Function-driven Linearly Independent Expansions of Boolean Functions and Their Application to Synthesis of Reversible Circuits

Pawel Kerntopf[1], Marek A. Perkowski[2]
[1]Institute of Computer Science, Warsaw University of Technology, Warsaw, Poland, pke@ii.pw.edu.pl
[2]Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, mperkows@ee.kaist.ac.kr

Abstract

The paper presents a family of new expansions of Boolean functions called Function-driven Linearly Independent (fLI) expansions. On the basis of this expansion a new kind of a canonical representation of Boolean functions is constructed: Function-driven Linearly Independent Binary Decision Diagrams (fLIBDDs). They generalize both Function-driven Shannon Binary Decision Diagrams (fShBDDs) and Linearly Independent Binary Decision Diagram (LIBDDs). The diagrams introduced in the paper, can provide significantly smaller representations of Boolean functions than standard Ordered Binary Decision Diagrams (OBDDs), Ordered Functional Decision Diagrams (OFDDs) and Ordered (Pseudo-) Kronecker Functional Decision Diagrams (OKFDDs) and can be applied to synthesis of reversible circuits.

1. Introduction

Ordered Binary Decision Diagrams (OBDDs) [1] have become a popular representation of Boolean functions due to their canonicity and efficient algorithms for manipulating them [2, 13, 32, 47]. However, there exist functions of practical significance for which sizes of OBDDs are exponential in the number of variables. Variants of OBDDs have been proposed to overcome this obstacle. Efforts to modify the Shannon expansion have led to especially fruitful research. New useful bit-level and word-level decision diagrams have been developed, among them Functional Decision Diagrams (FDDs), Kronecker Functional Decision Diagrams (KFDDs), Pseudo-Kronecker Functional Decision Diagrams (PKFDDs) and Binary Moment Diagrams (BMDs) [2, 3, 13, 16, 32, 47]. Many researchers have also studied BDD transformations [32, 13, 10-12, 14, 18, 20, 31-32]. Nevertheless, the search for more compact representations of Boolean functions, which preserve good algorithmic features of OBDDs (canonicity, simple reduction rules, efficient manipulation algorithms), is still ongoing [13].

Although many algorithms have been developed to minimize the size of the BDDs, the efficient representation and manipulation of Boolean functions is critical to some computer-aided design applications including logic synthesis, formal verification and testing. For BDD-based design tools, the size of the BDDs can determine their run-time efficiency, the problem size that they can handle and/or the quality of hardware or software they synthesize.

Recently, two more generalizations of the Shannon expansion have been studied, namely Linearly Independent Logic (LI Logic) [36-37, 44, 38, 5-7, 41, 43, 45, 8-9, 40] and Function-driven expansions [17-19, 23]. The first one expands a function over a set of variables and with respect to a set of (linearly independent) basis functions. The second expansion is based on an observation that replacing a variable in the Shannon formula by an arbitrary Boolean function depending linearly on that variable determines a *unique* pair of Boolean functions called function-driven generalized cofactors. In this paper, we show that it is possible to combine these two concepts thus enabling to define Function-driven Linearly Independent DDs that can be significantly more compact than standard BDDs and FDDs. We also present a possible solution to the important current problem of synthesizing reversible circuits.

The paper is organized as follows. In Section 2 previous work is summarized. In Section 3 new types of decision trees, generalized forms (expressions) and decision diagrams based on the new expansion are defined. Section 4 presents an application of fLIBDDs to synthesis of reversible circuits. Finally, conclusions are formulated. We assume that the reader is familiar with OBDDs, OFDDs and OKFDDs (the reader is referred to [2, 13, 32, 47] for explanation of basic notions and algorithms).

2. Previous work

Binary Decision Diagrams (BDDs) are based on the so-called Shannon expansion of a Boolean function *f*:

$$f = x_i' f^0_i + x f^1_i$$

where x_i is a variable of *f*, x_i' is the negation of x_i, f^0_i and f^1_i are cofactors of *f* obtained by replacing *x* by constants, respectively *0* and *1* (Figure 1a presents a node type for this expansion).

Two well-known expansions used for defining popular variants of decision diagrams are based on the following modifications of the Shannon expansion:

$$f = f^0_i \oplus x_i f^2_i \quad \text{- positive Davio expansion}$$
$$f = f^1_i \oplus x_i' f^2_i \quad \text{- negative Davio expansion,}$$

where $f^2_i = f^0_i \oplus f^1_i$. Figure 1b presents a node type for positive Davio expansion (in a node type for negative Davio expansion the label attached to 1-edge is x_i' instead of x_i). On the basis of these expansions Functional Decision Diagrams, Kronecker Functional Decision Diagrams and Pseudo-Kronecker Functional Decision Diagrams have been introduced [16, 3, 47, 2, 32].

