ERROR-CORRECTING DECISION DIAGRAMS

Helena Astola, Stanislav Stanković, Jaakko T. Astola

Department of Signal Processing, Tampere University of Technology,
P.O.Box 553, FIN-33101 Tampere, FINLAND
helena.astola@tut.fi, stanislav.stankovic@tut.fi, jaakko.astola@tut.fi

ABSTRACT

Decision diagrams are an efficient way of representing switching functions and they are easily mapped to technology. A layout of a circuit is directly determined by the shape and the complexity 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 presented in this paper is analogous to the decoding process of linear codes, and is based on simple matrix and look-up operations.

1. 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].

In this paper, we introduce a new approach for generating robust circuits using decision diagrams and codes. 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 [6]. 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. Our 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 2 and 3 we recall the basic definitions for decision diagrams and error correcting codes. In section 4 we introduce the new application for error correcting codes and decision diagrams for generating robust decision diagrams. We provide a full description of the procedure of generating the robust diagram for both specific functions and all functions of a given number of variables. In addition to this, we provide examples using Hamming codes, which are particularly useful for our purpose for their properties as perfect codes.

2. 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, a multi-terminal decision diagram (MTDD) and a shared decision diagram, 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 $\{0,1\}^n \to \{0,1\}$. 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 for 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 binary decision diagrams is not limited to two nodes. Such decision diagrams are called multiterminal decision diagrams and are to represent functions with an image set having more than two 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 either use multiterminal decision diagrams where terminal nodes are labelled by the values that the system can get, or use shared decision diagrams, which are constructed by first constructing the decision diagrams for different outputs and then sharing the isomorphic sub-graphs.

3. 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

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

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.*

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 $\mathbf{v}$.

Example 1. *The parity check matrix $\mathbf{H}$ of the binary (7,4) Hamming code can be constructed by listing all vectors of length 3 as columns:*

$$\mathbf{H} = \left[\begin{array}{cccc|ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right].$$

Therefore, its generator matrix is

$$\mathbf{G} = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right].$$

Hamming codes are easily defined for larger 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 $\mathbf{H}$.

For general properties of error-correcting codes we refer to [6], [7].

4. ERROR-CORRECTING DECISION DIAGRAMS

In this section, two methods of constructing robust decision diagrams, i.e. decision diagrams, which are able to correct decision errors, are introduced. The methods are based on representing functions using error-correcting codes and constructing decision diagrams for these representations. This way, we are able to generate a decision diagram, which can correct decision errors. This construction can directly be mapped to technology, which gives a robust circuit realizing the original function. The first approach is for a specific already determined function and the second approach is for the general case, where the function can be determined after the construction. In the latter part of the section, both methods are applied using Hamming codes.

4.1. Construction of the robust DD for a specific function

In the first approach, we have a function $f(x_1, x_2, \ldots, x_k)$ for which we want to construct a robust q-ary decision diagram. The basic idea is to map the function f 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 the function for which we construct a reduced binary decision diagram. This construction will be the robust decision diagram of the function f.

Let $f(\mathbf{x}) : \mathbb{F}_q^k \to \mathbb{F}_q$. Using some error-correcting code C with parameters (n,k), minimum distance $2e+1$ and a generator matrix $\mathbf{G}$ we may define the function $g(\mathbf{y})$ as:

$$g(\mathbf{y}) = \begin{cases} f(\mathbf{x}) \in \mathbb{F}_q & \text{if } d_H(\mathbf{y}, \mathbf{xG}) \leq e \\ * & \text{otherwise,} \end{cases} \tag{4.1}$$

where $*$ is some arbitrary label. In other words, every codeword $\mathbf{xG} \in \mathbb{F}_q^n$ and each vector of $\mathbb{F}_q^n$ within distance e from $\mathbf{xG}$ is assigned to the value of $\mathbf{x}$ under the function f. 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 decision diagram of the function g and reduce the obtained diagram. This construction can correct e-bit errors, since the correct function value is obtained even if a decision error occurs in at most e nodes of the diagram.

Example 2. *Let $f(\mathbf{x}) : \mathbb{F}_2^2 \to \mathbb{F}_2$, i.e. f is a function of 2 variables. Define the function as $f(0,0) = 0$, $f(0,1) = 1$, $f(1,0) = 0$ and $f(1,1) = 1$. Now, let C be a $(5,2)$-code defined by the generator matrix*

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

The parity check matrix of C is

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

Since the sum of no two columns of $\mathbf{H}$ is zero, the code has minimum distance 3 and corrects 1-bit errors.

