Performance Test of Viterbi Decoder for Wideband CDMA System

Jang-Hyun Park

Signal Processing Section
ETRI
Daejon, Korea, 305-350
Tel: 82-42-860-5691
Fax: 82-42-860-6482
e-mail: jhpark@amadeus.etri.re.kr

Yea-Chul Rho

Signal Processing Section
ETRI
Daejon, Korea, 305-350
Tel: 82-42-860-5444
Fax: 82-42-860-6482
e-mail: ycrho@amadeus.etri.re.kr

Abstract — This paper describes the design, the implementation, and the performance test of the Serial Viterbi decoder(SVD) using VHDL and FPGAs. The decoding scheme assumes the transmitted symbols were coded with a K=9, 32Kbps, and rate 1/2 convolutional encoder with generator function $g_0=(753)_8$ and $g_1=(561)_8$ as defined JTC TAG-7 W-CDMA PCS standard. The SVD is designed using VHDL and implemented using FPGAs. Main algorithm except memories is implemented in two Altera FLEX81500 FPGAs. And the performance test results with 3DB Gaussian noises show that the function of SVD works well.

I. INTRODUCTION

The CDMA[1] cellular mobile communication system employs the Viterbi algorithm, which is a powerful error-correction code technique, for decoding data confidently[2]. The Viterbi decoder operates by finding the most likely decoding sequences for an input code symbol stream. The encoding process adds correlation to the input data stream to produce the encoded symbol stream. The symbol stream is subject to noise corruption and channel fading. The decoder analyzes this corrupted stream and exploits the known correlation to reconstruct the original symbol and data bit streams.

This paper describes aspects of the Viterbi algorithm apprised to convolutionally encoded binary sequences, the application of Viterbi algorithm to the serial Viterbi decoder, and the implementation of the SVD using FPGA chip and commercial memories. The SVD uses the Viterbi algorithm to near optimally decode a synchronized and quantized code symbol. The decoding scheme assumes the transmitted symbols were coded with a K=9, 32Kbps, and rate 1/2 convolutional encoder with generator function $g_0=(753)_8$ and $g_1=(561)_8$ as defined JTC TAG-7 W-CDMA PCS standard[3]. The Viterbi decoder called a SVD because the ACS(Add-Compare-Select) array processing is performed serially with a single ACS pair in order to reduce the total amount of circuitry required.

This paper is organized as follows. In section 2 we explained about Viterbi algorithm. In section 3 the major blocks and functions of Viterbi decoder are described. In section 4 we describe about simulation and performance test. Finally the conclusion is in section 5.

II. VITERBI ALGORITHM[4][5][6]

A Viterbi decoder and a convolutional encoder operate by finding the most likely decoding sequences for an input code symbol stream. A convolutional encoder is selected for error-correction with digital mobile telecommunication. Because the structure of the convolutional encoder is simple, and the encoder is powerful for AWGN channel environment, and it is possible to correct the burst errors using the interleaver.

The decoding unit supports the decoding for the sync channel, the paging channel, and the traffic channel. The transmitting data is decoded from a channel with code rate 1/2. The sync and the paging channel data is transmitted at 32Kbps and the decoding unit receives a 160-symbol every 5ms frame(32Kbps*5ms=160 symbols). The traffic channel data is transmitted at 72Kbps, and the decoding unit receives a 360-symbol every 5ms frame(72Kbps *5ms=360 symbols).

A. Convolutional Encoder[4]

Convolutional codes add correlation to the input data sequences by using sequential elements(flip-flop) and modulo adders. Binary convolutional encoders can be implemented with a feedforward shift register and exclusive-or gates (modulo-2 adders) as shown in Fig.1. The encoder shown in Fig.1 contains a shift register tapped at various positions. The shift register taps terminate at modulo-2 adders forming generator function(g_0 and g_1). Input bits enter the shift register by one bit at a time. The outputs of the generator functions in Fig.1 produce an output symbol for each input bit, which corresponds to a code rate of 1/2.

