INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. 2013; **26**:68–83
Published online 7 September 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/dac.1331

Operation reduced low-density parity-check decoding algorithms for low power communication systems

Chia-Yu Lin*,†, Shu-Cheng Chou and Mong-Kai Ku

Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan

SUMMARY

Decoding operation reduction algorithms on min-sum layered low-density parity-check (LDPC) decoders are proposed in this paper. Our algorithm freezes selected operations in high reliable nodes to reduce power while preserving error correcting performance. Both memory accesses and active node switching activities can be reduced. A novel node refresh mechanism reactivates frozen nodes to minimize coding gain degradation. We propose three decoding operation reduction algorithm variations to trade-off complexity and operation reduction for LDPC decoders with different degrees of parallelism and memory requirement. Simulation results show that the number of LDPC decoding operations is reduced across all SNR ranges. The decoding convergence speed is not affected. Hardware architecture and FPGA implementation for IEEE 802.16e LDPC codes are presented. Copyright © 2011 John Wiley & Sons, Ltd.

Received 14 October 2010; Revised 14 December 2010; Accepted 8 July 2011

KEY WORDS: low power communication systems; LDPC codes; decoding; operation reduction

1. INTRODUCTION

Low-density parity-check (LDPC) codes were first introduced by Gallager in 1962 [1]. To achieve good error correcting performance, the parity check matrix of an LDPC code is commonly designed as a sparse matrix with size ranging from several hundreds to thousands. In 1995, LDPC codes were rediscovered by MacKay and Neal [2]. LDPC codes gathered great amount of research interest for capacity-approaching performance and parallelizable decoder architecture. With advanced VLSI technologies, medium-sized to large-sized LDPC codes can be realized with high decoding throughput. LDPC codes are currently adopted by several industrial wireless and wired communication standards, such as satellite (DVB-S2 (Second Generation Digital Video Broadcasting via Satellite)) [3], WLAN (Wireless Local Area Network) (IEEE 802.11n) [4], WMAN (Wireless Metropolitan Area Network) (IEEE 802.16e) [5], and 10 Gigabit Ethernet (IEEE 802.3an) [6].

In general, LDPC codes with larger parity check matrices can achieve better coding gain. However the decoder complexity and power consumption rise significantly as the size of parity check matrix increases. It is critical to design a low hardware cost decoding algorithm that reduces power consumption while maintaining good error correcting performance. To reduce the power consumption of LDPC decoders, the following three aspects must be considered: (1) node processor complexity; (2) number of iterations required for decoding a codeword; and (3) number of active nodes or operations participating in the message passing process [7]. Most partially parallel LDPC decoders store temporary likelihood messages in large banks of memories. Read and write accesses of these large message memories consume the majority of power in LDPC decoders. Hence, minimizing the

*Correspondence to: Chia-Yu Lin, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan.

†E-mail: f93084@csie.ntu.edu.tw

number of memory-access operations is vital in lowering the power consumption of LDPC decoders. A smart decoding algorithm can be employed to shut down reliable nodes to save power. However, the coding gain degradation needs to be kept to a minimum for any operation reduction scheme to be practical.

In this paper, we propose three decoding algorithms that adaptively deactivate selected operations of node processors to reduce switching activities and memory-access operations. The goal is to reduce power consumption without sacrificing the coding gain. Existing partially parallel decoder architectures can be easily modified to use the proposed decoding algorithms with low hardware overhead. The remainder of this paper is organized as follows. In Section 2, the techniques for reducing operations and saving power in LDPC decoders are reviewed. Section 3 describes the proposed decoding operation reduction algorithms in detail. The operation reduction capability and complexity of three algorithms are discussed. Section 4 presents the hardware decoder architectures for IEEE 802.16e LDPC codes. Section 5 shows the field-programmable gate array (FPGA) implementation results. Finally, Section 6 concludes this paper.

2. BACKGROUND

2.1. *Decoding of LDPC codes*

An (n, k) LDPC code is a special class of linear block code whose codewords are defined by a very sparse $m \times n$ parity check matrix $\mathbf{H}$, where m equals to $n - k$. A codeword x consisting of k information bits and m parity bits satisfies $\mathbf{H}x^T = 0$. The code can be represented by a Tanner graph [8] $(V,\ C,\ E)$ where the n columns and m rows of $\mathbf{H}$ are expressed by two disjoint sets, V and C, corresponding to n bit nodes $(v_1,\ v_2,\ \ldots,\ v_n)$ and m check nodes $(c_1,\ c_2,\ \ldots,\ c_m)$ respectively. The set E represents edges between bit nodes and check nodes, which correspond to the location of 1s in $\mathbf{H}$. A Tanner graph is usually used to explain the decoding process in a more intuitive way. Figure 1 shows the 4×8 parity check matrix and the corresponding Tanner graph. We will use this matrix as an example to explain the decoding behavior throughout the paper.

Belief propagation (BP) [2] is a widely used method to decode LDPC codes. The BP decoding algorithm passes log-likelihood ratio (LLR) messages between bit nodes and check nodes according to the Tanner graph of the code. Bit and check node processors take turns to update the LLR messages for all nodes. The decoding process continues for several iterations until a valid codeword is produced. Hardware decoders typically allow a predefined fixed number of iterations for the codeword to converge. Most practical LDPC decoder designs implement the BP algorithm with partially parallel architectures. Partially parallel LDPC decoders offer the best combination of hardware complexity and decoding throughput. The large likelihood memory and the node processors consume significant amount of power in most designs.

2.2. *Min-sum decoding algorithm with horizontal layered scheduling*

The scaled min-sum algorithm (SMSA) [9] significantly lowers the node processor complexity of the traditional sum-product algorithm (SPA) [10] with minor error correcting performance degradation. By changing the message passing schedule, algorithms such as layered decoding [11, 12] can

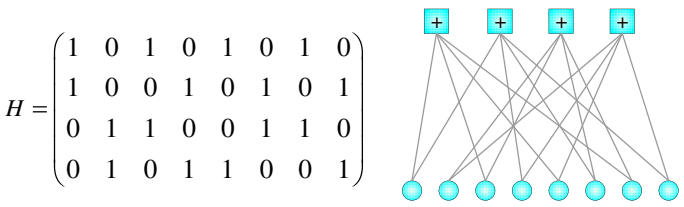


Figure 1. An (8,4) LDPC code: (a) parity check matrix, (b) corresponding Tanner graph.

