International Journal of Electronics Signals and Systems

Volume 2 | Issue 4 Article 15

April 2013

LDPC ARCHITECTURE IMPLEMENTATION BY REDUCING THE MEMORY UTILIZATION

B. L DOSS
Department of ECE, JNTU ANANTAPUR, bl.doss@gmail.com

LDPC ARCHITECTURE IMPLEMENTATION BY REDUCING THE MEMORY UTILIZATION

B L DOSS[1] & M SAILAJA[2]

[1,2]Department of ECE, JNTU ANANTAPUR

Abstract- As the low density parity check codes has proved their accuracy in error correcting .considering the ldpc as reference the architecture of ldpc is studied .ldpc coding contains check nodes and variable nodes which has their memory elements respectively .so an efficient use of memory can decrease the computation time. Further the arrays of memory requirement has been decreased by making the memory global to all the nodes . ldpc is considered as a finite state machine in which each node is a state .An efficient memory utilization method has been proposed to decrease the memory utilization in the fpga.

INTRODUCTION:

Ldpc codes are error correcting codes . LDPC architecture contains check nodes and variable nodes in which the iterations between them is continued till the stopping rule is met or error correction is completed.in this paper hard decoding method is considered in which complexity and memory requirement is high.

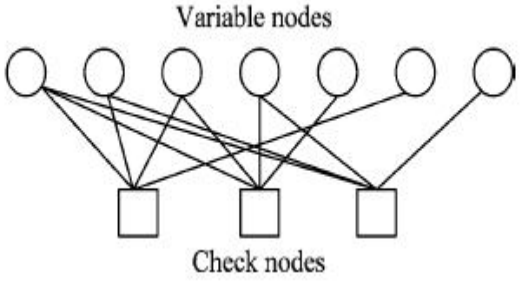


Fig.1 ldpc architecture tanner graph representation

LDPC matrix: LDPC error correction is preformed based on the matrix given which contains 1 and 0 .which relicates the communication between the variable nodes and check nodes.one example matrix is shown in the fig 2

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

Fig.2 ldpc matrix

The rest of the paper contains low density parity check implementation in section 2 .memory utilization in section 3.proposed method in section 4.results simulated for the proposed method in section 5.

Section 2
Ldpc implementation:

Considering a qth iteration between the check nodes and variable nodes the process of the ldpc as follows :

Inputs of the algorithm are the intrinsic Log-Likelihood Ratios (LLRs) of the received bits (i.e., the variable nodes), also referred to as a priori information.

Step 2:

The following describes the belief propagation algorithm. $R_{m,n}^{(q)}$ is the check-to-variable message from check node m to variable node n in the qth iteration. $Q_{m,n}^{(q)}$ is the variable-to check message from variable node n to check node m in the qth iteration. Mn is the set of the neighboring check nodes of variable node n, and Nm is the set of the neighboring variable nodes of check node m. In the qth iteration, the variable node process and the check node process are computed as follows.

Variable node process: the variable node n receives the messages $R_{m,n}^{(q)}$ from the neighboring check nodes and propagates back the updated messages $Q_{m,n}^{(q)}$ as

$$Q_{m,n}^{(q)}=\lambda_n+\sum_{i\in\{M_n\backslash m\}}R_{i,n}^{(q)} \qquad (2)$$

where λ_n is the intrinsic LLR of the variable node . At the same time, the posterior reliability

value, also referred to as the soft output for the variable node , is given by

$$\Lambda_n^{(q)} = \lambda_n + \sum_{i \in \{M_n\}} R_{i,n}^{(q)}, \qquad (3)$$

Check node process: the check node m combines together messages $Q_{m,n}^{(q)}$ from the neighboring variable nodes to compute the updated messages $R_{m,n}^{(q+1)}$, which are sent back to the corresponding variable nodes. Updates are performed separately on signs and magnitudes as follows:

$$-\mathrm{sgn}\left(R_{m,n}^{(q+1)}\right) = \prod_{j \in \{N_m \setminus n\}} -\mathrm{sgn}\left(Q_{m,j}^{(q)}\right) \qquad (4)$$

$$\left|R_{m,n}^{(q+1)}\right| = \Phi^{-1}\left\{\sum_{j \in \{N_m \setminus n\}} \Phi\left(\left|Q_{m,j}^{(q)}\right|\right)\right\} \qquad (5)$$