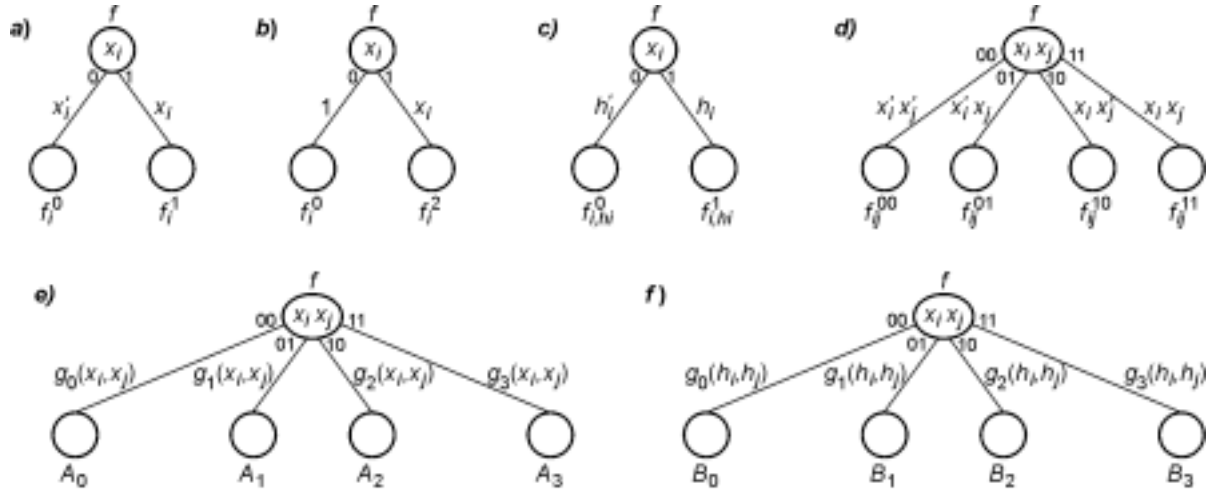

Figure 1. Types of expansion nodes: a) Shannon for one variable, b) positive Davio for one variable, c) function-driven Shannon for one variable, d) Shannon for variable pair, e) LI for variable pair, f) fLI for variable pair.

Let us now consider the Shannon expansion of a Boolean function f with respect to a variable pair:

$$f = x_i' x_j' f^{00}_{ij} + x_i' x_j f^{01}_{ij} + x_i x_j' f^{10}_{ij} + x_i x_j f^{11}_{ij}$$

where x_i and x_j are variables of f; functions f^{00}_{ij}, f^{01}_{ij}, f^{10}_{ij}, f^{11}_{ij} are cofactors of f obtained by replacing x_i and x_j by constants, respectively *00*, *01*, *10* and *11* (Figure 1d presents a node type for this expansion).

Linearly Independent (LI) expansions with respect to a variable pair are analogs of Davio expansions:

Theorem 1 [40] Let $f(x_1, x_2, ..., x_n)$ be an arbitrary Boolean function. The following formula holds true:

$$f = g_0(x_1,x_2)\, A_0(x_3, \dots, x_n) \oplus g_1(x_1,x_2)\, A_1(x_3, \dots, x_n) \oplus g_2(x_1,x_2)\, A_2(x_3, \dots, x_n) \oplus g_3(x_1,x_2)\, A_3(x_3, \dots, x_n)$$

where $g_0(x_1, x_2)$, $g_1(x_1, x_2)$, $g_2(x_1, x_2)$, $g_3(x_1, x_2)$ are the so-called (linearly independent) basis functions, $A_0(x_3, \dots, x_n)$, $A_1(x_3, \dots, x_n)$, $A_2(x_3, \dots, x_n)$, $A_3(x_3, \dots, x_n)$ are the so-called data functions (in other words, generalized cofactors w.r.t. g_0, g_1, g_2, g_3).

Ordered sets of 2-variable functions (g_0, g_1, g_2, g_3) uniquely determining the functions $A_0(x_3, \dots, x_n)$, $A_1(x_3, \dots, x_n)$, $A_2(x_3, \dots, x_n)$, and $A_3(x_3, \dots, x_n)$ are called nonsingular because they correspond to nonsingular 4x4 0-1 matrices, i.e. matrices in which the set of rows is linearly independent with respect to bit-by-bit EXOR-ing of rows (there exist 840 nonsingular 4x4 0-1 matrices). Nonsingular expansions exist for basis functions of arbitrary number of variables $k \geq 1$. For $k = 1$ they are identical with Functional expansions (Shannon, positive Davio or negative Davio). For the lack of space, in this paper we consider exclusively LI expansion with respect to a variable pair (or $k = 2$ case). Figure 1e presents a node type for this expansion. Let us recall that in standard Kronecker tree only one of Shannon, positive Davio and negative Davio expansions is used in every level of the tree and in standard Pseudo-Kronecker tree any of these expansions may be used in each node of a tree. In LI (Pseudo-) Kronecker trees any of 840 expansions with respect to a variable pair may be used in an analogous manner.

Similarly to Davio expansions the functions $A_0(x_3, \dots, x_n)$, $A_1(x_3, \dots, x_n)$, $A_2(x_3, \dots, x_n)$, and $A_3(x_3, \dots, x_n)$ can be expressed as EXOR sums of subsets of the set of cofactors $\{f^{00}_{12}, f^{01}_{12}, f^{10}_{12}, f^{11}_{12}\}$.

Example 1 The set of basis functions $g_0 = x_1 + x_2$, $g_1 = x_2'$, $g_2 = x_1'$, $g_3 = 1$ defines a nonsingular expansion. Using nonsingular matrices (see [40]) it can be shown that

$$A_0 = f^{00}_{12} \oplus f^{01}_{12} \oplus f^{10}_{12} \oplus f^{11}_{12}, \qquad A_1 = f^{10}_{12} \oplus f^{11}_{12}, \qquad A_2 = f^{01}_{12} \oplus f^{11}_{12}, \qquad A_3 = f^{00}_{12} \oplus f^{01}_{12} \oplus f^{11}_{12}.$$

Selecting good basis functions for a given set of expansion variables is a hard problem due to a very large search space. Some algorithms for finding such selections have been described in [40]. A similar problem was considered in [47], although for a different case. Namely, it is the problem of selecting good transformations of a binary input multi-valued output function with respect to one variable (using nonsingular matrices) with the aim of reducing the number of product terms in ESOPs.

Before introducng function-driven Shannon expansion we will need two additional notions.

Definition 1 A Boolean function in n variables $h(X) = h(x_1, x_2, \dots, x_n)$ is called *linear w.r.t.* x_i if it may be written as $h(x_1,\dots,x_{i-1}\, x_i, x_{i+1},\dots\, x_n) = x_i \oplus g(x_1 \dots\, x_{i-1}, x_{i+1},\dots,x_n)$, where the function g does not depend on the variable x_i.

Definition 2 Let $f(x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$ be a Boolean function and $h(x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$ be a Boolean function linear w.r.t. x_i. Let us define the function obtained from f by substituting the function h in place of the variable x_i:

$$f_{i,h}(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, h, x_{i+1}, \ldots, x_n).$$

The functions $f^0_{i,h} = f_{i,h}(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$ and $f^1_{i,h} = f_{i,h}(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$ are called *negative and positive function-driven generalized cofactors of f w.r.t. x_i and the function h*, respectively.

