Design of A Viterbi Decoder for Satellite Communication

Jung-Il Han, Staff Engineer, Jong-Moon Choi, Engineer, Woo-Young Choi, Professor,
Bong-Ryul Kim, Professor
Microelectronics Lab., Dept. of Electronic Eng., Yonsei University

[ABSTRACT]

In this paper, we proposed a Viterbi decoder for practical implementation of Forward Error Correction(FEC). The design was mainly focused on real-time processing by using the memory organization and its control method with constraint length 7, coding rate 1/2. Management of memory contents in a Viterbi decoder is a major design problem for both hardware and software realization. In this design, we solved that problem with sequence processing. The design was done with a standard 0.6 ㎛ CMOS process and 2-layer metal technology. It is composed of logic circuit with 15,887 gates. It runs on a 3.3 V supply and operates at 20 MHz. The estimated power consumption is about 98 ㎽ and the chip size is about 4 ㎜ × 4 ㎜.

I. Introduction

In many digital communication systems, error correction circuitry is widely used in order to reduce the bit error rate of transmitted data [2,5]. Convolutional coding with Viterbi decoding is a powerful method for forward error correction [1-5]. In present commercial applications, Viterbi decoders with R(coding rate)=1/2 and K(constraint length)=7 are widely used.

A very simple variation of the Viterbi algorithm permits the use of soft-decision demodulated data in which signal values are quantized into multiple levels and digitized [2,5].
The advantage of soft decision demodulation is that the signal values not only indicate whether they represent one or zero, but also indicate the magnitude of the corruption of the signal by noise at the instant of quantization. A significant processing gain can be achieved by using soft decision data, typically about 2 dB for 3-bit data.

We designed a Viterbi decoder that has R=1/2, K=7 and 3-bit soft decision. Our design has a trace-back architecture and achieves tracing, updating and storing of the real-time sequences within a single clock cycle.

II. VITERBI Algorithm

The Viterbi algorithm is widely known to be an optimal decoding algorithm for convolutional codes. The code paths of a convolutional code can be described by a trellis diagram which is a time expansion of state diagram [1-3]. This performs the maximum likelihood decoding [1-5]. The advantage of Viterbi decoding is that the complexity of the decoder is not a function of the number of symbols in the codeword sequence [2]. The algorithm determines a measure of similarity between received signals, at time t_i, and all the trellis paths entering each state at time t_i. Then, it removes trellis paths that can not be a candidate for the maximum likelihood choice. When two paths enter the same state, the one having the smallest metric is chosen. This path is called the survivor path. The selection of survivor path is performed for all the states. The decoder continues in this way and makes decisions by eliminating the least likely paths [2-5]. The early rejection of the unlikely paths reduces the decoding complexity [2].

Viterbi decoder has three major parts. The first part calculates branch metrics. The branch metric is the likelihood metric of received codes, and is calculated based on the comparison between the trellis and received convolutional codes [2-5]. The second part is "Add-Compare-Select(ACS)" part. In ACS, path metrics are updated by adding branch metrics associated with each possible state transition [6]. The number of states N of a convolutional encoder which generates n encoded bits is a function of the constraint length K and input bits b [1,2].

$$N = 2^{b(K-1)} \qquad (1)$$

$$R = b/n \qquad (2)$$

The third part performs path selection and memery managenent. The smaller path metric is the path metric for the state and the resulting decision is stored in the survivor sequence memory management unit where the most likely path is determined [6]. Fig. 1 shows the block diagram of the convolutional encoder with K = 7, R = 1/2 which is widely used in satellite communication and DBS(Digital Broadcasting System). Fig. 2 also shows the block diagram of the Viterbi decoder.

