UDC 519.7

Stanković Milena (Dept. of Computer Science, Serbia)

REVERSIBLE SYNTHESIS BY USING DECISION DIAGRAMS

Reversible logic synthesis has been gaining much attention recently. There are many synthesis approaches including those using spectral techniques. In this paper we present a synthesis procedure based on the transformation of the Reed-Muller spectrum of the reversible function by using Functional Decision Diagram (FDD) as the underlying data structure. Transformation of] the Reed-Muller spectrum required in the synthesis procedure will be performed trough transformation of the corresponding FDD. This procedure consists of the following: We generate FDD for the given reversible function and then we transform it into the FDD of the identity function.

Оборотному логічному синтезу останнім часом приділяється велика увага. Існує багато підходів до логічного синтезу, включаючи підходи, що застосовують спектральний апарат. В цій статті розглядається процедура синтезу, яка базується на перетворенні спектра Ріда-Малера оборотної функції за допомогою функціональної розв'язної діаграми (ФРД) в базову структуру даних. Перетворення спектра Ріда-Малера, яке потрібно зробити в процедурі синтезу, відбувається за допомогою відповідної ФРД. Ця процедура складається з наступних кроків: спочатку генерується ФРД для заданої оборотної функції, а потім вона трансформується у ФРД тотожної функції.

Introduction. The synthesis of reversible networks has received much attention in recent years, [5, 7, 8]. In particular, the interest in reversible logic is motivated by its applications in low-power computing and quantum computing.

A completely specified n-input n-output Boolean function is called reversible if it maps each input assignment to a unique output assignment and vice versa. A reversible function of n variables can be defined as a truth table or as a permutation on the set of integers $(0, 1, \ldots, 2^n - 1)$. For example, the reversible function in Table 1 can also be specified as the sequence of integers $\{7, 2, 3, 5, 4, 6, 0, 1\}$.

Table 1.

Example 1. A reversible function $f(c, b, a)$

c	b	a	c'	b'	a'
0	0	0	1	1	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	0	0	1

Definition 1. *An n-input n-output gate (orcircuit) is reversible if it realizes an $n \times n$ reversible function.*

Several reversible gates have been proposed in the literature [2,3]. In this paper, we consider realization with generalized Toffoli gates only, witch is often used to construct reversible logic circuits. This gate is defined as follows:

Definition 2. *For the domain variables* $\{x_n, x_{n-1}, \cdots, x_1\}$ *the generalized Toffoli gate has the form* $TOF(C;T)$*, where* $C = \{x_{i_1}, x_{i_2}, \cdots, x_{i_k}\}$*,* $T = \{x_j\}$*, and* $C \cap T = \Phi$*. It maps a Boolean pattern* $\{x_n, x_{n-1}, \cdots, x_{j+1}, x_j, x_{j-1}, \cdots, x_1\}$ *into* $\{x_n, x_{n-1}, \cdots, x_{j+1}, x_j \oplus x_{i_1}x_{i_2}\cdots x_{i_k}, x_{j-1}, x_1\}$*. Usually, the set* C *is called the control set and the set* T *is called the target.*

The most commonly used Toffoli gates are: the NOT gate (denoted as $TOF\{x_j\}$), the CNOT gate, which is also known as a Feynman gate ($TOF\{x_i; x_j\}$), and the original Toffoli gate denoted by ($TOF\{x_{i_1}, x_{x_2}, \cdots, x_{i_k}; x_j\}$), shown in Fig. 1(a),(b) and (c). If a reversible network is constructed by the Toffoli gates only, than is called the Toffoli network.

The problem of reversible network synthesis has received much attention in recent years and several synthesis methods are proposed [5–7,10]. Some of them are based on the transformations in the spectral Reed-Muller domain.

However, most existing synthesis method are inefficient in the case of functions with a large number of variables. In this paper we consider a reversible algorithm for synthesis of Toffoli networks, based on the Reed-Muller spectrum represented by a *Functional Decision Diagrams* (FDD). Since decision diagram are a data structure used to represent large functions, we expecting that this approach will be capable of handling reasonably large functions.

