IMPLEMENTATION OF A VLSI LEVEL ZERO PROCESSING SYSTEM UTILIZING THE FUNCTIONAL COMPONENT APPROACH

Jianfei Shi
RMS Technologies, Inc.
Code 520.9

Ward P. Horner
Gerald J. Grebowsky
James R. Chesney
Data Systems Technology Division, Code 520

Mission Operations and Data Systems Directorate
NASA/Goddard Space Flight Center
Greenbelt, Maryland 20771

ABSTRACT

A high rate Level Zero Processing system is currently being prototyped at NASA/Goddard Space Flight Center (GSFC). Based on state-of-the-art VLSI technology and the functional component approach, the new system promises capabilities of handling multiple Virtual Channels and Applications with a combined data rate of up to 20 Megabits per second (Mbps) at low cost.

1. INTRODUCTION

Level Zero Processing (LZP) is a core function in NASA's return link telemetry data processing systems. Its main objective is to remove all artifacts and disturbances from delivered data products so that it appears as originally generated by the on-board experiment. To accomplish this goal, the following functions are considered essential to a LZP system: 1) synchronizing serial data streams to telemetry frames; 2) checking and correcting errors; 3) reassembling user packets from the frames; 4) reversing "backward" playback data; 5) merging together realtime data and playback data in the proper time order; and 6) deleting redundant data due to the overlap between realtime and playback data.

To perform these basic LZP functions solely by software demands expensive mainframe computers with high computational power or an array of medium performance workstations each handling some portion of the overall task. This is especially true when the data rate is high and the delivery time is short. However, as the Space Station Freedom

era approaches, even the power of these approaches may not be enough. In addition, these systems are costly to replicate, house, and maintain and are therefore not suitable for future distributed systems environments where similar capabilities would be required at numerous sites around the world and on other planetary objects (i.e., the lunar surface). To meet NASA's needs for drastically increased data speed and volume in the Space Station Freedom era, the Data Systems Technology Division (DSTD) at Goddard Space Flight Center proposed a new processing algorithm and a new architecture for a LZP system utilizing VLSI technologies [1]. The new LZP system is based on the VMEbus with multiple microprocessors running concurrently. The telemetry data will be processed by an array of 32-bit microprocessors and custom VLSI controllers while flowing through a custom telemetry data pipeline. Redundant Array of Inexpensive Disks (RAID) technology is used for system mass storage.

Since the initial proposal, a team of engineers from DSTD has been formed to design a prototype LZP system based on the proposed architecture and Consultative Committee for Space Data Systems (CCSDS) data format recommendations. This paper presents the implementation of this prototype system, with emphasis on the design of its core data take processing subsystem and mass storage subsystem.

2. LZP PROCESSING REQUIREMENT

The prototype LZP system being implemented uses CCSDS recommendations [2][3] as the format standard and supports all six essential LZP functions defined in the introduction. The input to the system is serial data in either telemetry transfer frames, VCDUs, or Coded VCDUs (CVCDUs). The outputs of the system are data takes consisting of time-ordered packets grouped by Application ID (or source) with associated annotation. The actual interfaces on both input and output depend on the system to which the LZP is integrated and are not covered in this project.

The system will operate in three non-exclusive operational modes: realtime, quick-look, and production processing. In the realtime processing mode, customer's packets are transmitted as soon as each packet has been received and reassembled. The data are also retained for normal production processing. In the quick-look processing mode, a high priority subset of the data take will be made available to the user. No redundancy deletion is performed. Again, the data are retained for normal production processing. The last and most important mode is production processing mode. Data from different acquisition sessions, including both realtime (forward ordered) and playback (reverse ordered), are processed and merged into data takes.

