Discrete Implementation of a NASA Planetary Standard Viterbi Decoder

S.S. Pietrobon
Digital Communication Group, SAIT, Adelaide SA

The design and construction of a discrete Viterbi decoder for the rate half, NASA planetary standard convolutional code is described. This soft-decision Viterbi decoder can operate at data rates up to 144 kbit/s using 84 Shottky TTL integrated circuits.

1 INTRODUCTION

As part of a consortium under contract to the Overseas Telecommunications Commission (Australia) to design and construct a ground station for the Intelsat Business Service (IBS), the Digital Communications Group at the South Australian Institute of Technology was set the task of building the modem and codec for this station [1]. The IBS ground station is expected to operate to 2 Mbit/s using QPSK modulation. The coding that is used for forward error correction is the half rate, memory six (m=6) convolutional code which is a NASA planetary standard [2]. The encoder for this code is shown in Figure 1.

In order to decode the received data from the demodulator a maximum likelihood soft-decision Viterbi decoder is specified. At high data rates the complexity of a Viterbi decoder lends itself to implementation in VLSI. However since a VLSI Viterbi decoder was not available during construction of the modem/codec an iterim solution was sought. Previous published work demonstrated how a 64 kbit/s Viterbi decoder for this code can be implemented using standard Schottky TTL integrated circuits [3]. The design produced for the IBS ground station is able to operate to 144 kbit/s using 84 integrated circuits,

After a brief description of the Viterbi algorithm the implementation of the decoder is described. This is followed by some comments on the implementation of high speed Viterbi decoders.

2 THE VITERBI ALGORITHM

Before describing the Viterbi algorithm in its maximum likelihood decoding sense a description will be given on the nature and "trellis" description of convolutional codes.

2.1 Convolutional Codes

In convolutional coding the output of the encoder depends on the input and the current state of the encoder. In this fashion the encoder can be viewed as a finite state machine. The encoder as used in the open network is given in Figure 1. This encoder has memory 6 (m=6) since the output terms will depend on the contents of these six delay elements. The number of states that the encoder can be in is $2^6=64$.

The encoder of Figure 1 adds redundancy to the transmitted sequence (two bits are transmitted per symbol period T instead of one bit, giving a coding rate of one half) such that the Euclidean distance between all the possible transmitted coded sequences is made as large as possible. The signal set as used by this code is given in Figure 2 which shows the imaginary and real components of a sinusoidal signal. The mapping used in this QPSK signal set is a Gray mapping. The squared Euclidean distances between the signal points is proportional to the number of bits that are different between the signal points.

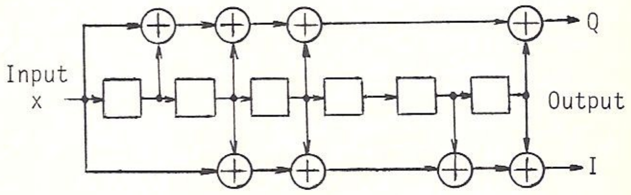

Figure 1: Memory 6, half rate convolutional encoder

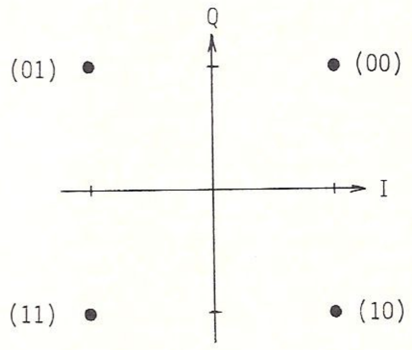

Figure 2: QPSK Signal Set with Gray Mapping.

