ABSTRACT

While the complete process of preparing telemetry data for delivery to NASA’s customers requires a number of steps or levels (level 0,1,2...etc.), the initial processing, generally referred to as Level Zero Processing (LZP), poses a real technical challenge for NASA in the 1990s Space Station Freedom era. This challenge is the result of requirements to provide real-time or near real-time LZP products at rates up to 150 Mbps. In addition, increased use of sophisticated data formats places even more demand on NASA’s future ground telemetry data systems.

A new high speed Level Zero Processing system approach has been proposed for the Space Station Information System. By using a new processing algorithm, the complexity of database management necessary for reconstructing payload data streams has been significantly reduced. This new approach is based on the state-of-art distributed real-time processing and VLSI hardware systems and offers potential processing rates up to 150 Mbits per second (Mbps).

1. INTRODUCTION

In the Space Station Freedom era, telemetry data will be transported from payloads to the ground and be distributed to customers by the Space Station Information System (SSIS) [1]. One primary goal of the SSIS system is to make the data transportation transparent to the customer so that he interacts with his payload as if the payload were at his facility. This requires a process to remove from delivered data products all artifacts and disturbances introduced during data transport through the SSIS. This type of processing is a key part of what is usually referred to as the Level Zero Processing (LZP).
Besides error decoding and correcting functions to eliminate bit errors, the main object of the LZP system is to restore 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.

Traditionally, such tedious LZP tasks are performed by software running on main frame computers. To meet NASA’s needs for drastically increased data speed and volume in the Space Station Freedom era, Goddard Space Flight Center’s (GSFC) Data Systems Technology Division has proposed a new processing algorithm and a new architecture for the LZP system utilizing VLSI technologies. The new LZP system is based on 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 paper will discuss the LZP processing scenario and algorithm, followed by descriptions of system architecture, implementation and components.

2. LZP PROCESSING

2.1 PROCESSING REQUIREMENT

The proposed LZP system will support all four basic LZP functions defined in the previous section. It will use Consultative Committee for Space Data Systems (CCSDS) Recommendations [2] for format standards. The operation of the LZP system will be schedule-driven. The customers specify datatakes by source IDs and observation periods for each orbit. The input to the system can be serial data in NASCOM block format, or synchronized VCDUs or Coded VCDUs (CVCDUs). The outputs of the system are datatakes for delivery to the customers. The actual interfaces for both input/output to/from the LZP, are not well defined as yet will not be covered by this paper.

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 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 will be available to the customers within 90 minutes after the capture of data.

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.

2.2 PROCESSING ENVIRONMENT

Though the basic definition about LZP processing has been widely agreed upon, the Space Station Information System is not well defined as yet. Based on a study of research papers on the subject \[1\][3][4][5], The following assumptions have been made for the processing environment in which the proposed LZP system will operate:

1. All data complies with the Consultative Committee for Space Data Systems (CCSDS) Recommendations [2]. This implies that packetized data of multiple sources are transferred through multiple virtual channels.

2. 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.

3. 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.

4. Data packets bearing the same source ID have fixed length and the average packet size is 1 Kbytes, or 8 Kbits.

5. The number of swaps between real-time and playback data per orbit is less than 1000 for each source.

2.3 PROCESSING ALGORITHM

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.

With the data rate of 150 Mbps and average packet size of 8 Kbit, a 15% duty cycle operation can generate 15 million packets each orbit. To trace every packet in the buffer
would result in a huge database in the magnitude of Gbytes. The management of such a large database would be an extremely cumbersome and time consuming task, even for a high performance main frame type computer. Therefore, a new processing algorithm is employed that significantly reduces the size of the database. One key is to define a new class of data unit called the data segment. A data segment is a group of user packets that have identical source IDs, data direction, and continuous time sequence. The other key is to presort packets from different sources in separate logical buffers. Accordingly, the disk farm storage system is partitioned into a number of logical buffers. Each buffer is dynamically sized to accommodate maximum data load generated by the source for up to three orbits.

User packets from various source IDs are saved in corresponding logical buffers as data segments. Any change in the data stream that results in a reversal of data direction or a break in the time sequence for a source will result in the closing of the current data segment, opening a new data segment, and an entry into the data segment’s database. Because this type of action occurs relatively infrequently, the size of data segment is significantly larger than that of packet. It is reasonable to assume there are at least 1000 packets per segment on average. Thus, the data segment database will be reduced to the level of megabytes, which is easily handled by microcomputers. Because the size of packets from the same source is a constant and the starting address of a packet is a known offset from the beginning of data segment, individual packets in a data segment are easily retrievable. The offset is calculated from the packet sequence number in that segment and the packet size.

