DCA

NASA CR-170603

Symbol Synchronization for the TDRSS Decoder

Technical Report

to

NASA
Goddard Space Flight Center
Greenbelt, Maryland

Grant Number NAG 5-234

Daniel J. Costello, Jr.
Department of Electrical Engineering
Illinois Institute of Technology

September 3, 1982

Symbol Synchronization for the TDRSS Decoder

A block diagram of the TDRSS coding system is shown in Figure 1.

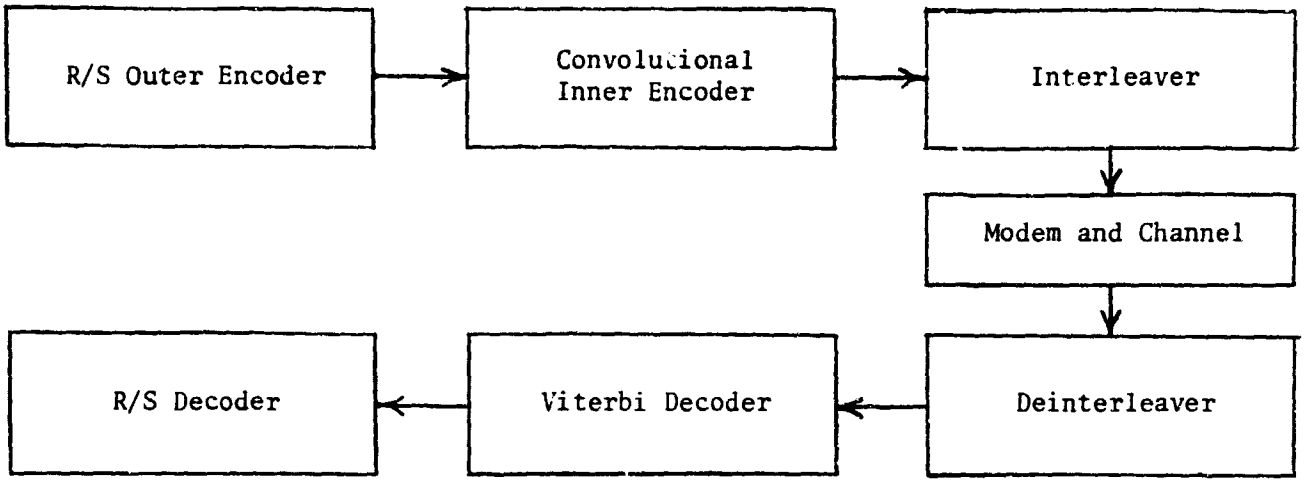

Figure 1.

The R/S code is a (255, 223) code with symbols in $GF(2^8)$. Each symbol in an R/S codeword is represented as an 8-bit sequence prior to convolutional encoding. The convolutional encoder is a (2, 1, 7) code, so each 8-bit sequence results in 16 bits at the output of the convolutional encoder. These bits are then interleaved by a (30, 116) periodic interleaver prior to transmission over the channel. The deinterleaver reassembles the bits in correct order prior to decoding. Monitoring of the path metrics in the Viterbi decoder allows synchronization to be maintained between the deinterleaver and the interleaver [1]. Each 8 bits out of the Viterbi decoder correspond to one symbol of the R/S code. Synchronization must also be maintained here, so that each 8-bit symbol delivered to the R/S decoder corresponds to an 8-bit symbol from the R/S encoder.

Lack of symbol synchronization, even in the absence of noise, would cause an error in almost every R/S symbol. This follows since even a 1-bit sync slip shifts every bit in each 8-bit symbol by one position, thereby confusing the mapping between 8-bit sequences and symbols. Clearly, the error-correcting-capability of the R/S code would be exceeded. One possibility for correcting this condition would be

to design the R/S decoder to recognize this overload, and then to shift the output sequence of the inner decoder to establish a different sync state. A maximum of 8 different sync states would have to be tested. Failure of any of the 8 sync states to pass the overload test could indicate loss of frame sync or a high channel error rate condition.

Another possibility for establishing symbol synchronization for the outer code is to use the characteristics of the inner decoder. The inner decoder currently is designed to provide sync for the deinterleaver, i.e., it must accurately determine the timing of a 30-bit sequence. This is accomplished by adding a 30-bit PN sequence cyclically to the output of the convolutional encoder prior to interleaving. After deinterleaving, but before decoding, the same 30-bit PN sequence is added. This is illustrated in Figure 2. If sync is not present, this is equivalent to adding a pseudorandom noise sequence to the decoder input, thereby causing the metrics of all paths to increase at roughly a uniform rate. This condition can easily be detected, and the deinterleaver can then be declared out of sync. A search procedure for correct sync, which involves testing a maximum of 30 sync states, then commences.

