# Error Resilience and Energy Efficiency: An LDPC Decoder Design Study

Philipp Schläfer*, Chu-Hsiang Huang†, Clayton Schoeny†, Christian Weis*, Yao Li‡, Norbert Wehn*, Lara Dolecek†
* Microelectronic Systems Design Research Group, University of Kaiserslautern, Kaiserslautern, Germany
{schlaefer, weis, wehn}@eit.uni-kl.de
† Department of Electrical Engineering, University of California, Los Angeles, California
{seanhuang0522, cschoeny}@ucla.edu, dolecek@ee.ucla.edu
‡ Akamai Inc., Pasadena, California
yli@akamai.com

***Abstract*—Iterative decoding algorithms for low-density parity check (LDPC) codes have an inherent fault tolerance. In this paper, we exploit this robustness and optimize an LDPC decoder for high energy efficiency: we reduce energy consumption by opportunistically increasing error rates in decoder memories, while still achieving successful decoding in the final iteration. We develop a theory-guided unequal error protection (UEP) technique. UEP is implemented using dynamic voltage scaling that controls the error probability in the decoder memories on a per iteration basis. Specifically, via a density evolution analysis of an LDPC decoder, we first formulate the optimization problem of choosing an appropriate error rate for the decoder memories to achieve successful decoding under minimal energy consumption. We then propose a low complexity greedy algorithm to solve this optimization problem and map the resulting error rates to the corresponding supply voltage levels of the decoder memories in each iteration of the decoding algorithm. We demonstrate the effectiveness of our approach via ASIC synthesis results of a decoder for the LDPC code in the IEEE 802.11ad standard, implemented in 28 nm FD-SOI technology. The proposed scheme achieves an increase in energy efficiency of up to 40% compared to the state-of-the-art solution.**

## I. Introduction

With increasing integration density in nanometer CMOS technology, devices become more susceptible to various dependability issues. In particular, memories suffer from severe reliability problems [1]. The increase in memory failure rates is due to several factors, including shrinking dimensions, high integration densities, and lower operating voltages [2].

In many applications, not all bits stored in the memories (or components in the system) are equally significant in determining the output quality. *Unequal error protection* (UEP) was introduced to provide different degrees of protection for bits/components which differ in importance to the output quality [3]. UEP techniques were previously developed for image transmission/storage systems [3]. Recently, UEP techniques were applied in robust system design for unreliable hardware. In this work, we focus our attention on widely popular low-density parity-check (LDPC) decoders (cf. Fig. 1). Several works on noisy LDPC decoders have already applied UEP to design robust decoders. In these robust decoders, only the sign bits of variable node messages ($\tilde{m}_{v,c}$ in Fig. 1) were protected since errors in these bits can have a large impact on the decoder output error rate [4], [5]. Furthermore, in [6], [7], the hardware

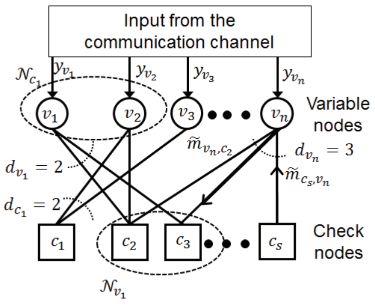

Fig. 1. An LDPC code (and decoder) example. $d_v$ ($d_c$) are variable (check) node degrees, and $\tilde{m}_{v,c}$ ($\tilde{m}_{c,v}$) are variable (check) node messages.

errors in the variable nodes with different degrees ($d_v$'s in Fig. 1) were shown to have different effects on the residual error rate. The design of protection levels for different variable nodes was formulated as a linear programming problem in [6], [7]. UEP for a polar code decoder was studied in [8]; protecting the computations in the last few decoding iterations was shown to be sufficient to improve the overall decoder performance. The result in [8] is consistent with a general feature of iterative decoders: for well-designed decoders, the errors introduced in earlier iterations are typically corrected by later iterations. Therefore, UEP across different iterations in the iterative decoder is expected to bring significant improvement (in terms of energy efficiency) without compromising the final output error rate. In this paper, based on the analysis of the LDPC decoder message error rate at each iteration, we propose a theory-guided design methodology for UEP across different iterations under the constraint of successful decoding in the final iteration, and demonstrate the energy savings relative to a state-of-the-art architecture.