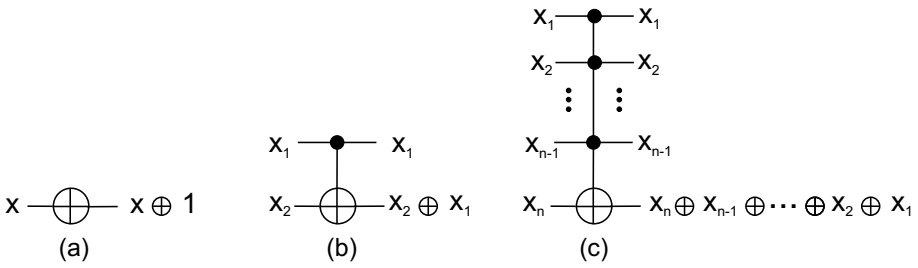

Figure 1. Main Toffoli gates

Background Theory. In this section we present some basic definitions that will be used later.

1. Reed-Muller spectrum.

Definition 3 ([5])**.** *Any n-variable Boolean function* $f(x_n, x_{n-1}, \ldots, x_1)$ *can be uniquely represented by the Reed-Muller expression* $f(x_n, x_{n-1}, \ldots, x_1) = a_0 \oplus a_1 x_n \oplus \oplus a_2 x_{n-1} \oplus \cdots \oplus a_{n+1} x_n x_{n-1} \oplus \cdots \oplus a_{2^n-1} x_n x_{n-1} \cdots x_1$ *with* $a_i \in \{0, 1\}, i = 1, 2, \ldots 2^n - 1$*. Here* $\oplus$ *denotes the Exclusive OR* (XOR) *operation. The vector of the coefficients in this expansion is called the Reed-Muller* (RM) *spectrum of* f*.*

The RM-spectrum $\mathbf{RM}_f$ of a Boolean function can be efficiently computed by using the Reed-Muller transform defined as:

$$\mathbf{RM}_f = \mathbf{M}(n)\mathbf{F}(n).$$

where

$$\mathbf{M}(n) = \begin{bmatrix} \mathbf{M}(n-1) & 0 \\ \mathbf{M}(n-1) & \mathbf{M}(n-1) \end{bmatrix} \quad \text{and} \quad \mathbf{M}(1) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

In this relation the summation is modulo-2, i.e. XOR, and $\mathbf{F}(n)$ is the truth vector of f.

2. Decision diagrams for reversible functions. Decision diagrams are a data structure convenient for representation of discrete functions. They are usually used for manipulation with functions of a large number of variables [9].

By the recursive application of the Shannon decomposition to the variables in a Boolean function $f(x_n, x_{n-1}, \ldots, x_1)$ to derive the complete disjunctive normal form, f can be represented by a *Binary Decision Tree* (BDT). The values in the terminal nodes in the BDT are the values of the function represented (elements of the truth-vector $\mathbf{F}(n)$). In a BDT, usually there are some isomorphic subtrees. Due to that, a BDT can be reduced into a *Binary Decision Diagram* (BDD). *Shared DD* (SDD) are used to represent a system of Boolean functions. Also, it is possible to represent a system of Boolean functions by one integer function which can be represented by a *Multi Terminal Decision diagram*(MTDD) [4]. We will use such representations for the reversible functions. For instance, the MTDD for the reversible function defined in Table 1 is shown in Fig. 2(a).

A BDD representing the Boolean function $f(x_n, x_{n-1}, \ldots, x_1)$ can be transformed into a *Functional Decision Diagram* (FDT), by performing the calculation defined by the transformation matrix $\mathbf{M}(1)$ in each non-terminal node of the BDD. This calculation is performed over the subtrees which ensures the efficiency of this calculation method [9].

If the function values shown in terminal nodes of the MTDD are viewed as multi-bit values and in each non-terminal node of the MTDD the calculation defined by $\mathbf{M}(1)$ is performed, we will get a FDD representing the RM-spectrum of the reversible function considered. For example, FDD for the reversible function in the Table 1 is shown in Fig. 2(b).

