Philippine Engineering Journal (2000) XXI (1): 65-72

VITERBI CONVOLUTIONAL ERROR-CORRECTING CODER-DECODER

Azaleah Amina P. Chio* and Louis P. Alarcon
Department of Electrical and Electronics Engineering
University of the Philippines, Diliman

ABSTRACT

In digital communications systems, it is common to use error-correcting codes to maintain reliable data reception. Error correction coding adds redundancy to the input, and at the receiver, this redundancy is used to correct errors. There are two types of error-correcting codes, block codes and convolutional codes.

Convolutional codes are preferred over block codes in error-correction due to better reliability. Although somewhat more complex when implemented, it is easier to decode. Convolutional encoders are implemented using shift-registers and modulo-2 summers. An (n, k, K) convolutional encoder accepts k bits input, and encodes it into n bits using K previous inputs and the present inputs.

However, regardless of whether the coding is block or convolutional, the majority of the effort lies on the decoder, which needs to find the sequence which best corresponds with the noisy received sequence. A well known decoder is the Viterbi algorithm, which produces a maximum-likelihood estimate and is optimal in minimizing the probability of error, given equally likely occurring information sequences.

The Viterbi Algorithm performs a search of all the paths in the trellis diagram, and selects the most likely path traversed, which is then decoded as the original data.

An optimally designed VHDL implementation VLSI architecture of a (3, 2, 3) convolutional encoder and error-correcting Viterbi decoder is aimed in this project.

I. Introduction

In communications, it is critical that information be received as accurately as possible. However, when information is transmitted through a channel, distortion or errors may occur. Several precautions may be implemented, one of which is to use an error-correction algorithm.

Forward Error Correcting (FEC) codes improve the capacity of the channel by making use of redundancy : adding bits to the encoded word for determination of errors. There are two types of error-correcting codes, block codes and convolutional codes. Convolutional coding in telecommunications systems is the most reliable technique in error-correction. It makes use of not only the present inputs but previous inputs as well in determining the output.

*currently involved with Virtual Center for Technology Innovation-Micoelectronics, Advanced Science and Technology Institute.

A convolutional encoder is usually characterized by the following parameters : the coding rate and the constraint length. The coding rate *k/n* represents the number of input bits, *k*, and the number of output bits generated by the encoder, *n*. The constraint length, *K*, corresponds to the number of bits the output is dependent on, previous and present alike. Additional parameters are : the memory order, *m*, which represents the number of shift register stages in path to any output; memory constraint length, *v*, the total number of shift register stages excluding any buffering of inputs and outputs [1].

Convolutional coding is a bit-level encoding technique rather than block-level techniques such as Reed-Solomon coding. Advantages of convolutional codes over block-level codes for telecom/datacom applications are : with soft-decision data, convolutionally encoded system gain degrades gracefully as the error rate increases. Block-level codes correct errors up to a certain point, after which the gain drops off rapidly [2].

The Viterbi Algorithm (VA) is a type of convolutional code. It has the advantage of reduction in computational complexity due to the use of recursion. It also has the property of providing the best interpretation given the entire context. It performs maximum-likelihood detection of data, thus finding the path with the smallest distance when traversing the trellis.

The Viterbi decoder selects a code sequence closest to the received sequence and recovers the original data in a traceback process. The most likely pattern with the fewest errors compared to the received word will be chosen and outputted.

Convolutional encoding and Viterbi decoding has been developed to be used for satellite networks [3], Code Division Multiple Access (CDMA) mobile communication systems [4], modems [5] and for high-quality digital telecommunications, such as high definition television (HDTV) [6]. As it has been proven to be useful for many applications, several developments have surfaced like high speed, low power or pipelined [7] decoders.

In this paper, a development of a previous project Viterbi Convolutional Error-Correcting Coder-Decoder [8], was implemented. Parallel implementation as compared to the previous serial was applied. The result is approximately eight times faster than the previous implementation. However, as always, the trade-off between speed and size crops up resulting to a design that was not tested in the FPGA due to size constraint.

The presentation of this paper is as follows. In Section II, the design methodology is presented and deals with the major blocks used. Section III presents the results acquired in the VHDL simulation of the encoder and the decoder. The conclusion of the project follows in Section IV.

II. Design Method

The convolutional encoder was designed as seen in *figure 1*. It accepts a two-bit input and based on three previous inputs and the present inputs, it produces a three-bit output.

