Low Power LDPC Decoder with Efficient Stopping Scheme for Undecodable Blocks

Tinoosh Mohsenin, Houshmand Shirani-mehr and Bevan Baas

***Abstract*—An efficient technique for early detection of undecodable blocks during LDPC decoding is introduced. The proposed method avoids unnecessary decoding iterations by predicting decoding failure and therefore results in significant improvement in power and latency in low SNR values. The proposed method which has a low hardware overhead compares the parity checksum against predefined threshold values for three iterations and terminates decoding if a condition is met. A 5.25 mm^2 10GBASE-T Split-Row Threshold decoder is implemented using the proposed technique in 65 nm CMOS. The postlayout results show that at low SNR value of 3.0 dB, the decoder requires 2.3 times fewer decoding iterations which results in 23 pJ/bit energy dissipation. This is 2.4 times lower than the energy dissipation of Split-Row Threshold decoder without the proposed early stopping technique.**

***Index Terms*—early termination, undecodable blocks, full parallel, LDPC, low power, energy efficiency, 10GBASE-T, 65 nm CMOS**

I. Introduction

Low-density parity-check (LDPC) codes [1], [2] have been adopted by many communication standards due to their significant error correction performance. An efficient technique to reduce the energy dissipation and decoding latency is through controlling the number of decoding iterations that a block requires for a successful decoding convergence. The common method is to verify if the computed codeword satisfies all parity check constraints, at the end of each iteration. Once convergence has been verified, the decoding process is terminated. However at low SNR ranges it happens frequently that a block can not converge to a valid codeword even after a maximum number of decoding iterations (*Imax*). Thus early detection of an undecodable block is desired to avoid unnecessary iterations and therefore to reduce power dissipation and decoding latency.

There have been recent studies to determine undecodable blocks at early decoding stages. The proposed stopping methods in [3], [4] monitor the convergence of mean magnitude of variable nodes and check nodes, respectively. The proposed technique in [5] uses the number of satisfied parity check constraints as the decision metric. In contrast, in [6] the summation of unsatisfied parity check constraints are monitored. These two methods track the fluctuation of parity checksum at a certain number of iterations and compare its magnitude with threshold values which are SNR dependent. All the previous methods propose a tradeoff between error performance and decoder hardware complexity.

In this work instead of checking the fluctuation of parity checksums at every iteration, we compare the parity checksum against a set of predetermined threshold values at three iterations. The threshold values and the iteration number are obtained empirically for the LDPC codes which are used in the system. For illustration of our proposed algorithm we use (6,32) (2048,1723) RS-LDPC code [7] which is adopted by the 10GBASE-T standard [8]. The proposed algorithm can be used in both MinSum and SPA decoding methods. Because of the significant benefits of recently proposed Split-Row Threshold decoding [9] in circuit area, delay and power reduction we implement the proposed algorithm for Split-Row Threshold decoder.

II. Background

An LDPC code is defined by an $M \times N$ parity check matrix H, which encapsulates important matrix parameters: the number of rows, M, is the number of check nodes; the number of columns (or *code length*), N, is the number of variable nodes; row weight W_r and column weight W_c, which define the 1's per rows and columns, respectively.

The LDPC decoding algorithm works by performing an iterative computation known as *message passing*. MinSum Normalized [10] is among the most common iterative message-passing algorithms. The recently proposed *Split-Row* [11] and *Split-Row Threshold* [12], [9] algorithms significantly reduce the interconnect complexity and circuit area by partitioning the links needed in the message-passing algorithm.

Both algorithms are defined by a check node update equation that generates α, and a variable node update equation that generates β.