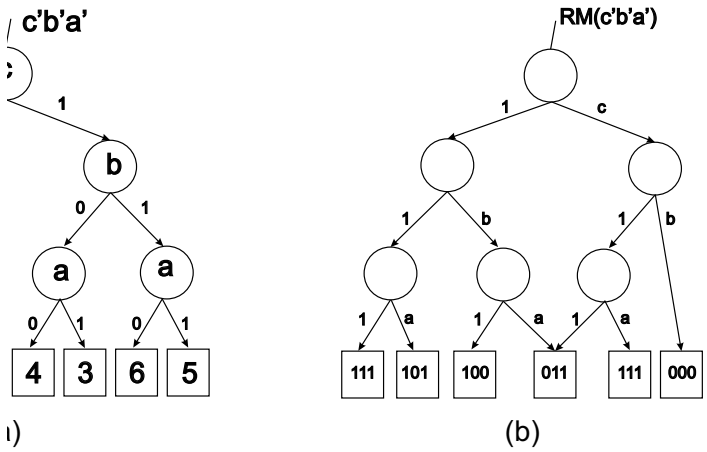

Figure 2. MTDD and FDD for the function in Table1

Iterative Synthesis by using Reed-Muller Spectrum. In [5], it is proposed a procedure for iterative Toffoli network synthesis by using the RM-spectrum of the function to be realized. Later, in the paper [6], some implementation of this procedure by using a compact representation of the RM-spectral table, to reduce runtime and memory occupation, is proposed. This method is proved to be able to speed up synthesis for many benchmark reversible functions. In this paper we will modified this procedure by using FDD of the reversible function for the iterative transformation of it and for generation of the Toffoli network.

We will start with a brief description of the basic version of the algorithm in [5]. It consists of maximum $2^n - 1$ steps (numbered 0 to $2^n - 2$). At each step i, the first i-th row of the tabular RM-spectrum of the considered reversible function is transformed into i-th row of the RM-spectrum of the identity function.

In that procedure, the $(r_n, r_{n-1}, \ldots, r_1)$ denote values in a row of the tabular representation for the RM-spectrum for the reversible function under consideration.

For different values of i, the following steps are performed:.

A: For $i = 0$. This step is unique since only in this step the NOT gates is used. Given the 0-th row has values $(r_n, r_{n-1}, \ldots, r_1)$, for every $r_j = 1, j \in \{1, 2, \ldots, n\}$ one NOT gate $TOF(x_j)$ will be applied.

B: For $i = 2^k, k = 1, 2, \ldots, n$. Each of these rows is a variable row. Such a row, $(r_n, \ldots, r_1)$, has to be transformed into the form $(0, 0, \ldots, 0, 1, 0, \ldots, 0)$ with 1 at the position k. This is done through the following two procedures. First it is necessary to check if $r_k = 1$. If it is not, r_k will be transformed to be equal 1 by assigning a gate $TOF(x_s; x_k)$ such that $s = max\{j | r_j = 1, j \in \{1.2, \ldots, n\}\}$ and $s > k$. In [5], it is shown that such an s exists, and it is verified that application of the gate $TOF(x_s; x_k)$ does not affect the RM-spectra in the rows appearing earlier in the table.

After this transformation, the row we are working with has the form $(r_n, r_{k+1}, 1, r_{k-1}, r_1)$. Next will be necessary to use gates $TOF(x_k; x_j)$ for every $r_j = 1, j \in \{1, 2, \ldots, n\}$. By applying such gates we do not change rows appearing earlier in the table than the row we are working with and at the same time the i-th row will be transformed into the desired form $(0, 0, \ldots, 0, 1, 0, \ldots, 0)$ with 1 at the position k.

C: For $i \neq 2^k, i > 0$. (Note, for those i, we know that we are working with a non-variable row). Assume it has values $(r_n, r_{n-1}, \ldots, r_1)$. It has to be transformed into the form $(0, 0, ..., 0)$, which is the form of each non-variable row of the RM-spectrum of the identity function.

