FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 20, no. 3, December 2007, 507-539

Representations, Operations, and Applications of Switching Circuits in the Reversible and Quantum Spaces

Anas N. Al-Rabadi

Abstract: This paper reviews recent developments in the use of reversible synthesis techniques in the reversible and quantum logic circuit design of binary and multiple-valued switching functions. In particular, this paper reviews the specific developments and various uses of spectral transforms and functional expansions, group theory, and cellular automata in the reversible logic synthesis of digital functions.

Keywords: Reversible logic, quantum circuits, Galois logic, spectral techniques, group theory, cellular automata.

1 Introduction

A (k,k) reversible circuit is a circuit that has the same number of inputs (k) and outputs (k) and is a one-to-one mapping between the vectors of inputs and the vectors of outputs, thus the vector of input states can be always uniquely reconstructed from the vector of output states [1–13]. Thus, a (k,k) reversible map is a bijective function which is both injective ("one-to-one") and surjective ("onto"). The auxiliary outputs and inputs that are needed only for the purpose of reversibility are called "garbage" outputs and "garbage" inputs respectively. These are auxiliary outputs and inputs from which a reversible map is made. A (k,k) conservative circuit has the same number of inputs (k) and outputs (k) and has the same number of values in inputs and outputs (e.g., the same number of ones in inputs and outputs for binary, the same number of ones and twos in inputs and outputs for ternary, etc) [1, 5, 10].

Manuscript received August 25, 2007.

The author is an Associate Professor in the Computer Engineering Department at the University of Jordan, Jordan. The author is also associated with the Office of Graduate Studies and Research (OGSR) at Portland State University, U.S.A. (e-mail: alrabadi@yahoo.com).

Motivations for pursuing the possibility of implementing circuits using reversible logic (RL) would include items such as: (1) power: the fact that, theoretically, the internal computations in RL systems consume no power. It is shown in [7] that a necessary (but not sufficient) condition for not dissipating power in any physical circuit is that all system circuits must be built using fully reversible logical components. For this reason, different technologies have been studied to implement reversible logic in hardware, such as adiabatic CMOS [14], optical [11], and quantum [10]. Fully reversible digital systems will greatly reduce the power consumption (theoretically eliminate) through three conditions: (i) logical reversibility: the vector of input states can always be uniquely reconstructed from the vector of output states, (ii) physical reversibility: the physical switch operates backwards as well as forwards, and (iii) the use of "ideal-like" switches: switches that have no parasitic resistances; (2) size: since the newly emerging quantum computing technology must be reversible [10], the current trends related to more dense hardware implementations are heading towards 1 Angstrom, at which quantum mechanical effects have to be accounted for; and (3) speed: if the properties of quantum superposition and entanglement can be usefully employed in the reversible circuit design context, significant computational speed enhancements can be expected [1, 10]. One of the advantages of using such RL circuits is their potential utilization in adiabatic low-power VLSI circuit designs [14] for several applications in analogy to the role of classical circuits in non-adiabatic VLSI design [15, 16].

Several spectral-based techniques for the reversible logic synthesis are presented in Section 2. Group-based method for RL representation is presented in Section 3. Elementary cellular automata (ECA) - based methods for RL synthesis is shown in Section 4. Conclusions and future work are presented in Section 5. Although the review presented in this paper is conducted for the second and third radices of Galois logic, the extensions to higher radices and other logics use similar methods as well.

2 Spectral-Based Synthesis

Spectral techniques have been extensively used in the classical domain in the design of logic functions [1, 17–19]. This section presents several methods that are used in the spectral-based design of two-valued and multiple-valued reversible and quantum circuits.

2.1 Reversible expansions and reversible spectral transforms

Galois field (GF) possesses desired properties that are useful in circuit applications like testing [20]. Figure 1 shows GF addition and multiplication for ternary radix.

	0	**1**	**2**
0	0	1	2
1	1	2	0
2	2	0	1

(a)

	0	**1**	**2**
0	0	0	0
1	0	1	2
2	0	2	1

(b)

Fig. 1. (a) GF3(+), and (b) GF3(∗).

Let us define the 1-Reduced Post Literal (1-RPL) [1]:

$$ {}^{i}x = 1 \quad \text{iff} \quad x = i \quad \text{else} \quad {}^{i}x = 0 \tag{1}$$

For example, ${}^{0}x$, ${}^{1}x$, ${}^{2}x$ are the zero, first, and second polarities of the 1-RPL, respectively. The fundamental Shannon expansion over GF(3) for a ternary function with a single variable is [1, 19]:

$$ f(x) = {}^{0}x f_0 + {}^{1}x f_1 + {}^{2}x f_2 \tag{2}$$

where $f_0 = f(x = 0)$, $f_1 = f(x = 1)$, and $f_2 = f(x = 2)$. Using the addition and multiplication over GF(3), and the axioms of GF(3), it can be shown that the 1-RPLs defined in Equation (1), are related to the shifts of variables over GF(3) in terms of powers as follows [1]:

$$ {}^{0}x = 2(x)^2 + 1 \tag{3}$$
$$ {}^{0}x = 2(x')^2 + 2(x') \tag{4}$$
$$ {}^{0}x = 2(x'')^2 + x'' \tag{5}$$
$$ {}^{1}x = 2(x)^2 + 2(x) \tag{6}$$
$$ {}^{1}x = 2(x')^2 + x' \tag{7}$$
$$ {}^{1}x = 2(x'')^2 + 1 \tag{8}$$
$$ {}^{2}x = 2(x)^2 + x \tag{9}$$
$$ {}^{2}x = 2(x')^2 + 1 \tag{10}$$
$$ {}^{2}x = 2(x'')^2 + 2(x'') \tag{11}$$

