Memory Efficientlow Dendity Parity Check Codes On FPGA

Shrugal Varde [1], Dr. NishaSarwade [2]
Electrical Engineering department,VJTI,Mumbai,India

Electrical Engineering department,VJTI,Mumbai,India

Abstract

Low-density Parity Check(LDPC) codes have been in focus of intense research in Error-correction Codingin recent years. High throughput decoder design for them has been a big challenge for these codes. In thispaper, we introduce decoder design based on projective geometry (PG) structure of LDPC code sum product algorithm. The corresponding fully-parallel VLSI architecture was implemented on Xilinx LX110T FPGA. MATLAB was used to simulate the code.

Keywords *: LPDC , MATLAB, Projective geometry, Virtex 5,Xilinx.*

1. Introduction

LDPC codes are an emerging class of codes which exhibit superior bit error rate(BER) performance. Relative ease of decoder design, coupled with better performance, has made LDPC codes useful in recent digital transmission and storage systems. All LDPC decoding algorithms need large number of parallel working memories. Hence designing for memory efficiencyis one of the significant task in its decoder design. In this work, we introduce an LDPC decoder design that uses on chip memory for storing data. In general, different code structures result in different architectures, and hence different memory management schemes. Our choice of structures is derived out of geometry of projective planes. We report prototype implementation results on FPGA, to demonstrate simplicity of design, and high efficiency of the hardware, for PG based LDPC codes, apart from throughput improvement. The design has been implemented on Xilinx Virtex 5 FPGA board.

2. Low density parity check codes

LDPC codes are generally decoded using probabilisticsoft decision decoding process. The knowledge of channel noise statistics is used to generate probabilistic information for received bits, and given to the decoder. The reliability of this bit information is then successively improved over iterations, and hard decisions on bit values made. Such decoders make use of graphs known as Tanner graphs to represent codes, passing probabilistic messages along the graph's edges iteratively. The matrix of this graph is called parity-check matrix H. A Tanner graph has two sets of nodes: n bit nodes, and m parity-check nodes, for a m*n sized parity-check matrix. Each parity-check node is connected by an edgeto bit nodes corresponding to the code bits includedin that parity-check equation. The sequence of stepsinvolved in iterative decoding using log-sum-product

Algorithm is as follows

1. The initial message is first sent by bit nodes to check nodes. This message is based on the calculated log-likelihood ratio(LLR) of received signal.
2. Next, check nodes calculate and send updated LLRs to the bit nodes using the received messages. The formulae is as given

$$E_{j,i} = \log\left(\frac{1+\prod_{i'\in B_j, i'\neq i}\tanh(M_{j,i'}/2)}{1-\prod_{i'\in B_j,\ i'\neq i}\tanh(M_{j,i'}/2)}\right)$$

Ej,i – data sent from jth check node to ith bit node.

Mj,i - data sent from ith bit node to jth check node.

3. Bit node add all the data received form the check nodes connected to that bit node

$$L_i = \mathrm{LLR}(P_i^{\mathrm{int}}) = r_i + \sum_{j\in A_i} E_{j,i}$$

4. Hard decision is done based on sign of Li. If in some iteration syndrome vector is zero then code word Li belongs to code block, else process is repeated.
5. Updates messages are sent back to check node. Formula for updating the check data is as given

$$M_{j,i} = \sum_{j'\in A_i,\ j'\neq j} E_{j',i} + r_i$$

ri – intrinsic data or apriori data received by bit node at the start of decoding process.

3. Design

In this design fixed point data format is used to represent binary data. The advantage of using the fixed point data representation is that arithmetic operation is much simpler compared to floating point arithmetic operation. In the design 16 bits are used to represent the data .MSB indicated the sign bit. Integer data is represent by 5 bits and remaining 10 bits are used to represent fractional data . The data format is as shown in figure 1

SIGN (1 BIT)	INTEGER DATA(5 BITS)	FRACTION DATA (10 BITS)

Figure 1: Data representation format

The choice of message precision is dictated by the biterror rate and the resources of storage and computation available on the FPGA.

4. Bit node design

The bit nodes are relatively simpler to implement as they do only addition of several check node messages. As the number of check node messages is relatively more in our case, we employ a partly serial adder for our design. The check node messages are first stored in a register bank. Addition process of 9 check nodes along with apriori data is completed in 4 clock .Architecture is as shown in figure 2Then,each check node message is subtracted from this total sum to get the individual bit-to-check messages.