We briefly review LDPC decoding theory in Section II. Section III overviews LDPC decoder hardware architecture and presents synthesis results. In Section IV, we introduce a memory error model and analyze the message error rate at each iteration based on this model. According to the synthesis results in Section III, our memories can have different protection levels, i.e., the decoder memories can operate at different error rates. This feature is achieved by employing dynamic voltage scaling (DVS) of the supply voltage across decoding iterations [9], [10]. Via a density evolution analysis, we derive the message error rate at each iteration as a function of both the memory error rates in all up-to-date iterations and the parameters of the communication channel that produces a noisy codeword at the input to the decoder. Based on this function, we propose a decoder design that minimizes the energy consumption for successful decoding: a suitable choice of the memory error rate (determined by the supply voltage value) is chosen at each decoding iteration. Qualitatively, the energy savings are achieved by using memories with low protection levels (high error rates) in the first few iterations, and higher protection levels (lower error rates) in the last few iterations. In Section V, we report on the experimental results and the amount of energy saved by our decoder design.

## II. LDPC Decoding Theory

In this section, we briefly review LDPC codes and the min-sum decoding algorithm.

An LDPC code is a linear code defined by a sparse binary parity check matrix $H$. A valid codeword $x$ satisfies the constraint $Hx^T = 0 \mod 2$ (also known as syndrome checking). This LDPC code can be conveniently represented by a sparse bipartite graph, a graphical representation of the parity check matrix $H$. The bipartite graph is composed of variable nodes (coded bits), check nodes (check-sum constraints), and edges each connecting a variable node to a check node. The edges correspond to '1' entries in the matrix $H$. Let $\mathcal{N}_v$ and $\mathcal{N}_c$ denote the nodes connected to the variable node $v$ and to the check node $c$, respectively, on a bipartite graph associated with the LDPC code. An LDPC code example is shown in Fig. 1.

LDPC codes are iteratively decoded by exchanging messages between variable nodes and check nodes. In this paper, we consider the min-sum algorithm, which is widely applied in many communication systems with good performance [11]. Input to the decoder is a codeword passed though a noisy communication channel; $y_v$ is the decoder input at the variable node $v$. Denote the (error-free) messages sent in the $\ell$th iteration by $\tilde{m}^{(\ell)}$. More specifically, $\tilde{m}^{(\ell)}_{v,c}$ denotes the message sent from variable node $v$ to its incident check node $c$, while $\tilde{m}^{(\ell)}_{c,v}$ denotes the message passed from check node $c$ to its incident variable node $v$. Since the messages are log-likelihood ratios, the sign of a message indicates the decision of the associated coded bit made by this message, and the magnitude of a message represents the certainty of the decision. We summarize the message exchange and update steps of the min-sum decoding algorithm in the following [11].

- (Initialization) Each variable node $v$ sends the message $\tilde{m}^{(0)}_{v,c} = \ln \frac{P(y_v|x_v=0)}{P(y_v|x_v=1)}$ to each check node $c$, $c \in \mathcal{N}_v$ at iteration $\ell = 0$, where $x_v$ is the transmitted bit corresponding to the variable node $v$.
- (Check node) Each check node $c$ sends a message $\tilde{m}^{(\ell)}_{c,v}$ to each variable node $v$, $v \in \mathcal{N}_c$ :
$$\tilde{m}^{(\ell)}_{c,v} = \Big( \prod_{v' \in \mathcal{N}_{c \setminus \{v\}}} \operatorname{sign}(\tilde{m}^{(\ell)}_{v',c}) \min_{v' \in \mathcal{N}_{c \setminus \{v\}}} |\tilde{m}^{(\ell)}_{v',c}| \Big)$$
at each iteration $\ell$, $\ell \geq 0$.
- (Variable node) Each variable node $v$ sends a message $\tilde{m}^{(\ell)}_{v,c}$ to each check node $c$, $c \in \mathcal{N}_v$ :
$$\tilde{m}^{(\ell)}_{v,c} = \tilde{m}^{(0)}_{v,c} + \sum_{c' \in \mathcal{N}_v \setminus \{c\}} \tilde{m}^{(\ell-1)}_{c',v}$$
at each iteration $\ell$, $\ell \geq 1$.
Each variable node also computes a sum message, capturing the a posteriori probability (APP) of the corresponding bit,
$$\tilde{m}^{(\ell)}_{v} = \tilde{m}^{(0)}_{v} + \sum_{c \in \mathcal{N}_v} \tilde{m}^{(\ell-1)}_{c,v}$$
at each iteration $\ell$, $\ell \geq 1$.

The decoding algorithm terminates when all the parity check constraints are satisfied or a maximum number of iterations, $L_{max}$, is reached.

## III. LDPC Decoder Hardware

In this section, we present the used LDPC decoder architecture and ASIC synthesis results.

### A. LDPC Decoder Architecture

State-of-the-art LDPC decoder architectures are based on structured codes. We construct the parity check matrix $H$ using square submatrices of size $P \times P$, which are either a cyclically shifted identity matrix or an all-zeros matrix. This structure allows for the efficient processing of the LDPC codes.