2.2 Trellis Description of Convolutional Codes

A graphical means of illustrating convolutional codes is by the use of a trellis. A trellis shows for each state that the encoder could possibly be in, what the next state and output of the encoder will be for every possible input. With the (2,1,6) code there are 64 states with two paths leaving and entering each state and next state respectively so that a diagram illustrating a trellis of Figure 1 would be complicated. For illustration purposes the trellis of a memory two (four state) encoder (Figure 3) is shown in Figure 4. As can be seen there are two paths leaving every state and two paths entering each next state. The top path from each state represents an input of 0 and the bottom path an input of 1 into the encoder. As a sequence of symbols are transmitted a "path" is traced through a number of trellises. These transitions occur such that the next state equals the starting state of the next transition.

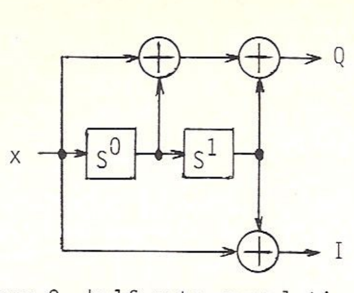

Figure 3: Memory 2, half rate convolutional encoder

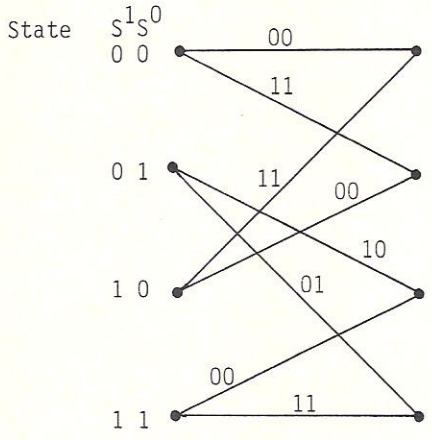

Figure 4: Trellis diagram for memory 2 encoder.

2.3 Maximum Likelihood Decoding

On receiving the transmitted signal there will be noise added from the channel and possibly non-linear distortion as well. In the presence of additive white Gaussian noise (AWGN) it is required to find the coded sequence that is the most likely to have been sent given the received noisy sequence. The Viterbi algorithm provides an effective means of finding the best possible sequence given the received soft detected sequence. This is accomplished by finding , for each of the the 2^m states, the most likely path (or sequence) leading into that state and eliminating the other paths. This implies that at any one time there are only 2^m paths being considered out of the possible sequences that could have been sent.

Let $\mathbf{x}_i=(x_i^0,\ldots,x_i^{k-1})$ represent the k information bits at time i of the information sequence $\mathbf{x}$. The encoder will produce a transmitted sequence $\mathbf{y}$ with n transmitted bits at time i, $\mathbf{y}_i=(y_i^0,\ldots,y_i^{n-1})$. On passing through the channel the received sequence $\mathbf{r}$ is assumed to be in the form of quantised values of I and Q which are the Inphase and Quadrature components of the received signal respectively. The encoder of Figure 1 has k=1, n=2 and is used with QPSK modulation. At the demodulator, the received I and Q values are quantised into eight levels or three bits each. Given that the noise components of the received samples are independent of one another then the likelihood function is

$$P(\mathbf{y}|\mathbf{r}) = \prod_i P(\mathbf{y}_i|\mathbf{r}_i), \tag{1}$$

In maximum likelihood decoding we wish to find the sequence $\mathbf{y}$ that maximises (1). After taking the logarithm of (1) we need to maximise

$$\log P(\mathbf{y}|\mathbf{r}) = \sum_i \log P(\mathbf{y}_i|\mathbf{r}_i). \tag{2}$$

By inverting the probabilities this can be restated as minimising

$$-\log P(\mathbf{y}|\mathbf{r}) = \sum_i -\log P(\mathbf{y}_i|\mathbf{r}_i). \tag{3}$$

We can redefine the terms in (3) as

$$M(\mathbf{y}|\mathbf{r}) = \sum_i BM(\mathbf{y}_i|\mathbf{r}_i), \tag{4}$$

where $M(\mathbf{y}|\mathbf{r}) = -\log P(\mathbf{y}|\mathbf{r})$ is defined as the "metric" of the transmitted sequence $\mathbf{y}$ given the received noisy sequence $\mathbf{r}$ and $BM(\mathbf{y}_i|\mathbf{r}_i) = -\log P(\mathbf{y}_i|\mathbf{r}_i)$ is defined as the "branch metric" for the transmitted symbol $\mathbf{y}_i$ given the received noisy symbol $\mathbf{r}_i$ at time i.

