Low Complexity Memory Architectures Based on LDPC Codes: Benefits and Disadvantages

Bane Vasić, Predrag Ivaniš and Srdan Brkic

***Abstract*—In this paper we investigate the problem of information storage in inherently unreliable memory cells. In order to increase the memory reliability, information is stored in memory cells as a codeword of a low-density parity-check (LDPC) code, while the memory content is updated periodically by an error correction scheme. We first present an overview on the state-of-the memory architectures based on LDPC codes, and then asses the benefits of using the coded architectures expressed through the increased reliability. In addition, we provide upper bounds on the complexity of such memories.**

I. Introduction

Due to huge density integration increase, lower supply voltages, and variations in technological process, complementary metal-oxide-semiconductor (CMOS) and emerging nanoelectronic devices are inherently unreliable. Moreover, the demands for energy efficiency require reduction of energy consumption by several orders of magnitude, which can be done only by aggressive supply voltage scaling. Consequently, the signal levels are much lower and closer to the noise level, which reduces the component noise immunity and leads to unreliable behavior. It is widely accepted that future generations of circuits and systems must be designed to deal with unreliable components [1]. Recently, there has been a surge in interest in error control schemes that can ensure fault-tolerance in unreliable hardware.

Von Neumann [2] proposed the first error control scheme which increases the reliability of faulty logic gates. In his approach logic units operating under unreliable hardware are multiplexed, and the majority vote of multiplexed values is propagated to the rest of the circuit. Von Neumann's results are not in the spirit of Shannon's work, since rely on repetition codes, which require high redundancy to achieve low error probability. However, Dobrushin and Ortyukov [3] and Elias [4] showed that for the majority of 2-input Boolean functions there is no error coding technique that can provide higher reliability than the von Neumann multiplexing. On the other hand, results are much more promising if a coding technique is used to protect information stored in inherently unreliable storage devices.

The theoretical fundamentals in this area are given by Taylor in his pioneering work [5]. In Taylor's memory architecture information are stored in unreliable memory cells as a codeword of a low-density parity-check (LDPC) code. The memory cells are updated periodically by an correcting circuit built also from unreliable logic gates. The significance of Taylor's work lies in the fact what he showed that a such memory is able to preserve all stored information in the asymptotic case, i.e., for such memory is said to be stable. His work was improved by Kuznetsov [6] who investigated the same architecture, and usually their memory is called Taylor-Kuznetsov (TK) memory.

An update cycle in the TK memory is functionally equivalent to one iteration of Gallager-B decoder, as was observed by Vasic and Chilappagari [7], who rediscover the original Taylor's work and connected it with state-of-the-art research on codes on graphs. The same authors proposed the memory architecture based on the bit-flipping decoder [8], [9] and used expander arguments to show that a such memory is stable. In addition, Ivkovic *et al.* [10] showed that the TK memory can be improved if the reliable syndrome checker is added into architecture. Recently, Varshney [11] used the density evolution technique to investigate decoders of LDPC codes in the presence of hardware failures. He has also proved the stability of the memory architecture based on the Gallager-A decoder. The density evolution technique was popular over the years and it was used to analyzed faulty Gallager-B [12], min-sum [13], [14] and FAID [15] decoders. However, due to high complexities of above decoders, novel memory architectures were not proposed.

In this paper we present an overview on state-of-the-art research in memory architectures that employ decoders of LDPC codes, as well as approaches used for modeling failures in memory cells and logic gates. We consider three stable memory architectures that are based on Gallager-A, Gallager-B and bit-flipping decoders. We compare these memories in terms of complexity, providing new expression for the complexity of majority logic gates. We then investigate the performance of the architecture based on the bit-flipping decoder in terms of the number of component failures that can be tolerated by the memory. The theoretical derivations are illustrated by the numerical examples. The similar results for other types of memories are not known.

The rest of the paper is organized as follows. In Section II the preliminaries on codes and decoding algorithms on graphs are discussed. In Section III we give a description of state-of-the-art approaches to modeling memory cell and gate failures. Section IV is dedicated to the description of memory architectures, which includes the conditions needed to achieve memory stability. The complexity analysis is presented

B. Vasić is with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, 85721 USA (e-mail: vasic@ece.arizona.edu).

P. Ivaniš and S. Brkic are with the University of Belgrade, Serbia, School of Electrical Engineering, (e-mails: predrag.ivanis@etf.rs, srdjan.brkic@ic.etf.rs).

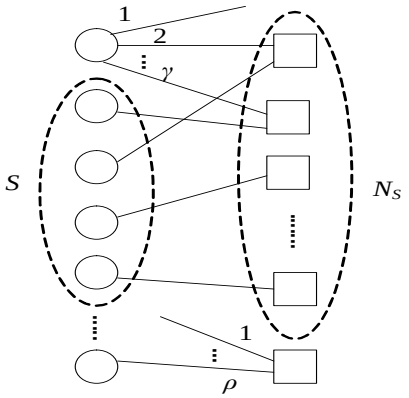

Fig. 1: Illustration of Tanner graph of an LDPC code.

in Section V, together with the reliability analysis of the bit-flipping-based memory. Finally, some concluding remarks and open questions are given in Section VI.

II. Preliminaries

A. LDPC codes and Decoding on Graphs

Consider a (γ, ρ)-regular binary LDPC code, denoted by (N, K), with code rate $R = K/N \geq 1 - \gamma/\rho$ and parity check matrix $\mathbf{H}$. The parity check matrix is the bi-adjacency matrix of a bipartite (Tanner) graph $G = (V \cup C, E)$, where V represents the set of N variable nodes, C is the set of $N\gamma/\rho$ check nodes, and E is the set of $N\gamma$ edges. The length of shortest cycle in G is called *girth* and denoted by g. Bipartite Tanner graph of an LDPC code is illustrated in Fig. 1. Each matrix element $\mathbf{H}_{c,v} = 1$ indicates that there is an edge $e = (v, c)$ between nodes $c \in C$ and $v \in V$, which are referred as *neighbors*. Let $\mathcal{N}_v$ ($\mathcal{N}_c$) be the set of neighbors of the variable node v (check node c). Then, $|\mathcal{N}_v| = \gamma$, $\forall v \in V$ and $|\mathcal{N}_c| = \rho$, $\forall c \in C$, where $|\cdot|$ denotes cardinality. Similarly, a set of neighbours of a set S is denoted as $\mathcal{N}_S$.