First, it is necessary to find $s = max\{j | r_j = 1, j \in \{1, 2, \ldots, n\}\}\}$ such that the item 2^s does not appear in the binary expansion of i. In other words, choose a variable whose the i-th value in the RM-spectrum is 1 and which is not included in the product associated with the i-th element of the RM-spectrum. (In [5], is shown that such an s exists). The gates $TOF(x_s; x_j)$ will be applied for every $r_j = 1, j \neq i, j \in \{12, \ldots, n\}\}$. This transforms the row we are working with into $(0, 0, \ldots, 0, 1, 0, \ldots, 0)$ with 1 at the position s.

Second, it is necessary to apply the gate $TOF(X_i; x_s)$, where X_i is the product of variables x_j such that the j-th bit of the binary expansion of the number i equals 1. Such an operation transforms the row we are working with into the desired $(0, 0, \ldots, 0)$. Finally, it is necessary to undo $TOF(x_s; x_j)$ if such gates changed the RM-spectra rows appearing earlier in the table than the row i. Clearly, such "undo" operations do not change the correct form of the pattern we are working with. In the mentioned paper [5], the complete proof of convergence for the above algorithm is done.

Table 2 shows the results of synthesis of the reversible function in Table 1 by applying the algorithm presented. In this table, $T(b; c)$ denotes the Toffolli gate $TOF(b; c)$. The resulting Toffolli network is shown in Fig. 3.

Table 2.

Synthesis of reversible function in Table 1.

Coef.	Fun	Step1	Step2	Step3	Step4	Step5	Step6	Ident.
1	111	000	000	000	000	000	000	000
a	101	101	001	001	001	001	001	001
b	100	100	100	110	010	010	010	010
ab	011	011	111	101	101	100	000	000
c	011	011	111	101	101	100	100	100
ac	111	111	011	011	111	110	010	000
bc	000	000	000	000	000	000	000	000
abc	000	000	000	000	000	000	000	000
Gate	$T(a)$ $T(b)$ $T(c)$	$T(a;c)$	$T(c;b)$	$T(b;c)$	$T(c;a)$	$T(ab;c)$	$T(ac;b)$	

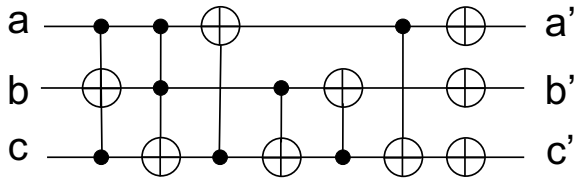

Figure 3. Toffoli network for f in Table 1.

The iterative procedure trough FDD. From an observation of RM-spectra of many benchmark functions, in [6] it is shown that for most of the functions, their RM-spectral table contains a large number of zero-rows, which means that all elements in that row have a logic value zero. In [6], this property is used to speed up the synthesis procedure in [5] briefly presented in the previous section by memorizing only the non-zero rows of the RM-spectra. However, in the RM-spectrum of a reversible function there are a lot of non-zero coefficients with the same values, which is documented by our experiments for the benchmark functions shown in Table 3. In this table, n_1, n_2 and n_3 are the number of values in the truth vector of the considered function, the number of non-zero coefficients in the the RM-spectrum and the number of different non-zero coefficients, respectively. In columns d_1 and d_2 the relative differences: $d_1 = \frac{(n_1-n_3)}{n_1}$ and $d_2 = \frac{(n_2-n_3)}{n_2}$ are shown.

We examined the number of RM coefficients in the set of all reversible function with $n = 4$ variables. Fig. 4 shows the number of reversible functions with a particular number of non-zero RM coefficients and the number of functions with a particular number of different non-zero coefficients. how many reversible functions with $n = 4$ variables have some number of coefficients. In the case of non-zero coefficients that dependence appears to be linear, while in the case of different non-zero coefficients, after several number, the number of functions decreases.

Note that the number of terminal nodes in FDD representing the RM-spectrum is equal to the number of different non-zero coefficients plus 1 $n_3 + 1$.

