A 1.1-Gb/s 4092-bit Low-Density Parity-Check Decoder

Engling Yeo

Read Channel Architecture
STMicroelectronics, San Diego, CA, USA

Borivoje Nikolić

Dept. of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA, USA

Abstract—**A 4092-bit low-density parity-check decoder, based on staggered decoding schedule, is implemented in a 130nm 6M CMOS technology. The rate 0.75 code is based on finite-field geometries. Serial, shift-register based architecture enables a compact decoder implementation. The chip has a 4.0mm^2 core and operates at 1.1 GHz with 1.2V supply, resulting in a throughput of 1.1Gb/s per iteration.**

I. INTRODUCTION

Low-density parity-check (LDPC) codes have been recently proposed for use in many wireless and wireline communications systems as well as in the data storage. The use of LDPC codes in forward-error correction allows the operation near the Shannon capacity, thus enabling increased data throughputs, data densities or longer communications range. Encoding of a LDPC code is linear and usually not challenging, but the decoder design can be complex. LDPC codes are commonly decoded by applying the message-passing algorithm [1]. The throughput demands of modern communications and storage systems range from several tens of Mb/s up to several Gb/s, with block sizes ranging from several hundreds to several tens of thousands of bits.

II. LDPC CODE CONSTRUCTION

LDPC codes comprise a number of parity checks applied sparsely to a block of user data. Although a random assignment of parity checks often yields better performance, regular code constructions can guarantee certain error correcting performance while providing significant architectural advantages for the decoder implementation. This work implements a decoder for a 4092-bit LDPC code with code rate 0.75 and row weight 32. High-rate codes commonly lead to faster decoder convergence and consequently require far less number of iterations than low-rate codes. The design of the particular code used in this work is based on two-dimensional projections in finite field geometries [6]. These construction methods eliminate short cycles in the graph associated with this code, which are known to degrade the error-correction performance. Starting from a two-dimensional space defined by a Galois field GF(2^5), a 1023x1023 LDPC matrix is constructed. Consecutive rows are cyclic shifts, though this matrix is not full-ranked. In order to increase the overall block size to 4092 bits, each column in the matrix is expanded into four consecutive columns in a process known as column splitting. The non-zero entries in each column of the initial matrix are rotated to form four columns in the new matrix. The resulting 1023x4092 matrix describes a LDPC code with 3070 user bits and 1022 parity bits [4].

III. LDPC DECODER ARCHITECTURES

Message-passing algorithm iteratively passes messages, computed as log domain probabilities, between two classes of nodes: the variable nodes and the constraint (parity) nodes. A fully-parallel LDPC decoder [3] can be organized as a large number of check node and variable node processing elements (PE) linked through an interconnect fabric which corresponds to the edge connectivity of the graph. The direct mapping of the network using hard-wired routes leads to congestion in the interconnect that can be alleviated only by reducing the logic density [2], [3], [4]. Using a fully parallel decoder for larger block sizes becomes impractical, despite its large throughput promise. For example, implementing a fully-parallel 4092-bit decoder in 130nm technology would require more than 130mm^2, which is prohibitively large for most communications applications.

In contrast, a serial architecture uses a small number of time-multiplexed computation blocks to evaluate all arithmetic. Interconnect wires are replaced with memories, which address the routing congestion, and results in a compact design, but with significant reduction in throughput.

The memory requirement of serial decoders applying the message-passing decoding algorithm is proportional to the total number of edges in the underlying graph [3]. For example, each of the 16384 edges in this particular LDPC code relays a 5-bit message, resulting in the total memory requirement of more than 10kB. The necessity to realize 32 parallel and independent accesses to the memory would severely limit the throughput.

The LDPC code construction with well-defined structure allows for architectural tradeoffs between the decoder size and its throughput. Such codes permit the use of multiple parallel processing elements, which then communicate with independent memory blocks for increased throughput. An example of this approach is the 64,800-bit DVB-S2-compliant LDPC decoder with 360 arithmetic elements operating in parallel [5].