As an example of how the encoding process operates, consider the binary information sequence 0101110. The resulting output sequence of the example encoder becomes 00 11 01 00 10 01 10, as shown in Fig.1.

Ex) K=3, $g_0=(05)_8$, $g_1=(07)_8$

Input : 0 1 0 1 1 1 0
Output : 00 11 01 00 10 01 10

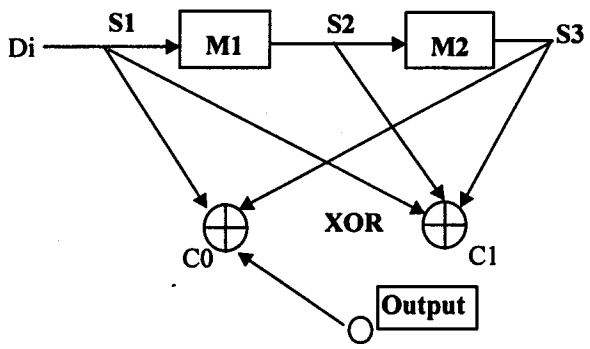


Fig.1. K=3 Convolutional Encoder

B. State Diagram and Trellis Diagram

The binary memory elements in the shift register define the state of the encoder. So, an encoder containing a shift register with n memory elements has 2^n possible states. The number of delay elements in the serial shift register plus 1 establish the constraint length, K, of a convolution encoder. Using this formula, the constraint length of the encoder shown in Fig.1 is 3. Correlation enters into the system because each input data bits will influence K sets of output symbols. For example, K=3 and code rate 1/2 as shown in Fig.1. State numbers are 2^2=4 because K is 3. There exist 4 states(00, 10, 01, 11). The encoder is assumed to begin in the zero state. First zero state moves to state 00 or state 01 with convolutional code(c0, c1) 00 or 11.

A trellis diagram showing the possible paths of the example encoder is shown in Fig.2. The encoder is assumed to begin in the zero state. The left-side numbers of slash represent the input bits entering the encoder. The right-side numbers of slash represent C_0 and C_1, the output code symbols. The dark circles are called nodes, which correspond to the state of the encoder at a given level. Each line represents a branch. For all binary encoders using a serial shift register, there are two possible branches entering each node, and two possible branches leaving each node in steady state(regardless of the code rate).

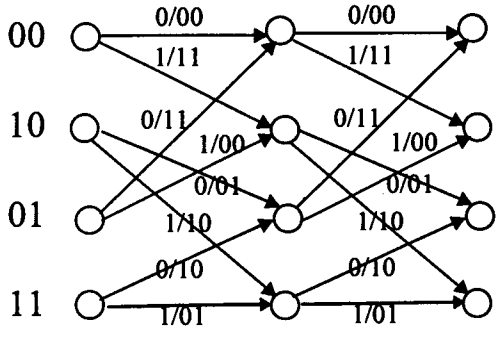


Fig.2. K=3 Trellis Diagram

C. Viterbi Decoding Algorithm

The Viterbi decoder operating by finding the most likely decoding sequences for an input code symbol stream. The four main steps in this process are :

- o. Branch metric generation
- o. State metric generation
- o. Add-Compare select
- o. Trace Back

Branch metrics are functions of probabilities. The probabilities are based on groups of 2 consecutive soft decision received code symbols for code rate 1/2. Path metrics are calculated by adding the appropriate branch metric to the state metric of the previous level. State metrics are the surviving path metrics of the current state. Surviving branches (paths) are chosen based on path metrics. Trace-Back is when the most likely transition into each state are saved in path memory for many bit symbols, and then the decoder traces and chains backward in time through the most likely sequence to choose the decoded output.

