High Performance VLSI Telemetry Data Systems

J. Chesney, N. Speciale, W. Horner, S. Sabia
Data Systems Technology Division
Mission Operations and Data Systems Directorate
NASA, Goddard Space Flight Center
Greenbelt, Maryland 20771

Abstract - **NASA's deployment of major space complexes such as Space Station Freedom (SSF) and the Earth Observing System (EOS) will demand increased functionality and performance from ground based telemetry acquisition systems well above current system capabilities. Adaptation of space telemetry data transport and processing standards such as those specified by the Consultative Committee for Space Data Systems (CCSDS) standards and those required for commercial ground distribution of telemetry data, will drive these functional and performance requirements. In addition, budget limitations will force the requirement for higher modularity, flexibility, and interchangability at lower cost in new ground telemetry data system elements. At NASA's Goddard Space Flight Center (GSFC), the design and development of generic ground telemetry data system elements, over the last five years, has resulted in significant solutions to these problems. This solution, referred to as the *functional components approach* [1], includes both hardware and software components ready for end user application. The hardware functional components consist of modern data flow architectures utilizing Application Specific Integrated Circuits (ASICs) developed specifically to support NASA's telemetry data systems needs and designed to meet a range of data rate requirements up to 300 Mbps. Real-time operating system software components support both embedded local software intelligence, and overall system control, status, processing, and interface requirements. These components, hardware and software, form the superstructure upon which project specific elements are added to complete a telemetry ground data system installation.**

This paper describes the *functional components* approach, some specific component examples, and a project example of the evolution from VLSI component, to basic board level functional component, to integrated telemetry data system.

1 Introduction

To insure the "routine" nature of space activities in the future, many essential science and engineering developments must occur over the next ten years. One of the most crucial of these developments is the evolution of current space telemetry systems. The widespread

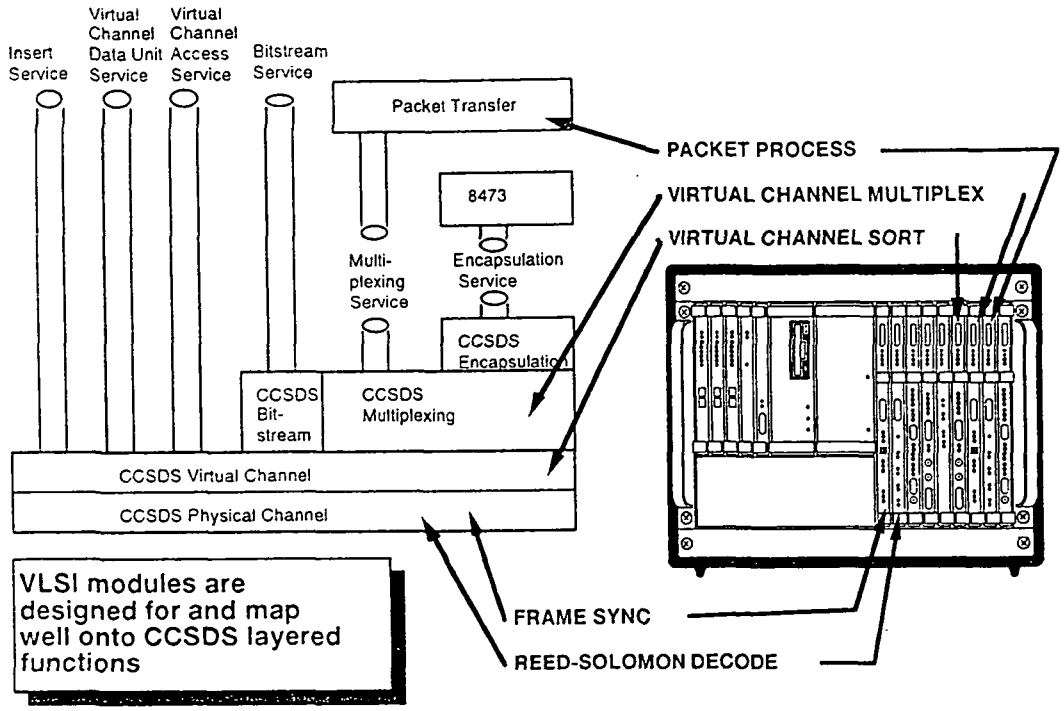

Figure 1: VLSI Systems Mapping To CCSDS Layered Architecture

development and/or adoption of various international standards affecting space telemetry data (CCSDS, CCITT, ISO etc.) promises a great potential to acquire, exchange, process, and distribute space telemetry data with fewer unique system designs. This step, however, is only one part of the required solution. In fact, the adoption of these standards without significant advancements in ground and flight system elements has caused some measure of concern on the part of developers attempting to provide the added functionality required to meet new standards at todays higher data rates (up to hundreds of Mbps).

The goals of today's space telemetry system developers are very similar to those of early television pioneers. Technology had provided a revolutionary system for the transmission and reception of information. The great goal of those individuals was the very broad distribution of this technology to many potential providers and consumers. Three essential elements were required (from a purely developmental perspective) to meet this goal. First, standards were required from which a broad range of commonality in performance and functionality could be insured. Second, development of essential engineering capability to meet the performance and functional standards. Third, the development of low cost, easy to use, and reliable consumer products that apply the fundamental engineering advances to the standards required to support the television industry.

Standards, the first of these elements, are well underway in most areas of space telemetry system development. With so many advances in basic hardware and software engineering capability today, the second element, the development of essential engineering capability, has, in most regards, been met. The development of low cost, easy to use, and reliable consumer telemetry products is becoming more of a reality today and is the basic subject of this paper.

2 Functional Components

The Data Systems Technology Division (DSTD), at NASA's Goddard Space Flight Ce nter (GSFC) has, over the last five years, applied state of the art technology to enhance the performance and reduce the costs of NASA ground telemetry data system design and development. As part of this effort, the MicroElectronic Systems Branch of the DSTD, has designed and developed a variety of generic hardware and software processing elements used to capture, process, and distribute space telemetry data. The functional components approach includes a general philosophy which attempts to combine these basic elements into larger functional components which are easily integrated into a high performance, configurable, low cost, and high reliability space telemetry system. The key elements in this approach are Very Large Scale Integrated (VLSI) circuits and advanced, highly integrated, real time software system environments. The VLSI devices meet the high performance, small size, and low cost associated with the more standard functions required in modern telemetry systems, while the real-time software systems provide the necessary flexibility to meet project specific needs. The high densities and clock speeds now available from a variety of commercial programmable logic devices allow these devices to meet nearly all other combinational logic needs. The general requirements for control, status, and data exchange/processing are supported by extensive use of commercially available:

1. modules (e.g. CPUs, memory & I/O, etc.);
2. system environments (e.g. VME, Multibus, NuBus, etc.);
3. communication protocols (Ethernet/FDDI); and
4. software languages, real-time operating systems, and support utilities.

