A Fully Parallel Nonbinary LDPC Decoder With Fine-Grained Dynamic Clock Gating

Youn Sung Park, *Member, IEEE*, Yaoyu Tao, *Student Member, IEEE*, and Zhengya Zhang, *Member, IEEE*

***Abstract*—Nonbinary LDPC (NB-LDPC) codes, defined over Galois field, offer better coding gain and a lower error floor than binary LDPC codes. However, the complex decoding and large memory requirement have prevented any practical chip implementations. We present a 1.22 Gb/s fully parallel decoder of a GF(64) (160, 80) regular-(2, 4) NB-LDPC code in 65 nm CMOS. The reduced number of edges in NB-LDPC code's factor graph permits a low wiring overhead in the fully parallel architecture. The throughput is further improved by a one-step look-ahead check node design that increases the clock frequency to 700 MHz, and the interleaving of variable node and check node operations that shortens one decoding iteration to 47 clock cycles. We allow each processing node to detect its own convergence and apply dynamic clock gating to save power. When all processing nodes have been clock gated, the decoder terminates and continues with the next input to increase the throughput to 1.22 Gb/s. The dynamic clock gating and decoder termination improve the energy efficiency to 3.03 nJ/b, or 259 pJ/b/iteration, at 1.0 V and 700 MHz. Voltage scaling to 675 mV improves the energy efficiency to 89 pJ/b/iteration for a throughput of 698 Mb/s at 400 MHz.**

***Index Terms*—Dynamic clock gating, LDPC code, LDPC decoder architecture, nonbinary LDPC code.**

I. Introduction

STATE-OF-THE-ART communication and storage systems have adopted sophisticated channel codes to achieve a higher reliability in transmission and storage at the lowest signal-to-noise ratio (SNR). To close the gap towards the ultimate channel capacity, known as the Shannon limit, binary LDPC and Turbo codes [1]–[3] have been adopted by the latest standards [4]–[10] and numerous architecture implementations have been demonstrated [11]–[16]. Nonbinary LDPC (NB-LDPC) codes, defined over Galois field GF(q), where $q > 2$, offers better coding gain than binary LDPC codes [17]. NB-LDPC codes' excellent coding gain can be achieved even at a short block length, and a low error floor has also been demonstrated.

The decoding of NB-LDPC codes follows the same belief propagation (BP) algorithm [17] that is used in the decoding of binary LDPC codes. However, the complexity of an NB-LDPC decoder is notably higher: each message exchanged between processing nodes in an NB-LDPC decoder carries an array of log-likelihood ratios (LLR); parity check processing follows a forward-backward algorithm; and high-order GF operations require expensive matching and sorting, in contrast to the much simpler addition and compare-select used in binary LDPC decoding. The complex NB-LDPC decoding has prevented any large-scale high-throughput chip implementations in silicon. Only FPGA designs, synthesis and layout have been demonstrated prior to this work [18]–[30].

The complexity of the NB-LDPC decoder and its error-correcting performance are determined by code construction. Quasi-cyclic LDPC codes have been invented to provide a good error-correcting performance [31]–[33], and their regular structures are amenable to efficient decoder architectures. Compared to the quasi-cyclic LDPC codes, the $(2, d_c)$ codes feature a very low variable node degree $d_v = 2$, and a check node degree as low as $d_c = 4$, reducing the processing complexity, the wiring, and the quantization loss. Therefore, the $(2, d_c)$ codes are attractive for practical implementations. A $(2, d_c)$ NB-LDPC code offers a competitive error-correcting performance even at a short block length. The performance can be further improved by increasing q, the order of the GF field, but a higher q increases the size and complexity of the decoder.

The direct implementation of the BP decoding for NB-LDPC codes results in a check node complexity of $O(q^2)$ and a variable node complexity of $O(q)$. A fast Fourier transform (FFT) implementation [34] reduces the check node complexity to $O(q \log q)$, but it requires check node processing in the linear domain and the conversion between linear- and log-domain messages. The extended min-sum (EMS) algorithm [35] in the log domain reduces the check node complexity to $O(qn_m)$ using only a small subset of n_m values among an array of q LLRs in a message, where $n_m \ll q$. A further simplification of the EMS algorithm truncates the least significant values in a message and keeps only the most significant n_m values in memory [36]. The processing is done entirely using truncated messages, thereby reducing the complexity of the check node to $O(n_m \log n_m)$ and the complexity of the variable node to $O(n_m)$. The truncated EMS algorithm has demonstrated minimal loss in error-correcting performance at low SNR compared with BP, while the performance surpasses BP at high SNR [36]. The truncated EMS algorithm makes it possible to design an NB-LDPC decoder with a reasonable complexity that is within the range of binary LDPC decoders. A further simplification using the min-max algorithm [37] suffers from a noticeable degradation in the error-correcting performance.

Manuscript received February 17, 2014; revised May 14, 2014, and August 12, 2014; accepted September 15, 2014. Date of publication November 13, 2014; date of current version January 26, 2015. This paper was approved by Associate Editor Stefan Rusu. This work was supported in part by NSF CCF-1054270 and the Broadcom Foundation. Chip fabrication was donated by STMicroelectronics.

The authors are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122 USA (e-mail: parkyoun@umich.edu; taoyaoyu@umich.edu; zhengya@eecs.umich.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2014.2362854

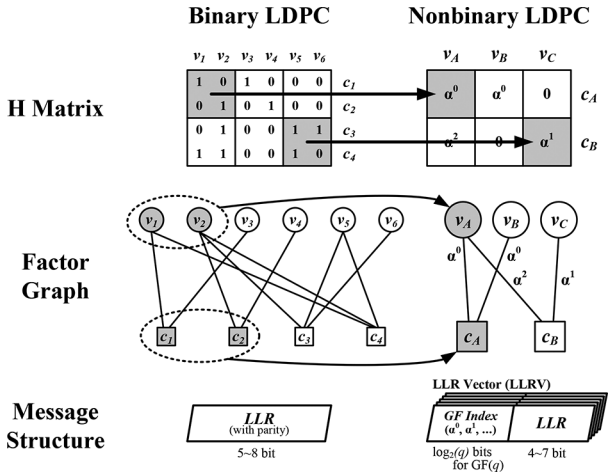


Fig. 1. Comparison of binary and nonbinary LDPC (NB-LDPC) code.

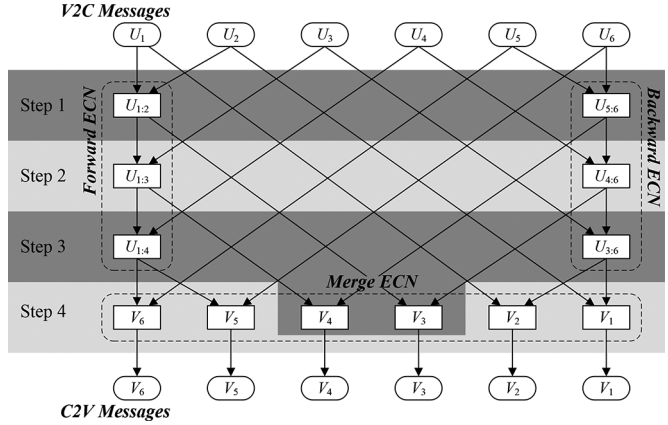


Fig. 2. Illustration of forward-backward algorithm with $d_c = 6$.

In this paper, we present a 7.04 mm^2 65 nm CMOS NB-LDPC decoder chip for a GF(64) (160, 80) regular-(2, 4) code using the truncated EMS algorithm. We use a fully parallel architecture and scheduling techniques to enhance the throughput to 1.22 Gb/s at 700 MHz. To reduce the power consumption, we design a fine-grained dynamic clock gating based on node-level convergence detection to save 50% power.

II. Background

An NB-LDPC code is formed by grouping bits to symbols using GF elements, an example of which is shown in Fig. 1. In the example, two bits are grouped to a 2-bit symbol using GF(4). In the binary LDPC H matrix shown in Fig. 1, 2×2 submatrices are replaced by GF(4) elements, resulting in a GF(4) nonbinary H matrix. A regular-(d_v, d_c) NB-LDPC code has a constant column weight of d_v and a constant row weight of d_c. An NB-LDPC code can also be illustrated using a factor graph of variable nodes (VN) and check nodes (CN). An edge connects VN v_j and CN c_i if the corresponding entry in the H matrix $H(i,j) \neq 0$.

A. Truncated EMS Decoding Algorithm

An NB-LDPC code is decoded by iteratively passing messages between VNs and CNs over the factor graph. The VN to CN message will be referred to as the V2C message, or $U_{j,i}$ (from v_j to c_i); and the CN to VN message as the C2V message, or $V_{i,j}$ (from c_i to v_j).