D. Decoding of W-CDMA channel

The W-CDMA channel is applied by convolutional encoder with code rate 1/2, constraint length K=9, and generation polynomial $(753, 561)_8$. Input bits enter the shift register one bit at a time. The outputs of the generator functions produce an output symbol for each input bit, which corresponds to a code rate of 1/2. If constraint length is K, the number of state of the trellis diagram is 2^{k-1}, and the number of butterfly decision is $2^{k-1}/2$ pair. One decision butterfly out of 128 pair is shown in Fig.3.

Here X is 7 bits data of the current state except MSB. And that is not changed part from state transition. 0x state represents for S with convenience. Two branches are shown merging at one state at the transition.

As an example of the Viterbi decoding process, consider the information sequence 11, that is sent through the encoder in Fig.1, to produce the encoded sequence 11, 10. This sequence is transmitted to the channel where it is corrupted by noise and fading. Suppose that the received sequence of the soft decision symbol is as shown in Fig.4. The received signal is quite corrupted. One of the received symbols has

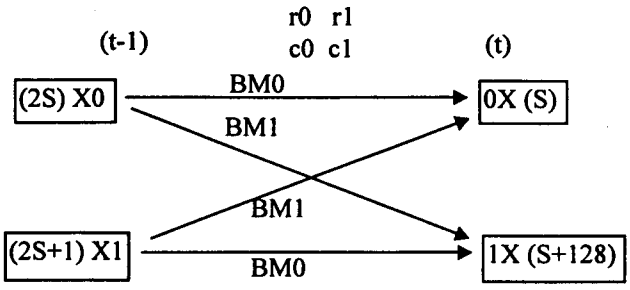


Fig.3. Decision Butterfly for ACS Pair

different polarity with the corresponding transmitted symbols. The decoding process begins by initializing the decoder. It is assumed that the convolutional encoder began its encoding in the zero state. The Viterbi decoder ensures that the decoded sequence begins in the zero state by placing very large state metric on every state except the zero state for level zero. Fig.4 shows the metric calculation for the two transitions. Surviving paths are shown as solid lines. Nonsurviving paths are shown as dotted lines. The state metric for each state is shown above the corresponding node. Note that after the state metric has been calculated, nonsurviving path metrics may be wholly discarded. This process continues until the specified traceback depth is reached. Once the traceback depth has been reached, the decoder can trace the path with the best state metric to determine the most likely transmitted sequence. Fig.4 shows the decoded sequence with a bold line. Decoded symbols are simply the hypotheses of the most likely path. Decoded information bits can be determined by stepping through each level, and recording the most significant bit in the for each level in the most likely path.

III. THE DESIGN OF THE SVD

A. The Specification and Organization of the SVD

1) Serial Viterbi Decoder with 5 MHz Bandwidth
2) Code Rate : 1/2
3) Constraint Length : K=9
4) Generator Polynomials : $g_0 : (753)_8$, $g_1 : (561)_8$
5) Trace Back Depth : 64
6) 1 frame : 5 ms (input and decoding)
7) Data Rate : 32Kbps(Sync and Paging Ch.)
   72Kbps(Traffic Ch.)
8) 4 bits soft decision and 2's complement format

Data : 11
Transmitted : 1110 (Convolutional Codes)
Received : 1010 (+3 -4 +5 -4 : 4 bits soft decision)

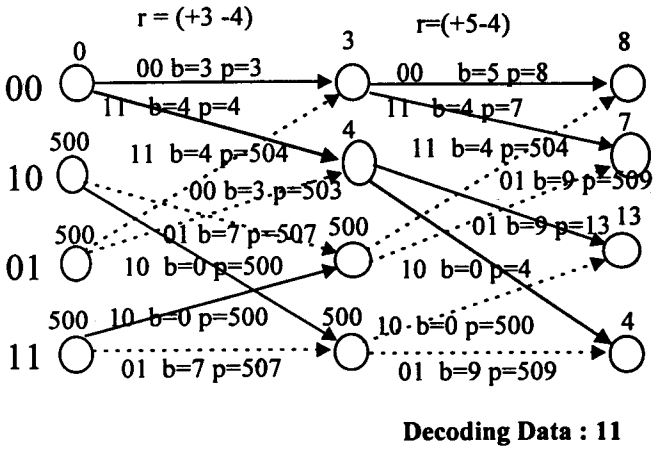


Fig.4. Example of Viterbi Decoding

9) Data Transmission format : Continuous Only
10) Depunctured data from Deinterleaver
11) Clock : 4.096 MHz * 8
12) Output : CPU : 8 bits Data Bus using Interrupt
    ADPCM : 1 bit data using 1MHz and 8KHz clock

