Error-Correcting Decision Diagrams for Multiple-Valued Functions

Helena Astola, Stanislav Stanković, Jaakko T. Astola
Dept. of Signal Processing, Tampere University of Technology,
FI-33101 Tampere, Finland

Abstract—**Decision diagrams are an efficient way of representing switching functions and they are easily mapped to technology. The layout of a circuit is directly determined by the shape of the decision diagram. By combining the theory of error-correcting codes with decision diagrams, it is possible to form robust circuit layouts, which can detect and correct errors. The method of constructing robust decision diagrams is analogous to the decoding process of linear codes, and can be based on simple matrix and look-up operations. In this paper, we focus on error-correcting decision diagrams for multiple-valued functions, considering them for both the Hamming metric and the Lee metric. The performance of robust decision diagrams is analyzed by determining the error probabilities for such constructions. Depending on the error-correcting properties of the code used in the construction, the error probability of a circuit can be significantly decreased by a robust decision diagram.**

I. Introduction

Binary decision diagrams are an efficient way to represent discrete functions and they have many applications in logic design, e.g. in logic circuit minimization [1] and probabilistic analysis of digital circuits [2]. The idea of representing switching circuits using reduced binary decision diagrams was formalized by Bryant in [3], and the topic has been further explored by numerous authors. Binary decision diagrams are easily mapped to technology, since the layout of a circuit is directly determined by the shape of the decision diagram and the complexity of the decision diagram determines the complexity of the final design. Decision diagrams are used for representing both binary and multiple-valued functions [4], [5].

There are several techniques for providing tolerance against hardware component failures, the most well-known being triple modular redundancy (TMR), for which the groundwork was laid in [6]. In the TMR technique, each module of a non-redundant circuit is simply triplicated, and the output is then determined by majority vote. The TMR technique has been studied and improvements have been discussed in several papers, e.g. in [7], [8]. Techniques of increasing fault-tolerance based on error-correcting codes have also been proposed in a number of papers, e.g. the (N, K) concept in [9].

In modern logic circuits, the transistors are shrinking so much that even atomic-scale imperfections and variations within each transistor become a problem. This means that in addition to testing and fault detection procedures it is important to have systematic ways to increase fault tolerance already in the representations of switching functions. In [10], an approach for generating robust circuits using decision diagrams and codes was introduced. When combining decision diagrams with the theory of error-correcting codes, it is possible to form robust decision diagrams that are able to correct decision errors and which can directly be mapped to circuits. This idea gives a basis for error-correcting circuits designed based on decision diagrams.

Linear codes have many useful properties and they are most typically used in data transmission to detect and correct errors on noisy communication channels [11]. The encoding and decoding processes for linear codes are in general a deep topic and many algorithms have been developed for particular code classes. However, in principle and for short codelengths, coding and decoding can be done using simple matrix and lookup operations, so their implementation is easy. The application of linear codes in decision diagrams is basically another way of representing a code, which gives robust diagrams, capable of correcting decision errors.

In sections II and III we recall the basic definitions for decision diagrams and error correcting codes. In section IV we explain the application for error correcting codes and decision diagrams for generating robust decision diagrams for multiple-valued functions. In section V, we analyze the performance of robust decision diagrams by determining the probabilities of correct outputs and comparing them to corresponding traditional diagrams.

II. Decision Diagrams

In this section, we recall the definition of a binary decision diagram (BDD), a multiple-valued decision diagram, i.e. q-ary decision diagram and a multi-terminal decision diagram (MTDD), which are used later in this paper. For basic concepts and properties related to decision diagrams, we refer to [4], [5].

Binary decision diagrams are used to represent switching functions, i.e. functions of the form $\{0, 1\}^n \rightarrow \{0, 1\}$. We define binary decision diagrams using binary decision trees, which are graphic representations of functions in the complete disjunctive normal form.

Definition 1: A binary decision tree (BDT) is a rooted directed graph having $n + 1$ levels with two different types of vertices. On levels 1 to n are the non-terminal nodes, each having two outgoing edges labeled by 0 or 1. On level $n + 1$ are the terminal nodes having the label 0 or 1 and no outgoing edges.

