A Bandwidth Efficient Coding Scheme for the Hubble Space Telescope

S.S. Pietrobon* & D.J. Costello, Jr.*

SUMMARY A trellis coding scheme using 8PSK modulation is investigated for use on the Hubble Space Telescope (HST). Currently, HST's bandwidth is limited due to its use of S-band frequencies. Also, a bandwidth inefficient rate 1/3 BPSK coding scheme is being used which limits the data rate from Hubble to 1 Mbit/s. By using a rate 5/6 four-dimensional (4-D) 8PSK trellis code, it is possible to obtain a bandwidth efficiency of 2.5 information bits per 2-D signalling interval. This implies that the data rate from HST can be increased to 7.5 Mbit/s without an increase in bandwidth. The code selected has 16 states which gives an ideal coding gain of 1.5 dB compared to uncoded QPSK at a bit error ratio of 10^{-6}. Due to the multidimensional signal set, this code is also fully rotationally invariant, making the coding scheme robust to phase slips within an 8PSK demodulator. A 2.1 Mbit/s serial implementation of a prototype trellis decoder is described. This codec has been successfully tested over a Tracking and Date Relay Satellite (TDRS) from the White Sands Ground Station in New Mexico. With the coding scheme now proven, a rate 5/6 encoder and 8PSK modulator could be installed on Hubble during a Space Shuttle servicing mission in early 1997.

1 INTRODUCTION

To increase the data rate through bandwidth limited channels, trellis coded modulation [1] can be used. Trellis codes obtain their redundancy by expanding the size of the signal set, instead of expanding bandwidth, as in convolutional codes. We define bandwidth efficiency as the number of information bits transmitted per two-dimensional (2-D) signalling interval (K bit/sym).

To obtain non-integer values of K, trellis codes with multidimensional (multi-D) signal sets can be used [2,3]. These codes transmit k information bits over L 2-D symbols, to obtain K = k/L bit/sym. The multi-D signal set is constructed from the cartesion product of L 2-D signal sets.

Multi-D trellis codes have a number of other advantages as well. A large advantage is that many multi-D trellis codes are rotationally invariant. That is, a decoder is able to correctly decode the received data regardless of which rotational symmetry the 2-D signal set is in. Thus, a multi-D trellis coded system will be very robust to phase slips within a demodulator.

* Australian Space Centre for Signal Processing, University of South Australia, The Levels, SA 5095.

** Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556, USA.

Originally presented at the NASA Space Communications Technology Conference, Cleveland, Ohio, USA 12-14 November 1991 and the Institution of Engineers, Australia Seventh National Space Engineering Symposium Canberra 21-26 September 1992

Another advantage is in the decoders complexity. Each decoder iteration decodes $k \geq 2$ information bits. This is in comparison to many convolutional or punctured codes where only one information bit is decoded per decoder iteration. Thus, very high data rates can be achieved with moderate amounts of hardware.

The Hubble Space Telescope (HST) currently has a bandwidth limitation due to its use of S-band frequencies for data transmission. The data from HST is transmitted to a Tracking and Data Relay Satellite (TDRS) in geosynchronous orbit. The TDRS satellite amplifies the signal from HST for transmission to the White Sands ground station in New Mexico, U.S.A. To increase the data return from the satellite, bandwidth efficient coding schemes are required to minimise the required E_b/N_0 for transmission.

The following section describes how a multi-D trellis code was selected for possible use on HST. The code that was selected can increase the data rate from the current 1 Mbit/s to 7.5 Mbit/s. The next section describes in detail the implementation of a prototype 2.1 Mbit/s trellis decoder (using the maximum likelihood Viterbi algorithm) to actually test the coding scheme over a TDRS satellite. This is followed by a description of the features and interfaces of the trellis decoder and encoder. The performance of the codec and outcome of the satellite tests is then described.

2 CHOICE OF CODING SCHEME

The choice of code was based on two factors. The first factor was its application as a possible replacement for

the coding scheme currently used on HST. Hubble at present uses the rate 1/3 $\nu = 6$ (with $2^\nu = 64$ states) convolutional code with BPSK modulation. With the modulator restricted to 3 Msym/s (the number of 2-D symbols transmitted per second), this implies a data rate of only 1 Mbit/s, since the bandwidth efficiency K = 1/3 bit/sym. This is a very bandwidth inefficient scheme, although the system has the advantage of simplicity and large coding gain.