After the substitution of Equations (3) through (11) in Equation (2), and after the minimization of the terms according to the axioms of Galois field, one obtains the following Equations:

$$ f = 1 \cdot f_0 + x \cdot (2f_1 + f_2) + 2(x)^2 (f_0 + f_1 + f_2) \tag{12}$$
$$ f = 1 \cdot f_2 + x' \cdot (2f_0 + f_1) + 2(x')^2 (f_0 + f_1 + f_2) \tag{13}$$
$$ f = 1 \cdot f_1 + x'' \cdot (2f_2 + f_0) + 2(x'')^2 (f_0 + f_1 + f_2) \tag{14}$$

Equations (2) and (12) - (14) are the fundamental ternary Shannon and Davio expansions for a single variable, respectively. These Equations can be re-written in the following matrix-based forms [1, 19], respectively:

$$f = \begin{bmatrix} {}^0x & {}^1x & {}^2x \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} \tag{15}$$

$$f = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} \tag{16}$$

$$f = \begin{bmatrix} 1 & x' & (x')^2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 2 & 1 & 0 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} \tag{17}$$

$$f = \begin{bmatrix} 1 & x'' & (x'')^2 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} \tag{18}$$

The extension of the multiple-valued Shannon and Davio expansions for two or more variables is obtained using the Kronecker product of the vector of the basis functions and the Kronecker product of the transform matrix [19] (cf. Equation (33)).

Definition 1. The matrix that is constructed from the permutations of many basis functions of the same type for the corresponding spectral transform is called Generalized Basis Function Matrix (GBFM) [1].

Definition 2. From the total space of the all possible Generalized Basis Function Matrices, the matrices that produce reversible expansions are called Reversible Generalized Basis Function Matrices (RGBFM) [1].

A necessary and sufficient condition to generate the reversible multi-valued Shannon expansions is that the order of the permuted basis functions in the GBFM should satisfy the following constraint: *in any given row or column the elements in that row or column are different than the elements in the adjacent positions of the other rows or columns* [1].

Example 1. The following is the ternary Shannon transform over GF(3).

$$f = \begin{bmatrix} {}^0c & {}^1c & {}^2c \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix}$$

The following is one possible GBFM:

$$\begin{bmatrix} {}^0c & {}^1c & {}^2c \\ {}^0c & {}^2c & {}^1c \\ {}^2c & {}^1c & {}^0c \end{bmatrix}$$

Yet the upper GBFM is not reversible; it does not produce a reversible expansion. One possible Reversible Shannon Generalized Basis Function Matrix (RSGBFM) that leads to a reversible expansion is the following GBFM:

$$\begin{bmatrix} {}^0c & {}^1c & {}^2c \\ {}^1c & {}^2c & {}^0c \\ {}^2c & {}^0c & {}^1c \end{bmatrix}$$

The reversibility constraint, mentioned above, can be illustrated by means of tables as follows: since the Shannon matrix is *orthogonal*, then the following (3,3) ternary Shannon expansion in Equation (19a):

$$\vec{f} = \begin{bmatrix} {}^2c & {}^0c & {}^1c \\ {}^1c & {}^2c & {}^0c \\ {}^0c & {}^1c & {}^2c \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_{r0} \\ f_{r1} \\ f_{r2} \end{bmatrix} \tag{19a}$$

is reversible given that input c is produced in the output for which a modified form of Eq. (19a) would be as follows:

$$\vec{f} = \begin{bmatrix} {}^2c & {}^0c & {}^1c & 0 \\ {}^1c & {}^2c & {}^0c & 0 \\ {}^0c & {}^1c & {}^2c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ c \end{bmatrix} = \begin{bmatrix} f_{r0} \\ f_{r1} \\ f_{r2} \\ c \end{bmatrix} \tag{19b}$$

(As an example we are using in this case Shannon set of basis functions $\{{}^0c, {}^1c, {}^2c\}$ that appear in the third row, but Shannon set of basis functions $\{{}^0c, {}^1c, {}^2c\}$ can appear in any of the rows of the corresponding RGBFM.) The reversible Shannon expansion in Equation (19b) is reversible as shown in Table 1.

Table 1. Proof of the reversibility of the 4-input 4-output (i.e., (4, 4)) Equation (19b).

Inputs (I)				Outputs (O)			
f_0	f_1	f_2	c	f_{r0}	f_{r1}	f_{r2}	c
f_0	f_1	f_2	0	f_1	f_2	f_0	0
f_0	f_1	f_2	1	f_2	f_0	f_1	1
f_0	f_1	f_2	2	f_0	f_1	f_2	2

Table 1. (continued.)