The encoder is composed of two blocks, the main encoder and a counter. The counter counts from 1 to 20 to accept the 32-bit data. At count equal to 1,2, 19 and 20, it is assumed that the user will input "00", which is required to initialize and end the encoder.

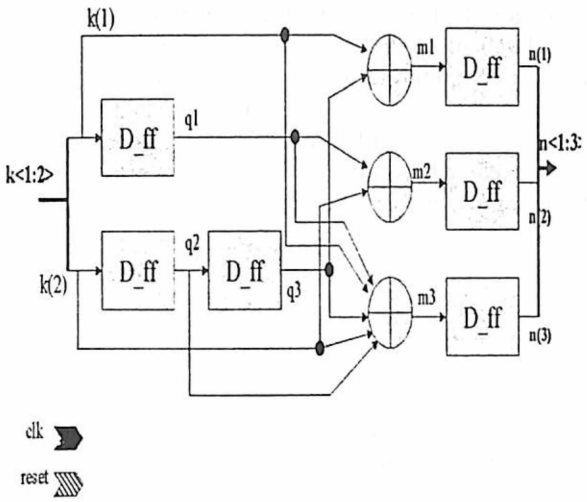

Figure 1: Schematic Diagram of the Convolutional Encoder

The encoder outputs the 3-bit data that will be transmitted and decoded by the Viterbi Decoder. After a delay of one clock cycle, the encoder enables the decoder that the incoming data are valid.

The Viterbi decoder is composed of three major blocks, the Branch Metric Unit (BMU), the Add-Compare-Select Unit (ACS) and the Traceback Memory Unit (TB) [9] *(refer to figure 2)*.

A. Branch Metric Unit

The Branch Metric Unit (BMU) computes the Hamming Distance for hard-decision, or the Euclidean distance for soft-decision decoding, between the received code and the corresponding output in the trellis.

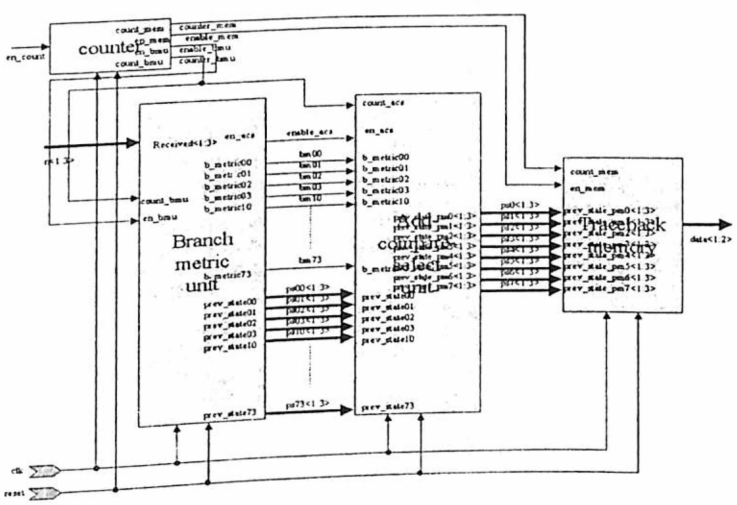

Figure 2 : Viterbi Decoder block diagram

The BMU is composed of 8 smaller blocks, each corresponding to the 8 states in the trellis *(refer to figure 3)*. Each block has a specific function since the traversal in trellis is not symmetric for all states. To compute the branch metrics per state, the matching output from the state transition table is compared with the present state, and the Hamming Distance is the corresponding branch metric. There are 4 branch metrics per state corresponding to the 4 branches in the trellis that may enter each state. Thus, there are also 4 previous states. For all even states, the 4 previous states are the same, and for all odd states, the previous states are also the same. Only 2 bits are needed to represent the previous states.

B. Add-Compare-Select Unit

This unit calculates the path metrics to find the minimum path. It adds the branch metrics to each stored path metric, compares them, two at a time, and selects the path with the lowest metric, the survivor [9].

The ACS unit is composed of two smaller blocks, the instantiated ACS blocks and the path update block *(refer to figure 4)*. The path update block keeps track of the surviving paths, the path metrics. It returns the survivor path metrics to the *inst_acs* unit to be used as the previous path metric for the computation of the survivor path metric in the next clock cycle. Since the previous path metrics will not be needed anymore, the path update just acts like a feedback, without storing the data it receives.