The basic requirement from NASA was for a scheme that has as large a K as possible. Since a satellite channel with non-linear amplifiers was being used, constant amplitude 8PSK modulation was selected. This gives a relatively high bandwidth efficiency (between 2 and 3 bit/sym) and reduces the effects of the channel. Also, trellis codes gives the best performance for these bandwidth efficiencies with 8PSK modulation [3]. The next influencing factor was INTELSAT's intention of transmitting the SDH 155.52 Mbit/s standard data rate over the 72 MHz transponders on its satellites. This requires a bandwidth efficiency of around 2.5 bit/sym. A Reed-Solomon block code can be used as an outer code to give very low bit error ratios (BER).

With the above requirements, the $\nu = 4$ (16 state) rate 5/6, 2.5 bit/sym, 4-D 8PSK trellis code from [2] was selected. This code has reasonable complexity and has a coding gain of 1.5 dB compared to uncoded QPSK for a BER of 10^{-5} [4]. This trellis code also has the advantage that it is 45° rotationally invariant. This means that the decoder needs only to synchronise to one of the two naturally mapped 8PSK signals in the signal set.

This code has $\tilde{k} = 2$ of the k = 5 input bits checked by a rate $\tilde{k}/(\tilde{k}+1) = 2/3$ encoder. The $\tilde{k}+1 = 3$ output bits of the encoder and the $k-\tilde{k} = 3$ unchecked bits are passed to a signal set mapper which maps these n = 6 bits to two 8PSK signal points. The output of the rate 2/3 encoder can be thought of as selecting some subset of 4-D points. A 4-D point within this subset is then selected by the three unchecked bits. There are $2^{\tilde{k}+1} = 8$ possible outputs of the rate 2/3 encoder. Thus, eight possible subsets can be selected. There are also $2^{k-\tilde{k}} = 8$ 4-D points in each of these eight subsets.

The code is designed so as to maximise the smallest squared Euclidean distance between paths of subsets generated by the rate 2/3 encoder (equal to 3.515 for this code assuming an average signal amplitude of one). The smallest squared Euclidean distance between the 4-D points must also be maximised (equal to 4.0 for this code). To improve the code further, the number of paths or points for these distances is also minimised (equal to 56 for the paths and 6 for the subsets points). Thus, we minimise the probability of selecting an incorrect path or point in a decoder due to noise on the channel.

Since the smallest distance between paths is smaller than the smallest subset distance, the free squared Euclidean distance $d^2_{free} = 3.515$. Correspondingly, the number of nearest neighbours $N_{free} = 56$. Compare this to uncoded QPSK (with K = 2.0 bit/sym) which has a minimum squared Euclidean distance of 2.0 and two nearest neighbours. With a larger distance between paths, the multi-D code will perform better than uncoded QPSK for moderate BERs and with a greater bandwidth efficiency. The asymptotic coding gain is 3.42 dB compared to uncoded QPSK.

Since the code is rotationally invariant, a differential encoder must be included before the rate 5/6 encoder. Of the six bits that map into the 2-D signal set, only three are affected by a 45° rotation. Of these three bits only one is checked by the rate 2/3 encoder. With the code used this allows rotational invariance to be achieved.

At first, a systematic (feedback) encoder was used in the design. However, it was found that in designing a trellis decoder, it would be simpler if a non-systematic (feedforward) convolutional encoder was used. Non-systematic encoders also have a slight performance advantage since the length of the error events is nearly always one less than with systematic encoders.

The largest simplification is in the Survivor Sequence Memory where the traceback operation is performed. Simple shift registers can be used to determine the next state in the traceback. With a systematic code, additional logic is required which increases the complexity and slows down the traceback operation. Also, non-systematic encoders all have the same trellis structure, regardless of which code is used. This allows existing trellis decoder designs to be used for different codes with a minimal number of modifications.

To convert the systematic encoder to a non-systematic form, the technique described in [5] is used. This method uses the fact that the impulse response of each shift register in a non-systematic encoder will produce output sequences that are equivalent to the generator polynomials. Since a systematic encoder must also produce the same sequences, it is relatively easy to find $\tilde{k}$ linearly independent output sequences from a systematic encoder that can be used as generators of a non-systematic encoder.

