A 220mW 1Gb/s 1024-Bit Rate-1/2 Low Density Parity Check Code Decoder

Chris Howland and Andrew Blanksby

High Speed Communications VLSI Research Department
Agere Systems
Holmdel NJ 07733
Email: {howland,blanksby}@agere.com

Abstract

A 1024 bit rate-1/2 Low Density Parity Check (LDPC) code decoder has been implemented that matches the coding gain of equivalent turbo codes. The parallel decoder architecture supports throughputs up to 1Gb/s and convergence in the decoding algorithm translates into extremely low switching activity with power dissipation under 220mW.

Introduction

Error correcting codes are used to increase the bandwidth and power efficiency of communications systems [1]. The invention of turbo codes and block turbo codes has moved the limit of achievable coding gain much closer to the Shannon limit for low and high code rates[1] respectively[2, 3]. However, the iterative nature of the decoding algorithms for turbo codes and block turbo codes present significant implementation challenges. Each pass through the data block to perform a decoder iteration requires the fetching, computation, and storage of large amounts of state information. Performing multiple iterations to achieve high coding gain reduces throughput and increases power dissipation [4].

The interest in iterative decoding algorithms has led to the rediscovery of Low Density Parity Check (LDPC) codes. LDPC codes were invented by Gallager in 1962 but were not pursued for due to implementation complexity [5]. It has been recently shown that LDPC codes are asymptotically superior to turbo codes with respect to coding gain [6]. The most powerful code currently known is a 1 million bit, rate-1/2 , LDPC code, achieving a capacity which is only 0.13dB from the Shannon limit for a bit error probability (BER) of 10^{-6} [6]. However, unlike turbo and block turbo codes it is possible to decode LDPC codes using a block-parallel algorithm instead of a block-serial algorithm.

In this paper we present a parallel architecture for LDPC decoders that achieves both very high throughput and extremely low power dissipation. This architecture is demonstrated through the implementation of a 1024 bit, rate-1/2 , soft decision LDPC decoder.

1. The rate of a code is defined as the ratio of the information bits to the sum of the information and parity bits. A low-rate code has a large redundancy overhead while a high-rate code has a small overhead.

Low Density Parity Check (LDPC) Codes

Low density parity check codes are linear block codes thus the set of all codewords, x, span the null space of a parity check matrix $\boldsymbol{H}$:

$$\boldsymbol{H}x = 0 \tag{1}$$

The parity check matrix $\boldsymbol{H}$ for LDPC codes is a sparse binary matrix where the set row and column elements are chosen to satisfy a desired row and column weight profile [6], and maximize the length of cycles in the graph representation of the matrix [7, 8]. The general structure of $\boldsymbol{H}$ is shown in Fig. 1.

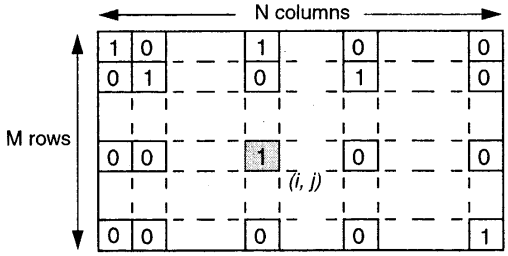

Fig. 1. General structure of a low-density parity check matrix $\boldsymbol{H}$.

Each row of $\boldsymbol{H}$ corresponds to a parity check and a set element (i, j) indicates that data symbol j participates in parity check i. In a block of N bits or symbols, there are M redundant parity symbols and the code rate R is given by:

$$R = (N - M)/N \tag{2}$$

A. Graph Representation of LDPC Codes

Low density parity check codes can also be represented using a bipartite graph, where one set of nodes represents the parity check constraints and the other set represents the data symbols or variables as illustrated in Fig. 2.

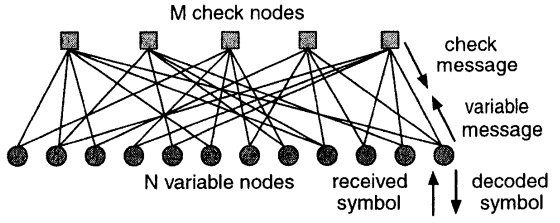

Fig. 2. Example of the bipartite graph representation for a LDPC code and information flow in the message passing algorithm.

