FPGA Implementation of Viterbi Decoder

HEMA.S, SURESH BABU.V, RAMESH P

Dept of ECE, College of Engineering Trivandrum
Kerala University
Trivandrum, Kerala.
INDIA

Abstract: - Convolutional encoding with Viterbi decoding is a powerful method for forward error correction. It has been widely deployed in many wireless communication systems to improve the limited capacity of the communication channels. The Viterbi algorithm, which is the most extensively employed decoding algorithm for convolutional codes. In this paper, we present a field-programmable gate array implementation of Viterbi Decoder with a constraint length of 11 and a code rate of 1/3. It shows that the larger the constraint length used in a convolutional encoding process, the more powerful the code produced.

Key-Words: - Convolutional codes, Viterbi Algorithm, Adaptive Viterbi decoder, Path memory, Register Exchange, Field-Programmable Gate Array (FPGA) implementation.

1 Introduction

With the growing use of digital communication, there has been an increased interest in high-speed Viterbi decoder design within a single chip. Advanced field programmable gate array (FPGA) technologies and well-developed electronic design automatic (EDA) tools have made it possible to realize a Viterbi decoder with the throughput at the order of Giga-bit per second, without using off-chip processor(s) or memory. The motivation of this thesis is to use VHDL, Synopsys synthesis and simulation tools to realize a Viterbi decoder having constraint length 11 targeting Xilinx FPGA technology.[5]

The Viterbi algorithm develops as an asymptotically optimal decoding algorithm for convolutional codes. It is nowadays commonly using for decoding block codes. Viterbi Decoding has the advantage that it has a fixed decoding time. It is well suited to hardware decoder implementation.Viterbi decoding of convolutional codes found to be efficient and robust. Although the viterbi algorithm is, simple it requires $O(2^{n-k})$ words of memory, where n is the length of the code words and k is the message length, so that $n-k$ is the number of appended parity bits. In practical situations, it is desirable to select codes with the highest minimum Hamming distance that decodes within a specified time and an increased minimum Hamming distance $d_{\min}$ implies an increased number of parity bits. Our viterbi decoder necessarily distributes the memory required evenly among processing elements [1].

2. Convolutional Code

2.1 Convolutional Encoding

Convolutional code is a type of error-correcting code in which each *(n≥m) m*-bit information symbol (each *m*-bit string) to be encoded is transformed into an *n*-bit symbol, where *m/n* is the code rate *(n≥m)* and the

Hema S. is M.Tech scholar with the Department of ECE, College of Engineering Trivandrum.E-mail: hemarajen@gmail.com
Suresh Babu V. is with the Department of ECE, College of Engineering Trivandrum.E-mail:vsb_sreeragam@yahoo.co.in
Ramesh P is with the Dept of ECE,,Munnar Engineering.
Email : ramp1718009@rediffmail.com

transformation is a function of the last k information symbols, where *K* is the constraint length of the code. To convolutionally encode data, start with *k* memory registers, each holding 1 input bit. Unless otherwise specified, all memory registers start with a value of 0.The encoder has *n* modulo-2 adders, and *n* generator polynomials—one for each adder (see figure1).An input bit m_1 is fed into the leftmost register. Using the generator polynomials and the existing values in the remaining registers, the encoder outputs *n* bits [1].

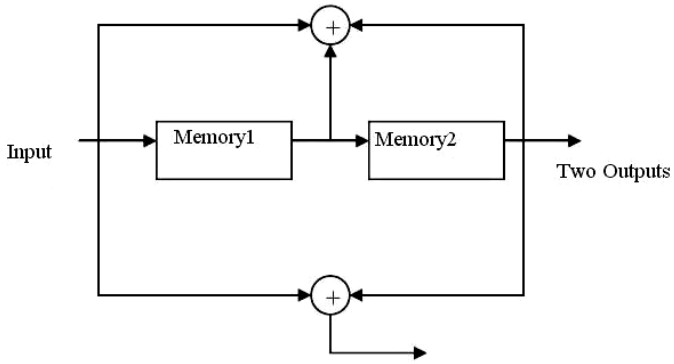

Figure 1: The rate ½ Convolutional encoder

2.2 Viterbi Algorithm

A. J. Viterbi proposed an algorithm as an 'asymptotically optimum' approach to the decoding of convolutional codes in memory-less noise. The Viterbi algorithm (VA) is knows as a maximum likelihood (ML)-decoding algorithm for convolutional codes.