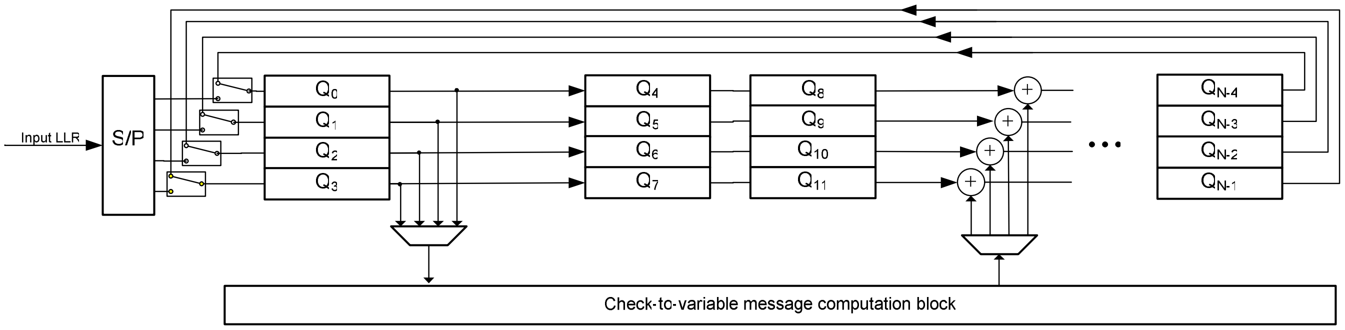


Figure 1. Shift register-based implementation of LDPC code generated from 2D GF(2M) with 1:4 column splitting and message computation latency of 2.

IV. Complexity Reduction

This work implements a serial decoder architecture and uses a number of complexity reduction techniques to decrease the overall memory and computational requirements. Low average number of iterations associated with the high code rate lowers the throughput penalty of serial decoding.

After the 1:4 column-splitting, each non-zero value in the original parity-check matrix is transformed into a permutation of the [1 0 0 0] vector; therefore, each check node is connected to a maximum of one out of every four consecutive variable nodes. Consequently, the variable-to-check messages from four consecutive variable nodes, Q'_n for $n = 4k,\ 4k+1,\ 4k+2,\ 4k+3$, can be distributed amongst four parallel shift register chains. At 32 different positions along the shift register chains, the appropriate register contents are multiplexed into the message computation block, which evaluates the corresponding check-to-variable messages. The 32 locations can be determined from the non-zero entries of the first row in the original 1023×1023 matrix. The outputs from the message computation block are, likewise, updated to the register contents via corresponding 32-input demultiplexers.

This decoder uses a stopping criteria based on the number of satisfied parity constraints. A simple parity checker stops any further iteration once the number of failed parity constraints falls below a preset threshold.

The use of a staggered decoding schedule [4] not only lowers the overall memory requirement, but also leads to increased convergence rate. In the original, concurrent decoding schedule [1], all of the variable nodes are updated once per iteration, followed by the complete update of all constraint nodes. The staggered decoding schedule, however, processes only a limited subset of the check nodes per cycle. While the check node operations remain unchanged, each variable node n computes altered messages Q_{nm} according to (1).

$$Q_{nm} \approx \ln\left[\frac{p_i(1)}{p_i(0)}\right] + \left(\sum_{m'\in\mu(n)} R_{m'n}\right) \tag{1}$$

Each variable node, n, is therefore, associated only with a single message $Q'_n(k)$, which is broadcasted to the subset of active check nodes, $S(k)$, at step k. Each message, $Q'_n(0)$ for $n = 1, 2, \ldots, N$ is initialized with the input LLR of the corresponding variable n, and updated according to (2) at the end of each step. An implementation of the staggered schedule only needs to store one message, $Q_n(k)$, for each variable n. This provides more than 87% savings in memory.

$$Q'_n(k) = Q'_n(k-1) + \sum_{m'\in\{S(t)\cap\mu(n)\}} R_{m'n} \tag{2}$$

The overall decoder architecture is shown in Fig. 1. This serial LDPC decoder interfaces with the inner receiver using 5-bit input/output messages that represent the log-likelihood ratios of the received bits. It comprises of a single message computation block for calculation of check-to-variable messages, and 32 computation blocks for variable-to-check messages. These messages are stored in four shift-register chains, which are also connected through a feedback loop that permits repeated iterations of decoding. The decoder outputs 32 check-to-variable messages per clock cycle. Another 32 computation blocks are producing the variable-to-check messages. These ensure that data dependencies are fulfilled and prevent the processing pipeline from stalling.