A. MinSum Normalized and Split-Row Threshold Decoding Method

The MinSum Normalized check node update equation is given as:

$$\alpha_{ij} = Sfactor_{MS} \times \underbrace{\prod_{j' \in V(i) \setminus j} \text{sign}(\beta_{ij'})}_{\text{Sign Calculation}} \times \underbrace{\min_{j' \in V(i) \setminus j} (|\beta_{ij'}|)}_{\text{Magnitude Calculation}} \quad (1)$$

where α_{ij} message magnitude is generated using the minimum value of $|\beta|$ messages from all variable nodes $V(i)$ connected to check node C_i as defined by H (excluding V_j). The normalizing scaling factor $Sfactor_{MS}$ is included to improve error performance.

In Split-Row Threshold the check node processing is partitioned into *Spn* nearly-independent simplified partitions. α_{ij}

message in each partition *Spi* is computed as follows:

$$\alpha_{ij:Spi} = Sfactor_{Split} \times \underbrace{\prod_{j' \in V(i) \setminus j} \text{sign}(\beta_{ij'})}_{\text{Sign Calculation}} \times \underbrace{Min_{Spi}}_{\text{Magnitude Calculation}} \quad (2)$$

In Eq.2, sign computation is complete. However, Min_{Spi} is computed based on a comparison between the minimum value of $|\beta|$ messages in partition *Spi* and a predefined threshold value (T). A minimal amount of information is transferred amongst partitions to ensure computational accuracy while reducing global communication: One is the sign bit (*Sign*) computed in each partition. The other is a single bit information based on a comparison with threshold T and is called $Threshold\_en$. These two signals are sent between each partition. Details of the algorithm are found in [9].

Both algorithms are followed by the variable processing step which computes β messages according to Eq.4. At the end of each iteration l, sign of $z_j^{(l)}$ is taken as the estimated codeword bit $\hat{x}_j$ (mapping +1 to 0 and -1 to 1). The parity checksum value ($checksum(i)$) corresponding to check node C_i is computed by Eq.6, where $\bigoplus$ represents binary addition. Syndrome ($s_{check}^{(l)}$) is found by adding all $checksum(i)$ values according to Eq.7. If $checksum(i) = 0$ for every check node C_i, a valid code is found and the decoding process can be terminated.

III. Proposed Stopping Scheme

It is shown that the block is most likely decodable if syndrome value (s_{Check}) monotonically decreases as decoding iteration count (l) increases [6], [3]. In contrast, for undecodable blocks, s_{Check} fluctuates in a range of magnitudes. Figure 1 shows the variation of checksum for a (6,32) (2048,1723) LDPC code versus iteration count, at SNR=4.0 dB for 1000 blocks. As shown in the figure, for majority of decodable blocks $s_{check}^{(l)}$ starts decreasing within iteration interval of 4 to 6. Based on this observation, the algorithm works as follows: at the end of each iteration, it computes the syndrome ($s_{check}^{(l)}$) from decoded bits $\hat{x}$. If $s_{check}^{(l)} = 0$, it means decoder has converged and decoding process is terminated. Otherwise, if current iteration $l = Iter_{check} - 2, Iter_{check} - 1$ or $Iter_{check}$, it compares S_{Check} with a corresponding threshold value ($TH1$, $TH2$ or $TH3$). If $s_{check}^{(l)}$ is larger than all three values for corresponding iterations **Condition 1** is met, which implies that the decoder most probably can not converge within maximum number of iterations thus it terminates decoding. The algorithm is summarized in **Algorithm 1**.

A. BER Simulation Results

The error performance depends strongly on the choice of threshold values. If the threshold values $TH1$, $TH2$ and $TH3$ are very large, it is less likely that **Condition 1** can be met and therefore algorithm converges to original Split-Row Threshold, thus the error performance and average iteration remain the same. On the other hand, if the threshold values are very small, **Condition 1** is always met which results in decoder

