

Hardware Implementation of Min-Sum Decoder for Low Density Parity Check Codes

Mamta Prakash¹, Girraj Prasad Rathore²

¹M.E. Student, Department of Electronics and Communication, Technocrat Institute of technology, Bhopal

²Assistant Professor, Department of Electronics and Communication, Technocrat Institute of technology, Bhopal

Abstract: *Low Density Parity Check (LDPC) technique is highly used in the communication protocol to effectively transfer data from transmitter end to receiver end. In this paper a highly efficient decoding technique viz. min-sum has used to transfer data. The data from the communication channel is used for the decoding process. Min-Sum algorithm decoder is implemented and simulation is done using Model sim. The hardware synthesis results are shown using Xilinx ISE 14.1 and Spartan 6 FPGA board.*

Keywords: LDPC, Decoder, Min-sum Algorithm, FPGA

1. Introduction

LDPC was introduced by Robert Gallager at MIT in 1960 in his PhD thesis[1]. Low density parity check codes are linear block codes using generator matrix G in an encoder and parity check matrix H in a decoder[1]. The parity check matrix has M rows and N columns, where M represents check nodes and N represents variable nodes. The Tanner graph of LDPC codes represents the check nodes and variable nodes and. Information bits depends on check nodes and code word bits are depends on variable nodes. Tanner Graph is the bipartite graph introduced to graphically represent these codes. They also helps to describe decoding algorithms. Tanner graphs are separated into two distinctive sets and edges are only connecting nodes of two different types mainly known as check nodes and variable nodes. The iterative decoding of code is the true optimum decoding if Tanner graph contains no cycles. Therefore we want LDPC codes with few cycles.

Low density parity-check (LDPC) codes are a class of linear block codes. The name comes from the characteristic of their parity-check matrix which contains only a few 1's in comparison to the amount of 0's. The advantage is that they provide a performance which is very close to the capacity for a lot of different Channels and linear time complex algorithms for decoding [2].

Current LDPC decoding methods are derived from the Sum-Product Algorithm (SPA) [3]. However, the SPA requires considerable multipliers during decoding, thereby elevating the difficulty of hardware implementation. Subsequent scholars have employed concepts in the logarithm domain to modify the multiplication equation of the SPA into a logarithmic equation, which is also known as the logarithm-domain sum-product algorithm [4]. Subsequently, the logarithm-domain algorithm (Log-SPA) is simplified to derive the MSA [5]. Compared with the SPA, which requires excessive multipliers during decoding, and the Log-SPA that requires complex logarithmic computation during decoding, the MSA only requires a comparator to complete decoding, thereby significantly reducing the difficulty of hardware implementation. The MSA decoder developed in this study used two types of iteration (i.e., a single iteration

and 10 iterations) to verify and compare the decoding performance and coding gains.[6],[7].

The MSA performs simple arithmetic and logical operations that makes suitable for hardware implementation. But the performance of the algorithm is significantly impacted by the quantization of soft input messages used [8]. Reducing the quantization of the message is invariably important to reduce the implementation complexity and hardware resources of the decoder. But this advantage comes with degradation in decoding performance. Performance issues and hardware implementation of such low complexity algorithms, especially the 2-bit MSA has limited information in the literature.

2. Decoding of LDPC Codes

There are different iterative decoding algorithms having two derivations. They are mainly classified as in hard decision decoding and soft decision decoding respectively. Bit flipping is hard decision decoding algorithm and Sum Product is soft decision decoding algorithm.

2.1 Hard Decision Decoding (Bit Flipping):

A hard decision is made on each received bit from the channel and then those bits are transferred to Tanner graph structure. In this algorithm message passed along the edges of Tanner graph are binary bits. Initially a variable node sends a message to check nodes declaring if it is a 1 or 0. Then each check nodes calculates message for each variable node that what bit it should receive based on the information available to check node from other connected variable nodes i.e. Check node performs modulo-2 sum to verify the parity check equations. If the sum is zero then equation is satisfied otherwise bit is flipped and sent back to variable node. Now variable nodes have several bits one is initially received bit and other are various bits received from connected check nodes. Then variable nodes performs the majority check if result of the majority checks are same as the initial received bit then bit remains same else bit is flipped. This above process is continued until the all the parity checks equations are satisfied and all errors are detected.

2.2 Soft Decision decoding (Sum-Product):

Hard Decision Decoding operates on data that take on a fixed set of possible values (typically 0 or 1 in a binary code), the inputs to a soft-decision decoder may take on a whole range of values in-between. This extra information indicates the reliability of each input data point, and is used to form better estimates of the original data. Therefore, a soft-decision decoder will typically perform better in the presence of corrupted data than its hard-decision counterpart.