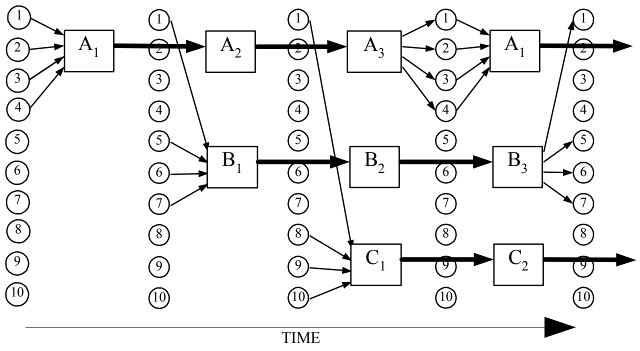
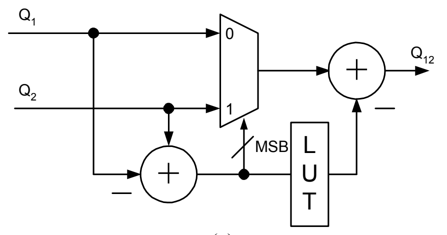
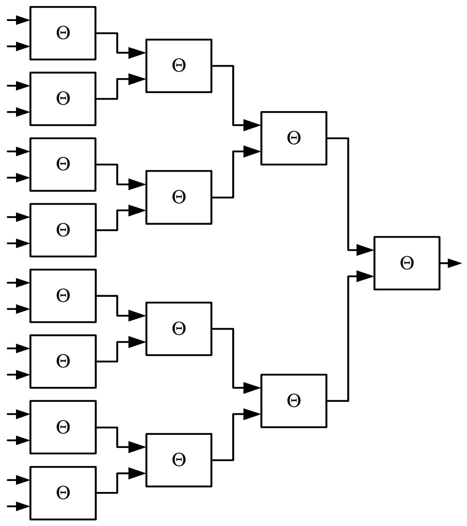


Figure 2. Staggered decoding schedule using one processing element with a 3-stage pipeline.



(a)



(b)

Figure 3. The (a) Θ operator and the (b) tree structure of 2-input Θ operators to evaluate check-to-variable messages.

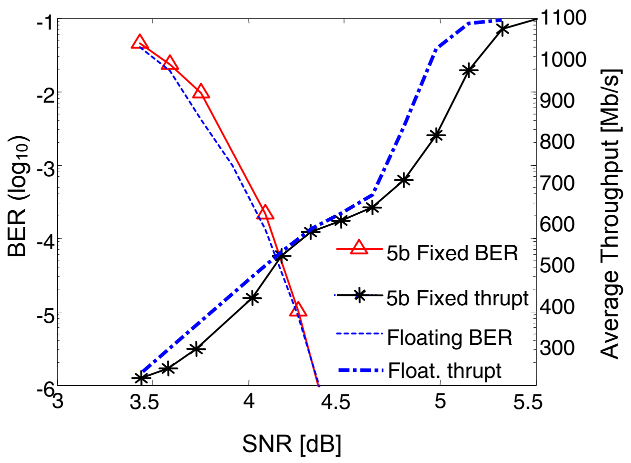


Figure 4. Simulation results with a 4092-bit, rate 0.75, LDPC code based on finite-field construction, with the Θ approximation.