We define the iterative decoder by an 5-tuple $D = (\mathcal{B}, \mathcal{Y}, \Phi^{(v)}, \Phi^{(c)}, \Phi^{(a)})$. A set $\mathcal{B}$ defines the alphabet of messages passed over edges of Tanner graph. In this paper we focus on hard-decision decoders and thus $\mathcal{B} = \{0, 1\}$. Similarly, a set of possible values received from the channel is also binary, i.e., $\mathcal{Y} = \{0, 1\}$. Let a sequence of bits received from the channel be $\mathbf{y} = (y_1, y_2, \ldots, y_N)$, $y_i \in \mathcal{B}, 1 \leq i \leq N$. In addition, let $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ denote a codeword of an LDPC code that appear at inputs of the channel whose output is $\mathbf{y}$.

The decoder works by sending binary messages over the edges of the graph. The messages are calculated based on the *nodes update functions*, $\Phi^{(v)}$ and $\Phi^{(c)}$, where $\Phi^{(v)} : \{0, 1\}^{\gamma+1} \to \{0, 1\}$ denotes an update function performed in the variable node v, while $\Phi^{(c)} : \{0, 1\}^{\rho} \to \{0, 1\}$ denotes an update function that corresponds to the check node c. Let $\mu_e^{(l)}$ be a message passed on edges $e = (v, c)$ from variable node v to the check node c, during the iteration ℓ, while $\mu^{(l)}$ denote a vector of all messages received by the node c at time ℓ. Similarly, a message passed from check node c to the variable node v, during the iteration ℓ, we denote as $\nu_e^{(\ell)}$, and a vector of all received messages as $\nu^{(\ell)}$. We next summarize the operations performed in the variable node $v \in V$ and the check node $c \in C$ during the iteration ℓ.

- The outgoing messages from the node v at $\ell = 0$ are initialized by values received from the channel, i.e., $\mu_e^{(0)} = y_v \ \forall e \in \mathcal{N}_v$. At iteration $\ell > 0$ we have
$$\mu_e^{(\ell)} = \Phi^{(v)}(\nu^{(\ell)}, y_v). \tag{1}$$
- The outgoing message from check node c on an edge e is calculated by
$$\nu_e^{(\ell)} = \Phi^{(c)}(\mu^{(\ell-1)}). \tag{2}$$

A mapping $\Phi^{(a)} : \{0, 1\}^{\gamma+1} \to \{0, 1\}$ is used for the final decision-making on transmitted bits.

In this paper we assume that hardware unreliability in the decoder comes from failures of gates used for computation of functions $\Phi^{(c)}$ and $\Phi^{(v)}$. The computation of $\Phi^{(a)}$ is assumed to be reliable, and logic gates used for the implementation are called *golden gates*. If decision-making gates were faulty the error probability of the decoding would be determined by the error probabilities of these gate, not the iterative scheme. Thus, it is reasonable to protect this part of the decoder, for example making it from larger transistors, or by slowing the frequency of operating clock if the timing-related errors are dominant. The same assumption was used in the original Taylor's work [5].

B. Expander Codes and Bit-Flipping Decoding

Expander codes belong to a class of LDPC codes with asymptotically good performance. Expander codes satisfy certain structural properties, which enable them to correct a number of worst case errors by using iterative decoders. We next formally define the expander codes in the same way they were originally introduced by Sipser and Spielman in their classical paper [16].

Definition 1. *A Tanner graph G of a (γ, ρ)-regular LDPC code is a $(\gamma, \rho, \alpha, \delta)$ expander if for every subset S of at most an αn variable nodes, at least $\delta|S|$ check nodes are incident to S.*

Constructing a graph whose subsets of nodes have unusually high number of neighbours, enables many check nodes to be unshared between variable nodes that are corrupt. In that way different iterative decoding algorithms can be chosen, which reduce the number of erroneous bits during each decoding iteration. It is known that such decoders are serial and parallel bit-flipping decoders [16], Gallager-B and min-sum decoders [17] and linear programming decoders [18]. In this paper we mostly investigate parallel bit-flipping algorithm, which can be summarized as follows [16]:

- In parallel, flip each variable that is in more unsatisfied than satisfied parity checks.
- Repeat until no such variable remains.

It is known that random graphs are good expanders, and that a desired expansion can be achieved with high probability [16]. The existence of random graphs with the arbitrary expansion was also considered in [18]. The explicit construction of

expanders using the zig-zag graph product was investigated by Capalbo *et al.* [19].

The above results describe the sufficient conditions that guarantee the existence of expander graphs. Recently, Chilappagari *et al.* [20] directly linked the expansion property with construction parameters like column (row) weight of a code or girth, which are usually inputs of LDPC codes construction algorithms [21], [22]. Their work is presented in the following theorem.

Theorem 1. *Consider a (γ, ρ)-regular LDPC code with Tanner graph with $\gamma \geq 8$ and girth $g = 2g_0$. Then, every set of $|V|$ variable nodes such that $|V| < 3n_0(\gamma/4, g_0)/4$ has the expansion higher then $7\gamma/8$ where*

$$n_0(\gamma/4, g_0) = n_0(\gamma/4, 2j+1) = 1 + \frac{\gamma}{4}\sum_{i=0}^{j-1}\left(\frac{\gamma}{4}\right)^i, \ g_0 \quad odd,$$

$$n_0(\gamma/4, g_0) = n_0(\gamma/4, 2j) = 2\sum_{i=0}^{j-1}\left(\frac{\gamma}{4}\right)^i, \ g_0 \quad even. \tag{3}$$

Proof: See [20], [23]. ■

We used the previous theorem to investigate the number of component failures that can be tolerated by the memory architecture based on the bit-flipping decoder in Section V.

III. Failure Modeling

There are two types of hardware failures considered in this paper: memory cell failures and logic gate failures. We first explain the most commonly used memory cell failure models and then stat-of-the-art modeling approaches of logic gate failures.