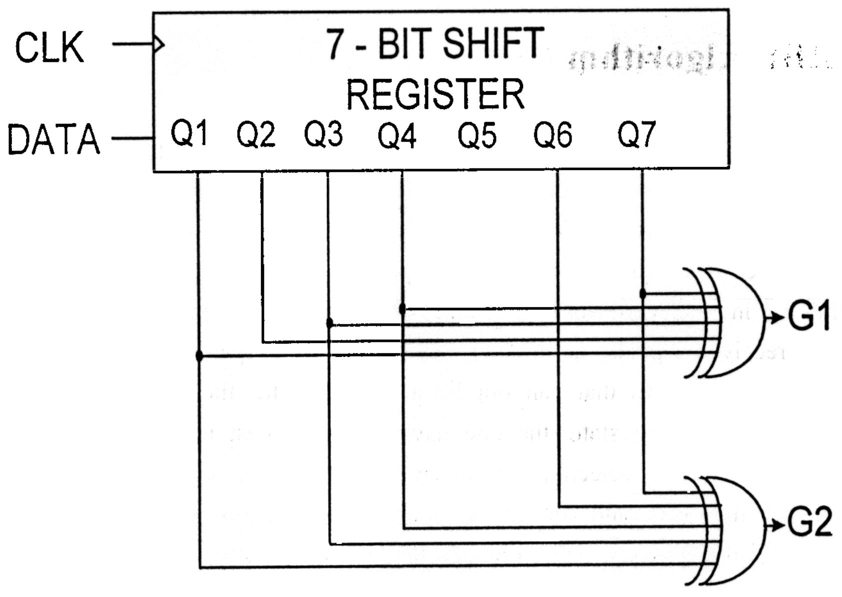

Fig. 1. Block diagram of the convolutional encoder(K = 7, R = 1/2)

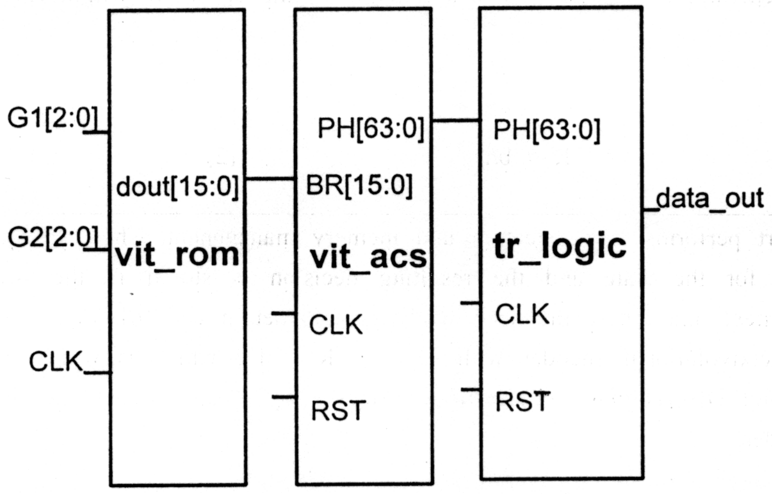

Fig. 2. Block diagram of the Viterbi decoder

Ⅲ. The Sequence Processing

The sequence processing means that data come out without suspension. When the data are sent to destination, the time it takes for the decoder to correct the errors in data degrades system performance. To solve this problem, our decoder is designed with sequence-processing capability. This makes it possible to implement the decoder with real-time processing [6].

There are two known methods for the storage of survivor sequences from which the decoded sequence is retrieved[9,10]. The register exchange method is the simplest conceptually and a commonly used technique. Because of the large power consumptions and large area required in VLSI implementations of the register exchange method, the traceback method is the preferred method in the design of large constraint length, high performance Viterbi decoders. The traceback method stores path information in the form of an array of recursive pointers. Unfortunately, a direct implementation of the traceback method requires further thought, snce it treats memory as infinite in size, while any actual implementation contains only finite memory resources with a limited number of address and data ports.

It is advantageous to think of traceback memory as organized in a two-dimensional structure with rows and columns. The number of rows is equal to the number of states. Each column stores the results of N comparisons corresponding to one symbol interval or one stage in a trellis diagram. Since the stream of symbols is, in general, semi-infinite, storage locations are periodically reused. There are three types of operations performed inside a traceback decoder.