1) VN Initialization: The decoding starts by initializing each VN with the prior LLRs based on the information received from the communication channel. Because each VN in an NB-LDPC code represents a GF(q) element, the prior LLR for a VN v_j, L_j, is an LLR vector (LLRV) of length q, and each element of the LLRV corresponds to a GF(q) element α_k, $k \in \{1, \ldots, q\}$

$$L_j = [L_j(1), L_j(2), \ldots, L_j(q)]$$

where

$$L_j(k) = \log \frac{P(v_j = \hat{\alpha}|y)}{P(v_j = \alpha_k|y)}$$

and

$$\hat{\alpha} = \{\arg\max_{\alpha_k \in \mathrm{GF}(q)} P(v_j = \alpha_k|y)\} \quad (1)$$

and y is the channel information. $\hat{\alpha}$ is the GF element with the maximum likelihood. Based on this definition, a lower LLR value indicates a higher likelihood. In the following, we assume that the LLRV is sorted in ascending order unless specified otherwise. A GF index vector is associated with each LLRV to keep track of the GF elements that correspond to the entries of the LLRV. In the GF index vector, each GF(q) element is stored in its $\log_2 q$-bit binary representation. Using the truncated EMS algorithm, only the minimum n_m entries, $n_m < q$, in the LLRV are kept. In the first decoding iteration, the prior LLRV is used as the V2C message, i.e., $U_{j,i} = L_j$.

2) CN Operation: Each GF element α_k in the GF index vector of the V2C message $U_{j,i}$ is multiplied by $H(i,j)$ before the message is sent to the CN. α_k is stored in the binary representation and $H(i,j)$ is known, so the GF(q) multiplication is described by a q-entry lookup table. This GF multiplication is known as permutation.

Suppose a CN receives messages from d_c VNs, v_j, $j \in \{1, 2, \ldots, d_c\}$, where d_c is the degree of the CN. The CN computes the C2V messages for each VN using the forward-backward algorithm in three steps: forward, backward, and merge that are illustrated in Fig. 2. Forward and backward can be carried out in parallel.

As Fig. 2 shows, in the forward step, the message from v_1 is combined with the message from v_2, and message combining continues until reaching v_{d_c-2}, following (2a). The "combine" operation is known as the elementary CN (ECN) that is represented by $\oplus$ in (2a). An ECN takes two length-n_m LLRV inputs, e.g., U_1 and U_2, and calculates a length-n_m LLRV output $U_{1:2}$ that contains the n_m minimum values in the set $\{U_1(i) + U_2(j), i \in [1, n_m], j \in [1, n_m]\}$. An ECN is made using an insertion sorter of length n_m, and the complexity of the ECN is $O(n_m^2)$. An efficient bubble check algorithm [38] reduces the insertion sorter length to $\lceil (1 + \sqrt{1 + 8(n_m - 1)})/(2) \rceil$ and the operation complexity to $O(n_m \sqrt{n_m})$. The forward step requires$d_c - 3$ ECNs in total.

$$\text{Forward} : U_{1:j+1} = U_{1:j} \oplus U_{j+1}, \quad j = 1, \ldots, d_c - 3 \ (U_{1:1} = U_1). \quad (2a)$$

$$\text{Backward} : U_{j-1:d_c} = U_{j:d_c} \oplus U_{j-1}, \quad j = d_c, \ldots, 4 \ (U_{d_c:d_c} = U_{d_c}). \quad (2b)$$

$$\text{Merge}: V_j = U_{1:j-1} \oplus U_{j+1:d_c}, \quad j = 2, \ldots, d_c - 1. \quad \text{(2c)}$$

The backward step follows (2b), and it is identical to the forward step, except that it is done in the reverse direction, as shown in Fig. 2. After the forward and backward are complete, the C2V messages can be readily calculated by merging the messages obtained from the forward and backward, as described by (2c) and illustrated in Fig. 2. Merge requires d_c ECNs. To sum up, the forward-backward algorithm for CN requires $3d_c - 6$ ECNs in total, and each ECN is of complexity $O(n_m\sqrt{n_m})$.

Afer the C2V messages are calculated, each GF element α_k in the GF index vector of the C2V message $V_{i,j}$ is divided by $H(i,j)$ before the message is sent to the VN. α_k is stored in the binary representation and $H(i,j)$ is known, so the GF(q) division is described by a q-entry lookup table. This GF division is known as inverse permutation.

3) VN Operation: Each VN receives d_v C2V messages and computes the posterior LLR, L_j^{post}, and the V2C messages following (3). Note that the operator $+$ and $\sum$ are not ordinary addition and summation. They represent pair-wise elementary VN (EVN). An EVN takes two length-n_m LLRV inputs, V_1 and V_2, and calculates a length-n_m LLRV output V_3 that contains the n_m minimum values in the set $\{V_1(i) + V_2(j), V_1^{gf}(i) = V_2^{gf}(j), i \in [1, n_m], j \in [1, n_m]\}$. An EVN requires matching of GF index, which is done using a content-addressable memory (CAM). An EVN uses an insertion sorter of length n_m, and the complexity of the EVN is $O(2n_m)$.

$$L_j^{\text{post}} = L_j + \sum_{i'=1}^{d_v} V_{i',j}, \quad U_{j,i} = L_j + \sum_{i'=1, i' \neq i}^{d_v} V_{i',j}. \quad \text{(3)}$$

VN makes a hard decision in each iteration based on the most likely GF element. If the hard decisions of all VNs meet all parity checks defined by the H matrix, decoding terminates.

B. Decoder Design Challenges

Compared to a binary LDPC code, the factor graph of an NB-LDPC code is more compact with fewer nodes and much fewer edges, suggesting a simpler wiring in its decoder implementation. However, grouping $\log_2 q$ binary bits to a GF(q) symbol expands the message memory from $\log_2 q$ words to q words. The truncated EMS algorithm reduces the message memory to n_m ($n_m < q$) words, e.g., a GF(64) NB-LDPC code can be decoded using $n_m = 16$, requiring 16 words in message storage, but still higher than what is needed in a binary LDPC decoder.

The VN and CN operations in an NB-LDPC decoder as described above are more complex than a binary LDPC decoder. The CN of a binary LDPC decoder performs compare select and XOR in a tree structure of complexity $O(d_c)$, thus the CN can be easily parallelized for a high throughput. The CN of an NB-LDPC decoder performs forward, backward and merge with a complexity of $O(d_c n_m \sqrt{n_m})$ using the truncated EMS algorithm with bubble check ECN. The VN of an NB-LDPC decoder is also more complex than the VN of a binary LDPC decoder, with a complexity of $O(d_v n_m)$ compared to $O(d_v)$. For practical implementations of NB-LDPC decoders, the CN and VN operations have to be serialized, resulting in a lower throughput. The larger memory, expensive sorters and CAMs all contribute to larger VNs and CNs.

III. High-Throughput Fully Parallel Decoder Architecture

The NB-LDPC decoder is heavy on logic and memory but low on wiring compared to the binary LDPC decoder. A parallel implementation of NB-LDPC decoder does not incur the same wiring overhead seen in the implementations of binary LDPC decoder. A fully parallel implementation also simplifies the control and message scheduling, leading to a more efficient design.

The GF(64) (160, 80) regular-(2, 4) NB-LDPC code constructed based on the algebraic properties of their binary images features low VN and CN degrees, thus the complexity of VN and CN can be kept low. The block diagram of the fully parallel decoder is illustrated in Fig. 3. The 960 bits of a codeword are grouped into 160 6-bit GF(64) symbols. The factor graph of the code contains 160 VNs and 80 CNs. The fully parallel decoder is the direct mapping of the factor graph with 160 2-input VNs and 80 4-input CNs as shown in Fig. 3. Each edge in the factor graph carries an LLRV. The entries of the LLRV are sent serially to reduce the bit width of the wires and to match the pipelined CN and VN processing. Permutation and inverse permutation are placed between the VNs and CNs, and messages are normalized in each iteration to prevent saturation. The messages in this design are quantized to 5 bits to ensure a good performance while minimizing storage. The decoder implements the truncated EMS algorithm with $n_m = 16$. The word length and truncated EMS setting have been simulated extensively to ensure a good error-correcting performance down to very low BER levels.

We further improve the throughput of the fully parallel decoder using architecture transform and scheduling techniques: (1) by applying a one-step look-ahead to the ECN bubble check algorithm, we remove the data dependency to produce a fast ECN design; (2) by dividing the ECN and EVN schedules into two phases, we allow the interleaving of VN and CN for a short iteration latency.