A. Failures of memory cells

Failures in the memory cells are model as so called *soft errors*, that appear as a consequence of supply and threshold voltage variations. These failures are transient and manifest as random flips that corrupt the values stored in memory cells without damaging the cells. This means that each value stored in the memory is periodically passed through the Binary Symmetric Channel (BSC) with the fixed crossover probability p_m.

There are also other types of memory failures related to process variations with permanent "stuck-at" defect that cause *hard errors* [24], [25]. In [26], [27] the partitioned linear block codes (PLBC), that efficiently incorporate the stuck-at defect information in the encoding process, were proposed. The encoding algorithm masks the defects by choosing a codeword whose values at the locations of defects match the stuck-at values at those locations. Another way of dealing with permanent cell failures is by using *error-correction pointers* to specify the addresses of failed cells, and to pair each pointer with a replacement memory cell [25]. Similar replacement algorithm was also proposed in [24]. Our memory architecture is not applicable for the stuck-at defect correction since relies on the fact that all errors corrected by the error correction scheme can be correctly written into memory cells. For that reason here we do not investigate these defects.

B. Modeling Logic Gate Failures

On the other hand, gate failures are dependent on gate input patterns and can not be represented in the same manner as memory failures [28]. Here we summarize the gate-state model recently proposed in [29].

Let $f : \{0,1\}^m \rightarrow \{0,1\}$, $m > 1$, be an m-argument Boolean function, which at time instant k produces the result $z^{(k)} = f(y_1^{(k)}, y_2^{(k)}, \ldots, y_m^{(k)})$, where $\mathbf{y}^{(k)} = [y_1^{(k)}, y_2^{(k)}, \ldots, y_m^{(k)}]$ is a vector of input arguments at time k. This function is realized by a faulty logic gate which actual output at time k is

$$\hat{z}^{(k)} = f(y_1^{(k)}, y_2^{(k)}, \ldots, y_m^{(k)}) \oplus \xi^{(k)}, \tag{4}$$

where an error at time k, $\xi^{(k)} \in \{0,1\}$, depends on M successive input arguments $y_1^{(k-M+1)}, \ldots, y_m^{(k-M+1)}, \ldots, y_1^{(k)}, \ldots, y_m^{(k)}$, which form a gate state at time k, denoted by $\mathbf{s}^{(k)} = \{\mathbf{y}^{(j)}\}_{j\in[k-(M-1),k]}$. The value $\xi^{(k)}$ is particular realization of a random variable Ξ, and can be represented probabilistically as

$$\Pr\{\xi^{(k)} = 1\} = \int_{-\infty}^{X} w_\Xi(x; \mathbf{s}^{(k)})dx, \tag{5}$$

where $w_\Xi(x, \mathbf{s}^{(k)})$ is the probability density function (PDF) of Ξ, x denotes the technological parameter whose variations cause failures of the logic gate, X is a parameter threshold used in the implementation, while the effects of different inputs are taken into account by PDF parameters, like mean value, variance, shaping parameters.

The selection of the PDF function depends on the parameters x that cause failures, such as voltage supply, clock delay, or transistor thresholds. In general, if the parameter x is chosen, the PDF can be obtained by measurements or simulation of the selected semiconductor technology. For example, in [30], the mathematical model for delay estimation in the clocked logic circuits is presented. The authors have shown that, under reduced voltage supply, time (delay) needed for the output of a logic gate to become stable is a random variable. The distribution of this delay can be well estimated using inverse Gaussian distribution given by

$$w_\Xi(x; \mu^{(k)}, \lambda^{(k)}) = \sqrt{\frac{\lambda^{(k)}}{2\pi x^3}}\, e^{\frac{-\lambda^{(k)}(x-\mu^{(k)})^2}{2(\mu^{(k)})^2 x}}, x \geq 0, \tag{6}$$

where $\mu^{(k)}$ represents the mean value, and $\lambda^{(k)}$ is the shape parameter associated to the time instant k. It can be observed that timing errors depend on gate inputs from two successive clock intervals, translated to our model $M = 2$. The values $\mu^{(k)}$ and $\lambda^{(k)}$ depend on the gate state $\mathbf{s}^{(k)}$, and can be evaluated empirically. In addition, these parameters typically differ form a gate type to a gate type. In [30] circuits composed of inverters and 2-input AND gates were considered, i.e., circuits described by AIG (AND-Inverter Graph) representations. In Table 1 we give the values of parameters for several gate inputs transitions. These values are for illustration purpose only and we do not give the technological parameters that lead to their evaluation. Based on the PDFs the gate output error probability can be easily obtained, for different gate states. In Fig. 2 we numerically express the gate output error

TABLE I: The parameters of the PDFs for inverter and AND gates [30].

Logic gate	$\mathbf{s}^{(k)}$	$\mu^{(k)}$	$\lambda^{(k)}$
Inverter	$1 \to 0$	0.65×10^{-10}	0.36×10^{-10}
AND	$10 \to 11$	2.3×10^{-10}	3.4×10^{-10}
AND	$11 \to 01$	1.9×10^{-10}	3.4×10^{-10}
AND	$00 \to 11$	2.6×10^{-10}	3.8×10^{-10}
AND	$11 \to 00$	1.5×10^{-10}	3.1×10^{-10}

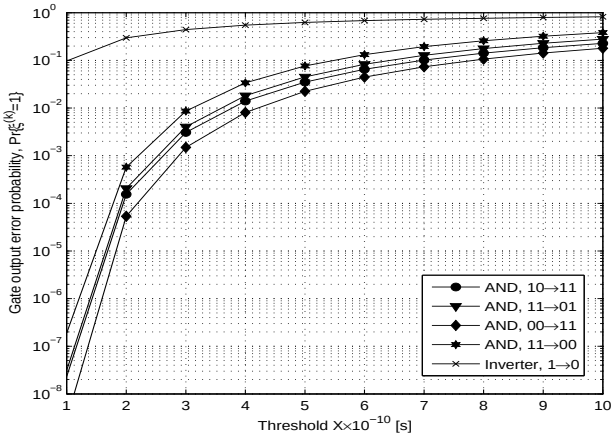

Fig. 2: Probability of error at output of different gates.

probability as a function of a threshold X, which represents the time period assigned for the gate output decision-making. The threshold value is fixed for the specific gate hardware realization. If the time required that the output signal become stabile is higher then X an error occurs. It is obvious that if we prolong the decision-making time, the reliability of the logic gate increases. From Fig. 2 also follows that the reliability of different gates can be significantly different. For example, when $X = 100ns$, the error rate for the Inverter gate is approximately 10^{-1}, while for an AND gate, made in the same technology, the error probability do not exceed 2×10^{-7}. In addition, note that the error probabilities for the same gate can differ by an order of magnitude. In our example the transition $11 \to 00$ produces the erroneous AND gate output more often then other three presented cases. However, analysis that incorporates different gate states is usually to time consuming, since the number of states grows exponentially with the length of input arguments vector, i.e. $O(2^{2m})$. In addition, mean and the shaping parameter of the distribution need to be estimated by Monte Carlo simulation for every component gate in the circuit. Cases where the distribution parameters increase linearly with the level of the circuit were presented in [30], but this is not the general conclusion. Different circuit topologies as well as logic masking effects make the search for the universal solution difficult.