These systems provide easy access to next stage computer processors often required for higher level data processing of space telemetry data. Figure 1 shows a mapping of a functional component system to the CCSDS layered system architecture. The actual line between the various hardware, firmware, and software implementations is actually quite variable but on a steady course to higher VLSI integration providing ever more performance, functionality, small size, lower cost, and higher degrees of reliability than thought possible just a few short years ago.

3 System Overview - Platforms

The VME open bus system and the Apple Macintosh II NuBus system are the two main platforms presently being used to support telemetry data system development and operation under the functional components approach. These systems presently support data rates up to 20 Mbps and a broad range of space telemetry functionality such as synchronization, Reed/Solomon decoding, packet processing, Virtual Channel sorting/multiplexing, data simulation, and high density (256x256) cross-point switching. Architectural stability over the spectrum of supported functionality is required in order to minimize design

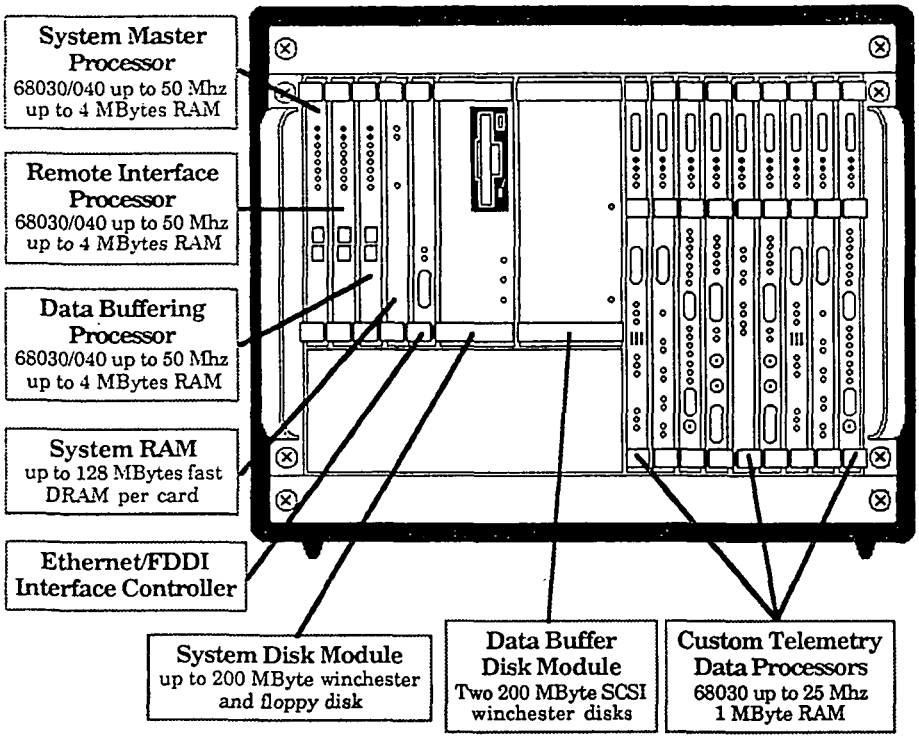

Figure 2: Typical VME platform

changes and error introduction when providing various degrees of performance for the same function. This implies that once a functional component design has been completed and tested, a new design for that same function to increase bit rate (for example) will use the same architecture to the maximum extent possible. As a result of this policy, the architecture and the actual logic design for the Gallium Arsenide (GaAs) frame synchronizer VLSI device (300 Mbps rate) are nearly identical to the Complementary Metal Oxide Semiconductor (CMOS) technology device used to support much lower bit rates (to 50 Mbps). Current activity utilizing both Emitter Coupled Logic (ECL) and GaAs logic gate arrays will provide a general capability to 300 Mbps for virtual channel sorting and multiplexing by June 1991 [2].

The basic configuration of a typical VME system generally consists of:

1. VME standard open bus enclosure, power supply, and a 21 slot backplane including a VME bus (J1 & J2) and a custom telemetry bus (J3);

2. a selection of commercial VME card modules;

3. a selection of GSFC custom cards (A/B/C channel cards);

4. an operator's console;

5. a real-time, multi-processor operating system environment; and

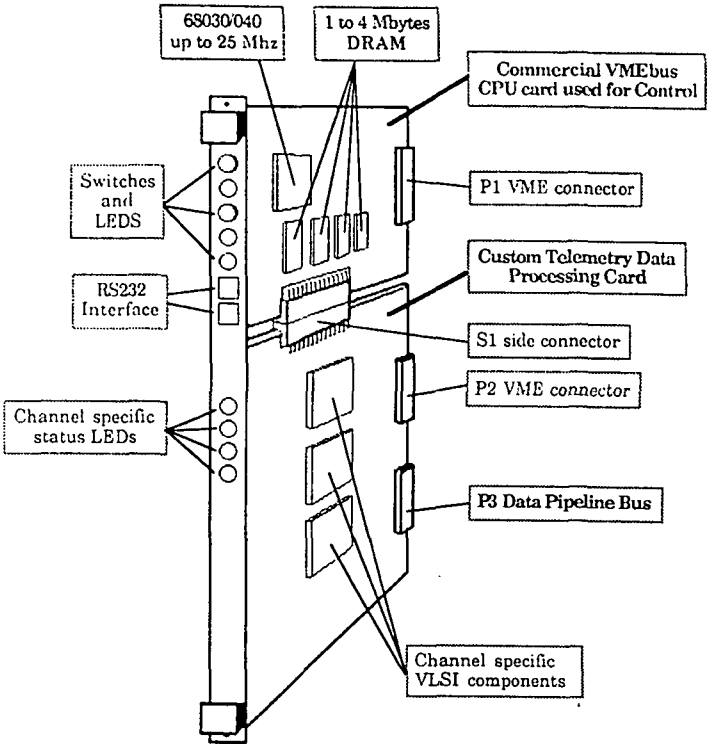

Figure 3: Custom card diagram

6. application programs supporting specific application requirements [3,4].

The physical diagram of a typical VME configuration is shown in Figure 2.

The VME environment and its commercial cards are used to support the general exchange and storage of control, status, and quality data required for local and remote operator console and control operations. These elements provide the general CPU processing, storage, and I/O functions required by the overall system application including the network (e.g. Ethernet, FDDI etc.) or host interface. They also initiate diagnostic tasks and format test results for presentation as required.