A. Look-Ahead Elementary Check Node

CN takes 4 V2C messages, U_1, U_2, U_3, U_4, and computes 4 C2V messages, V_1, V_2, V_3, V_4, using the forward-backward algorithm illustrated in Fig. 2. The forward step takes U_1 and U_2 to compute $U_{1:2}$; and concurrently, the backward step takes U_4 and U_3 to compute $U_{3:4}$. Next, the four merges are done in parallel to compute V2C messages, as illustrated in Fig. 4. The forward step, backward step, and merge are all done using ECN.

ECN implements the bubble check algorithm to find the n_m minimum values in the set $T_\Sigma = \{U_1(i) + U_2(j), i \in [1, n_m], j \in [1, n_m]\}$, where U_1 and U_2 are two input LLRVs. The set T_Σ is represented in a 2-dimensional matrix. The entries of T_Σ are computed on the fly by reading one entry from $U_1(i)$ and one from $U_2(j)$ and summing them. The GF element that corresponds to the sum is computed by adding the GF element associated with $U_1(i)$ and the GF element associated with $U_2(j)$.

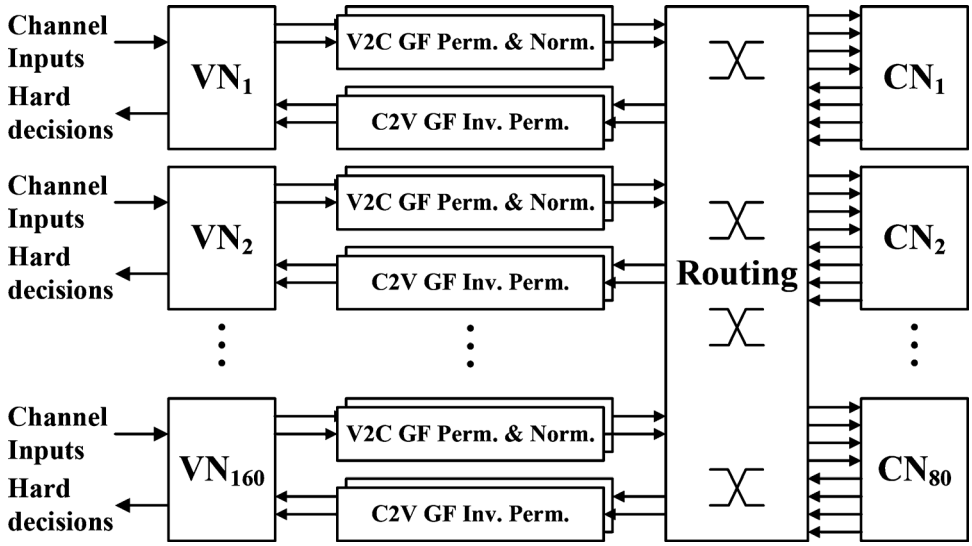


Fig. 3. Architecture of the fully parallel nonbinary LDPC decoder.

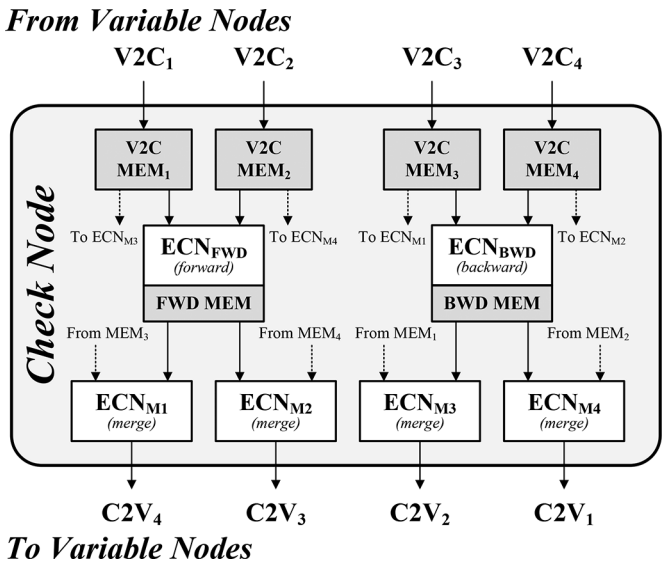


Fig. 4. Architecture of the check node.

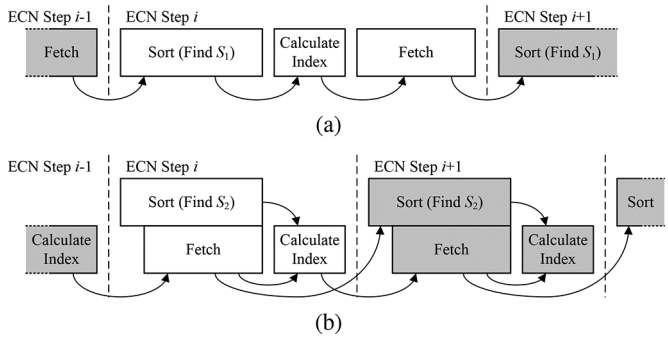


Fig. 5. Suboperation schedule of (a) the bubble check elementary check node and (b) the look-ahead elementary check node.

Since the pair of GF elements are stored in binary representation, the addition is done by the bitwise XOR of the pair. ECN uses an insertion sorter of length 6 for $n_m = 16$. The ECN sorter is initialized with $T_\Sigma(1,1)$, $T_\Sigma(2,1), \ldots T_\Sigma(6,1)$. The ECN sorter outputs the minimum entry, e.g., $T_\Sigma(i_1, j_1)$, every step and a new entry $T_\Sigma(i_n, j_n)$ is inserted. ECN is complete after n_m steps. Note that we allow duplicate GF outputs because it simplifies the control logic and ensures a constant latency per iteration. Our simulation results show that the loss in error-correcting performance due to duplicate GF outputs is negligible.

Using bubble check [38], the new entry from T_Σ to be inserted to the sorter is determined based on the minimum entry in the sorter. Each ECN step consists of three substeps as illustrated in Fig. 5(a): (1) sort: find the minimum entry in the sorter, $T_\Sigma(i_1, j_1)$; (2) bubble check: calculate the index of the new entry (i_n, j_n) in T_Σ to be inserted to the sorter based on the bubble check algorithm using a "horizontal flag" H described below [38]; and (3) fetch: read $U_1(i_n)$ and $U_2(j_n)$, calculate the sum and insert it to the sorter. Each substep depends on the previous one. The data dependency requires that the three substeps to be done in series, which results in a long clock period $T = t_{\text{sort}} + t_{\text{bubble}} + t_{\text{fetch}}$, where t_{sort}, t_{bubble}, and t_{fetch} are the maximum time needed for the sort, bubble check and fetch.

---

if $i_1 = 1$ **then**
  $H = 1,\ \bar{H} = 0$
end if
if $j_1 = 1$ **and** $i_1 \geq n_b$ **then**
  $H = 0,\ \bar{H} = 1$
end if
if $T_\Sigma(i_1 + \bar{H}, j_1 + H)$ has never been inserted to the sorter **then**
  $i_n = i_1 + \bar{H},\ j_n = j_1 + H$
else
  $i_n = i_1 + H,\ j_n = j_1 + \bar{H}$
end if

We apply one-step look-ahead to shorten the clock period. The new sorter keeps track of not only the minimum $T_\Sigma(i_1, j_1)$, but also the second minimum $T_\Sigma(i_2, j_2)$. With this change, each

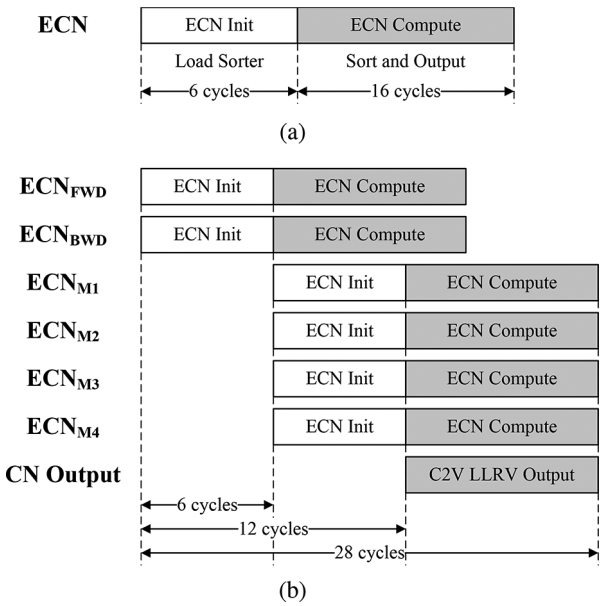


Fig. 6. Operation schedule of (a) the elementary check node and (b) the check node.