A binary decision tree has a direct correspondence to the truth-table of a function. Let $f(x_1, x_2, \ldots, x_n)$ be a switching function. In the binary decision tree of f, each node on level i corresponds to a specific variable x_i, and by following the edges the value of the function at $(x_1, x_2, \ldots, x_n)$ is found in the terminal node.

Definition 2: A binary decision diagram is a rooted directed graph obtained from a binary decision tree by the following reduction rules:

1) If two sub-graphs represent the same function, delete one, and connect the edge pointing to its root to the remaining subgraph.
2) If both edges of a node point to the same sub-graph, delete that node, and directly connect its edge to the sub-graph.

The definition of a binary decision diagram is easily extended to the q-ary case. Again, we define the decision diagram using the definition of a decision tree.

Definition 3: A q-ary decision tree is a rooted directed graph having $n+1$ levels with two different types of vertices. On levels 1 to n are the non-terminal nodes, each having q outgoing edges with a label from the set $\{0, 1, \ldots, q-1\}$. On level $n+1$ are the terminal nodes, which have a label from the set $\{0, 1, \ldots, q-1\}$ and no outgoing edges.

Definition 4: A q-ary decision diagram is a rooted directed graph obtained from a q-ary decision diagram by the reduction rules in Definition 2.

The number of terminal nodes in decision diagrams is not limited to q nodes. Such decision diagrams are called multiterminal decision diagrams and are to represent functions with an image set having more than q elements. The only difference between a DD and an MTDD is the number of terminal nodes. When dealing with multi-output functions or systems of functions, we use multiterminal decision diagrams where terminal nodes are labelled by the values that the system can get.

III. Error-Correcting Codes

In this section we recall basic definitions and properties of error-correcting codes that will be used later. We focus on linear codes, i.e. subspaces of $\mathbb{F}_q^n$, and, in particular, Hamming codes, which have good properties suitable for the application to binary decision diagrams introduced in this paper. Recall that $\mathbb{F}_q^n$ is a linear (vector) space over the field $\mathbb{F}_q$, i.e. the set $\{(x_1, x_2, \ldots, x_n) \mid x_i \in \mathbb{F}_q\}$ with vector addition and scalar multiplication satisfying the vector space axioms.

Definition 5: A code C is a subset of $\mathbb{F}_q^n$. C is called a *linear code* if C is a linear subspace of $\mathbb{F}_q^n$.

The elements of C are called *codewords*. A linear code C of dimension $k \leq n$ is spanned by k linearly independent vectors of C. A matrix $\mathbf{G}$ having as rows any such k linearly independent vectors is called a *generator matrix* of the code C. If the code has length n and dimension k it is called an (n, k) code.

The code C of dimension k can equivalently be specified by listing $n-k$ linearly independent vectors of $C^\perp$, where $C^\perp$ is the subset of $\mathbb{F}_q^n$ consisting of all vectors orthogonal to C. Any matrix $\mathbf{H}$ having as rows such $n-k$ linearly independent vectors is called a *parity check matrix* of C.

A generator matrix $\mathbf{G}$ of the code C is in *systematic form* if

$$\mathbf{G} = [\mathbf{I}_k | \mathbf{P}]$$
$$= \left[\begin{array}{ccccc|cccc} 1 & 0 & 0 & \cdots & 0 & p_{1,1} & p_{1,2} & \cdots & p_{1,n-k} \\ 0 & 1 & 0 & \cdots & 0 & p_{2,1} & p_{2,2} & \cdots & p_{2,n} \\ & & & \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & p_{k,1} & p_{k,2} & \cdots & p_{k,n-k} \end{array}\right],$$

where $\mathbf{P}$ is called the *parity* part of the generator matrix. It is clear that any generator matrix of C can be put into this form by column permutations and elementary row operations, since the rows are linearly independent. Also, if the generator matrix of C is $\mathbf{G} = [\mathbf{I}_k | \mathbf{P}]$, it is clear that the parity check matrix $\mathbf{H}$ is of the form $\left[-\mathbf{P}^T | \mathbf{I}_{n-k}\right]$.