Custom cards and the third bus connector (9U VME card cages are used) comprise the telemetry data pipeline. This pipeline allows both incoming and outgoing telemetry data to be processed and transferred through the system without overburdening the VME bus with every data word transfer. This "pipelined" technique allows great flexibility in the processes performed on the telemetry data and as well as for increased data bandwidth. The custom cards directly support the CCSDS standards for telemetry data acquisition and processing through the extensive use of the NASA VLSI Application Specific Integrated Circuits (ASICs) developed at GSFC. Each custom card is composed of a custom hardware section especially designed to implement a particular NASA communication function(s) and a commercial CPU section used for local and global control/status exchange (Figure 3). Some very high performance cards (e.g. packet processor card) require up to three 68020/30 CPU mezzanines (custom built) over a full 9U card area of support hardware. These

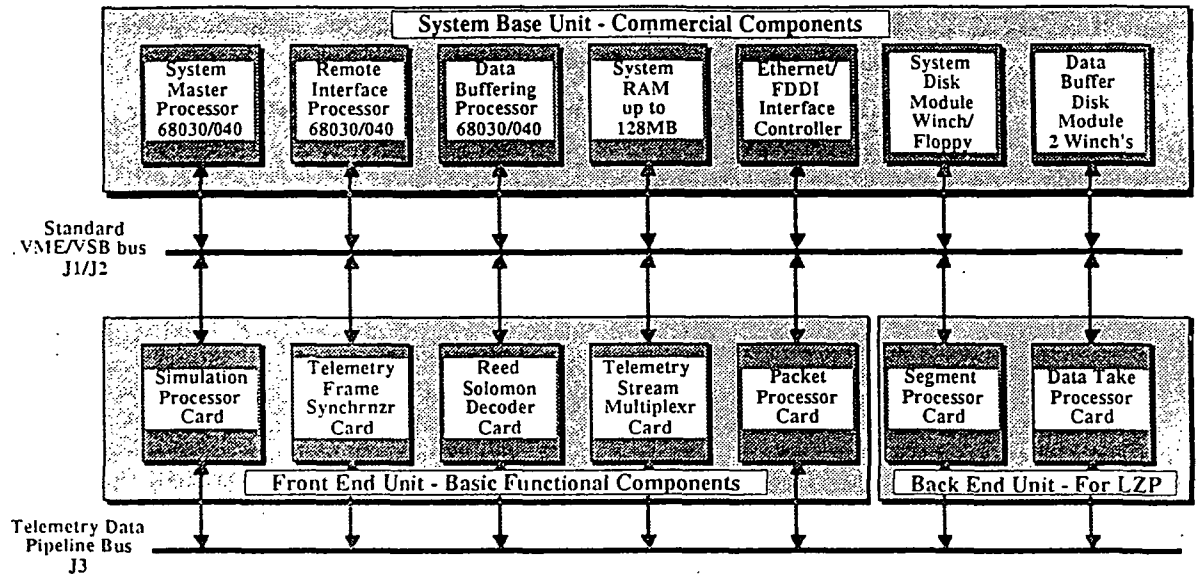

Figure 4: Typical VME block diagram

components and card systems provide programmability to support format or operation variations normally encountered during data capture and process operations.

The block diagram of this general VME implementation shown in Figure 4 demonstrates the flexibility and power available to the developer and to the end user. Because the custom cards adhere to VME/VSB standards and because of the flexibility of the telemetry software environments supporting this system (see SOFTWARE COMPONENTS), virtually any commercial VME card or card system can be easily ported to this environment. The separate telemetry pipeline bus offers great flexibility at the telemetry channel interface. The processing of telemetry data is actually a pipeline process by which telemetry data moves from custom card to custom card as needed to complete the processing requirement. Movement of data into (or out of) the VME/VSB environment for reasons of status, control, or data exchange/processing is possible at every card stage.

The general Macintosh based telemetry system [5], referred to as the Transportable Telemetry Workstation or TTW (see Figure 5), consists of:

1. a custom telemetry backplane bus;
2. a telemetry system software environment including user interface;
3. a Macintosh II/IIx host computer;
4. a MC68020 CPU NuBus card (real-time system controller);
5. a NuBus custom telemetry Synchronizer Card (same GSFC VLSI design as VME); and
6. Additional cards for such functions as Reed/Solomon decoding, Packet Processing, Sorting, and Multiplexing of telemetry data.

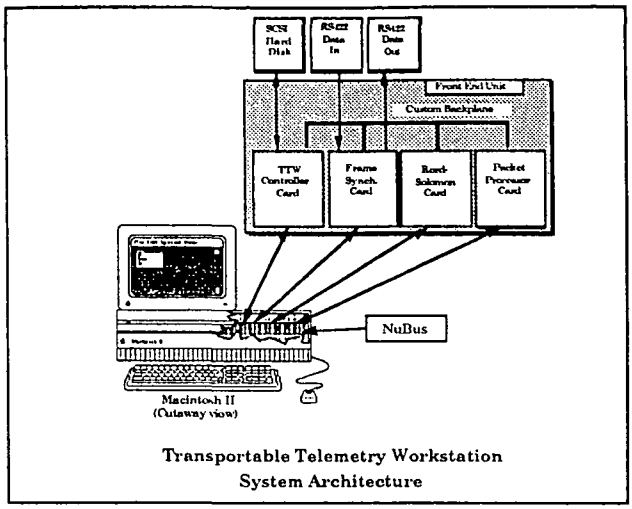

Figure 5: Macintosh Platform

Many of the general comments relating to the VME platform apply also to the TTW system.

The specific hardware (e.g. synchronization, Reed/Solomon decoding, packet processing etc.) associated with a particular functionality is defined within the Computer Aided Engineering (CAE) environment used at GSFC to design, develop, and test these components. The porting of this functionality to yet another platform such as the IBM PS/2 or RS6000 (Micro Channel) is similarly achievable.

4 Hardware Components

All hardware components support NASAs requirement to acquire, process, and distribute space telemetry data. These components include some 12 VLSI Application Specific Integrated Circuits (ASICs) and more than 14 card level components supporting some 11 different projects. These components were developed in-house at GSFC utilizing modern Computer Aided Engineering (CAE) tools. These tools also provide such capability as state-of-the-art Printed Circuit Board (PCB) place and route. The DSTD also supports an in-house fabrication capability including Surface Mount Technology (SMT) to meet limited production of system components. Above the card level, the actual performance level of a particular component (chip, card, or subsystem) is transparent. This implies that the look and feel of the user application and development environments is largely unchanged by bit rate performance requirements.