Traceback Read(TB) - This is one of the two read operations and consist of reading a bit and interpreting this bit in conjunction with the present state number as a pointer that indicates the previous state number. Pointer values from this operation are not output as decoded values, instead they are used to ensure that all paths have converged with some high probability, so that actual decoding may take place. The traceback operation is usually run to a predetermined depth T before being used to initiate the decode read operation.

Decode Read(DC) - This operation proceeds in exactly the same fashion as the traceback operation, but operates on older data with the state number of the first decoder read in a memory bank being determined by the previously completed traceback. Pointer values from this operation are the decoded values and are sent to the bit-order reversing circuit. A decode read can serve as a dual decode and traceback read, this allows us to decode read multiple columns using one traceback read operation of T columns.

Writing New Data(WR) - The decisions made by the ACS are written into locations corresponding to the states. The write pointer advances forward as ACS operations move from one stage to the next in the trellis, and data are written to locations just freed by the decode read operation.

For every set of column write operations, an average of one decode read must be performed. The overhead of T-column traceback read can be spread over one or more column decode read operations, resulting in k read operations(k > 1); k includes both decode read operations and traceback read operations.

Traceback algorithms consist of k-pointer algorithm and one-pointer algorithm. The one-pointer algorithm differs significantly from the k-pointer algorithm. Instead of utilizing k read pointers to perform the k reads for every column write operation, we chose to use a single read pointer, but accelerate read operations, so that every time the write counter advances by one column, k column reads occur. This acceleration of read operations is made possible by the fact that among the three operations, writing new data, traceback read and decode read, writing new data is by far the most time consuming. The one-pointer algorithm with $k_l = 3$ is illustrated in Fig. 3. Only $k_l + 1$ memory banks, each $T/(k_l - 1)$ columns long, are required, for a total of $(k_l + 1)T/(k_l - 1)$ columns. A single read pointer produces the decoded bits in burst. While reading k_l memory banks, no decoded data are available from the first $k_l - 1$ memory banks. During the decode read operation in the k_lth memory bank, decoded bits are generated at a rate of k_l per stage. Fortunately, the two-stack structure can perform both bit order reversal and burst elimination at the same time. We utilized the one-pointer algorithm for memory management in traceback operation of state values.

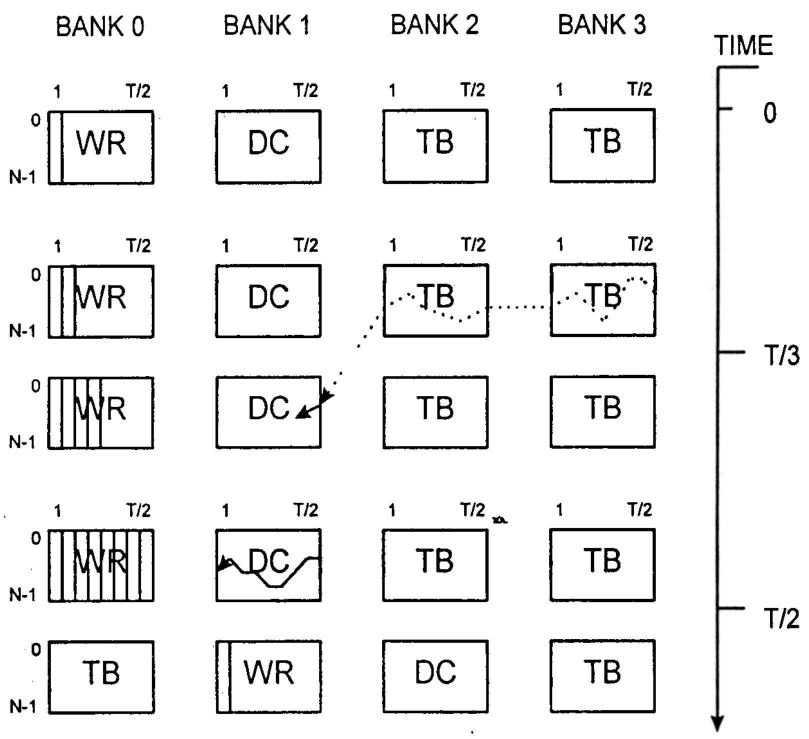

Fig. 3. Survivor sequence update in one-pointer method

Forney has shown that if τ, defined as memory path, is on the order of five times the encoder memory or more, decoding decision is maximum likelihood [8]. Thus, the value traced back with enough depth in any state converges into a fixed position. This property can be implemented by four RAMs.