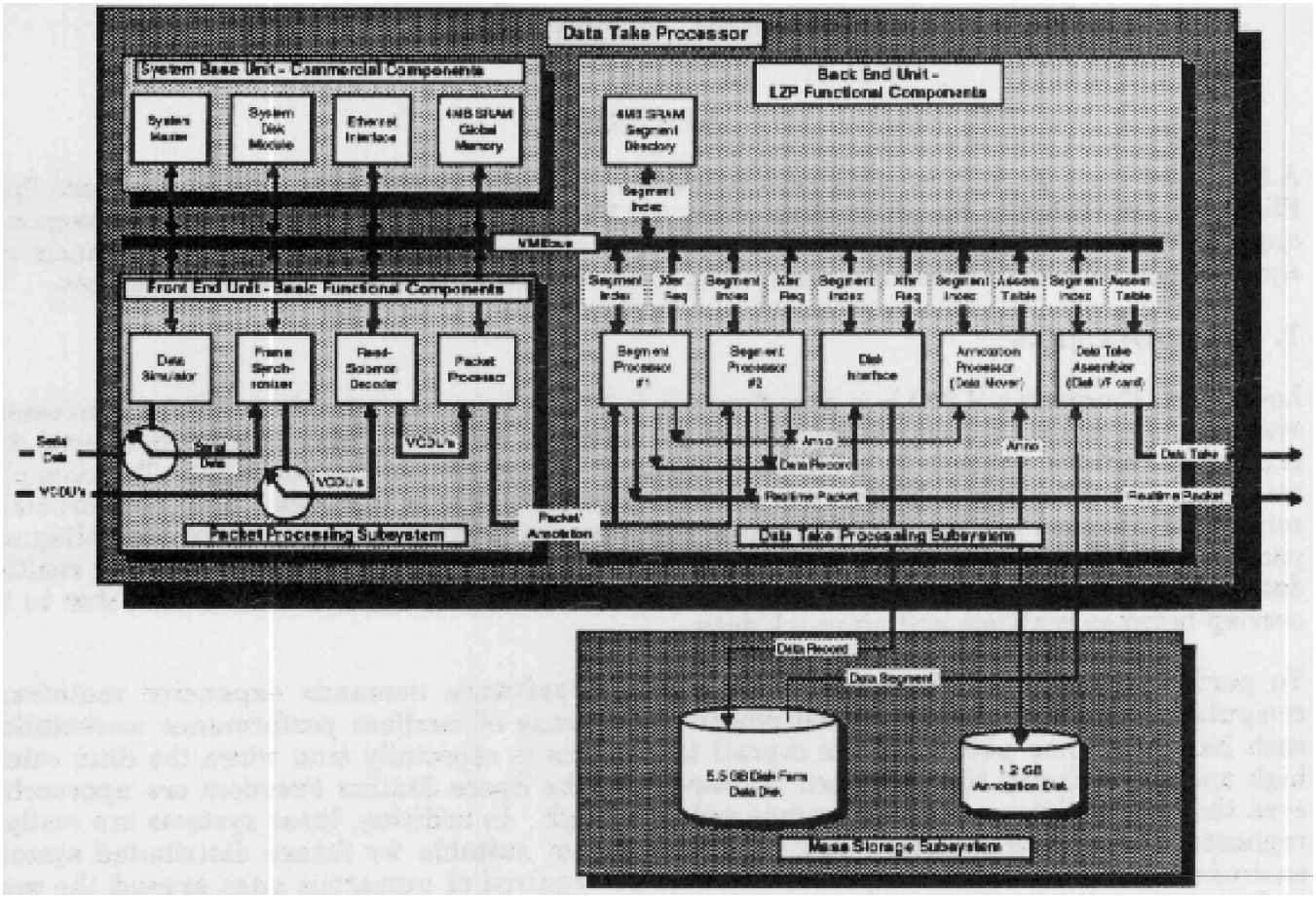

Figure 1. Functional Block Diagram of the Level Zero Processing System

The annotation of packets and data takes will be made for the customers by the LZP system. Also supported are data quality and accounting functions. Moreover, the catalog files for system operation, quality, and production will be maintained.

As a prototype system, the capacity of mass storage is chosen only to demonstrate the validity of system design.

3. THE DESIGN OF THE LZP SYSTEM

3.1 System Configuration

In order to achieve high speed and low cost with VLSI technology, the functional component approach [4] is taken in designing the LZP system. Off-the-shelf commercial microcomputer modules and standard NASA telemetry data processing functional modules developed at DSTD are used wherever available. Functional components specially developed for the LZP system are designed with emphasis on modularity and configurability for reuse in the future.