From the previous discussion follows that the gate failure model needs to be simplified, in order to be used for the analysis of faulty LDPC decoders. Note that timing-related errors occur as consequence of inability of the gate output to switch. This lead to the gate-output switching (GOS) error model, proposed in [31]. According to the GOS model

$$\Pr\{\xi^{(k)} = 1 | z^{(k)} \neq z^{(k-1)}\} = \varepsilon, \quad \varepsilon > 0, \tag{7}$$

where ε can be chosen as the highest error rate obtained by simulations for the desired technology, or by some other criterion. When the gate output remains stable during two consecutive intervals there is no propagation delay and the function is always correctly computed, i.e.,

$$\Pr\{\xi^{(k)} = 1 | z^{(k)} = z^{(k-1)}\} = 0. \tag{8}$$

Observe that timing faults can occur even when the ideal output remains stable, but multiple inputs change during a single bit interval. Such a situation is possible when the difference between delay times of consecutive input signals is larger than the propagation delay from the first signal to the output. This case is related to the so-called functional hazard which in the well known phenomenon in logic circuits. The GOS model neglect these phenomenon and assumes that functional hazard is resolved at implementation level. The GOS model was recently used in the analysis of different hard-decision decoders [23], [29], [32].

The previous modeling approach incorporates the data-dependent and correlated nature of logic gate failures. It is restricted to timing-related errors caused by reduced supply voltage, which has been recently studied in the context of energy-efficient VLSI designs. Traditionally logic gates failures are modeled as spatially and timely independent events, the model originally proposed by von Neumann [2]. In the von Neumann error model each component in a circuit fails with the probability ε. This failure probability is independent of the gate inputs, as well as of errors in other components. Although the von Neumann model does not adequately describe physical processes that lead to gate failures, it simplicity makes it popular in the literature. The original Taylor's paper [5] is based on the von Neumann error model, as well the most of the recent relevant literature [7], [9], [11], [12], [14], [15], [33], [34]. In addition, this error model is pessimistic since allows that every use of a logic gate can be followed by the gate failure. At the same time the von Neumann model is robust and can be used if a more precise model is unknown.

There are also some important sources of failures that are much more complicated to describe mathematically. For example, new nano-scale CMOS circuits operating under alpha particle bombardment can not be adequately described by the previous modeling approaches. Alpha particles can affect the neighbouring transistors creating spatially correlated errors. This important failure sources have not been investigated in the context of memory architectures based on LDPC codes.

IV. System Model

A. The Memory Architectures

The general scheme of coded memory architecture is presented in Fig. 3. A collection of $N_c \gg 1$ codewords $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N_c)}$, denoted as $\{\mathbf{x}^{(k)}\}_{k \in [1, N_c]}$, need to be stored in unreliable memory cells, which are periodically updated based on the error correction scheme. If the bit-flipping correction scheme is used each bit is stored in one

memory cell, while if the message-passing scheme is used γ copies of every bit are stored. We denote the number of memory cells per a codeword as $N' \in \{N, \gamma N\}$. This means that in the memory cells values of outgoing massages from variable nodes $\mathbf{X}^{(k)} = [\mu_1^{(k)}, \mu_2^{(k)}, \ldots, \mu_{N'}^{(k)}]$, $k \in [1, N_c]$ are stored. The memory unreliability is modeled by an N'-dimensional binary random variable $\boldsymbol{\Upsilon}$ defined over $\{0,1\}^{N'}$ with independent entries Υ_j such that $\Pr\{\Upsilon_j = 1\} = p_m$, $1 \leq j \leq N'$. The values read from the memory can be described by a random variable vector $\mathbf{R}^{(k)} = \boldsymbol{\Upsilon} \oplus \mathbf{X}^{(k)}$, where its particular realizations are denoted as $\mathbf{y}^{(k)}$, $1 \leq k \leq N_c$. The memory cells are updated by the round-robin scheduling principle, which means that k-th codeword cells $\mathbf{y}^{(k)}$ are updated at $(a-1)N_c + k$, $a \in \mathbb{N}$, memory update cycles. During an update cycle the sequence $\mathbf{r}^{(k)}$ is decoded by one iteration of iterative decoder and replaced by the newly estimated sequence $\hat{\mathbf{r}}^{(k)}$. The cells update mechanism is formally expressed in Algorithm 1.

Algorithm 1 L cycles of memory cells update

Input: $\{\mathbf{r}^{(k)}\}_{k\in[1,N_c]}$
$j \leftarrow 1$
while $j \leq L$ **do**
  $k \leftarrow \mod(j, N_c)$
  for $\forall v$ **do**
    $\forall e \in \mathcal{N}_v : \nu_e^{(k)} = \Phi^{(c)}(r_{e_1}^{(k)}, \ldots, r_{e_\rho}^{(k)})$, $e_i \in \mathcal{N}_c$,
    $n \leftarrow 1$
    while $n < N'/N + 1$ **do**
      $\hat{r}_{e_n}^{(k)} = \Phi^{(v)}(\nu^{(k)})$
      $n \leftarrow n + 1$
    end while
  end for
  $j \leftarrow j + 1$
end while
Output: $\{\hat{\mathbf{r}}^{(k)}\}_{k\in[1,N_c]}$