All processes in the second stage LZP are greatly simplified, because now they are performed on a data segment basis rather than on individual packets. The typical derivation of an assembly table from which a datatake will be grouped is illustrated in Figure 1. Suppose that a payload generates 10,000 packets during one observation in an orbit and its user requests this group of data as one datatake. During downlink transmission, The data is broken into 5 segments. Three segments, packets 1-2100, 4000-6100 and 9000-10000, are transmitted in real time, while the rest, packets 2000-4100 and 6000-9100, are recorded and played back later.

On the ground, all data is received and ingested into the LZP system. Through the first stage processing, the data is reassembled into five data segments and saved in the disk buffer. A segment index table is established with one entry for each segment. Meanwhile an assembly table is derived as well. When the first segment (packets 1-2100) arrives, its index is put into the first entry of the assembly table. The second index for packets 4000-6100 goes to the third entry, leaving the second entry to be filled by the playback data later. Similarly the index for packets 9000-10000 fills in the fifth entry of the table. Then comes the playback data. Arriving in reversed time order, the segment of packets 6000-
9100 gets in first. Its index will naturally fill into the last gap left in the table, which is the fourth entry. The index for last segment then fills up the gap in entry 2.

Because real-time data always comes in forward time order and playback data in reversed time order, the above example shows that the order of data segments normally leads them to the appropriate position in the assembly table of the segment database. Occasionally, sorting operations are required when the coming segment does not fit the gap or there is more than one gap for that entry. There is plenty of time to perform this sort if needed between data segments.

Once all entries in the assembly table are filled, overlap is checked at segment boundaries. Redundant data (packets 2000-2100, 6000-6100 and 9000-9100) will be deleted on packet basis by using the quality criteria. When the check is completed, the final phase of datatake assembly begins. From the top of assembly table, data segments are read out of the disk buffer and sent out. The complete playback reversal is done in this phase. Packets in a playback data segment will be read out in reversed order, i.e., the last packet received will be read out first and the first packet received will be read out last.
3. THE IMPLEMENTATION OF LZP SYSTEM

3.1 SYSTEM CONFIGURATION

In order to achieve high speed and low cost with VLSI technology, the functional VLSI component approach [6] is taken in designing the LZP system. Implemented with commercial microcomputer modules and custom telemetry data processing cards, the LZP system consists of two subsystems: the Datatake Processor and the Mass Storage Subsystem. Figure 2 depicts the system functional block diagram.

The Datatake Processor is based on a dual bus multiprocessor architecture housed in a 9U VME rack, as shown in Figure 3. The communication between microprocessors is through the VMEbus, while the telemetry data flows through a custom data pipeline bus. As illustrated in Figure 2, the configuration of the Datatake Processor includes three processing units: the System Base Unit, the Front End Unit, and the Back End Unit.
The Mass Storage Subsystem has two disk buffers, one for data and one for annotation. They are implemented by commercial disk farms with maximum capacity of 45 GBytes and data rate up to 128 Mbps. The disk buffers are dynamically divided into partitions, each of which is dedicated to one source. The disk partitions will be discussed further in the next section.

3.2 SYSTEM OPERATION

The System Base Unit is responsible for overall system control and serves as the system master. Through the use of the Base System Environment (BaSE) and the Modular Environment for Data Systems (MEDS) (GSFC developed system software packages)[9], the operator interfaces with the System Base Unit and controls the LZP operations by sending commands and gathering status to and from other system components through it.
In addition, the System Base Unit maintains the operational database such as quality and accounting data files and production catalog files. It also provides system disk and memory, as well as the Ethernet interface linking to remote terminals or work stations.

The Front End Unit is responsible for reassembling user packets from serial data in NASCOM block format. It consists of the Frame Synchronizer, Reed-Solomon Decoder, Packet Processor and Data Simulator. The serial telemetry data is ingested to the Frame Synchronizer and synchronized to VCDUs or CVCDUs. Reed-Solomon error decoding is done when the CVCDUs are passed through the Reed-Solomon Decoder card. At the next processing node, the Packet Processor strips off VCDU headers and trailers, reassembles user packets, reverses “backward” data for playback packets, and sends them to the Back End Unit. The Packet Processor also generates quality and accounting data and annotates the packets. The Data Simulator simulates NASCOM blocks and VCDUs/CVCDUs used for system test and diagnostics.