5 VLSI Components

The VLSI components provide the backbone of the functional component approach and define the actual potential of this approach to provide high performance, low cost telemetry systems. These components currently range in density from 4,400 to 15,000 gates and in-

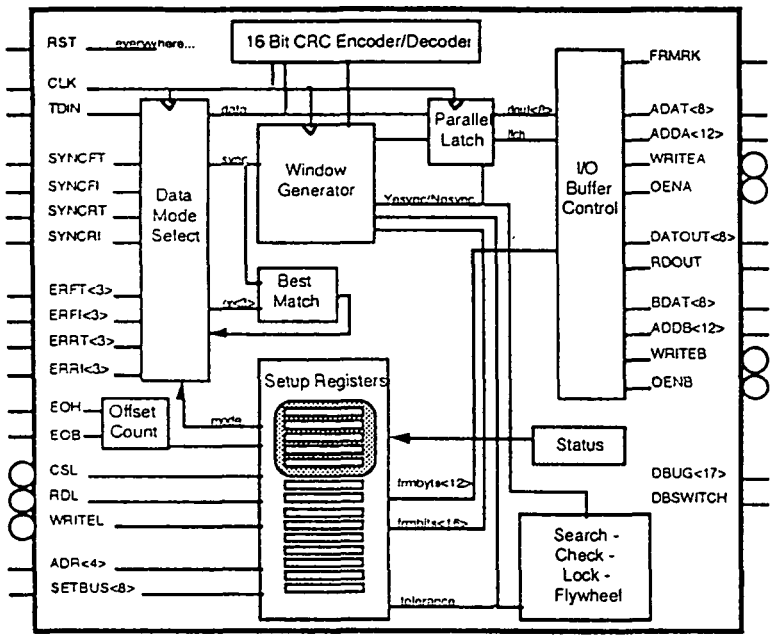

Figure 6: Telemetry frame synchronizer chip II block diagram

clude CMOS, ECL, and GaAs semi-custom arrays and a few full custom chips. To enhance each component's flexibility and reusability (in other designs), all components include such features as a microprocessor interface, a comprehensive set of internal read/write registers, hardware/software resets, and external mode control pins. Figure 6 show the component block diagram of the Telemetry Frame Sync II chip (TFS II). Figure 7 shows the associated register model used to program the TFS II.

The following are typical VLSI components currently used to support general system development:

- **Telemetry Frame Sync Chip (CMOS)**
  - Programmable search, check, lock, and flywheel strategy.
  - Synchronization of frames up to 32 kbits in length.
  - Inversion correction, reversal correction, and slip correction.
  - Double buffering of data.
  - 16 bit CRC check.
  - Microprocessor Controlled.
  - 4,400 gate - 2 micron CMOS.
  - Operation up to 20 Mbits/sec.

- **Telemetry Correlator Chip (CMOS)**
  - Correlation to any synchronization pattern up to 32 bits in length.
  - True and inverted sync indications.
  - Programmable error tolerance for Search, Check, and Lock modes.
  - Independent 22 bit CRC encoder/decoder.
  - Microprocessor Controlled.
  - 4,400 gate - 2 micron CMOS.
  - *Operation up to 20 Mbits/sec.*
- **Telemetry Frame Sync Chip (GaAs)** [1]
  - Programmable search, check, lock, and flywheel strategy.
  - Synchronization of frames up to 32 kbits in length.
  - Inversion correction, reversal correction, and slip correction.
  - Real Time Quality Trailer generation.
  - Double buffering of data.
  - Programmable CRC check.
  - Microprocessor Controlled.
  - 15,000 gate - GaAs.
  - Operation up to 300 Mbits/sec.
- **NASCOM Block Processor chip**
  - Extraction of telemetry data from 4800 bit NASCOM block.
  - Storage of entire 144 bit header.
  - CRC and sequence check.
  - Double buffering for both telemetry and non-telemetry data.
  - Timing signals for end of header and end of block.
  - Microprocessor Controlled.
  - 4,400 gate - 2 micron CMOS.
  - Operation up to 20 Mbits/sec.

Other key VLSI components include a NASA - 36 time decoder (CMOS), Random Access Memory Controller (CMOS), Support Chip for MC68020/30 CPU (CMOS), Correlator Chip (ECL), Test Pattern Generator chip (CMOS & GaAs), and the Tri-buffer Controller Chip (CMOS).

[1] Note: prototypes of this part expected by 10/90.

Over the next two years, the second generation of VLSI components will be designed, developed, and tested. With new array gate counts up to 200,000 for CMOS and 100,000 for GaAs, these components promise the ultimate in integration of functionality, a single chip solution for much of the combinational logic now used on *all* functional component card subsystems. In addition, RAD hard, JAN 38510 gate arrays in the 50,000 plus densities could provide 'ready for flight' chip sets meeting much of NASA's up and down link functional and performance requirements.

6 Board Level Components - The Synchronizer Card

The VME Synchronizer Card is one of 14 card level components designed at GSFC and is an example of the potential power and flexibility of this approach. A general functional block diagram of this card is shown in Figure 8. This 9U VME card consists of a commercial single board computer referred to as the Synchronizer Card Channel Controller (SCCC) and a custom logic card connected via a side connector to the SCCC. The SCCC is a dedicated processor (68020/30 class) which provides setup, self-test, hardware diagnostics, debug, and control over the custom logic card. Extensive software to control the card and provide complete status information has been written and integrated on the SCCC.

The Synchronizer card develops most of its performance from four VLSI chip set s. While operational, these custom chip sets provides the hardware functions necessary to perform high speed NASCOM block processing, telemetry frame synchronization, real time frame trailer appendage, and cumulative quality generation. It can also perform data simulation for self-checking purposes.

The NASCOM Block Processor Chip (NBPCS) Set can be enabled to accept data from either the Data Simulation Chip Set or from the RS422 interface. Data formatted in 4800 bit NASCOM blocks is synchronized to the programmed sync pattern. Two output paths from this subsystem exist. For both paths the header (entire 144 bits) of each NASCOM Block can be read by the SCCC. If data is determined to be telemetry data, the NASCOM Block header and trailer is stripped away and a serial stream of only telemetry bits (fill bits ignored) is output to the Telemetry Frame Synchronizer Chip Set (TFSCS). If the block is determined to be non-telemetry (i.e. command block) the entire block can be transported off the Synchronizer Card. Additionally, the NBPCS performs quality checks on each block received including cyclic redundancy code (CRC) and sequence checks.