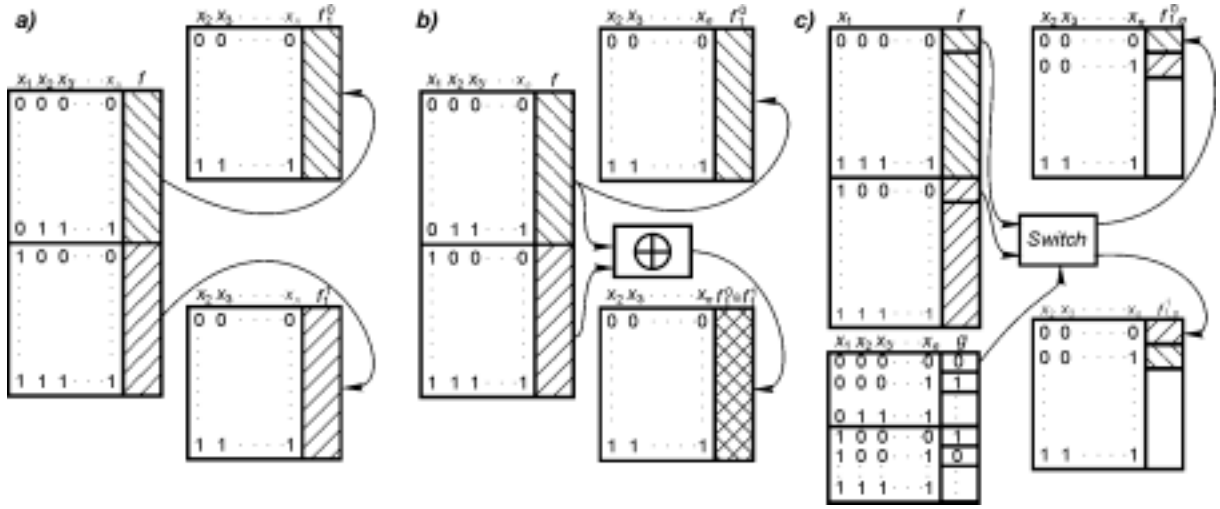

Figure 2. Types of expansions for the variable x_1: a) Shannon, b) positive Davio, c) function-driven Shannon.

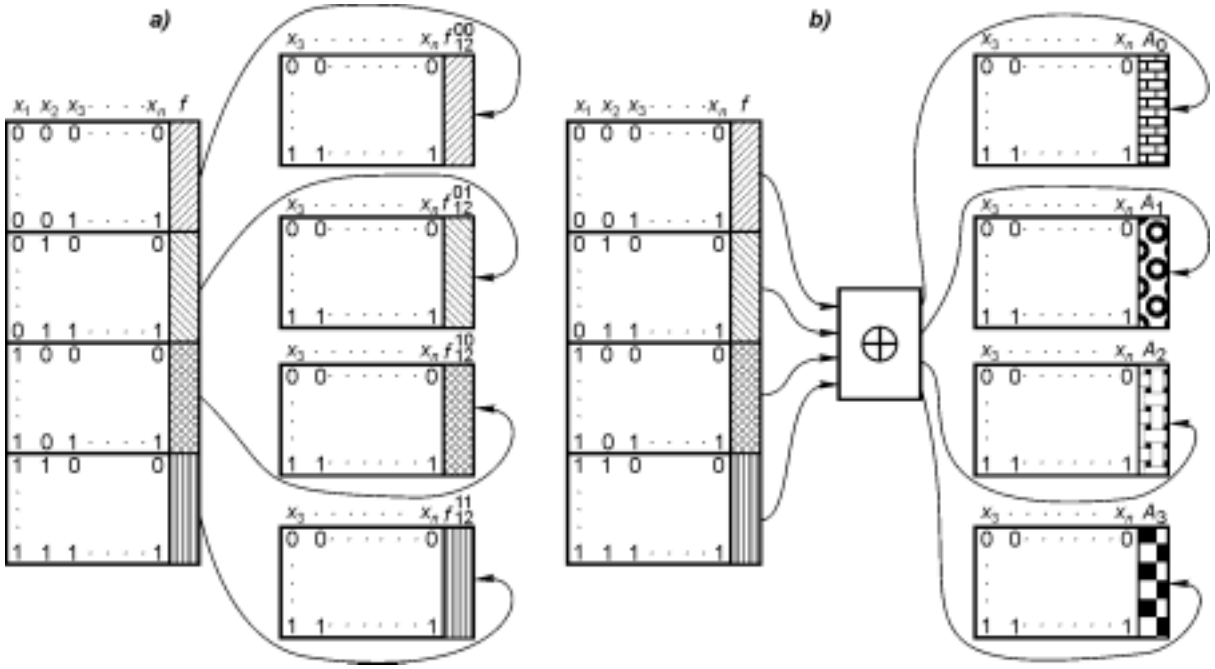

Figure 3. Types of expansions for variable pair $\{x_1, x_2\}$: a) Shannon, b) linearly independent.

As f^0_h and f^1_h are uniquely determined by an arbitrary function h linear w.r.t. a variable of f it is possible to use them for defining new types of decision diagrams (called Function-driven Shannon Binary Decision Diagrams) in the same manner as it is done for standard BDDs. Note that f^0_i and f^1_i (similarly to the standard cofactors f^0 and f^1) do not depend on the variable x_i.