Maximum likelihood decoding means finding the code branch in the code trellis that was most likely to be transmitted. Therefore, maximum likelihood decoding is based on calculating the hamming distances for each branch forming encode word. The most likely path through the trellis will maximize this metric.[7]

Viterbi algorithm performs ML decoding by reducing its complexity. It eliminates least likely trellis path at each transmission stage and reduce decoding complexity with early rejection of unlike pathes.Viterbi algorithm gets its efficiency via concentrating on survival paths of the trellis. The Viterbi algorithm is an optimum algorithm for estimating the state sequence of a finite state process, given a set of noisy observations.[2]

The implementation of the VA consists of three parts: branch metric computation, path metric updating, and survivor sequence generation. The path metric computation unit computes a number of recursive equations. In a Viterbi decoder (VD) for an N-state convolutional code, N recursive equations are computed at each time step (***N*** = 2k-1**, *k*=** constraint length). [12] Existing high-speed architectures use one processor per recursion equation. The main drawback of these Viterbi Decoders is that they are very expensive in terms of chip area. In current implementations, at least a single chip is dedicated to the hardware realization of the Viterbi decoding algorithm The novel scheduling scheme allows cutting back chip area dramatically with almost no loss in computation speed.[15]

3. Viterbi Decoder

A viterbi decoder uses the Viterbi algorithm for decoding a bitstream that has been encoded using Forward error correction based on a code. There are other algorithms for decoding a convolutionally encoded stream (for example, the Fano algorithm). The Viterbi algorithm is the most resource consuming, but it does the maximum likelihood decoding. Figure 2 shows the block diagram of viterbi decoder

It consists of the following modules: [7]

Branch Metrics, ACS, register exchange, maximum path metric selection, and output register selection.

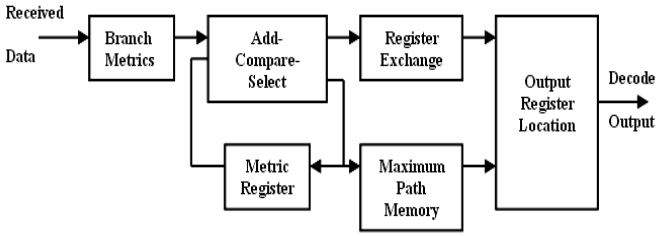

Figure:2 Block Diagram of Viterbi Decoder

3.1. Branch Metrics

The branch metric computation block compares the received code symbol with the expected code symbol and counts the number of differing bits .Figure 3 shows the block diagram of branch metrics[7]

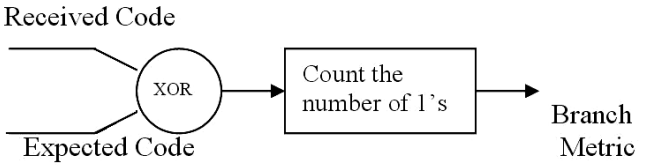

Figure: 3 Branch Metrics of viterbi decoder

3.2. Add-Compare-Select (ACS)

The two adders compute the partial path metric of each branch, the comparator compares the two partial metrics, and the selector selects an appropriate branch. The new partial path metric updates the state metric of state, and the survivor path-recording block records the survivor path.[7]