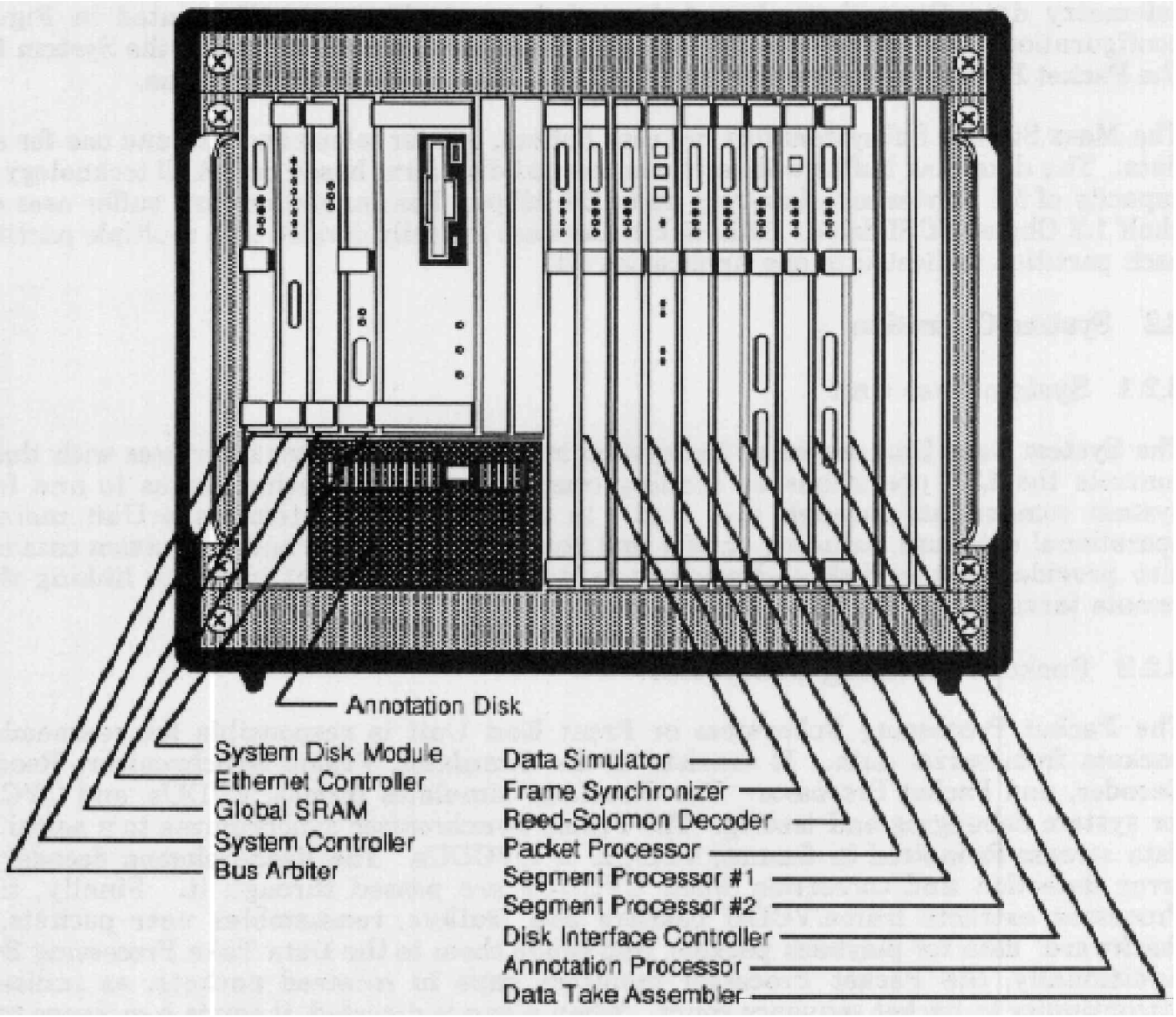

Figure 2. The Data Take Processor

The prototype LZP system consists of two subsystems: the Data Take Processor and the Mass Storage Subsystem. Shown in Figure 1 is the system functional block diagram. The Data Take Processor is based on a dual bus multiprocessor architecture housed in a 9U VME rack, as depicted in Figure 2. The microprocessors communicate through the VMEbus, while the telemetry data flows through a dedicated data pipeline. As illustrated in Figure 1, the configuration of the Data Take Processor includes three processing units: the System Base Unit, the Packet Processing Subsystem, and the Data Take Processing Subsystem.

The Mass Storage Subsystem has two disk buffers, one for telemetry data and one for annotation data. The data disk buffer utilizes a commercial disk farm based on RAID

technology with data capacity of 5.5 Gbytes and data rate up to 128 Mbps. The annotation disk buffer uses an off-the-shelf 1.2 Gbytes SCSI drive. Both disk buffers are logically divided into multiple partitions, with each partition dedicated to one Application ID.

3.2 System Operation

3.2.1 System Base Unit

The System Base Unit serves as the system master. The operator interfaces with this unit and controls the LZP operations by sending commands and gathering status to and from other system components through this unit. In addition, the System Base Unit maintains the operational database including quality and accounting data files and production catalog files. It also provides system disk and memory as well as the Ethernet interface linking the LZP to remote terminals or workstations.

3.2.2 Packet Processing Subsystem

The Packet Processing Subsystem or Front End Unit is responsible for reassembling user packets from serial data. It consists of the Simulator, Frame Synchronizer, Reed-Solomon Decoder, and Packet Processor. The Simulator simulates frames, VCDUs, and CVCDUs used for system debugging and testing. The Frame Synchronizer synchronizes to a serial telemetry data stream formatted in frames, VCDUs, or CVCDUs. The Reed-Solomon decoder performs error detection and correction when CVCDUs are passed through it. Finally, the Packet Processor extracts frame/VCDU headers and trailers, reassembles user packets, reverses “backward” data for playback packets, and sends them to the Data Take Processing Subsystem. Additionally, the Packet Processor monitors gaps in received packets, as indicated by a discontinuity in packet sequence count. When a gap is detected, it sends a message to the Data Take Processing Subsystem for updating the Segment Directory (the database to track the received telemetry data).

The Packet Processor extracts the sequence count and spacecraft time of each packet and builds them into an 8-byte time unit. Time records are formed when 512 time units from one Application ID have been received. For any packet with detected errors, an 8-byte annotation unit is generated by the Packet Processor. Annotation records are formed when 512 annotation units from one Application ID have been received. When either type of record is ready, the record is sent by the Packet Processor to the Segment Processor who then immediately forwards it to the Annotation Processor. Since annotation data is generated only for erroneous packets, the amount of annotation to be handled by the Data Take Processing Subsystem is significantly reduced.

3.2.3 Data Take Processing Subsystem

The major function of the Data Take Processing Subsystem or Back End Unit is to reassemble data takes from user packets. The four processing modules in the subsystem are the Segment Processor, Disk Interface Controller, Annotation Processor, and Data Take Assembler.

The Segment Processor sorts user packets according to their Application IDs. The sorted data are stored in separate buffers allocated for each Application ID. When data in the buffer exceeds the size of a disk record, that data record will be transferred to the Disk Interface Controller.

The Disk Interface Controller receives data records from the Segment Processor and transfers them to their corresponding logical buffers on the Data Disk. While the records are temporarily buffered on the card, disk access time can be minimized by rearranging the order that the records are written to the disk.