The TFSCS can be programmed to select data from either the RS422 interface, the DSCS, or the NBPCS. A complete synchronization strategy can be programmed for the selected data stream. Complete search, check, lock strategies, and slip windowing is provided. Additionally, the TFSCS has the ability to correct inverted or reversed frames. A status word for each frame is reported back to the SCCC. This status word contains sync mode, sync errors, slip indication, CRC errors (if applicable), and data type (forward, reverse, inverted, or true) of frame received. The TFSCS provides complete double buffering of frames and automatic output with framing control signals. Output control strobes allow for automatic status counting.

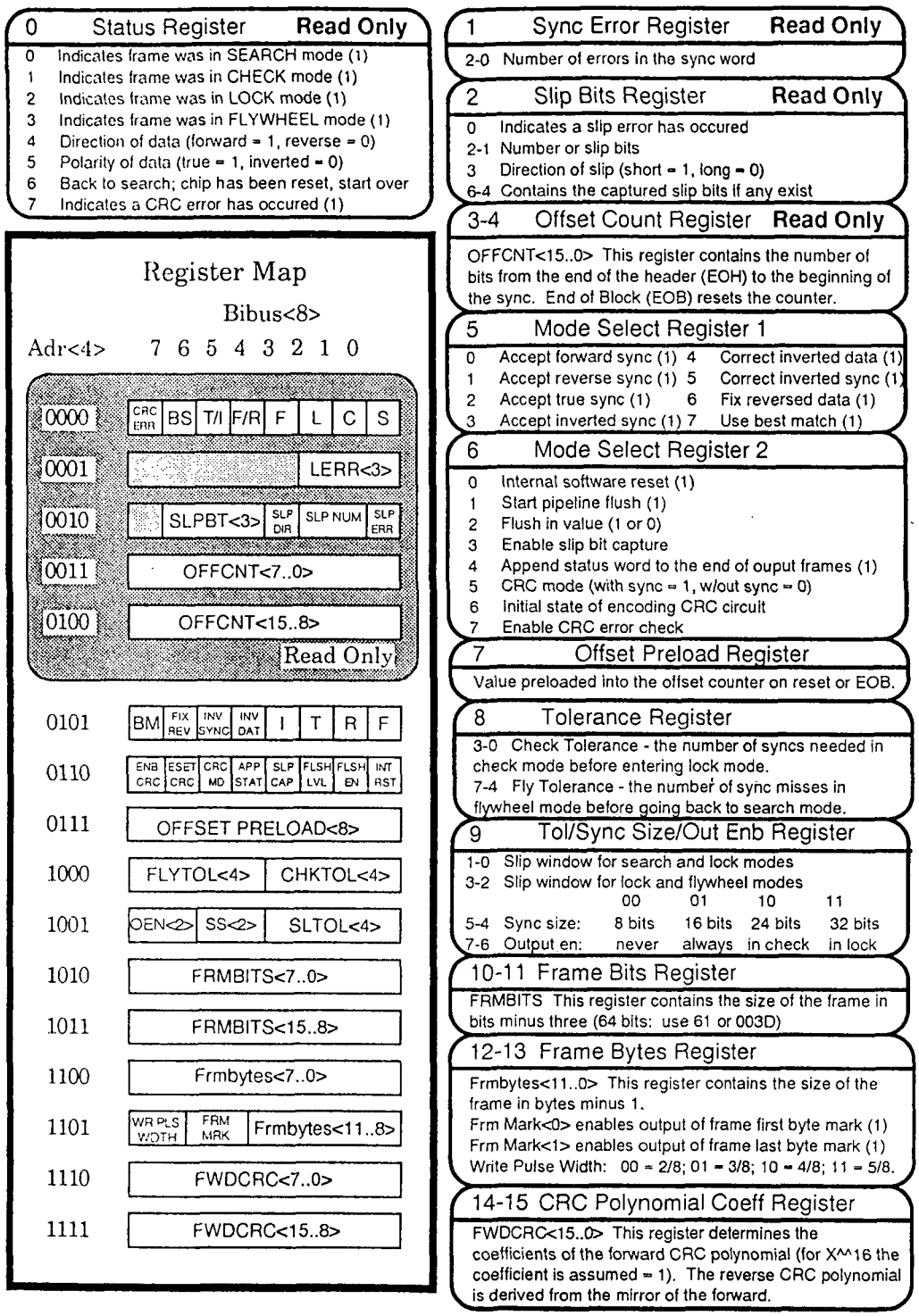

Figure 7: Telemetry frame synchronizer Chip II register model

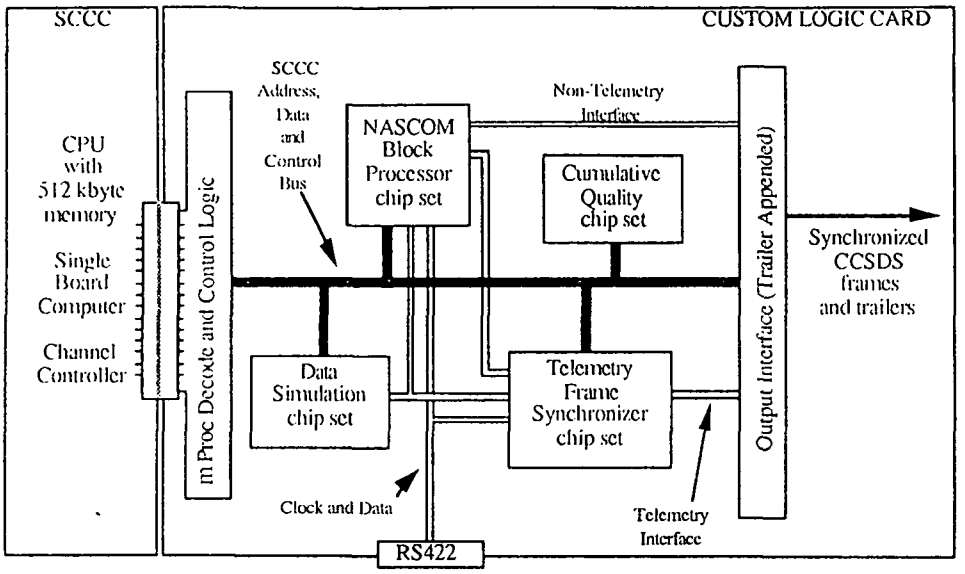

Figure 8: Telemetry frame synchronizer card block diagram

The Output Interface can be programmed to select either the telemetry or non-telemetry data path. Quality data generated by the SCCC is appended to the selected output path. The output interface then controls the transfer of this annotated data to the next processing system.