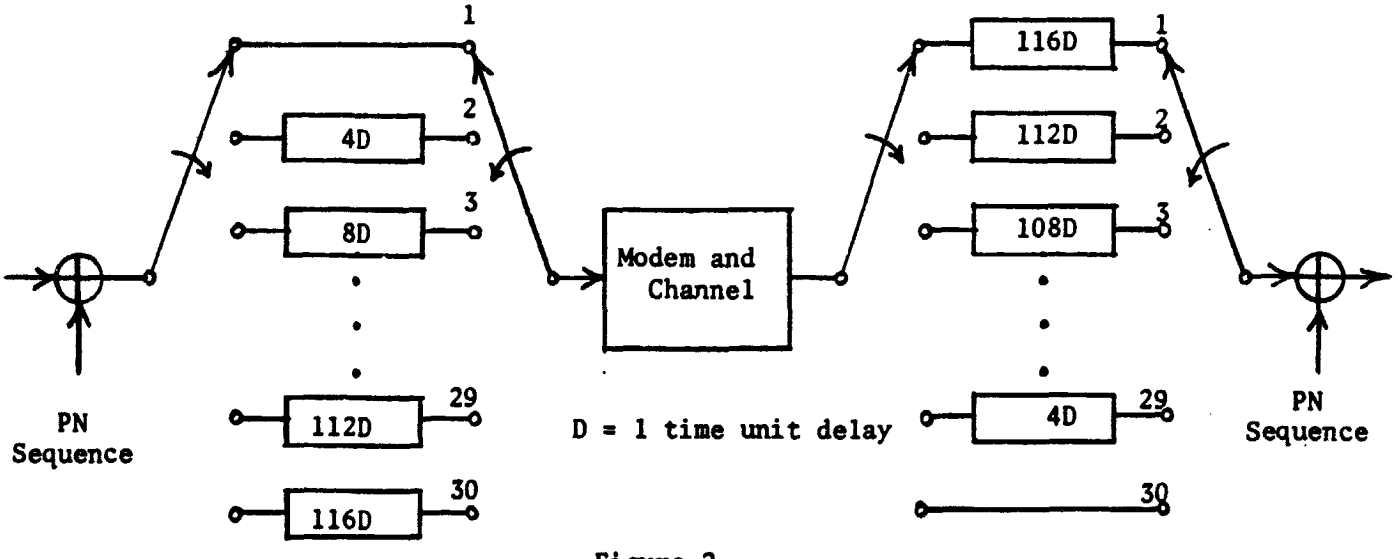

Figure 2.

Note that an 8-bit R/S symbol, after encoding by the convolutional encoder, results in a sequence of 16 encoded bits. Assume that the 30-bit PN sequence is added in such a way that its first bit corresponds to the leading bit in a 16-bit encoded symbol. Then the leading bit in that encoded symbol will be transmitted from position 1 of the interleaver. Successive leading bits will be transmitted from positions 17, 3, 19, 5, 21, ···. This sequence repeats itself with a period of 15. If the deinterleaver is properly synchronized, the leading position of an encoded symbol may be any of these 15 positions. In other words, position 1 of the deinterleaver may correspond to the leading position in an encoded symbol in one cycle of the deinterleaver, and then not again until 7 more deinterleaver cycles have passed. The complete sequence of leading position numbers is shown below:

1, 17; 3, 19; 5, 21; 7, 23; 9, 25; 11, 27; 13, 29; 15, 1; 17, 3; 19, 5;

21, 7; 23, 9; 25, 11; 27, 13; 29, 15; 1, 17; ···

one deinterleaver cycle

In this case, even though proper deinterleaver sync is established, correct symbol sync would be ambiguous because no single deinterleaver position always corresponds to the beginning of an encoded symbol.

The situation changes dramatically if a (32, 124) interleaver is used, and a 32-bit PN sequence is added to the encoder output so that exactly two 16-bit encoded symbols are covered. Any two bits in a 32-bit burst are now separated by at least 126 symbols after deinterleaving, thus effectively lengthening the interleaving depth. In addition, if the deinterleaver is properly synchronized, positions 1 and 17 will <u>always</u> correspond to leading bits in an encoded symbol. Hence deinterleaved symbols 1 through 16, after Viterbi decoding, will produce a single 8-bit R/S symbol and deinterleaved symbols 17 through 32 will produce the next R/S symbol. Two properly synchronized R/S symbols will therefore be produced during each deinterleaver

cycle. As long as the interleaver is designed to separate bursts of length B, where B is a multiple of 16, and the PN sequence length is also B, correct symbol sync will automatically be achieved when the deinterleaver is synchronized. Thus no additional hardware or searching is required to produce symbol sync, and the sync acquisition time remains fixed.

If no interleaving is used, symbol sync can be achieved by adding a cyclic PN sequence, whose length is a multiple of 16, directly to the encoder output sequence, with the first bit in the PN sequence corresponding to the leading edge of a symbol. At the decoder, the same PN sequence is added to the received sequence. If synchronization is not present, this looks like random noise to the decoder. The decoder path metrics will then tend to become roughly equal, a situation which can be detected and sync loss declared. A spiral sync search strategy, similar to that used for the deinterleaver [1], can then be employed. When correct sync is acquired, the leading edge of the PN sequence always corresponds to the beginning of an encoded symbol.