The Annotation Processor receives annotation records from the Segment Processor and transfers them to the Annotation Disk. At the end of each acquisition session, the Annotation Processor checks the Segment Directory to determine if data take output should commence. If the Segment Directory indicates that data disk utilization exceeds 50% capacity, then an output process begins. For each Application ID, the Annotation Processor checks to see if all segments received can be merged together. If so, then these segments will be output as a data take; if not, then the segments from earlier sessions which can be merged together will be output as a data take. To merge data segments together, their time boundaries will be compared. Redundant packets during session overlap will be deleted according to the quality of each individual packet. Using information stored in the Annotation Disk, a Data Take Assembly Table will be generated for each data take to be output. The table will include a header, a session list, an error list, and a gap list. The error and gap lists annotate the data take with all detected errors. The complete Assembly Table will be passed to the Data Take Assembler.

The Data Take Assembler uses the Data Take Assembly Table and Segment Directory to assemble a data take. The first table specifies data pieces in the data take and the second table is used to compute the addresses of these data pieces in the data disk. The data is transferred from the disk, block by block, to a local buffer, then from the local buffer to the output port. Thus, the data can be selectively output to perform redundant data deletions. If the data segment to be output contains playback data, (i.e., all packets in that segment are in reversed time order) the Data Take Assembler restores the order simply by transferring the last-received packet first and the first-received packet last. The Data Take Assembler also inserts data take headers and annotation into the output.

3.3 Data File Organization

To reduce the magnitude of data base management, the Data Disk is divided into X number of session partitions (as shown in Figure 3) which operate like ring buffers. The session partitions are then subpartitioned for Y number of Application Buffers. Each buffer will hold a data segment and its size is large enough to hold all the packets at the maximum rate for the session. All packets in a data segment have a unique Application ID, direction, and size; and they follow the time order strictly. The time span of a data segment is defined by the space times of the first and last packets of that segment. Time spans of various segments should be exclusive, except for overlap data. An Application buffer is further divided into Disk Records for physical disk storage purposes.

The partitions on the Data Disk and the partitions on the Annotation Disk are similar in format.

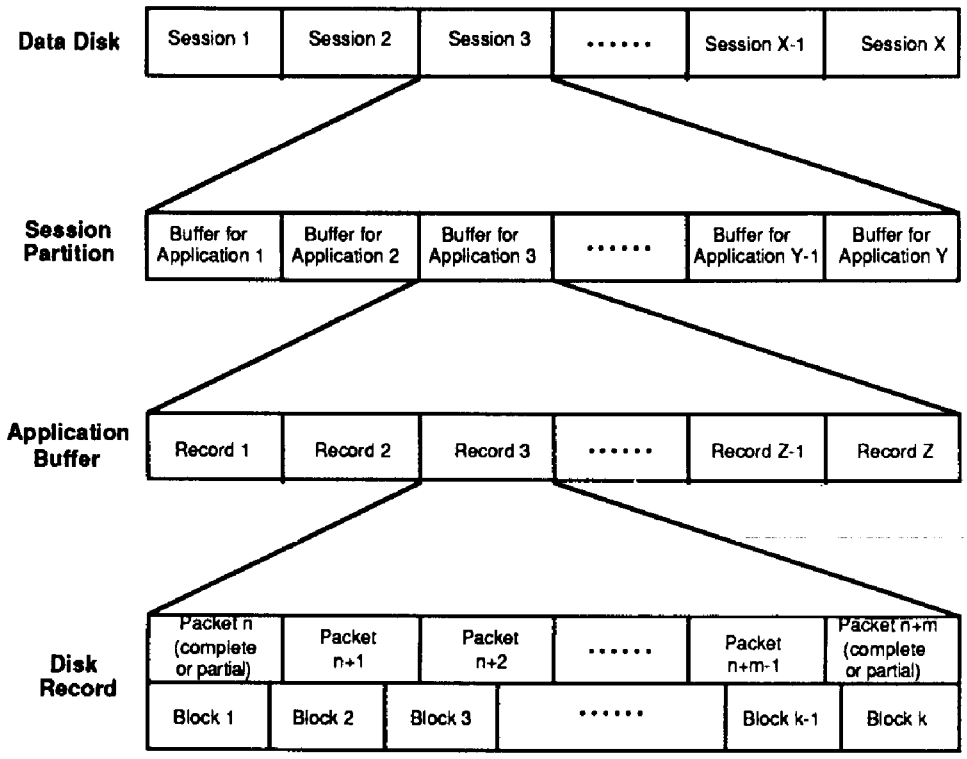

Figure 3. Data Disk File Organization

4. THE DESIGN OF SYSTEM COMPONENTS

4.1 System Base Unit

The System Base Unit consists of a VMEbus CPU module with a 25 MHz 68030 and 2 Mbytes of no-wait-state dual ported SRAM, a disk module with one 2 Mbyte floppy drive

and one 90 Mbyte Winchester drive, an 8 Mbyte SRAM card, and an Ethernet interface card. The integration of these high-performance commercial products provides a powerful base for system functions.

4.2 Packet Processing Subsystem

The Packet Processing Subsystem consists of generic telemetry processing modules developed at DSTD to support a number of NASA's operational projects. As described in various literature [5][6][7], these generic telemetry processing modules employ numerous VLSI semi-custom components, such as a test pattern generator, a frame synchronization chip set, a Reed-Solomon decoder chip set, a tribuffer controller, etc. to carry out generic telemetry processing functions such as test pattern generation, frame synchronization, Reed-Solomon error checking and correction, packet reassembly, error checking, and annotation.