ECN step is done in three substeps that can be partially overlapped: (1) sort: find the second minimum $T_\Sigma(i_2, j_2)$ (the minimum $T_\Sigma(i_1, j_1)$ has been found in the previous ECN step); (2) fetch: read $U_1(i_n)$ and $U_2(j_n)$, calculate the sum and insert to the sorter; (3) bubble check: compare $T_\Sigma(i_2, j_2)$ with $T_\Sigma(i_n, j_n)$, one of which will be the new minimum $T_\Sigma(i_1, j_1)$ to be output next, and the index of the new entry (i_n, j_n) is calculated based on the bubble check algorithm above. Though the three substeps still remain, the look-ahead design allows sort and fetch to be done in parallel. The new sequence illustrated in Fig. 5(b) allows the overlapping of the substeps to shorten the clock period to $T = \max\{t_{\text{sort}}, t_{\text{fetch}}\} + t_{\text{bubble}}$. Since t_{sort},t_{fetch} are longer than t_{bubble}. The clock period is almost halved compared to the baseline version.

The schedule of the ECN is divided into two phases: initialization phase and compute phase, according to Fig. 6(a). The initialization phase spans the first 6 cycles to initialize the sorter. The compute phase spans $n_m = 16$ cycles, during which ECN outputs one value every cycle. In the CN schedule shown in Fig. 6(b), the forward and backward ECNs are started at the same time. After the initialization phase, the forward and backward ECNs $(\text{ECN}_{\text{FWD}}, \text{ECN}_{\text{BWD}})$ move to the compute phase, while the four merge ECNs $(\text{ECN}_{\text{M1}-4})$ start their initialization phase. The phase pipelining shortens the latency of the CN to 28 cycles.

B. Two-Pass Variable Node

VN takes 2 C2V messages, V_1, V_2, and the prior LLRV to compute 2 V2C messages, U_1, U_2, and the posterior LLRV. The low VN degree of 2 simplifies the implementation, as shown in Fig. 7. Three EVNs are used: EVN_1 and EVN_2 start first to compute U_2 and U_1, followed by EVN_3. This design shortens the VN critical path, as EVN_3 has been excluded from the critical path.

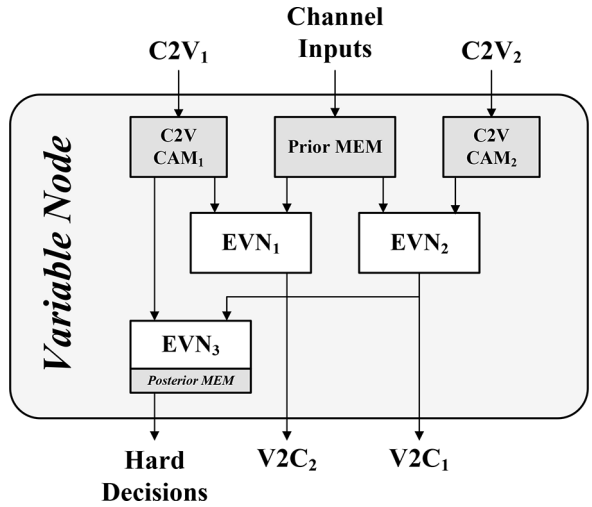


Fig. 7. Architecture of the variable node.

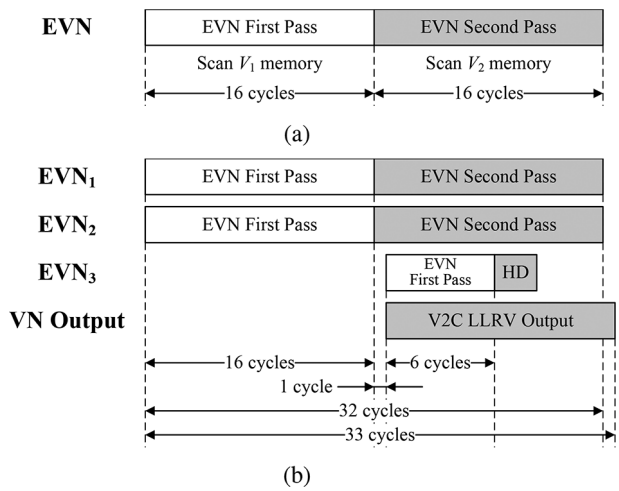


Fig. 8. Operation schedule of (a) the elementary variable node and (b) the variable node. Note that EVN_3 uses a shorter sorter length since only the minimum is required.

An EVN finds the n_m minimum values in the set $\{V_1(i) + V_2(j), V_1^{gf}(i) = V_2^{gf}(j), i \in [1, n_m], j \in [1, n_m]\}$, where V_1 and V_2 are two input LLRVs. The condition $V_1^{gf}(i) = V_2^{gf}(j)$ requires matching of GF indices. Therefore, one of the input LLRVs, e.g., V_2, is stored in a content-addressable memory (CAM) to enable searching of the GF index. EVN implements a two-pass scan: (1) in the first pass, EVN scans V_1 memory, and searches matching GF index in V_2 memory. If a matching entry is found, e.g., $V_1^{gf}(i) = V_2^{gf}(j)$, the entry $V_2(j)$ is read to calculate $V_1(i)+V_2(j)$; if no matching entry is found, a fixed offset is added to $V_1(i)$ and the sum is inserted to the EVN sorter; (2) in the second pass, EVN scans V_2 memory. A fixed offset is added to $V_2(j)$ and the sum is inserted to the sorter. The insertion sorter performs a sort every cycle and keeps its stored items in ascending order.

To support the two-pass scan, the EVN sorter length is kept at least $n_m + 1$ to consider all n_m V_1 entries of the first pass and the first V_2 entry of the second pass. Simulations show that the EVN sorter length directly impacts the BER performance of the decoder. Therefore we choose the EVN sorter length 17

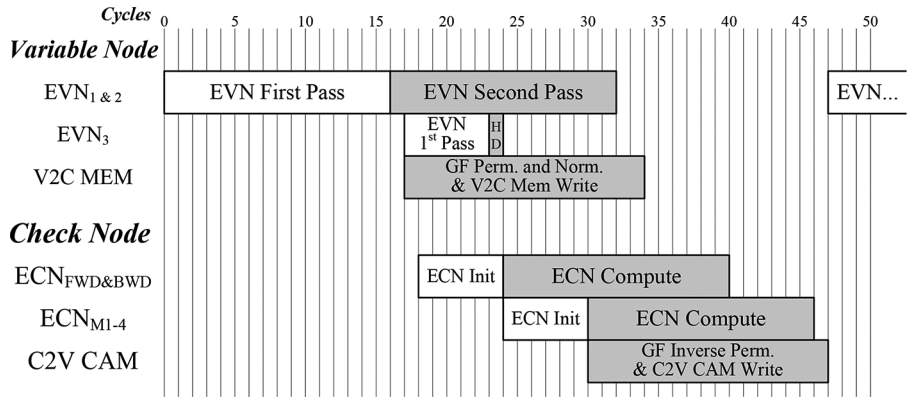


Fig. 9. Operation schedule of the decoder which includes the variable node, check node, permutation & normalization, and inverse permutation stages.

to avoid a degradation in BER. However, note that EVN_3 is different from EVN_1 and EVN_2 in that only the top (minimum) entry in the posterior LLRV is needed to determine the hard decision. We take advantage of this property to shorten the two passes performed by EVN_3. The shortening of the two passes are verified by simulation to guarantee accurate hard decisions.

The EVN schedule is divided into two phases: first pass and second pass, as in Fig. 8(a). In the VN schedule shown in Fig. 8(b), EVN_1 and EVN_2 start in parallel. The first pass takes $n_m = 16$ cycles, followed by n_m cycles for the second pass. EVN_1 and EVN_2 each outputs one value every cycle starting from the second cycle of the second pass. After EVN_1 and EVN_2 begin to produce outputs, ECN_3 starts with a 6-cycle first pass followed by a 1-cycle second pass to obtain the hard decision.

C. Interleaving Check Node and Variable Node

The phased schedule of ECN and EVN allows the CN and VN to be interleaved for a shorter latency and higher throughput. The interleaved schedule is illustrated in Fig. 9 and it is executed in the following order: (1) EVN_1 and EVN_2 first pass, followed by second pass. In the second pass, each EVN outputs one entry of the V2C message per cycle to be permuted and forwarded to the CNs; (2) forward and backward ECN initialization, followed by merge ECN initialization and compute. Merge ECN outputs one entry of the C2V message per cycle in the compute phase to be inverse permuted and forwarded to the VNs. EVN_1 and EVN_2 of the next iteration need to wait until all n_m entries of the C2V message to be stored in the CAM for searching. The overall latency of one decoding iteration is 47 cycles according to Fig. 9. Note that EVN_3 is not in the critical path and it can be overlapped with EVN operations.