To obtain the function $g(\mathbf{y})$, first multiply the generator matrix $\mathbf{G}$ by each vector $\mathbf{x} \in \mathbb{F}_2^2$ to obtain the codewords. Map each codeword $\mathbf{c} = \mathbf{xG}$ to the value $f(\mathbf{x})$. For example, since for $\mathbf{x} = [0,1]$ we have $[0,1] \cdot \mathbf{G} = [0,1,1,0,1]$, then $g(0,1,1,0,1) = f(0,1) = 1$.

Then, map each vector $\mathbf{y} \in \mathbb{F}_2^5$ within distance 1 from $\mathbf{c}$ to $f(\mathbf{x})$. For example,

$$\begin{aligned} g(1,1,1,0,1) = g(0,0,1,0,1) = g(0,1,0,0,1) \\ = g(0,1,1,1,1) = g(0,1,1,0,0) = g(0,1,1,0,1) \\ = f(0,1) = 1. \end{aligned}$$

The vectors $\mathbf{y} \in \mathbb{F}_2^5$ at distance > 1 from all the codewords are mapped to the label $*$.

Now, the MTBDD for g can be constructed and reduced to obtain the robust MTBDD of f (Figure 1).

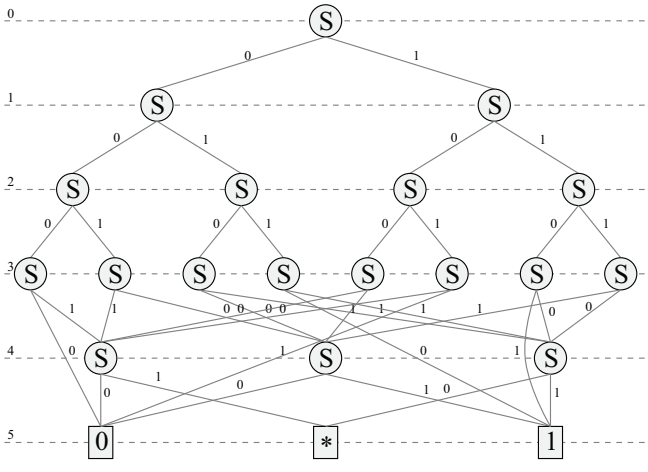

Figure 1: A robust MTBDD for the function f.

With this approach, the procedure of generating the robust decision diagram goes as follows:

1. For the function $f : \mathbb{F}_q^k \to \mathbb{F}_q$, determine the generator matrix $\mathbf{G}$ of the desired code C. Then, multiply $\mathbf{G}$ by each $\mathbf{x}_i \in \mathbb{F}_q^k$ to obtain $\mathbf{c}_i \in \mathbb{F}_q^n$.

2. For each $\mathbf{c}_i$, list all the vectors $\mathbf{y}_{i,1}, \mathbf{y}_{i,2}, \ldots, \mathbf{y}_{i,m}$ of $\mathbb{F}_q^n$ within distance e from each $\mathbf{c}_i$.

3. Map each $\mathbf{c}_i$ to $f(\mathbf{x}_i)$ and all $\mathbf{y}_{i,1}, \mathbf{y}_{i,2}, \ldots, \mathbf{y}_{i,m}$ to the same value $f(\mathbf{x}_i)$.

4. Map each $\mathbf{y} \in \mathbb{F}_q^n$ at distance $> e$ from all the codewords $\mathbf{c}_i$ to $*$.

5. Construct a MTDD for the obtained function g and reduce it to get the robust diagram for the function f.

4.2. General construction of a robust DD

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). As in the first approach, the basic idea is to map an arbitrary function $f(x_1, x_2, \ldots, x_k)$ to some function $g(y_1, y_2, \ldots, y_n)$.

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

$$g(\mathbf{y}) = \begin{cases} f(\mathbf{x}) & \text{if } d_H = (\mathbf{y}, \mathbf{xG}) \leq e \\ * & \text{otherwise.} \end{cases} \tag{4.2}$$

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 $\mathbf{x}$. The vectors $\mathbf{y} \in \mathbb{F}_q^n$ at distance $> e$ from all the codewords are assigned to the label $*$.

We now have the function g for which we can generate a multiterminal decision tree, which will have in total q^n terminal nodes. After reducing the obtained decision diagram, we have a diagram with $q^k + 1$ terminal nodes labelled by $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{F}_q^k$ and $*$.