The hardware implementation of LDPC decoders is performed using serial, partially parallel, or fully parallel methods. Serial architectures can be very flexible in supporting numerous codes; however, such a design cannot achieve the throughput required for modern applications. Partially parallel decoders instantiate a subset of variable and check nodes as functional units in hardware. A multiple of the submatrix size $P$ is chosen as the degree of parallelism which allows for efficient architectures. Fully parallel architectures instantiate all nodes of the Tanner graph simultaneously. The large area, routing congestions, and lack of flexibility make this approach prohibitive for most modern applications.

Partially parallel architectures give the best tradeoff between flexibility, throughput, and area efficiency. An exhaustive overview of state-of-the-art LDPC decoder architectures is given in [12]. Most LDPC decoders used in commercial designs are based on partially parallel architectures. Thus, as a showcase for our further investigations, we chose a slot-layered, partially parallel approach which has a small area footprint and allows for relatively high throughput [13].

Using this architecture, we implemented a decoder for the LDPC code used in the IEEE 802.11ad standard. The code has a length of 672 bits and code rate 0.81. More details about the specific architecture can be found in [13] and [14]. The top level decoder architecture is shown in Fig. 2. In this code, $P$ is 21. Three slots of 21 variable node functional units (VFUs) feed messages to 21 check node functional units (CFUs). For an efficient implementation, the VFUs and CFUs are merged into check node blocks (CNBs). The network between CFU and VFU (described by $d_v, d_c, \mathcal{N}_v$ and $\mathcal{N}_c$ defined in Section II, and with $d_v \in \{1, 2, 3\}$ and $d_c \in \{14, 15, 16\}$

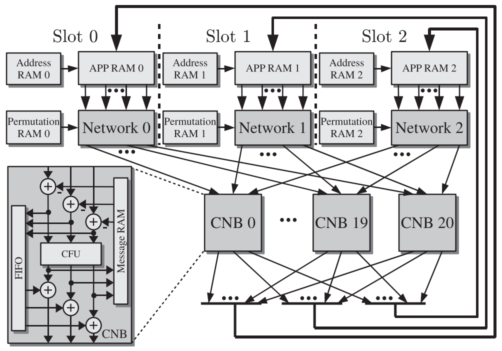

Fig. 2. Slot-Layered LDPC decoder hardware architecture.

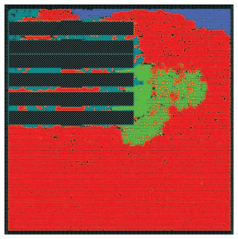

Fig. 3. LDPC decoder chip photo. The dark green blocks show the APP RAM and directly associated logic, red the actual decoding logic (CNBs), light green the top controller, and blue the interface logic.

for this code) is implemented by use of cyclic barrel shifters. A single network approach, using offset shifts is chosen to simplify routing in the design. The main memory of the architecture is the RAM for the APP values $\tilde{m}_v^{(\ell)}$ of the VFU, see Section II for $\tilde{m}_v^{(\ell)}$ computations. The APP RAM is implemented with an SRAM block in the ASIC architecture. We consider this memory block to be erroneous, and show how one can take advantage of the underlying robustness of the decoding algorithm to overcome memory errors.

Memory error rates are adjusted using dynamic voltage scaling applied to the supply voltage. How to choose the supply voltage level per decoding iteration will be discussed in Section IV-C. Since the chosen voltage levels increase as the iterations progress, we only need $K-1$ switches for $K$ voltage levels. The switching can be performed within one or two clock cycles, thus inducing only a small overhead.

### B. Synthesis Results

The previously presented design was implemented on a 28 nm low power FD-SOI CMOS library. Synthesis as well as P&R were performed with the Worst Case PVT settings of the 28 nm library. The target frequency for the design was fixed at 220 MHz. However, to make the design work under reduced voltage schemes, the target frequency for synthesis was chosen significantly higher to provide voltage margin. The physical layout for the decoder is shown in Fig. 3.

Table I shows area, throughput (TP), and energy efficiency for all characterized voltage working points. Energy efficiency normalizes power consumption to throughput and allows for a fair comparison of different operating points. In our DVS scheme, the considered operating voltages range from 0.7 V to 1.0 V where 1.0 V is the nominal operation point for the used technology. As expected, reducing the supply voltage from 1.0 V to 0.7 V approximately doubles the energy efficiency.