Figure 4 shows the block diagram of ACS block

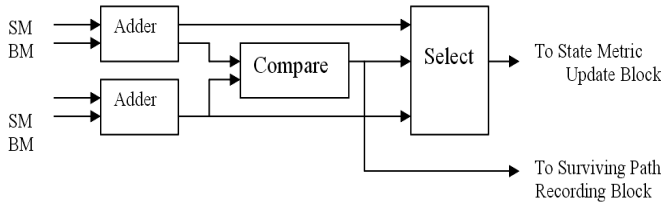

Figure:4 ACS block

3.3. Path Metric Calculation and Storage

The ACS circuit consisting of adders, a comparator, a selector, and several registers calculates the path metric of each convolutional code state. 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 .The path metric calculations just assigned the measurement functions to each state, but the actual Viterbi decisions on encoder states is based on a traceback operation to find the path of states. The important characteristic is that if every state from a current time is follow backwards through its maximum likelihood path, all of the paths converge at a point somewhere previous in time. This is how traceback decisively determines the state of the encoder at a given time, by showing that there is no better choice for an encoder state given the global maximum likelihood path.[6]

3.4 .Register-exchange and Traceback

The register-exchange approach assigns a register to each state. The register records the decoded output sequence along the path starting from the initial state to the final state, which is same as the initial state. This approach eliminates the need to traceback, since the register of the final state contains the decoded output sequence. Hence, the approach may offer a high-speed operation, but it is not power efficient due to the need to copy all the registers in a stage to the next stage.[10]

The other approach called traceback records the survivor branch of each state. It is possible to traceback the survivor path provided the survivor branch of each state is known. While following the survivor path, the decoded output bit is '0' ('1') whenever it encounters an even (odd) state. A flip-flop is assign to each state to store the survivor branch and the flip-flop records '1' ('0') if the survivor branch is the upper (lower) path. [6]

3.4.1) Traceback read (tb)

There are three types of operations performed inside a TB decoder:

This is one of the two read operations and consists of reading a bit and interpreting this bit in conjunction with the present state number as a pointer that indicates the previous state number (i.e. state number of the predecessor) .[4]

3.4.2) 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 decode 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.

3.4.3) Writing new data (wr)

The decisions made by the ACS write 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 .

3.4.5) Selective Update and Shift Update

It is possible to form registers by collecting the flip-flops in the vertical direction or in the horizontal direction. When a register is formed in vertical direction, it is referred to as "selective update". When a register is formed in horizontal direction, it is referred to as "shift update". In selective update, the survivor path information is filling from the left register to the right register as the time progresses. In contrast, survivor path information is applied to the least significant bits of all the registers in "shift update". Then all the registers perform a shift left operation. Hence, each register in the shift update method fills in survivor path information from the least significant bit toward the most significant bit.[9]

3.4.6) Survivor Path Memory

To implement the survivor path memory architecture, three types of path memory management schemes are commonly used: register-exchange (RE), trace-back (TB), and RE-TB-combined. The RE approach is suitable for fast decoders, but occupies large silicon real estate and consumes lots of power. On the other hand,

the TB path memory usually consumes less power, but is slower than its RE counterpart, or requires a clock rate higher than the decoding throughput. The RE-TB-combined approach, is a good alternative to the RE approach for high-speed applications. [7]

4. The Field Programmable Gate Array (FPGA)

A Field Programmable Gate Array (FPGA) is a semiconductor device containing programmable logic components and programmable interconnects. The programmable logic components can be programmed to duplicate the functionality of basic logic gates such as AND, OR, XOR, NOT or more complex combinational functions such as decoders or simple math functions. In most FPGAs, these programmable logic components (or logic blocks, in FPGA parlance) also include memory elements, which may be simple flip-flops or more complete blocks of memories. Each process is assigned to a different block of the FPGA and operates independently. A shared register between the processors implements the arcs, which represent the transmission of the weights and paths for each state to another processor. All systems were implementing in behavioral VHDL.A synthesis tool is used to construct the RTL level VHDL for the decoders. This synthesized unit is then simulated using a commercial simulation tool for VHDL.In VHDL the initial conditions such as the location of the weights and paths needed to update a state are readily coded and so don't need to be calculated for each cycle of the decoding process. The received message is fan out into all the processors a bit at a time and this is the logical clock for the machine. On receiving each input bit, each processor reads the shared registers, updates the weights and paths and writes the results to the shared registers. [11]