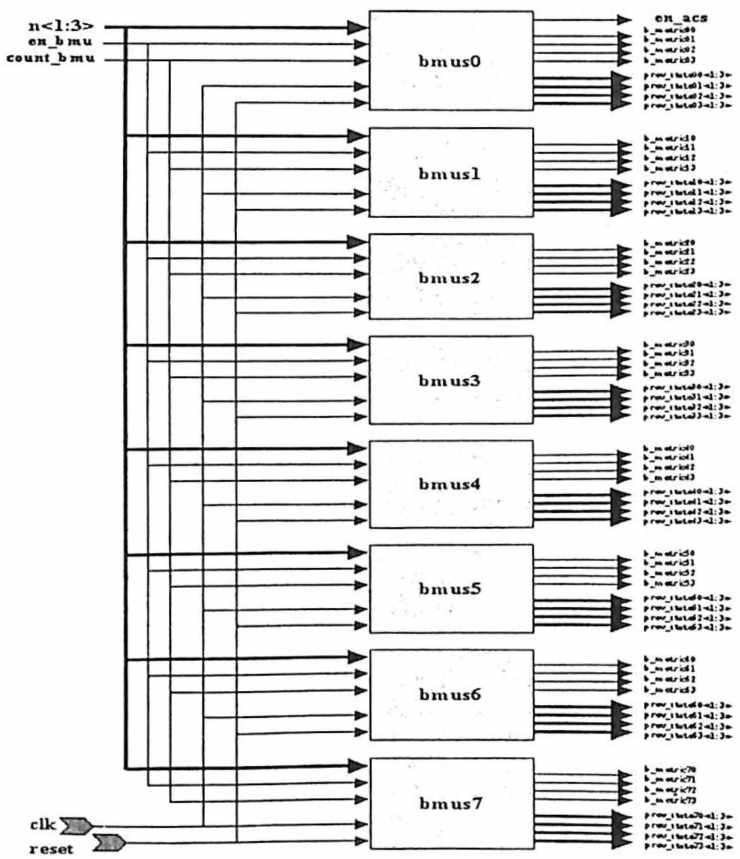

Figure 3: Branch Metric Unit

A. C. Traceback Memory Unit

The memory block *(refer to figure 5)* stores the path traversed in the trellis. The memory stores paths traversed starting when *counter* has count equal to 3, which is when the trellis is normalized. The last two counts force the paths to state 0 as can be seen in the trellis diagram. However, in the traceback, the state is converted into 3 bits to specify whether it came from an odd state or even state. Traceback starts when traversal in the trellis is completed. The traceback starts at state 0, since the path is forced to end at this state. After the whole path has been determined, decoding starts. The decoding process simply obtains the 2 bit original data, depending on the path followed.

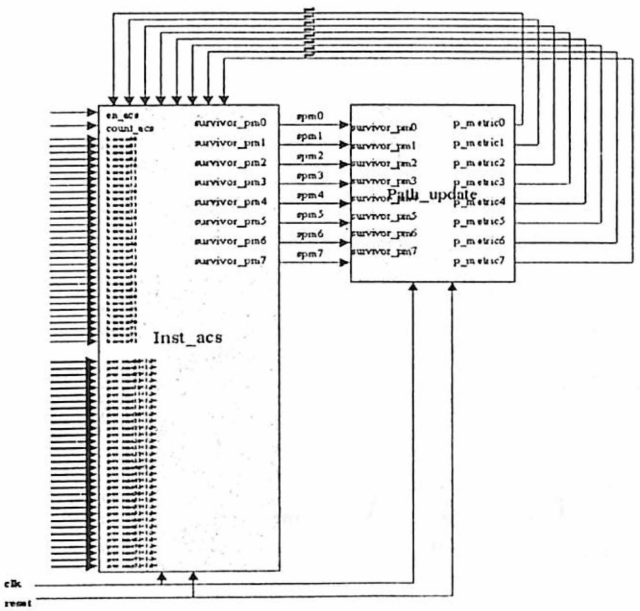

Figure 4 : Add-Compare-Select Unit

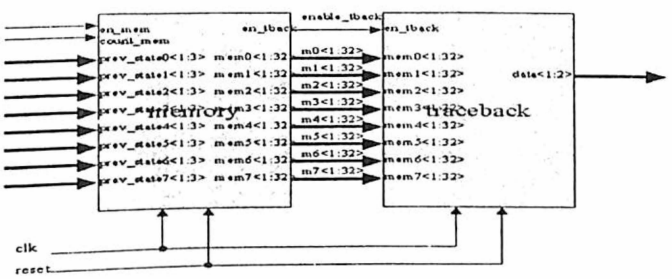

Figure 5 : Traceback Memory Unit

III. Results

The designing process involves encoding *in Cadence's Leapfrog Hardware Description Language (HDL)* and testing, and downloading to *Xilinx's Field Programmable Gate Array (FPGA).*