The code C codes an information word $\mathbf{i} = [i_1, i_2, \ldots, i_k]$ to a length n codeword $\mathbf{c} = [c_1, c_2, \ldots, c_n]$ by matrix multiplication $\mathbf{c} = \mathbf{i} \cdot \mathbf{G}$. Thus, the code C can be defined as $C = \{\mathbf{iG} \mid \mathbf{i} \in \mathbb{F}_q^k\}$ and equivalently with the parity check matrix $\mathbf{H}$ as $C = \{\mathbf{c} \in \mathbb{F}_q^n \mid \mathbf{cH}^T = 0\}$.

Definition 6: The Hamming distance $d_H(\mathbf{x}, \mathbf{y})$ of vectors $\mathbf{x}$ and $\mathbf{y}$ of length n is the number of coordinates where $\mathbf{x}$ and $\mathbf{y}$ differ, i.e. $d_H(\mathbf{x}, \mathbf{y}) = |\{i \mid x_i \neq y_i\}|$.

Definition 7: A code C is e-error correcting if the minimum Hamming distance between two codewords is $2e+1$.

Definition 8: A code $C \in \mathbb{F}_q^n$ is called a q-ary r-covering code of length n if for every word $\mathbf{y} \in \mathbb{F}_q^n$ there is a codeword $\mathbf{x}$ such that the Hamming distance $d_H(\mathbf{x}, \mathbf{y}) \leq r$. The smallest such r is called the covering radius of the code.

In other words, the covering radius of the code is the smallest r such that the finite metric space $\mathbf{F}_q^n$ is exhausted by spheres of radius r around the codewords.

Definition 9: A code is called perfect if it is e-error correcting and its covering radius is e.

The simplest linear code is the repetition code. In an $(r, 1)$ binary repetition code a digit is encoded as a sequence of r repetitions of the digit itself, and the decoding is done by majority-vote decoding. For example, in the binary $(3, 1)$ repetition code, a 0 is encoded as the sequence 000, and if the decoder receives either 000, 001, 010 or 100, it decodes the sequence as a 0 by majority-vote decoding. Therefore, the $(3, 1)$ repetition code is one error correcting.

Binary Hamming codes are a family of $(2^m-1, 2^m-m-1)$ codes with parity check matrices consisting of all 2^m-1 distinct non-zero m-tuples. Since all of the columns are distinct, no sum of two columns can be the zero vector, and hence the code has minimum distance of ≥ 3. Therefore, Hamming codes are one error correcting. Hamming codes have the covering radius of one, hence they are perfect codes. This means that for each binary vector $\mathbf{v}$ of length 2^m-1 there is a unique codeword within distance 1 from $\mathbf{v}$.

Hamming codes are easily defined for vector spaces $\mathbb{F}_q^n$ having parameters $(\frac{q^m-1}{q-1}, \frac{q^m-1}{q-1} - m)$. For each m, there are

$(q^m - 1)$ different nonzero vectors, but since the minimum distance of the code is ≥ 3, the columns of the parity check matrix have to be pairwise linearly independent. Therefore, there are $\frac{q^m-1}{q-1}$ distinct q-ary vectors that we can list as columns of the parity check matrix $\mathbf{H}$.

The above codes are for the Hamming metric, but we may consider error-correcting codes in other metrics also. Let us call a Lee-error-correcting code any error-correcting code defined for the Lee metric.

Definition 10: The Lee distance $d_L(\mathbf{x}, \mathbf{y})$ of q-ary vectors $\mathbf{x}$ and $\mathbf{y}$ of length n is the sum $\sum_{i=1}^{n} \min(|x_i - y_i|, q - |x_i - y_i|)$.

For example, for $q = 5$, $d_L(0,1) = 1$, $d_L(0,2) = 2$, $d_L(0,3) = 2$ and $d_L(0,4) = 1$. Also, $d_L(\mathbf{a}+\mathbf{k}, \mathbf{b}+\mathbf{k}) = d_L(\mathbf{a}, \mathbf{b})$, i.e., like the Hamming metric, the Lee metric is translation invariant. Notice, that for $q = 2$ and $q = 3$, the Hamming and the Lee metric coincide.

Perfect codes exist for the Lee metrics also. For example, for any given e, there exists a perfect e-Lee-error-correcting code with $n = 2$ over $\mathbb{F}_q$ such that $q = 2e^2 + 2e + 1$.

For general properties of error-correcting codes we refer to [11], [12], [13].