Where

$$\Phi(x) = \Phi^{-1}(x) = -\ln\left(\tanh\left(\frac{x}{2}\right)\right). \qquad (6)$$

The layered decoding scheduling improves the convergence speed and reduces the number of iteration by viewing the parity check as a sequence of check through horizontal or vertical layers. The intermediate updated messages are used in the updating of the next layer

$$\Gamma_{m,n}^{(q+1)} = \Lambda_n^{(q+1)}[k-1] - R_{m,n}^{(q)}$$

$$\Lambda_n^{(q+1)}[k] = \Gamma_{m,n}^{(q+1)} + R_{m,n}^{(q+1)}$$

Section 3
Memory utilization of ldpc implementation

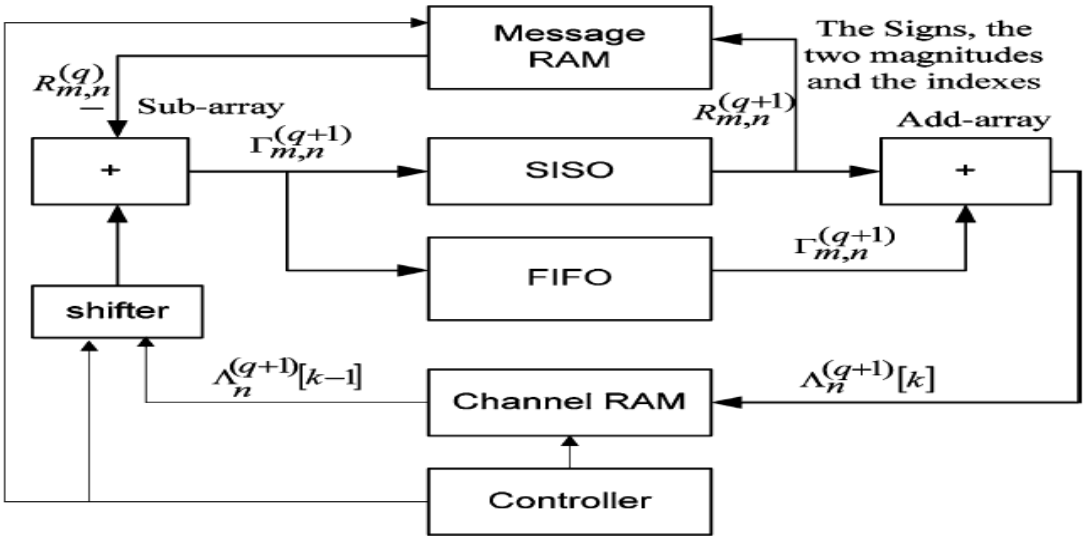


Fig 3 memory utilization factors as channel ram and message ram

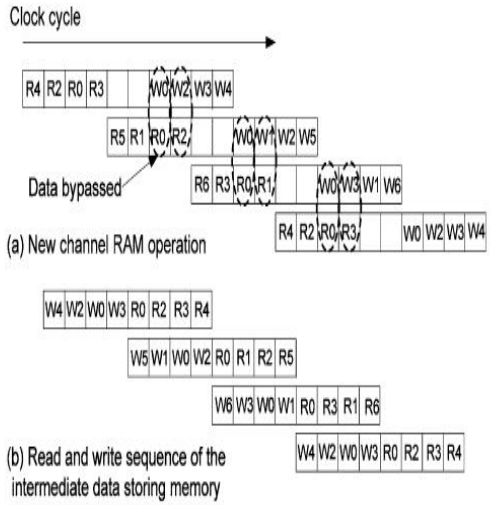
In the layer decoding architecture, during the decoding, for every layer, the soft messages are read from and wrote into the Channel RAM and the FIFO every cycle. The Channel RAM stores the soft posterior reliability values of the variable nodes. The updated values are stored back from the Add-array and will be used in the decoding of the subsequent layer. In the memory-bypassing scheme [10], when two consecutive layers have non-null entry at the same column, the results of the Addarray can be directly sent to the cyclic shifter and used for the decoding of the next layer without the need of storing the intermediate result in the Channel RAM. The memory bypassing scheme saves the write operation for the current layer and the read operation for the next layer. Fig. 4 shows an example. Fig. 4(a) shows a base matrix with three layers and Fig. 4(b) shows the timing diagram of the pipeline.Without any memory bypassing, the number of read and write access of the Channel RAM is equal to the non-null entries in the matrix. In this example, the total number of read and write operation is 12. If

