A New Sequential Decoder for the DSN Telemetry Subsystem

J. H. Wilcher
DSN Data Systems Development Section

A new sequential decoder has been implemented in the DSS Telemetry Subsystem for the DSN MARK III Data System Implementation. This decoder performs the same decoding function as the Data Decoder Assembly performs in the Telemetry and Command Data Handling Subsystem. However, the new decoder is much faster, allowing potentially high data rates in the future.

I. Introduction

The DSS Telemetry Subsystem (DTM) being implemented as a part of the DSN MARK III Data System (MDS) Implementation Project must be capable of performing the same functions which are presently performed by the Telemetry and Command Data Handling Subsystem (TCD). Among these functions is sequential decoding. Sequential decoding is presently being performed by the Data Decoder Assembly (DDA) in conjunction with the Telemetry and Command Processor (TCP).

Early in the design phase of the MDS Implementation Project a decision was made to explore the possibility of implementing the sequential decoder function directly in the Telemetry Processor Assembly (TPA), which replaces the TCP, thereby doing away with the DDA as well.

A contract was issued with Modular Computer Systems, Ft. Lauderdale, Florida, to design and fabricate, to JPL specification, a prototype sequential decoder which would interface directly with the TPA, a Modular Computer Systems (MODCOMP) II-25 minicomputer.

This article presents a description of the implementation of this sequential decoder and the performance obtained from this decoder.

II. Decoder Implementation

The specification for the new sequential decoder required that the decoder be implemented using the Fano-Algorithm, the same algorithm which was used in implementing the sequential decoder function in the DDA. This was made a requirement since the algorithm is well understood within the DSN and that the probability of a successful implementation was greater with a well known algorithm versus a new, untried algorithm.

The specification also called for the consideration of failure diagnostic capabilities in the implementation of the

decoder. This feature would also prove to be beneficial in the testing of the decoder implementation.

A. Sequential Decoder Interface

The sequential decoder was implemented as a single-plane microcoded, read-only memory processor interfaced to the TPA and MODCOMP II-25 via Modular Bus Control (MBC) and Port 0 of the 4-port memory (see Fig. 1). This method of implementation allows the decoder to be initiated by the central processing unit's (CPU's) execution of any of the decoder's custom macroinstruction and then the CPU is released to run in parallel with the decoder.

The decoder accesses memory through the highest priority port (Port 0), of the 4-port memory controller to obtain parameter values and the received data symbols to decode. Access to all the required tables is also via the 4-port memory controller. One module (16 kwords) of the 64 kwords of memory is reserved for the sequential decode function. The received data buffers, tables, encoder parameters, etc. are contained in specific locations of this module. This feature allows the decoder to operate relatively independent of the CPU and the external Direct Memory Processor (DMP). The decoder competes with the CPU for access time to this dedicated module only when it is necessary for the CPU to perform housekeeping tasks such as formatting data for transmission, etc. The decoder competes with the external DMP when the DMP must input a new frame of received data symbols. Therefore, the decoder is not placing demands for time on the CPU or DMP when they are accessing other modules of memory and other tasks may be carried on in the CPU independent of the sequential decoder.

B. Sequential Decode Macroinstructions

The following macroinstructions are used in the normal operation of the sequential decoder:

1. Compute Tail Correlation (CTC). This instruction makes use of the so-called "quick look" property, of the class of convolutional codes currently being used, to compute the likelihood that a given position in the received data stream is the end or tail of a frame of coded data. Repeated execution of this instruction at all possible positions in the input symbol stream is sufficient to find frame synchronization with arbitrarily high confidence.

2. Sequential Decode (SEQD). This instruction implements the sequential decoding algorithm. It is necessary that there exist a properly formatted data buffer containing the received symbols and tail sequence associated with one spacecraft data frame. It is also necessary that the processor's general registers be loaded with all the parameters required by the instruction such as the location of metric tables, the impulse response, and the tail length. The execution time of this instruction is a variable depending upon the frame size and the details of the noisy received data. The instruction is terminated only by successfully decoding the frame or by the execution of a terminate instruction.

3. Terminate Sequential Decode (TSD). This instruction is used to terminate the sequential decode instruction when a frame of coded data has not been decoded in the allowable time limit as determined by the input data rate and the data buffer management scheme. The execution of this instruction will result in the controlled termination of the sequential decode instruction with the 'Erased Frame' bit of the sequential decoder status word set.