Another possible method of achieving symbol sync would be to modify the encoder so as to alternate periodically between two sets of generators. The period would equal 8, the number of bits in each R/S symbol. In other words, successive R/S symbols would be encoded by different generator sets. The decoder trellis would then alternate periodically every 8 branches between the different generators. In an out-of-sync condition, a portion of the received sequence encoded with one set of generators would be decoded on the basis of the other set of generators. This would create a "mismatch" condition at the decoder, making the received sequence appear noisier than normal, and the path metrics would tend to converge. Clearly this condition would be more pronounced as the amount of sync slip increased. The sync slip could vary anywhere from 1 to 8 branches. In the former case, only 1 out of every 8 received branches would be decoded with the wrong generator set. In the latter case,

every received branch would be decoded with the wrong generator set. The sync search algorithm would have to measure the change in the separation between path metrics as different sync states were tried, and choose the sync state which gives the best separation. A slightly more complex, but much improved, sync search strategy which uses the same general approach will now be discussed.

In the above sync strategy, a single branch sync slip may be hard to distinguish from noise. Hence the acquisition time, or the time required to acquire correct sync, may be long. This situation can be improved by switching between the two generator sets with a period of 4 rather than 8, if care is taken to always encode the first 4 bits in a symbol with the first set of generators. In this case, a single branch slip will affect the decoding of 2 out of every 8 branches. Further improvement can be achieved, at a cost of additional encoder complexity, by switching, periodically between 4 or 8 different sets of generators, always taking care to encode the first bit in each R/S symbol with the first set of generators. With 8 different sets of generators, the first bit in each encoded symbol would be encoded with the first set of generators, the second bit with the second set of generators, etc. A single branch sync slip would then cause every received branch to be decoded with the wrong generator set. This condition would be easy for the Viterbi decoder to detect, and a spiral sync search strategy could be used to acquire correct sync. The encoding concept is illustrated in Figure 3.

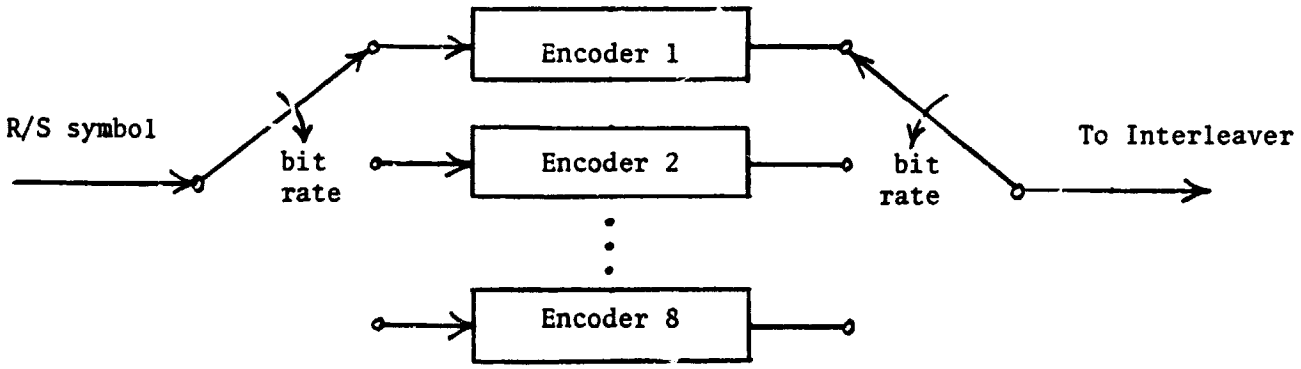

Figure 3.

The above sync strategy requires that different sets of (2, 1, 7) generators be found with equivalent distance properties. The major constraint is that the same generator sequence not be used in different generator sets. The two generator sequences for the (2, 1, 7) TDRSS code are 1111001 and 1011011. One possibility for producing another generator set with exactly the same weight distribution, but different codewords, is simply to reverse the TDRSS generators: 1001111 and 1101101. These are the only distinct (2, 1, 7) generator sets with $d_{free} = 10$. However, many sets exist with $d_{free} = 9$. In fact, a periodic time-varying (2, 1, 7) encoder with period 8 which uses a combination of $d_{free} = 9$ and $d_{free} = 10$ generator sets could have a free distance greater than 10. If true, this would provide even better performance than the TDRSS code and allow correct symbol sync as well. If this approach to synchronization is deemed feasible from an implementation point of view, a computer search can be undertaken to find the (2, 1, 7) periodic code with period 8 with maximum free distance.

Reference

[1]. "A Simulation System for Validating the Analytical Prediction of Performance of the Convolutional Encoded and Symbol Interleaved TDRSS S-Band Return Link Service in a Pulsed RFI Environment", Linkabit Corp., Final Report.