The staggered decoding is combined with a 22-stage pipelined check-to-variable message computation, to achieve the desired throughput. Fig. 2 shows the update schedule of the staggered decoding combined with a simplified example of a 3-stage pipeline. To make the figure clearer, the variable and check nodes are labeled with letters and numbers. *A*, *B*, and *C* represent three check nodes. A_i refers to the i^{th} stage of the pipeline in the check node processing corresponding to check node *A*. The nodes 1-10 represent the variable nodes. Each node, *k*, $1 \le k \le 10$, is associated with a message Q'_k. Since the value of Q'_1 is not updated until time $3T$, the same message is sent from node 1 to A_1, B_1, and C_1 in the first three cycles. In contrast, standard concurrent decoding would require update of all messages in each cycle of iteration.

In addition, the standard check-to-variable message computation requires the evaluation of (3). Quantization effects of fixed-point implementations are exacerbated by the nonlinear nature of the $\Phi(\cdot)$ function. As an alternative, we compute the check-to-variable messages, R_{mn}, with a set of reduced-complexity operations (4) using a correction term to counteract the effects of fixed-point implementations.

$$\Phi(x) = -\log(\tanh(x/2)) = \Phi^{-1}(x); \qquad x \ge 0 \tag{3}$$

$$R_{mn} = \underset{n' \in N(m)\setminus n}{Min} Q_{n'm} + g(Q_{n'm} \quad \forall \quad n' \in N(m) \setminus n) \tag{4}$$

The operations are implemented with a binary tree of Θ operators (5) as shown in Fig. 3.

$$\Theta(x,y) = \Phi^{-1}[\Phi(x) + \Phi(y)] \approx Min\{x,y\} + f(x,y) \tag{5}$$

Each Θ operator inputs a pair of $Q_{n'm}$ and outputs the minimum value corrected by a function, $f(\cdot)$. This function is realized using a two-dimensional lookup table that outputs two fractional bits; their values range from 0.75 (when the two input values are equal, to 0 as their difference increases. The fixed-point behavior of the decoder system is shown in Fig. 4. The BER performance with 5-bit messages suffers less than 0.1dB degradation compared to the floating-point system simulations.

V. Prototype Implementation

A test chip (Fig. 5) has been fabricated in 1.2V, 130nm 6-metal general-purpose CMOS technology. There are 256k gates with 1.8 million transistors in the chip. The area of the 8.6mm^2 chip is pad-limited by 103 pads; the core measures only 4.0 mm^2. The physical design was performed using a semi-custom design flow for direct mapping of the datapath description into silicon implementation [7], [8]. This flow was further enhanced for high-speed implementation through customization of the clock tree to achieve low clock skews. The chip test features include a functional built-in self-test (BIST) and a full register scan. An on-chip clock generator along with the BIST is used to evaluate the operation of the chip. The chip is fully functional, operating at 1.16GHz at 1.2V supply. Measured maximum power dissipation is 1.5W.

The maximum throughput of the decoder is 1.16Gb/s/iteration. The average number of iterations decreases with increasing operating SNR, thus affecting the effective throughput. This is shown in Fig. 4. With supply voltage scaled to 1V, the clock frequency is reduced to 900MHz and the power dissipation is 950mW, Fig 6. The chip performance is summarized in Table 1.

The throughput of this architecture can be increased by adding three more check-to-variable computational blocks to process the four independent data streams available through this code construction, similarly to the idea from [5]. One possible option for reducing the power consumption is to replace the library flip-flop-based shift register with a custom register file.

VI. CONCLUSION

A compact, 4092-bit serial LDPC decoder has been verified to decode at 1.16Gb/s, using a modified decoding schedule. Serial decoder design provides a more compact alternative with no routing congestion that larger parallel designs suffer. Consequently, the core area measures 4.0 mm^2. To achieve an even smaller implementation and further reduction in power dissipation, scanable library registers can be replaced with custom FIFO buffers. The column splitting used in the code construction provides an opportunity for parallelization in future decoder implementations.

ACKNOWLEGMENT

The authors thank P. Pakzad and V. Anantharam of UC Berkeley for their technical advice. This work was performed at UC Berkeley and supported by Marvell and STMicroelectronics under a UC Micro program. STMicroelectronics fabricated the test chip.

REFERENCES

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

[2] E. Yeo, B. Nikolic, and V. Anatharam, “Iterative decoder architectures,” *IEEE Comm. Magazine*, pp. 132-140, Aug 2003.