The Cumulative Quality Chip Set (CQCS) accepts control strobes from the NBPCS, the TFSCS, and from the SCCC and accumulates 32 status counts for up to 24 million events. These counts are read by the SCCC periodically and formatted into a complete status block. The status block can be formatted on a display terminal connected to the SCCC or communicated to a higher level system controller.

The Data Simulation Chip Set (DSCS) provides complete self-testing of all functions on the Synchronizer Card with high speed (up to 20 Mbps) simulation data. Once set up for output, the DSCS can provide independent output of a serial telemetry data stream. This stream can be programmed to provide any type of data (forward, inverted, reverse, or true) in any format for a known number of repetitions. By comparing the status results for the test run against known correct results, the SCCC can determine if the card is functioning correctly before activating it for operational data.

7 Software Components

The design and development of highly functional and flexible telemetry data systems requires the use and application of state-of-the-art real-time software techniques and approaches that are tightly coupled with the high performance VLSI based hardware systems [1,6]. The automated data driven operation of NASA's next generation telemetry data handling systems will require that standard system functions be virtually turnkey in every aspect of their operation. These system characteristics for the VME platform

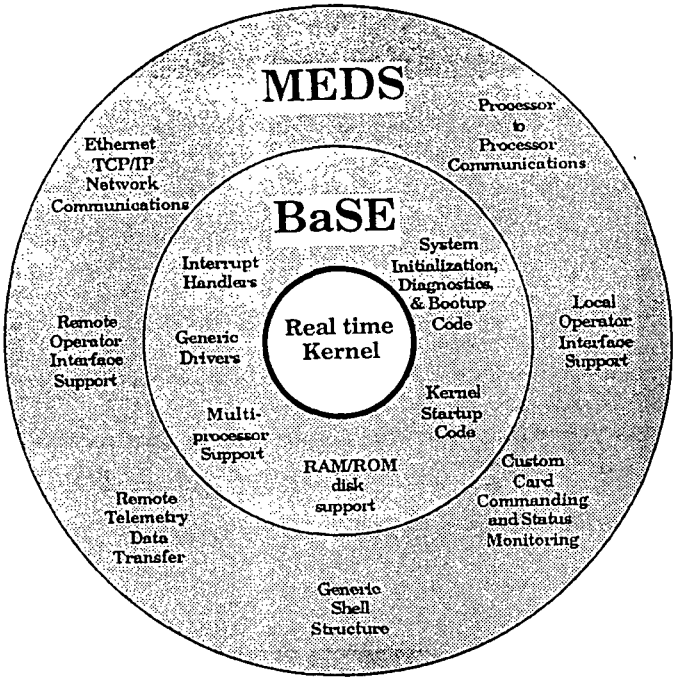

Figure 9: Software Environment

are provided by a commercial high performance real-time Operating System, and by three software systems environments referred to as the Base System Environment (BaSE), Modular Environment for Data Systems (MEDS), and the Network Environment for Telemetry Systems (NETS) (see Figure 9).

The BaSE allows system interactions between various manufacturer's cards to be prototyped, explored, and tested before being placed into operational use. Using the MEDS software package, application specific real-time code has a strong modular foundation that begins with a generic multiprocessing shell and supports the basic software functions needed by all multiprocessor based telemetry data systems. While still in the conceptual stage of development at GSFC, NETS is intended to ease the design and development effort required to integrate various telemetry data system nodes (e.g. VME capture system, workstation data processor, workstation host center controller etc.) using wide and local area networks.

Current development work with the Macintosh II NuBus system will also include the basic 'look and feel' of the original VME Base and MEDS environments.

8 BaSE Software Environment

NASA's next generation telemetry systems must provide a fairly simple and fast path to future enhancements. For this reason, the electronics hardware is based around widely supported open bus systems such as VME and NuBus. Likewise, a versatile software environment that can support the flexibility of these systems is also required. The telemetry system environment must contain enough intelligence to automatically configure the sys-

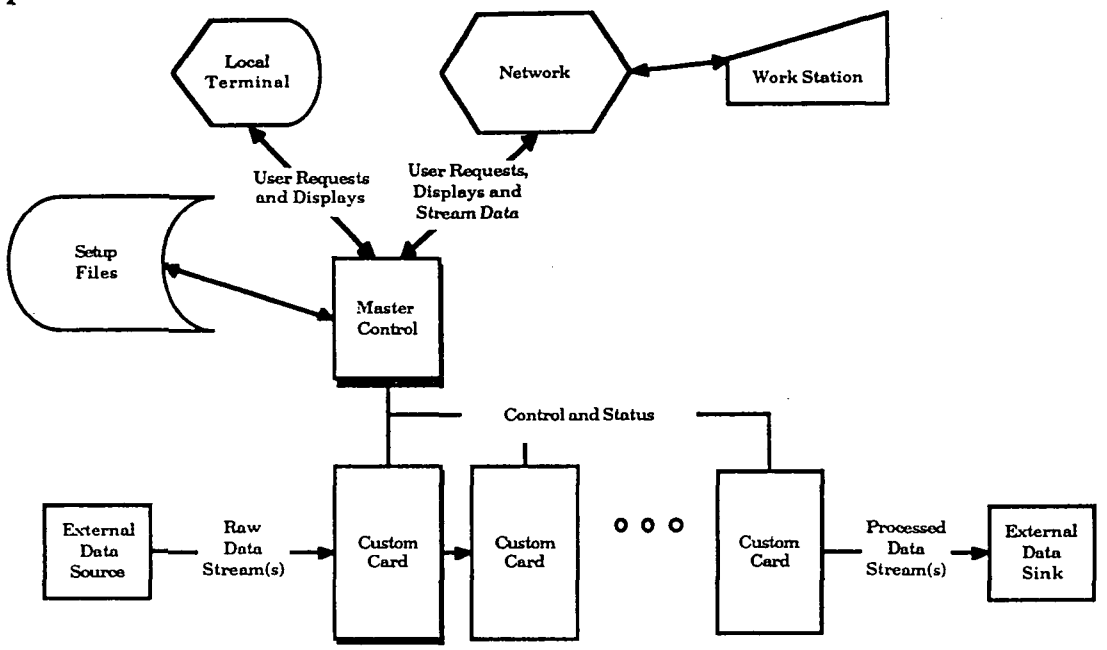

Figure 10: MEDS General Flow Diagram

tem based on a (often) changing hardware environment. It is the goal of BaSE to provide this type of functionality for the VME system and to provide the 'basic' interface at the bus hardware/software interface level. Through the use of BaSE, the systems developer can easily apply (or port) any of the hundreds of commercial cards or card systems to his particular telemetry system configuration. Once incorporated into the functional components BaSE environment, this new card(s) is compatible with and potentially part of any functional components system. BaSE allows the designer to pick and choose those VMEbus based products that best fit the application at hand and it ensures that products from different manufacturers can be used together in a *plug and play* fashion. BaSE allows seamless integration of NASA's custom VMEbus based telemetry processing cards into the operational environment and provides a single environment that is used for all phases of system development. BaSE is used from initial hardware test and checkout, to system software development, to final operational system deployment. The goal of the BaSE environment is to provide a cost effective telemetry platform environment that is generic in both hardware and software, can be used for both development and operations, and can be quickly and easily tailored to meet changing system needs.