DOI: 10.1002/dac

reduce the number of average iterations by a factor of two compared to traditional two-phase decoding. There are two different ways to schedule layered LDPC decoders depending on the orientation of **H**. Rows or columns of **H** can be viewed as horizontal or vertical layers, respectively. The SMSA with horizontal layered scheduling is introduced first with the following notations.

- P_v: channel *a priori* message for bit node v
- Q_v: APP (*a posteriori* probabilities) (soft output) message for bit node v
- R_{cv}: message passing from check node c to bit node v
- $Qtemp_{vc}$: temporary message passing from bit node v to check node c (It does not have to be stored in memory.)
- $N(c)$: the set of bit nodes connected to check nodes c
- $M(v)$: the set of check nodes connected to bit nodes v

SMSA with Horizontal Layered Scheduling

[Initalization]

$\forall (v,c) \in E \qquad R_{cv} \leftarrow 0$

$\forall v \in V \qquad Q_v \leftarrow P_v$

[Iteration]

$\forall c \in C$

$\forall v \in N(c)$

$$Qtemp_{vc} \leftarrow Q_v - R_{cv} \tag{1}$$

$\forall v \in N(c)$

$$R_{cv} \leftarrow \alpha \times \prod_{v' \in N(c)\backslash v} sign(Qtemp_{v'c}) \times \min_{v' \in N(c)\backslash v} |Qtemp_{v'c}| \tag{2}$$

$$Q_v \leftarrow Qtemp_{vc} + R_{cv} \tag{3}$$

[Termination]

If all parity checks are satisfied or the maximum number of iteraitons are reached, stop; else go to [Iteration].

In each decoding iteration, the layers are processed sequentially so that the newly updated LLR messages can be used by the subsequent layers. Hence, the average number of required decoding iterations can be reduced by half because of more reliable message propagation [13]. Figure 2 shows message passing of the horizontal layered schedules on the Tanner graph. Layered scheduling also requires less memory storage than two-phase scheduling because intermediate messages are computed on-the-fly.

Horizontal layered scheduling has two advantages over vertical layered scheduling. First, in terms of decoding complexity, the former uses min-sum decoding algorithm while the latter requires a more complex SPA. In the min-sum algorithm, calculating R_{cv}, as shown in (2), involves selecting the minimum magnitude among several bit-to-check messages. Note that vertical layered scheduling

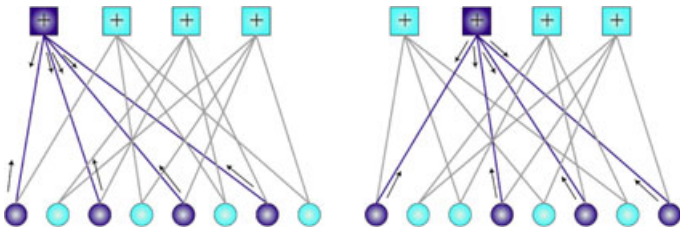


Figure 2. Horizontal layered schedule: (a) first layer message passing and (b) second layer message passing.

DOI: 10.1002/dac

sequentially updates bit-to-check messages in each iteration, and a more newly updated bit-to-check message is likely to have a larger magnitude. Hence, the min-sum algorithm does not work for vertical layered scheduling. Second, horizontal layered decoders require less memory than vertical layered decoders, because channel *a priori* messages P_v should be stored in memory for computing APP messages Q_v in vertical layered decoders. In contrast, a horizontal layered decoder computes APP messages on-the-fly without channel messages, as shown in (3), and thus saves memory.

2.3. Related work

Several techniques have been proposed in the past for lowering the switching activities within the LDPC decoding process. Early termination schemes can detect decodable and undecodable codewords [14, 15] to reduce average iterations in low-SNR conditions. Threshold decoding methods such as forced convergence decoding [7] and lazy scheduling [16] can be applied to reduce the number of active nodes in each iteration by monitoring the node reliabilities [7, 16–23]. Forced convergence decoding [7] uses a threshold rule to decide if a bit node should update its information in subsequent iterations or be considered as converged. This approach is further applied to check nodes for more reduction [17]. However, these approaches result in error-correcting performance degradation especially at high SNR regions. In [18], dynamic thresholds were adopted in threshold decoding to improve the performance, but the error floor still exists. In [18–20], the standard SPA or bit-flipping decoding algorithms were applied as post processing methods to alleviate the performance loss caused by threshold decoding. This approach will increase the hardware complexity and is not feasible for practical systems. Lazy scheduling [16] was proposed as a hybrid of vertical layered scheduling and forced convergence to reduce both the number of average iterations and number of active nodes in each iteration. The node deactivation is based on a probability function that should be optimized by density evolution analysis in each iteration to ensure that no error correcting performance loss is introduced. Hence, this approach is not suitable for practical implementation. In addition, as discussed previously, vertical layered scheduling only works under the SPA with more complicated decoder hardware and larger memory size. A decoder architecture was proposed in [21, 22] to show the effectiveness of threshold decoding in power saving. This architecture is also based on the SPA and only designed for a (3,6)-regular LDPC code. In [23], a threshold rule was used to reduce the memory-access operations of LDPC decoders. However, reducing node operations by threshold decoding is not considered.

3. PROPOSED DECODING ALGORITHMS WITH REDUCED OPERATIONS

Three LDPC decoding operation reduction algorithms (DORAs) are described in this section. Two novel ideas are used in our proposed algorithms. First, the proposed algorithms employ a threshold to disable node processors with periodical node refresh. Periodically reactivating those deactivated nodes minimizes error performance degradation with very low hardware cost. Second, instead of deactivating the whole node processor as mentioned in related works [7, 16–23], individual decoding operations are selectively deactivated for more fine-grained operation reduction.

Horizontal layered scheduling is considered to combine with the proposed min-sum node processor to provide good decoding convergence speed and low implementation cost. To reduce the switching activities in horizontal layered decoding, we propose to use the magnitude of APPs or bit-to-check messages as the threshold to deactivate node processors. APPs are the soft outputs of codeword bits after each iteration and bit-to-check messages are the messages form bit nodes to check nodes. By disabling the incoming and outgoing message update of bit nodes, considerable amount of memory-access operations can be saved. The choice of APPs and bit-to-check messages in three DORA variations offer combinations of operation reduction, logic overhead, throughput and BER performance for low power LDPC decoder designs.