IV. Low-Power Design by Fine-Grained Dynamic Clock Gating

To estimate the power consumption, a fully parallel nonbinary LDPC decoder has been synthesized and place and routed in a 65 nm CMOS process. Fig. 10(a) shows the power breakdown of the decoder. The switching power of sequential circuits is the dominant portion, claiming 65% of the total power. The leakage power and the switching power of combinational circuits claim the remaining 21% and 14% of the total power,

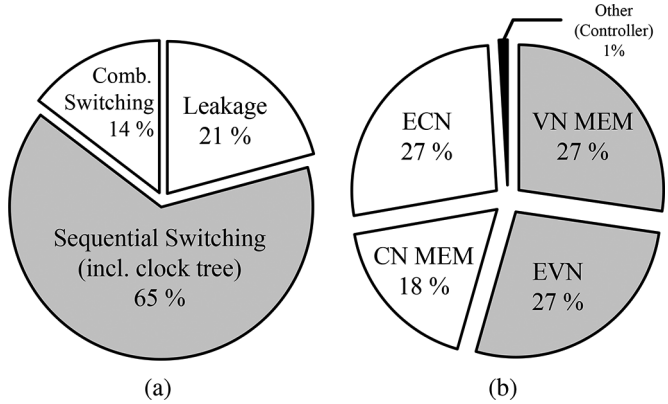


Fig. 10. (a) Power breakdown of the 65 nm synthesized fully parallel nonbinary LDPC decoder, and (b) the distribution of sequential logic used in the decoder.

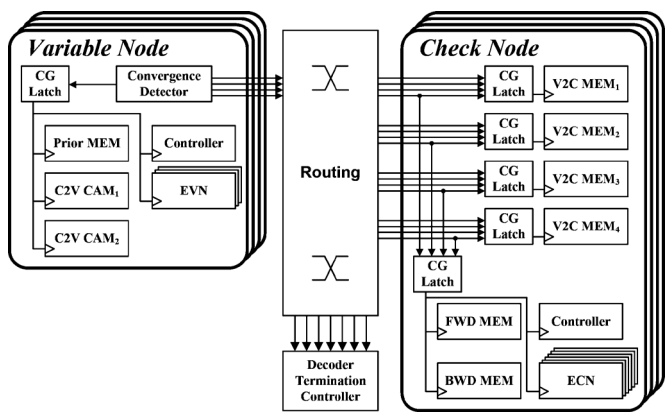


Fig. 11. Implementation of fine-grained dynamic clock gating for the variable and check node.

respectively. Further breakdown of the switching power of sequential circuits in Fig. 10(b) shows that the switching power of the VN and CN memories and the sorters in EVNs and ECNs account for almost all of the sequential switching power.

The high dynamic power consumption prompts us to design a dynamic clock gating strategy to reduce the power consumption of the decoder. Clock gating disables the clock input to sequential circuits to save switching power, which in turn cuts the switching of combinational circuits. The use of clock gating is

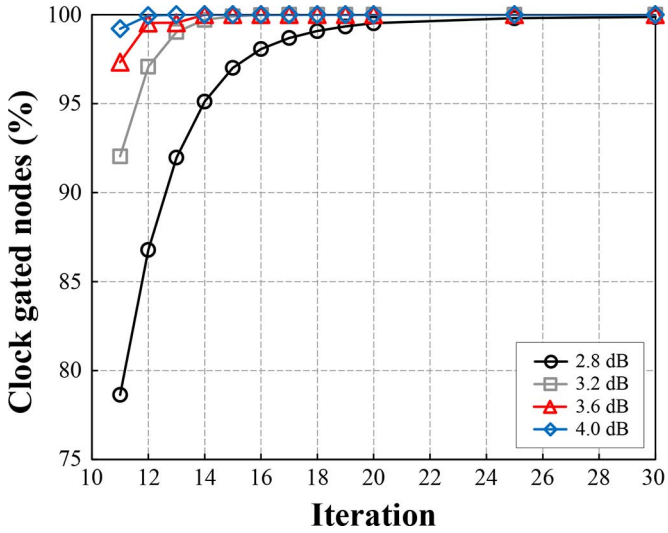


Fig. 12. Cumulative distribution of clock gated nodes at each iteration for various SNR levels with a decoding iteration limit of 30. The parameters used for clock gating are $M = 10$ and $T = 10$.

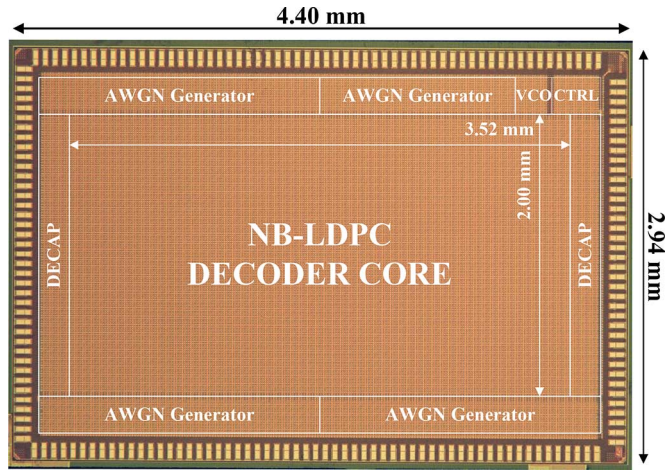


Fig. 13. Chip microphotograph of the decoder test chip. Locations of the test peripherals and the decoder are labeled.

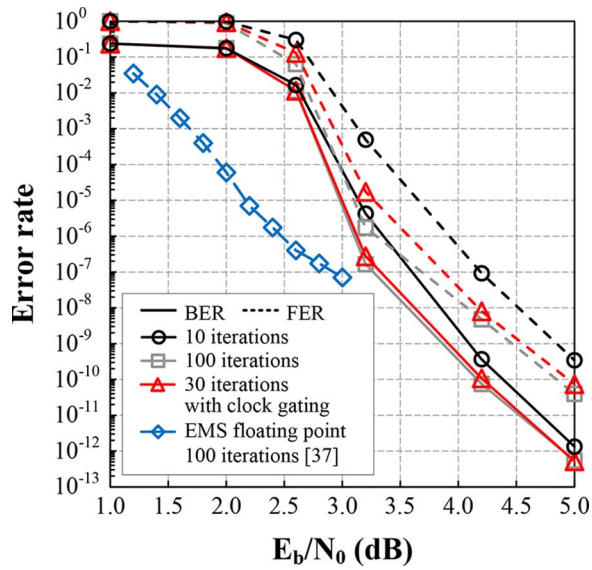


Fig. 14. Bit error rate and frame error rate performance of the GF(64) (160, 80) regular-(2, 4) NB-LDPC code using 5-bit quantization and floating point.

motivated by the observation that the majority of the VNs converge within a few decoding iterations before reaching the decoding iteration limit. Therefore, it is possible to clock gate the VNs and CNs that have reached convergence to save power.

To achieve the most power savings, the clock gating is implemented at a fine-grained node level, i.e., at each VN and CN, and the clock gating is enabled dynamically during run time. The fine-grained dynamic clock gating requires convergence detection at the node level, i.e., each VN detects when it has reached convergence and can be clock gated. The node-level convergence detection is different from the conventional convergence detection done at the global level by checking whether all parity checks have been met [39]. Although clock gating can also be based on global convergence detection, the power savings would be greatly diminished.

A. Node-Level Convergence Detection

Node-level convergence detection is not equivalent to global convergence detection. Our proposed node-level convergence detection is designed to match the accuracy of the global convergence detection without causing BER degradation. The node-level convergence detection is based on two convergence criteria: (1) meet the minimum number of decoding iterations M, and (2) VN's hard decisions remain unchanged for the last T consecutive iterations. The two criteria are designed to prevent false convergence and ensure stability. Each VN checks the criteria upon completing each decoding iteration. If the criteria are met, the VN is clock gated. If a VN is clock gated, parts of the CN that are used for storing and processing messages from and to the VN are also clock gated. A CN is completely clock gated when all its connected VNs have been clock gated.

With node-level convergence detection, it is possible that a VN converges to the correct decision and is frozen, preventing it from changing to an incorrect decision. On the other hand, it is also possible that a VN converges to an incorrect decision and is frozen, and preventing other VNs from correcting this VN later. To best match the BER and FER performance of global convergence detection, M and T need to be set appropriately.