There is usually more than one set of possible generator polynomials. We choose the polynomials so that the inputs to the rate 2/3 encoder are affected by a 45° phase rotation in the same way as in a systematic encoder. Using poynomial notation where D is the delay operator, the non-systematic encoder equations that were found for the rate 2/3 encoder are

$$z^2(D) = x^2(D) \oplus (D^2 \oplus 1)x^1(D), \quad (1a)$$
$$z^1(D) = D^2x^2(D) \oplus (D^2 \oplus D \oplus 1)x^1(D), \quad (1b)$$
$$z^0(D) = Dx^2(D). \quad (1c)$$

Figure 1 illustrates the rate 5/6 encoder, showing the differential encoder, rate 2/3 non-systematic encoder, and 4-D signal set mapper. After a 45° phase rotation, we have $z^2_r(D) = z^2(D)$, $z^1_r(D) = z^1(D) \oplus 1(D)$, and $z^0_r(D) = z^0(D)$. Rotating the equations in (1) gives

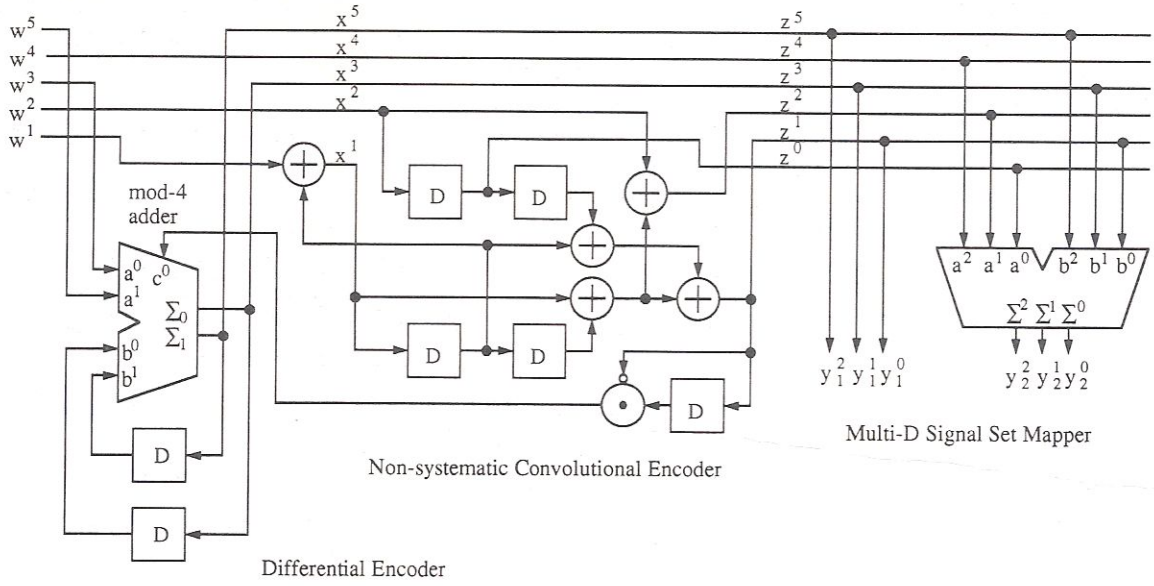


Figure 1 Non-systematic encoder block diagram for the 16 state 2.5 bit/sym 4-D 8PSK trellis code

$x_r^2(D) = x^2(D)$ and $x_r^1(D) = x^1(D) \oplus 1(D)$, the same as for a systematic encoder.

The encoder that was implemenmted uses a Phase Locked Loop (PLL) to generate the two times clock for transmitting the two 2-D symbols. This PLL is based on the 74HC4046 Integrated Circuit (IC) and will lock from 4 Hz to 840 kHz (corresponding to data rates from 10 bit/s to 2.1 Mbit/s).

3 DECODER IMPLEMENTATION

Due to the complexity of the decoder design, only a brief description is given here. As such, only the important design decisions are described.

To reduce the cost of the codec, a serial implementation of the decoder was chosen. That is, one clock cycle would be required for each state of the code. Since there are 16 states, at least 16 clock cycles are required to process each received 4-D point. As will be described in more detail later, an extra seven clock cycles are required for start-up purposes. Thus, a total of 23 clock cycles are required for each iteration of the Viterbi algorithm.

The technology and clock speed in our design is the same as used in another Viterbi decoder designed by the author [6]. This gave us greater confidence that the design would work, even though the actual design is twice as complicated. Our design uses a 10 MHz clock (giving 100 ns clock cycles) and Schottky TTL logic for its ease of use and large variety of functions. The actual technologies used are 74LS (Low-power Schottky TTL) for non-time critical sections of the circuit and 74F (Advanced Schottky TTL) for time critical sections. Other technologies are used for functions not available in 74F or 74LS.

The decoder is operated asynchronously to the received data clock. This requires one of the seven extra clock cycles described above. Internally, the decoder operates synchronously to the 10 MHz clock. The decoder starts operation after detecting the first rising edge of the received 4-D symbol clock. After 23 clock cycles, the decoder stops and waits for the next rising edge of the 4-D symbol clock. This allows the decoder to operate at any data rate from 0 to 2.1 Mbit/s.