It was previously stated that the error correction scheme can be chosen based on two basic approaches originally introduced by Gallager [35]: bit-flipping (BF) and message passing (MP) decoding. Recently, memories based on bit-flipping decoding were analyzed in a number of papers [29], [36], [37]. The check node c update function of the BF decoder can be implemented as follows [29]

$$\Phi_{BF}^{(c)}(r_{e_1}^{(k)}, \ldots, r_{e_\rho}^{(k)}) = \bigoplus_{e' \in \mathcal{N}_c \setminus \{e'\}} r_{e'}^{(k)}, \quad \forall e \in \mathcal{N}_c. \tag{9}$$

Note that in each check node ρ update functions need to be implemented. Each update function $\Phi_{BF}^{(c)}(r_{e_1}^{(k)}, \ldots, r_{e_\rho}^{(k)})$ can be implemented as $(\rho - 1)$-input XOR gate. In the variable node v, messages received form neighbouring nodes are compared by the majority voting principle, i.e.,

$$\Phi_{BF}^{(v)}(\nu^{(k)}) = \begin{cases} s, & \text{if } |\{e' \in \mathcal{N}_v : \nu_{e'}^{(k)} = s\}| > \lceil \gamma/2 \rceil, \\ r_v^{(k)}, & \text{otherwise,} \end{cases} \tag{10}$$

In each variable node γ-input majority logic (MAJ) gate needs to be implemented.

The check node operations of MP algorithms are the same as in the BF decoder, i.e.,

$$\Phi_{BF}^{(c)}(r_{e_1}^{(k)}, \ldots, r_{e_\rho}^{(k)}) = \Phi_{MP}^{(c)}(r_{e_1}^{(k)}, \ldots, r_{e_\rho}^{(k)}). \tag{11}$$

Differences exist in the variable node operations which can be defined as follows

$$\Phi_{MP}^{(v)}(\nu^{(k)}) = \begin{cases} s, & \text{if} |\{e' \in \{\mathcal{N}_v \setminus e'\} : \nu_{e'}^{(k)} = s\}| \geq b^{(k)}, \\ r_v^{(k)}, & \text{otherwise,} \end{cases} \tag{12}$$

We need $\gamma - 1$ operations to be implemented in each variable node. When $b^{(k)} = \lceil \gamma/2 \rceil$ our memory is based on Gallager-B (GB) decoder, and the variable node operations can be implemented as $(\gamma-1)$-input MAJ gate. On the other hand, for $b^{(k)} = \gamma - 1$ the decoder reduces to Gallager-A (GB) decoder, and variable node operation is implemented as a $(\gamma - 1)$-input comparator gate.

Following the original Taylor's work, Vasic and Chilappagari [7] proposed the memory based on the Gallager-B decoder, while the memory based on the Gallger-A decoder was investigated by Varshney [11].

B. The Memory Stability

In this subsection we formally introduce the terms needed for the theoretical evaluation of memory architectures, proposed by Taylor [5]. Here we assume that only one coded sequence is stored in the memory cells, i.e. $N_c = 1$. A such memory is denoted by M_K, where subscript K denotes the number of information bits, i.e., *information capacity*. The information capacity when an LDPC code is used satisfied $K \geq N(1 - \gamma/\rho)$. All information is preserved if at any time no *memory failure* occurs.

Definition 2. *The memory failure at time instant k is declared if the memory content cannot be successfully decoded by a noiseless correcting circuit.*

Taylor proposed that for the final decoding, the same correcting circuit is used, as for the memory cell updates, while Varshney used the Maximum-likelihood decoder.

Definition 3. *The complexity of the memory M_K is defined as the total number of memory cells and 2-input logic gates used in the memory.*

Definition 4. *The redundancy of the memory architecture $\mathcal{R}$ is the ratio of the complexity of the memory to the complexity of an irredundant memory built from perfectly reliable cells, which has the same information capacity.*

Definition 5. *The memory M_K is stable if the following is satisfied:*

i) *The complexity of M_K must be bounded by θK, where θ is a fixed parameter.*
ii) *For every time instant $k > 0$, and $\delta > 0$, the probability of memory failure at time k satisfies $P_k < \delta$.*

From the first condition follows that the redundancy of the stable memory must be a constant. When decoders described in the previous section are used, the first condition always

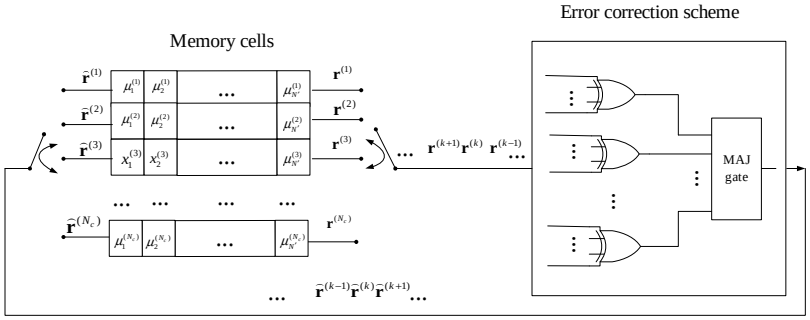

Fig. 3: The block diagram of the memory architecture.

holds, since the node operation complexity do not depend on the information storage capacity. The second condition assures that the number of erroneous cells do not uncontrollable increases over time. It was shown that this condition satisfy Shannon's capacity approaching LDPC codes. In the spirit of the Shannon's work Taylor also defined the *storage capacity* as follows.

Definition 6. *The storage capacity, $\mathcal{C}$, of memory is a number such that there exist the stable memory for all memory redundancy values θ greater than $1/\mathcal{C}$.*

It is said that all memories that are stable have non-zero storage capacity.

V. The Reliability of Memories

A. The Complexity Analysis

In this subsection we compare redundances of memories that are based on BF, GA and GA decoders, under the same information capacity. The check node operations are common for all three architectures. It is known that a $(\rho-1)$-input XOR gate can be implemented as serial concatenation of $\rho-2$ 2-input XOR gates. As there are $N\gamma/\rho$ check nodes, the total number of 2-input XOR gates needed for the decoder implementation is equal to $N\gamma(\rho-2)$.

The complexity of variable node operations is equal to ND_γ, where D_γ denotes the complexity of the γ-input MAJ gate. The following lemma bounds D_γ.