In Viterbi decoding we determine the metrics of the most likely sequence (the path with the smallest metric) into each state that the encoder could have been in at time i. The metrics associated with each state are called "state metrics" or SM. For the received symbol $\mathbf{r}_i$ we determine the branch metrics (BM) for the 2^n possible transmitted symbols $\mathbf{y}_i$ at time i. From the trellis structure the appropriate BM's are then added to the SM's at time i to form 2^{m+1} new metrics at time i+1. For example (see Figure 4) the BM for the transmitted point 00 is added to the state metric of state 0 to form the top path from state 0 (which then goes to state 0 again). The two new paths into each state are then compared and the path with the smallest metric is chosen. The other path is eliminated since it is less likely. This is commonly called the Add-Compare-Select (ACS) operation. The values that the BM's can take are usually quantised from a minimum to a maximum value. This quantisation is chosen so as to optimise the reliability of the path decisions. Since a finite number of quantisation levels is used, the decoder is not truly optimal. Other factors reducing optimality is the length L of the stored ACS decisions and the number of quantisation levels used to represent the received values for I and Q. Due to the nature of the algorithm, soft decisions can be readily accomodated in hardware (or software). They provide approximately 2 dB in coding gain over hard decisions. References [4-5] provide additional detail on the Viterbi decoding algorithm.

4 IMPLEMENTATION

The major sections of the Viterbi decoder are

.Branch Metric Calculator	(BMC),
.State Metric Calculator	(SMC),
.Survivor Sequence Memory	(SSM),
.Signal Set Synchronisation	(SSS),
.Minimum State Metric Selector	(MSMS).

A block diagram showing the relationship between these five basic sections is given in Figure 5. The demodulator I and Q outputs are each three-bit quantised. For each value of I and Q the BMC calculates four Branch Metrics (BM) corresponding to the four possible transmitted signal points. The SMC finds the 64 new State Metrics (SM) given the previous state metrics and the newly determined BM's. This requires 64 Add-Compare-Select (ACS) operations. The MSMS finds the smallest new SM. The SSS monitors the rate at which these minimum metrics are increasing. This is necessary since the signal may have been rotated by 90^o in the demodulator which will cause erronous decoding. If the rate of increase is too high this indicates that a phase rotation is likely to have occurred. A