Fine-grained dynamic clock gating can be compared to early termination [39], [40] that is commonly used in decoder designs. Early termination relies on global convergence detection, whereas fine-grained dynamic clock gating is based on node-level convergence detection, and it allows a large number of VNs and CNs to be turned off before the global convergence is reached.

The idea of early termination can be combined with fine-grained dynamic clock gating to save power and improve throughput by terminating the decoder once all the VNs and CNs are clock gated. We term the approach decoder termination to differentiate it from early termination, because decoder termination relies on node-level convergence detection, whereas early termination commonly relies on global convergence detection.

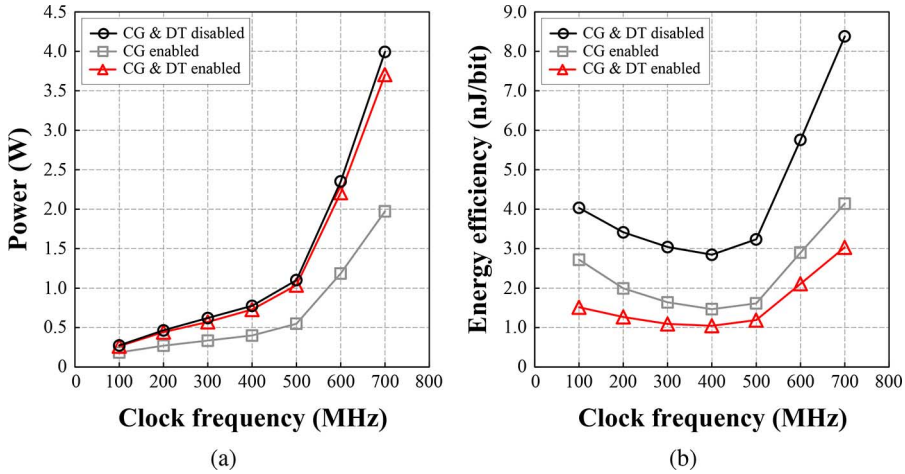


Fig. 15. Measured NB-LDPC decoder (a) power and (b) energy efficiency at 5.0 dB SNR and 30 decoding iterations. CG denotes clock gating and DT denotes decoder termination. The parameters used for clock gating and decoder termination are $M = 10$ and $T = 10$. This minimum supply voltage is used at each clock frequency.

B. Fine-Grained Dynamic Clock Gating

The clock gating architecture is illustrated in Fig. 11. The convergence detector inside each VN monitors the hard decisions in each iteration to check whether the hard decisions have changed between iterations. A counter keeps track of the number of consecutive iterations that the hard decisions have remained unchanged. When the convergence criteria are met, the convergence detector enables the clock gating latch (CG latch) to turn off the clock input to all sequential circuits with the exception of essential control circuits that are needed for recovering from the clock gating state. The majority of the VN's dynamic power is saved, leaving only leakage.

The convergence detector propagates the clock gating signal to the CNs to enable the CG latch of V2C message memories in the CNs, as noted in Fig. 11. Clock gating V2C memories eliminates the unnecessary memory updates to save dynamic power. In this way, CN is partially clock gated. When all the connected VNs are clock gated, as indicated by their clock gating signals, a central CG latch is enabled to completely turn off the CN.

A decoder termination controller monitors the VN clock gating signals. When all the VNs are clock gated (and CNs are clock gated as a result), the decoder terminates the current frame and moves on to the next input frame. Decoder termination reduces the average number of decoding iterations per code frame and therefore improves the decoding throughput for a net gain in energy efficiency.

In our implementation, each VN stores only the hard decision (6 bit) from the previous iteration. In each iteration, the VN compares the hard decision with the previous hard decision, and increments a 4-bit counter if they agree. If not, the counter is reset. After the comparison, the stored hard decision is replaced by the current hard decision for the next iteration. The node-level convergence detection requires only 6 bits of storage per VN (or 960 bits for the entire decoder), a small logic in each VN to compare a pair of 6-bit decisions, and a 4-bit counter. Compared to the size of the nonbinary VN and CN, the overhead for node-level convergence detection is negligible.

To check the effectiveness of fine-grained dynamic clock gating, we simulated the decoder's behavior with node-level convergence detection. Fig. 12 shows the percentage of nodes that have been clock gated in each decoding iteration across various SNR levels. The decoding iteration limit is set to 30, and the convergence criteria are set to $M = 10$ and $T = 10$. Even at a low SNR (E_b/N_0) of 2.8 dB, more than 85% of the VNs are clock gated after 12 iterations. After 14 iterations,

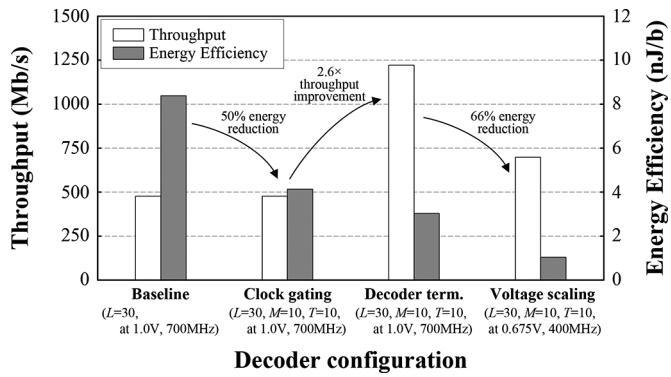


Fig. 16. Illustration of throughput and energy efficiency of various decoder configurations at 5.0 dB SNR. L, M, and T represents decoding iteration limit, minimum decoding iteration, and consecutive iteration threshold, respectively.

TABLE I
MEASUREMENT SUMMARY

Decoder Configuration	Throughput[1] (Mb/s)	Power[1] (W)	Energy Efficiency[1] (nJ/b)
700MHz @ 1.0V 30 Iterations	477	3.993	8.38
700MHz @ 1.0V 30 Iterations \w CG[2]	477	1.974	4.14
700MHz @ 1.0V 30 Iterations \w CG & DT[2]	1221	3.704	3.03
400MHz @ 0.675V 30 Iterations \w CG & DT[2]	698	0.729	1.04

[1] Measured at 5.0 dB SNR.
[2] CG denotes clock gating, and DT denotes decoder termination. The parameters used for clock gating and decoder termination are $M = 10$ and $T = 10$.

TABLE II
COMPARISON OF STATE-OF-THE-ART NB-LDPC DECODERS (ASIC LAYOUT)

	This Work		TVLSI'14 [30]	TVLSI'13 [29]	TSP'13 [26]	TCAS-I'12 [22]
Technology	65nm		90nm	28nm	90nm	90nm
Design	silicon		layout	layout	layout	layout
Code Length (symbols)	160		837	110	837	248
Code Rate	0.5		0.86	0.8	0.87	0.55
Galois Field	GF(64)		GF(32)	GF(256)	GF(32)	GF(32)
Decoding Algorithm	Truncated Extended Min-Sum		SES-GBFDA[a]	RTBCP[a]	Trellis-based Max-Log-QSPA[a]	Selective-input Min-Max
Core Area (mm^2)	7.04		6.6	1.289	46.18	10.33
Utilization (%)	87		-	75.7	-	-
Gate Count	2.78M (NAND)		0.468M (XOR)	2.57M (NAND)	8.51M (NAND)	1.92M (NAND)
Core Supply (V)	1.0	0.675	-	-	-	-
Clock Frequency (MHz)	700	400	277	520	250	260
Iterations	10-30[b]	10-30[b]	10	10	5	10
Throughput (Mb/s)	1221	698	716	546	234	47.7
Power (mW)	3704	729	-	976	893	480
Energy Efficiency (nJ/b)	3.03	1.04	-	1.78	3.82	10.06
Energy Efficiency (pJ/b/iter)	259	89	-	178	765	1006
Area Efficiency (Mb/s/mm^2)	173	99.1	108	424	5.06	4.62
Normalized to 65nm, 1.0V[c]						
Energy Efficiency (nJ/b)	3.03	2.29	-	4.15	2.76	7.27
Energy Efficiency (pJ/b/iter)	259	196	-	415	552	727
Area Efficiency (Mb/s/mm^2)	173	99.1	288	33.9	13.4	12.3

[a] SES-GBFDA stands for simplified enhanced serial generalized bit-flipping decoding algorithm, RTBCP stands for reduced memory complexity trellis-based check node processing, QSPA stands for q-ary sum-product algorithm.
[b] Iteration varies from 10 to 30 iterations based on decoder termination. The average iteration at 5.0 dB SNR is 11.71.
[c] General scaling theory is used to scale area, frequency (and throughput), and power by $1/s^2$, s, and $1/u^2$ respectively where s is the dimension scale factor and u is the voltage scale factor. The core supply of [22], [26], [29] are assumed to be 1.0V for normalization purpose.