0-7803-6591-7/01/$10.00 © 2001 IEEE

A variable node v_j, corresponding to column j in $\boldsymbol{H}$, is connected to check node c_i, corresponding to row i in $\boldsymbol{H}$, if the entry (i, j) in $\boldsymbol{H}$ is set, i.e. non zero.

B. The Message Passing Algorithm

The message passing algorithm is an iterative algorithm for decoding LDPC codes best understood with reference to the graph representation of an LDPC code [5]. It can be summarized as follows:

1. Initialize all variable nodes and their outgoing variable messages to the value of the corresponding received bit.
2. Propagate the variable messages from the variable nodes to the check nodes along the edges of the graph.
3. Perform a parity check (XOR) on the incoming variable messages at the check nodes. Form each check message as the XOR of the incoming variable message and the parity check result. This is the value all other connected variables imply the variable corresponding to each edge should take.
4. Pass the check messages from the check nodes back to the variable nodes along the edges of the graph.
5. At the variable nodes update estimates of the decoded bit and outgoing variable messages for each edge connected to the variable node using a weighted majority function or summation.
6. Repeat steps 2-5 until a termination condition is met. Possible iteration termination conditions include:
   - The estimated decoded block x satisfies (1).
   - The current messages passed to the parity check nodes satisfy all of the parity checks. This does not guarantee that (1) is satisfied but is almost sufficient and is simple to test.
   - Stop decoding after a fixed number of iterations.

Both hard and soft decision forms of the message passing algorithm are possible.

A Parallel 1024 bit, Rate 1/2 Soft Decision LDPC Decoder

A parallel architecture for the message passing algorithm can be obtained by directly instantiating the LDPC graph in hardware. As seen in Fig. 2 the computational dependencies for any node depend only on nodes of the opposing type. Therefore, all variable nodes and all check nodes can alternately be updated in a block-parallel fashion supporting very high throughput.

To demonstrate the utility of a parallel architecture for decoding LDPC codes a prototype decoder was implemented for a 1024 bit, rate-$1/2$ LDPC code. This data point corresponds to one of the block size and code rates proposed for 3rd generation (3G) wireless turbo codes [9]. To achieve the highest possible coding gain at low signal-to-noise ratios (SNR) the decoder uses a soft decision version of the message passing algorithm. The parity check matrix $\boldsymbol{H}$ for the code has an average column weight of 3.25, with columns of weight t=3, 6, 7 and 8. This corresponds to an average row weight of 6.5 and 256 weight k=6 and 256 weight k=7 parity checks were chosen to satisfy this constraint and the code rate of $1/2$.

Although the coding gain of a 1024 bit, rate-$1/2$ LDPC code is theoretically inferior to that of a turbo code [6], implementation restrictions on the number of turbo code decoder iterations do not enable the realization of this difference. In Fig. 3 the coding gain of the LPDC decoder is compared to the 1024 bit rate-$1/2$ turbo code in the 3G proposal decoding using the SOVA and MAP decoding algorithms [10, 11]. The coding performance of the LDPC code is comparable to that of the turbo code when decoded with 6 iterations of a MAP decoder for packet error rates (PER) lower than 1%.

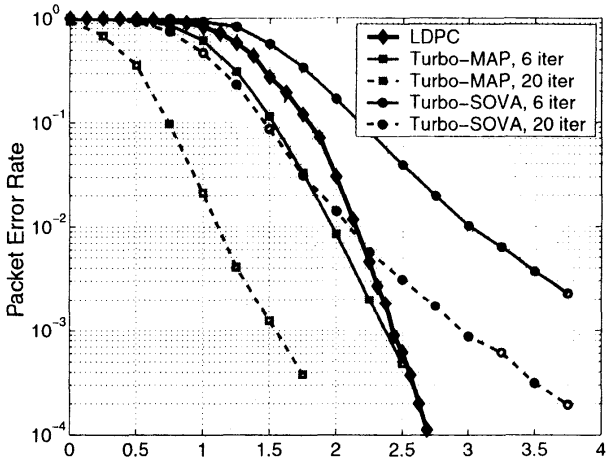

Fig. 3. 1024-bit, rate-1/2, LDPC code (4-bit messages) and 3G wireless turbo code using MAP and SOVA decoders (full floating point precision)[1]

Although the decoder was intended to be compared to 3G turbo codes which have a throughput of 1-2Mb/s, the prototype was designed to achieve a maximum throughput of 1Gb/s to demonstrating the extremely high throughput capability of the parallel architecture.