3.1. Proposed decoding operation reduction algorithm

The first decoding operation reduction algorithm employs SMSA in node processors with horizontal layered scheduling at the message passing level. APP threshold-based node deactivation with fixed

DOI: 10.1002/dac

interval node wake-up reduces the switching activities while maintaining decoding performance. This base DORA-1 algorithm for high operation reduction is presented below.

Algorithm DORA-1

[Initalization]

$\forall (v,c) \in E \qquad R_{cv} \leftarrow 0$

$\forall v \in V \qquad Q_v \leftarrow P_v$

[Iteration]

$\forall c \in C$

$\forall v \in N(c)$

If $|Q_v| < t_v$ or i_u iterations are passed

$Qtemp_{vc} \leftarrow Q_v - R_{cv}$

Else

$Qtemp_{vc} \leftarrow Q_v$ **(4)**

End

$\forall v \in N(c)$

If $|Q_v| < t_v$ or i_u iterations are passed

$$R_{cv} \leftarrow \alpha \times \prod_{v' \in N(c) \setminus v} sign(Qtemp_{v'c}) \times \min_{v' \in N(c) \setminus v} |Qtemp_{v'c}|$$

$Q_v \leftarrow Qtemp_{vc} + R_{cv}$

End

[Termination]

If all parity checks are satisfied or the maximum number of iteraitons are reached, stop; else go to [Iteration].

The node processor compares the APP message magnitude $|Q_v|$ with a threshold t_v to determine if the node should be deactivated. If the APP is higher than t_v, the bit node is deemed reliable and certain node processor operations and memory-access operations can be deactivated. First, the computation of APP message and check-to-bit message does not have to proceed because the hard decision of this bit node is less likely to be changed in later iterations. Hence, the operations in (2) and (3) can be disabled and the memory write of R_{cv} and Q_v can be skipped. Second, the updating of bit-to-check message $Qtemp_{vc}$ can also be disabled. However, (1) cannot be simply skipped because the computation in (2) still needs $Qtemp_{vc}$ from all connected bit nodes. Consider the operation in (1), when Q_v is highly reliable, that is, $|Q_v| \geqslant t_v$, the sign of $Qtemp_{vc}$ is mainly decided by Q_v and would not be affected by R_{cv}. In addition, $|Qtemp_{vc}|$ is very likely to be larger than the minimum magnitude obtained in (2). Hence, (1) can be simplified to (4) and the memory read of R_{cv} can be skipped. In summary, one memory read operation for R_{cv} and two memory write operations for R_{cv} and Q_v can be saved. On the node processor side, two additions for (1) and (3) as well as one comparison and one XOR operation for the magnitude and sign computation in (2) can be saved.

Table I shows the operation reduction of DORA-1 algorithm with different thresholds (t_v). In the following simulations, the IEEE 802.16e LDPC code with code rate 1/2 and code length 2304 is used on the binary-input AWGN channel with a maximum of 30 iterations. The simulation results show that a lower threshold saves more operations and a higher threshold improves BER performance.

3.2. Low complexity coding gain compensation scheme

Simply skipping the above-mentioned operations may induce decoding errors if the APP message with a large magnitude leads to a wrong decision. This effect leads to error performance degradation in high SNR regions. Similar problems also appear in previous works [7, 18–20]. Although we can adjust the threshold to trade operation reduction for the coding gain loss, the error floor still

DOI: 10.1002/dac

Table I. Operation reduction against SMSA with different thresholds.

	Reduction (%) of $Qtemp_{vc}$, R_{cv} and Q_v operations with $i_u = 5$			
SNR (dB)	$t_v = 6$	$t_v = 8$	$t_v = 10$	$t_v = 12$
1.5	53.23	42.96	32.21	23.58
2	53.87	45.57	35.63	27.20
2.5	54.58	50.32	40.50	31.37
3	55.92	59.05	50.85	41.49
3.5	57.04	64.15	57.19	48.22
Avg.	54.93	52.41	43.28	34.37

exists even when the threshold value is increased significantly. To alleviate the coding gain loss, other works use the standard SPA or bit-flipping decoding algorithm as the post processing method when decoding fails [18–20]. This will increase both the implementation hardware cost and decoding latency. Instead, in the proposed algorithms, all nodes are waked up periodically to update their messages at a fixed iteration interval i_u. Hence, the wrong decision in deactivated nodes will have chances to be recovered.

Simulations show that this approach can achieve better error floor behavior and switching activity reduction than adjusting the threshold alone. As shown in Figure 3, the BER degradation can be limited to within 0.1 dB with the periodical reactivation. Note that all node processors in the simulation of Table I are refreshed at an interval $i_u = 5$. As shown in Table II, we see that a larger update interval saves more operations and a smaller update interval improves BER performance. For $i_u = 5$

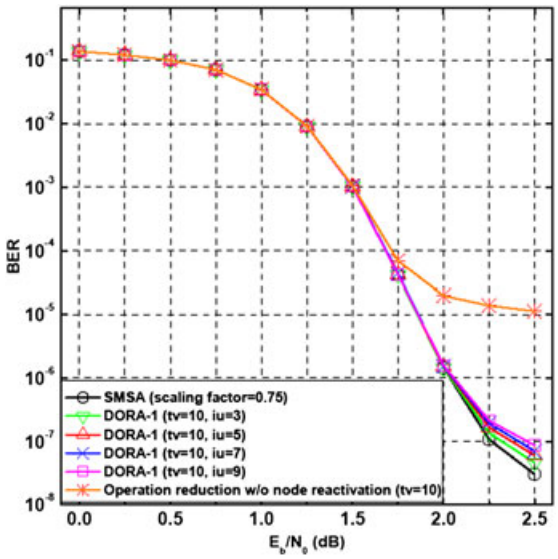


Figure 3. BER comparison with different update interval parameters.

Table II. Operation reduction against SMSA with different update intervals.