IV. Error-correcting Decision Diagrams

In [10], two methods of constructing robust decision diagrams, i.e. decision diagrams, which are able to correct decision errors were introduced. The methods are based on representing functions using error-correcting codes and constructing decision diagrams for these representations. This way, it is possible to generate a decision diagram, which can detect and correct decision errors. This construction can directly be mapped to technology, which gives a robust circuit realizing the original function. We explain the general construction of robust q-ary decision diagrams and derive the construction of a robust decision diagram for a specific function using this general construction.

We want to generate a robust DD for q-ary functions with k variables without specifying a certain function. To do this, we need an error-correcting code with parameters (n, k). The basic idea is to map an arbitrary function $f(x_1, x_2, \ldots, x_k)$ to a function $g(y_1, y_2, \ldots, y_n)$ of a larger domain using error-correcting codes with parameters (n, k) and having the minimum distance $2e+1$. This way we obtain a redundant representation for all q-ary functions of k variables, from which we construct a reduced decision diagram. This construction will be the robust decision diagram of the k-variable q-ary functions.

The procedure begins by determining an (n, k)-code with minimum distance $2e + 1$ and its generator matrix $\mathbf{G}$. Then, the function $g(\mathbf{y})$ is defined as

$$g(\mathbf{y}) = \begin{cases} f(\mathbf{x}) & \text{if } d_H = (\mathbf{y}, \mathbf{xG}) \leq e \\ * & \text{otherwise.} \end{cases} \qquad (1)$$

In other words, each $\mathbf{xG}$ and the vectors $\mathbf{y} \in \mathbb{F}_q^n$ within distance e from $\mathbf{xG}$ are assigned to the symbolic value $f(\mathbf{x})$. The vectors $\mathbf{y} \in \mathbb{F}_q^n$ at distance $> e$ from all the codewords are assigned to the label $*$. The symbol $*$ can be some arbitrary value, which can be defined in some suitable way. The function now behaves as the decoding algorithm of the code C, i.e. the vectors of $\mathbb{F}_q^n$ within distance e from a codeword $\mathbf{xG}$ are interpreted as the codeword itself when determining the function value. This is just the decoding process, where each received n-ary sequence is interpreted as the codeword within distance e from the received sequence. If the label $*$ is obtained, then more than e decision errors have been made indicating at least $e + 1$ faults in the corresponding circuit.

The next step is to construct the multiterminal decision tree for the function g, which will have in total q^n terminal nodes, and reduce the obtained diagram. After reducing, we have a diagram with $q^k + 1$ terminal nodes labelled by $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{F}_q^k$ and $*$. This construction can correct e-digit errors, since the correct function value is obtained even if a decision error occurs in at most e nodes of the diagram.

From the obtained reduced decision diagram we get a robust decision diagram of a specific function f by replacing the labels $f(\mathbf{x})$ by the actual values of the function f and reducing the diagram with respect to that function. This diagram will then give a robust layout for a circuit realizing the desired function of k variables.

Example 1: We want to construct a robust decision diagram for all ternary functions of 2 variables. For this purpose, we may use the $(4, 2)$ ternary Hamming code, which corrects one error. The generator matrix $\mathbf{G}$ for this code is

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 2 & 2 \end{bmatrix}.$$

The function g is obtained by multiplying $\mathbf{G}$ by the ternary vectors of length 2 and then mapping the obtained codewords and the length 4 vectors within distance 1 from the codewords to the corresponding function values. The obtained ternary decision diagram will have 9 terminal nodes corresponding to $f(0,0), f(0,1), f(0,2), \ldots, f(2,2)$ (Figure 1). The code is perfect, hence there are no $*$-valued outputs in the obtained decision diagram.

A robust decision diagram for a specific ternary function of 2 variables is obtained by assigning the function values to the terminal nodes and reducing the obtained diagram.

Example 2: Consider the $(3, 1)$ repetition code for $\mathbb{F}_q$. This code is perfect in $\mathbb{F}_2$, but will result in $*$-valued outputs for $\mathbb{F}_q$, where $q > 2$. However, we may use this code for generating a robust decision diagram for quaternary logic. The resulting decision diagram is a robust construction for a single quaternary node, which corrects one error (Figure 2). Obtaining the output $*$ indicates at least two decision errors.