Lemma 1. *The complexity of γ-input MAJ gates, $\gamma \geq 4$, satisfies*

$$D_\gamma \leq \binom{\gamma}{\lceil \gamma/2 \rceil} - 1 + \sum_{i=0}^{\lceil \gamma/2 \rceil - 2} \binom{\gamma - i}{\lceil \gamma/2 \rceil - i}. \quad (13)$$

Proof: See Appendix A. ■

Since the number of memory cells is equal to N, the redundancy of the BF-based architecture satisfies

$$\begin{aligned} \mathcal{R}_{BF} &\leq N(1 + D_\gamma + \gamma(\rho-2))/(RN) \\ &\leq (1 + D_\gamma + \gamma(\rho-2))/((1-\gamma/\rho)). \end{aligned} \quad (14)$$

On the other hand, γ $(\gamma-1)$-input MAJ gates need to be implemented in each variable node if the GB-based architecture is used. In this case we have

$$\mathcal{R}_{GB} \leq \gamma(1 + D_{\gamma-1} + \gamma(\rho-2))/((1-\gamma/\rho)). \quad (15)$$

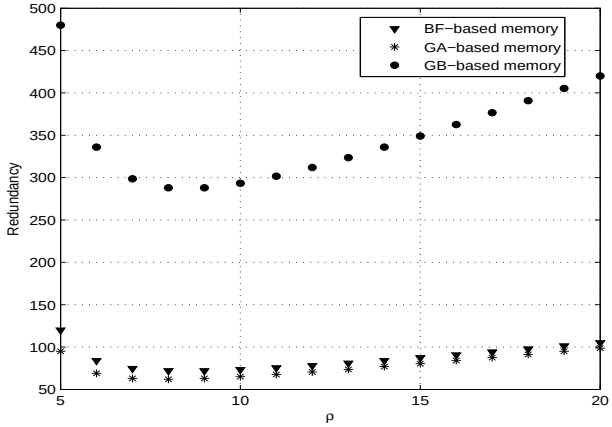

Fig. 4: Complexities of different memory architectures ($\gamma = 4$).

In the GA-based architecture γ $(\gamma-1)$-input comparator gates are implemented in each variable node. It is known that the $(\gamma-1)$-input comparator gate can be implemented as $(\gamma-2)$ 2-input comparator gates, which gives the following redundancy [11]

$$\mathcal{R}_{GA} \leq (\gamma\rho - 1)/((1-\gamma/\rho)). \quad (16)$$

We illustrate the redundancies of different architectures in Fig. 4, for $\gamma = 4$. It can be observed that the GA-based architecture is slightly less complex then the BF-based architecture, while the GB-based architecture requires much higher redundancy. However, the GB-based memory enables the strongest protection against component failures. It can be noted that for all architectures exist optimal code parameters that guarantee minimal redundancy. The lowest redundancy is achieved when the GA-based memory architecture with (3,6)-regular LDPC code is used and it is upper bound to 34. From Taylor's definition follows that the storage capacity satisfies $\mathcal{C} \geq 1/34$ [11].

B. Guaranteed Error Correction of the BF-Based Memory

All three architectures perform well in the asymptotic case and are considered to be stable. In this subsection we investigate the performance of memories that employ finite length codes, in terms of the number of failures that can be

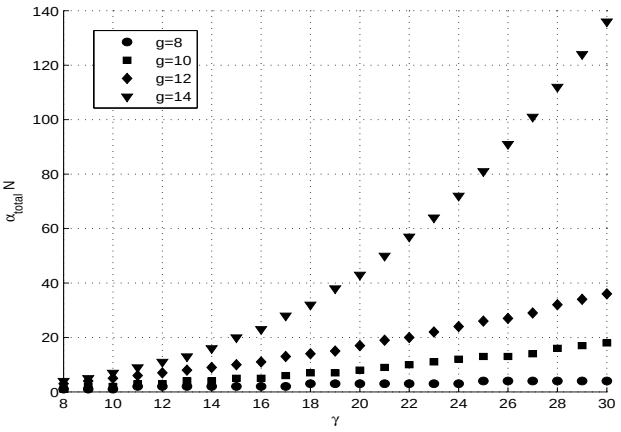

Fig. 5: Number of component failures that can be tolerated by BF-based architecture.

tolerated by the correcting circuit. We consider the BF-based memories since similar results are not known for the other two types of memory architectures.

Let α_m be a fraction of memory cell failures between two update cycles. Similarly, let $\alpha_\oplus$ and α_γ denote fractions of 2-input XOR gates and MAJ logic gates, respectively, that can fall during and update cycle. Then, the following theorem gives a fraction of component failures that can be tolerated if a good expander code is used.

Theorem 2. *Consider a (γ, ρ, α, $7/8\gamma$) expander. The BF-based memory architecture can tolerate a constant fraction of errors in all the components if*

$$\alpha_m + \gamma(\rho - 2)\alpha_\oplus + \alpha_\gamma < 3\alpha/8. \tag{17}$$

Proof: See [9]. ■

Let $\alpha_{total} = \alpha_m + \gamma(\rho - 2)\alpha_\oplus + \alpha_\gamma$. Then, $\alpha_{total}N$ represents the total number component failures that can be tolerated by the BF-based memory. The following theorem bounds the total number of component failures when the finite-length code is used.

Theorem 3. *Consider a ($\gamma \geq 8$, $\rho \geq \gamma$)-regular LDPC code whose Tanner graph has girth $g = 2g_0$. Then, the BF-based memory architecture can tolerate $\alpha_{total}N$ component errors in all the components if*

$$\alpha_{total}N < 9n_0(\gamma/4, g_0)/32. \tag{18}$$

Proof: Follows directly from Theorem 1 and Theorem 2. ■

We next numerically express $\alpha_{total}N$ for different values of γ and g in Fig. 5. It can be observed that if a code with low girth value (for example $g = 8$) is chosen the number of memory failures that can be tolerated is relatively small even for large γ. On the other hand, increasing g leads to polynomial increase of the number of tolerable errors. For example, if $\gamma = 15$ for $g = 12$ the memory architecture can tolerate 12 errors, while for $g = 14$ the number of tolerable errors is equal to 20.