Example 2 Let

$$f(x_1, x_2, x_3, x_4) = x_1 \oplus x_1x_3 \oplus x_1x_4 \oplus x_2x_3 \oplus x_2x_4 \oplus x_1x_2x_4 \oplus x_2x_3x_4.$$

We will determine function-driven generalized cofactors of f w.r.t. x_1 and

$$h = 1 \oplus x_1 \oplus x_4 \oplus x_3x_4:$$

First the function $f_{1,g}$ has to be determined:
$f_{1,h}(x_1, x_2, x_3, x_4) = f(h, x_2, x_3, x_4) =$
$= (1 \oplus x_1 \oplus x_4 \oplus x_3x_4) \oplus (1 \oplus x_1 \oplus x_4 \oplus x_3x_4)\, x_3 \oplus (1 \oplus x_1 \oplus x_4 \oplus x_3x_4)\, x_4 \oplus x_2x_3 \oplus x_2x_4 \oplus (1 \oplus x_1 \oplus x_4 \oplus x_3x_4)\, x_2x_4 \oplus x_2x_3x_4 =$
$= 1 \oplus x_1 \oplus x_3 \oplus x_4 \oplus x_1x_3 \oplus x_1x_4 \oplus x_2x_3 \oplus x_2x_4 \oplus x_1x_2\, x_4$
Hence $f^0_{1,h} = 1 \oplus x_3 \oplus x_4 \oplus x_2x_3 \oplus x_2x_4$ and $f^1_{1,h} = x_2x_3$.

The theorem below presents a function-driven Shannon expansion.

Theorem 2 [19] Let $f(x_1, x_2, \dots, x_n)$ be an arbitrary Boolean function. The following formula holds true:

$$f = h' f^0_{i,h} + h f^1_{i,h}$$

where $h(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n)$ is an arbitrary function linear w.r.t. x_i, $i = 1, 2, \dots, n$. It is called *function-driven Shannon expansion w.r.t. x_i and the function h* (in short, *h*-fS *expansion* or simply fS *expansion)*.

Functions linear w.r.t. x_i ($i=1,2,\dots,n$) form the set of all functions g such that for the given function f there exists a unique solution of the above equation with respect to the unknowns $f^0_{i,h}$ and $f^1_{i,h}$. Hence, for each variable x_i there exist as many fS expansions as there are functions linear w.r.t. x_i, $i=1,2,\dots,n$ (Figure 1c presents a node type for this expansion). Similarly to Shannon expansion also function-driven Shannon expansion corresponds to a multiplexer operation but with the function h as the controlling variable instead of a variable x_i.

It is easy to see that Shannon expansion is a special case of function-driven Shannon expansions. Thus, for each BDD there exist smaller or equal size decision diagrams based on fS expansions (called fShBDDs [23]). It has been proved that some linearly as well as nonlinearly transformed BDDs (being a subset of fShBDDs) can represent in polynomial size such functions that have only exponential size BDDs (the proof for linear case is given in [12], and for nonlinear case in [30]).

Function-driven expansions can be combined with Reed-Muller (Davio) expansions:

Theorem 3 [19] Let $f(x_1, x_2, \dots, x_n)$ be an arbitrary Boolean function. The following two expansions hold true

$f = f^0_i \oplus h f^2_{i,h}$ - *function-driven positive Reed-Muller (Davio) decompositions* (in short, *h*-fpD)

$f = f^1_i \oplus h' f^2_{i,h}$ - *function-driven negative Reed-Muller (Davio) decompositions* (in short, *h*-fnD),

where $h(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n)$ is an arbitrary x_i-*SD* function, $i=1,2,\dots,n$, and $f^2_{i,h} = f^0_{i,h} \oplus f^1_{i,h}$..

3. Function-driven Linearly Independent Expansions, Trees, Forms and Diagrams

Function-driven Linearly Independent expansions with respect to a variable pair are defined as follows:

Theorem 4 Let $f(x_1, x_2, \dots, x_n)$ be an arbitrary Boolean function. The following formula holds true:

$$f = g_0(h_1,h_2)\, A_0(x_3, \dots, x_n) \oplus g_1(h_1,h_2)\, A_1(x_3, \dots, x_n) \oplus g_2(h_1,h_2)\, A_2(x_3, \dots, x_n) \oplus g_3(h_1,h_2)\, A_3(x_3, \dots, x_n)$$

where $g_0(x_1, x_2)$, $g_1(x_1, x_2)$, $g_2(x_1, x_2)$, and $g_3(x_1, x_2)$ are the so-called (linearly independent) basis functions, $h_1(x_1, x_2, x_3, \dots, x_n)$ and $h_2(x_1, x_2, x_3, \dots, x_n)$ are such functions that for each $(a_3, \dots, a_n)$ all values of functions $h_1(x_1, x_2, a_3, \dots, a_n)$ and $h_2(x_1, x_2, a_3, \dots, a_n)$ of two variables x_1, x_2 form the set $B=\{00,01,10,11\}$ (i.e. h_1, h_2 define a bijective mapping from B onto B), $A_0(x_3, \dots, x_n)$, $A_1(x_3, \dots, x_n)$, $A_2(x_3, \dots, x_n)$, and $A_3(x_3, \dots, x_n)$ are the so-called data functions (in other words, generalized cofactors with respect to the functions g_0, g_1, g_2, g_3 and h_1, h_2). Figure 1f presents a node type for this expansion.

Proof Function-driven Linearly Independent (fLI) expansion may be considered as a composition of two bijective mappings: first the function-driven mapping is done and next the LI mapping is applied to the result of the previous mapping. Thus, it is a bijective mapping given by the expanding formula.

All expansions considered in this paper have a simple interpretation in terms of truth tables of functions. In Shannon expansions the function vector (the output column) is partitioned into halves (Figure 2a). In Davio expansions, one of these halves is replaced by EXOR bit-by-bit sum of cofactors (Figure 2b). In function-driven expansions the corresponding pairs of bits belonging to different cofactors can be flipped depending on the values of the controlling function g (Figure 2c) In Shannon expansions for a variable pair, the function vector is partitioned into four subvectors (cofactors) of equal size (Figure 3a). In LI expansions for a variable pair four generalized cofactors are equal to EXOR bit-by-bit sums of subsets of standard cofactors (Figure 3b). Similarly, in fLI expansions this is generalized to EXOR bit-by-bit sums of four generalized cofactor vectors (obtained through permutations of corresponding bits in four Shannon cofactor vectors, depending on the values of two controlling functions h_1, h_2).