The five major blocks of the SVD are shown in Fig.5. They are the input block, the BM/ACS block, SM/TB block, Control block, and Output and Interface block. The input block collects the soft decision symbol data from the deinterleaver. The BM/ACS block takes the soft decision symbols from input block for decoding. The received symbols are processed by ACS logic, and the result are stored as state metrics in external RAM. Decisions from ACS process are stored into an external path memory. Traceback process through the path memory determines the data bits after tracing backward in time through the decision words. After decoding, the output data is transferred to CPU and ADPCM directly. The control block arranges all of the control signals in the decoding sequence.

Fig.5. SVD Block Diagram (on the last page)

B. The Implementation of SVD

Logic design of the SVD is applied with top-down design methodology using VHDL, and simulated by VHDL logic simulator. The logic timing is verified using output VHDL file by VHDL synthesis and Altera FPGA place and routing. VHDL source code is about 6 thousands lines. The implementation time used is 8 months/2men.

Altera EPF81500AGC280-2 FPGA and commercial RAMs are used for the implementation of SVD. Main Viterbi algorithm except memories is implemented in two Altera FLEX81500 FPGAs. One FPGA utilization is 68% and the other is 13%. That means that about 12 thousands gates are used for implementation of the SVD.

IV. SIMULATION AND PERFORMANCE TEST

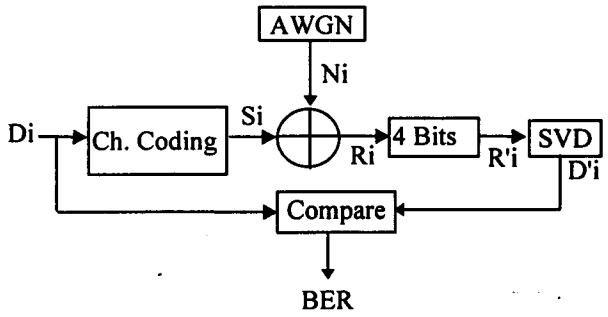
A. Simulation Procedures

Procedures of the SVD simulation are as follows:

1) Logic design using VHDL and logic simulation using VHDL simulator
2) Automatic VHDL synthesis with Altera library and generation of EDIF netlist file for Altera FPGA
3) Altera FPGA place and Route using EDIF netlist and VHDL file generation with timing information
4) VHDL post-simulation for logic timing confirm
5) Downloading Altera FPGA

B. Performance Test Vector Generation

Input vector files are generated by C program for simulation of total SVD and each unit. First the convolutional encoder is made with K=9 and rate 1/2. This encoder makes



Di : Random Number Generator
Si : Convolutional Codes
Ni : AWGN (3 dB Noise)
Ri : Si + Ni
R'i : 16 level quantization(SVD Input)
D'i : Decoded Output

Fig.6. Test Vector Generation

convolutional codes(0 or 1) using random function. The coded data are added by the 3DB AWGN (Additive White Gaussian Noise), and are represented 4 bits data for 16-level soft-decision as shown in Fig.6. These 4 bits data are used for SVD input.

C. SVD Test Environment

We construct the test environment using system emulation tool in order to test the SVD function which is implemented with Altera FLEX81500AGC280-2. This test environment can reduce the test timing, the effort of PCB artwork, and the test cost by FPCB programming for the connection of all components. SVD test environment using Aptix tool is shown in Fig. 6. And the logic analyzer displays the wave form of each unit.