Each iteration of the Viterbi algorithm decodes k = 5 bits for each received 4-D signal point. The maximum 4-D symbol rate of the decoder is the internal clock speed divided the number of clock cycles required to decode the five bits, i.e., 4.35×10^5 4-D symbols per second. Therefore, the maximum bit rate of the decoder is 2.17 Mbit/s. For actual use on the HST, it is intended that the decoder would be implemented using programmable gate arrays, where the required 7.5 Mbit/s decoding speed would be achieved.

There are six main sections in the Viterbi decoder. These are

- Branch Metric Calculator (BMC)
- State Metric Calculator (SMC)
- Survivor Sequence Memory (SSM)
- Signal Set Synchronisor (SSS)
- Minimum State Metric Selector (MSMS)
- Branch Point Selector (BPS)

Figure 2 illustrates a block diagram of the decoder. The above sections are described as follows.

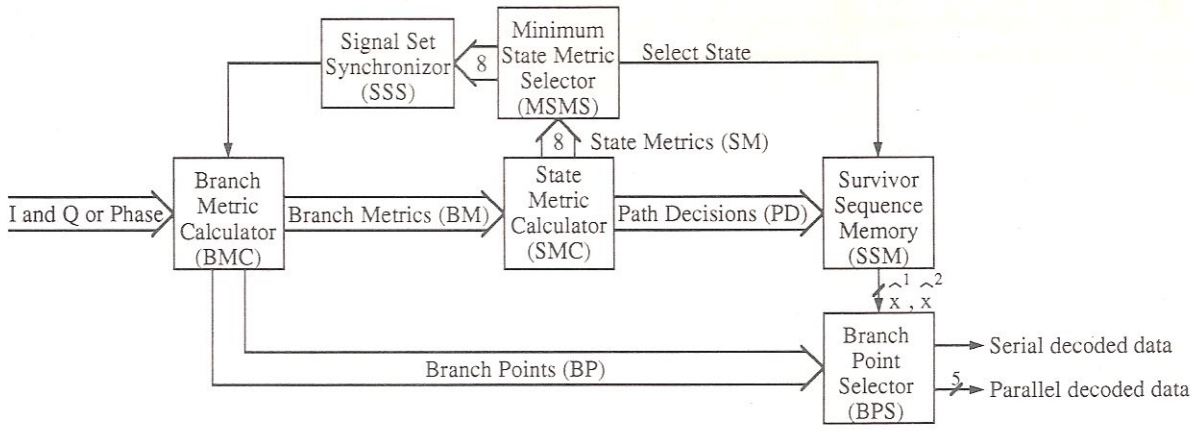


Figure 2 Block diagram of a trellis decoder for the 16 state 2.5 bit/sym 4-D 8PSK trellis code

3.1 Branch Metric Calculator

For each transition of the trellis there are eight parallel 4-D points (due to the three unchecked bits in the encoder). The BMC must determine which of the points is closest to the received 4-D signal point (the Branch Point (BP)) as well as how close the received point point is to the BP (the Branch Metric (BM)). The BM can be calculated in a number of ways. The optimum BM's for AWGN channels with quantisation are neg-log-likelihood metrics [6]. Alternatively, one could make an approximation based on the squared Euclidean distance between the received point and the points along the transitions.

In our design we have chosen to use Read Only Memory's (ROM's) to store the precalculated BP (three bits are used to represent each parallel path) and BM (using squared Euclidean distance). The encoder can select one of eight subsets of 4-D points (with each subset containing eight 4-D points). The BP and BM must be determined for each of these eight subsets of 4-D points.

We have chosen four bits to represent the BM value. This gives a BM range from 0 (closest to the received 4-D point) to 15 (furthest from the 4-D point). Decoder simulations in [7] for another multi-D trellis code indicate that this amount of quantisation results in little performance degradation. To maximise the decoder's performance, the branch metrics are scaled to 31 and then limited to 15. This way, the more likely BMs will have greater resolution. This more than makes up for the distortion caused by the less likely BMs which are limited. It was found that the implementation loss could be reduced from 0.1 to 0.15 dB using this technique.

To minimise the number of address bits to the ROM, each received 2-D signal point has been quantised to seven bits. After extensive simulations in [7] for a 6-D 8PSK trellis code, it was found that pie-chart or angular quantisation results in the least performance degradation. The simulations included the "dartboard" quantisation pattern proposed in [2].

Each ROM therefore has an address space of 14 bits (seven bits for each 2-D symbol). The ROM's used for the BMC are 32Kx8 27C256's. A total of 6 ROM's were used, two for determining the BP's and four for the eight BM's.