A. Data Input/Output

The parallel decoder can be considered as a three block pipeline. While one block is being iteratively decoded, the next block is loaded into the decoder, and the previous block is read-out from the decoder. Data is loaded and unloaded from the decoder using a scan chain, with a width of w samples and depth d, such that:

$$w \cdot d = N \tag{3}$$

The number of decoder iterations performed on all blocks is set to d. The columns of $\boldsymbol{H}$, and hence the variable nodes, are divided into w groups, each separated by w bits as shown in

1. One decoder iteration for the turbo codes is taken to mean one constituent codes trellis decoding.

Fig. 4. The decoder was implemented with $w = 16$ and $d = 64$, performing 64 decoder iterations for every packet. To achieve the desired throughput of 1 Gb/s the required clock frequency is only 64MHz.

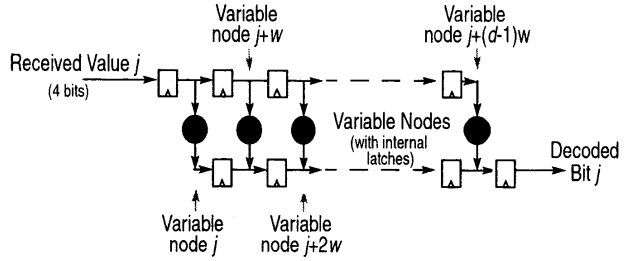

Fig. 4. Architecture of variable scan group j.

B. Check Node Architecture

Each check node performs a parity check across all variables in a row of $\boldsymbol{H}$. As shown in Fig. 5(a) the row parity is XOR'd with each check node input to calculate the value that all other variables in the group imply each individual variable node should take. Along with the parity determination, an implied reliability of the parity in the Log-Likelihood domain is calculated as shown in Fig. 5(b). The reliability update is performed on the log of the incoming log-likelihoods so that the calculation can be performed using additions. At the output the result is converted back into the log-likelihood domain. This arithmetic conversion is similar to the log-MAP implementation of MAP decoders [11]. The approximate logarithm is performed by a few gates of combinatorial logic. The exponentiation is approximated as a leading zeros count.

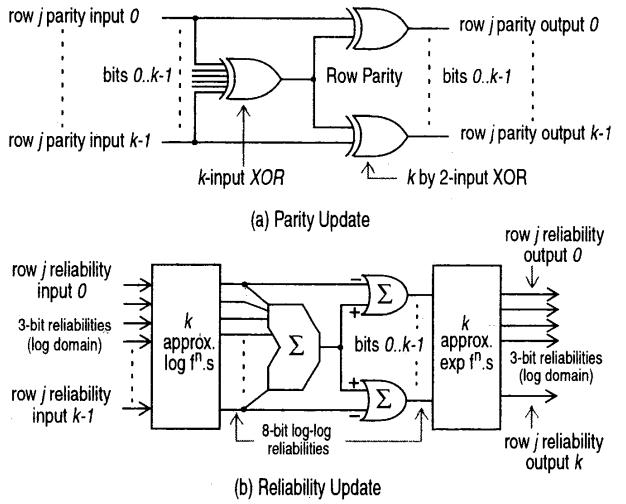

Fig. 5. Check node implementation

C. Variable Node Architecture

The architecture of the variable nodes is shown in Fig. 6. The variable nodes contain all of the latches in the decoder, both the scan-chain latches and decoder message latches. At the packet start signal the decoding of a new packet is commenced and the previous packets results are loaded into the output scan chain. For the first decoding iteration the messages sent to the check nodes are the signed magnitude representations of the received value, since for a Gaussian channel the received values are the log-likelihoods. All messages passed between the variable and check nodes are represented as a sign bit and 3 magnitude bits. For subsequent iterations, each message entering the variable node together with the received value are converted to 2's complement and summed. The sign of the sum represents the decoders current estimate of the decoded bit at each variable node. Outgoing messages are then formed as the group sum minus each individual edge's input message. This is the value all other connected checks and the received value imply each check should use for the next parity update. All values are converted back to a signed magnitude representation and latched, to be used by the check nodes connected to the variable node in the next decoder iteration. In the case that the group sum is zero or the outgoing messages sum is zero, the sign-bit used is that of the received bit, as this is the most probable value for the decoded bit.