Below we define trees, forms and decision diagrams corresponding to repeated use of fLI expansions or flattening of fLI expansions.

Definition 3 The *fLI Kronecker Tree* is created as follows. The set of all input variables is partitioned into an ordered set of disjoint and nonempty subsets (blocks). The blocks are ordered. Each of them corresponds to a level of the tree. For each level of the tree a fLI expansion is selected. For the block with n variables there are 2^n children nodes of a node.

Definition 4 *The fLI Forms* are obtained by flattening the fLI trees. Flattening has the following stages:

1. find all paths of the tree that lead from the root to constant 1 in terminal nodes,
2. for each such path make a product term by multiplying the labels (expressions) attached to the edges of the path,
3. make EXOR sum of these product terms.

Definition 5 (Reduced) *fLI binary decision diagrams* (fLIBDDs) are created from respective fLI trees by the rules:

1. share isomorphic subdiagrams,
2. if all outgoing edges of a node point to the same node, then delete the node and connect the incoming edges of the deleted node to the corresponding successor. Relabel the edges, e.g. if there was only one incoming edge of the deleted node, labeled by *a,* and there were 4 outgoing edges of this node, labeled by *b, c, d, e,* respectively, then label the new edge by $a(b \oplus c \oplus d \oplus e)$, etc.

4. Application of fLIBDDs to Synthesis of Reversible Circuits

Although advantages of using reversible computing have been known for many years all papers published in 1980s and 1990s were concerned mostly with proposing simple binary reversible gates and studying universality as well as basic properties of these gates. Research on synthesis of reversible logic circuits has started only very recently [42, 48, 15, 35, 27, 39, 24-26, 28, 33-34, 29, 4]. Generalized families of binary and multiple-valued gates have also been proposed [24-27, 35, 39].

As mentioned in the previous section an fLIBDD representing a Boolean function $f(x_1, x_2, \ldots, x_n)$ corresponds to a sequence of bijective transformations on the domain of *f* determined by the defining functions of function-driven and LI expansions. It can be interpreted as a decomposition of a circuit realizing the function *f* into two parts: a preprocessor performing a domain transformation (or a composition of transformations) followed by a circuit realizing the transformed function (see Figure 4). On the other hand, a reversible gate (or circuit) also corresponds to a bijective mapping (a domain transformation). Thus the preprocessor may be built up using reversible gates as there is a one-to-one correspondence between the reversible gates in the preprocessor and the transformations determined by defining functions. This approach would be useful in designing layout for quantum logic circuits because it is easier to design such layouts when the preprocessor is made exclusively of (generalized) Feynman and Toffoli gates (note that the defining functions for function-driven expansions can be expressed using such gates).

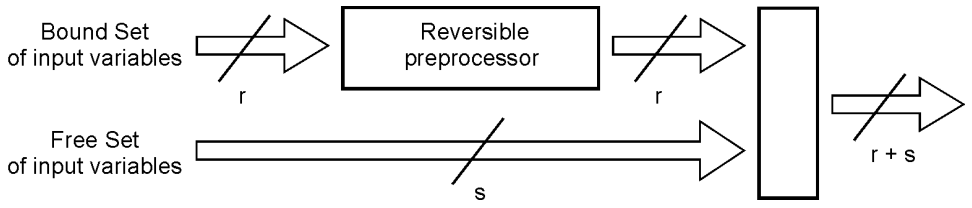

Figure 4. General scheme of new decomposition type.

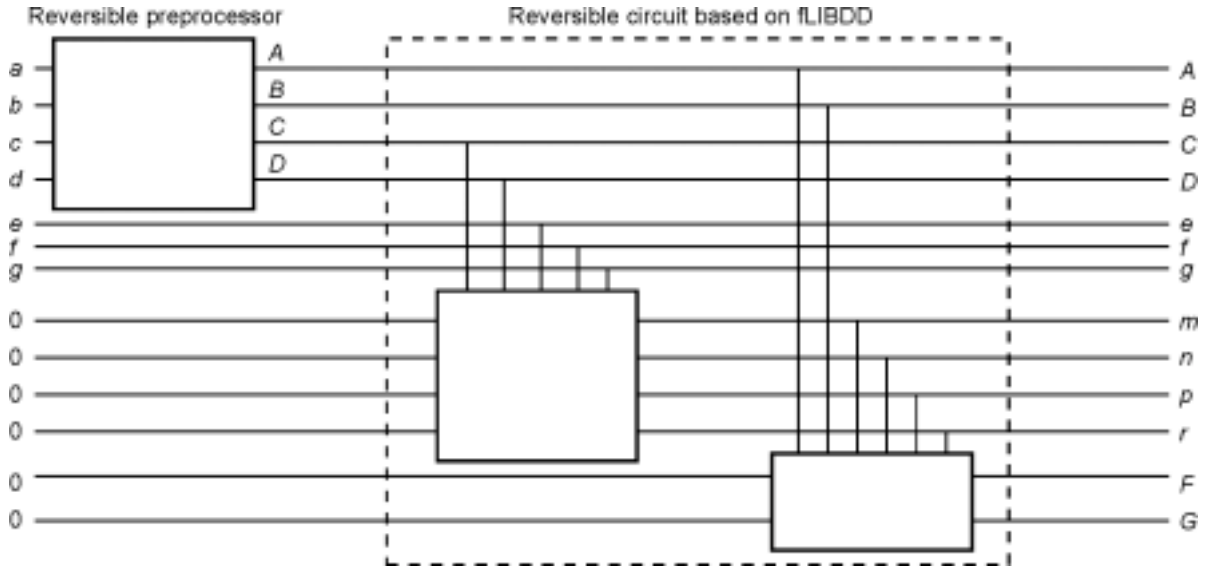

Figure 5. Decomposition on a reversible preprocessor and a reversible circuit based on an fLIBDD.