Table 1 shows the sequence processing using four RAMs. At t0, 64 state values are written into RAM0. At t1, they are also written into RAM1. At t2, they are written into RAM2, and the state values written in RAM1 are read. At t3, they are written into RAM3 and the state values written in RAM0 are read. The state values read in RAM0 are decoded in tr_back module and decoded data are written in TB_RAM0. At the same time, the state values written in RAM2 are read to find convergence value in RAM1. At t4, they are written into RAM0, and the state values written in RAM1 are read. Then, they are decoded in tr_back module and decoded data are written into TB_RAM1. Simultaneously, the state values written in RAM3 are read to find convergence value in RAM2 and the decoded data written in TB_RAM0 are read. After this, decoded data come out of the Viterbi decoder continuously, and the process is similar to what was performed at t4. Fig. 4 shows the block diagram of the hardware implementation of trace-back logic which performs the sequence processing. Fig. 5 shows the tr_back module which actually performs the trace-back operation.

Table 1. Management of memory for the Sequence Processing

	t0	t1	t2	t3	t4
RAMA addr_1	Write 0→127			Read 127→0	Write 0→127
RAMB addr_2		Write 0→127	Read 127→0		Read 127→0
RAMC addr_1			Write 0→127	Read 127→0	
RAMD addr_2				Write 0→127	Read 127→0
TB_RAM0 addr_2				Write 0→127	Read 127→0
TB_RAM1 addr_1					Write 0→127