	Reduction (%) of $Qtemp_{vc}$, R_{cv} and Q_v operations with $t_v = 10$			
SNR (dB)	$i_u = 3$	$i_u = 5$	$i_u = 7$	$i_u = 9$
1.5	26.30	32.21	34.24	36.02
2	30.66	35.63	41.24	45.14
2.5	34.53	40.50	49.46	50.34
3	33.09	50.85	53.97	54.09
3.5	31.94	57.19	57.81	57.83
Avg.	31.30	43.28	47.34	48.68

DOI: 10.1002/dac

and $t_v = 10$, a large number of operations can be reduced while the coding gain degradation is less than 0.1 dB at BER $= 10^{-7}$ compared with SMSA.

3.3. Low overhead DORA-2/DORA-3 algorithm

The power saving logic overhead can be further reduced by modifying the base DORA-1 algorithm. We explore the design space of our algorithm and propose two low overhead algorithm variations. The first variation, DORA-2, reduce power saving logic overhead by removing the computation reduction in bit nodes. The modified algorithm is shown below. The computation of $Qtemp_{vc}$ in (1) is carried out all the time without examining the reliability of Q_v so the computation reduction in (4) is removed and extra memory accesses are needed to read the check-to-bit message memory for R_{cv}.

Algorithm DORA-2 (the iteration part)

[Iteration]
$\forall c \in C$
  $\forall v \in N(c)$
    $Qtemp_{vc} \leftarrow Q_v - R_{cv}$
  $\forall v \in N(c)$
    If $|Q_v| < t_v$ or i_u iterations are passed
      $R_{cv} \leftarrow \alpha \times \prod_{v' \in N(c)\setminus v} sign(Qtemp_{v'c}) \times \min_{v' \in N(c)\setminus v} |Qtemp_{v'c}|$
      $Q_v \leftarrow Qtemp_{vc} + R_{cv}$
    End

Following the changes in DORA-2, the second variation denoted DORA-3 replaces Q_v with bit-to-check messages $Qtemp_{vc}$ when making node deactivation decisions in (5). With this modification, the decision result does not have to propagate through the node processor, so the FIFO sizes in node processors can be reduced. Hence, the DORA-3 has the lowest logic overhead of the three variations.

Algorithm DORA-3 (the iteration part)

[Iteration]
$\forall c \in C$
  $\forall v \in N(c)$
    $Qtemp_{vc} \leftarrow Q_v - R_{cv}$
  $\forall v \in N(c)$
    If $|Qtemp_{vc}| < t_v$ or i_u iterations are passed **(5)**
      $R_{cv} \leftarrow \alpha \times \prod_{v' \in N(c)\setminus v} sign(Qtemp_{v'c}) \times \min_{v' \in N(c)\setminus v} |Qtemp_{v'c}|$
      $Q_v \leftarrow Qtemp_{vc} + R_{cv}$
    End

3.4. Operation reduction comparison and performance analysis

The operation reduction and coding gain performance of the proposed DORA algorithms are again evaluated using the IEEE 802.16e LDPC code with code rate 1/2 and code length 2304. The simulations are performed on the AWGN channel with BPSK (binary phase shift keying) modulation and maximum 30 iterations. All three DORA algorithms use the same parameters $i_u = 5$ and $t_v = 10$ in the simulations.

DOI: 10.1002/dac

The most critical operations to be reduced in terms of power consumption are read/write accesses to the large likelihood memory. There are four memory-access operations when updating layer APP messages: read/write of APP message memory and read/write of check-to-bit message memory. The read operation of APP message memory is inevitable while the other three operations can be skipped when the nodes are deactivated. Figure 4 shows the memory-access operation reduction compared to SMSA. The results show that DORA-1, DORA-2, and DORA-3 save an average of 32.38%, 21.07%, and 13.45% memory accesses respectively. DORA-1 outperforms the other two because of the saving of memory read for R_{cv} in check-to-bit message memory.

The detailed breakdown of decoding operation saving of DORA algorithms is elaborated below. Based on (1), the node processor needs $|N(c)|$ multibit adding operations to calculate $|N(c)|$ bit-to-check messages. Before calculating (2), $|N(c)| \times 2$ multibit comparing operations are required to find the first and second minimum values among the $|N(c)|$ bit-to-check messages. This part cannot be removed even if our proposed algorithm is applied. Then, $|N(c)|$ multibit comparing operations, $|N(c)| \times 2$ 1-bit XOR operations, and $|N(c)|$ multibit multiplying operations are required to obtain the magnitude and the sign of the check-to-bit messages. Finally, based on (3), $|N(c)|$ multibit adding operations are used to calculate the $|N(c)|$ layer APP messages. The multibit comparison is generally as complex as an adding operation. In addition, the one-bit XOR operations are negligible because they are much less complex than multibit adding operations. Therefore, in summary, for a node processor, 1/8, 1/2, and 1/8 operations are dedicated to calculate (1), (2), and (3) respectively. The other 1/4 operations are dedicated to find the first and second minimum values. Based on this weighting, the total operation reduction can be calculated.

Figure 5 compares the weighted total operation reduced of the three proposed algorithms. The percentage of memory access and node operations reduction increases as the SNR gets higher. Over the range of 1.5 dB to 3.5 dB, 32.38%, 26.49%, and 16.84% operations can be reduced by DORA-1, DORA-2, and DORA-3, respectively. As a reference point, DORA algorithms achieve 54.65%, 45.17%, and 37.73% reduction at SNR = 5.5 dB. Thus, DORA algorithms can save significant power for high signal strength situations. Note that DORA-1 provides operation reduction for $Qtemp_{vc}$, R_{cv} and Q_v computation while DORA-2 and DORA-3 provide operation reduction only for R_{cv} and Q_v computation. DORA-3 has less reduction than DORA-2 for both memory access and decoding operations because of the fact that $Qtemp_{vc}$ is likely to be smaller than Q_v.