We may extend the concept of error-correcting decision diagrams for other metrics than the Hamming metric, e.g., the Lee metric. Using the Lee metric, we define the error-correcting decision diagrams as follows. Determine an (n, k) Lee-error-correcting code with minimum distance $2e + 1$ and its generator matrix $\mathbf{G}$. Then, the function $g(\mathbf{y})$ is defined as

$$g(\mathbf{y}) = \begin{cases} f(\mathbf{x}) & \text{if } d_L = (\mathbf{y}, \mathbf{xG}) \leq e \\ * & \text{otherwise.} \end{cases} \qquad (2)$$

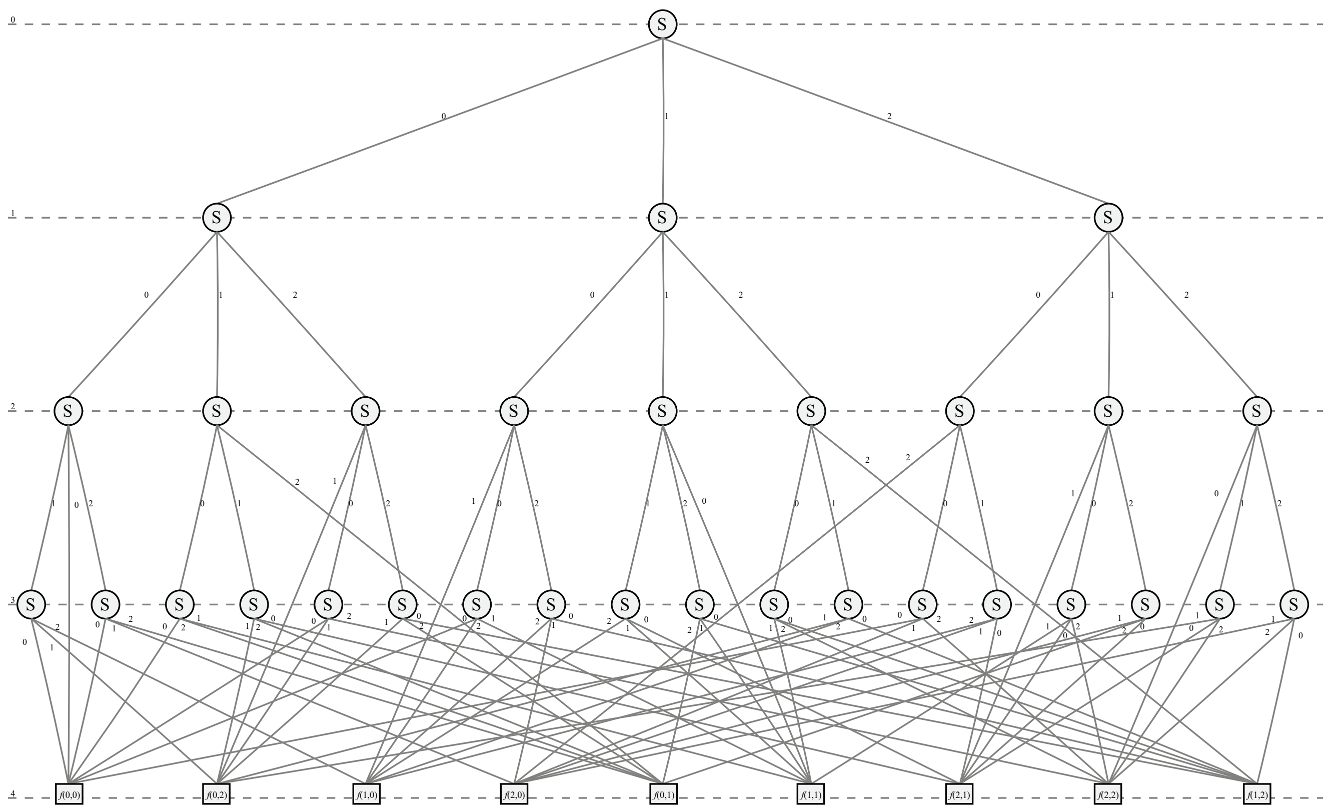

Fig. 1: A robust diagram for ternary functions of 2 variables using (4,2) Hamming code.