It means that for the representation of the RM-spectra of reversible functions will be convenient to use decision diagrams (FDDs). In decision diagrams equal subtrees

are represented by one instance. Information about positions of the RM-coefficients is also saved in the decision diagram. We will use this information in our modified synthesis procedure.

Also, in this new procedure the following notation will be used. $(r_n, \ldots, r_1)$, $r_i \in 0, 1$, will denote values in the terminal nodes of the FDD representing the RM-spectrum of the reversible function $f(x_n, x_{n-1}, \ldots, r_1)$ and $(p_n, p_{n-1}, \ldots, p_1), p_i \in 0, 1$ denotes a path in the FDD.

Note that in the FDD, edges between levels are denoted by 1 and x_i. The path in the FDD is defined in the following definition.

Definition 4. *A path in a FDD, consists of edges from the root node to a terminal node and it is denoted by* $(p_n, p_{n-1}, \ldots, p_1), p_i \in 0, 1$ *where* $p_i = 0$ *if the edge from the level* $i + 1$ *to the level* i *has the label* 1 *and* $p_i = 1$ *if the edge is labeled by* x_i.

Table 3.

Number of non-zero nodes and the number of different nodes.

Fun	Bin/out	n_1	n_2	n_3	d_1 Perc	d_2
3-17-6	3	8	7	6	25.0	14.3
4-46-7	4	16	15	8	50.0	46.7
Ckt2-cycle-73	8	256	229	144	43.8	37.1
Ckt1-cycle-72	9	512	457	290	43.4	36.5
Ckt3-cycle-75	10	1024	874	403	60.6	53.9
Ckt5-cycle-76	9	512	246	111	78.3	54.9
Graycode6-11	6	64	6	6	90.6	0.0
Cycle10-2-61	12	4096	37	12	99.7	67.6
Ham3-28	3	8	4	3	62.5	25.0
Ham7-29	7	128	11	11	91.4	0.0
Hwb5-13	5	32	30	20	37.5	33.3
Hwb6-14	6	64	60	21	67.2	65.0
Hwb7-15	7	128	126	63	50.8	50.0
Hwb8-64	8	256	244	63	75.4	74.2
Hwb9-65	9	512	510	228	55.5	55.3
Mod5adder-66	6	64	46	10	84.4	78.3
Mod5d1-16	5	32	9	5	84.4	44.4
Mod5d2-17	5	32	10	6	81.3	40.0
Mod5mils-18	5	32	10	5	84.4	50.0
Plus127mod8192-78	13	8192	769	14	99.8	98.2
Plus63mod4096-79	12	4096	385	13	99.7	96.6
Plus63mod8192-80	13	8192	449	14	99.8	96.9

New procedure trough FDD. In this section, we define the procedure for synthesis of the Toffoli networks by exploiting transformations of the Reed-Muller spectrum over Functional decision diagrams. The procedure consists of the following steps.

Step A: *Transformation of the terminal node in the path* $(p_n, p_{n-1}, \ldots, p_1) =$

$= (0, 0, \ldots, 0)$: If the terminal node in the path $(0, 0, \ldots, 0)$ has the value $(r_n, r_{n-1}, \ldots, r_1) \neq (0, 0, \ldots, 0)$ we apply NOT gates $Tof(x_j)$ for every $r_j = 1$. To perform the corresponding transformation on the FDD representing the RM-spectrum of the reversible function it is necessary to complement r_j, (to calculate $r_j = r_j \oplus 1$) in the terminal node in the path $(0, 0, \ldots, 0)$.

Step B: *Transformation of the terminal nodes in the paths* $(p_n, \ldots, p_j, \ldots, p_1) = (0, \ldots, 1, \ldots, 0)$ *for* $j = 1, 2, \ldots, n$. (Each of these terminal nodes corresponds to some variable row in tabular representation).