The major function of the Back End Unit is to reassemble the specified datatakes from user packets. The three processing modules in the unit are the Segment Processor, Annotation Processor, and Output Processor. The Segment Processor separates the annotation data and user packets, and sorts them according to their source IDs in a local memory buffer. When the data in any local memory buffer reaches a predetermined size, that memory buffer will be transferred to the disk farm storage system into a partition allocated for that source ID. Meanwhile the Segment Processor monitors the direction and space time of each packet to define data segments. If a change in data direction or a gap in space time is detected, the Segment Processor will terminate the current data segment, make a new entry in the Segment Index Table and start a new segment with the new packet.

The Annotation Processor reads any new entry in the Segment Index Table and compares it with its current Datatake Request Table. If the new data segment is recognized as a part of a requested datatake, then its index will be inserted into the corresponding Datatake Assembly Table following the algorithm described in Section 2.3. When all data has been received for a datatake, the processor deletes redundant packets from the assembly table. The complete Datatake Assembly Table is then passed to the Output Processor. The other function of the Annotation Processor is to sort source packets that are designated as real-time from the Packet Processor and to pass them through for immediate output.

The Output Processor uses the Datatake Assembly Table and Segment Index Table to assemble a datatake. The first table specifies data pieces in the datatake and the second table is used to compute the addresses of these data pieces on the disk farm storage system. The data is transferred from the disk, block by block, to a local buffer, then from the local buffer to the output port. In this way, the data can be selectively output, e.g., to
delete redundant data. If the data segment to be output contains playback data, i.e., all packets in that segment are in reversed time order, the Output Processor restores the order simply by transferring the last-received packet first and the first-received packet last. Furthermore, the Output Processor generates datatake annotations and attaches them to the datatakes as either headers or trailers. If requested, the packet annotations can also be output along with corresponding packets.

### 3.3 DATA/ANNOTATION FILE ORGANIZATION

To reduce the magnitude of database management, the data from different sources and orbits is saved in different logical buffers or source partitions on the disk farm storage system. As shown in Figure 4, the storage system will be partitioned according to the number of sources and volume of data. Each source partition serves one source and is dynamically sized to hold data at maximum rate for three orbits.

The source partition is further divided into three orbit partitions which operate like ring buffers. The decision to choose three orbits as the logical buffer size is based on two assumptions. First, as stated in Section 2.2, all data from one orbit will be received at the LZP system by the end of the following orbit. Second, all data of one orbit can be processed and sent out within 90 minutes, i.e., one orbit time. Thus at any given time, new data will be written into one orbit partition, while data in the other two partitions are being processed. If data from one orbit cannot be received by the end of following orbit, then the size of source partition should be increased accordingly.

Inside each orbit partition, there are data segments which contain packets formatted into fixed size disk farm storage system blocks. All packets in a segment have unique direction and size and follow the time order strictly. The time span of a segment is defined by the space time of the first and last packets of that segment. Time spans of various segments should be exclusive, except for overlap data.

### 4. VLSI SYSTEM COMPONENTS

To facilitate the understanding of the LZP operations discussed in Section 3.2, brief descriptions of major system components are given below. Except for the System Base Unit, all data processing cards are designed and implemented by the Data Systems Technology Division. At present all cards are fabricated with CMOS technology and their data rates range from 30 to 80 Mbps. ECL based versions of these cards are under development and will run at up to 300 Mbps.
4.1 SYSTEM BASE UNIT

The System Base Unit consists of a VMEbus CPU card with a 25 MHz Motorola MC68020/030 processor and 512 to 2048 KBytes of no-wait-state dual-ported SRAM; a system disk module with a 45 MByte Winchester, floppy drive, and disk controller; a 4 MByte SRAM card; and an Ethernet Interface card. The integration of these high-performance commercial products uses the Base System Environment (BaSE) software package [9] and provides a powerful engine for system functions.
4.2 FRAME SYNCHRONIZER

The Frame Synchronizer card [7] performs generic NASA Communication (NASCOM) block processing, telemetry VCDU synchronization, real-time quality trailer appendage, and cumulative quality statistic generation functions. It utilizes four distinct sets of GFSC developed, VLSI semi-custom and custom chips to accomplish its functions.

4.3 REED-SOLOMON DECODER

The primary function of the Reed-Solomon Decoder is to perform error detection and correction on CVCDUs. The decoder either sends out completely corrected VCDUs or non-correctable yet still routable VCDUs, or filters (deletes) non-routable VCDUs. The Reed-Solomon Decoder utilizes a custom VLSI controller and a LSI Reed-Solomon decoding chip set for hardware processing. Two 20 MHz microcontrollers are implemented to assist the microprocessor in software processing.