I	O	I	O	I	O	I	O	I	O	I	O	I	O	I	O	I	O
0000	0000	0100	1000	0200	2000	1000	0010	1100	1010	1200	2010	2000	0020	2100	1020	2200	2020
0001	0001	0101	0011	0201	0021	1001	0101	1101	0111	1201	0121	2001	0201	2101	0211	2201	0221
0002	0002	0102	0102	0202	0202	1002	1002	1102	1102	1202	1202	2002	2002	2102	2102	2202	2202
0010	0100	0110	1100	0210	2100	1010	0110	1110	1110	1210	2110	2010	0120	2110	1120	2210	2120
0011	1001	0111	1011	0211	1021	1011	1101	1111	1111	1211	1121	2011	1201	2111	1211	2211	1221
0012	0012	0112	0112	0212	0212	1012	1012	1112	1112	1212	1212	2012	2012	2112	2112	2212	2212
0020	0200	0120	1200	0220	2200	1020	0210	1120	1210	1220	2210	2020	0220	2120	1220	2220	2220
0021	2001	0121	2011	0221	2021	1021	2101	1121	2111	1221	2121	2021	2201	2121	2211	2221	2221
0022	0022	0122	0122	0222	0222	1022	1022	1122	1122	1222	1222	2022	2022	2122	2122	2222	2222

Example 2. Lets produce the reversible ternary Shannon expansions for ternary Galois logic using *all* possible permutations of the RGBFM of the ternary Shannon transform matrix [I] [1] given that input c is produced in the output.

$$\vec{f} = \begin{bmatrix} {}^0c & {}^1c & {}^2c \\ {}^1c & {}^2c & {}^0c \\ {}^2c & {}^0c & {}^1c \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_{r0} \\ f_{r1} \\ f_{r2} \end{bmatrix} \tag{20}$$

$$\vec{f} = \begin{bmatrix} {}^0c & {}^1c & {}^2c \\ {}^2c & {}^0c & {}^1c \\ {}^1c & {}^2c & {}^0c \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_{r0} \\ f_{r1} \\ f_{r2} \end{bmatrix} \tag{21}$$

$$\vec{f} = \begin{bmatrix} {}^1c & {}^2c & {}^0c \\ {}^0c & {}^1c & {}^2c \\ {}^2c & {}^0c & {}^1c \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_{r0} \\ f_{r1} \\ f_{r2} \end{bmatrix} \tag{22}$$

$$\vec{f} = \begin{bmatrix} {}^2c & {}^0c & {}^1c \\ {}^0c & {}^1c & {}^2c \\ {}^1c & {}^2c & {}^0c \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_{r0} \\ f_{r1} \\ f_{r2} \end{bmatrix} \tag{23}$$

$$\vec{f} = \begin{bmatrix} {}^1c & {}^2c & {}^0c \\ {}^2c & {}^0c & {}^1c \\ {}^0c & {}^1c & {}^2c \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_{r0} \\ f_{r1} \\ f_{r2} \end{bmatrix} \tag{24}$$

$$\vec{f} = \begin{bmatrix} {}^2c & {}^0c & {}^1c \\ {}^1c & {}^2c & {}^0c \\ {}^0c & {}^1c & {}^2c \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_{r0} \\ f_{r1} \\ f_{r2} \end{bmatrix} \tag{25}$$

Reversible circuits for the synthesis of Equations (20) - (25) were shown [1] using (3,3) multiplexers (cf. Figure 2). Using Equations (20) - (25), reversible Davio expansions have been also derived [1]. This is done as follows: Let us produce reversible Davio expansion, for the ternary case over GF(3), for one possible multi-valued reversible Shannon expansion:

$$\vec{f} = \begin{bmatrix} {}^0c & {}^1c & {}^2c \\ {}^1c & {}^2c & {}^0c \\ {}^2c & {}^0c & {}^1c \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_{r0} \\ f_{r1} \\ f_{r2} \end{bmatrix} \tag{26}$$

Using the same method that was utilized for deriving Equation (16) from Equation (15) using Equations (1) and (3)-(11), there exist for Equation (26) three Davio types for each row of the RSGBFM. The following are the D_0-type expansions for the first row, second row, and third row of the above RSGBFM $\begin{bmatrix} {}^0c & {}^1c & {}^2c \\ {}^1c & {}^2c & {}^0c \\ {}^2c & {}^0c & {}^1c \end{bmatrix}$, respectively (where the subscripts indicate the orderings of the literals ${}^kc$ in the RSGBFM in Equation (26), and D_0 indicates reversible Davio expansion of type D_0) [1]:

$$f_{012,D0} = 1 \cdot f_0 + c \cdot (2f_1 + f_2) + (c)^2(2f_0 + 2f_1 + 2f_2) = \begin{bmatrix} 1 & c & c^2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} \quad (27)$$

$$f_{120,D0} = 1 \cdot f_2 + c \cdot (2f_0 + f_1) + (c)^2(2f_0 + 2f_1 + 2f_2) = \begin{bmatrix} 1 & c & c^2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 2 & 1 & 0 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} \quad (28)$$

$$f_{201,D0} = 1 \cdot f_1 + c \cdot (2f_2 + f_0) + (c)^2(2f_0 + 2f_1 + 2f_2) = \begin{bmatrix} 1 & c & c^2 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} \quad (29)$$