FPGAs originally began as competitors to CPLDs and competed in a similar space, that of glue logic for PCBs. As their size, capabilities and speed increase, they began to take over larger and larger functions to the state where they are now market as competitors for full systems on chips. They now find applications in any area or algorithm that can make use of the massive parallelism offered by their architecture.[2]

The typical basic architecture consists of an array of configurable logic blocks (CLBs) and routing channels. Multiple I/O pads may fit into the height of one row or the width of one column. Generally, all the routing channels have the same width (number of wires).

An application circuit must be map into an FPGA with adequate resources. The typical basic architecture consists of an array of configurable logic blocks (CLBs) and routing channels. Multiple I/O pads may fit into the height of one row or the width of one column. Generally, all the routing channels have the same width (number of wires). [13]

FPGAs are an extremely valuable tool in learning VLSI design. While the traditional techniques of full- and semi-custom design certainly have their places for analog, high performance or complex applications, the prospect of putting "their" chip to the decisive test of a real hardware environment motivates students tremendously. [10]

To define the behavior of the FPGA the user provides a hardware description language (HDL) or a schematic design. Common HDLs are VHDL and Verilog. Then, using an electronic design automation tool, a technology-mapped netlist generates. The netlist can then be fitted to the actual FPGA architecture using a process called place-and-route, usually performed by the FPGA Company's proprietary place-and-route software. The user will validate the map, place and route results via timing analysis, simulation, and other verification methodologies. Once the design and validation process is complete, the binary file generated (also using the FPGA company's proprietary software) is used to (re)configure the FPGA device. Figure: 5 show the design flow of FPGA. [2]

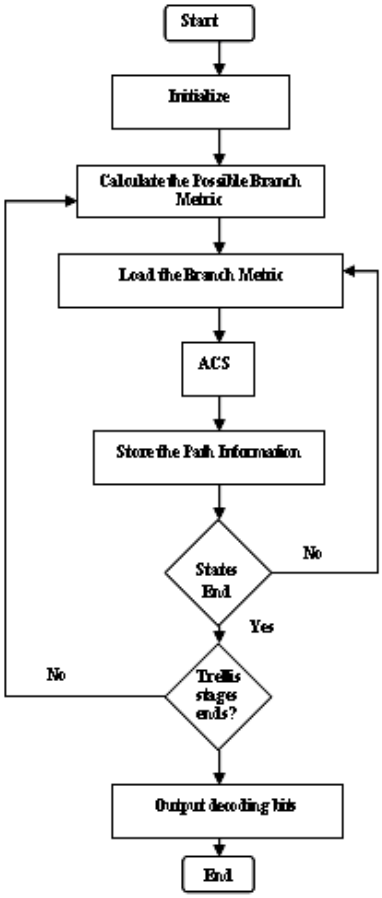

Figure: 5 Design flow of FPGA

To simplify the design of complex systems in FPGAs, there exist libraries of predefined complex functions and circuits that have been tested and optimized to speed up the design process. These predefined circuits are called IP cores, and are available from FPGA vendors and third-party IP suppliers (rarely free and typically released under proprietary licenses).

In a typical design flow, an FPGA application developer will simulate the design at multiple stages throughout the design process. Initially the RTL description in VHDL or Verilog is simulated by creating test benches to stimulate the system and observe results. Then, after the synthesis engine has mapped the design to a netlist, the netlist is translated to a gate level description where simulation is repeated to confirm the synthesis proceeded without errors. Finally, the design is laid out in the FPGA at which point propagation delays can be added and the simulation run again with these values back annotated onto the netlist. The typical FPGA logic block consists of a 4-input lookup table (LUT), and a flip-flop, as shown in Figure 6.[3]