4.4 PACKET PROCESSOR

The Packet Processor [8] inputs synchronized error corrected telemetry VCDUs and outputs annotated user packets derived from the VCDUs. Up to 8 Virtual Channels and 64 sources can be processed concurrently by the processor. It uses two GFSC developed semi-custom VLSI controllers and three Motorola MC68020 microprocessors to support the implementation of the multi-channel multi-source packet processing. The programmability of processing tasks executed on the three microprocessors makes the processor adaptable to different data formats. A specially designed data pipeline with two separate RAM systems yields high data throughput.

4.5 SEGMENT PROCESSOR

The Segment Processor separates, sorts and buffers data and annotation of the user packets for up to 24 sources. It moves the data and annotation to the appropriate disk partitions. The data segments are recognized and the Segment Index Table is maintained. The Segment Processor also provides primary control functions for the Disk Farm Mass Storage Subsystem.

4.6 DATA MOVER

Both the Annotation Processor and the Output Processor are adaptations of the generic Data Mover card which moves data between six ports: the data pipeline, VMEbus, VSBbus, two interfaces for the Disk Farm Mass Storage Subsystem and the CPU bus. In addition, there are two Dual Ported Ram buffers used for data manipulation during
transfers. A programmable 20 MHz microcontroller is used at each port to control the data routing from port to port. Multiple channels may be used simultaneously, and more than one port may output the data received from a single port.

5. DISK FARM MASS STORAGE SUBSYSTEM

The mass storage requirements for the prototype LZP system described above include sustained transfer rates above 100 Mbps, random access storage up to 45 GBytes, and multi read/write ports. These requirements are minimal, however, when compared to the potential expected rates of up to 300 Mbps and storage capacity beyond 100 GBytes. Only a few years ago, storage systems which could support these requirements were either not available or very expensive, one of a kind, research tools.

Today a number of vendors offer systems which will (or soon will) meet these high performance requirements. These systems utilize parallel disk arrays, often referred to as “disk farms”, to achieve various combinations of high speed, density, flexibility, and reliability [10]. Figure 5 outlines the block diagram and some of the basic requirements and characteristics which can be expected from state-of-the-art parallel disk controllers/systems and the type of functional and performance required for the above prototype LZP system. The term Kernel refers to a basic multiple disk farm configuration.

The complexity and performance requirements of advanced telemetry processing systems (such as this LZP system) can still require the use of multiple disk farm arrays (kernels). Figure 6 demonstrates one such multiple disk farm array basic kernel configuration and its functional and performance characteristics.

6. CONCLUSIONS

The prototype architecture and implementation of a high speed Level Zero Processing system has been discussed. Because of the new processing algorithm and VLSI technology, the prototype LZP system features compact size, low cost, high processing throughput, easy maintainability and increased reliability. Though extensive control functions have been done by hardware, the programmability of processing tasks makes it possible to adapt the system to different data formats and processing requirements.

The single LZP system can handle up to 8 Virtual Channels and 24 sources with combined data volume of 15 Gbytes per orbit. For greater demands, multiple LZP systems can be configured in parallel, each being called a processing channel and assigned a subset of Virtual Channels. The telemetry data stream will be steered into different processing channels in accordance of their Virtual Channel IDs. This super system can cope with virtually unlimited number of Virtual Channels and sources.
The proposed LZP system will have the data throughput of 120 Mbps with ECL versions of VLSI data processing cards. This limitation is imposed by the data transfer rate of current commercial disk farms. In the near future, it is expected that new disk farms with data rate exceeding 150 Mbps will be available from commercial vendors due to the advance in the disk drive technology. Therefore, the data rate of the LZP system will reach 150 Mbps or more with new faster disk farms.
REFERENCES


NOMENCLATURE

CCSDS Consultative Committee for Space Data Systems
CDOS Customer Data Operation Service
CMOS Complimentary Metal Oxide Semiconductor
CPU Central Processing Unit
CVCDU Coded Virtual Channel Data Unit
ECL Emitter Coupled Logic
GSFC Goddard Space Flight Center
LSI Large Scale Integration
LZP Level Zero Processing
Kbit Kilobit
Kbyte Kilobyte
Mbps Megabit per second
Mbyte Megabyte
NASA National Aeronautics and Space Administration
NASCOM NASA Communication
RAM Random Access Memory
SSIS Space Station Information System
VCDU Virtual Channel Data Unit
VLSI Very Large Scale Integration
VME Versabus Module Eurocard
VSB VME Subsystem Bus