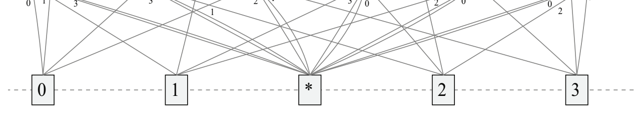

Fig. 2: A robust construction for a single quaternary node using the (3,1) repetition code.

Example 3: We wish to have a robust representation for a 5-ary decision node. We may construct such a representation using a perfect one-error-correcting code in the Lee metric, having the generator matrix

$$\mathbf{G} = \begin{bmatrix} 3 & 1 \end{bmatrix}.$$

The construction is done similarly as in Examples 1-2, but the mapping of length 2 vectors to corresponding values in $\mathbb{F}_5$ is done with respect to the Lee metric. Table I shows the radius one spheres around the codewords in the vector space $\mathbb{F}_5^2$, illustrating how the vectors $\mathbf{y}$ of length 2 satisfying $d_L = (\mathbf{y}, \mathbf{x}\mathbf{G}) \leq 1$, where $\mathbf{x} \in \mathbb{F}_5$, are found.

TABLE I: The radius one spheres around codewords (in boldface) labeled by the corresponding $\mathbf{x} \in \mathbb{F}_5$.

4	0	4	**4**	4	3
3	3	2	4	3	**3**
2	2	**2**	2	1	3
1	0	2	1	**1**	1
0	**0**	0	4	1	0
	0	1	2	3	4

The obtained robust representation for a single 5-ary decision node is in Figure 3.

V. Fault-Tolerance Analysis

In this section, the performance of robust decision diagrams is analyzed by determining the probability of correct outputs for these constructions. For traditional decision diagrams, an incorrect output is obtained whenever at least one incorrect decision is made on a path. For robust decision diagrams, up to e incorrect decisions can be made on a path, and the construction still gives the correct output. For determining the probability of a correct output, we need to list all the possible combinations of incorrect decisions, for which at most e are made on each path.

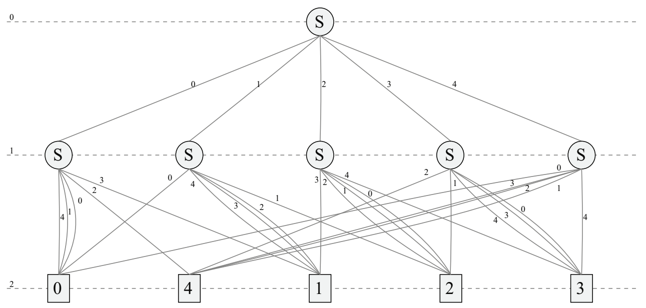
Fig. 3: A robust construction for a single 5-ary decision node using a perfect Lee-error-correcting code.

We start by analyzing the error probabilities for the decision diagrams constructed using the Hamming metric. Let us denote by p the probability of an incorrect decision in a decision node. We assume that the probability of an incorrect decision in a node is independent from the decisions made in other nodes. It could be so, that the correct output is obtained even if incorrect decisions are made in more than e nodes, but we are not interested in these outputs. We call decision nodes, which make an incorrect decision, as faulty nodes. This does not correspond to the actual circuit design and we are only looking at incorrect decisions, which may be caused by any possible reason, e.g. a broken gate or just a temporary malfunction. Notice, that in the Hamming metric, it is irrelevant to the error-correcting properties, which incorrect decision is made in the faulty node, whereas in the case of the Lee metric, only some incorrect decisions are allowed.

For computing the probability of a correct output, we use a brute force method to list all the combinations of nodes, for which there are up to e faulty nodes on each path. It is clear that if there are no faulty nodes in the whole diagram, we may write the output probability as $(1-p)^n$, where n is the total number of nodes in the diagram. This gives the first term of the probability function for a correct output. Also, the following $e-1$ terms of the function are given by the binomial expansion, i.e., $np(1-p)^{n-1}, \dots, \binom{n}{e}p^e(1-p)^{n-e}$. The rest of the terms depend on e and the structure of the diagram. It follows, that the error probability of a robust construction is $1-(\sum_{k=0}^{e}\binom{n}{k}p^k(1-p)^{n-k}+\cdots)$, which gives an expansion having a lowest degree term $A \cdot p^{e+1}$, where A is some constant. For a traditional diagram, since a single incorrect decision causes an incorrect output, the lowest degree term of the error probability function is always $B \cdot p$, where B is some constant.