The value $r_n, \ldots, r_i, \ldots, r_1)$ in the terminal node in the path $(p_n, \ldots, p_j, \ldots, p_1)$, has to be transformed into the form $(0, \ldots 1, \ldots, 0)$ with 1 at the position j. This is done through the following two procedures. We first check if $r_j = 1$. If it is not, we make it equal one by assigning a gate $TOF(x_s; x_j)$ such that $s = max\{k | r_k = 1, k \in \{1, 2, \ldots, n\}$ and $s > j$. After this step in the terminal node in the path $(p_n, \ldots p_j, \ldots, p_1) = (0, \ldots, 1, \ldots, 0)$ we have the value $(r_n, \ldots, r_{j+1}, 1, r_{j-1}, \ldots, r_1)$. We next use gates $TOF(x_k, x_j)$ for every $r_k = 1, k \in \{1, 2, \ldots, n\}$ and $k \neq j$. For each applied Toffoli gate $TOF(x_k, x_j)$ in this step, to perform this transformation on the FDD of the reversible function, it is necessary to transform the values of r_j in all terminal nodes by applying $r_j = r_j \oplus r_k$.

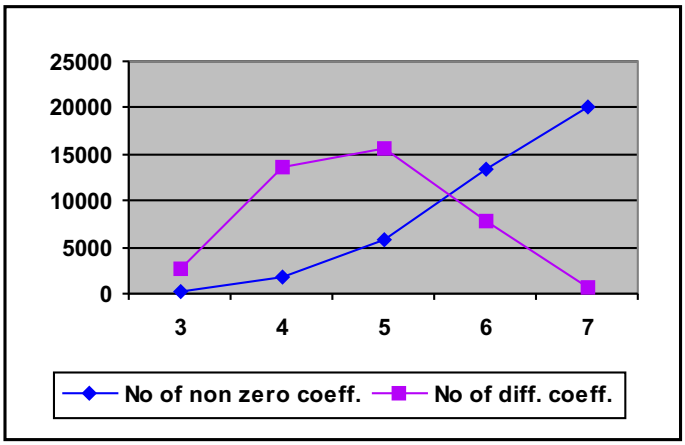

Figure 4. The number of functions with particular number of RM coefficients, for $n = 4$

Step C: *Transformation of the terminal nodes on all other paths* $(p_n, p_{n-1}, \ldots, p_1)$: Assume that, in general, we have the values $(r_n, r_{n-1}, \ldots, r_1), r_i \in \{0, 1\}$, for $i = 1, 2, \ldots, n$. They have to be transformed into $(0, 0, \ldots, 0)$. We first find $s = max\{k | r_k = 1, k \in \{1, \ldots, n\}\}$ and such that in the path $(p_n, \ldots p_j, \ldots, p_1)$ it is $p_s = 0$. We first apply gates $TOF(x_s; x_j)$ for every $r_j = 1, j \in \{1, 2, \ldots, n\}$. This transforms the value in the terminal node in the considered path into $(0, \ldots 1, \ldots, 0)$ with 1 at the position s. Second, we apply gate $TOF(X_k, x_s)$, where X_s is the product of variables $\{x_{j_1}, x_{j_2}, \ldots x_{j_k}\}$ which corresponds to $\{p_{j_1}, p_{j_2}, \ldots p_{j_k}\}$ in the considered path $(p_n, p_{n-1}, \ldots p_j, \ldots, p_1)$ with the values equal one. Finally, it will be necessary to undo $TOF(x_s; x_j)$ if such gates changed the value in the terminal node in some path already transformed in the correct value. Such "undo" operations do not change the correct form of the pattern we are working with. Such operations will transform the value in the terminal node of the path we are working with into

the desired $(0, 0, \ldots, 0)$.

To perform transformation which corresponds to application of the $TOF(X_k, x_s)$ it will be necessary to apply $r_s = r_s \oplus 1$ at all the terminal nodes in the paths $(p_n, p_{n-1}, \ldots, p_1) = (p_n, p_{n-1}, \ldots, p_1)_{j_1} \vee (p_n, p_{n-1}, \ldots, p_1)_{j_2} \vee \cdots \vee (p_n, p_{n-1}, \ldots, p_1)_{j_k}$ where $\vee$ denotes componentwise logic OR operation and $(p_n, p_{n-1}, \ldots, p_1)_{j_l}$ denotes the path to the terminal node with the value $r_{j_1} = 1$.