TABLE I
LDPC DECODER ASIC SYNTHESIS RESULTS

| Post P&R Area [mm$^2$] | 0.142 | | | |
|---|---|---|---|---|
| V$_{supply}$ [V] | 1.0 | 0.9 | 0.8 | 0.7 |
| F$_{max}$ [MHz] | 600 | 500 | 364 | 228 |
| TP @ 10 iterations [Mbit/s] | 790 | 659 | 474 | 290 |
| Energy Eff. [pJ/bit/iter] | 9.89 | 7.84 | 6.11 | 4.82 |

TABLE II
MEMORY ERROR RATES FOR DIFFERENT SUPPLY VOLTAGES

| V$_{supply}$ [V] | 1.0 | 0.9 | 0.8 | 0.7 |
|---|---|---|---|---|
| SER [BER/s] | $2.2\times10^{-11}$ | $4.3\times10^{-11}$ | $8.7\times10^{-11}$ | $1.5\times10^{-10}$ |
| RWER [BER/access] | $1.0\times10^{-15}$ | $8.0\times10^{-13}$ | $3.7\times10^{-7}$ | $3.0\times10^{-3}$ |

## IV. EXPLOITING THE ERROR RESILIENCE

In this section, we introduce a noisy decoder model and a density evolution analysis. Based on the density evolution analysis, we propose a UEP decoder design methodology.

### A. Noisy Decoder Model

In real-world circuits, numerous sources of errors exist. An extensive overview of possible sources is given in [2], [15].

At very small processes, such as 28 nm technology, there is a nontrivial probability of failure. For memories, the probability of an error increases exponentially with reduced voltage. However, the critical path of a combinational circuit is built from a number of serially operating logic cells. The variability of weak (i.e., slow) logic cells is averaged in the critical path. Hence it is less likely that failures appear in the combinational part of a circuit. Thus we focus on SRAM errors.

For our investigation, we considered two different kind of error sources:

- Soft Errors: Soft Errors result from signal corruption, such as a particle strike changing the content of a storage element. Reducing the operating voltage reduces the charge required to upset a storage cell, known as the critical charge, and makes the cell more vulnerable to Soft Errors. The Soft Error Rate (SER) is expressed as Bit Error Rate (BER) per second.
- Read/Write Errors: An SRAM cell is specified to be read/written within a specified time. If the memory cell cannot perform the operation within this defined time, an error occurs. The corresponding Read/Write Error Rate (RWER) is expressed as BER per access.

Memories with different supply voltages consume different amounts of energy per iteration (cf. Section III-B). A lower memory error rate implies higher energy consumption. The corresponding error rates for the specified error types were derived from [2], [15], [16], and are given in Table II.

The bit-flipping model is widely used in fault-tolerant system design to link physically induced faults from the lower level (hardware) to the higher level (algorithm) [4], [17]. In the bit-flipping model for faulty memories, the value retrieved from a memory cell at read time differs from the original value stored in the memory cell with a probability larger than zero

[4], [18]. Following the modeling methodology proposed in Stochastic Computation [19] and assuming that bit flips are equally probable in both directions (1 to 0 and 0 to 1), we interpret the bit-flipping model for a faulty memory cell as passing a binary input through a binary symmetric channel (BSC) with cross-over probability $\rho$. The probability $\rho$ is the sum of the rates of the two error types (Table II). Following the bit-flipping model used in [4], [18], we assume the BSCs modeling the memory errors are independent.

In this paper, we assume that all the memory cells in our decoder have the same bit-flipping probability (error rate) in the same iteration; however, in different iterations, the memory cells can have different protection levels (governed by the choice of the voltage levels), and can therefore operate at different error rates. We denote the memory error rate at iteration $\ell$ by $\rho^{(\ell)}$, which is a function of the iteration number $\ell$. Previous works have shown that hardware errors in check nodes do not have much impact on the residual error rates of the LDPC decoders, [4], [20]. Therefore we consider only the APP RAM storing the sum messages ($\tilde{m}_v$) to be erroneous. This APP RAM is the only memory which is implemented as SRAM in the presented decoder architecture.

## B. Density Evolution Analysis

Our goal is to provide a theory-guided UEP decoder design methodology based on the decoder memory error model in Section IV-A. We first derive the decoder message error rate in each decoding iteration as a function of memory error rate $\rho^{(\ell)}$. In the next section, we determine the optimal protection level for memories in each decoding iteration based on the message error rate analysis. When we study the average decoder message error rate and consider the decoder input from the communication channel as random variables, the messages passed at each round of the min-sum decoding algorithm are random variables [21]. Density evolution analysis is a well-known technique for analyzing the probability mass function (PMF) of the average decoder message in each iteration. In a density evolution analysis, the transmitted codewords are assumed to be all-zero, and the (average) message error rate in each iteration is the probability of the message value being negative, which can be derived from the message PMF [21]. Density evolution analysis of the noisy finite precision min-sum decoder with the same memory error rate for all decoding iterations was derived in [5], [18].