For example, take the diagram in Figure 1, for this diagram $e = 1$ and there are in total 31 nodes in the diagram. We need to list all the combinations of nodes for which there are either zero or one faulty nodes on each path. The first terms of the probability function for a correct output are therefore $(1-p)^{31}$ and $31p(1-p)^{30}$. The following terms can be determined by first considering all cases with exactly two faulty nodes in the diagram. It is clear that these faulty nodes can be situated on any single level of the diagram, since each node on a specific level never belongs to the same path as the other nodes on that level. It is possible for the faulty nodes to be on two different levels, but in this case we need to make sure that these faulty nodes are never on the same path. We may continue to list all possible cases when there are exactly 3 faulty nodes, 4 faulty nodes, etc. The last case is when all the 18 nodes on level 3 are faulty. Adding up all these situations gives us the probability of a correct output. We may now compare the probability of a correct output in the robust construction to $(1-p)^4$, which would be the probability of a correct output in the corresponding traditional diagram (Figure 4).

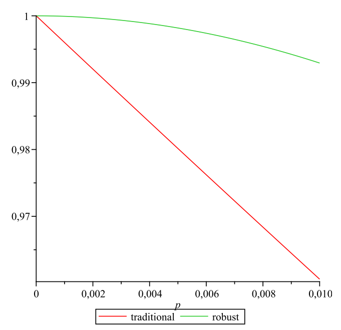

Fig. 4: The probability of a correct output in the traditional diagram (red) and in the robust diagram (green) for ternary 2-variable functions.

We may do this similar analysis for the diagram in Figure 2, which is a robust construction corresponding to a single quaternary decision node. By our assumptions, a single quaternary node gives the correct output with probability $(1-p)$, since the probability of a faulty node is p. For the diagram in 2, we have $e = 1$ and in total 11 decision nodes, therefore the first terms of the probability function are given by $(1-p)^{11}$ and $11p(1-p)^{10}$. The rest of terms are determined similarly above for the ternary diagram. The obtained probability is compared to $(1-p)$ in Figure 5.

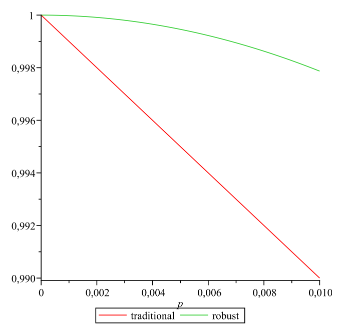

Fig. 5: The probability of a correct output in the traditional diagram (red) and in the robust diagram (green) for a quaternary decision node.

In analyzing the fault-tolerance of robust decision diagrams constructed in the Lee metric, we have to take into account that in a decision node, we can make either the correct decision, an incorrect decision which is at Lee distance $\leq e$ from the correct value, or an incorrect decision which is at distance $> e$ from the correct value. Therefore, we must change our analysis slightly to make sense for the Lee-error-correcting decision diagrams.

Let us denote by p^k the probability of an incorrect decision within distance k from the correct value. This assumption is made, since values within smaller distance from each other are obtained with smaller difference in voltage in the circuit level, and it is therefore more probable that an incorrect value at a smaller distance is obtained. Again we assume that a decision in one node does not depend on decisions made in other nodes. The probability of a correct decision is now $1 - \sum_{k=1}^{m} p^k$, where m is the maximum distance of an incorrect decision from the correct value. To get a correct output, no incorrect decision at distance $> e$ can be made on a path, but incorrect decisions, which added together are at distance $\leq e$ from the correct value, are allowed. For example, if $e = 3$, it is possible to make 3 incorrect decision at distance 1, or an incorrect decision at distance 2 and an incorrect decision at distance 1, or one incorrect decision at distance 3, and still obtain the correct output. We may formulate the probability of a correct output starting with the term $(1 - \sum_{k=1}^{m} p^k)^n$, where n is the total number of nodes. In the following terms, all combinations of incorrect situations at distance $\leq e$ must be taken into account. As for the Hamming metric, it can be shown that in the Lee metric the probability of an incorrect output in a robust diagram will have the lowest degree term $A \cdot p^{e+1}$, where A is some constant.