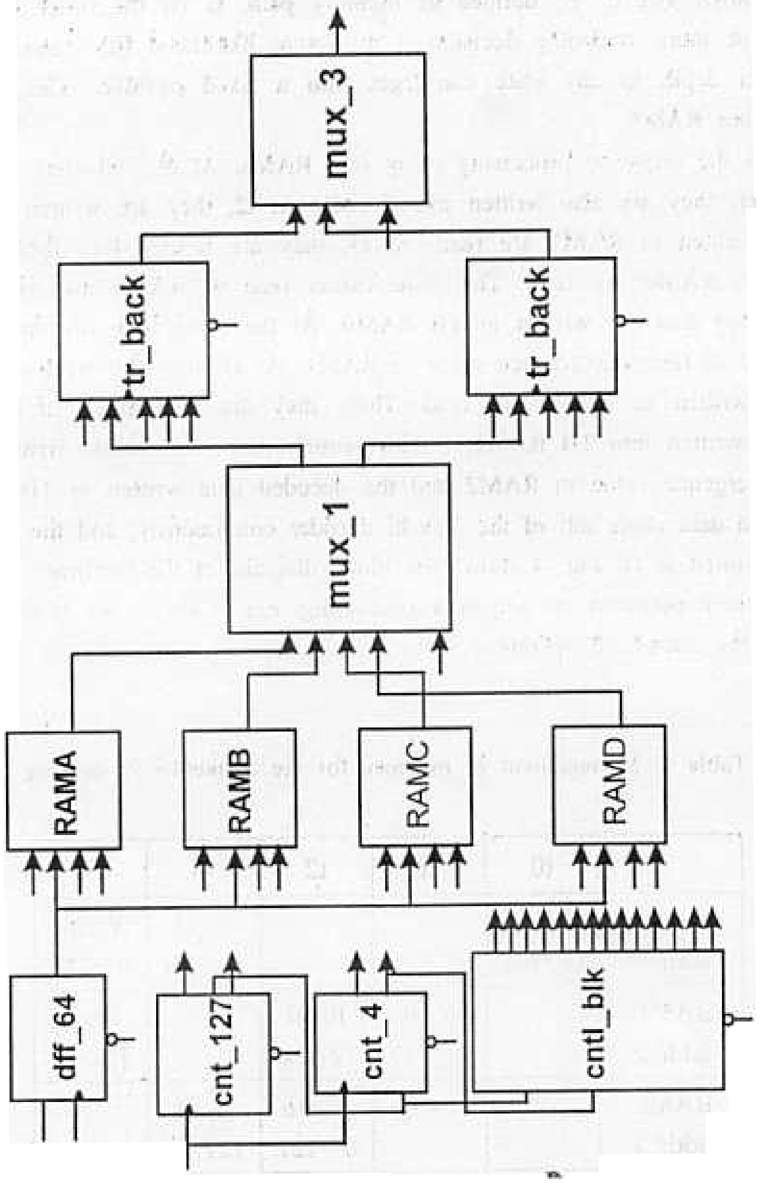

Fig. 4. Block diagram of trace-back logic

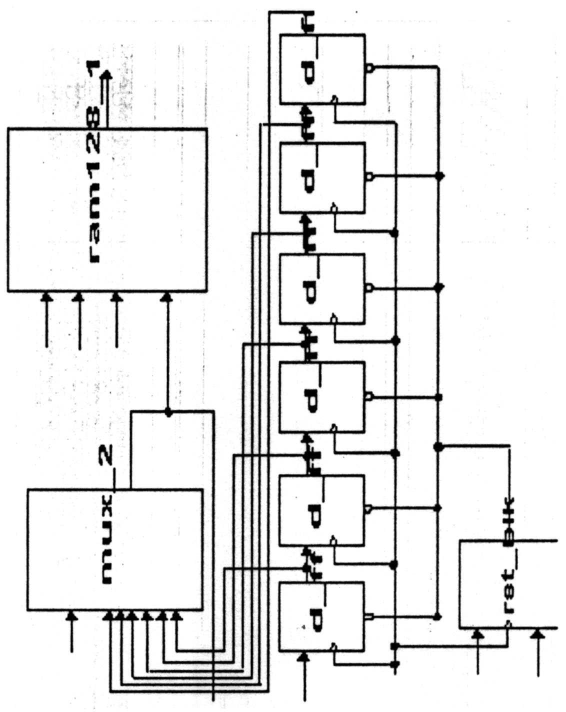

Fig. 5. Block diagram of tr_back

Ⅳ. Simulation And Layout Results

To confirm the operation of Viterbi decoder designed for real-time processing, we performed simulation with VHDL(Very high speed integrated circuit Hardware Description Language) [7]. We also performed the gate level simulation. Viterbi decoder designed in this paper was simulated and synthesized. As test bench, we put '1', '1', '0', '0', '1', repeatedly. Fig. 6 shows the gate level simulation results which make it possible to confirm the sequence processing.

Fig. 7 shows layout of our Viterbi decoder. The design was done with a standard 0.6 ㎛ CMOS process and 2-layer metal technology. It is composed of logic circuit with 15,887 gates, four 128×64 RAMs, two 128×1 RAMs, and one 64×16 ROM. It runs on a 3.3 V supply and operates at 20 MHz. The estimated power consumption is about 98 ㎽ and the chip size is about 4 ㎜ × 4 ㎜.