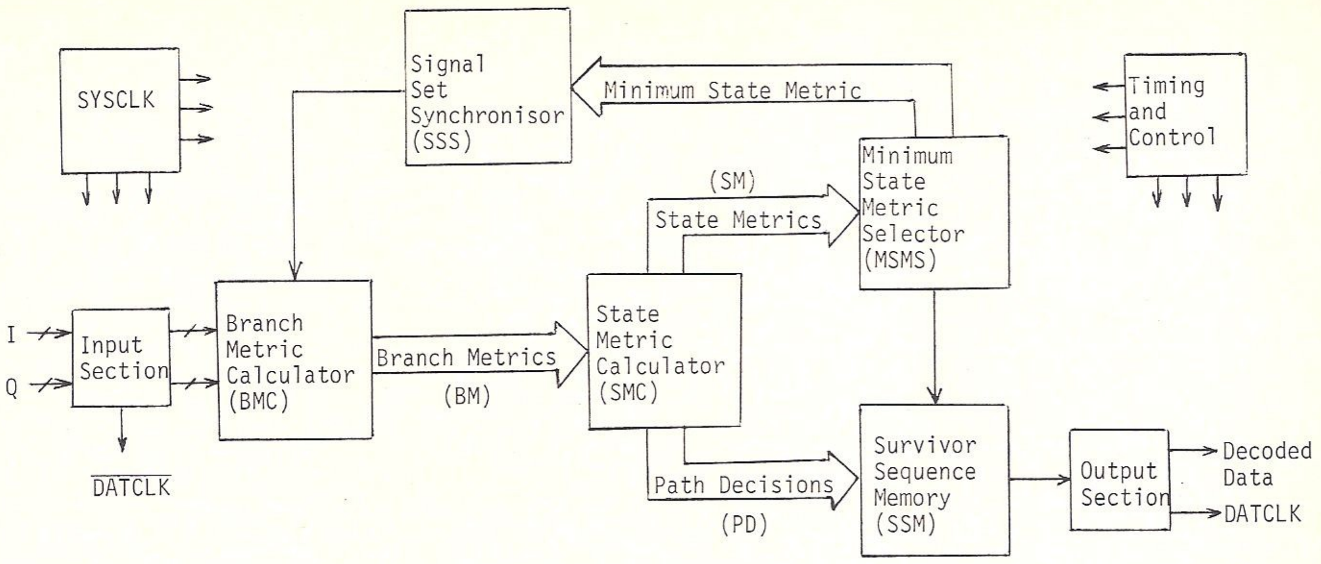
SYSCLK
Signal Set Synchronisor (SSS)
Minimum State Metric
Timing and Control
Minimum State Metric Selector (MSMS)
(SM)
State Metrics
I
Q
Input Section
DATCLK
Branch Metric Calculator (BMC)
Branch Metrics
(BM)
State Metric Calculator (SMC)
Path Decisions
(PD)
Survivor Sequence Memory (SSM)
Output Section
Decoded Data
DATCLK

Figure 5: Block Diagram of Serial Viterbi Decoder.

signal is then provided to the SSS so that any 90° phase ambiguity in the signal set is resolved. The SSM must store the path decisions and also perform the traceback operation. Traceback involves starting with the state with the minimum metric (determined from the MSMS) and "tracing back" through the past trellis decisions until reaching the last recorded set of path decisions. The last decision bit found in the traceback is the decoded bit. This traceback is performed for every pair of received I and Q values to produce the decoded bit stream.

The serial operation of a Viterbi decoder requires that each ACS operation is performed one at a time. Thus there is only one ACS circuit. Note that a fully parallel decoder has an ACS circuit for each state (this is why the complexity of a parallel Viterbi decoder is proportional to 2^m) and all the operations can be achieved in a short period of time. The speed of a parallel decoder can be very high. However with serial implementation the decoder complexity increases incrementally with m (except for memory requirements) and the speed is proportional to 2^{-m}.

Using serial implementation, a single ACS is used 64 times with each ACS operation taking one clock cycle to complete. The internal clock needs to be at least 64 times the symbol rate in order to calculate the 64 new SM's. While the new state metrics are being calculated they need to be stored. Similarly the path decisions need to be stored. This design is based on the state metrics being stored in two static random access memories (RAM) (device number 82S09, 64x9 addressing and 45 ns access time) and the path decisions in another two static RAM's (device number 93425A, 1024x1 addressing and 30 ns access time). These RAM's were chosen as they enable the internal clock speed to be in the order of 10 MHz (100 ns period).

Pipe-lining and parallel processing have been extensively used in order that all required operations can be accomplished in the time available. For example the SMC has the operations of reading state metrics, ACS and storing state metrics. These operations are occuring in parallel, each following the other. This effect increases the number of clock cycles by three to 67 clock cycles. Since the decoder operates at a much higher clock rate than the symbol rate, the decoder was made to operate asynchrously. This also allows for a simpler design. In this way the internal clock speed is kept constant and the decoder can easily operate at a variety of received symbol rates. The internal decoder clock is called SYSCLK and the received symbol clock is named DATCLK. The effect of synchronisation requires an extra clock cycle plus a fraction of a clock cycle (assuming there is no jitter on DATCLK). Rounding up, the number of clock cycles per decoded bit is brought to 69. If a 10 MHz SYSCLK is used the maximum DATCLK rate is

$$\begin{aligned} f_{max}(\text{DATCLK}) &= f(\text{SYSCLK})/69 \\ &= 145 \text{ kbit/s.} \end{aligned}$$

Parallelism is introduced by producing the BM's and having the traceback operation occur at the same time as the SM's are being calculated. The serial Viterbi decoder can be seen as a highly dedicated parallel computer where address generation for reading and writing into the memories and controlling operations are fixed by two counters.