For example, take the diagram in Figure 3, which has in total 6 nodes and $e = 1$. Since $q = 5$, an incorrect value can be at most at distance 2 from the correct value. We may compute the probability for a correct output, starting with terms $(1 - (p + p^2))^6$ and $6p(1 - (p + p^2))^5$. It is clear that the correct output is obtained even if all the nodes on level 1 give incorrect values within distance 1 from the correct value, if the top level node is correct. Therefore, the rest of the terms are given by $\sum_{k=2}^{5} \binom{5}{k} p^k (1 - (p + p^2))^{6-k}$. The probability of a correct output in the robust construction is compared to the probability of a correct output in a single 5-ary node in Figure 6.

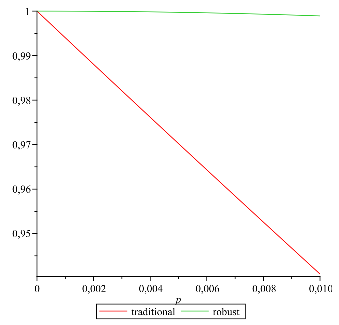

Fig. 6: The probability of a correct output in the traditional diagram (red) and in the robust diagram (green) for a 5-ary decision node.

VI. Conclusions

In this paper we discussed error-correcting decision diagrams for multiple-valued logic and analyzed their performance. Decision diagrams are useful in logic design, since the layout of the diagram determines the layout of the circuit. Combining the theory of error-correcting codes with decision diagrams it is possible to create diagrams, which are capable of detecting and correcting decision errors.

The fault-tolerance analysis of robust decision diagrams shows, that the probability of incorrect outputs can be significantly decreased depending on the error-correcting properties of the code. With traditional diagrams, a single incorrect decision causes the output to be incorrect, and the lowest degree term of the error probability function is always a multiple of p, where p is the error probability of a single node in the diagram. For robust diagrams, the lowest degree term is always $A \cdot p^{e+1}$, where A is some constant. This means, that even with moderately high gate error probabilities, e.g, 10^{-3}, a robust construction will have a significantly decreased probability for an incorrect output. For the example diagrams given in this paper, the probability of a correct output with reasonable gate error probabilities is ≈ 1, which means that the constructions are extremely fault-tolerant. However, better error-correcting properties increase the complexity of the design.

References

[1] S. B. Akers, "Binary decision diagrams," *IEEE Trans. Computers*, vol. C-27, No. 6, pp. 509–516, 1978.

[2] M. A. Thornton and V. S. S. Nair, "Efficient calculation of spectral coefficients and their applications," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-14, No. 11, pp. 1328–1341, 1995.

[3] R. E. Bryant, "Graph-based algorithms for Boolean functions manipulation," *IEEE Trans. Computers*, vol. C-35, No. 8, pp. 667–691, 1986.

[4] J. T. Astola and R. S. Stanković, *Fundamentals of Switching Theory and Logic Design*. Springer, 2006.

[5] D. M. Miller and M. A. Thornton, *Multiple Valued Logic: Concepts and Representations*. Morgan & Claypool, 2008.

[6] J. von Neumann, "Probabilistic logics and synthesis of reliable organisms from unreliable components," in *Automata Studies*, C. Shannon and J. McCarthy, Eds. Princeton University Press, 1956, pp. 43–98.

[7] J. Abraham and D. Siewiorek, "An algorithm for the accurate reliability evaluation of triple modular redundancy networks," *IEEE Transactions on Computers*, vol. 23, pp. 682–692, 1974.

[8] W. van Gils, "A triple modular redundancy technique providing multiple-bit error protection without using extra redundancy," *IEEE Transactions on Computers*, vol. 35, pp. 623–631, 1986.

[9] T. Krol, "(n, k) concept fault tolerance," *IEEE Trans. Comput.*, vol. 35, pp. 339–349, April 1986.

[10] H. Astola, S. Stanković, and J. T. Astola, "Error correcting decision diagrams," in *Proceedings of The Third Workshop on Information Theoretic Methods in Science and Engineering*, Tampere, Finland, Aug 16-18 2010.

[11] J. H. van Lint, *Introduction to Coding Theory*. New York: Springer Verlag, 1982.

[12] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam: North-Holland, 1997.

[13] E. R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw-Hill, 1968.