VI. Discussion

This paper represents a survey on fault-tolerant memories that are based on LDPC codes. We presented low complexity memory architectures that show good asymptotic behaviour, and provided finite-length analysis for the case of the BF-based memory. The numerical results reveal the number of component failures that can be tolerated in terms of structural parameters such as column wight γ and code girth g.

Low complexity, high code rate and guaranteed error correction are main reason for dominant use of short-length Hamming codes in modern storage devices. However, the complexity of LDPC-based memories grows only linearly with the code length, which makes this memories competitive when large number of bits need to be stored. In addition, there is a high level of robustness in selection of code rates of LDPC codes. Although, for low γ and g the number of component failures that can be tolerated is rather small, the BF-based memory architecture outperforms architectures based on single-error correction Hammnig codes in terms of guaranteed error correction capability. Implementation of LDPC-based architectures with higher g or γ is even more beneficial, if the memory complexity is not the issue. Note also that decoding of Hammning codes is not resistent to failures in logic gates, which can significantly degrade the memory performance.

There is a number of open questions related to the coded memories. There were little attempts to give closer bounds on the storage capacity, which remains open research topic. The more practical results which would incorporate specifics of data-dependent failure model, presented in Section III-B, are also welcome. Although, data-dependent failures were investigated recently by Brkic *et al.* [23], [29], [32] the explicit memory reliability analysis were not provided. Recently, it was shown by Vasic *et al.* [38] that logic gate failures in certain scenarios can improve the performance of the Gallager-B decoder. Harvesting these surprising effects in memory architectures seams challenging research direction.

Appendix A (Proof of Lemma 1)

The Boolean function that performs majority voting over γ inputs can be decomposed into two parts: the first part that examines every combinations of $\lceil \gamma/2 \rceil$ inputs and the second part that collects observations from the first part. For the first part we require $\binom{\gamma}{\lceil \gamma/2 \rceil}$ $\lceil \gamma/2 \rceil$-input AND gates, while $\binom{\gamma}{\lceil \gamma/2 \rceil}$-input OR gate is used in the second part. We know that n-input AND gate can be further decomposed into $n-1$ 2-input AND gates. However, if decompose AND gate in a such way, some 2-input AND gates would appear more then ones in the implementation. To avoid this situation and to calculate what is the the minimal number of gates that is actually needed, we decompose all AND gates in parallel. Let $f_{\lceil \gamma/2 \rceil}(x_1, x_2, \ldots, x_{\lceil \gamma/2 \rceil})$ denote $\lceil \gamma/2 \rceil$-input AND, where input arguments $x_1, x_2, \ldots, x_{\lceil \gamma/2 \rceil}$ are chosen from a set of γ inputs. This function can be divided by

$$\begin{aligned} &f_{\lceil \gamma/2 \rceil}(x_1, x_2, \ldots, x_{\lceil \gamma/2 \rceil}) \\ &= f_{\lceil \gamma/2 \rceil - 1}(x_1, x_2, \ldots, x_{\lceil \gamma/2 \rceil - 1}) x_{\lceil \gamma/2 \rceil}, \end{aligned} \tag{19}$$

which produces $\binom{\gamma}{\lceil \gamma/2 \rceil}$ different 2-input AND gates. The previous decomposition can be continued, and we have $f_{\lceil \gamma/2 \rceil -1}(x_1, x_2, \ldots, x_{\lceil \gamma/2 \rceil -2}) x_{\lceil \gamma/2 \rceil -2}$, where a set of possible inputs is reduced to $\gamma - 1$. This gives additional $\binom{\gamma-1}{\lceil \gamma/2 \rceil -1}$ different 2-input AND gates. Further iterative decomposition of multi-input AND gates leads to Eq. (13).

ACKNOWLEDGEMENT

This work was supported by the Seventh Framework Program of the European Union, under Grant Agreement number 309129 (i-RISC project), and in part by the NSF under Grants CCF-0963726 and CCF-1314147. Bane Vasic acknowledges generous support of The United States Department of State Bureau of Educational and Cultural Affairs through the Fulbright Scholar Program.

REFERENCES

[1] S. Ghosh and K. Roy, "Parameter variation tolerance and error resiliency: New design paradigm for the nanoscale era," *Proceedings of the IEEE*, vol. 98, no. 10, pp. 1718–1751, Oct. 2010.

[2] J. Von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," in *Automata Studies*, C.E. Shannon and J. McCarty, eds., Princeton Univ. Press, July 1956, pp. 43–98.

[3] R. Dobrushin and S. Ortyukov, "Upper bound on the redundancy of self-correcting arrangements of unreliable functional elements," *Problemy Peredachi Informatsii*, vol. 13, no. 3, pp. 82–89, 1958.

[4] P. Elias, "Computation in the presence of noise," *IBM Journal of Research and Development*, vol. 2, no. 4, pp. 346–353, Oct. 1958.

[5] M. Taylor, "Reliable information storage in memories designed from unreliable components," *Bell System Technical Journal*, vol. 47, pp. 2299–2337, 1968.

[6] A. Kuznetsov, "Information storage in a memory assembled from unreliable components," *Problems of Information Transmission*, vol. 9, pp. 254–264, 1973.

[7] B. Vasic and S. K. Chilappagari, "An information theoretical framework for analysis and design of nanoscale fault-tolerant memories based on low-density parity-check codes," *IEEE Transactions on Circuits and Systems I, Regular Papers*, vol. 54, no. 11, pp. 2438–2446, Nov. 2007.

[8] S. K. Chilappagari and B. Vasic, "Reliable memories built from unreliable components based on expander graphs," *arXiv:0705.0044v1 [cs.IT]*, May 2007.

[9] S. Chilappagari and B. Vasic, "Fault tolerant memories based on expander graphs," in *Proceedings of IEEE Information Theory Workshop*, Tahoe City, CA, USA, 2–7 Sep. 2007, pp. 126–131.

[10] M. Ivkovic, S. K. Chilappagari, and B. Vasic, "Construction of memory circuits using unreliable components based on low-density parity-check codes," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM 06)*, San Francisco, CA, USA, Nov. 2006, pp. 1–5.

[11] L. Varshney, "Performance of LDPC codes under faulty iterative decoding," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4427–4444, July 2011.