From the obtained reduced decision diagram we get a robust q-ary 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 3. *Consider the one error correcting code C from Example 2. Using C, we can construct the robust decision diagram for binary 2-variable functions. Take all $\mathbf{x} \in \mathbb{F}_2^2$ and for each, compute the codeword $\mathbf{c} = \mathbf{x} \cdot \mathbf{G}$, where $\mathbf{G}$ is the generator matrix of C. Now, to obtain g, map each of the codewords to the label $f(\mathbf{x})$. For example, $[0,1] \cdot \mathbf{G} = [0,1,1,0,1]$ is mapped to $f(0,1)$.*

Then, for each $\mathbf{c}$, map all the vectors $\mathbf{y} \in \mathbb{F}_2^5$ within distance 1 from $\mathbf{c} = \mathbf{xG}$ to the corresponding $f(\mathbf{x})$. For example,

$$g(1,1,1,0,1) = g(0,0,1,0,1) = g(0,1,0,0,1)$$
$$= g(0,1,1,1,1) = g(0,1,1,0,0) = g(0,1,1,0,1) = f(0,1).$$

The vectors $\mathbf{y} \in \mathbb{F}_2^5$ at distance > 1 from all the codewords are mapped to $*$.

Now, the MTBDD for g can be constructed and reduced to obtain the robust BDD for all 2-variable functions (Figure 2). To get the MTBDD of a specific binary function, replace the labels $f(\mathbf{x})$ with the actual value of the function and reduce. For example, to get the MTBDD of the function f of Example 2, assign the value 0 to the nodes labeled $f(0,0)$ and $f(1,0)$, and the value 1 to the nodes labeled $f(0,1)$ and $f(1,1)$.

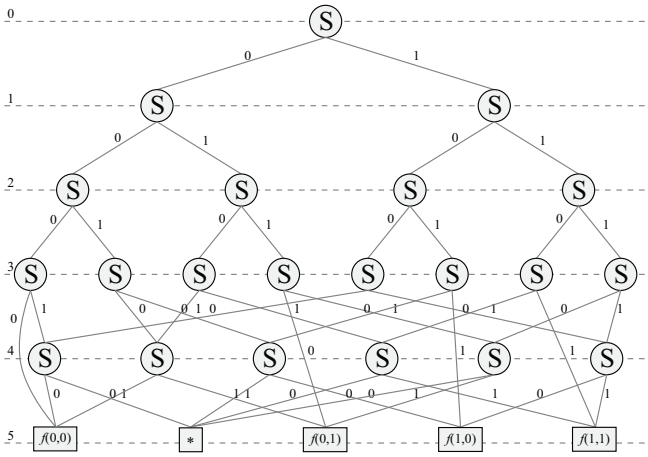

Figure 2: A robust MTBDD for 2-variable functions.

With this approach, the procedure of generating the robust decision diagram goes as follows:

1. Multiply each vector $\mathbf{x}_i \in \mathbb{F}_q^k$ by the generator matrix $\mathbf{G}$ of the desired (n,k)-code to obtain $\mathbf{c}_i \in \mathbb{F}_q^n$.

2. For each $\mathbf{c}_i$, list all the vectors $\mathbf{y}_{i,1}, \mathbf{y}_{i,2}, \ldots, \mathbf{y}_{i,m}$ of $\mathbb{F}_q^n$ within distance e from each $\mathbf{c}_i$.

3. To obtain the function g, assign the value $f(\mathbf{x}_i)$ to each $\mathbf{c}_i$ and to the corresponding set of $\mathbf{y}_{i,1}, \mathbf{y}_{i,2}, \ldots, \mathbf{y}_{i,m}$.

4. Map each $\mathbf{y} \in \mathbb{F}_q^n$ at distance $> e$ from all the codewords $\mathbf{c}_i$ to $*$.

5. Construct a MTDD for the function g and reduce it.

6. To obtain the robust decision diagram of a specific function f, assign the values of f into the terminal nodes of the MTDD of the function g and reduce.

4.3. Error Correcting BDDs using Hamming Codes

Binary Hamming codes are particularly suitable for generating robust BDDs for binary functions, since they are perfect codes. This means, that the whole space $\mathbb{F}_2^n$ will

be covered by spheres of radius 1, and each n-tuple of $\mathbb{F}_2^n$ is therefore within distance 1 from some codeword of the (n,k)-Hamming code. It follows, that no $*$-valued nodes will be needed in the constructed BDD.