The above Davio expansions are reversible. Reversibility can be shown by utilizing the identities in Equations (3), (6), and (9) to simplify (re-write) Equations (27) - (29) as follows:

$$\begin{aligned} f_{012,D0} &= 1 \cdot f_0 + c \cdot (2f_1 + f_2) + (c)^2(2f_0 + 2f_1 + 2f_2) && (30) \\ &= (1 + 2c^2) f_0 + (2c + 2c^2) f_1 + (c + 2c^2) f_2 = {}^0c f_0 + {}^1c f_1 + {}^2c f_2 \\ f_{120,D0} &= 1 \cdot f_2 + c \cdot (2f_0 + f_1) + (c)^2(2f_0 + 2f_1 + 2f_2) && (31) \\ &= (2c + 2c^2) f_0 + (c + 2c^2) f_1 + (1 + 2c^2) f_2 = {}^1c f_0 + {}^2c f_1 + {}^0c f_2 \\ f_{201,D0} &= 1 \cdot f_1 + c \cdot (2f_2 + f_0) + (c)^2(2f_0 + 2f_1 + 2f_2) && (32) \\ &= (c + 2c^2) f_0 + (1 + 2c^2) f_1 + (2c + 2c^2) f_2 = {}^2c f_0 + {}^0c f_1 + {}^1c f_2 \end{aligned}$$

By using results from Equation (20), this shows the reversibility in Equations (30) - (32).

In addition to reversibility, the reversible Shannon and Davio expansions shown in this Section are also conservative [1]. The importance of conservativeness in such expansions stems from the fact that the property of conservativeness reflects the physical law of energy preservation: *no energy can be created or destroyed, but can be transformed from one form to another*. Thus reversible conservative expansions will incorporate the fundamental law of energy preservation into the logic design of systems [5].

2.2 Reversible primitives (gates)

The reversible formulation in Section 2.1 has been used in deriving families of reversible gates [1]. For example, Figures 2a and 2b represent one possible logic circuit realizations for Equations (21) and (24) respectively, where all inputs $\{c, f_0, f_1, f_2\}$ and outputs $\{c, f_{r0}, f_{r1}, f_{r2}\}$ can have any of the ternary values $\{0,1,2\}$. The process of forward permutation of cofactors [1] is illustrated at the outputs of the ternary reversible Shannon primitives in Figures 2a and 2b, respectively.

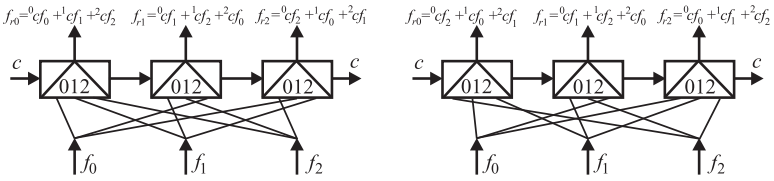


Fig. 2. Logic circuit realization of the reversible expansion in Equations (21) and (24), respectively.

2.3 Reversible decision trees (RDTs)

The classical logic Shannon and Davio expansions and spectral transforms are produced for higher dimensions using the Kronecker (tensor) product [1, 19]. The Kronecker product is defined for a transform matrix $[M]$ as follows:

$$\begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} \otimes [M] = \begin{bmatrix} a[M] & d[M] & g[M] \\ b[M] & e[M] & h[M] \\ c[M] & f[M] & i[M] \end{bmatrix} \tag{33}$$

Although the Kronecker recursion defined in Equation (33) is used for the recursive generation of the transform matrices in the logic expansions such as in Equations (15) through (18), the same Kronecker recursion can be used for the recursive generation of the RGBFMs for functions of several input variables [21]. This is shown as follows:

$$\vec{f}_{r^n\times 1} = [\otimes[RGBFM]]_{r^n\times r^n}[\otimes[T.M.]]_{r^n\times r^n}\vec{F}_{r^n\times 1} \tag{34}$$

where n is the number of variables, r is the logic radix, RGBFM is any reversible generalized basis function matrix, and T.M. is the corresponding transform matrix. For example, for a two-variable third-radix Galois logic, Equation (34) becomes:

$$\begin{aligned} \vec{f}_{3^2\times 1} = &\quad [\otimes[RGBFM]]_{3^2\times 3^2}[\otimes[T.M.]]_{3^2\times 3^2}\vec{F}_{3^2\times 1} \\ = &\quad \vec{f}_{9\times 1} = [\otimes[RGBFM]]_{9\times 9}[\otimes[T.M.]]_{9x9}\vec{F}_{9\times 1} \end{aligned} \tag{35}$$

The general mathematical form in Equation (34) implies the recursive generation of the corresponding RDTs [21]. Example 3 shows an example for the generation of RDTs for ternary logic functions of two variables for the case of reversible Shannon DTs.

Example 3. For a ternary two-variable truth vector $\vec{F}$, the following shows the use of the operation of Kronecker product for the recursive reversible expansion using the reversible Shannon expansion in Equation (20) [21] given that inputs a and b are produced in the outputs.

$$\vec{f} = \begin{bmatrix} f_{r0} \\ f_{r1} \\ f_{r2} \\ f_{r3} \\ f_{r4} \\ f_{r5} \\ f_{r6} \\ f_{r7} \\ f_{r8} \end{bmatrix} = \left[\begin{bmatrix} {}^0a & {}^1a & {}^2a \\ {}^1a & {}^2a & {}^0a \\ {}^2a & {}^0a & {}^1a \end{bmatrix} \otimes \begin{bmatrix} {}^0b & {}^1b & {}^2b \\ {}^1b & {}^2b & {}^0b \\ {}^2b & {}^0b & {}^1b \end{bmatrix} \right] \left[\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right] \left[\begin{bmatrix} f_{a0} \\ f_{a1} \\ f_{a2} \end{bmatrix} \otimes \begin{bmatrix} f_{b0} \\ f_{b1} \\ f_{b2} \end{bmatrix} \right] \tag{36}$$