For mathematical convenience, we also specify the idealized quantized variable messages at iteration $\ell$ by $\hat{m}_{v,c}^{(\ell)}$. With the faulty memory cells described above, the actual variable messages retrieved from the memory cells, denoted by $m_{v,c}^{(\ell)}$, might be different from $\hat{m}_{v,c}^{(\ell)}$.. Note that the $\hat{m}$ messages themselves are not available since the relationship between $\hat{m}$'s and $m$'s is probabilistic, as described by the bit-flipping model specified earlier.

Following the density evolution analysis in [5], [18], we have the recursive expression for the PMF of $m_{v,c}^{(\ell)}$ in the min-sum decoder with the above hardware error model:

$$p^{(\ell)} = A(\rho^{(\ell)})\hat{p}^{(\ell)}, \tag{1}$$

where $p^{(\ell)}$ is a length-$2^b$ vector representing the PMF of $m_{v,c}^{(\ell)}$, $\hat{p}^{(\ell)}$ is a length-$2^b$ vector representing the PMF of $\hat{m}_{v,c}^{(\ell)}$, and $A(\rho^{(\ell)})$ is a $2^b \cdot 2^b$ matrix whose entries are functions of $\ell$. Note that $b$ is the number of bits in the two's complement representation of each message. Detailed derivation of $\hat{p}^{(\ell)}$ can be found in [18]. The matrix $A(\rho^{(\ell)})$ transforms the PMF of the messages $\hat{m}_{v,c}^{(\ell)}$ (before the memory errors are applied) to the PMF of the messages $m_{v,c}^{(\ell)}$ (after the memory errors are applied). Since the memory errors occur independently across memory cells, we derive the matrix $A(\rho^{(\ell)})$ based on a binomial distribution as follows. The derivation is the same as the derivation of the perturbation matrix in [5], but here the memory error rate is a function of the iteration number $\ell$. Let $\{\alpha_i\}_{i=1}^{2^b}$ be the set of possible values of messages $\hat{m}_{v,c}^{(\ell)}$ and $m_{v,c}^{(\ell)}$. The range of $\alpha_i$ is determined by the two's complement representation we choose. For example, when we choose a two's complement representation with length 5, the range of $\alpha_i$ is the integers from -16 to 15. Let the $i$th entry in $p^{(\ell)}$ denote the probability of the event $m_{v,c}^{(\ell)} = \alpha_i$, $1 \leq i \leq 2^b$. Similarly, let the $i$th entry in $\hat{p}^{(\ell)}$ denote the probability of the event $\hat{m}_{v,c}^{(\ell)} = \alpha_i$. The $(i, j)$ entry in $A(\rho^{(\ell)})$, $A_{i,j}(\rho^{(\ell)})$, denotes the conditional probability of the event $m_{v,c}^{(\ell)} = \alpha_i$ given $\hat{m}_{v,c}^{(\ell)} = \alpha_j$. We then have

$$\begin{aligned} A_{i,j}(\rho^{(\ell)}) &= P(m_{v,c}^{(\ell)} = \alpha_i | \hat{m}_{v,c}^{(\ell)} = \alpha_j) \\ &= (1-\rho^{(\ell)})^{b-d_{i,j}}(\rho^{(\ell)})^{d_{i,j}}, \end{aligned} \tag{2}$$

where $d_{i,j}$ is the the number of different bits between the two's complement representation of $\alpha_i$ and $\alpha_j$.

The message error rate at iteration $\ell$, $p_r^{(\ell)}$, is derived by summing the entries in $p^{(\ell)}$ corresponding to the negative $\alpha_i$'s, which are by convention the message values leading to the incorrect coded bit value decision.

## C. UEP Decoder Design

When the communication channel error rate is small enough, message passing decoders on average reduce the message error rates at each iteration in the iterative decoding process. When the memory error rate is much smaller than the communication channel error rate, we observe via density evolution analysis [18], [20] that in the first few iterations, the message error rates are close to the communication channel error rate. Increasing the memory error rates in these initial iterations has a negligible effect on the message error rates. A decoder successfully decodes the received signal (the decoded bits pass the syndrome checking step) with a high probability when the message error rate at the final iteration is sufficiently small. Since the message error rate at the final iteration is dominated by the memory error rate (as shown in the density evolution analysis [18], [20]), we can achieve successful decoding with a high probability by operating the memories at a small enough (but larger than zero) memory error rate in the final iteration. Therefore, by dynamically adjusting memory errors rates in each iteration, we are able to decode the received signal from the communication channel with a lower overall energy consumption.