It is similar to bit flip algorithm but with the messages representing each decision (check met, or bit value equal to 1) now probabilities. Whereas bit-flipping decoding accepts an initial hard decision on the received bits as input, the sum-product algorithm is a soft decision algorithm which accepts the probability of each received bit as input. The input bit probabilities are called the a priori probabilities for the received bits because they were known in advance before running the LDPC de-coder. The bit probabilities returned by the decoder are called the a posteriori probabilities. In the case of sum-product decoding these probabilities are expressed as log-likelihood ratios.

For a binary variable x it is easy to find $p(x = 1)$ given $p(x = 0)$, since $p(x = 1) = 1 - p(x = 0)$ and so we only need to store one probability value for x . Log likelihood ratios are used to represent the metrics for a binary variable by a single value $L(x) = \{\log p(x=0)/p(x=1)\}$ where $L(x)$ is positive and the greater the difference between $p(x = 0)$ and $p(x = 1)$, i.e. the more sure we are that $p(x = 0)$, the larger the positive value for $L(x)$. Conversely, if $p(x = 1) > p(x = 0)$ then $L(x)$ is negative and the greater the difference between $p(x = 0)$ and $p(x = 1)$ the larger the negative value for $L(x)$.

The sum-product algorithm iteratively computes an approximation of the Posteriori value for each code bit. However, the a posteriori probabilities returned by the sum-product decoder are only exact probabilities if the Tanner graph is cycle free. Briefly, the extrinsic information obtained from a parity-check constraint in the first iteration is independent of the a priori probability information for that bit (it does of course depend on the a priori probabilities of the other codeword bits). LLR should be obtained with high accuracy, but it becomes complicated for some channel models such as wireless fading channel.

3. Min-Sum Decoding Algorithm

1. Interconnect representation of H matrix

- Two sets of nodes: Check nodes and Variable nodes.
- Each row of the matrix is represented by a Check node
- Each column of matrix is represented by a Variable node

$$H = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

2. The message received from the channel will be the values of variable nodes.

3. Message passing.
4. Variable Node Processing

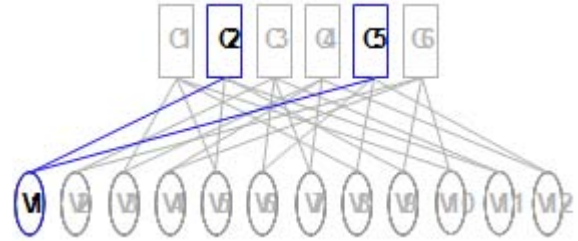


Figure 1: Variable node diagram

- According to matrix c2 and c5 are the check nodes having 1.
- The values stored in them will be transferred to the variable node, In 1 iteration these check nodes will not be having any values thus the values of variable nodes will remain the same but in later iterations these values will get updated until the code is error free.

5. Check Node Processing

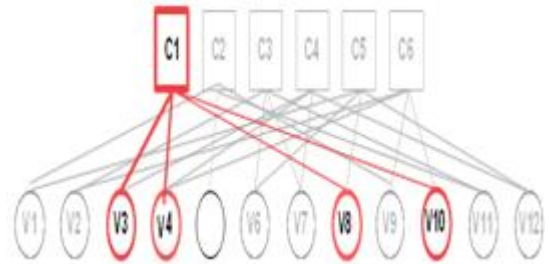


Figure 2: Check node diagram

- According to matrix V3, V4, V8, V10 are the variable nodes having 1.
- The values stored in the above nodes will be processed as: first node will be left as it is and the minimum of remaining three nodes will be checked as well as sign will be calculated.
- During this process the check nodes will get updated.

6. BPSK will be applied on the result(i.e. on values of variable nodes)

$$H = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

7. Syndrome Check: Binary Multiplication b/w the result and h matrix

$$\hat{V} = \begin{cases} 1, & \text{if } z_i \leq 0 \\ 0, & \text{if } z_i > 0 \end{cases}$$

8. If the output attain is zero, the variable node is the land else the iteration will continue from the updating of check nodes.

9. Diagrammatical Representation

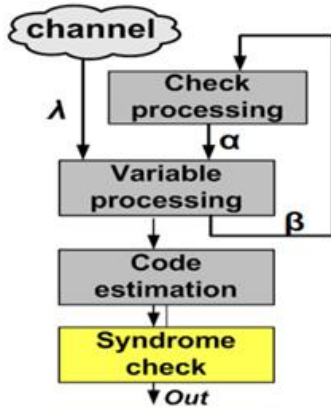


Figure 3: Decision diagram for the min-sum decoding Algorithm

4. Comparative Analysis

The proposed method in this paper has compared with the various papers and found better in the hardware utilization, We have compared the proposed and the conventional results in the form of table, it is showing that the proposed work have utilised less registers and slices. A comparison of the proposed decoder to that presented in [10],[11] is shown in Table 1

Table 1: Comparison of Fully Parallel LDPC Decoders

Paper	Ref[10]	Ref[11]	IMPROVEMENT
Register	8555	938	601
Slices	7755	1169	603

5. Results

Simulation results are achieved using Questa Sim .