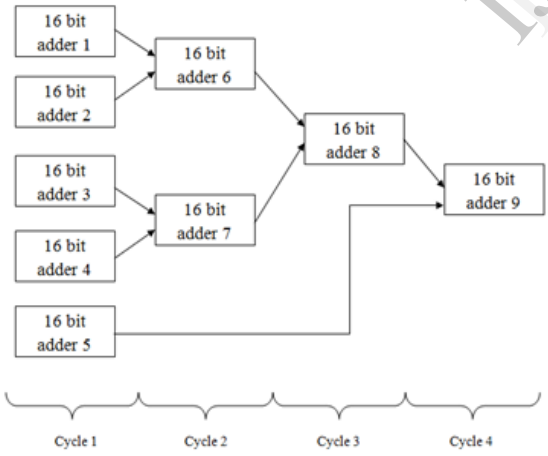

Figure 2 : Bit addition architecture

5. Check node design

The check nodes are comparatively much more complex.Check node computation includes calucu-lating hyperbolic tan , multiplication and natural logarithm of nine set of 16 bit data. These arithmetic operations are calculated by simple shift and add algorithm called CORDIC algorithm first proposed by Volder in 1961.Hyperbolic tan structure is as shown in figure 3

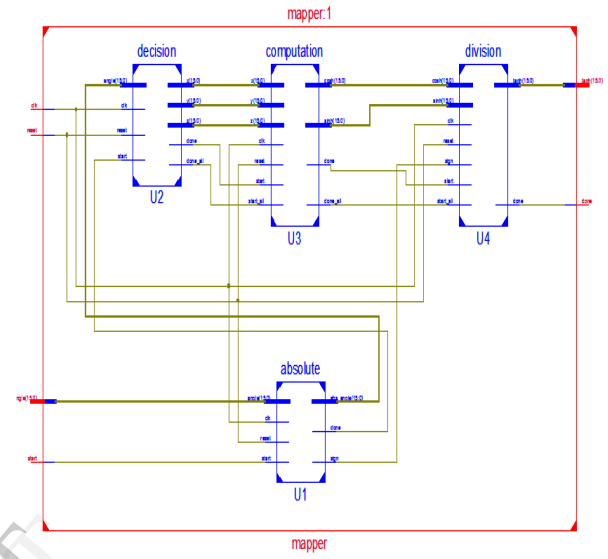

Figure 3 : Hyperbolic tan structure

The absolute block separates the sign and the magnitude part of the angle entered. The magnitude part of the values is given to the decision block

The decision block checks the range of the angle . If the value is out of range(greater than 3.5) then decision block generates a done_all control signal. If not, then decision block initializes the value of X, Y and Z and generates a control signal done. These values are given to the computation block.

The computation block implements the hyperbolic CORDIC equation given by walther. This block at the end of 13 iterations generates the values of hyperbolic sin and hyperbolic cos. These values are given to division block. To increase the speed barrel shifters are used.[5]

The division process used in this code is also implemented using CORDIC algorithm. It computes the value of tanh (tanh = sinh/cosh). After calculating the value of hyperbolic tan sign value is taken into consideration and final result is generated

Multiplier block structure using CORDIC algorithm is as shown in figure 4

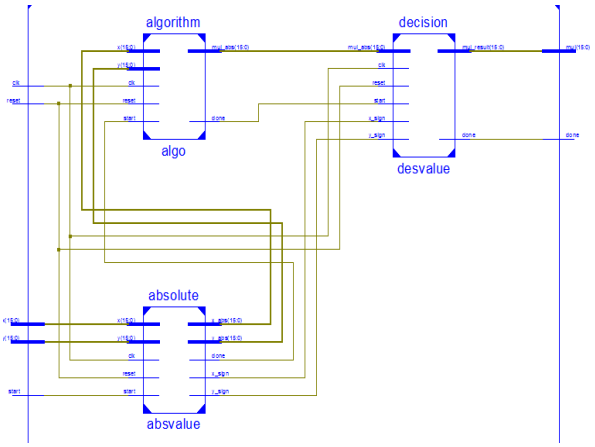

Figure 4 : Multiplier block structure

The absolute block separates the sign and the magnitude part of the angle entered. The magnitude part of the values is given to the decision block

The algorithm block performs the most important task of multiplying two 16 bit binary data using CORDIC algorithm .Matlab simulation results have shown that the multiplication block using CORDIC algorithm provides accurate results with error less than 0.01% when the number of iteration used to calculate the multiplication result is more than 12. After the computation process the decision block decides the sign of the result. For positive number the sign bit is 0 and for negative sign bit is 1.

Natural logarithm block structure is as shown in fig. 5

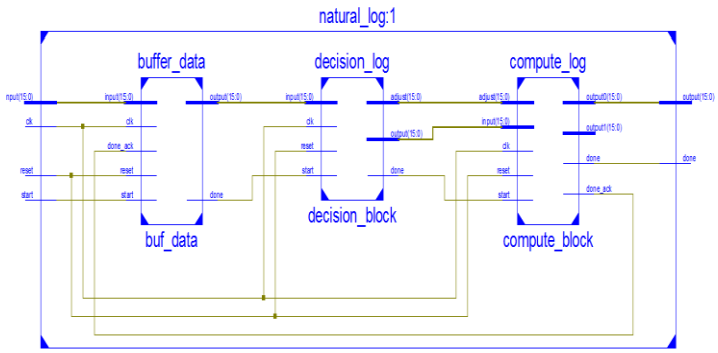

Figure 5 : Natural logarithm block structure

Buffer data receive the data from multiplication block. The main task of the buffer block is to generate set of data which is (1 + input) and (1 – input). Depending on the condition (control signals) buffer data either gives (1+ input) as output or (1- input). The decision block decides the input and adjustment value depending on the input data of decision block. The compute block performs the actual operation of calculating the natural logarithm of the input data. At the end compute block performs the task of log(1+input) + log(1-input).