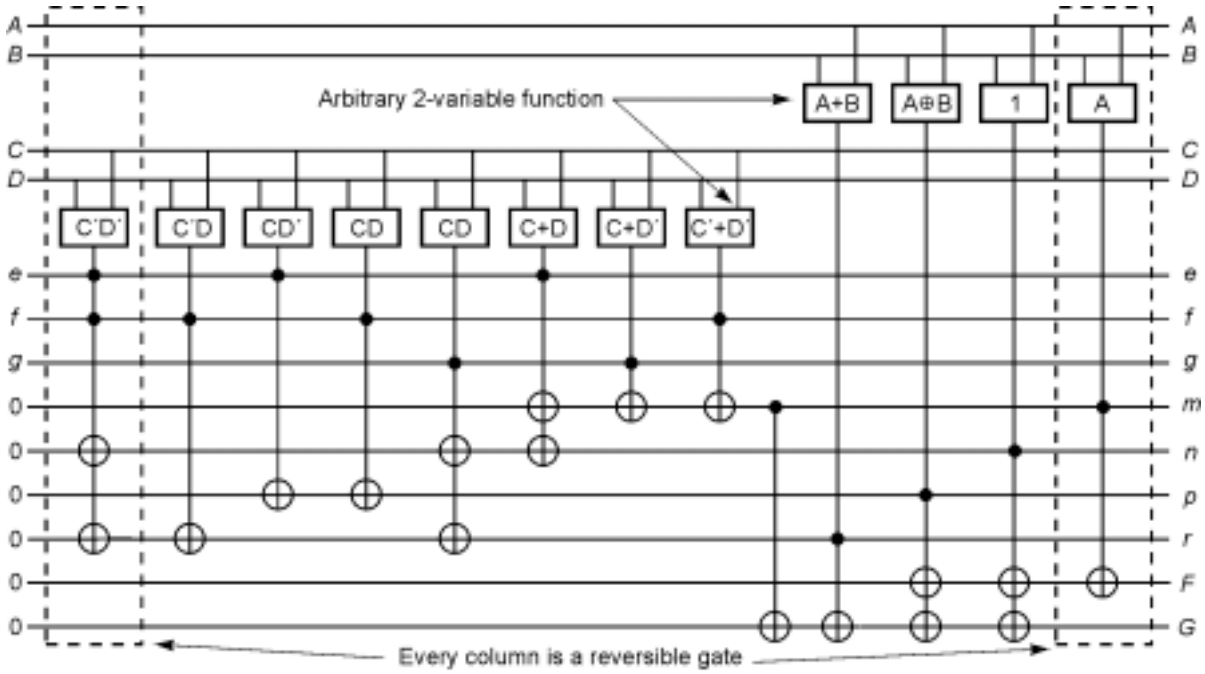

Figure 6. Reversible circuit realizing a two-output circuit defined in Example 3.

We sketch below the idea of reversible logic decomposition shown in Figure 4. This decomposition is illustrated here by using an example based on an fLIBDD (not shown to save space).

Example 3 The input data to the decomposition are two incompletely specified functions *F(a,b,c,d,e,f,g)*, *G(a,b,c,d,e,f,g)*, given in form of ON and OFF BDDs. Functions $h_1(a,b,c,d,e,f,g)$ and $h_2(a,b,c,d,e,f,g)$ are found that reduce the variables *a* and *b* from functions *F* and *G* realized as fLIBDDs. These functions h_1 and h_2 are control inputs to the first level nodes of the fLIBDD (corresponding to the output functions *F* and *G*, respectively). To realize these functions, any method of realizing reversible circuits can be used, and particularly the cascade methods [35, 27, 33-34, 24-25, 4, 29] that produce no garbage or little garbage signals. Functions h_1 and h_2 are realized as a reversible cascade with inputs *a, b, c, d, e, f, g* and outputs $A = h_0$, $B = h_1$, *c, d, e, f, g*. The outputs *c, d, e, f, g* of this cascade go to the next reversible cascade that has the inputs *c, d, e, f, g* and the outputs $C = i_0(c,d,e,f,g)$, $D = i_1(c,d,e,f,g)$, *e, f, g*. This second cascade reduces input variables *c* and *d*. Thus, analogously, every next cascade in the preprocessor reduces a pair of original input variables and decreases the complexity of the fLIBDD on transformed variables *A, B, C, D,...* . After finding such a preprocessor, being a sequence of cascades for variable pairs, the fLIBDD is greatly reduced. Sometimes, like in this case, it is sufficient to do the transformation only for a subset of input variables (here the variables *e, f, g* remain unaffected because the useful functions depending on them in fLIBDD are simple).

The block diagram for the circuit obtained from this fLIBDD is shown in Figure 5 (surrounded by a dashed line). In this diagram every block has, in general , two control inputs and four data inputs, but because the functions $A_i(x_3,...,x_n)$ can be constants, the number of data inputs can be smaller, as in our example (the left block in the reversible circuit shown in Figure 5 has only three data inputs *e, f, g*). The schematic from Figure 5 can be rewritten to ESOP-like-based quantum-PLA notation (Figure 6), where the columns correspond to AND-ing done in Toffoli-like gates and rows collect subfunctions by the EXOR-ing operation. The equations for this schematics are as follows:

$$F = = \boldsymbol{(A \oplus B)}p \oplus \boldsymbol{1}\, n \oplus \boldsymbol{A}m \oplus \boldsymbol{(A + B)}\, 0 = \boldsymbol{(A \oplus B)}p \oplus n \oplus \boldsymbol{A}m$$

$$G = \boldsymbol{(A + B)}r \oplus \boldsymbol{(A \oplus B)}p \oplus \boldsymbol{1}\,(m \oplus n) \oplus \boldsymbol{A}\, 0 = \; n \oplus \boldsymbol{(A + B)}r \oplus \boldsymbol{(A \oplus B)}p \oplus m,$$

where the product terms corresponding to basis functions are written in bold, and the data functions with respect to them are:

$$m = \boldsymbol{(C+D)}e \oplus \boldsymbol{(C+D')}g \oplus \boldsymbol{(C'+D')}f \oplus \boldsymbol{(C'+D)}\, 0$$

$$n = \boldsymbol{(C'D')}ef \oplus \boldsymbol{(CD)}g \oplus \boldsymbol{(C+D)}e \oplus \boldsymbol{(C'D)}\, 0$$

$$p = \boldsymbol{(CD')}e \oplus \boldsymbol{(CD)}f \oplus \boldsymbol{(C'D')}\, 0 \oplus \boldsymbol{(C'D)}\, 0$$

$$r = \boldsymbol{(C'D')}ef \oplus \boldsymbol{(C'D)}f \oplus \boldsymbol{(CD)}g \oplus \boldsymbol{(CD')}\, 0$$