In encoding in HDL, the encoder and decoder were divided into small blocks, as can be seen in Appendix B : VHDL codes. Each block was designed, and then tested. When the HDL simulations were verified, bigger blocks were formed by instantiation. Thus, the software design was divided into four main blocks: the encoder *(encoder_block)*, the branch metric unit *(bmu)*, the add-compare-select unit *(acs_n)* and the traceback memory unit *(mem_min)*.

The encoder was sub-divided into two blocks : the convolutional encoder and the encoder counter. Eight blocks further described the branch metric unit, a branch metric unit for each state, and then instantiated. The ACS unit was divided into two blocks, the instantiated ACS *(inst_acs)* and the path update unit *(path_update)*. The instantiated ACS was composed of two major blocks : the ACS for even states *(acs_even)* and that for the odd (*acs_odd)*. The traceback memory unit was composed of the memory block *(memory)* and the traceback and decode block *(tback)*. Included in the instantiation of the Viterbi decoder *(integrated)* was the decoder counter *(counter)*.

When the design was tested, it was synthesized using *Galileo Exemplar Logic* which simulates the design in actual circuit of gates and flip-flops. From this step, total number of Configurable Logic Blocks (CLBs) used can be obtained, which determines the size of the FPGA to be used.

After synthesis, the file generated by *Galileo* was imported back to *Cadence*. After which, *Cadence's Design Flows* performs four steps : edit design, netlist, place and route and generate physical.

The *Xilinx* schematic is then acquired, which indicates the pin outs of the FPGA. Then, *XACT Design Manager (XDM)* downloads the design into the FPGA and using the *HP Logic Analysis System*, the design can be tested.

However, only *Xilinx* 4003PG84 and 4010PG191 FPGAs were available, having 100 CLBs and 400 CLBs respectively.

The encoder, when synthesized, used up 20 CLBs. The decoder, when synthesized in *Galileo* used 1024 CLBs. However, when *Galileo Leo Exemplar Logic* was used, 711 CLBs were occupied. But, when the *xnf* file generated by *Galileo Leo* was used in *Cadence's Design Flows*, the *Xilinx* schematic diagram was not created.

The encoder was downloaded to the *Xilinx* 4010PG191 FPGA. The decoder was simulated using *Xilinx* 4025EHQ240-2 FPGA and the Verilog Simulation was verified. However, as this FPGA is not available, the decoder circuit was not downloaded and tested in hardware.

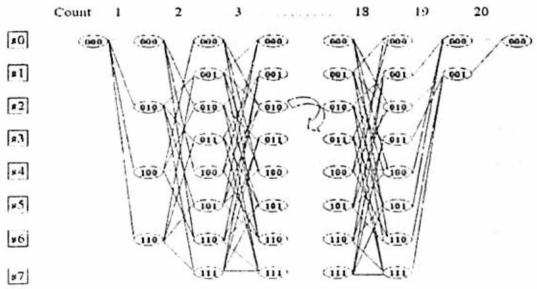

Figure 6 : Trellis Diagram

Noise was simulated by changing up to 5 of the 32 bits in the received word input to the decoder, the Viterbi Decoder was able to correct and attain the same original data was long as the errors were separated by the minimum distance, d_{free}.

Thus, the Viterbi Decoder can correct a certain received code word depending on the errors. If the received code word is closer to another sequence when it goes through the trellis diagram *(refer to figure 1)*, then this would be the output of the decoder. Meaning, there could be a 5-bit error in the received word, which the decoder can correct, and there are some codes, which it cannot correct, depending on the maximum-likelihood to any code sequence in the trellis.