In the case of binary Hamming codes, the parameters of the code are $n = 2^m - 1$ and $k = 2^m - m - 1$. Therefore, for each binary function with $k = 2^m - m - 1$ variables, we can find a Hamming code and construct the robust BDD using this code. The properties of the Hamming code guarantee that by following the BDD, the correct function value is obtained even if one decision error occurs during the determination of the function value. In other words, when following the edges of the decision diagram, even if we take the wrong turn once, we will end up at the right function value.

The procedure of generating the robust BDD for a specific function as in section 4.1 goes as follows. Let $f : \mathbb{F}_2^k \to \mathbb{F}_2$ be an arbitrary binary function, where $k = 2^m - m - 1$. For each such k, there is a Hamming code with parameters $(2^m - 1, k)$. Now, the function f can be represented by a function $g : \mathbb{F}_2^{2^m-1} \to \mathbb{F}_2$, where each k-tuple $\mathbf{x} \in \mathbb{F}_2^k$ corresponds to a $(2^m - 1)$-tuple $\mathbf{c}$ obtained by multiplying the generator matrix $\mathbf{G}$ of the code by $\mathbf{x}$. The function g is obtained by mapping each such codeword and each word within distance 1 of the codeword to the value $f(\mathbf{x})$. The reduced BDD of the function g is the robust diagram for the function f, which can correct single bit errors.

With the approach introduced in 4.2, each $(2^m - 1)$-tuple $\mathbf{c}$ and all vectors of length $2^m - 1$ within distance 1 from $\mathbf{c}$ are assigned the label $f(\mathbf{x})$. This gives a function $g : \mathbb{F}_2^{2^m-1} \to \{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{2^k})\}$ for which we can generate a multiterminal binary decision diagram, which will have in total 2^{2^m-1} terminal nodes. After reducing the obtained decision diagram, we have a diagram with 2^k terminal nodes. When assigning each terminal node with a value from $\mathbb{F}_2$, we can represent all binary functions of k variables by this robust construction. The BDD of a specific function f is then obtained by assigning the values of the function to the terminal nodes and reducing with respect to those values.

By shortening an existing linear code, it is possible to form a new linear code, which has some of the properties of the original code. The shortening process is usually done by taking the subspace of the chosen (n,k) code consisting of all codewords, which begin by 0. The 0 is then deleted from the beginning, giving a new linear code of parameters $(n-1, k-1)$. For example, by shortening the Hamming $(7,4)$ code we get a $(6,3)$ code, which has the same minimum distance and a generator matrix

$$\mathbf{G} = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right].$$

However, this code is not perfect.

Example 4. *Take the full adder of 2 variables. With the carry in, it is a function of 3 variables, which has two outputs (Figure 3). We can construct a robust decision diagram for the full adder by using the shortened Hamming code of parameters* $(6,3)$.

As in Example 2, the function is mapped into a function of a larger domain by multiplying the generator matrix of the code by the vectors of $\mathbb{F}_2^3$. *The MTBDD of the new function is then constructed, giving a robust decision diagram for the original function (Figure 4).*

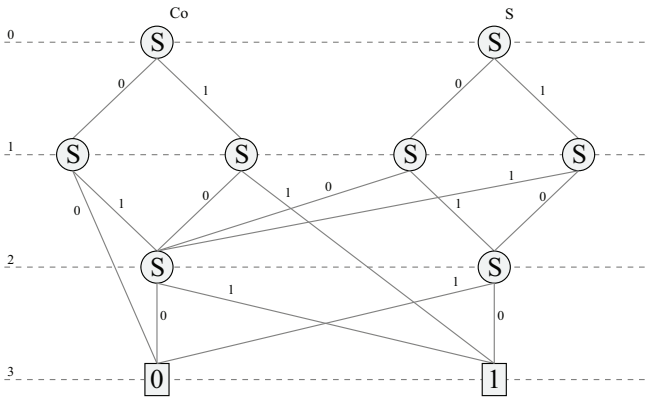

Figure 3: A shared decision diagram of the 2-variable full adder.

5. CONCLUSIONS

In this paper we presented a method for constructing robust decision diagrams and error-correcting codes. Decision diagrams are useful in logic design, since the layout of the diagram determines the layout of the circuit. We combined the theory of error-correcting codes with decision diagrams to create diagrams, which are capable of detecting and correcting decision errors.