We formulate the protection level design problem as follows, using the bit-flipping model described in Section IV-A. Denote the ordered list of available memory error rates (corresponding to different protection levels) by $\Omega_\rho, \Omega_\rho = (\rho_1, \ldots, \rho_K)$, where $\rho_1 > \ldots > \rho_K \geq 0$, where $K$ is the number of available memory error rates. Denote the energy consumption per iteration corresponding to the memory error rate $\rho_k$ by $E_k$, and denote the ordered list of energy consumption values by $\Omega_E, \Omega_E = (E_1, \ldots, E_K)$, where $E_1 < \ldots < E_K$.

We now show how to compute the memory error rate at each iteration in the decoding process sufficient to successfully decode the input to the decoder with a minimized energy consumption. Instead of targeting zero message error rate by operating at nominal voltages as the traditional decoder, we first set a target error rate $p_f$, where $p_f$ is small enough so that the decoding process terminates with a high probability when the message error rate reaches $p_f$. Then, we formulate an optimization problem of reaching the residual error rate $p_f$ within $L_{max}$ iterations with a minimized energy consumption, using the protection level function $\phi(\ell)$. Note that $\phi(\ell)$ is a function of the iteration number $\ell$ and the output of the function is the number indicating the chosen protection level. Thus,

$$\begin{aligned} &\text{Minimize: } \sum_{\ell=1}^{L} E_{\phi(\ell)}, \\ &\text{Subject to: } L = \min\{\ell | p_r^{(i)} \leq p_f, \forall i \geq \ell\}, L \leq L_{max}, \end{aligned} \quad (3)$$

where $p_r^{(i)}$ is a function of the communication channel noise and $\rho_{\phi(j)}, j \leq i$ (derivation of $p_r^{(i)}$ is discussed in Section IV-B), and $L_{max}$ is the maximum allowable number of iterations. The optimal protection level function $\hat{\phi}(\ell)$ derived by solving this optimization problem is then the optimal protection level for each iteration $\ell$. Note that the constraint $L = \min\{\ell | p_r^{(i)} \leq p_f, \forall i \geq \ell\}$ guarantees that the decoder achieves message error rate $p_f$ at iteration $L$.

When UEP decoder terminates at iteration $L$, the UEP decoder with the optimal protection level function $\hat{\phi}(\ell)$ is guaranteed to achieve minimal energy consumption for successful decoding. However, since the decoding process does not always terminate at iteration $L$, this algorithm does not always guarantee the minimum energy consumption for a successful decoding. But if we choose $p_f$ appropriately so that the decoding process terminates at iteration $L$ with a high probability, the energy consumption is close to its minimum, since we minimize the energy consumption of the decoding process when it terminates at iteration $L$.

The above optimization problem can be solved by exhaustive search, since the sets $\Omega_\rho$ and $\Omega_E$ have finitely many elements. However, the complexity of the exhaustive search grows exponentially with the maximum allowable number of iterations $L_{max}$. We therefore propose a greedy algorithm with lower complexity by observing the following two facts. First, the optimal solution, $\hat{\phi}(\ell)$, is a monotonically increasing function of $\ell$. Since the message error rate is on average decreasing as we proceed in the decoding process, we can use memories with smaller error rates in later iterations. Second, the memory error rate at iteration $L$ (the iteration in which we achieve $p_f$) is on the order of $p_f$. This observation follows from [18] where it was shown that, when $\ell$ is sufficiently large, the message error rate is on the order of (and larger than) the memory error rate. We can thus achieve $p_f$ with a minimized energy consumption by choosing the memory error rate at iteration $L$ on the order of (but smaller than) $p_f$.

TABLE III
UEP DECODER SUPPLY VOLTAGES UNDER DIFFERENT SNRS.

| Iteration/ SNR(dB) | 1 | 2 | 3 | 4 ∼15 | Changes |
|---|---|---|---|---|---|
| 3.3 | 0.75 | 0.75 | 0.8 | 0.8 | 1 |
| 3.5 | 0.75 | 0.8 | 0.8 | 0.8 | 1 |
| 3.7 | 0.75 | 0.8 | 0.8 | 0.8 | 1 |
| 3.9 | 0.75 | 0.8 | 0.8 | 0.8 | 1 |
| 4.1 | 0.75 | 0.8 | 0.8 | 0.8 | 1 |
| 4.3 | 0.75 | 0.8 | 0.8 | 0.8 | 1 |
| 4.3 | 0.8 | 0.8 | 0.8 | 0.8 | 0 |