95% of the VNs are clock gated. At higher SNRs, the VNs are clock gated at an even faster pace.

The setting of M determines how much power can be saved. The lower the M, the earlier the clock gating can be applied, and the more power we can save. However, a lower M degrades the BER. There is a tradeoff between power consumption and BER. We set $M = 10$ to ensure no loss in BER across a wide range of SNR. We could use a lower M at high SNR to achieve more power savings without affecting BER.

V. DECODER CHIP IMPLEMENTATION AND MEASUREMENT RESULTS

A decoder test chip for the GF(64) (160, 80) regular-(2, 4) NB-LDPC code was implemented in a STMicroelectronics 65 nm 7-metal general-purpose CMOS technology [41]. The chip microphotograph is shown in Fig. 13. The test chip measures 4.40 mm $\times$ 2.94 mm, and the core measures 3.52 mm $\times$ 2.00 mm, or 7.04 mm^2. The memory used in this decoder is implemented using registers. The test chip incorporates AWGN generators to model the communication channel and provide input vectors in real time. An on-chip controller keeps track of the decoding errors for calculating the bit error rate (BER) and frame error rate (FER).

The chip supports two test modes: a scan mode which takes external inputs through scan chains and provides outputs through scan chains for functional verification, and an automated mode for the real-time testing of the decoder using on-chip generated AWGN noise to emulate the communication channel. Error statistics are collected for plotting BER and FER against SNR.

Fig. 14 shows the BER and FER curves for various configurations. The 5-bit decoder incurs a relatively large quantization loss (compared to floating point) at low SNR, because the 5 bit word length is not sufficient to separate the candidate elements for a VN at low SNR. At moderate to high SNR, the candidate elements can be more easily separated, which explains the much smaller quantization loss at high SNR. Note that the highest SNR point in [41] was inaccurate. The error rate reported in Fig. 14 is based on two months of extensive testing. With a decoding iteration limit of 100, the decoder achieves a BER of 7.53×10^{-11} at 4.2 dB, a significant improvement over binary LDPC codes of similar block length, e.g., the rate-1/2 672-bit binary LDPC code for the IEEE 802.11ad standard provides a BER of 4.36×10^{-8} at 4.2 dB [14]. Structured binary LDPC codes of similar block length also encounter severe error floors, which is not seen in the NB-LDPC code. With a more practical

TABLE III
COMPARISON OF STATE-OF-THE-ART NB-LDPC DECODERS (ASIC SYNTHESIS)

	This Work		TVLSI'13 [28]	TCAS-I'12 [25]	TCAS-I'12 [21]	TVLSI'11 [19]
Technology	65nm		180nm	180nm	180nm	-
Design	silicon		synthesis	synthesis	synthesis	analysis
Code Length (symbols)	160		837	837	837	837
Code Rate	0.5		0.87	0.87	0.87	0.87
Galois Field	GF(64)		GF(32)	GF(32)	GF(32)	GF(32)
Decoding Algorithm	Truncated Extended Min-Sum		Relaxed Min-Max	Simplified Min-Sum	Min-Max	Reduced-complexity Min-Max
Gate Count	2.78M (NAND)		0.871M (NAND)	1.29M (NAND)	1.37M (NAND)	0.639M (XOR)
Clock Frequency (MHz)	700	400	200	200	200	150
Iterations	10-30[a]	10-30[a]	15	15	15	15
Throughput (Mb/s)	1221	698	64	64	16	10
Normalized to 65nm[b]						
Throughput (Mb/s)	1221	698	183	177	44	-

[a] Iteration varies from 10 to 30 iterations based on decoder termination. The average iteration at 5.0 dB SNR is 11.71.
[b] General scaling theory is used to scale frequency (and throughput) by s where s is the dimension scale factor.

30 iterations and our proposed node-level convergence criteria of $M = 10$ and $T = 10$, the decoder still provides an excellent BER performance that is very close to the 100-iteration BER performance.

The NB-LDPC decoder test chip operates at a maximum clock frequency of 700 MHz at 1.0 V and room temperature for a coded throughput of 477 Mb/s with 30 decoding iterations. The test chip consumes 3.993 W, which translates to an energy efficiency of 8.38 nJ/b. Figs. 15, 16 and Table I summarize the measured power consumption of the NB-LDPC decoder test chip. To improve the energy efficiency, fine-grained dynamic clock gating is enabled with node-level convergence criteria of $M = 10$ and $T = 10$, reducing the power consumption by 50% and improving the energy efficiency to 4.14 nJ/b. To achieve a higher throughput, decoder termination is enabled to increase the throughput from 477 Mb/s to 1.22 Gb/s at 5.0 dB SNR $(\mathrm{E}_b/\mathrm{N}_0)$. The power consumption increases due to a higher activity, but the energy efficiency improves to 3.03 nJ/b, or 259 pJ/b/iteration. Voltage and frequency scaling can be applied to further reduce the power consumption and improve the energy efficiency. Scaling the supply voltage from 1.0 V to 675 mV reduces the maximum clock frequency from 700 MHz to 400 MHz and improves the energy efficiency to 1.04 nJ/b, or 89 pJ/b/iteration, at a reduced throughput of 698 Mb/s.

Tables II and III list the nonbinary LDPC decoder test chip along with other state-of-the-art synthesized designs published recently [19], [21], [22], [25], [26], [28]–[30]. To summarize, [19] is a min-max decoder using the most reliable, compressed messages; [22] is a layered min-max decoder for QC-LDPC codes using barrel-shifter-based permutation network to implement multiplications in the check nodes; [28] is a min-max decoder using relaxed check node that finds the minimum basis of the messages and derives remaining messages from the minimum basis; [25] is a simplified min-sum decoder for a higher throughput and smaller memory; [26] is a layered q-ary sum-product decoder using trellis-based check node implementation; [30] is a generalized bit-flipping decoder for a smaller memory and area; [21] is a configurable QC-LDPC decoder using barrel shifter and multithreaded pipelining to improve the throughput per unit area; and [29] is a fully parallel decoder with trellis-based check node. It is important to note that none of the previous designs has been fabricated in silicon. This work is the first silicon that has been published to the best of our knowledge. The decoder claims higher throughput and energy efficiency (in pJ/b/iter), when normalized to 65 nm and 1.0 V, than the best previously reported post-layout results. The truncated EMS algorithm allows us to achieve excellent BER performance compared to other simplified algorithms.

VI. CONCLUSION

We present a fully parallel NB-LDPC decoder to take advantage of the low wiring overhead that is intrinsic to NB-LDPC codes. To further enhance the throughput, we apply a one-step look-ahead to the elementary CN design to reduce the clock period, and interleave the CN and VN operations for a short iteration latency of 47 cycles. We implement a fine-grained clock gating at the node level to allow the majority of the processing nodes to be clock-gated long before reaching the iteration limit. A 7.04 mm^2 65 nm decoder test chip is designed for the GF(64) (160, 80) regular-(2, 4) NB-LDPC code. The decoder implements fine-grained dynamic clock gating and decoder termination to achieve a high throughput of 1.22 Gb/s at 700 MHz, consuming 3.03 nJ/b, or 259 pJ/b/iteration. The test chip demonstrates a superior error correcting performance compared to binary LDPC decoders. Voltage and frequency scaling of the test chip to 675 mV and 400 MHz further improve the energy efficiency to 89 pJ/b/iteration at a reduced throughput of 698 Mb/s.

REFERENCES

[1] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA, USA: MIT Press, 1963.

[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] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261–1271, Oct. 1996.

[4] *ETSI Standard TR 102 376 V1.1.1: Digital Video Broadcasting (DVB) User Guidelines for the Second Generation System for Broadcasting, Interactive Services, News Gathering and Other Broadband Satellite Applications (DVB-S2)*, ETSI Std. TR 102 376, Feb. 2005.

[5] *IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1*, IEEE Std. 802.16e, Feb. 2006.

[6] *IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements Part 3: Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, IEEE Std. 802.3an, Sep. 2006.

[7] *IEEE Draft Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment : Enhancements for Higher Throughput*, IEEE Std. 802.11n/D2.00, Feb. 2007.

[8] *IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band*, IEEE Std. 802.11ad, Dec. 2012.

[9] *3GPP Standard TS 25.944 Rev. 4.1.0: 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Channel Coding and Multiplexing Examples*, 3GPP Organizational Partners Std. HSDPA, Jun. 2001.

[10] *3GPP Standard TS 36.212 Rev. 8.3.0: 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Coding (Release 9)*, 3GPP Organizational Partners Std. 3GPP-LTE, May 2008.