Finally, Figure 6 compares the coding gain and average number of iterations of three DORA algorithms. As shown in Figure 6(a), the performance of DORA-3 is almost identical to SMSA. The performance degradation of DORA-1 and DORA-2 is less than 0.1 dB at BER $= 10^{-7}$. More aggressive parameters can be used for DORA-3 to obtain better results in operation reduction. Our low complexity coding gain compensation scheme significantly improves the error floor performance of the DORA decoders. LDPC decoder power consumption is proportional to the average

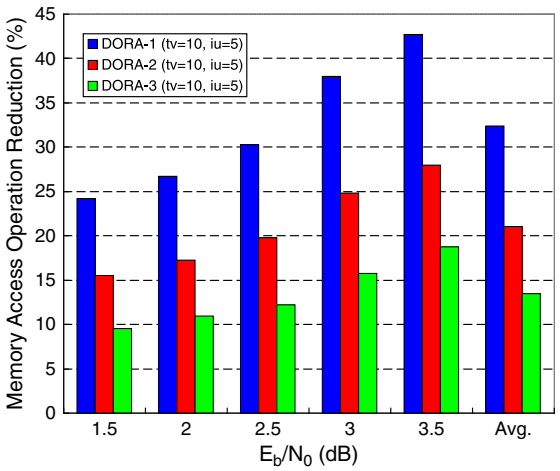


Figure 4. Memory-access operation reduction (%) of DORA decoders.

DOI: 10.1002/dac

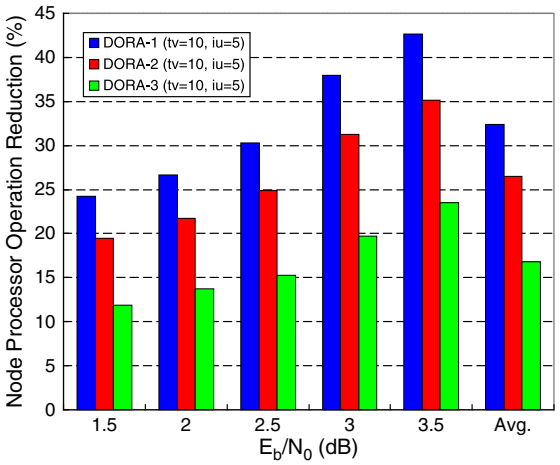
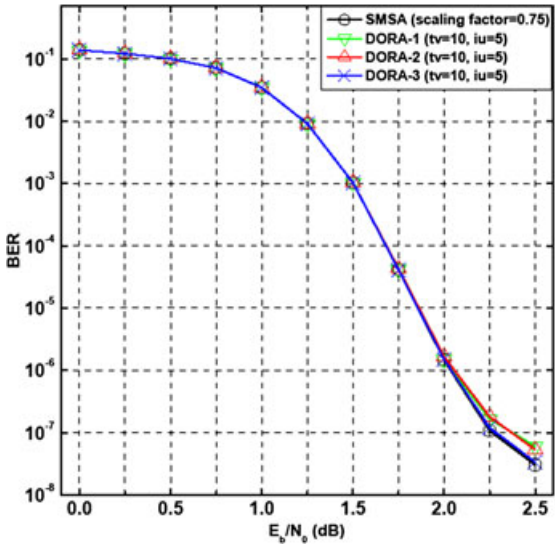
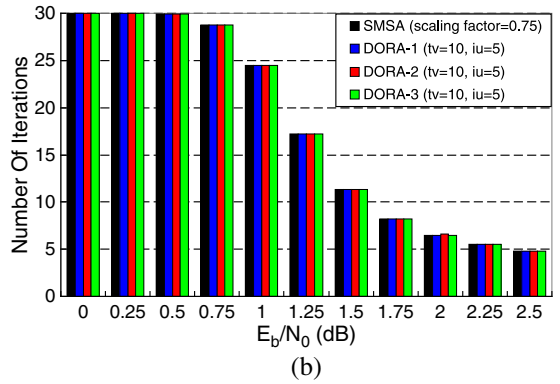


Figure 5. Node processor operation reduction (%) of DORA decoders.



(a)



(b)

Figure 6. Comparison of proposed algorithms: (a) BER and (b) average number of iterations.

number of iterations required to successfully decode one codeword. Figure 6(b) shows that the average number of decoding iterations of the proposed DORA algorithms is virtually identical to SMSA. Thus the convergence speed of DORA has no negative impact on the overall operation reduction.

DOI: 10.1002/dac

4. PROPOSED DORA DECODER ARCHITECTURE

4.1. Architecture overview

We have implemented all three DORA decoding algorithms for the IEEE 802.16e LDPC code with code rate 1/2 and code length 2304. The parity check matrix is composed of size 96×96 zero matrices and identity matrices with various cyclic shifts. The block diagram of the proposed partially parallel decoder architecture is illustrated in Figure 7. The decoder is designed to have 96 node processors to achieve the throughput specified in the IEEE 802.16e standard. The node processor consists of the bit-to-check message processing unit (BPU), the minimum selection unit (MSU), and the check-to-bit message processing unit (CPU). To implement the proposed algorithm DORA-1, the node processor is modified to add a DORA decision unit (DU), two-stage registers, and a FIFO. As shown in Figure 7, the DORA logic is marked with dotted lines. The DORA logic can be easily added with only minor modification in the overall decoder architecture.

The proposed architecture is described below. In the beginning of the decoding process, the channel messages are read from the I/O interface controller to the APP message memory. The control unit (CU) then generates the specific addresses according to the matrix **H** described in ROMs to read channel messages as initial layer APP messages. The rotators route the messages to the APP message memory according to the shifting quantities of **H**. In each node processor, the BPU gets bit-to-check messages by subtracting the received check-to-bit messages from the channel messages. Then the MSU selects the minimum and second minimum values among the input bit-to-check messages. The magnitudes and indices of these two minimum values are sent to the CPU. The CPU uses the information from MSU and bit-to-check messages stored in the FIFO to produce the new check-to-bit messages and corresponding layer APP messages. The layer APP message is circularly shifted by the right rotator before writing back to the original check-to-bit message memory.