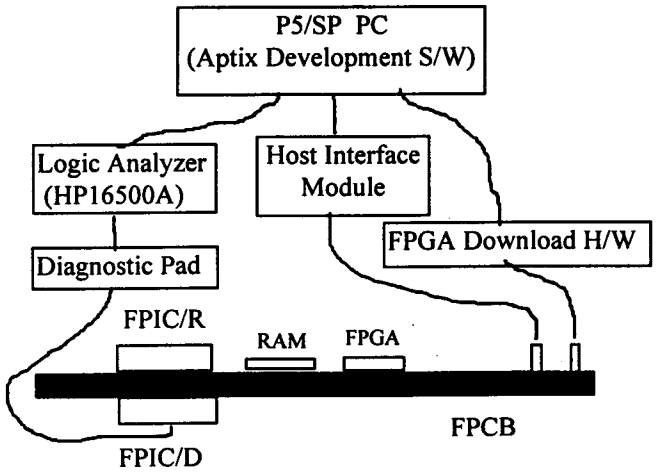


Fig.6. SVD Test Environment

D. SVD Performance Test Procedures

We modify the original circuit to test SVD function, and construct the test environment to verify the coding results of one frame using Aptix FPCB. FPGA, ROM and RAMs are put on FPCB, and they are connected by 2 FPIC programming. First SVD gets the input data from ROM stored original data. After entering input data, SVD executes decoding procedure using input RAM, state RAM, trace-back RAM in FPCB. Self test circuit is designed to display LEDs which are the value of difference between the decoding data and the original input data. Test results show that the designed SVD works well with 3DB Gaussian noises.

V. CONCLUSION

This paper describes the design and the implementation of the serial Viterbi decoder(SVD) using VHDL and FPGAs. The SVD uses the Viterbi algorithm to near optimally decode a synchronized and quantized code symbol. The decoding scheme assumes the transmitted symbols were coded with a K=9, 32Kbps, and rate 1/2 convolutional encoder with generator function $g_0=(753)_8$ and $g_1=(561)_8$ as defined JTC TAG-7 W-CDMA PCS standard. The SVD which employs Viterbi algorithm is selected for error-correction at decoding of convolutional code. The SVD input data come from deinterleaver and decoding data is transferred to CPU and ADPCM directly.

The SVD is designed using VHDL and implemented using FPGAs. Main algorithm except memories is implemented in two Altera FLEX81500 FPGA. One FPGA utilization is 68% and the other is 13%. That means that about 12 thousands gates are used for implementation of SVD. We test the product with real 32MHz clock and 3DB Gaussian noise data. Performance test results show that the error-correction of 3DB noise is exact.

REFERENCES

[1] William C. Y. Lee, "Overview of cellular CDMA", *IEEE Transactions on Vehicular Technology, Vol. 40, No. 2*, May 1991

[2] TIA/EIA Interim Standard(IS-95), *Mobile Station - Base Station Compatibility Standard for Dual-Mode Wideband Spread Spectrum Cellular System*, July, 1993

[3] Proposed Wideband CDMA PCS Standard, Wideband CDMA(W-CDMA) System for Wideband Spread Spectrum Digital 1.80~1.99 GHz PCS Micro- cellular System, October, 1994

[4] Jerrold A. Heller, Irwin Mark Jacobs, "Viterbi decoding for satellite and space communication", *IEEE Transactions on Communication Technology, Vol. COM-19, No. 5*, October 1971

[5] Andrew J. Viterbi, Jim K. Omura, *"Principles of digital communication and coding"*, McGraw-Hill, Inc., 1979

[6] I.M. Onyszchuk, K.-M. Cheung, O. Collins, "Quantization loss in convolutional decoding", *IEEE Transactions on Communications, Vol. 41, No. 2*, February, 1993