The procedure proposed above will be illustrated by the example from Table 1.

Example from Table 1 trough the above defined procedure. Initialization: We will start from the MTDD representation of the considered reversible function $c'b'a'$ in Example 1. This MTDD is shown in Fig. 2 and will be transformed into FDD in Fig. 2 representing the RM-spectrum of the considered function.

Step 1. Applying the step A of the proposed procedure we will transform the value in the terminal node in the path $(0, 0, 0)$ from $(1, 1, 1)$ into $(0, 0, 0)$. For this it will be necessary to apply $TOF(a)$,$TOF(b)$ and $TOF(c)$ and to perform the following calculations $a = a \oplus 1$, $b = b \oplus 1$, $c = c \oplus 1$, in the terminal node in the path $(0, 0, 0)$. The transformed FDD is shown in Fig. 5(a).

Step 2. By applying the step B of the procedure, the value in the terminal node on the path $(0, 0, 1)$ will be transformed from $(1, 0, 1)$ into $(0, 0, 1)$. That will be performed by applying $TOF(a; c)$ and transforming the values in all terminal nodes by using $c = c \oplus a$. The resulting FDD is shown in Fig. 5(b).

Step 3. By applying the step B of the procedure, the value in the terminal node in the path $(0, 1, 0)$ will be transformed from $(1, 0, 0)$ into $(0, 1, 0)$. First by applying $TOF(c; b)$ and transforming the values in all terminal nodes by using $b = b \oplus c$ this value will be transformed into $(1, 1, 0)$, and then by applying $(TOF(b; c)$ and $c = c \oplus b$ we will get the value $(0, 1, 0)$ in the terminal node in the path $(0, 1, 0)$. The resulting FDD is shown in Fig. 5(c).

Step 4. By applying the step C of the proposed procedure, the value in the terminal node $(0, 1, 1)$ will be transformed from $(1, 0, 1)$ into $(0, 0, 0)$. First will be applied $TOF(c; a)$ and the corresponding transformation $a = a \oplus c$ which transforms the value $(1, 0, 1)$ into $(1, 0, 0)$, see Fig. 5(d). Then, by applying $TOF(ab; c)$ and the transformation $c = c \oplus 1$ at the terminal node in the paths $(0, 1, 1) = (0, 0, 1)_{a=1} \vee (0, 1, 0)_{b=1}$ and $(1, 0, 1) = (0, 0, 1)_{a=1} \vee (1, 0, 1)_{b=1}$ this value will be transformed into $(0, 0, 0)$, see Fig. 5(e).

Step 5. The value at the terminal node in the path $(1, 0, 0)$ is $(1, 0, 0)$ and, therefore, it is correct. For that node further transformation unnecessary.

Step 6. By applying the step C of the procedure the value of the terminal node in the path $(1, 0, 1)$ will be transformed from $(0, 1, 0)$ into $(0, 0, 0)$. That will be performed by applying $TOF(ac; b)$ and the transformation $b = b \oplus 1$ at the terminal node $(1, 0, 1) = (1, 0, 0)_{c=1} \vee (0, 0, 1)_{a=1}$. After this transformation the FDD will provide the targeted structure. This FDD is shown in Fig. 5(f). As result of this procedure the Toffoli network shown in the Fig. 3 is generated.

Conclusion In the paper, the procedure for iterative synthesis of Toffolli networks proposed in [5] is redefined to be implemented trough decision diagrams representing the Reed-Muller spectrum of a reversible function. In this case, compared with the previous procedure, smaller memory is sufficient to store values of the RM-spectrum of the reversible function processed. We expect that this saving will permit realization of reversible functions with a large number of variables. Our next step

will be to confirm this conjecture by a series of experiments over some benchmark functions.