[12] S. M. S. Tabatabaei Yazdi, H. Cho, and L. Dolecek, "Gallager B decoder on noisy hardware," *IEEE Transactions on Communications*, vol. 61, no. 5, pp. 1660–1673, May 2013.

[13] C. Kameni Ngassa, V. Savin, and D. Declercq, "Min-Sum-based decoders running on noisy hardware," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM 13)*, Atlanta, USA, Dec. 2013, pp. 1–5.

[14] A. Balatsoukas-Stimming and A. Burg, "Density evolution for min-sum decoding of LDPC codes under unreliable message storage," *IEEE Communications Letters*, vol. 18, no. 5, pp. 849–852, May 2014.

[15] E. Dupraz, D. Declercq, B. Vasic, and V. Savin, "Finite alphabet iterative decoders robust to faulty hardware: Analysis and selection," in *8th International Symposioum on Turbo Codes and Iterative Information Processing (ISTC)*, Bremen, Germany, Aug. 2014, pp. 1–10.

[16] M. Sipser and D. Spielman, "Expander codes," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1710–1722, Nov. 1996.

[17] D. Burshtein and G. Miller, "Expander graph arguments for messagepassing algorithms," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 782–790, Feb. 2001.

[18] J. Feldman, T. Malkin, R. A. Servedio, C. Stein, and M. J. Wainwright, "LP decoding corrects a constant fraction of errors," *IEEE Transactions on Information Theory*, vol. 53, no. 1, pp. 82–89, Jan. 2007.

[19] M. Capalbo, O. Reingold, S. Vadhan, and A. Wigderson, "Randomness conductors and constant-degree lossless expanders," in *STOC 02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, New York, NY, USA: ACM Press, 2002, pp. 659–668.

[20] S. K. Chilappagari, D. V. Nguyen, B. Vasic, and M. W. Marcellin, "On trapping sets and guaranteed error correction capability of LDPC codes and GLDPC codes," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1600–1611, Apr. 2010.

[21] D. V. Nguyen, S. K. Chilappagari, M. W. Marcellin, and B. Vasic, "On the construction of structured LDPC codes free of small trapping sets," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2280–2302, Apr. 2012.

[22] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2012.

[23] S. Brkic, P. Ivanis, and B. Vasic, "Majority logic decoding under data-dependent logic gate failures," submitted for publication, http://arxiv.org/abs/1507.07155.

[24] S. Mukhopadhyay, H. Mahmoodi, and K. Roy, "Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS," *IEEE Transactions on Computer-aided Design of Integreted Circuits and Systems*, vol. 24, no. 12, pp. 1859–1880, Dec. 2005.

[25] S. Schechter, G. H. Lohy, K. Strauss, and D. Burger, "Use ECP, not ECC, for hard failures in resistive memories," in *Proc. International Symposium on Computer Architecture - ISCA*, June 2010.

[26] C. Heegard, "Partitioned linear block codes for computer memory with 'stuck-at' defects," *IEEE Transactions on Inforamtion Theory*, vol. 29, no. 6, pp. 831–842, Nov. 1983.

[27] Y. Kim and V. K. V. Kumar, "Coding for memory with stuck-at defects," *[Online Available] http://arxiv.org/ftp/arxiv/papers/1304/1304.4821.pdf*.

[28] S. Zaynoun, M. S. Khairy, A. M. Eltawil, F. J. Kurdahi, and A. Khajeh, "Fast error aware model for arithmetic and logic circuits," in *Proceedings of 30th IEEE International Conference on Computer Design (ICCD)*, Montreal, QC, Sept.–Oct. 2012, pp. 322–328.

[29] S. Brkic, P. Ivanis, and B. Vasic, "Analysis of one-step majority logic decoding under correlated data-dependent gate failures," in *Proceedings of IEEE International Symposium on Information Theory (ISIT 2014)*, Honolulu, USA, June–July 2014, pp. 2599–2603.

[30] J. Chen, C. Spagnol, S. Grandhi, E. Popovici, S. Cotofana, and A. Amaricai, "Linear compositional delay model for the timing analysis of sub-powered combinational circuits," in *Proc. of IEEE Comp. Soc. Annual Symp. on VLSI*, July 2014.

[31] A. Amaricai, S. Nimara, O. Boncalo, J. Chen, and E. Popovici, "Probabilistic gate level fault modeling for near and sub-threshold CMOS circuits," in *Proc. 17th Euromicro Conf. on Digital Syst. Design (DSD)*, Verona, Avg. 2014, pp. 473–479.

[32] S. Brkic, O. Al Rasheed, P. Ivanis, and B. Vasic, "On fault tolerance of the Gallager B decoder under data-dependent gate failures," *IEEE Communications Letters*, vol. 19, no. 8, pp. 1299–1302, Aug. 2015.

[33] O. Al Rasheed, P. Ivanis, and B. Vasic, "Fault-tolerant probabilistic gradient-descent bit flipping decoder," *IEEE Communications Letters*, vol. 18, no. 9, pp. 1487–1490, Sept. 2014.

[34] F. Leduc-Primeau and W. Gross, "Faulty Gallager-B decoding with optimal message repetition," in *Proceedings of 50th Allerton Conference on Communication, Control, and Computing*, Monticello, USA, Oct. 2012, pp. 549–556.

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

[36] E. Dupraz, Declercq, and B. Vasic, "Analysis of Taylor-Kuznetsov memory using one-step majority logic decoder," in *Proceedings of 10th Information Theory and Applications Workshop (ITA 2015)*, San Diego, CA, Feb. 2015, paper 273, [Online Available:] http://ita.ucsd.edu/workshop/15/files/paper/paper 3446.pdf.

[37] S. Chilappagari, M. Ivkovic, and B. Vasic, "Analysis of one step majority logic decoders constructed from faulty gates," in *Proceedings of IEEE International Symposium on Information Theory (ISIT 2006)*, Seattle, USA, July 2006, pp. 469–473.

[38] B. Vasic, P. Ivanis, S. Brkic, and R. V., "Fault-resilient decoders and memories made of unreliable components," in *Proceedings of 10th Information Theory and Applications Workshop (ITA 2015)*, San Diego, CA, Feb. 2015, paper 273, [Online Available:] http://ita.ucsd.edu/workshop/15/files/paper/paper 273.pdf.