The low-complexity DORA DU compares the APP magnitude with a specific threshold t_v to determine if the node processor should be frozen. When APP magnitude exceeds t_v and global reactivation is not asserted, the node processor will be frozen. This one-bit decision is propagated

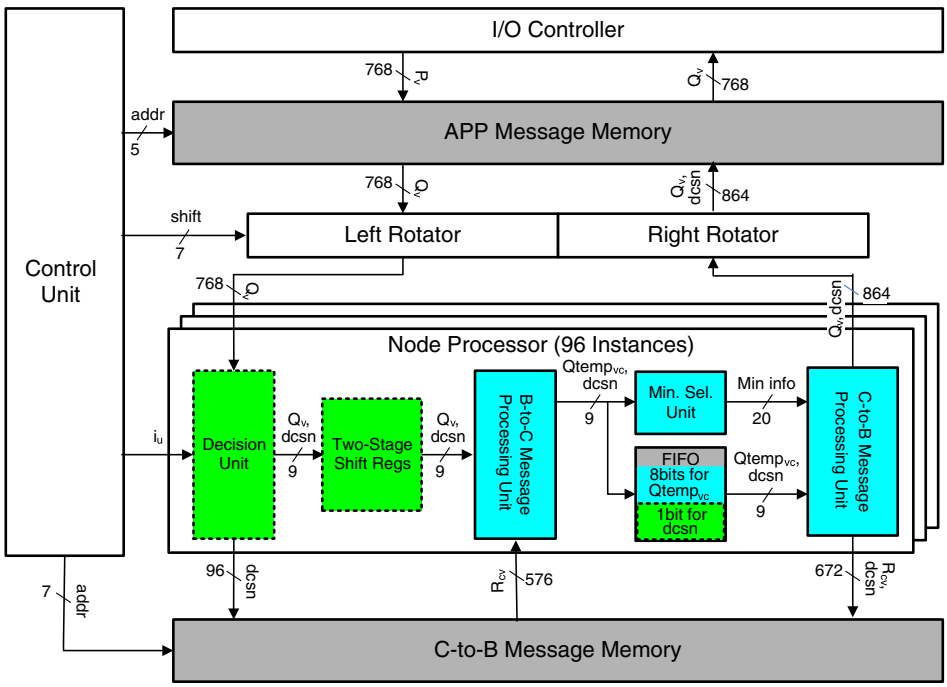


Figure 7. Block diagram of DORA-1.

DOI: 10.1002/dac

along with the layer APP message throughout the node processors to determine whether the processing units or memory accesses should be deactivated. Two stages of shift-registers are used to match the timing of the check-to-bit message memory read accesses. Hence, the DU will result in a 3-cycle delay. The FIFO for bit-to-check messages within each node processor is extended by one bit to transmit the decision. In our design, 96 nine-bit registers are required for 96 parallel node processors.

When DORA freezes the node processor, read/write operation of check-to-bit message memory and write operation of APP message memory can be skipped. More specifically, in BPU, the read operation of check-to-bit messages can be saved. The entire CPU can be disabled because newly updated check-to-bit and layer APP messages are not needed. Hence the check-to-bit and layer APP messages no longer need to be written back into memory. Only the read operation of APP message memory is still necessary and all BPU and CPU operations can be saved. DORA-1 effectively reduces both memory accesses and node processor switching activities.

The memory is segmented into 96 banks for 96 node processors. Each memory bank contains 24 words and each word stores one message corresponding to one of 24 submatrices. If a DU decides not to access memory, the corresponding memory bank will not be enabled. On the basis of our algorithms, almost all banks are accessed in the early iterations but after several iterations, only a small number of banks are still accessed so the power consumption is greatly reduced.

4.2. Low overhead DORA-2 and DORA-3 architecture

The two algorithm variations DORA-2 and DORA-3 can be implemented simply by modifying the node processor of DORA-1. Modified DORA-2 node processor is shown in Figure 8. The shift-registers are removed from DU because the decision result does not impact the read operations of check-to-bit message memory. Hence, the delay of the simplified DU is reduced from 3 cycles to 1 cycle. Because the computation of bit-to-check messages is not simplified from SMSA, the BPU logic overhead can be reduced. Thus, the read access for check-to-bit messages is not saved and the computation of bit-to-check messages is always carried out in full. Only the computation of CPU and write operations of check-to-bit message memory and layer APP message memory is saved in the DORA-2 decoder.

The block diagram of the DORA-3 node processor is shown in Figure 9. The order of the functional units in the node processor is changed. The DU is placed in front of the CPU to make decisions based on bit-to-check messages. The decision result is directly sent into the CPU so the shift-registers can be eliminated. In addition, the FIFO can be removed because the decision result does not need to be propagated. This architecture uses the least hardware overhead with the saving of computation of CPU and write operations of check-to-bit message memory and layer APP message memory.

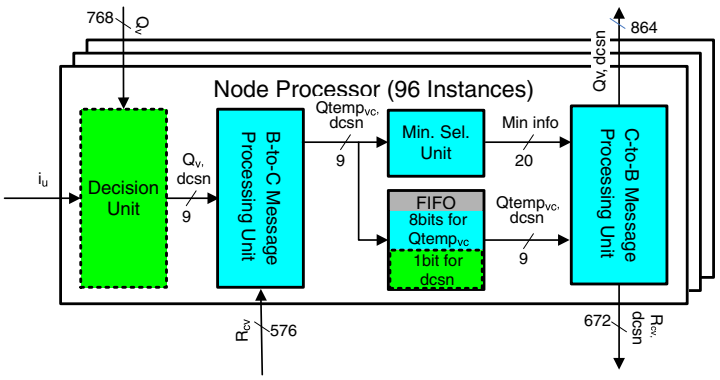


Figure 8. Block diagram of node processor for DORA-2.

DOI: 10.1002/dac

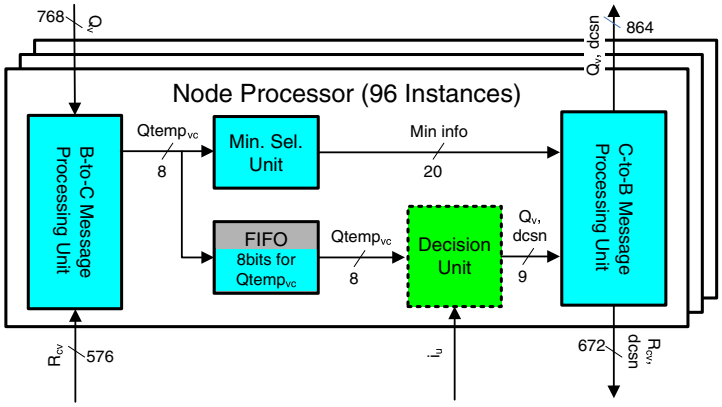


Figure 9. Block diagram of node processor for DORA-3.