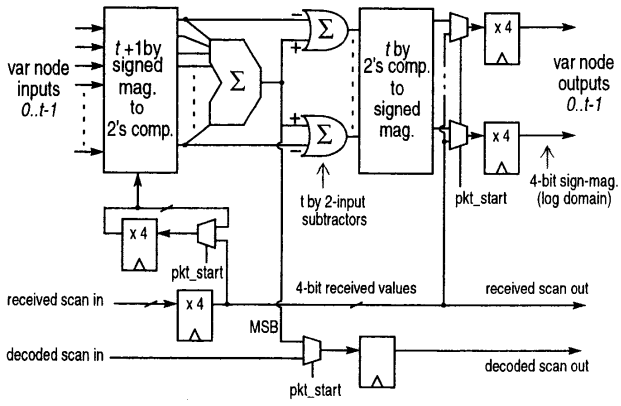

Fig. 6. Variable node implementation

D. Packet Error Detection

A packet error signal is derived by performing an OR operation of all row parities from the last decoder iteration, approximating a test of equation (1). The satisfaction of all parity checks in the final iteration is not equivalent to testing the parity using the final decoded bits but the difference is insignificant.

E. Physical Design

The decoder is being fabricated in a 0.16 µm, 1.5V CMOS process with 5 levels of metal. The variable and check nodes were synthesized from a VHDL description. As discussed in Section *A* the 1024 variable nodes were grouped into 16 macros, denoted *vgrp0* to *vgrp15,* to simplify the chip I/O and clock distribution. Macros were also created for the weight 6 and weight 7 check nodes. The main implementation challenge was the placement and routing of the macros at the top level with more than 26000 wires of average length 3mm representing the messages. Custom algorithms were developed to

place the macros and insert buffers to reduce the route lengths, reduce routing congestion, and achieve timing closure. The decoder was designed for a maximum clock frequency of 64MHz under worst case slow operating conditions. The layout of the decoder is shown in Fig. 7. The chip area is 7.5mm × 7.0mm and the utilization of 50% was limited by routing congestion.

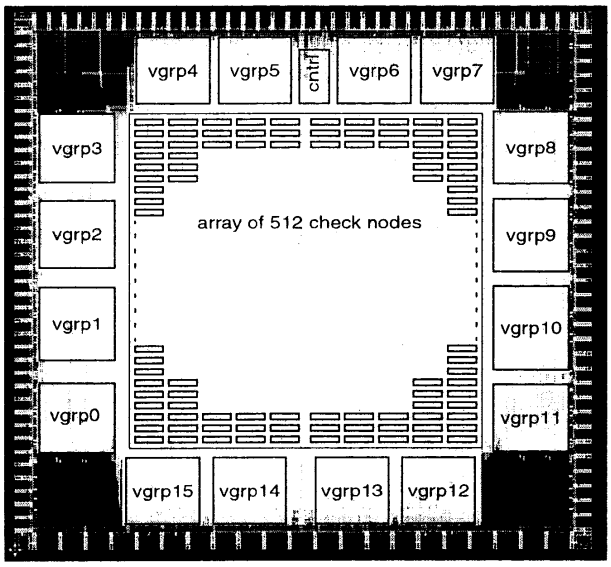

Fig. 7. Layout of the parallel 1024-bit, rate-1/2, soft decision LDPC decoder.

F. Switching Activity and Power Dissipation

It is shown in Fig. 8 that the decoder rapidly converges to a stable solution, particularly when the input bit error rate (BER) is low.