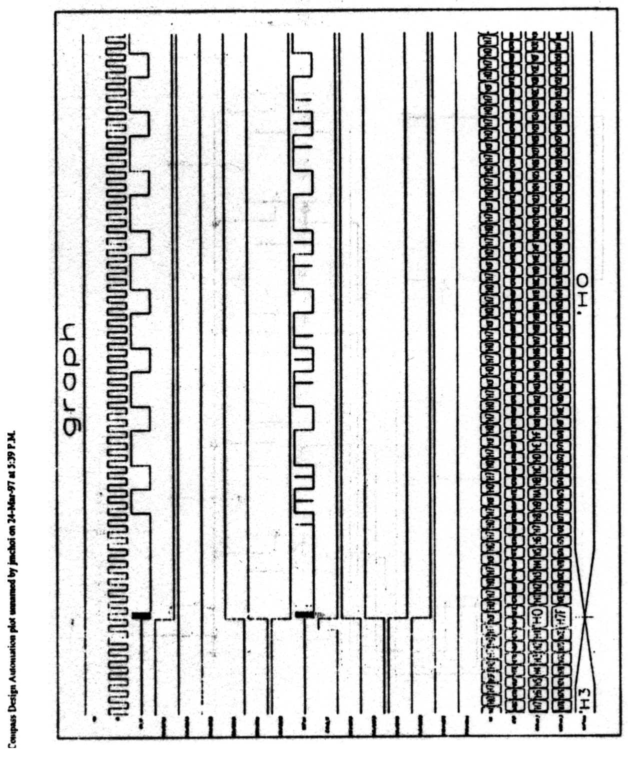

Fig.6 level simulation results

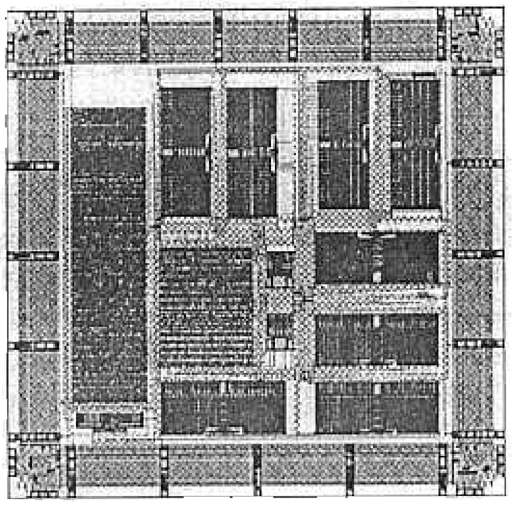

Fig. 7. Layout of designed Viterbi decoder

V. Conclusion

We have designed Viterbi decoder which performs the sequence processing. Trace-back logic architecture is based on the management of the memory contents. We have used one-pointer method for memory management in traceback of state values. The operation of our design was confirmed with VHDL simulation, and the actual circuit was synthesized and laid-out.

References

[1] A. J. Viterbi, "Convolutional Codes and their Performance in Communication Systems," IEEE Trans. Commun., Vol. 19, pp. 751-772, Oct. 1971.

[2] Bernard Sklar, Digital communications fundamentals and applications, Prentice-Hall, 1989.

[3] A. J. Viterbi, "Error Bounds for Convolutional Codes and Asymptotically Optimum Decoding Algorithm," IEEE Trans. Information Theory, Vol. IT-13, pp. 260-269, April 1967.

[4] G. D. Forney, "The Viterbi Algorithm," Proc. IEEE, Vol. 61, pp. 268-278, Mar. 1973.

[5] A. J. Viterbi and J. K. Omura, Principles of Digital Communications and Coding, McGraw-Hill : New York, 1979.

[6] C. Y. Chang and K. Yao, "Systolic Array Processing of the Viterbi Algorithm," IEEE Trans. Information Theory, Vol. 35, No. 1, pp. 76-86, Jan. 1989.

[7] J. R. Armstrong, "Chip Level Modeling with VHDL," Prentice-Hall, Englewood Cliffs, 1989.

[8] G. D. Forney, "Convolutional Codes II : Maximum Likelihood Decoding," Inform. Theory, Vol. 25, pp. 222-226, July 1974

[9] G. Feygin and P. G. Gulak, "Architectural Tradeoffs for Survivor Sequence Memory Management in Viterbi Decoders," IEEE Trans. Commun., Vol. 41, No. 3, pp. 425-429, Mar. 1993

[10] C. M. Rader, "Memory Management in a Viterbi Algorithm," IEEE Trans. on Commun., pp. 1399-1401, Sep. 1981