Alternative BMC schemes which exploit the finite length trellis structure of the parallel transitions were also considered. That is, a Viterbi like decoder can be used to decode the parallel transitions. However, their large complexity (in a discrete implementation) led us to choose the simpler ROM look-up method. For a gate array implementation though, the trellis decoding method would be preferable due to the flexibility that gate arrays provide in designing circuits.

3.2 State Metric Calculator

The SMC updates the State Metrics (SM) for each state of the code in each iteration of the Viterbi algorithm. A SM is an indication of how close the received sequence is to the closest path of all paths leading into a particular state. Since the code has $\tilde{k} = 2$ checked bits, there are $2^{\tilde{k}}=4$ paths leading into each state. For each of the four paths, we must add the BM for that path to its corresponding SM (also known as the old SM) from the previous iteration. The new SM for the four paths leading into a state is the smallest of these summations. This is called the Add-Compare-Select (ACS) operation.

With four paths into each state a 4:1 ACS circuit is required. With 16 states in our code, the ACS operation needs to be performed 16 times (explaining the need for 16 clock cycles). The ACS circuit also produces two Path Decision (PD) bits which indicate which of the four paths was chosen. This information is passed to the SSM where it is stored.

Since the decoder operates serially, only one ACS circuit is required. The 16 SM's are stored in two 74AS870 dual 16x4 static Random Access Memory (RAM) chips. Eight bits are used to represent each SM. As shown in [7] for a 6-D 8PSK trellis code, this is more than enough bits when two's complement arithmetic is used

in the ACS circuit to prevent overflow [6]. Before the first new SM can be calculated, four old SM's are read out from the RAM's. This takes four clock cycles. It takes another two clock cycles to perform the ACS operation. To achieve a slightly higher speed, we could have done the ACS operation in one clock cycle. However, this would have required six comparator chips to find the minimum SM. An increase of one clock cycle and the use of three comparator chips was chosen to decrease the complexity of the design.

Another clock cycle is used to write to the other half of the dual 16x4 RAM's. Since all the read and ACS operations are pipelined, an additional 15 clock cycles are required to write the 15 remaining new SM's. In the next iteration of the algorithm we read from where the SM's were written in the previous iteration and write to where the old SM's had been stored. The process then repeats.

For the ACS circuit, the appropriate BM's must be added to the correct old SM's. Twelve quad 2:1 multiplexer chips and a copy of the convolutional encoder are used to complish this task.

3.3 Survivor Sequence Memory

The SSM has two tasks. It must store the Path Decisions (PD's) generated by the SMC and "traceback" through the previously stored PD's to determine the final decoded bits for x^2 and x^1. The traceback depth is the required number of PD sets (each set consists of 16 two bit PD's) that the SSM must trace back through.

The PD's must be stored in the remaining 16 clock cycles that are available. There are two ways this can be achieved. Storing two PD bits in each clock cycle or storing four PD bits in every other cycle, leaving the alternate cycle to perform part of the traceback. With the first method at least two separate memories are required since the traceback operation cannot be performed simultaneously with the storage of the new set of PD's (due to the design of memory chips). Since there is a finite amount of memory, the oldest PD set must be written over.

There is usually a point where one method is better than the other (in terms of the total memory size required) based on the number of clock cycles available and the traceback depth. A traceback depth of around 25 to 30 results in little performance degradation [7]. Comparing the implementation complexity of the two methods, the alternating read/write method proved superior.

With this design only eight clock cycles are available to perform a traceback. To maintain integer power of two address spaces for the memories (and thus efficient use of practical memory designs), a traceback depth of seven is used for each SSM memory chip. To achieve the required traceback depth, four 64x4 memories are required. This gives a traceback depth of 28. The traceback is performed in a pipelined fashion, switching between memories when required and waiting for the next received set of data to continue with the traceback. Four separate memories are required since there are four tracebacks in operation at any one time.

Since there are no 64x4 RAM's commercially available, larger 256x4 93422A RAM's were used. This chip has separate input and output data buses which simplifies the SSM design. We use the state with the smallest SM to start the traceback. This is the best state the SSM could start with (since it corresponds to the path that is closest to the received signal) and helps give the decoder a slight performance improvement over choosing a random or a fixed state. The Minimum State Metric Selector (MSMS) provides the information needed to achieve this.