In this greedy algorithm, we determine the protection level function $\phi(\ell)$ step by step, starting from $\ell = 1$. Let $\xi$ be the index of the memory error rate in $\Omega_\rho$ with the maximum value among all the values in $\Omega_\rho$ that are less than $p_f$. (Recall that the entries in $\Omega_\rho$ are ordered in descending order. ) At iteration $j$, we assume that $\phi(\ell) = \xi$ for all $\ell > j$, and we minimize the total energy consumption by selecting $\phi(j)$. This one-step optimization problem is then formulated as follows:

$$\begin{aligned} &\text{Minimize: } \sum_{\ell=j}^{L} E_{\phi(\ell)}, \\ &\text{Subject to: } L = \min\{\ell | p_r^{(i)} \leq p_f, \forall i \geq \ell\}, \\ &\qquad\qquad \phi(\ell') = \xi, \forall \ell' > j. \end{aligned} \quad (4)$$

We start with $j = 1$ and terminate the algorithm when $j = L$. Using the fact that $\hat{\phi}(j)$ is a monotonically increasing function of $j$, the set of feasible solutions of $\phi(j)$ is just $\{a | a \geq \phi(j-1)\}$, with size $K - \phi(j-1) + 1$, where $K$ is the number of available memory error rates. For $j = 1$, the feasible solution set is $\{1, 2, \ldots, K\}$. We then compute $\phi(j)$ for each $j$ by comparing at most $K$ different summations ($\sum_{\ell=j}^{L} E_{\phi(\ell)}$). Therefore, the number of summations and comparisons grow linearly with $L_{max}$ and $K$, which significantly reduces the algorithm complexity.

Although we consider the noisy min-sum decoder subject to memory errors in this paper, the proposed algorithm applies to any noisy decoder subject to different kinds of hardware errors as long as we are able to derive the density evolution of the noisy decoder, i.e., $p_r^{(\ell)}$ as a function of iteration $\ell$, the hardware error rate, and the communication channel noise.

## V. RESULTS

In this section we compare the energy efficiency of the proposed scheme (referred to as the UEP decoder) with the state-of-the-art decoder (the reference decoder) for our running

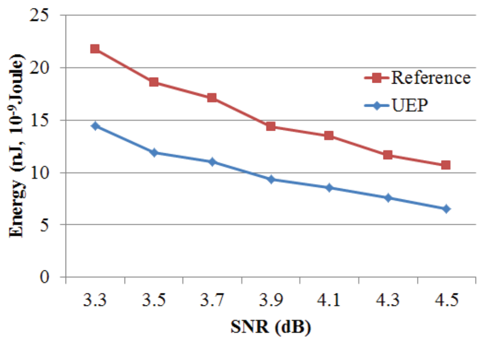

Fig. 4. Average energy consumption for the decoding of a codeword.

example, the IEEE 802.11ad LDPC code. The maximum number of iterations is $L_{max} = 15$.

We use the memory error rates and the corresponding energy consumption numbers listed in Table I and II to derive the optimal protection level function $\hat{\phi}(\ell)$ for different communication channel signal-to-noise ratios (SNRs). We run simulations for 50000 codeword transmissions. From our simulations, we observe that a large fraction of codewords is successfully decoded in the first iterations, in which we use the memories with low energy consumptions and high error rates. As discussed in Section IV-C, although we use memories with a slightly higher error rate than the reference decoder memories (and have a slightly higher message error rate than the reference decoder) for the initial few iterations, the capability of error correction at each decoding iteration and the syndrome checking step still guarantee successful decoding.

We compare the energy efficiency for the two decoders under different SNRs in Fig. 4. As expected, for both decoders, the average energy consumption decreases as we increase the SNR. However, the UEP decoder consumes significantly less energy overall than the reference decoder since it opportunistically employs memories with higher error rates in the initial iterations. For a very small fraction of codewords, the UEP decoder requires one or two more iterations to decode a codeword relative to the nominal decoder. The effect of the added latency on the average energy consumption is very small. The overall energy efficiency is increased by $30\% \sim 40\%$ over the entire SNR region. In the considered SNR range, in most cases, we only need one voltage change in the decoding process (Table III). Even for lower rate code operating in higher SNR, 2 voltage changes are sufficient for most cases (from our WIMAX LDPC code experiments). Therefore, the overhead of the DVS scheme is very small.

In addition to the IEEE 802.11ad LDPC code, we also applied the UEP LDPC decoder to codes with different rates and degree distributions, including the experiments on the LDPC code from the WIMAX standard [4] with code rate 0.5 and code length 2304. Prelimenary experiment results show that with a higher rate code, more voltages levels are used (up to 3 levels in the WIMAX code example) and larger average energy savings by DVS are observed in a wider SNR range (from $0 \sim 4$dB).