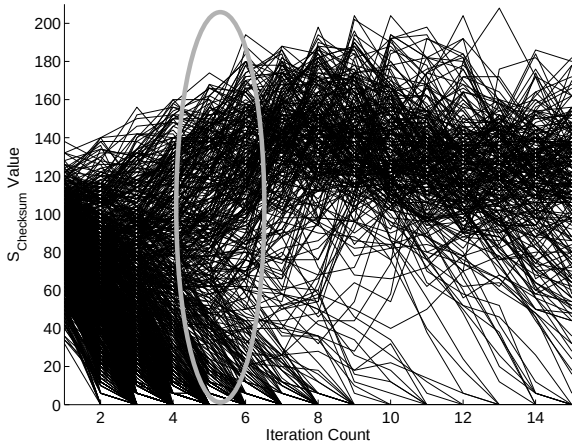


Fig. 1. Syndrome values versus iteration count for (2048,1723) LDPC code at SNR=4.0 dB. The likely values of iteration count that can be checked for undecodable blocks are highlighted.

Algorithm 1 Stopping Scheme Algorithm

Require: λ, i.e. channel information
for $l = 0, 1, \ldots, Imax - 1$ **do**
  for $i = 0, 1, \ldots, M - 1$ **do**
    for $j = 0, 1, \ldots, N - 1$ **do**
      do Eq.1 or Eq.2
    end for
  end for
  for $i = 0, 1, \ldots, M - 1$ **do**
    for $j = 0, 1, \ldots, N - 1$ **do**

$$z_j^{(l)} = \lambda_j + \sum_i \alpha_{ij}^{(l)} \quad (3)$$

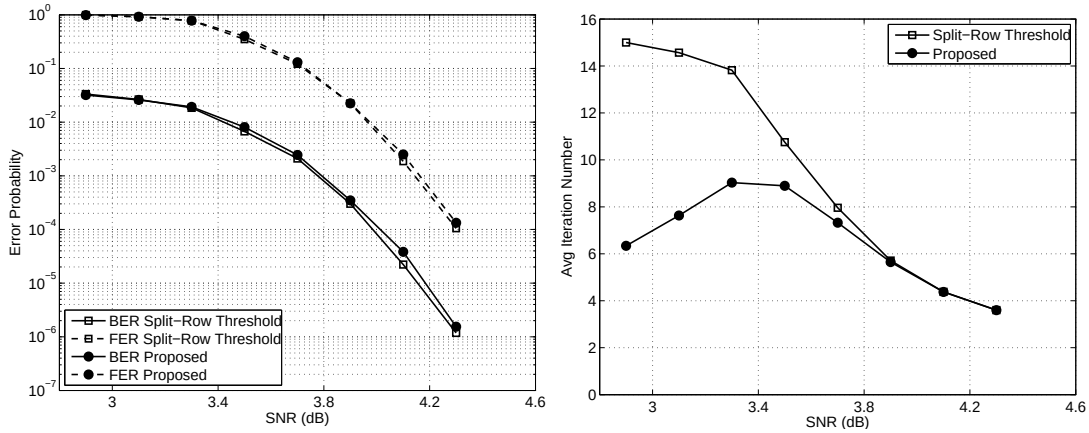
$$\beta_{ij} = z_j^{(l)} - \alpha_{ij}^{(l)} \quad (4)$$

$$\hat{x}_j = \begin{cases} 1, & \text{if } z_j \leq 0 \\ 0, & \text{if } z_j > 0 \end{cases} \quad (5)$$

$$checksum(i)^{(l)} = \bigoplus_{j \in V(i)} x_j^{(l)} \quad (6)$$

$$s_{check}^{(l)} = \sum_{i=1}^{M} checksum(i)^{(l)} \quad (7)$$

    end for
  end for
  if $s_{check}^{(l)} = 0$ **then**
    Terminate decoding
  else if $l = Iter_{check}$ **then**
    Condition 1:
    if $s_{check}^{(l-2)} \geq TH1$ **and** $s_{check}^{(l-1)} \geq TH2$ **and** $s_{check}^{(l)} \geq TH3$ **then**
      Terminate decoding
    end if
  end if
  $l = l + 1$
end for



(a) BER and FER (b) Average number of iterations

Fig. 3. (a) Bit error rate (BER) and frame error rate (FER) and (b) average number of iterations versus SNR for (2048,1723) LDPC code.