[3] A. Blanksby and C. J. Howland, “A 690mW 1-Gbit/s 1024-bit rate-1/2 low density parity check code decoder,” *IEEE J. Solid-State Circuits*, pp. 404-412, Mar 2002.

[4] E. Yeo, B. Nikolic, and V. Anantharam, "High throughput low-density parity-check architectures," *Proc. IEEE Globecom 2001*, San Antonio, TX, Nov 25-29, 2001. pp. 3019-3024.

[5] P. Urard, *et al*, “A 135/Mb/s DVB-S2 compliant codec based on 64800b LDPC and BCH codes,” *Proc. ISSCC’05*, San Francisco, CA February 2005, pp. 446-447, 609.

[6] Y. Kou, S Lin, and M. P. C. Fossorier, “Low-density parity-check codes based on finite geometries: A rediscovery and new results,” *IEEE Trans. Info. Theory*, pp. 2711–2736, Nov. 2001..

[7] W.R. Davis, *et al*, "A Design Environment for High Throughput, Low Power Dedicated Signal Processing Systems," *IEEE J. Solid-State Circuits*, 420-431, March 2002.

[8] K. Kuusilinna, *et al*, "Real Time System-on-a-Chip Emulation," in *Winning the SoC Revolution* by H. Chang, G. Martin, Norwell, MA: Kluwer Academic Publishers, 2003. pp. 229-253.

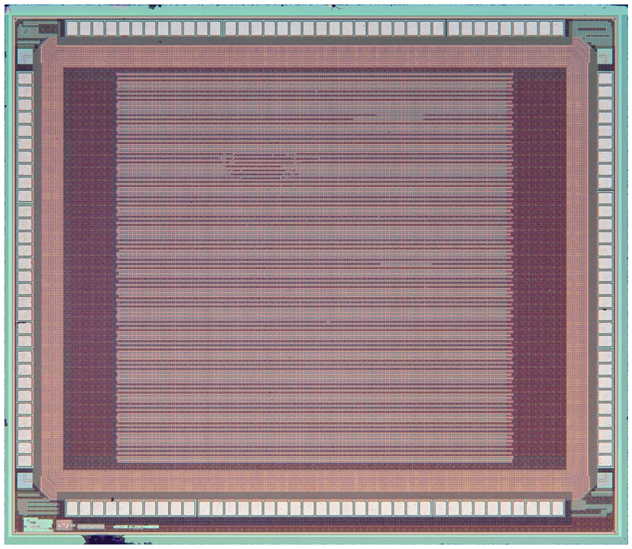


Figure 5. Micrograph of LDPC decoder test chip in 130nm CMOS, occupying 3.2mm × 2.7mm.

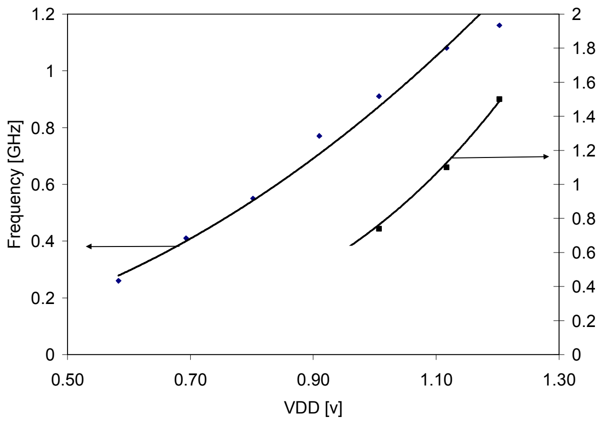


Figure 6. Measured frequency and power of the decoder chip.

TABLE I. SUMMARY OF LDPC DECODER

Design Parameter	Specification
Block size	4092 bits
Code rate	0.75
Technology	1.2V, 0.13µm CMOS with 6 metal layers
Transistor count	1.8M (256k gates)
Gate count	256,000
Core Area	4.0 mm^2
Clock frequency	1.16GHz @ 1.2V
Power	1.5W @ 1.2V