The Simulator, Synchronizer, and Reed-Solomon Decoder cards each have a 25 MHz 68030 microprocessor as its main software processing engine. Since most processing functions are performed by the custom VLSI components, the microprocessor is mainly responsible for hardware setup, hardware operation control, status trailer generation, and status gathering.

The Packet Processor supports up to 8 Virtual Channels and 64 Application IDs. It uses three 25 MHz 68020 microprocessors and two semi-custom VLSI controllers to implement this multi-channel, multi-source packet processing capability. While the VLSI hardware moves data through a special on-board data pipeline between two dedicated RAM subsystems, the software executed on the three microprocessors performs processing tasks. By altering this software, the processor can be easily adapted to different data formats.

4.3 Data Take Processing Subsystem

All modules in the Data Take Processing Subsystem are specially designed for the LZP system; however, hooks have been embedded into their design for alternate future applications. Since these cards provide the core LZP functionality, a more detailed description for each is given.

4.3.1 Segment Processor

The Segment Processor is a combination of a 3U commercial single board computer and a 6U custom logic card, as shown in Figure 4. The commercial CPU board, equipped with a 25 MHz 68030 and 1 Mbyte memory, is responsible for communication and hardware

control. The side card, equipped with a 25 MHz 68020, custom hardware and 2 Mbytes memory, is responsible for switching, moving, and buffering data. There is one input port and three output ports, all implemented on the VMEbus J3 connector.

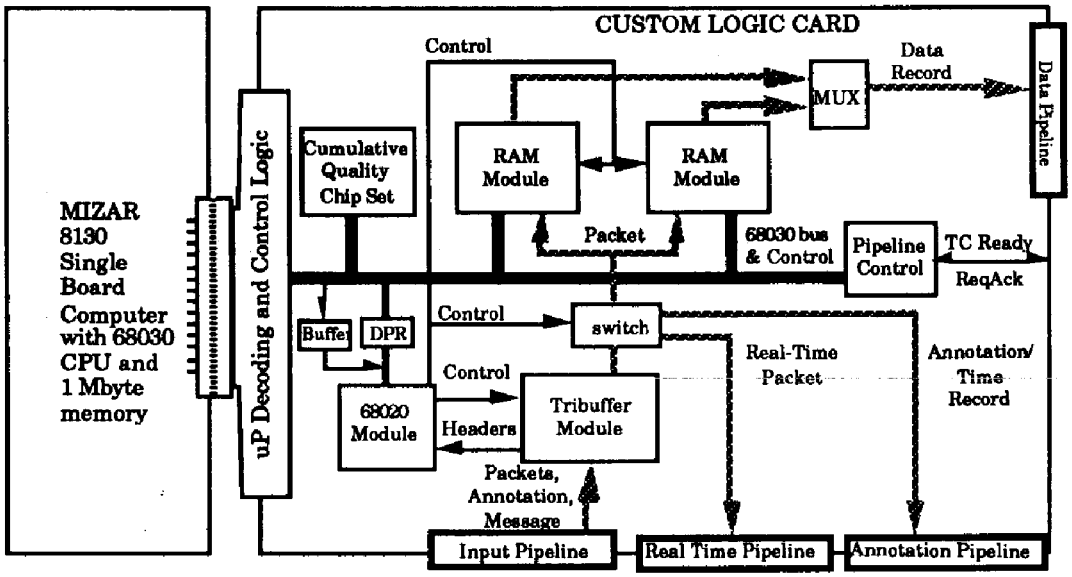

Figure 4. Functional Block Diagram of Segment Processor

All input to the Segment Processor comes from the Packet Processor. This input is a mixed data stream which includes user packets, annotation records, and messages from the Packet Processor. User packets are sorted by Application IDs and stored in separate buffers. While being saved in buffers, packets flagged as realtime are also routed to the realtime packet port for output. The annotation records are forwarded directly to the Annotation Processor through the annotation port. Messages are read and interpreted by the 68030 CPU. Such messages include information about the start and end of data segments as well as detected data gaps. The Segment Processor uses this information to update the Segment Directory.

When packets in a particular buffer have reached half full (128 Kbytes), the CPU forms a data record and sends a transfer request to the Disk Interface Controller. Upon receipt of permission from the Disk Interface Controller, the Segment Processor will send the record to the Disk interface Controller via the data pipeline port. Each buffer is 256 Kbytes and can hold two records. Thus, while one record is being output, new packets can be received concurrently. However, if the Segment Processor hasn't been granted permission from the Disk Interface Controller for its previous transfer request for a prolonged time, it can send a high priority request to expedite the process.

A Segment Processor manages 8 independent memory buffers and therefore can handle 8 Application IDs. To accommodate more data sources, up to 8 Segment Processors can be supported by the system, giving a maximum capacity of 64 Application IDs.

4.3.2 Disk Interface Controller

The Disk Interface Controller is another example of combining a 3U commercial CPU card and 6U custom logic card. As depicted in Figure 5, the custom card is comprised of a disk interface mezzanine, two 2 Mbyte memory buffers, and a pipeline interface. The disk interface circuit is implemented on a mezzanine card so that different mass storage devices can be accommodated by plugging in different interface mezzanines. For the prototype LZP system, a High Speed Interface (HSI) mezzanine is provided for data disk control and data transfer.