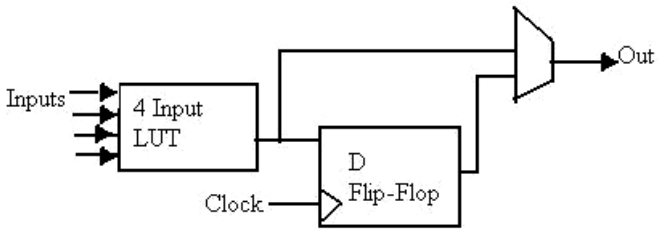

Figure:6 Logic Block of FPGA

5. Conclusion

In this paper, a Viterbi algorithm based on the strongly connected trellis decoding of binary convolutional codes has been presented. The use of error-correcting codes has proven to be an effective way to overcome data corruption in digital communication channels. The adaptive Viterbi decoders are modeled using VHDL, and post synthesized by Xilinx Design Manager FPGA logic. The design simulations have been done based on both the VHDL codes at RTL and the VHDL codes generated by Xilinx design manager after post synthesis.

We can implement a higher performance Viterbi decoder with such as pipelining or interleaving. So in the future, with Pipeline or interleave the ACS and the trace-back and output decode block, we can make it better.

References

[1].”FPGA Design and Implementation of a Low-Power Systolic Array-Based Adaptive Viterbi Decoder”,IEEE Transactions on circuits and systemsi: regular papers, vol. 52, no. 2, February 2005.

[2]. ”A Parallel Viterbi Decoder for Block Cyclic and Convolution Codes”, Department of Electronics and Computer Science, University of Southampton. April 2006

[3]. ”A Dynamically Reconfigurable Adaptive Viterbi Decoder”, International Symposium on FPGA 2002.

[4]. ”A Reconfigurable, Power-Efficient Adaptive Viterbi Decoder”, This work was sponsored by National Science Foundation grants.

[5]. X.Wang and S.B.Wicker,"An Artificial Neural Net Viterbi Decoder," IEEE Trans.Communications,vol.44,no.2,pp.165-171,February 1996.

[6]. D. Garrett and M. Stan, " Low power architecture of the soft-output Viterbi algorithm,"Electronic-Letters, Proceeding 98 for ISLEPD 98, p 262-267, 1998.

[7]. "RTL implementation of Viterbi decoder", Dept. of Computer Engineering at Linkpings universitet.June 2006

[8]. Ranjan Bose ,"Information theory coding and Cryptography", McGraw-Hill.

[9]. G. Marcone and E. Zincolini. "An efficient neural decoder for convolutional codes." European Trans.Telecomm.,6(4):439-445, July-August 1995.

[10]. J.G. Proakis. "Digital Communications." McGraw-Hill,second edition, 1989.

[11]. A. J. Viterbi,"Convolutional codes and their performance in communication systems," IEEE Trans. Commun., vol. COM-19, pp. 751 -772,Oct., 1971.

[12]. LM Dong, SONG Wentao, LIU Xingzhao, LUO Hanwen, XU Youyun, ZHANG Wenjun "Neural Networks Based Parallel Viterbi Decoder by Hybrid Design" Department of Electronic Engineering, Shanghai Jiaotong University.

[13]. C. B. Shung, H-D. Lin, R. Cypher, P. H. Siege1 and H. K. Thapar, "Area-efficient architecture for the Viterbi algorithm Part 11: Applications," IEEE Trans. Commun.,vol. 41, pp. 802-807, May 1993.

[14]. Shuji Kubota, Shuzo Kato, Member, IEEE, and Tsunehachi Ishitani, "Novel Viterbi Decoder VLSI Implementation and its Performance."
IEEE Transactions on Communications, VOL. 41, NO. 8, AUGUST 1993.

[15]. Stevan M. Berber, Paul J. Secker, Zoran A. Salcic, "Theory and application of neural networks for 1/n rate convolutional decoders",School of Engineering, The Department of Electrical and Computer Engineering, The University of Auckland,New Zealand.14 July 2005