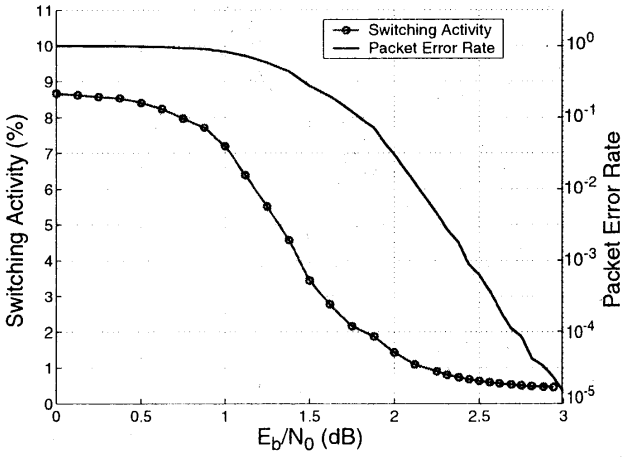

Fig. 8. Switching activity of message bits in a 1024-bit, R=1/2 soft decision LDPC decoder with 4-bit messages.

Once the decoder has converged to a stable result the only switching activity in subsequent iterations is that of the scan chains and clock. At this time the messages between nodes stop switching, reducing the power consumption of the decoder. From Fig. 8 for a packet error rate (PER) of 1% the switching activity factor is found to be only 1%, and for 100% PER the activity factor is 9%. Together with the extracted parasitics, these operating points can be used to estimate the average and worst case power dissipation of the decoder respectively. At a maximum clock frequency of 64MHz and 1Gb/s throughput the average total power dissipation of the decoder is calculated to be 220mW with a worst case value of 500mW. However, if a throughput of only 1Mb/s is required, e.g. 3G wireless data rate, the clock frequency is 64KHz and the power dissipation is a miserly 220μW average and 500μW worst case. This compares extremely favorably with 170mW reported for a turbo decoder performing 3 iterations with a block size of 256 bits and throughput of 1Mbit/s implemented in 0.6μm, 3.3V CMOS technology [4].

Conclusion

A parallel architecture for decoding LDPC codes has been presented that achieves comparable coding performance to equivalent turbo codes. The spectacular throughput and power dissipation advantages of the parallel architecture have been demonstrated through the implementation of a 1024 bit, rate-1/2 LDPC decoder that achieves power dissipation of 220mW and for a throughput of 1Gb/s , and 220μW for a throughput of 1Mb/s . The design represents a unique trade-off of area for high throughput and low power dissipation.

References

[1] M. Bossert, *Channel Coding for Telecommunications*, John Wiley & Sons, 1999.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting codes and decoding", *Proc. Int. Conf. Comm. '93*, pp 1064-1070, May 1993.

[3] R. Pyndiah, A. Glavieux, A. Picart, and S. Jacq, "Near optimum decoding of product codes", *Proc. IEEE GLOBECOM '94*, pp 339-343, 1994.

[4] S. Hong and W. Stark, "Design and implementation of a low complexity VLSI turbo-code decoder architecture for low energy mobile wireless communications", *J. VLSI Sig. Proc.*, Vol. 24, pp 43-57, 2000.

[5] R. Gallager, "Low density parity check codes", *IRE Trans. Info. Theory*, Vol. IT-8, pp 21-28, Jan. 1962.

[6] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of provably good low-density parity check codes", submitted to *IEEE Trans. Inf. Theory*, March 1999.

[7] F.R. Kschischang and B.J. Frey, "Iterative Decoding of Compound Codes by Probability Propagation in Graphical Models", *IEEE Journal on Selected Areas in Communications*, Vol. 16. No. 2, pp 219-30, Feb. 2000.

[8] C.J. Howland and A.J. Blanksby, "Parallel Decoding Architectures for Low Density Parity Check Codes", *Proc. IEEE ISCAS 2001*, 2001.

[9] "*3rd Generation Partnership Project (3GPP); Technical Specification Group Radio Access Network Multiplexing and channel coding (TDD)*", available at http://www.3gpp.org

[10] J. Hagenauer and P. Hoeher, " A Viterbi algorithm with soft-decision outputs and its applications", *Proc. GLOBECOM 1989*, pp 1680-1686, 1989.

[11] L.R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", *IEEE Trans on Inf. Theory*, pp 363-77, March 1974.