bypassing scheme is employed, instead of writing back the channel RAM, the updated soft output values are used directly for the decoding of the next layer, the number of memory access is reduced. For example memory access for columns 0 and 2 can be bypassed when the decoding proceeds from layer 0 to layer 1; memory access for columns 0 and 1 can be bypassed for the second layer decoding and memory access for columns 0 and 3 can be bypassed for the third layer decoding. Thus, 6 out of the 12 read and write operations are eliminated, and 50% of the power consumption of the Channel RAM can be saved. Consdrering the ldpc matrix in the section 1 .the read and write operation between the variable nodes and check nodes is as follows Fig 4 ldpc implementation as fsm .s1,s2,s3 are the check nodes and the rest are variable nodes

As an example considering the shortest path s0 to s1 to s7
Considering the inputs and the outputs

S0 state:

Input : ideal
Data output is : 01010101
Next state: s1



Section 4

Proposed method

As nodes are considersd as states the proposed method as follows

fsm Data output : 01010101
Next state: exit

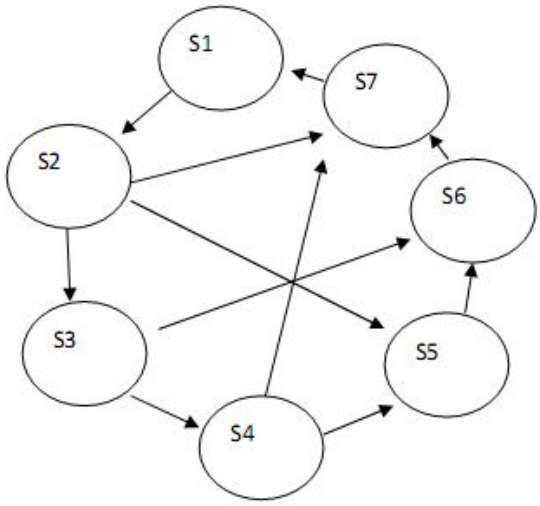
The memory architecture for the above example

S1 state

Input : s0 state
Data output :01011101
Next state: s7

S7 state

Input :s1 state



General fsm output:

Table 1: memory of general fsm

03	01010101
02	01011101
01	01010101

Proposed method output:

Table 2: memory of proposed fsm

02	01011101
01	01010101

Parameters

S1 state : pres state: s0
Next state: s1

Curr state: 01
S2 state: pres state: s1
Next state: s2
Curr state: 02
S3 state: pres state: s2
Next state: s3
Curr state:01

Section 5
Simulation results:

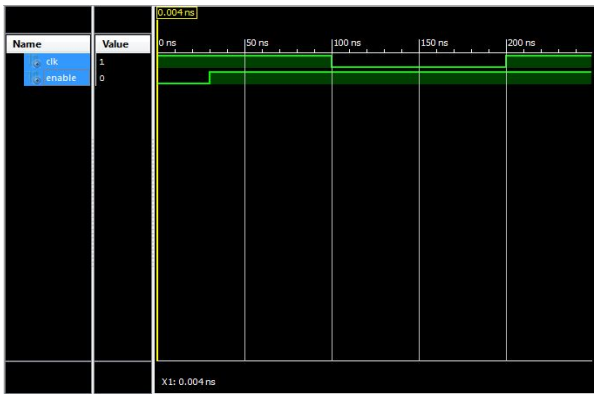


Fig 5.Clock and enable of the circuit

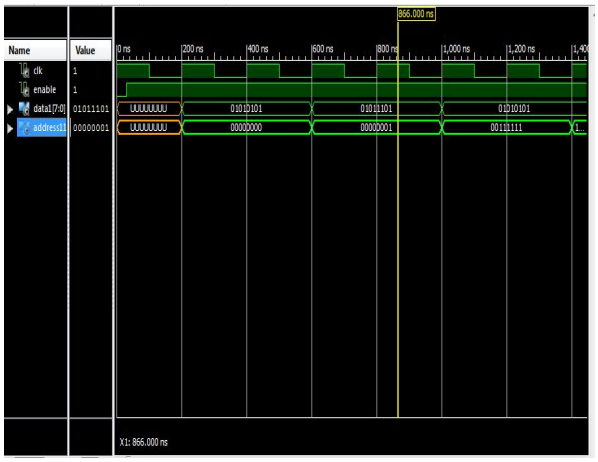


Fig 6 The data and state input (finite state machine s0 to s1 to s7)

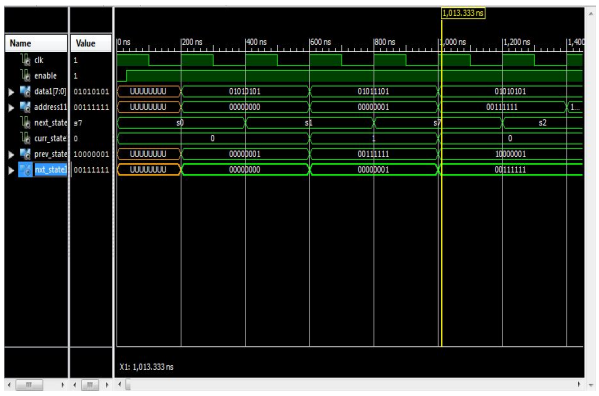


Fig 7 Curr state indicates the current state output

Next state and prev state are indicated as same The ram state clearly states the efficient memory pattern matching.the three states output are stored in the 2 arrays in the memory

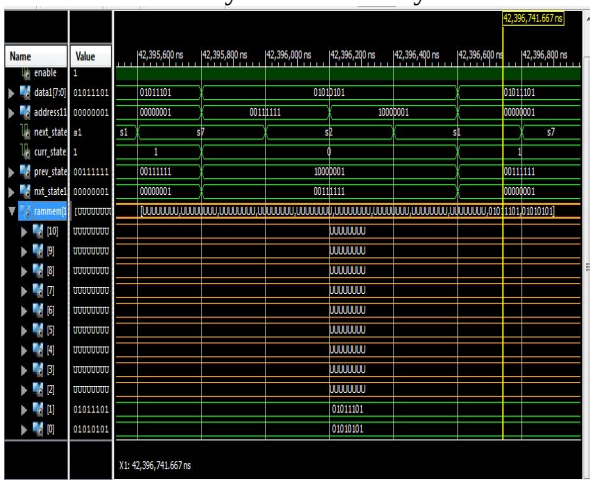


Fig 8.proposed method memory utilization

CONCLUSION

We presented an improved memory-efficient scheme to reduce the memory access and hence the energy consumption of the LDPC decoder by exploiting the characteristic of the LDPC parity check matrix.

REFERENCES:

[1] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. IT-8, pp. 21–28, Jan. 1962.

[2] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399 431, Mar. 1999.

[3] Digital Video Broadcasting (DVB); Second Generation Framing Structure, Channel Coding and Modulation Systems for Broadcasting, Interactive Services, News Gathering and Other Broadband Satellite Applications 2004.

[4] *LDPC coding for OFDMA PHY. 802.16REVe Sponsor Ballot Recirculation Comment* 2004, IEEE C802.16e-04/141r2.

[5] *Joint Proposal: High Throughput Extension to the 802.11 Standard: PHY. IEEE P802.11 Wireless LANs* 2006, IEEE 802.11-05/1102r4.

[6] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.

[7] A. J. Blanksby and C. J. Howland, "A 690-mW 1-Gb/s 1024-b, rate ½ low-density parity-check code decoder," *IEEE J. Solid State Circuits*, vol. 37, pp. 404–412, Mar. 2002.