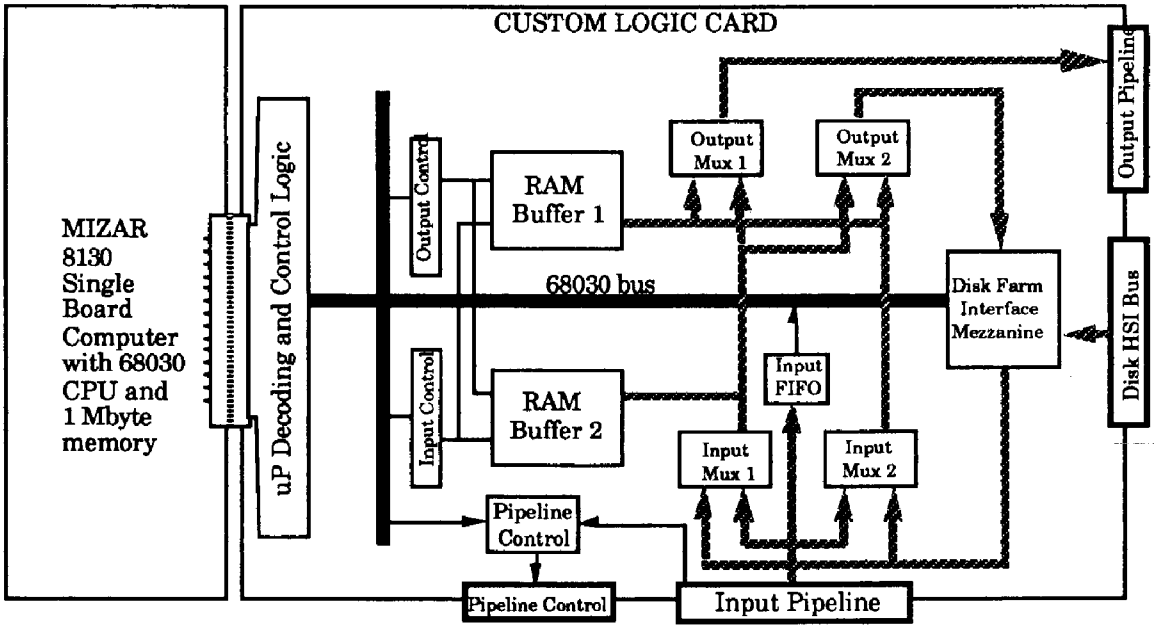

Figure 5. Functional block diagram of Disk Interface Controller

Two 2 Mbyte memory banks are designed to buffer data between the pipeline and the Data Disk. While one bank is taking input from the pipeline port, data in the second bank can be output to the Data Disk. The two banks are swapped when both input and output processes are completed. An extensive data flow analysis is included in Section 5 to justify the size of memory chosen. Moreover, pipeline control logic is designed to arbitrate the use of the pipeline bus by multiple Segment Processors.

4.3.3 Annotation Processor

The Annotation Processor is implemented by a generic High Rate Data Mover card designed at DSTD, shown in Figure 6. Upon software set up, the Data Mover card can move data at a rate up to 160 Mbps between six ports: the data pipeline on J3, VMEbus, VSBbus, CPU bus, and two interface mezzanines. Multiple data channels can be set up simultaneously, and data received by one port may be routed to multiple ports. Because the mass storage interfaces are implemented on mezzanines, they can be easily configured

to project needs. For the prototype LZP system, a SCSI interface mezzanine is installed to provide access to the Annotation Disk.