$$= \begin{bmatrix} {}^0a \begin{bmatrix} {}^0b & {}^1b & {}^2b \\ {}^1b & {}^2b & {}^0b \\ {}^2b & {}^0b & {}^1b \end{bmatrix} & {}^1a \begin{bmatrix} {}^0b & {}^1b & {}^2b \\ {}^1b & {}^2b & {}^0b \\ {}^2b & {}^0b & {}^1b \end{bmatrix} & {}^2a \begin{bmatrix} {}^0b & {}^1b & {}^2b \\ {}^1b & {}^2b & {}^0b \\ {}^2b & {}^0b & {}^1b \end{bmatrix} \\ {}^1a \begin{bmatrix} {}^0b & {}^1b & {}^2b \\ {}^1b & {}^2b & {}^0b \\ {}^2b & {}^0b & {}^1b \end{bmatrix} & {}^2a \begin{bmatrix} {}^0b & {}^1b & {}^2b \\ {}^1b & {}^2b & {}^0b \\ {}^2b & {}^0b & {}^1b \end{bmatrix} & {}^0a \begin{bmatrix} {}^0b & {}^1b & {}^2b \\ {}^1b & {}^2b & {}^0b \\ {}^2b & {}^0b & {}^1b \end{bmatrix} \\ {}^2a \begin{bmatrix} {}^0b & {}^1b & {}^2b \\ {}^1b & {}^2b & {}^0b \\ {}^2b & {}^0b & {}^1b \end{bmatrix} & {}^0a \begin{bmatrix} {}^0b & {}^1b & {}^2b \\ {}^1b & {}^2b & {}^0b \\ {}^2b & {}^0b & {}^1b \end{bmatrix} & {}^1a \begin{bmatrix} {}^0b & {}^1b & {}^2b \\ {}^1b & {}^2b & {}^0b \\ {}^2b & {}^0b & {}^1b \end{bmatrix} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_{a0b0} \\ f_{a0b1} \\ f_{a0b2} \\ f_{a1b0} \\ f_{a1b1} \\ f_{a1b2} \\ f_{a2b0} \\ f_{a2b1} \\ f_{a2b2} \end{bmatrix}$$

$$= \begin{bmatrix} {}^0a{}^0b & {}^0a{}^1b & {}^0a{}^2b & {}^1a{}^0b & {}^1a{}^1b & {}^1a{}^2b & {}^2a{}^0b & {}^2a{}^1b & {}^2a{}^2b \\ {}^0a{}^1b & {}^0a{}^2b & {}^0a{}^0b & {}^1a{}^1b & {}^1a{}^2b & {}^1a{}^0b & {}^2a{}^1b & {}^2a{}^2b & {}^2a{}^0b \\ {}^0a{}^2b & {}^0a{}^0b & {}^0a{}^1b & {}^1a{}^2b & {}^1a{}^0b & {}^1a{}^1b & {}^2a{}^2b & {}^2a{}^0b & {}^2a{}^1b \\ {}^1a{}^0b & {}^1a{}^1b & {}^1a{}^2b & {}^2a{}^0b & {}^2a{}^1b & {}^2a{}^2b & {}^0a{}^0b & {}^0a{}^1b & {}^0a{}^2b \\ {}^1a{}^1b & {}^1a{}^2b & {}^1a{}^0b & {}^2a{}^1b & {}^2a{}^2b & {}^2a{}^0b & {}^0a{}^1b & {}^0a{}^2b & {}^0a{}^0b \\ {}^1a{}^2b & {}^1a{}^0b & {}^1a{}^1b & {}^2a{}^2b & {}^2a{}^0b & {}^2a{}^1b & {}^0a{}^2b & {}^0a{}^0b & {}^0a{}^1b \\ {}^2a{}^0b & {}^2a{}^1b & {}^2a{}^2b & {}^0a{}^0b & {}^0a{}^1b & {}^0a{}^2b & {}^1a{}^0b & {}^1a{}^1b & {}^1a{}^2b \\ {}^2a{}^1b & {}^2a{}^2b & {}^2a{}^0b & {}^0a{}^1b & {}^0a{}^2b & {}^0a{}^0b & {}^1a{}^1b & {}^1a{}^2b & {}^1a{}^0b \\ {}^2a{}^2b & {}^2a{}^0b & {}^2a{}^1b & {}^0a{}^2b & {}^0a{}^0b & {}^0a{}^1b & {}^1a{}^2b & {}^1a{}^0b & {}^1a{}^1b \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_{a0b0} \\ f_{a0b1} \\ f_{a0b2} \\ f_{a1b0} \\ f_{a1b1} \\ f_{a1b2} \\ f_{a2b0} \\ f_{a2b1} \\ f_{a2b2} \end{bmatrix}$$

Figure 3 shows the RDT synthesis for Equation (36). In Figure 3, function inputs are values in the leaves, input control variables a and b propagate through each level to the outputs, and basis functions are located on the internal interconnects (edges) where — means 1-RPL for variable of value 0 (i.e., ${}^0a$ or ${}^0b$), - - - means 1-RPL for variable of value 1 (i.e., ${}^1a$ or ${}^1b$), and $-\cdot-$ means 1-RPL for variable of value 2 (i.e., ${}^2a$ or ${}^2b$).