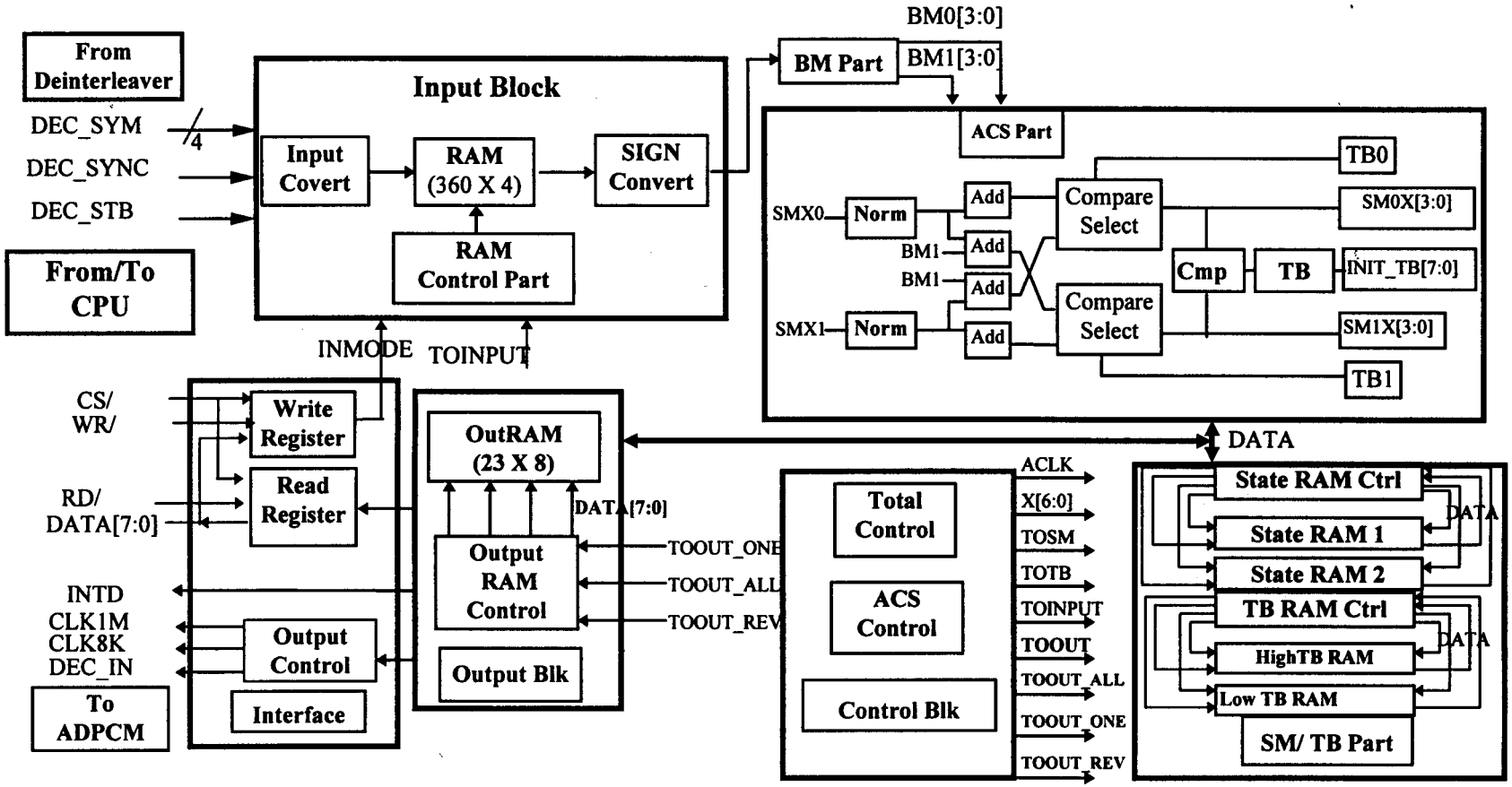


Fig. 5. SVD Block Diagram