4 HIGH SPEED DECODING

In high speed Viterbi decoders there is little time to perform all the required operations. For example, if a data rate of 2 Mbit/s is to be achieved and a serial approach is used then SYSCLK would have to be at least 138 MHz. Due to the complexity of the serial method this would be very difficult or almost impossible to construct with off-the-shelf components.

An alternative approach is to to do all the calculations in parallel. That is, there would be 64 ACS circuits which would be hard wired together for the passing of state metrics between the states. For the SSM the register exchange technique can be used [4]. The speed of the decoder will now be the same as DATCLK. However, the number of integrated circuits will be around 1000. This is not impossible to implement, but would be expensive, bulky and comsume a large amount of power.

There is also the possiblity of using a

serial/parallel approach. For example, a decoder would have four ACS circuits to perform four ACS operations in each SYSCLK period. SYSCLK would then only need to be 21 times DATCLK (16 cycles to calculate all the state metrics and 5 overhead cycles). This type of system could be implemented in ECL (emitter coupled logic) with about 150 chips which is still quite complicated.

A means of overcoming these problems is the use of VLSI technology which is being investigated by the Digital Communications Group. A fully parallel decoder has a number of problems. The most important of these is the interconnection of all the ACS circuits with themselves. This involves having eight lines emanating from each ACS and 16 lines entering each ACS for the passing of the SM's. All this wiring would take a considerable amount of chip area. Since chip area is a vital resourse this wiring area needed to be reduced. This was solved by having each ACS circuit operate on the SM's serially. Thus, there is now only one wire leaving and two wires entering each ACS. This greatly reduces the required wiring and the complexity of each ACS circuit. However, SYSCLK would now be six to eight times DATCLK. The implementation of the SSM would also take advantage of the higher SYSCLK clock rate.

5 CONCLUSION

A serial Viterbi decoder for a half rate 64 state convolutional code (NASA planetary standard) with QPSK mapping has been described. This decoder which uses 84 standard SSI and MSI Schottky integrated circuits is able to operate from DC to up to 144 kbit/s (provided timing jitter is not too severe). Although this decoder was built mainly for the IBS modem/codec it can be used in many other applications where this code is used.

A production version of this decoder should have a chip count of 71. This compares favourably with the serial decoder built by [3] which can operate to 104 kbit/s using 61 integrated circuits. The implications of implementing Viterbi decoders for this code at higher data rates has also been described.

6 REFERENCES

[1] W. G. Cowley and I. S. Morrison, "Modular System Architectural Approach to Modem/Codec Design for a Satellite Earth Station" Third National Space Engineering Symposium, Canberra, June 30 - July 2, 1987.

[2] M. K. Simon and J. G. Smith, "Alternate Symbol Inversion for Improved Symbol Synchronisation in Convolutionally Coded Systems", IEEE Trans. on Commun., COM-28 pp 228-237, February 1980.

[3] A. Shenoy and P. Johnson, "Serial Implementations of Viterbi Decoders", COMSAT Technical Review, Vol. 13, No. 2, Fall 1983, pp. 315-329.

[4] G. C. Clarke and J. B. Cain, "Error-Correction Coding for Digital Communications", Plenum Press, New York and London, 1983.

[5] S. Lin and D. J. Costello, Jr., "Error Control Coding: Fundamentals and Applications", Prentice Hall, Englewood Cliffs, 1983.