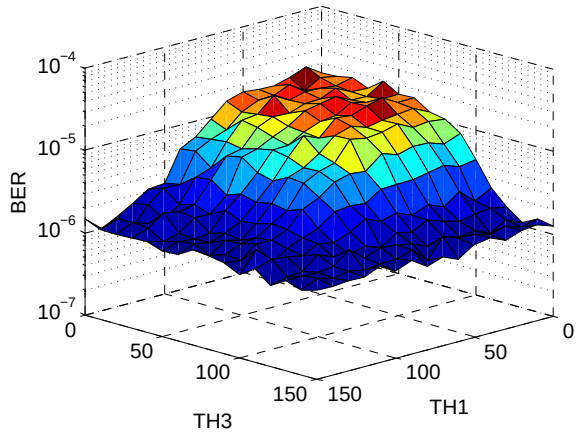


Fig. 2. Bit error performance of (2048,1723) LDPC code versus two threshold values. $TH2$ is chosen to be 95.

termination before maximum number of decoding iterations and therefore error performance becomes worse.

The optimum values for $iter_{check}$, $TH1$, $TH2$ and $TH3$ are obtained by empirical simulations. We use BPSK modulation and an additive white Gaussian noise (AWGN) channel for all simulations. Simulations were run until 80 error blocks were recorded. Maximum number of iterations $Imax$ is 15. For the 2048-bit 10GBASE-T code using Split-Row Threshold, $iter_{check}$ = 6, $TH1$ = 120, $TH2$ = 95 and $TH3$ = 100 result in optimum error performance and average iteration count. To further illustrate the impact of the threshold value on error performance, Fig. 2 plots the bit error performance (BER) of a (6,32) (2048,1723) versus $TH1$ and $TH3$ values at SNR=4.3 dB. As shown in the figure, for $TH1$ and $TH3$ larger than 100, there is no significant improvement on the BER indicating that the error performance is near to that of original Split-Row Threshold. Fig. 3 (a) and (b) show bit error rate (BER), frame error rate (FER) and average number of iterations versus SNR for the the 2048-bit 10GBASE-T code using Split-Row Threshold and the proposed stopping method with optimum threshold values $TH1$ = 120, $TH2$ = 95 and $TH3$ = 100. As shown in the figure, the proposed algorithm performs very closely to Split-Row Threshold with a 0.04 dB gap at BER = 10^{-7} and with 2.4 times less iteration at best case.

IV. Full parallel Decoder Implementation and Results

The block diagram of a full parallel implementation of Split-Row Threshold decoding with the proposed stopping method is shown in Fig. 4. The decoder consists of Spn partitions, each partition performs check node processing in parallel and sends its $Sign$ and $Threshold\_en$ signals to the next partition according to Split-Row Threshold algorithm [9]. These are the only wires passing between the partitions. In full parallel implementation, all check and variable processor outputs are updated in parallel, and as shown in the block diagram, it takes one cycle to update all messages for one iteration. At every iteration, the *Syndrome check* block computes S_{check} and compares it with zero and also with $TH1$, $TH2$, $TH3$ or Max_S_{Check} at corresponding iteration ($iteration = iter_{check} - 2, iter_{check} - 1$ or $iter_{check}$ or else). If **Condition 1** is met or $S_{check} = 0$, or iteration count reaches maximum number of iterations I_{max}, a signal ($Termination\_en$) is sent to registers to disable the clock and therefore terminate decoding process.

To further investigate the impact of the proposed stopping scheme on the hardware implementation, we have implemented two full parallel decoders using original Split-Row Threshold and the proposed method for the (6,32) (2048,1723) 10GBASE-T LDPC code in 65 nm, 7-metal layer CMOS.