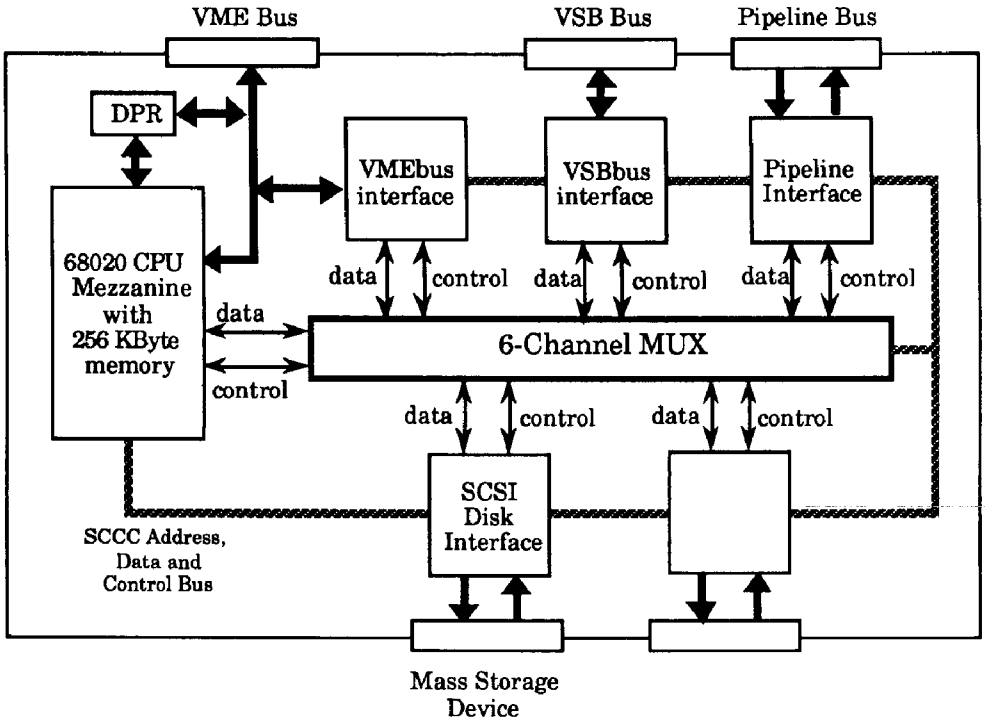

Figure 6. Functional block diagram of the Annotation Processor

The software processing capability is boosted by a 68020 mezzanine running at 25 MHz. To support full CPU operation, a CPU support chip, 256 Kbytes of memory, and a serial I/O port as well as interrupt and clock control logic are incorporated on the mezzanine. In addition to monitoring and controlling the Annotation Disk operation, the CPU mezzanine is also used to calculate boundaries of data takes and generate data take assembly tables.

4.3.4 Data Take Assembler

The Data Take Assembler uses the same hardware card as the Disk Interface Controller. The only difference is in the port configuration. Instead of taking input from the data pipeline and sending data to the data disk, the Data Take Assembler reads data off the data disk and writes data out to the data pipeline.

Automatic starting address generation, a special feature of the Disk Interface Card, is utilized by the Data Take Assembler to correct the order of packets received from a playback session in reversed order. Once data is loaded from the Data Disk into one of the memory banks, it will output the last packet in the data segment first and the first one last (note that data inside packets is always in forward order). After being set up with the starting address and the size of packet, the Starting Address Generator will calculate the

starting address of the next packet in the memory bank and load an on-board DMA engine with this address.

5. MASS STORAGE SUBSYSTEM

The mass storage requirements for the prototype LZP system described include sustained transfer rates above 100 Mbps, random access storage from 5 to 45 Gbytes, and multiple read/write ports. Several 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 meet these high performance requirements.

The system chosen for this LZP implementation can utilize from 1 to 4 banks of magnetic disk drives. Each bank includes 8 data drives, 1 parity drive, and 1 standby drive. The prototype LZP will use the minimum configuration of 1 bank of 760 Mbyte disk drives giving a total formatted storage of 5.5 Gbytes. This system is easily expanded to over 40 Gbytes in a single 7 foot tall equipment rack by filling all banks with 1.2 Gbyte disk drives.

Some of the mass storage subsystem characteristics include:

- 128 Mbps sustained storage/retrieval rates.
- Fault tolerant operation and realtime or background data reconstruction due to a bad disk drive.
- Complete media defect management and avoidance.
- Command queued logical block interface, with disk stripping invisible to user.
- Up to four High Speed Interface (HSI) channels for commands and data.
- Data channels burst at up to 40 Mbytes/s in a synchronous mode.

6. MASS STORAGE DATA FLOW ANALYSIS

In the past decade, disk drive technology has evolved dramatically in terms of single drive capacity. However, its speed, measured by data transfer rate, average seek, and latency time, still lingers where it was: 5 - 20 Mbps for transfer rate, 14 to 20 ms for seek time, and 6 - 9 ms for latency time. This inevitably worsens the already stressed disk I/O problem. Fortunately, RAID technology, employed recently by the mass storage industry, allows parallel access of an array of disk drives, synchronously or asynchronously, lifting their effective data rates by 8 to 64 times. This certainly alleviates the Disk I/O problem significantly, even though the seek and latency times remain the same.

The 20 Mbps LZP system uses disk storage as its working buffer under continuous data flow. Since all data going to the disk will be output later, the system requires double the

system rate; add to that a 16% operational margin and the average disk I/O rate is 46.4 Mbps. To meet this requirement, a state-of-the-art disk farm, based on RAID technology, was chosen as the Data Disk. The disk farm integrates ten disk drives into a disk bank, eight as data drives, one parity drive, and one stand-by drive. Controlled by a sophisticated controller, the entire disk bank acts as if it was a single disk drive. Eight data drives are accessed in parallel, yielding an effective data transfer rate of 128 Mbps. If the nature of disk accesses is strictly sequential, then the average disk I/O rate will be fairly close to this transfer rate.

The LZP system operational scenario demands random access to the Data Disk, since reading and writing occur on different logical session and Application partitions. With such a high data transfer rate, the transfer overhead, mainly incurred by the disk seek time and latency time, becomes very significant. For example, to transfer a record of 128 Kbytes of data to or from the disk takes only 8 ms, but average seek and latency time for one access takes 25 ms, more than 3 times longer than the time used to actually move the data! This fact dictates that the disk I/O rate is determined not only by the disk transfer rate, but also the transfer size and transfer overhead. The design of the LZP system mass storage path has taken all the above factors into account, as described in the following:

Let P_d be the disk I/0 rate,
R_d be the disk transfer rate,
t_o be the disk transfer overhead,
t_s be the disk seek time,
t_l be the disk latency time,
M_i be the input transfer size, and
M_o be the output transfer size.