If one multiplies each leaf value, going left-to-right, with all possible bottom-up paths (i.e., from the leaves to the roots) and add them over Galois field (using Figure 1) then one obtains the outputs $\{f_{r0}, \ldots, f_{r8}\}$, respectively. The (11,11) reversible Shannon decision tree (RSDT) in Figure 3 is a multi-input multi-output specific type of DTs [21].

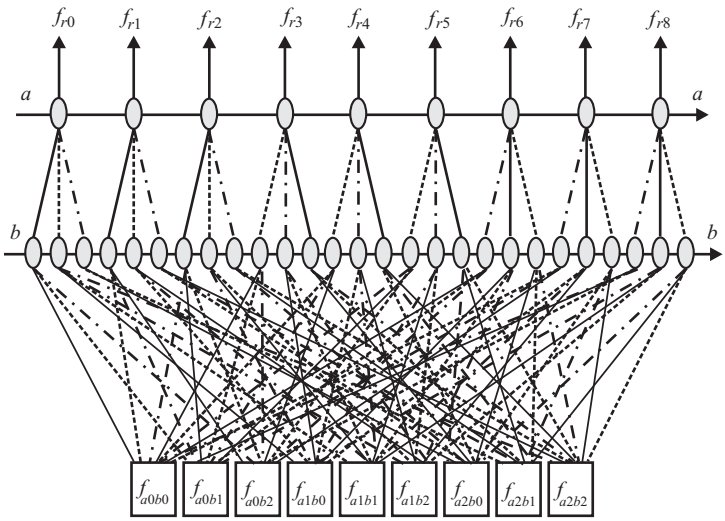


Fig. 3. Multiplexer-based (11,11) Reversible Shannon Decision Tree (RSDT) for Equation (36).

2.4 Reversible lattice circuits

Lattice circuits have been introduced as an important application of spectral transforms to reversible regular circuits [1, 22]. The algorithm for the synthesis of reversible lattice circuits depends on the hierarchical application of the process of permutation of cofactors that has been presented in Section 2.2 (e.g., see the output functions in Figures 2a and 2b, respectively). A general procedure for the construction of a reversible Shannon lattice circuit over n^{th} radix logic is as follows (cf. Example 4) [1, 22]:

Synthesis Stage:

1. Utilizing a reversible Shannon primitive (from Section 2.1), assign the multi-valued map of the function that is needed to be realized in the reversible lattice circuit for one output of the reversible Shannon primitive in the first level, and assign don't care maps for the rest of the primitive outputs at the first level. Also assign don't care maps to the "garbage" outputs of the primitives in each level of the reversible lattice circuit. These "garbage" outputs are needed only for the purpose of reversibility. (The process of assigning don't cares to multi-valued maps stems from the fact that one does not know a priori what will be the values of the leaves of the corresponding reversible lattice circuit.)
2. Following the output-to-input paths of the reversible Shannon primitive in the first level of the reversible lattice circuit, going from outputs-to-inputs,

and using the reverse of the method of permutation of cofactors from Section 2.2 (e.g., constructing inputs from outputs in Figures 2a and 2b, for instance), construct new maps at the input of the reversible Shannon primitive by permuting the output cofactors (in the output maps) that correspond to the expansion variable in the first level. This process of permuting the output cofactors will result in new maps at the inputs of the reversible Shannon primitive at the first level. Thus, the contents of the input maps will result from the permutation of the values of the cofactors in maps at the output of the same reversible Shannon primitives at the first level.

3. Going from top-to-bottom and left-to-right in the reversible lattice circuit, repeat Step 2 for each expansion variable in each level (i.e., for each reversible Shannon primitive in every level) until one reaches multi-valued maps at the bottom of the reversible lattice circuit with each map having *only* a constant value from the set $\{0, 1, 2\}$.

Analysis Stage: An opposite process to the synthesis.

4. Following the input-to-output paths of the reversible Shannon primitives at the last level of the reversible lattice circuit, going from inputs-to-outputs, and using the forward method of permutation of cofactors from Section 2.2, construct new maps at the output of the reversible Shannon primitives by permuting the input cofactors (in the input maps) that correspond to the expansion variable in the last level. This process of permuting the input cofactors will result in multi-valued maps at the outputs of the reversible Shannon primitives at the last level. Thus, the contents of the output maps (at the last level) will result from the permutation of the values of the cofactors in maps at the inputs of the same reversible Shannon primitives.

5. Going from bottom-to-top and right-to-left of the reversible lattice circuit, repeat Step 4 for each expansion variable in each level (i.e., for each reversible Shannon primitive in each level) until one reaches completely specified maps, in all wires through the reversible lattice circuit from bottom-to-top and right-to-left, with no don't cares.

Example 4. Figure 4 illustrates the creation of the reversible ternary lattice circuit for the ternary function (F) using the reversible Shannon gate from Figure 2a. Note that, in Figure 4, the desired output function is denoted as F and the "garbage" outputs (that are necessary only for reversibility) are denoted as G1 - G8.