When a small preprocessor for each pair of original input variables has been found, a simplified decision diagram is created that is next efficiently realized in a reversible circuit , paying only the price of increased quantum register width, but greatly saving on the number of ESOP terms as compared to a solution obtained directly from a decision diagram or from an ESOP of functions *F(a,b,c,d,e,f g) , G(a,b,c,d,e,f,g)* on original variables *a, b, c, d, e, f, g*.

The question remains how to find good preprocessors and thus some iteration and heuristics are needed. The most efficient optimization of BDDs has been done by variable ordering heuristics using the modifications of the well-known sifting algorithm. This method can also be generalized for minimization of fLIBDDs and thus for synthesis of reversible circuits. It seems to these authors that methods similar to nonlinear sifting should be used [21-22, 30].

5. Conclusions

This paper has presented new generalizations of Binary Decision Diagrams and Functional Decision Diagrams. They are called Function-driven Linearly Independent Binary Decision Diagrams (fLIBDDs) and can be used as canonical representations of Boolean functions. By using fLIBDDs it is possible to decrease substantially sizes of diagrams what is important in many applications. It is also possible to extend the notions of shared diagrams and complemented edges to the new types of decision diagrams as well as to extend fLIBDDs to multiple-valued decision diagrams. We expect that the fLIBDDs will find applications in compact Boolean function representation and multi-level logic synthesis with reversible gates, in particular quantum circuits.

6. References

[1] Bryant R.E., Graph-based algorithms for Boolean function manipulation, IEEE Trans. on Computers, vol. 35, 1986, pp. 677-691.

[2] Drechsler R., Becker B., *Ordered Binary Decision Diagrams. Theory and Implementation*, Kluwer Academic Publishers, Boston/Dordrecht/London 1998.

[3] Drechsler R., Sarabi A., Theobald M., Becker B., Perkowski M.A., Efficient representation and manipulation of switching functions based on ordered Kronecker functional decision diagrams, *Proc. Design Automation Conference*, 1994, pp. 415-419.

[4] Dueck G., Maslov, D. Reversible function synthesis with minimum garbage outputs, *Proc. 6th Int'l Symp. on Representations and Methodology of Future Computing Technology*, 2003, pp. 154-161.

[5] Falkowski B., Rahardja S., Fast transforms for orthogonal logic, *Proc. 28th IEEE Int. Symp on Circuits and Systems*, 1995, pp. 2164-2167.

[6] Falkowski B., Rahardja S., Family of fast transforms for GF(2) orthogonal logic, *Proc. 2nd Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, 1995, pp. 273-280.

[7] Falkowski B., Rahardja S., Classification and properties of fast linearly independent logic transformations, *IEEE Transactions on Circuits and Systems II – Analog and Digital Signal Processing*, vol. 44, 1997, pp. 646-655.

[8] Falkowski B., Rahardja S., Fast transforms for multiple-valued input binary output PLI logic, *Proc. 30th Int. Symp. on Multiple-Valued Logic*, 2000, pp. 47-52.

[9] Falkowski B., Rahardja S., PLI logic for multiple-valued functions, *IEE Proceedings, Part E – Computers and Digital Techniques*, vol. 148, 2001, pp. 7-14.

[10] Günther W., Drechsler R., Linear transformations and exact minimization of BDDs, *Proc. IEEE Great Lakes Symposium on VLSI*, 1998, pp. 325-330.

[11] Günther W., Drechsler R., Efficient manipulation algorithms for linearly transformed BDDs, *Proc. 4th Int'l Workshop on Applications of Reed-Muller Expansions*, 1999, pp. 225-232.

[12] Günther W., Drechsler R., On the computational power of linearly transformed BDDs, *Information Processing Letters*, vol. 75, 2000, pp. 119-125.

[13] Hassoun S., Sasao T., Brayton R.K. (eds.), *Logic Synthesis and Verification*, Kluwer Academic Publishers, Boston/Dordrecht/London 2002, chapter 11.

[14] Hett A., Günther W., Becker B., Application of linearly transformed BDDs in sequential verification, *Proc. Asia and South Pacific Design Automation Conference*, 2001, pp. 91-96.

[15] Iwama K., Kambayashi Y., Yamashita S., Transformation rules for designing CNOT-based quantum circuits, *Proc. Design Automation Conf.*, 2002, pp. 419-425.

[16] Kebschull U., Schubert E., Rosenstiel W., Multilevel logic synthesis based on functional decision diagrams, *Proc. European Conference on Design Automation*, 1992, pp. 43-47.

[17] Kerntopf P., Binary decision diagrams based on generalized decomposition types, *Proc. 6th Int. Conference on Mixed Design of Integrated Circuits and Systems*, 1999, pp. 95-98.

[18] Kerntopf P., Nonlinear transformations of decision diagrams, *Proc. 10th IEEE Int. Workshop on Logic and Synthesis*, 2001, pp. 173-178.

[19] Kerntopf P., New generalizations of Shannon decomposition, *Proc. 5th Int. Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, 2001, pp. 109-118.

[20] Kerntopf P., An approach to minimization of decision diagrams, *Proc. EUROMICRO Symposium on Digital Systems Design*, 2001, pp. 79-86.