Assuming $t_o = t_s + t_l$, i.e., other parts of the disk transfer overhead are negligible compared to t_s and t_l, then the disk I/O rate is

$$Pd = M / (M/Rd + to) \qquad (A)$$

where,

$$M = (2 \times M_i \times M_o) / (M_i + M_o) \qquad (A\text{-}1)$$

From equation A, the combined transfer size M can be expressed as

$$M = (P_d \times R_d \times t_o) / (R_d - P_d) \qquad (A\text{-}2)$$

Given P_d = 46.4 Mbps, or 5.8 Mbytes/s, R_d = 128 Mbps, or 16 Mbytes/s, and t_o = 25 ms,

$$M = (5.8 \times 16 \times 0.025) / (16 - 5.8) = 227 \text{ (Ebytes)}$$

That is, the combined transfer size should be at least 227 Kbytes. Further, select M_o 8 times bigger than M_i since data takes to be output are much greater than input data records. Substituting these numbers into equation A-1, we have

$$M = (2 \times M_i \times 8\ M_i) / (M_i + 8\ M_i) = 227 \text{ (Ebytes)}$$

Thus

$$M_i = 128 \text{ Kbytes}$$

$$M_o = 1024 \text{ Ebytes}$$

Having determined the required disk input and output transfer sizes, they are used in the design of the Segment Processor and Disk Interface Controller. On the Segment Processor, eight memory buffers of 256 Kbytes each are designed to hold up to two 128 Kbyte data records for eight Application IDs. While one record is moving out of a particular buffer, incoming packets from the same Application ID can simultaneously fill up the other.

On the Disk Interface Controller, there are two banks acting as a double buffer. Each bank is 2 Mbytes, capable of holding up to 16 data records. This is important because it enables disk access optimization to reduce the disk seek time. Such optimization is accomplished by writing data records to the Data Disk in the order of their disk logical addresses rather than the order they were received. This renders actual seek time lower than the given average seek time, thereby guaranteeing the specified disk I/O rate.

A simulation model of this design has been developed using the OPNET simulation tool [8]. The behavior of the system was studied under 6 different scenarios developed using data rates for EOS, ASTROMAG, TRMM, and OSL sources. These scenarios include cases of a few high rate data sources, a cluster of low rate data sources, and combinations of both. The results of this simulation demonstrate that the design is robust.

7. CONCLUSION

The implementation of a prototype VLSI Level Zero Processing system has been discussed. Based on VLSI technology and the functional component approach, the prototype LZP system features compact size, low cost, high processing throughput, high configurability, easy maintainability, and increased reliability.

The single LZP system can handle up to 8 Virtual Channels. Depending on the number of Segment Processors configured, data from 8 to 64 applications can be processed

simultaneously. Multiple LZP systems can be configured in parallel to process more Virtual Channels and data sources.

The design of the Data Take Processing Subsystem has been covered in detail. The application of the functional component approach is reflected in the board level design and all mass storage interfaces are implemented on mezzanines. This will significantly reduce development time and cost in the future, should different mass storage devices be chosen for the system.

A RAID technology based disk farm is selected as the Data Disk which offers large capacity, very high transfer rate, and reliability. An analysis of major data flow to and from the Data Disk is presented to illustrate the design. The results of a simulation study show that the design is robust under various scenarios based on space project data.

REFERENCES

1. Shi, J., Horner, W., Grebowsky, G., Chesney, J., "Prototype Architecture For A VLSI Level Zero Processing System", Proceedings of International Telemetering Conference, 1989, pp. 925-936.

2. "Packet Telemetry", CCSDS 102.0-B-2, Blue Book, Consultative Committee for Space Data Systems, January 1987.

3. "Advanced Orbiting Systems, Networks and Data Links", CCSDS 701.0-B-1, Blue Book, Consultative Committee for Space Data Systems, October 1989.

4. Chesney, J., Speciale N., Horner, W., Sabia, S., "High Performance VLSI Telemetry Data Systems", AIAA/NASA Second International Symposium on Space Information Systems, Pasadena, CA, September, 1990.

5. Speciale, N., Wunderlich, K., "High Speed Synchronizer Card Utilizing VLSI Technology", Proceedings of International Telemetering Conference, 1988, pp. 481-489.

6. "Reed-Solomon Card Hardware Definition Document", Microelectronic System Branch, Code 521, Goddard Space Flight Center, Greenbelt, MD, December 1989.

7. Grebowsky, G., J., Dominy, C., "VLSI High Speed Packet Processor", Proceedings of International Telemetering Conference, 1988, pp. 491-495.

8. “LZP Modeling Report”, CTA Inc., NASA Contract NAS5-30680, September 28, 1990.

NOMENCLATURE

AOS	Advanced Orbiting System
CCSDS	Consultative Committee for Space Data Systems
CPU	Central Processing Unit
CVCDU	Coded Virtual Channel Data Unit
DSTD	Data Systems Technology Division
EOS	Earth Observing System
GSFC	Goddard Space Flight Center
LZP	Level Zero Processing
Mbps	Megabits per Second
NASA	National Aeronautics and Space Administration
NASCOM	NASA Communication
RAID	Redundant Array of Inexpensive Disks
SSIS	Space Station Information System
VCDU	Virtual Channel Data Unit
VLSI	Very Large Scale Integration
VME	Versabus Module Eurocard
VSB	VME Subsystem Bus