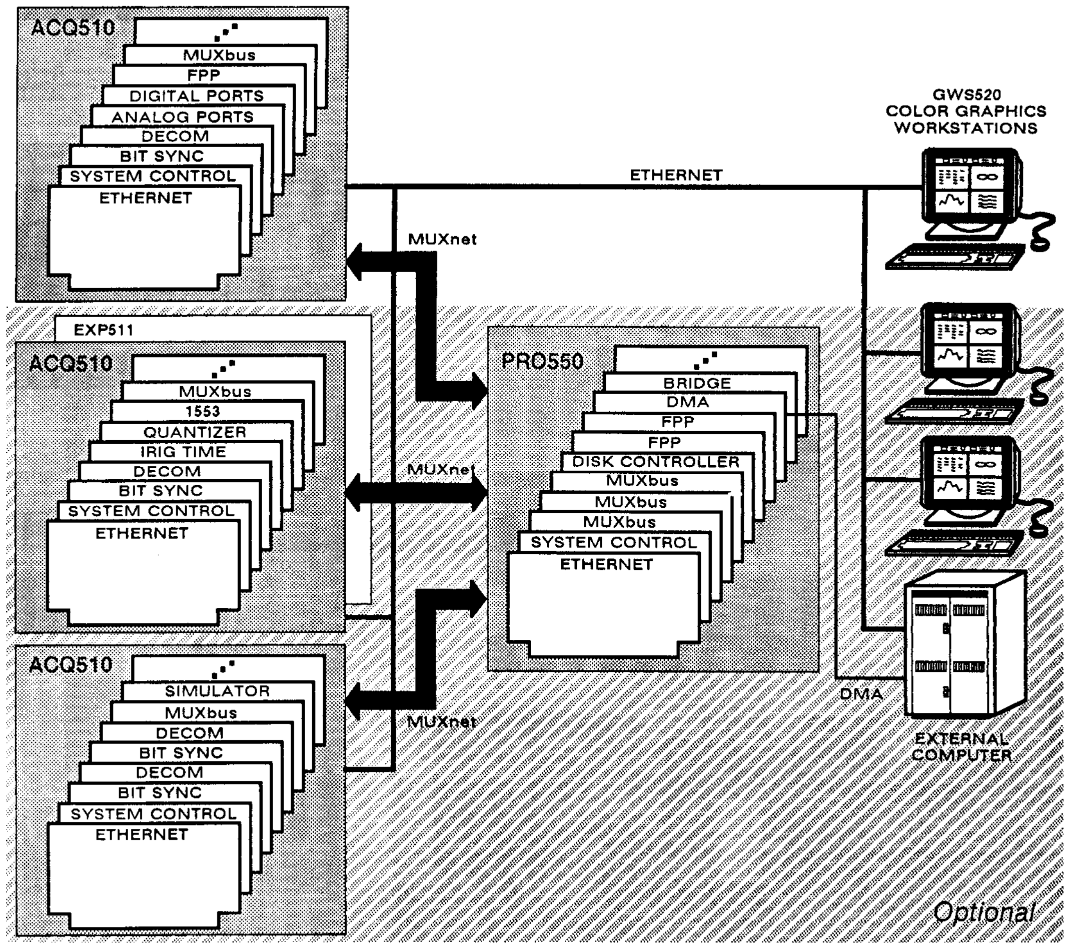

Figure 3. System 500 Modular Hardware

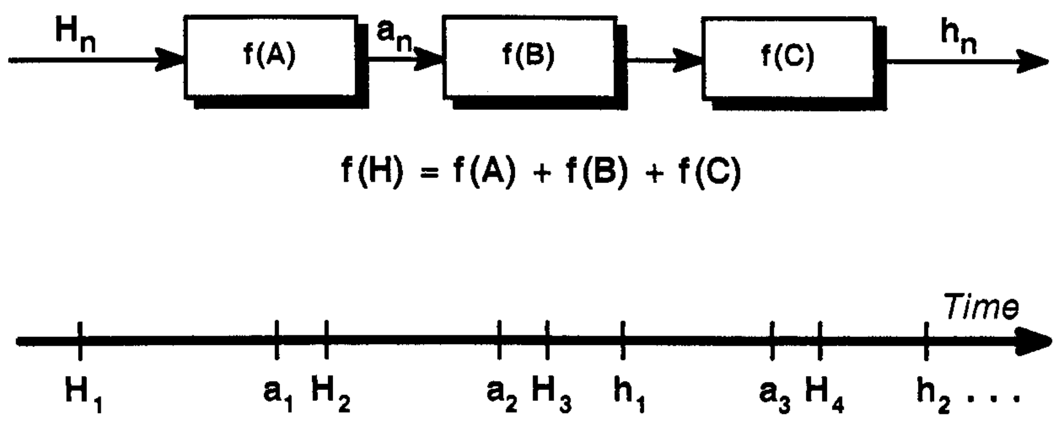

Figure 4. Pipeline Processing