[11] Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolić, "An efficient 10 GBASE-T Ethernet LDPC decoder design with low error floors," *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 843–855, Apr. 2010.

[12] B. Xiang, D. Bao, S. Huang, and X. Zeng, "An 847–955 Mb/s 342–397 mW dual-path fully-overlapped QC-LDPC decoder for WiMAX system in 0.13 μm CMOS," *IEEE J. Solid-State Circuits*, vol. 46, no. 6, pp. 1416–1432, Jun. 2011.

[13] S.-W. Yen, S.-Y. Hung, C.-H. Chen, H.-C. Chang, S.-J. Jou, and C.-Y. Lee, "A 5.79-Gb/s energy-efficient multirate LDPC codec chip for IEEE 802.15.3c applications," *IEEE J. Solid-State Circuits*, vol. 47, no. 9, pp. 2246–2256, Sep. 2012.

[14] Y. S. Park, D. Blaauw, D. Sylvester, and Z. Zhang, "Low-power high-throughput LDPC decoder using non-refresh embedded DRAM," *IEEE J. Solid-State Circuits*, vol. 49, no. 3, pp. 783–794, Mar. 2014.

[15] C. Benkeser, A. Burg, T. Cupaiuolo, and Q. Huang, "Design and optimization of an HSDPA turbo decoder ASIC," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 98–106, Jan. 2009.

[16] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang, "Design and implementation of a parallel turbo-decoder ASIC for 3GPP-LTE," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 8–17, Jan. 2011.

[17] M. C. Davey and D. Mackay, "Low-density parity check codes over GF(q)," *IEEE Commun. Lett.*, vol. 2, no. 6, pp. 165–167, Jun. 1998.

[18] X. Zhang and F. Cai, "Efficient partial-parallel decoder architecture for quasi-cyclic nonbinary LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 2, pp. 402–414, Feb. 2011.

[19] X. Zhang and F. Cai, "Reduced-complexity decoder architectures for non-binary LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 7, pp. 1229–1238, Jul. 2011.

[20] Y. Tao, Y. S. Park, and Z. Zhang, "High-throughput architecture and implementation of regular $(2, d_c)$ nonbinary LDPC decoders," in *Proc. IEEE Int. Symp. Circuits and Systems*, Seoul, Korea, May 2012, pp. 2625–2628.

[21] X. Chen, S. Lin, and V. Akella, "Efficient configurable decoder architecture for nonbinary quasi-cyclic LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 1, pp. 188–197, Jan. 2012.

[22] Y.-L. Ueng, C.-Y. Leong, C.-J. Yang, C.-C. Cheng, K.-H. Liao, and S.-W. Chen, "An efficient layered decoding architecture for nonbinary QC-LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 2, pp. 385–398, Feb. 2012.

[23] C. Zhang and K. K. Parhi, "A network-efficient nonbinary QC-LDPC decoder architecture," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 6, pp. 1359–1371, Jun. 2012.

[24] X. Zhang, F. Cai, and S. Lin, "Low-complexity reliability-based message-passing decoder architectures for non-binary LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 11, pp. 1938–1950, Nov. 2012.

[25] X. Chen and C.-L. Wang, "High-throughput efficient non-binary LDPC decoder based on simplified min-sum algorithm," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 11, pp. 2784–2794, Nov. 2012.

[26] Y.-L. Ueng, K.-H. Liao, H.-C. Chou, and C.-J. Yang, "A high-throughput trellis-based layered decoding architecture for non-binary LDPC codes using Max-Log-QSPA," *IEEE Trans. Signal Process.*, vol. 61, no. 11, pp. 2940–2951, Jun. 2013.

[27] J. Lin and Z. Yan, "Efficient shuffled decoder architecture for nonbinary quasi-cyclic LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 9, pp. 1756–1761, Sep. 2013.

[28] F. Cai and X. Zhang, "Relaxed min-max decoder architectures for non-binary low-density parity-check codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 11, pp. 2010–2023, Nov. 2013.

[29] J. Lin and Z. Yan, "An efficient fully parallel decoder architecture for nonbinary LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2013, (early access).

[30] F. Garcı´a-Herrero, M. J. Canet, and J. Valls, "Nonbinary LDPC decoder based on simplified enhanced generalized bit-flipping algorithm," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 6, pp. 1455–1459, Jun. 2014.

[31] L. Zeng, L. Lan, Y. Y. Tai, S. Song, S. Lin, and K. Abdel-Ghaffar, "Construction of nonbinary quasi-cyclic LDPC codes: A finite field approach," *IEEE Trans. Commun.*, vol. 56, no. 4, pp. 545–554, Apr. 2008.

[32] S. Song, B. Zhou, S. Lin, and K. Abdel-Ghaffar, "A unified approach to the construction of binary and nonbinary quasi-cyclic LDPC codes based on finite fields," *IEEE Trans. Commun.*, vol. 57, no. 1, pp. 84–93, Jan. 2009.

[33] B. Zhou, J. Kang, S. Song, S. Lin, K. Abdel-Ghaffar, and M. Xu, "Construction of non-binary quasi-cyclic LDPC codes by arrays and array dispersions," *IEEE Trans. Commun.*, vol. 57, no. 4, pp. 1652–1662, Jun. 2009.

[34] S. Kim and G. E. Sobelman, "Scaling, offset, and balancing techniques in FFT-based BP nonbinary LDPC decoders," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 5, pp. 277–281, May 2013.

[35] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633–643, Apr. 2007.

[36] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low-complexity decoding for non-binary LDPC codes in high order fields," *IEEE Trans. Commun.*, vol. 58, no. 5, pp. 1365–1375, May 2010.

[37] V. Savin, "Min-Max decoding for non binary LDPC codes," in *Proc. IEEE Int. Symp. Information Theory*, Toronto, Canada, Jul. 2008, pp. 960–964.

[38] E. Boutillon and L. Conde-Canencia, "Bubble check: A simplified algorithm for elementary check node processing in extended min-sum non-binary LDPC decoders," *IEEE Electron. Lett.*, vol. 46, no. 9, pp. 633–634, Apr. 2010.

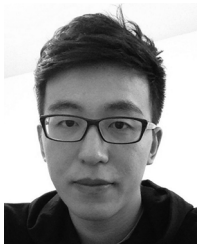
[39] X.-Y. Shi, C.-Z. Zhan, C.-H. Lin, and A.-Y. Wu, "An 8.29 mm^2 52 mW multi-mode LDPC decoder design for mobile WiMAX system in 0.13 μm CMOS process," *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 672–683, Mar. 2008.

[40] A. Darabiha, A. Chan Carusone, and F. R. Kschischang, "Power reduction techniques for LDPC decoders," *IEEE J. Solid-State Circuits*, vol. 43, no. 8, pp. 1835–1845, Aug. 2008.

[41] Y. S. Park, Y. Tao, and Z. Zhang, "A 1.15 Gb/s fully parallel nonbinary LDPC decoder with fine-grained dynamic clock gating," in *2013 IEEE Int. Solid-State Circuits Conf. Dig.*, San Francisco, CA, Feb. 2013, pp. 422–423.

Youn Sung Park (S'10–M'14) received the B.A.Sc. degree in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 2008, and the M.S. and Ph.D. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2012 and 2014, respectively. His Ph.D. thesis involved energy-efficient decoders of near-capacity channel codes.

Since then, he has been a senior design engineer at Marvell Semiconductor in the Data Storage division. His research interest is in the design of energy-efficient digital signal processors through algorithm, architecture, and circuit co-optimization.



Yaoyu Tao (S'11) received the B.Sc. and M.S. degrees in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2011 and 2013, respectively. Since then, he has been pursuing the Ph.D. degree at Stanford University, Stanford, CA, USA. His current research interests are high-speed wireless link based on millimeter-wave MIMO and design of energy-efficient digital signal processors.

Zhengya Zhang (S'02–M'09) received the B.A.Sc. degree in computer engineering from the University of Waterloo, Canada, in 2003, and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley, CA, USA, in 2005 and 2009, respectively.

Since 2009, he has been on the faculty of the University of Michigan, Ann Arbor, MI, USA, as an Assistant Professor in the Department of Electrical Engineering and Computer Science. His current research interests include low-power and high-performance VLSI circuits and systems for computing, communications and signal processing.

Dr. Zhang was a recipient of the National Science Foundation CAREER Award in 2011, the Intel Early Career Faculty Honor Program Award in 2013, the David J. Sakrison Memorial Prize for outstanding doctoral research in EECS at UC Berkeley, and the Best Student Paper Award at the Symposium on VLSI Circuits. He is an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I: REGULAR PAPERS.