5. IMPLEMENTATION RESULTS AND COMPARISONS

5.1. Implementation results

The proposed decoding algorithms are implemented on a Xilinx Virtex5 XC5VLX220T (Xilinx, Inc., San Jose, CA) FPGA device for the (2304, 1152) IEEE 802.16e LDPC code. All three DORA algorithms are implemented together with the original layered SMSA algorithm for comparison. In all the implementations, the channel messages are quantized to 8 bits and check-to-bit messages are quantized to 6 bits.

The FPGA implementation results are summarized in Table III. As shown in the table, the operation reduction logic overhead is relatively small compared with the original layered SMSA decoder. Because of the operation reduction logic, the decoder throughput is reduced slightly. All designs are implemented with single clock and can run up to 140 MHz. On the basis of our simulation results, most times the decoder produces correct codewords well before 8 iterations. The operation reduction with maximum 8 iterations is similar to maximum 30 iterations. Our decoder implementation can achieve 67 to 73 Mbps to meet the 30 Mbps throughput requirement specified in IEEE 802.16e standard. The implementation result is summarized in Table III.

5.2. Comparisons

Table IV summarizes the operation reduction, BER performance, DORA logic overhead, and throughput reduction for the three DORA decoder implementations. DORA-1 reduces the most memory-access and node operations, but the DORA-1 logic overhead is high. The decoder throughput is also reduced by 12%. DORA-2 lowers the logic overhead by a factor of 10 and still offers significant operation reduction with higher throughput. The coding performance of DORA-3 matches the regular SMSA decoder at the cost of lowest operation reduction among three DORA algorithms.

The high reduction DORA-1 algorithm is suitable for LDPC decoders with a lower level of parallelization (a smaller number of node processors) and longer code length (larger message memory). The likelihood message memory dominates the power dissipation and complexity in such a decoder. The higher memory and node processor operation reduction of DORA-1 can effectively lower overall power dissipation. Because of the smaller number of node processors, the DORA-1 logic overhead will occupy only a small portion of the overall complexity. On the other hand, the low

Table III. FPGA implementation results (with maximum 8 iterations).

	SMSA	DORA-1	DORA-2	DORA-3
Clock frequency	140 MHz	140 MHz	140 MHz	140 MHz
Throughput	76 Mbps	67 Mbps	73 Mbps	73 Mbps
Synthesized gate count	222631	348912	236024	236024
Memory size	68,736 bit	69,408 bit	69,408 bit	68,736 bit

DOI: 10.1002/dac

Table IV. IEEE 802.16e DORA LDPC decoder performance comparison.

	DORA-1	DORA-2	DORA-3
Avg. memory operation reduction	26.00%	16.89%	10.44%
Avg. node operation reduction	20.77%	23.90%	8.33%
Coding gain @ BER = 10^{-7}	− 0.1 dB	− 0.1 dB	−0.0 dB
DORA logic overhead	56.72%	6.02%	6.02%
Memory overhead	0.98%	0.98%	0%
Throughput reduction	−12.1%	−4.3%	−4.3%

complexity DORA-2 and DORA-3 algorithms are suitable for high throughput and highly parallel decoders because of its lower logic overhead and throughput reduction. In terms of error correcting performance, all three DORA algorithms can match the regular SMSA algorithm up to BER $= 10^{-7}$. DORA-3 offers the best error floor performance for applications with high coding gain requirement. These three DORA algorithms offer designers great flexibilities and trade-offs in low-power LDPC decoder design.

Table V compares our three DORA designs with two other works that reduce operations for LDPC decoders. The (3,6)-regular LDPC decoder in [21] is based on the two-phase sum–product algorithm. Their use of a more complicated sum–product algorithm increases overall complexity and power consumption; 25% memory read operations and 32% memory write operations are saved with a 0.6 dB performance loss at a BER of 10^{-5}. The area of the node processors in the decoder is increased by more than 10% after applying their threshold algorithm. In the IEEE 802.11n decoder [23], the threshold approach is used to reduce the intermediate memory-access operations by 5% to 37% at the SNR range from 3.2 dB to 4.8 dB. Layered min-sum algorithm is used in their design. The hardware overhead is not mentioned. The node processor operations are not reduced in both designs.

Compared with these two designs, our DORA algorithms can reduce both memory-access and node processor operations. For high throughput and highly parallel decoders, the additional reduction of node processor operations can significantly lower the power consumption. DORA-1 can achieve higher memory and node operation reduction than [21, 23] at the cost of higher node processor overhead. DORA-2 and DORA-3 have lower logic overhead than [21] while offering both memory and node processor operation reduction with much better error correcting performance. The coding gain of the proposed algorithms is superior at high SNR thanks to our coding gain compensation scheme.

6. CONCLUSION

In this paper, we proposed three decoding operation reduction algorithms for LDPC codes. The modified min-sum layered decoding algorithm adaptively freezes active node operations to reduce decoder power consumption. Both memory-access and node processor operations are reduced to maximize power saving. The processing node complexity, number of active nodes, and average number of iterations are jointly optimized to lower total operations. Our low complexity coding gain compensation scheme periodically reactivates the node processors to minimize coding performance degradation. Simulation results show that our algorithm reduces the number of the most power-consuming memory-access operations up to 43% compared to the original layered decoding algorithm. The percentage of operation reduction improves as the SNR gets higher. Thus, DORA algorithms can achieve significant power reduction for high signal strength situations. The convergence speed of the proposed DORA algorithms stays the same as the original SMSA decoder. FPGA implementations of the proposed algorithms on an IEEE 802.16e partially parallel decoder are presented. The DORA algorithms can be easily applied to existing LDPC decoder designs. DORA-1 offers the highest operation reduction and is suited for application on decoders with lower number of node processors and longer code lengths. The low-cost DORA-2 and DORA-3 are suitable for highly parallel, high throughput decoders with a large number of node processors. Three proposed DORA algorithms offer good combination of operation reduction, complexity and error correcting performance for a wide range of LDPC decoder architectures.

DOI: 10.1002/dac

Table V. Comparison with other works.