At the correct time and place in the circuit, the two decoded bits $\hat{x}^1$ and $\hat{x}^2$ are produced. The two bits are passed to the Branch Point Selector (BPS) where they are re-encoded to select one of the eight 3 bit branch points. The branch points are delayed by 34 4-D symbol periods, 28 due to the traceback, 4 due to the pipeline delay in the traceback, and 2 due to the re-encoding of the decoded data.

The five decoded bits are then differentially decoded (optional) and then parallel to serial converted for the final decoder output. Differential encoding and decoding are optional as there are some communication systems that do not require phase synchronisation. For example, a burst modem can provide phase information in the preamble of a burst. A 74HC4046 PLL is used to generate the required five times clock for the serial data. This PLL is tuned to lock within 1kHz to 2.1 MHz. The decoded data is also available in five bit bytes.

3.4 Signal Set Synchroniser

The SSS has the task of synchronising the decoder to the received sequence of 2-D symbols. Since the signal set consists of two 2-D signals, the decoder must synchronise to one of the two possible ways the received data can arrive.

The decoder is asynchronously locked to DATCLK, which is the received 2-D symbol clock whose frequency has been divided by two. A delay of zero or one 2-D symbol periods of DATCLK is used for timing synchronisation.

The SSS works by examining the rate of increase of the minimum SM from the MSMS. If the rate is high, this indicates that the decoder is out of synch and needs to be resynchronised. A variable threshold in the SSS is used for this purpose. If the threshold is exceeded, the SSS will toggle into the "arm symbol toggel" state.

If the threshold is again exceeded in the next V (V is a variable from 0 to 63) 4-D symbol periods the decoder will toggle the 2-D symbol delay (from zero to one or one to zero). The SSS then ignores the decoder for

128+V 4-D symbol periods to allow the decoder to settle into its new signal set configuration.

If the threshold is not exceeded the SSS will "disarm" and return to its normal monitoring state.

4 OTHER DECODER FEATURES

The encoder and decoder are mounted within a 3U high 19 inch rack as shown in Plate 1. On the front panel, two Light Emitting Diodes (LED's) are used to indicate the 2-D symbol delay.

Plate 1 Prototype 2.1 Mbit/s Rate 5/6 8PSK Codec

To test the decoder, the 2-D symbol delay can be independently set to manual control. In this way, the SSS can be isolated from the rest of the circuitry so that any problems with the rest of the decoder can be fixed without the SSS interfering. It can also be used to test the SSS by manually introducing delays into the received signal. There are two switches used for this.

Two rotary type switches are used to select the format of the received data. One switch is used to select between 3 bit phase (corresponding to hard decision), 7 bit phase quantisation, 5 bit I and Q quantisation, or internal loopback mode. The other switch selects between signed magnitude, reverse binary, straight binary, or two's complement data formats for I and Q received data.

There are also switches for disabling the differential decoder and the differential encoder. The encoder has another switch to select between five bit parallel or bit serial data. The decoder also has a reset button to force all the SM's to zero. The encoder/decoder interface diagram is given in Figure 3.

The 159 integrated circuits of the design are placed on two double height Speedwire Eurocards (233.4x220 mm). Speedwire allows quick and reliable connections (if it is done correctly) between the chips that can be easily changed. The speedwire boards also have good groundplanes, critical when operating at high clock speeds. The trellis decoder (which operates at 10 MHz) is placed on one board (taking 96 chips) while the encoder, SSS, and various interface chips are placed on the other board.

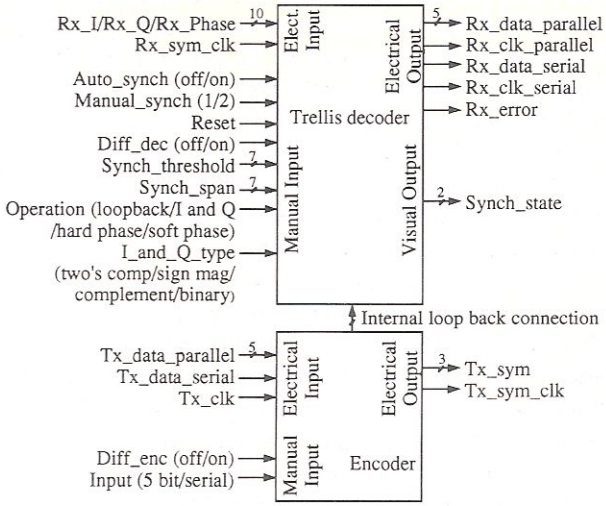


Figure 3 Trellis decoder/encoder interface diagram for 16 state 2.5 bit/sym 4D-8PSK trellis code

BNC connectors are used at the back of the rack for external data and clock connections. It is assumed that all received data changes on the rising edge of its clock. Similarly, the codec produces its signals in the same format. TTL 75 Ω interface signals are used for these external interfaces.