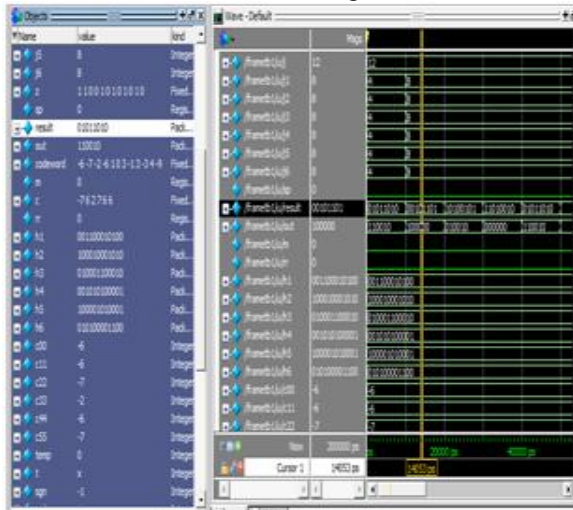


Figure 4: Simulation diagram for the min-sum decoding algorithm.

Synthesis Results

The synthesis results are evaluated using Virtex-5 FPGA and the table no. 2 showing the results of the hardware utilization.

Table 2: Device Utilization Summary

S. No	Logic utilization	Used	Available	Utilized in SPA
1	Number of slice registers	603	4800	13%
2	Number of slice LUTs	54243	2400	2260%
3	Number of fully used LUT-FF pairs	601	54245	1%
4	Number of bonded IOBs	8	102	8%
5	Number of BUFG/BUFGCTRLs	13	16	81%
6	Number of DSP48A1s	6	8	75%

6. Conclusion

The paper has given the complete hardware implementation of the min-sum LDPC decoder. By using the Verilog HDL we have optimized the code of decoder to achieve less hardware implementation results as described in the table. The implementation has achieved using less number of the hardware components viz. DFF, Registers.

References

- [1] GALLAGER R G. Low-density parity-check codes[J]. IRE Trans. Inform. Theory, 1962: 21-28.
- [2] KSCHISCHANG F R, FREY B J, LOELIGER H A. Factor graphs and the sum-product algorithm[J]. IEEETrans. Inform. Theory, 2001: 498-519.
- [3] A.nastasopoulos. A comparison between the sum-product and the min-sum iterative detection algorithms based on density evolution. in IEEE Global Telecommunications Conference. 2001.
- [4] FOSSORIER M P C, MIHALJEVIC M, IMAI H. Reduced complexity iterative decoding of low density parity check codes based on belief propagation[J]. IEEE Trans.Commun., 1999,47(5): 673-680.
- [5] CHEN J H, MARC P C. Near optimum universal belief propagation based decoding of low-density parity check codes[J]. IEEETrans. Commun., 2002,50(3):406-414.
- [6] CHEN Y H, HSIAO J H, HE J S. FPGA Implementation and Verification of LDPC Encoder with Weight (3, 6) Approximate Lower Triangular Matrix[C]. 2nd ICCSNT, 2012:531-534.
- [7] CHEN Y H, HSIAO J H, HE J S. FPGA Implementation and Verification of LDPC Decoder with Weight (3, 6) Approximate Lower Triangular Matrix[C]. 2nd ICCSNT, 2012:531-534.
- [8] R. Zarubica, et al. Efficient quantization schemes for LDPC decoders. in IEEE Military Communications Conference. 2008.
- [9] JIANG M, ZHAO C, ZHANG L, et al. Adaptive Offset Min-Sum Algorithm for Low- Density Parity Check Codes[J]. IEEE Commun. Letters, 2006,10(6):483-485.
- [10] Vikram Arkalgud Chandrasetty and Syed Mahfuzul Aziz .FPGA Implementation of High Performance LDPC Decoder using Modified 2-bit Min-Sum Algorithm Second International Conference on Computer Research and Development
- [11] Namrata P Bhavsar1, Asst. Prof.Brijesh Vala2, Asst. Prof.Ankit Pancholi. FPGA based Implementation of Decoding Algorithms of LDPC. <http://www.ijettjournal.org>