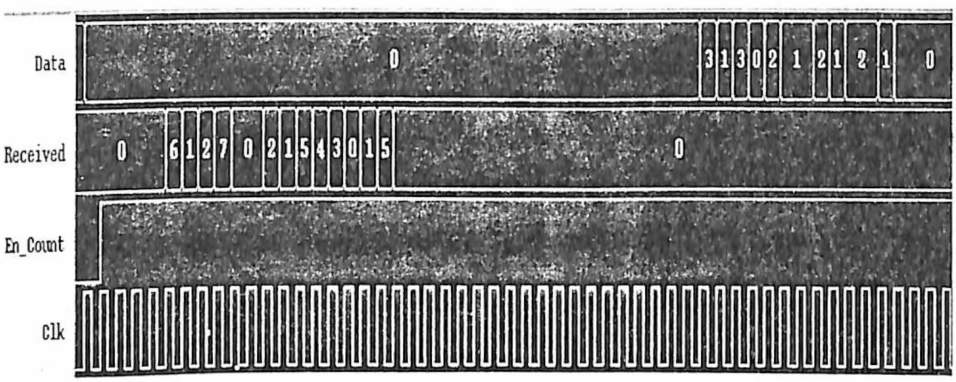

Figure 7: Output of the decoder without any errors simulated

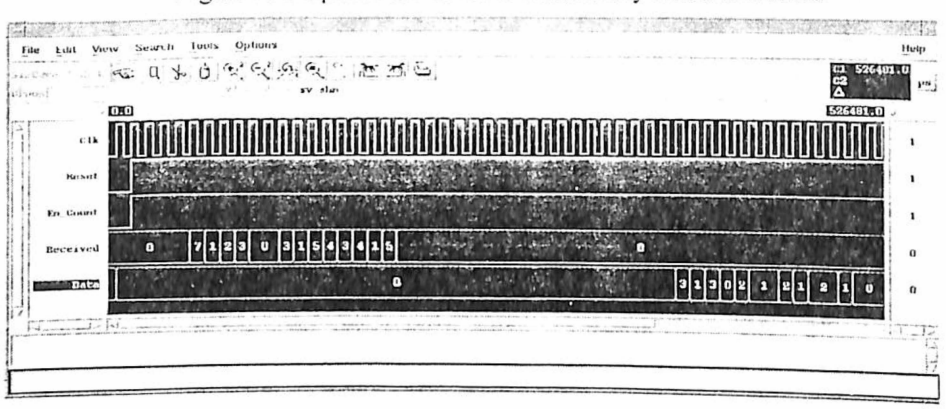

Figure 8: Output of the decoder with 4 errors simulated (corrected)

IV. Conclusion

Reliability of transmitted information is very important in telecommunications. Thus, error-correcting devices are becoming a necessity in the present.

The use of error-correcting codes may increase the operational range of communications systems, reduce the error rates and reduce the transmitted power requirements. The major advantage of trellis codes is that coding gain can be attained without bandwidth expansion or data rate reduction making it attractive for bandwidth limited systems.

Convolutional encoding and Viterbi decoding are commonly used in communications at present. Its applications include high-definition televisions, modems, satellites, digital televisions and digital audio broadcasting.

The Viterbi algorithm proves to be a very useful technique in error-control. It can be implemented serially, which is slower yet occupies less space or parallel processing which would be faster yet bigger, depending on the need of the user.

References

[1] Rhee, Man Young, *Error Correcting Coding Theory*, McGraw-Hill, 1989.

[2] Hendrix, Henry, *Viterbi Decoding Techniques in TM320C54x Family : Application Report*, Texas Instrument

[3] Fleming, Chip, *A Tutorial on Convolutional Coding with Viterbi Decoding*, Spectrum Applications

[4] Kindred, D., Butler, B., Zehavi, E., Wolf, J., *Multirate Serial Viterbi Decoder for Code Division Multiple Access System Applications*, Qualcomm Incorporated, October 1996

[5] Martinez, Kennet, Mack, Gregory, *Viterbi decoder for wireline modems*, Paradyne, March 1985

[6] Nam, Ho Jun, Kwak, Heung Sik, *Viterbi decoder for a high definition television*, January 1995

[7] Yeh; Nan-Hsiung, Olson; Charles R. , *Pipelined Viterbi decoder*, Ampex Corporation, October 1993

[8] Castro, Frederick, *Viterbi Convolutional Error-Correcting Coder-Decoder*, UP EEE, 1998

[9] Smith, Michael J. S., *Application-Specific Integrated Circuits*, Addison Wesley Longman, Inc., 1997