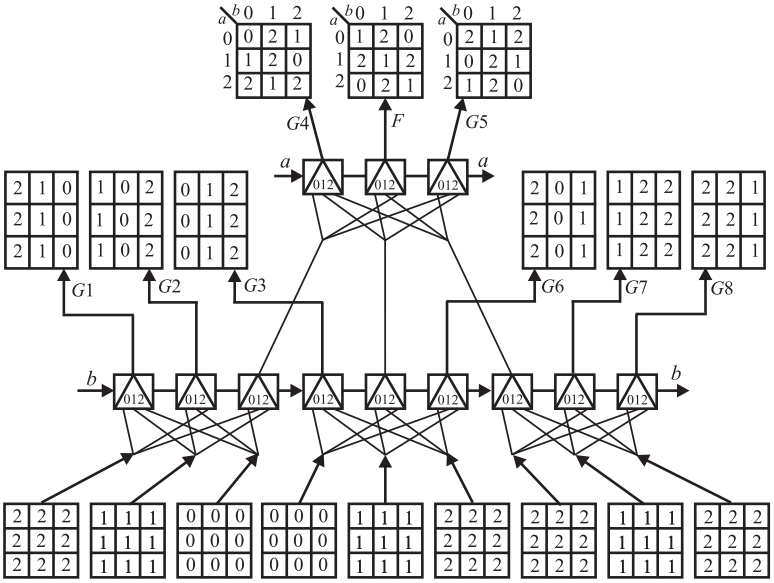


Fig. 4. Bottom-up analysis of the reversible lattice circuit for the synthesis of the ternary function (F) in Example 4.

2.5 Reversible fast transform circuits

Spectral methods have also been used to synthesize reversible fast transform circuits [23–25]. The spectral transform S (i.e., the vector of spectral coefficients) for n variables is defined as follows [1, 19]:

$$\vec{S}^n = [M][\vec{F}] \tag{37}$$

where $[M]$ is the transform matrix, $[\vec{F}]$ is the truth vector of function f, and $\vec{S}^n$ is the vector of spectral coefficients for n variables. For example, performing the same method used for obtaining Equations (27) - (29), one can obtain similar Equations for reversible Davio of types D_1 and D_2. The vectors of spectral coefficients for reversible Davio expansions of types D_0, D_1, and D_2 are obtained as follows: (where the subscripts indicate the orderings of the literals ${}^k c$ in the RSGBFM in Equation (20), D_0 indicates reversible Davio expansion of type D_0, D_1 indicates reversible Davio expansion of type D_1, and D_2 indicates reversible Davio expansion of type D_2) [24]:

$$\vec{S}^1_{012,D_0} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_0 \\ 2f_1 + f_2 \\ 2f_0 + 2f_1 + 2f_2 \end{bmatrix} \tag{38}$$

$$\vec{S}^1_{120,D_0} = \begin{bmatrix} 0 & 0 & 1 \\ 2 & 1 & 0 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_2 \\ 2f_0 + f_1 \\ 2f_0 + 2f_1 + 2f_2 \end{bmatrix} \tag{39}$$

$$\vec{S}^1_{201,D_0} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ 2f_2 + f_0 \\ 2f_0 + 2f_1 + 2f_2 \end{bmatrix} \tag{40}$$

$$\vec{S}^1_{012,D_1} = \begin{bmatrix} 0 & 0 & 1 \\ 2 & 1 & 0 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_2 \\ 2f_0 + f_1 \\ 2f_0 + 2f_1 + 2f_2 \end{bmatrix} \tag{41}$$

$$\vec{S}^1_{120,D_1} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ 2f_2 + f_0 \\ 2f_0 + 2f_1 + 2f_2 \end{bmatrix} \tag{42}$$

$$\vec{S}^1_{201,D_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_0 \\ 2f_1 + f_2 \\ 2f_0 + 2f_1 + 2f_2 \end{bmatrix} \tag{43}$$

$$\vec{S}^1_{012,D_2} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ 2f_2 + f_0 \\ 2f_0 + 2f_1 + 2f_2 \end{bmatrix} \tag{44}$$

$$\vec{S}^1_{120,D_2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_0 \\ 2f_1 + f_2 \\ 2f_0 + 2f_1 + 2f_2 \end{bmatrix} \tag{45}$$

$$\vec{S}^1_{201,D_2} = \begin{bmatrix} 0 & 0 & 1 \\ 2 & 1 & 0 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_2 \\ 2f_0 + f_1 \\ 2f_0 + 2f_1 + 2f_2 \end{bmatrix} \tag{46}$$

From Equations (38) - (46), one can note that:

$$\vec{S}^1_{012,D_0} = \vec{S}^1_{201,D_1} = \vec{S}^1_{120,D_2} \tag{47}$$

$$\vec{S}^1_{120,D_0} = \vec{S}^1_{012,D_1} = \vec{S}^1_{201,D_2} \tag{48}$$

$$\vec{S}^1_{201,D_0} = \vec{S}^1_{120,D_1} = \vec{S}^1_{012,D_2} \tag{49}$$

Utilizing Equations (47) - (49), Figure 5 shows all of the (3,3) reversible kernels for Equations (38) - (46), respectively [24].