5 CODEC PERFORMANCE

To test the codec with soft decision noise, a test program was written for use on a PC. This program generates a pseudo-random binary sequence which is then encoded. The program also generates additive white Gaussian noise which is added to the 8PSK signal. The program then quantises the noisy signal into a specified number of phase levels. The soft decision sample is passed to the parallel port of the PC and then to the trellis decoder via a specially made cable. An external clock to the PC (via the parallel port) and trellis decoder at the 2-D symbol rate is used to synchronise the PC to the trellis decoder. The decoded output from the trellis decoder is passed to a BER test set to measure the decoder's performance.

Figure 4 gives the performance of the trellis decoder for five and seven bit phase quantisation. A computer simulation of an ideal decoder (with infinite quantisation) is also given for comparison. For seven bit quantisation, the implementation loss is only a 0.25 dB for a BER of 10^{-4}. With five bit phase quantisation (where the two least significant bits of the seven bit input are set to 01 or 10) the implementation loss is 0.6 dB.

During August 1992, the codec was tested by the University of New Mexico, Las Cruces with an 8PSK modem over a TDRS satellite [8] at the White Sands ground station. Since the demodulator only gave five bit phase quantisation, the trellis decoder was configured with the two least significant bits set to a constant value. The tests were entirely succesful.

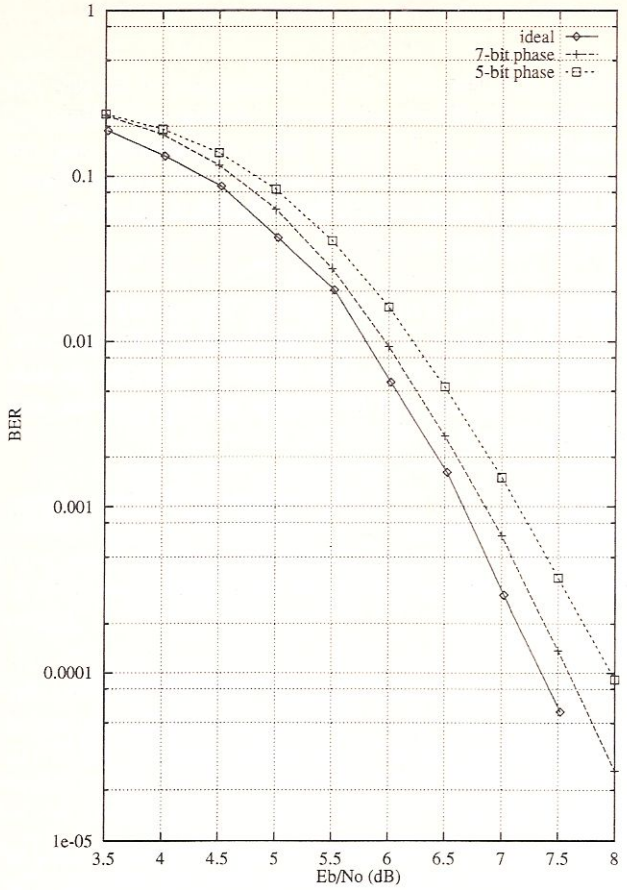


Figure 4 BER versus Eb/No for 16 state Rate 5/6 8PSK

6 CONCLUSIONS

A serial implementation of a prototype 2.1 Mbit/s trellis decoder for the 16 state 2.5 bit/sym code with a 4-D 8PSK signal set has been described. This code can be used to increase the data rate from the current 1 Mbit/s to 7.5 Mbit/s for the Hubble Space Telescope.

With the successful test of the prototype codec over a TDRS satellite, a new rate 5/6 encoder and 8PSK modulator could be placed on Hubble during a servicing mission in early 1997. The corresponding 8PSK demodulator and trellis decoder (using a programmable gate array implementation) would be placed at the White Sands ground station.

With a higher data rate, scientists will be able to extract their data more quickly and in larger amounts from HST. This will increase the data return from HST and use less time of the heavily used TDRS satellites.

7 REFERENCES

1. Ungerboeck, G., "Channel coding with multilevel/phase signals", IEEE Trans. Inform. Theory, Vol. IT-28, pp. 55-67, Jan. 1982.

2. Pietrobon, S. S., Deng, R. H., Lafanechère, A., Ungerboeck, G., and Costello, D. J., Jr., "Trellis-coded multidimensional phase modulation", IEEE Trans. Inform. Theory, Vol. 36, pp. 63-89, Jan. 1990.