[21] Kerntopf P., Nonlinear sifting of decision diagrams, *Proc.11th IEEE/ACM Int. Workshop on Logic and Synthesis*, New Orleans, USA, June 2002, pp. 85-90.

[22] Kerntopf P., Dynamic minimization of binary decision diagrams based on function-driven approach, *Proc. 9th Int. Conference on Mixed Design of Integrated Circuits and Systems*, Wroclaw, Poland, June 2002, pp. 265-270.

[23] Kerntopf P., Binary decision diagrams based on single and multiple generalized Shannon expansions, *Proc. 6th Int.Symp. on Representations and Methodology of Future Computing Technologies*, 2003, pp. 183-190.
[24] Khan M.H.A., Perkowski M., Logic synthesis with cascades of new reversible gate families, *Proc. 6th Int'l Symp. on Representations and Methodology of Future Computing Technology*, 2003, pp. 43-55.
[25] Khan M.H.A., Perkowski M., Multi-output ESOP synthesis with cascades of new reversible gate family, *Proc. 6th Int'l Symp. on Representations and Methodology of Future Computing Technology*, 2003, pp. 144-153.
[26] Khan M.H.A., Perkowski M., Kerntopf P., Multi-output Galois field sum of product (GFSOP) synthesis with new quantum cascades, *Proc. 33rd IEEE Int'l Symposium on Multiple-Valued Logic*, 2003.
[27] Khlopotine A., Perkowski M., Kerntopf P., Reversible logic synthesis by gate composition, *Proc. 11th IEEE/ACM Int'l Workshop on Logic and Synthesis*, 2002, pp. 261-266.
[28] Lukac M., Pivtoraiko M., Mishchenko A., Perkowski M., Automated synthesis of generalized reversible cascades using genetic algorithms", *Proc. 5th Int'l Workshop on Boolean Problems*, 2002, pp. 33-45.
[29] Maslov D., Dueck G., Garbage in reversible designs of multiple output functions, *Proc. 6th Int'l Symp. on Representations and Methodology of Future Computing Technology*,2003,pp.162-170.
[30] Meinel C., Mubarakzjanov R., Nonlinear sifting of decision diagrams, *Proc. Int. Conf. on VLSI*, Las Vegas, USA, June 2002, pp. 117-123.
[31] Meinel C., Somenzi F., Theobald T., Linear sifting of decision diagrams and its application in synthesis, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, 2000, pp. 521-533.
[32] Meinel C., Theobald T., *Algorithms and Data Structures in VLSI Design: OBDD - Foundations and Applications*, Springer-Verlag, Berlin/Heidelberg/New York 1998.
[33] Miller D.M., Spectral and two-place decomposition techniques in reversible logic, *Proc. Midwest Symposium on Circuits and Systems*, 2002.
[34] Miller D.M., Dueck G., Spectral techniques for reversible logic synthesis, *Proc. 6th Int'l Symp. on Representations and Methodology of Future Computing Technology*, 2003, pp. 56-62.,
[35] Mishchenko A., Perkowski M., Logic synthesis of reversible wave cascades, *Proc. 11th IEEE/ACM Int'l Workshop on Logic and Synthesis*, 2002, pp. 197-202.
[36] Perkowski M., The generalized orthonormal expansions of functions with multiple-valued inputs and some of its applications, *Proc. 22nd Int. Symp. on Multiple-Valued Logic*, 1992, pp. 442-450.
[37] Perkowski M., A fundamental theorem for EXOR circuits, *Proc. 1st Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, 1993, pp. 52-60.
[38] Perkowski M., Fundamental theorems and families of forms for binary and multiple-valued linearly independent logic, *Proc. 2nd Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, 1995, pp. 288-299.
[39] Perkowski M., Al.-Rabadi A., Kerntopf P., Multiple-valued quantum logic synthesis, *Proc. Int'l Symp. on New Paradigms VLSI Computing*, Sendai, Japan, 2002, pp.41-47.
[40] Perkowski M., Falkowski B., Chrzanowska-Jeske M., Drechsler R., Efficient algorithms for creation of linearly-independent decision diagrams and their mapping to regular layouts, *VLSI Design*, vol. 14, 2002, pp. 35-52.
[41] Perkowski M., Jozwiak L., Drechsler R., Falkowski B., Ordered and shared, linearly independent, variable-pair decision diagrams, *Proc. 1st Conf. on Information, Communications and Signal Processing*, 1997, pp. 261-265.
[42] Perkowski M., Jozwiak L., Kerntopf P., Mishchenko A., Al.-Rabadi A., Coppola A., Buller A., Xiaoyu Song, Khan M.H.A, Yanushkevich S.N., Shmerko V.P., Chrzanowska-Jeske M., A general decomposition for reversible logic, *Proc. 5th Int'l Workshop on Applications of Reed-Muller Expansion in Circuit Design*, 2001, p. 119-138.
[43] Perkowski M., Pierzchala, Drechsler R., Ternary and quaternary lattice diagrams for linearly-independent logic, multiple-valued logic and analog synthesis, *Proc. 1st Conf. on Information, Communications and Signal Processing*, 1997, pp. 269-273.
[44] Perkowski M., Sarabi A., Beyl F.R., XOR canonical forms of switching functions, *Proc. 1st Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, 1993, pp. 27-32.
[45] Rahardja S., Falkowski B., Fast linearly independent arithmetic expansions, *IEEE Transactions on Computers*, vol. 48, 1999, pp. 991-999.
[46] Sasao T., A transformation of multiple-valued input two-valued output functions and its application to simplification of exclusive-or sum-of-products expressions, *Proc. 21st Int. Symp. on Multiple-Valued Logic*, 1991, pp. 270-279.
[47] Sasao T., Fujita M. (eds.), *Representations of Discrete Functions*, Kluwer Academic Publishers, Boston/Dordrecht/London 1996.
[48] Shende V.V., Prasad A.K., Markov I.L., Hayes J.P., Reversible logic circuit synthesis, *Proc. Int'l Conf. on on Computer-Aided Design*, 2002, pp. 353-360.