	[21]	[23]	DORA-1	DORA-2	DORA-3
Supported LDPC code	(3,6)-regular	IEEE 802.11n		IEEE 802.16e	
Decoding algorithm	Two-phase sum-product	Layered min-sum		Layered min-sum	
Operation reduction Technique	Threshold for APP	Threshold for bit-to-check messages	Threshold for APP	Threshold for APP	Threshold for bit-to-check messages
Memory operation reduction	25% read, 32% write at BER = 10^{-5}	5%–37% at SNR = 3.2–4.8dB	24%–43% at SNR = 1.5–3.5dB	16%–28% at SNR = 1.5–3.5dB	9%–19% at SNR = 1.5–3.5dB
Node operation reduction	No	No	24%–43% at SNR = 1.5–3.5dB	19%–35% at SNR = 1.5 − −3.5dB	12%–24% at SNR = 1.5–3.5dB
Node processor overhead	13.25%	N/A	56.72%	6.02%	6.02%
Memory overhead	n bits	$> n$ bits	0.98% (n/24 bits)	0.98% (n/24 bits)	0%
Performance degradation at BER = 10^{-5}	−0.6dB	−0.044dB	−0.0dB	−0.0dB	−0.0dB

DOI: 10.1002/dac

REFERENCES

1. Gallager RG. Low-density parity-check codes. *IEEE Transactions on Information Theory* 1962; **8**:21–28.
2. MacKay DJC, Neal RM. Good codes based on very sparse matrices. *Proceedings of the IMA Conference Cryptography and Coding*, 1995; 100–111.
3. Digital video broadcasting (DVB); second generation. In ETSI EN 302 307 v1.1.1, 2005.
4. IEEE Draft Standard for Information Technology–Telecommunications and information exchange between system–Local and metropolitan area network–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 5: Enhancements for Higher Throughput. IEEE Unapproved Draft Std P802.11n/D7.0, 2008.
5. IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems, IEEE Std 802.16-2009 (Revision of IEEE Std 802.16-2004). 2009.
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-2006, 2006.
7. Zimmermann E, Pattisapu P, Bora PK, Fettweis G. Reduced complexity LDPC decoding using forced convergence. *Proceedings of the 7th International Symposium on Wireless Personal Multimedia Communications (WPMC 2004)*, Italy, Sept. 2004.
8. Tanner R. A Recursive Approach to Low Complexity Codes. *IEEE Transactions on Information Theory* 1981; **27**:533–547.
9. Heo J. Analysis of Scaling Soft Information on Low Density Parity Check Code. *Electronics Letters* 2003; **39**:219–221.
10. Kschischang FR, Frey BJ, Loeliger HA. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory* 2001; **47**:498–519.
11. Sharon E, Litsyn S, Goldberger J. An Efficient Message-Passing Scheduling for LDPC Decoding. *Proceedings of the 23rd IEEE Convention in Tel-Aviv*, 2004; 223–226.
12. Hocevar DE. A reduced complexity decoder architecture via layered decoding of LDPC codes. *Proceedings of the IEEE Workshop on Signal Processing Systems (SIPS 2004)*, Oct. 2004; 107–112.
13. Goldberger J, Kfir H. Serial schedules for belief-propagation: analysis of convergence time. *IEEE Transactions on Information Theory* 2008; **54**:1316–1319.
14. Kienle F, Wehn N. Low complexity stopping criterion for LDPC code decoders. *Proceedings of the IEEE 61th Vehicular Technology Conference (VTC 2005-Spring)*, June 2005; 606–609.
15. Shin D, Heo K, Oh S, Ha J. A stopping criterion for Low-Density Parity-Check codes. *Proceedings of the IEEE 65th Vehicular Technology Conference (VTC2007-Spring)*, Apr. 2007; 1529–1533.
16. Levin D, Sharon E, Litsyn S. Lazy Scheduling for LDPC Decoding. *IEEE Communication Letters* 2007; **11**:70–72.
17. Zimmermann E, Rave W, Fettweis G. Forced convergence decoding of LDPC codes - EXIT chart analysis and combination with node complexity reduction techniques. *Proceedings of the 11th European Wireless Conference (EW 2005)*, Apr. 2005.
18. Blad A, Gustafsson O, Wanhammar L. An early decision decoding algorithm for LDPC codes using dynamic thresholds. *Proceedings of European Conference on Circuit Theory Design (ECCTD 2005)*, Aug. 2005.
19. Blad A, Gustafsson O, Wanhammar L. A hybrid early decision-probability propagation decoding algorithm for low-density parity-check codes. *Proceedings of Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, Oct.–Nov. 2005; 586–590.
20. Zimmermann E, Pattisapu P, Fettweis G. Bit-flipping post-processing for forced convergence decoding of LDPC codes. *Proceedings of the 13th European Signal Processing Conference (EUSIPCO 2005)*, Sept. 2005.
21. Blad A, Gustafsson O, Wanhammar L. Implementation aspects of an early decision decoder for LDPC codes, Nov. 2005.
22. Blad A. Early-decision decoding of LDPC codes. *Ph.D. dissertation*, Linköping University, Linköping, Sweden, 2009.
23. Jin J, Tsui C. A low power layered decoding architecture for LDPC decoder implementation for IEEE 802.11n LDPC codes. *Proceedings of the IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED 2008)*, Aug. 2008; 253–258.

DOI: 10.1002/dac

AUTHORS' BIOGRAPHIES

Chia-Yu Lin received the B.S. degree in Computer Science and Information Engineering from the National Cheng Kung University, Taiwan in 2004. He received the Ph.D degree in Computer Science and Information Engineering from the National Taiwan University, Taiwan in 2011. His research mainly focuses on error correcting codes, especially low-density parity-check codes and their applications.

Shu-Cheng Chou received the M.S. degree in Computer Science and Information Engineering from the National Taiwan University, Taipei, Taiwan in 2008. He is currently working in MStar Semiconductor as a senior engineer.

Mong-Kai Ku was born in Taipei, Taiwan, on September 12, 1967. He received the B.S. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, in 1989. He received the M.S. and Ph.D degrees in electrical engineering from the University of California, Los Angeles, in 1994 and 1997, respectively. He joined Broadcom Corporation in 1993, where he was involved in many high speed communication IC design projects, with a focus in the forward-error-correction system design. Since February 2003, he has been with National Taiwan University, where he is currently an assistant professor in the department of computer science and information engineering. His research interests include high-speed communication systems, coding theory, and VLSI design.

DOI: 10.1002/dac