The parity check matrix of the 10GBASE-T code has 384 rows, 2048 columns, row weight 32 (W_r = 32), column weight 6 (W_c = 6) and information length 1723. The fully-parallel MinSum Normalized decoder has 384 check and 2048 variable processors corresponding to the parity check matrix dimensions M and N, respectively.

Table I summarizes the post layout implementation results for the original Split-Row Threshold. The amount of hardware overhead to implement the proposed stopping method is very small (less than 1% increase) compared to original Split-Row Threshold. Due to the nature of Split-Row Threshold algorithm, which significantly reduces wire interconnect complexity, both full parallel decoders achieve a very high

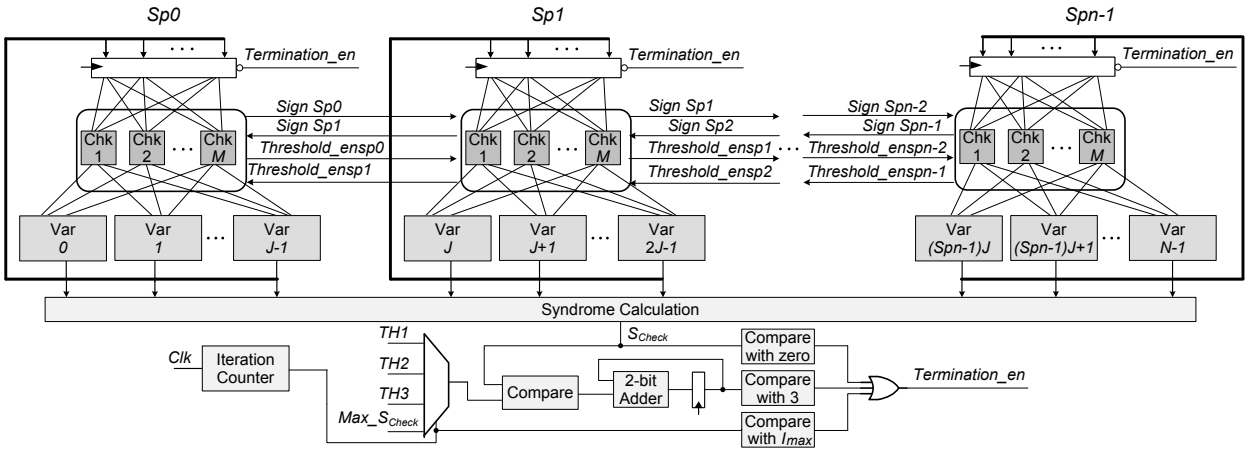


Fig. 4. Top level block diagram of a full parallel decoder corresponding to an $M \times N$ parity check matrix, using Split-Row Threshold with early stopping method. The decoder includes *Spn* partitions. The inter-partition $Sign$ and $Threshold\_en$ signals are highlighted. $J = N/Spn$, where N is the code length. Max_S_{check} is a predefined number which makes the comparison result with S_{Check} to be zero to make the stopping logic function correctly.

	Split-Row Threshold	Proposed
CMOS fabrication process	65 nm CMOS, 1.3 V	
Final area utilization	97%	96%
Area (mm^2)	5.2	5.25
Worst case speed (MHz)	188	186
Average No. of iterations @SNR=3.0 dB	15	6.3
Throughput @SNR=3.0 dB (Gbps)	25.7	**60.1**
Energy per bit @SNR=3.0 dB (pJ/bit)	56	**23**
Average No. of iterations @SNR=4.3 dB	3.6	3.6
Throughput @SNR=4.3 dB (Gbps)	106.8	105.8
Energy per bit @SNR=4.3 dB (pJ/bit)	13.4	13.2

TABLE I
COMPARISON OF FULLY-PARALLEL DECODERS IN 65 nm, 1.3 V CMOS, FOR A (6,32) (2048,1723) CODE IMPLEMENTED USING SPLIT-ROW THRESHOLD AND THE PROPOSED EARLY STOPPING TECHNIQUE.