## VI. Conclusion

In this paper, we exploited intrinsic error resilience of the min-sum decoder to build a highly energy efficient LDPC decoder architecture. We proposed a greedy algorithm to derive an optimal voltage schedule for the decoding. The synthesis results of an IEEE 802.11ad standard-conforming LDPC decoder increase the energy efficiency by up to 40% compared to the state-of-the-art. Future work will involve combining the proposed method with hardened LDPC decoders.

## Acknowledgment

We gratefully acknowledge financial support by the DFG (project-ID: WE 2442/8-1 and WE 2442/10-1).

## References

[1] P. Nikolaou *et al.*, "Memory array protection: Check on read or check on write?" in *DATE*, 2013.
[2] G. Wirth, M. Vieira, E. Neto, and F. Kastensmidt, "Generation and Propagation of Single Event Transients in CMOS Circuits," in *IEEE DDECS*, 2006.
[3] B. Masnick and J. Wolf, "On linear unequal error protection codes," *IEEE Trans. Inform. Theory*, vol. 13, no. 4, pp. 600–607, Oct. 1967.
[4] M. May, M. Alles, and N. Wehn, "A case study in reliability-aware design: a resilient LDPC code decoder," in *DATE*, 2008.
[5] C.-H. Huang, Y. Li, and L. Dolecek, "Adaptive error correction coding scheme for computations in the noisy min-sum decoder," in *IEEE ISIT*, 2015.
[6] S. M. S. Tabatabaei, C.-H. Huang, and L. Dolecek, "Optimal design of a Gallager B noisy decoder for irregular LDPC codes," *IEEE Commun. Lett.*, vol. 16, no. 12, pp. 2052–2055, Dec. 2012.
[7] C.-H. Huang and L. Dolecek, "Analysis of finite-alphabet iterative decoders under processing errors," in *IEEE ICCASP*, 2013.
[8] A. Balatsoukas-Stimming and A. Burg, "Faulty successive cancellation decoding of polar codes for the binary erasure channel," in *IEEE ISITA*, 2014.
[9] P. Vivet, E. Beign, H. Lebreton, and N.-E. Zergainoh, "On Line Power Optimization of Data Flow Multi-core Architecture Based on Vdd-Hopping for Local DVFS." in *PATMOS*, 2010.
[10] A. Rahimi, L. Benini, and R. Gupta, "Temporal memoization for energy-efficient timing error recovery in GPGPUs," in *DATE*, 2014.
[11] J. Chen and M. P. Fossorier, "Density evolution for two improved bp-based decoding algorithms of ldpc codes," *IEEE Commun. Lett.*, vol. 6, no. 5, pp. 208–210, 2002.
[12] P. Schläfer, M. Alles, C. Weis, and N. Wehn, "Design Space of Flexible Multi-Gigabit LDPC Decoders," *Hindawi VLSI Design Journal*, vol. 2012, 2012.
[13] C. Gimmler, F. Kienle, C. Weis, N. Wehn, and M. Alles, "ASIC Design of a Gbit/s LDPC Decoder for Iterative MIMO Systems," in *ICNC*, 2012.
[14] M. Alles, F. Berens, and N. Wehn, "A Synthesizable IP Core for WiMedia 1.5 UWB LDPC Code Decoding," in *ICUWB*, 2009.
[15] R. Reis, Y. Cao, and G. Wirth, *Circuit Design for Reliability*. Springer, 2015.
[16] G. Cesana, "A brief history of FD-SOI: a faster, cooler, simpler alternative technology for IoT, mobile and servers," in *Proc. nano-tera*, 2014.
[17] A. Herkersdorf *et al.*, "Resilience Articulation Point (RAP): Cross-layer dependability modeling for nanometer system-on-chip resilience," *Microelectronics Reliability*, vol. 54, no. 6, pp. 1066–1074, 2014.
[18] A. Balatsoukas-Stimming and A. Burg, "Density evolution for min-sum decoding of LDPC codes under unreliable message storage," *IEEE Commun. Lett.*, vol. 18, no. 5, pp. 849–852, May 2014.
[19] N. R. Shanbhag, R. A. Abdallah, R. Kumar, and D. L. Jones, "Stochastic computation," in *IEEE/ACM DAC*, 2010.
[20] S. M. S. Tabatabaei, H. Cho, and L. Dolecek, "Gallager B decoder on noisy hardware," *IEEE Trans. Commun.*, vol. 61, no. 5, pp. 1660–1673, May 2013.
[21] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.