9 MEDS

The systems based on the functional components approach all have a similar pipelined, multiprocessor, and dual bus hardware architecture, as a platform on which to build application specific hardware. When designing software for such a system, there are many questions which need to be answered. What data and parameters will each processor need to accomplish its job? How will the processors communicate with each other? And with the operator? How can the total job be subdivided into tasks? On which processors will

they run? What data and parameters will each task need to accomplish its job? How will each task get data? The Modular Environment for Data Systems (MEDS) was developed to help answer these questions and give an application programmer a starting point for designing a system based on the standard hardware platform. While BaSE provides a generic commonality at the bus hardware/software interface level, MEDS provides the knowledge of a "space telemetry data system". This implies the knowledge that a particular card system, say a Reed/Solomon decoder card, is not just a set of command, status, and data registers mapped in a specific memory space on the VME bus. Indeed, MEDS provide system wide knowledge that this card is in fact a Reed/Solomon decoder card with specific data in specific location with specific rules for formatting, debugging, controlling, user interface etc.

The Modular Environment for Data Systems (MEDS) is designed as a general purpose software platform which is expanded and customized by application programmers to suit their particular requirements. It supports the basic software functions needed in all systems, namely, the ability to setup application specific hardware and software, process the telemetry data based on the setup style parameters, monitor the processing and supply network support for remote operator interface and data transfer. MEDS supplies an infrastructure to pass data between systems, processors and tasks as well as support for operator interface development. A complete system is built by adding custom code to the general purpose MEDS code. Therefore MEDS spares the application developer from the burden of creating an infrastructure for each new system and adds consistency in all system design, implementation and maintenance.

A MEDS based system unites and manages the standard multiple processor hardware platform. The processors are organized as a single master processor directing multiple subordinate application specific custom cards (Figure 10). The master processor is the single point of control within the system; it interfaces with the operator, on either a local terminal or a remote workstation. Using a set of operator defined setup files, the master processor will initialize the custom cards and monitor their processing on various status pages. Telemetry data may enter and exit the system via the remote interface as well. In any case, it is the pipeline of custom cards that actually process the telemetry data.

The MEDS software resides mainly on the master with cooperating software running on each custom card. The basic MEDS functions include:

- Setup system and subsystems for processing (e.g. setup VLSI chip registers).
- Control the application specific processing (e.g. enable, disable, reset a card).
- Monitor the system and subsystems (e.g. gather and display card processing status).
- Stream data transfer over network (e.g transfer telemetry data to/from a workstation).

10 NETS

The Network Environment for Telemetry Systems (NETS), although currently in the conceptual design phase, is forseen as one of the basic software environments supporting the functional component concept. NETS will provide for the control and management of multiple telemetry processing systems in a data processing facility. Similar to MEDS in architecture, NETS will provide a set of standard functions for communicating, scheduling and configuring telemetry processing systems distributed in a facility through a shared commercial network such as Ethernet or FDDI.

11 Level Zero Processor Project

The Level Zero Processor (LZP) now under development at GSFC [7] is an excellent example of the real benefits of the VLSI functional components approach. A primary goal of NASA's space and ground network system is to make the telemetry data transportation transparent to the customer so that it seems like the experimenter's instrument is within his own facility. This requires a process to remove from delivered data products all artifacts and disturbances introduced during data transport. This type of processing is a key part of what is usually referred to as the Level Zero Processing. In addition to error decoding and correcting functions to eliminate bit errors, the LZP system restores the order of data for a given observation or collection period. Such a data set is called a datatake and is delivered to the customer as a LZP product. Restoration of a datatake requires one or all of the following four basic LZP functions to be performed:

1. reassembling user packets from Virtual Channel Data Units (VCDUs);
2. reversing "backward" playback data;
3. merging together real-time data and playback data with proper time order; and
4. deleting redundant data due to the overlap between real-time and playback data.

To perform these tasks a new processing algorithm and a new architecture for the LZP system utilizing VLSI technologies has been proposed by the DSTD at GSF C. The new LZP system is based on the functional components approach and utilizes the VMEbus with multiple microprocessors running concurrently. The telemetry data will be processed for datatakes by various microprocessors and custom VLSI controllers while flowing through a data pipeline. Disk farms will be used as mass storage to buffer data for up to three orbits. This LZP will use Consultative Committee for Space Data Systems (CCSDS) Recommendations for format standards. The LZP system will provide standard services for taking data either NASCOM blocked or in synchronous VCDU format into a time ordered data take for delivery to the experimenter or customer.

The system will operate in three non-exclusive operation modes: real-time, quick-look, and production processing. In the real-time processing mode, customer's packets are