logic utilization, 96%–97%. The throughput and energy data are reported for 15 maximum decoding iterations with early termination at two SNR = 3.0 dB and 4.3 dB. The throughput varies from 60.1 to 105.8 Gbps and the energy dissipation changes from 23 down to 13.2 pJ/bit. As shown in the table, at SNR=3.0 dB, the proposed stopping scheme results in 2.4 times reduction in the iteration count which results in 2.3 times and 2.4 times increase in the throughput and energy efficiency, respectively.

V. CONCLUSION

This paper presents an early stopping technique for undecodable blocks to reduce unnecessary decoding iterations and therefore power dissipation and latency. The method has a low hardware overhead while maintaining the error correction performance, only 0.04 dB reduction at BER = 10^{-7} compared to the original decoding method. Postlayout results show that the 10GBASE-T decoder implemented with the proposed method dissipates 23 pJ/bit at SNR=3.0 dB which is 2.4 times lower than the original decoder.

VI. ACKNOWLEDGMENTS

The authors gratefully acknowledge support from ST Microelectronics, Intel, UC Micro, NSF Grant 0430090, CAREER Award 0546907, and Grant 0903549, SRC GRC Grants 1598 and 1971 and CSR Grant 1659, Intellasys, SEM, and a UCD Faculty Research Grant.

REFERENCES

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

[2] D. MacKay, *Information Theory Inference and Learning Algorithms*, 3rd ed. Cambridge, UK: Cambridge University Press, 2003.

[3] Z. Cai, J. Hao, and L. Wang, "An efficient early stopping scheme for LDPC decoding based on check-node messages," in *Communication Systems, 2008. ICCS 2008. 11th IEEE Singapore International Conference on*, Nov. 2008, pp. 1325–1329.

[4] J. Li, X. hu You, and J. Li, "Early stopping for LDPC decoding: convergence of mean magnitude (CMM)," *Communications Letters, IEEE*, vol. 10, no. 9, pp. 667–669, Sep. 2006.

[5] D. Shin, K. Heo, S. Oh, and J. Ha, "A stopping criterion for low-density parity-check codes," in *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, 2007, pp. 1529 –1533.

[6] L. C. Z. Cui and Z. Wang, "An efficient early stopping scheme for LDPC decoding," in *13th NASA symposium on VLSI design*, Jun. 2007, http://www.cambr.uidaho.edu/symposiums/13TH_NASA_VLSI_Proceedings/.

[7] I. Djurdjevic, J. Xu, K. Abdel-Ghaffar, and S. Lin, "A class of low-density parity-check codes constructed based on Reed-Solomon codes with two information symbols," *IEEE Communications Letters*, vol. 7, no. 7, pp. 317–319, Jul. 2003.

[8] "IEEE P802.3an, 10GBASE-T task force," http://www.ieee802.org/3/an.

[9] T. Mohsenin, D. Truong, and B. Baas, "A low-complexity message-passing algorithm for reduced routing congestion in ldpc decoders," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 57, no. 5, pp. 1048–1061, May. 2010.

[10] J. Chen, A. Dholakia, E. Eleftheriou, and M. Fossorier, "Reduced-complexity decoding of LDPC codes," *IEEE Transactions on Communications*, vol. 53, no. 7, pp. 1288–1299, Aug. 2005.

[11] T. Mohsenin and B. Baas, "A split-decoding message passing algorithm for low density parity check decoders," *Journal of Signal Processing Systems*, vol. 61, pp. 329–345, Feb. 2010, 10.1007/s11265-010-0456-y.

[12] T. Mohsenin, D. Truong, and B. Baas, "An improved Split-Row Threshold decoding algorithm for LDPC codes," in *Communications, 2009. ICC '09. IEEE International Conference on*, Jun. 2009.