C. Sequential Decode Diagnostic Macroinstructions

The following macroinstruction when used with appropriate diagnostic software can be, and has been, very beneficial in ascertaining the condition of the sequential decoder, either as a troubleshooting aid in a maintenance depot to detect a failed part, or as an aid in analyzing the operation of the decoder:

1. Load Sequential Decoder (LSD). This diagnostic macroinstruction will cause the contents of CPU Register 1 to be read into each of the sequential decoder registers.

2. Dump Sequential Decoder (DSD). This diagnostic macro will dump the contents of the sequential decoder registers to a specific part of memory.

3. Step Sequential Decoder (SSD). Execution of this diagnostic MACRO is identical to SEQD except that the decoder will halt and interrupt the CPU after each step in the decoder algorithm.

Using the aforementioned diagnostic macroinstruction it is then possible to perform a complete checkout and failure analysis of the decoder by repeatedly loading the sequential decoder with a known pattern (LSD), stepping the sequential decoder (SSD), dumping the decoder (DSD), and comparing the results against known values. The "known" pattern could be a specific frame of coded data, therefore allowing the observation of the decoding process on a step-by-step basis. This feature could be useful for analyzing subtle failures in the decoder or for analyzing the data from a spacecraft for coder failures, for instance.

III. Performance

The performance of the sequential decoder was ascertained by comparing the results of the new decoder against known results. To accomplish this, frames of data were decoded by the DDA and these frames were retained as the "known". The same frames were then decoded by the new sequential decoder and the results compared on a bit-for-bit basis. If the new decoder was implemented to the same algorithm, the results of the decoding should match completely bit-for-bit and step-for-step. The only difference would be in the time to decode, which is a function of the method of implementing the decoder algorithm, not the algorithm. Approximately one-thousand frames of convolutionally coded data, of various frame lengths, bit rates, and signal-to-noise ratios, were processed by the new decoder and the results compared against the "known" results. The results matched in all cases.

One further parameter was measured while decoding the test frames with the new decoder. This was the decoder speed.

The time to decode a frame of data was recorded along with the data. Also recorded was the number of computations needed to decode the frame, a computation being defined as either a forward step or a backward step in the decoder. A figure of merit for a sequential decoder universally used is the computational rate (R_{comp}). In order to determine the R_{comp} for the new decoder, the time to decode and the number of computations were recorded and averaged over many hundreds of frames resulting in an average time per computation of 10.5 μsec. This translates into an R_{comp} of 95.2×10^3 computations per second. This compares with an average computation rate for the DDA of 22×10^3 computation per second, a factor of 4.3 improvement.

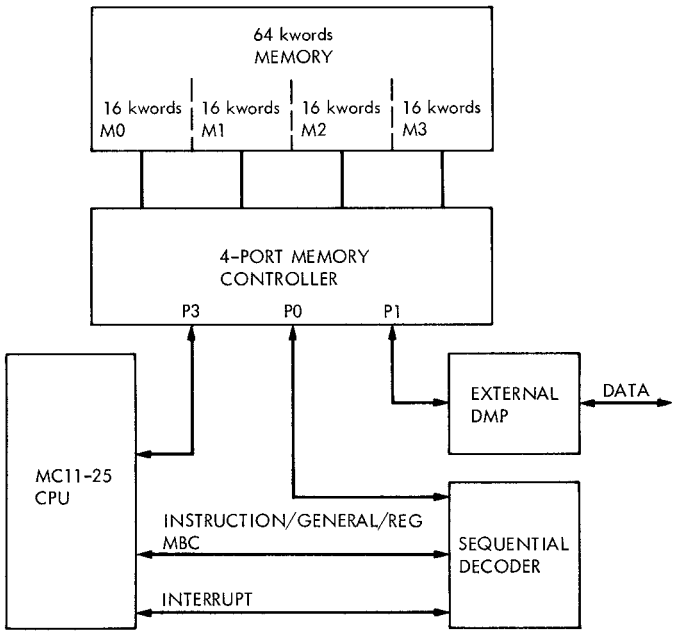

Fig. 1. The sequential decoder interface