For this purpose, Hamming codes are particularly useful since they are perfect one error correcting codes. This means that even if a decision error occurs in one node of the diagram, the function still has the correct value in the terminal node. Basically, to get the correct function value it is possible that on one level of the diagram, every node can be faulty if nodes on none of the other levels are.

A robust decision diagram of a function is more complex than the decision diagram of the function. A more powerful code leads to a more robust diagram. With increasing robustness increases also the minimal complexity of robust diagram. However, because the constructions are based on the the theory of error-correcting codes, there are well known bounds [6] on the minimal increase in complexity.

6. REFERENCES

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

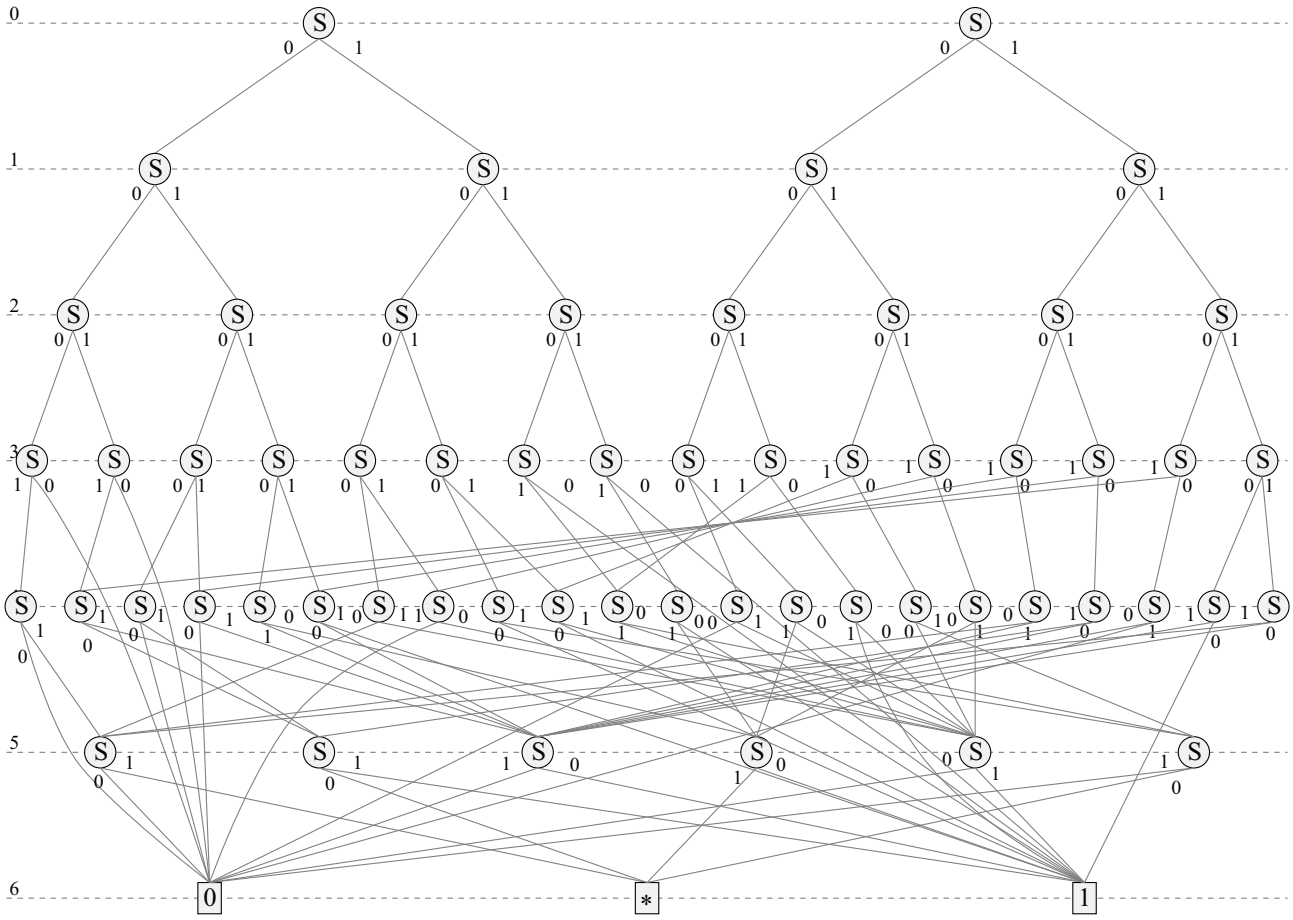

Figure 4: A shared robust diagram for the full adder of 2 variables using shortened Hamming code.

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

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