Each check node is connected to nine bit nodes . Hence check node at start should calculate hyperbolic tan of nine input data. After the calculation of hyperbolic tan eight 16 data should be multiplied . If multiplication is done serially then 192 clock cycles would be required to generate the result which will act as input to natural logarithm block. To increase the speed of the system semi parallel structure is adopted for multiplication for 8 16 bit data. Structure is a shown in figure 6

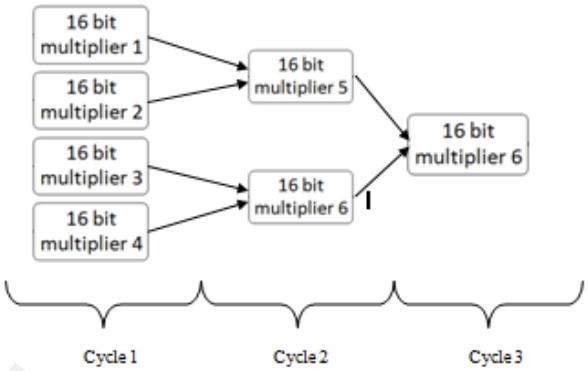

Figure 6 : semi parallel multiplication structure

6. Control logic

Due to the simplified logic of the decoder, the control unit design becomes extremely simple. This is implemented in terms of a simple FSM as shown in Figure 7. The strobe input is given a strobe after a data block has been input. The output signal goes to select the intrinsic information. Whenever all the parity check equations have been satisfied, they are output one and the unit becomes ready to accept new input blocks.

Also, a one cycle wide pulse is generated on the end output to notify the external device of the completion.

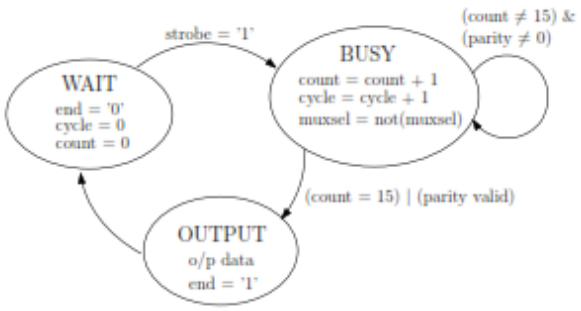

Figure 7 : Control unit schematic

7. Results and simulations

The vhdlcodes of sum product algorithm was implemented on VIRTEX 5 FPGA board. The device used is xc5vlx110t-1ff1136. For synthesis of vhdl code we used XILINX ISE DESIGN SUITE 14.1 . The code were verified on the board with the help of Xilinx IMPACT tool . MATLAB 2012a was used to generate the simulation graphs.

The synthesis report is as shown

Number of Slice registers	16909 of 69120 (24%)
Number of slice LUT's	69100 of 69120(86%)
Number of fully used LUT's	11200 of 1208 (13%)
Number of bonder IOB's	147of 640 (22 %)
Number of BUFG/BUFGCTRL's	2 of 32 (6%)

Maximum clock frequency at which system can work is 120 MHz. As block RAM is used in the system the number of interconnects have reduces.

For simulation purpose a matlab code was written that generates 73 random values in the range of -5 to 5. These values were then converted in 16 bit binary format used in the design. The values are stored in on chip RAM. The decoder was simulated using Xilinx ISIM simulator. After simulation it was observed that the system takes 2 to 3 iteration to generates the final result.The utilization of the resources is almost 86 %.As block RAM was used to transfer data from bit node to check node and vice versa the interconnection have reduced thus reducing the complexity of the system.Since the cross over probability of the channel is almost 5 % the number of bits in which error is allowed is 73*0.05 =3.6 ,approximately 4. Decoder was able to decode upto 4 bit errors with number of iterations required for decoding to be less than 3.

8. References

[1] R.G.Gallager,_Low Density Parity Check codes M.I.T.Press,1963

[2] D.J.C.MacKay and R.M.Neal,_Near Shannon limit performance of low density parity checkcodes, ElectronLett.,vol.32,no.18,pp.1645- 1646,1996.

[3] Y. Kou, S Lin, and M. P.C. Fossorier, "Low density parity check codes based on _nite geometries: a rediscovery", Proc. IEEE ISIT, Sorrento, Italy, p. 200, Jun 2000.

[4] Sarah J. Johnson, "Introducing Low-Density Parity-Check Codes".

[5] N.Karmarkar,_A New Parallel Architecture for Sparse Matrix Computation Based on Fi niteProjective Geometries_,AT&T Bell Labs,1994.

[6] William E Ryan, University of Arizona, "INTRODUCTION TO LDPC CODES",2001

[7] Hrishikesh Sharma, Subhasis Das," HIGH THROUGHPUT MEMORY-EFFICIENT VLSI DESIGNS FOR STRUCTURED LDPC DECODING",IIT Bombay.,May 2010.

[8] Subhasis Das, Hrishikesh Sharma, Sachin Patkar," A Min-Sum based 800 Mbps LDPC decoder on FPGA",IIT Bombay,March 2010