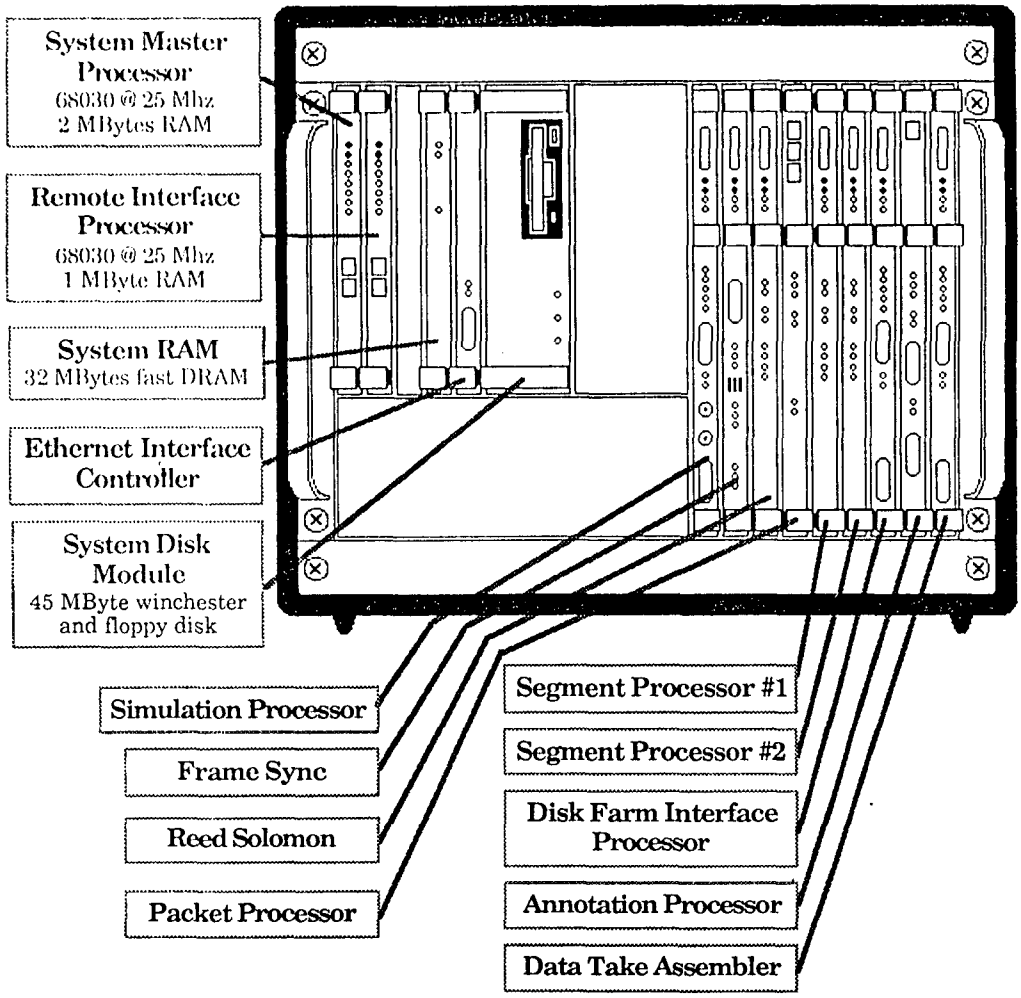

Figure 11: Level Zero Processor Physical Diagram

transmitted as soon as each packet is received and reassembled. Also, the data is retained for normal production. In the quick-look processing mode, a higher priority subset of the datatake will be made available to the customer. No redundancy deletion is performed. Again, the data is retained for normal production. The final and most important mode is the production processing mode. In this mode, data is processed and grouped into datatakes specified by the customer through scheduling tables. The completed datatake could be available to the customers within 90 minutes after receiving the last source packet from the datatake. (Additional factors in the actual throughput of the system include the time required to completely dump on-board recorder data and the performance achievable in the output product management and distribution.)

The LZP system can perform annotation of packets and datatakes, including data quality and accounting functions. Moreover, the catalog files for system operation, quality and production are maintained.

The following assumptions have been made for the processing environment in which the proposed LZP system will operate:

- All data complies with the Consultative Committee for Space Data Syste ms (CCSDS) Recommendations [2]. This implies that packetized data of multiple sources are transferred through multiple virtual channels.
- Maximum data rate of any virtual channel is 150 Mbps and total data from one virtual channel over one orbit does not exceed 15 Gbytes.
- Orbit time of the platforms is 90 minutes, 2/3 of which being daylight and 1/3 of which darkness and all data from one orbit will be received by the end of following orbit.
- Data packets bearing the same source ID have fixed length and the average packet size is 1 Kbytes, or 8 Kbits.
- The number of real-time and playback data segments is less than 1000 for each source per orbit.

Level Zero Processing is a two stage process. In the first stage, a serial data stream is assembled into user packets and then stored in a mass storage buffer. The bit ordering of playback data within each packet is corrected. In the second stage, packets are sorted according to their source IDs and time sequence, and grouped together to form datatakes.

In order to achieve high speed and low cost, the functional VLSI component approach was used in designing the LZP system. Figure 4 depicts the general system functional block diagram for the LZP system (mass storage system not shown). Implemented with commercial microcomputer modules and custom telemetry data processing cards, the LZP system consists of three subsystems: the front-end VLSI data capture system, the Datatake Processor, and the Mass Storage Subsystem. Figure 11 shows the actual physical diagram of the telemetry data system portion of the LZP without the required mass storage components.

12 Conclusion

To fully develop the potential of future space activities, NASA's telemetry data systems must do more than simply meet specific technical requirements. They must provide for reliable, low cost, and modular systems which NASA and its user community can tailor in size and performance to particular needs. These systems must allow for growth and expansion in the years to come. Also, with the push toward an automated data driven operation of NASA's next generation telemetry data handling systems, it is important that standard system functional components be virtually turnkey in operation. Even though these systems are a tightly integrated mix of hardware and software elements, they are but single elements in a large, highly complex NASCOM telemetry data handling system. The functional component approach was designed to meet these needs.

References

[1] Hand, S. and Sabia, S., "Functional Component Approach to Telemetry Data Capture Systems," VOL XXIV, *Proceedings of the International Telemetering Conference*, Las Vegas, Nevada, October 1988.

[2] Shi, J., Grebowsky, G., Horner, W., "Prototype Architecture for a VLSI Level Zero Processing System," VOL XXV, *Proceedings of the International Telemetering Conference*, San Diego, CA, October 1989.

Nomenclature

ASIC	Application Specific Integrated Circuit
BaSE	Base System Environment
CAE	Computer Aided Engineering
CCITT	Consultative Committee for International Telegraph and Telephone
CCSDS	Consultative Committee for Space Data Systems
CMOS	Complimentary Metal Oxide Semiconductor
CQCS	Cumulative Quality Chip Set
CRC	Cyclic Redundancy Check
CVCDU	Coded Virtual Channel Data Unit
DSCS	Data Simulation Chip Set
DSTD	Data Systems Technology Division
ECL	Emitter Coupled Logic
FDDI	Fiber Data Distribution Interface
GaAs	Gallium Arenide
GSFC	Goddard Space Flight Center
ISO	International Organization for Standards
LZP	Level Zero Processor
MEDS	Modular Environment for Data Systems
NASCOM	NASA Communications
NBPCS	NASCOM Block Processor Chip Set
NETS	Network Environment for Telemetry Systems
PCB	Printed Circuit Board
SCCC	Synchronizer Card Channel Controller
SMT	Surface Mount Technology
TFSCS	Telemetry Frame Synchronization Chip Set
TFS II	Telemetry Frame Synchronization Chip II
VCDU	Virtual Channel Data Unit
VLSI	Very Large Scale Integration