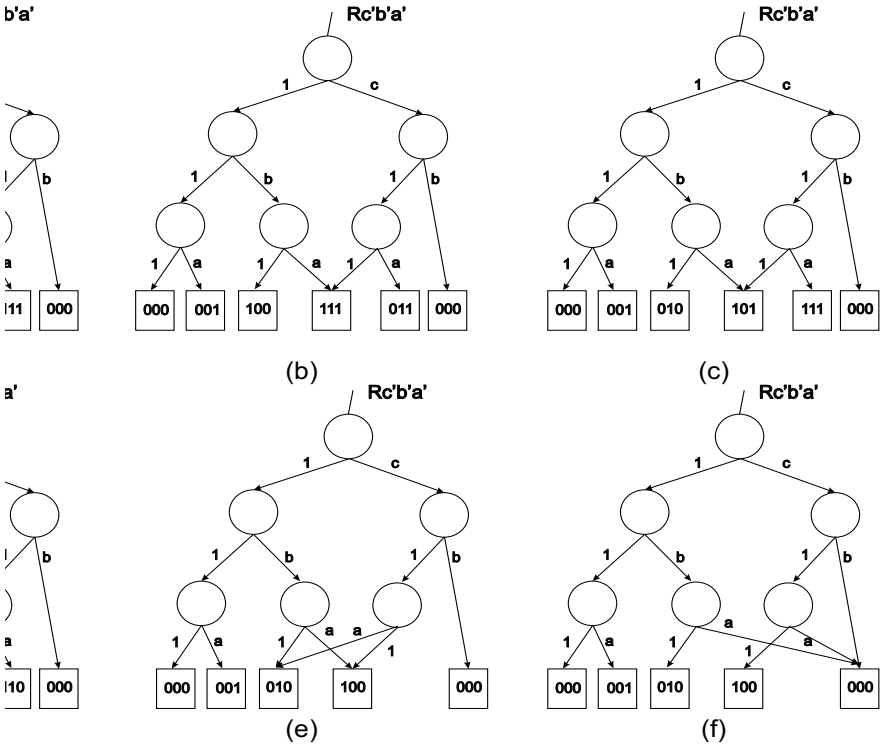

Figure 5. Synthesis procedure for the function f in Table 1.

1. *Aizenberg, N.N., Trofimljuk, O.T.* Conjunctive transforms for discrete signals and their applications of tests and the detection of monotone functions. // Kibernetika – No. 5 – K., 1981.
2. *Toffoli, T.* Reversible computing. Tech. Rep., MIT, Cambridge, MA, 1980.
3. *Fredkin, E., Toffoli, T.* Conservative logic. Int. J. Theor. Phys., vol. 21, no. 3/4, pp. 219–253, 1982.
4. *Sasao, T., Fujita, M., (eds.)* Representation of Discrete Functions, Kluwer Academic Publishers, 1996.
5. *Maslov, D., Miller, D.W., Duek, G.V.* Techniques for the synthesis of reversible Toffoli networks. ACM Transaction on Design Automation of Electronic Systems (TODAES), Vol.12, No. 4, September 2007, pp. 42:21–42:28.
6. *Zhong, J., Muzio, J.C.* Improved Implementation of a Reed-Muller Spectra Based Reversible Synthesis Algorithm. IEEE Pacific Rim Conference on Communication, Computers and Signal Processing, PacRim 2007., August 2007.
7. *Gupta, P., Agrawal, A., Jha, N.K.* An algorithm for synthesis of reversible logic circuits. IEEE Trans. on Computer-aided Design of Integrated Circuits and Systems, Vol. 25, No. 11, November 2006, P. 2317-2329.
8. *Wille, R., Grosse, D., Teuber, L., Duek, G.W., Drechsler, R.* RevLib: Online Resource for Reversible Functions and Reversible Circuits. Proc. 38th International Symposium on Multiple Valued Logic. Dallas, Texas, USA, May 22-24, 2008, P. 220-225.
9. *Stanković, R.S., Stanković, M., Janković, D.* Spectral Transforms in Switching Theory: Definitions and Calculations, Nauka, Beograd, 1998.
10. *Donald, J., Nirajk, J.* Reversible logic synthesis with Fredkin and Peres gates. ACM Journal of Emerging Technologies in Computing Systems, Vol. 4, No. 1, Article 2, March 2008.

Recived 10.10.2008