3. Pietrobon, S. S. and Costello, D. J., Jr., "Trellis coding with multidimensional QAM signal sets", IEEE Trans. Inform. Theory, Vol. 39, pp. 325-336, Mar. 1993.

4. Perez, L., "On the performance of multi-dimensional phase modulated trellis codes", NASA Tech. Report #89-10-02, Oct. 1989.

5. Porath, J. E., "Algorithms for converting convolutional codes from feedback to feedforward form and vice versa", IEE Electron. Lett., vol.25, pp. 1008-1009, 20 July 1989.

6. Pietrobon, S. S., "Rotationally invariant convolutional codes for MPSK modulation and implementation of Viterbi decoders", M.Eng. Thesis, School of Electron. Eng., South Australian Inst. of Technol. (now University of South Australia), Adelaide, SA, June 1988.

7. Gray, P. K., Morrison, I. S., and Cowley, W. G., "The development of a 45 Mbit/s modem/codec", Proc. IREECON'89, pp. 942-945, Melbourne, Australia, Sep. 1989.

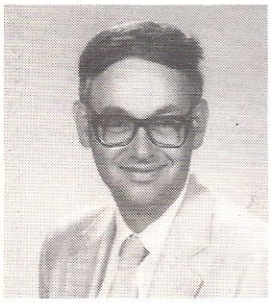
8. Osborne, W. P., Wolcott, T. J., Kopp, B. T., and Ross, M., "On the performance of trellis coded modulation with octal phase shift keying over the TDRSS channel", New Mexico State Univ. Tech. Rep. Series, NASA Grant NAG 5-1491, Dec. 1992.

S S PIETROBON

Dr. Pietrobon was born in Naracoorte, SA on 5 October 1963. He received the B.Eng. and M.Eng. degrees in electronic engineering from the South Australian Institute of Technology (now University of South Australia), Adelaide, SA, in 1986 and 1989, respectively, and the Ph.D. degree in electrical engineering from the University of Notre Dame, Notre Dame, Indiana, U.S.A., in 1991. He received a South Australian Institute Technology Medal for outstanding academic achievement for his B.Eng. degree.

In 1991 he joined the Australian Space Centre for Signal Processing, University of South Australia as a Research Fellow. In 1993 he was awarded a three year Australian Postdoctoral Research Fellowship at the University of South Australia. His research interests include convolutional coding, trellis coded modulation, satellite and deep space communications, and implementation of Viterbi and trellis decoders.

Dr. Pietrobon has been a member of the IEEE since 1986. He belongs to the Information Theory Society and Communication Society.



D J COSTELLO, JR.

Prof. Costello was born in Seattle, Washington, U.S.A., on 9 August 1942. He received the B.S.E.E. degree from Seattle University, Seattle, in 1964, and the M.S. and Ph.D. degrees in electrical engineering from the University of Notre Dame, Notre Dame, Indiana, U.S.A., in 1966 and 1969, respectively.

In 1969 he joined the faculty of Illinois Institute of Technology, Chicago, Illinois, U.S.A., as an Assistant Professor of Electrical Engineering. He was promoted to Associate Professor in 1973, and to Full Professor in 1980. In 1985 he became Professor of Electrical Engineering at the University of Notre Dame, and in 1989 was named Chairman of the Department of Electrical Engineering. In addition, he has served as a professional consultant for Western Electric, Illinois Institute of Technology Research Institute, Motorola Communications, Digital Transmission Systems, and Tomorrow, Inc. In 1991, he was selected as one of 100 Seattle University alumni to receive the Centennial Alumni Award in recognition of alumni who have displayed outstanding service to others, exceptional leadership, or uncommon achievement.

Prof. Costello's research interests are in the area of digital communications, with special emphasis on coding theory, information theory, multiuser systems, communications networks, error control, spread spectrum communications, and coded modulation. He has numerous technical publications in his field, and in 1983 co-authored (with Shu Lin) a textbook entitled "Error Control Coding: Fundamentals and Applications."

Dr. Costello has been a member if the IEEE since 1969 and was elected Fellow in 1985. He belongs to the Information Theory Society and the Communication Society. Since 1983, he has been a member of the Information Theory Society Board of Governors, and in 1986 served as President of the BOG. In 1992, he was named chairman of the Conferences and Workshops Committee of the BOG. He has also served as an Associate Editor of Communication Theory of the IEEE Transactions on Communications, as the Associate Editor for Coding Techniques for the IEEE Transactions on Information Theory, and as Co-Chairman of the IEEE International Symposium on Information Theory held in Kobe, Japan in June 1988.