Using Equations (47) - (49), the following Equations hold (where the subscripts indicate the orderings of the literals ${}^k c$ in the RSGBFM in each of Equations (20) - (25), respectively) [24]:

$$\vec{S}^1_{(012,120,201),D_0} = \vec{S}^1_{(201,012,120),D_1} = \vec{S}^1_{(120,201,012),D_2} \tag{50}$$

$$\vec{S}^1_{(012,201,120),D_0} = \vec{S}^1_{(201,120,012),D_1} = \vec{S}^1_{(120,012,201),D_2} \tag{51}$$

$$\vec{S}^1_{(120,012,201),D_0} = \vec{S}^1_{(012,201,120),D_1} = \vec{S}^1_{(201,120,012),D_2} \tag{52}$$

$$\vec{S}^1_{(201,012,120),D_0} = \vec{S}^1_{(120,201,012),D_1} = \vec{S}^1_{(012,120,201),D_2} \tag{53}$$

$$\vec{S}^1_{(120,201,012),D_0} = \vec{S}^1_{(012,120,201),D_1} = \vec{S}^1_{(201,012,120),D_2} \tag{54}$$

$$\vec{S}^1_{(201,120,012),D_0} = \vec{S}^1_{(120,012,201),D_1} = \vec{S}^1_{(012,201,120),D_2} \tag{55}$$

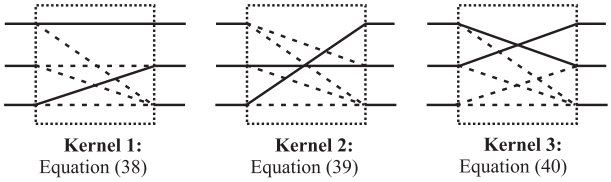


Fig. 5. (3,3) Reversible kernels for Equations (38) - (40), where dashed lines indicate multiplications by 2, and addition operation over GF(3) is used at the intersects of end lines. The three inputs to each of the three circuits are $\{f_0, f_1, f_2\}$, and output vectors from three circuits are: $\vec{S}^1_{012,D_0}$,$\vec{S}^1_{120,D_0}$ and $\vec{S}^1_{201,D_0}$, respectively.

Figure 6 shows all of the six (9,9) classes of reversible circuits using the parallel orderings of spectral vectors from Equations (50) - (55), respectively [24].

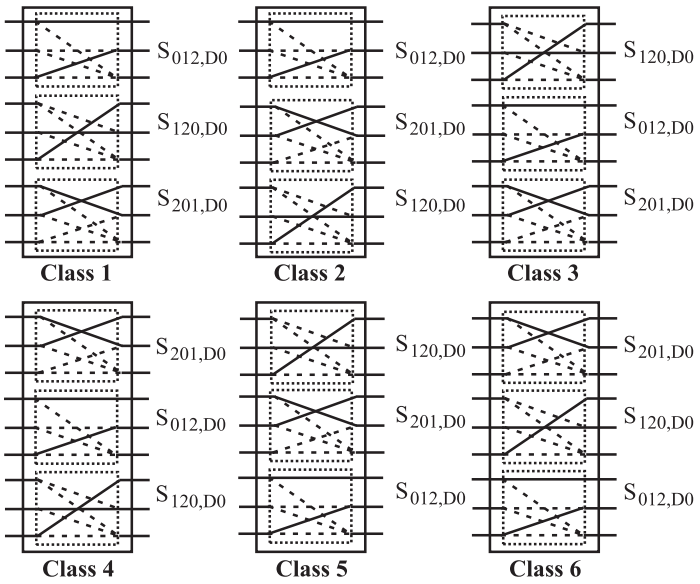


Fig. 6. (9,9) Reversible butterfly circuits for Equations (50)-(55). The inputs to each of the six circuits are $\{f_0, \ldots, f_8\}$.

Another type of (9,9) reversible butterfly circuits can be implemented for a ternary two-variable discrete function:

$$\vec{F} = \begin{bmatrix} f_0 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 \end{bmatrix}^T.$$

This is done by utilizing the (3,3) reversible kernel circuits in Figure 5, Equations (38) - (40), and by using the Kronecker product (defined in Equation (33)) over

GF(3) as follows:

$$\vec{S}^2_{012,D_0} = \left[\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix} \right] [\vec{F}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 \\ 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} [\vec{F}] \quad (56)$$

$$\vec{S}^2_{120,D_0} = \left[\begin{bmatrix} 0 & 0 & 1 \\ 2 & 1 & 0 \\ 2 & 2 & 2 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 & 1 \\ 2 & 1 & 0 \\ 2 & 2 & 2 \end{bmatrix} \right] [\vec{F}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 2 \\ 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 2 & 2 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 \\ 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} [\vec{F}] \quad (57)$$

$$\vec{S}^2_{201,D_0} = \left[\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 2 & 2 & 2 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 2 & 2 & 2 \end{bmatrix} \right] [\vec{F}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 & 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 1 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 1 \\ 2 & 2 & 2 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 \\ 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} [\vec{F}] \quad (58)$$

Figure 7 shows the two-variable (9,9) reversible butterfly circuits for Equations (56) - (58), respectively.

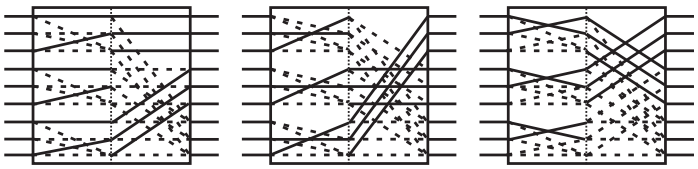


Fig. 7. (9,9) Reversible butterfly circuits for Equations (56) - (58), respectively. The nine inputs to each of the three circuits take the following order $\{f_0, f_1, \ldots, f_8\}$.

The same two methods of (1) parallel orderings (cf. Figure 6) and (2) Kronecker recursion (cf. Figure 7) can be implemented for any Galois radix N and

arbitrary number of variables N^m $(m = 1, 2, \ldots)$. Various families of fast permutation transform circuits for the reversible Shannon expansions have been also created [23, 26].

2.6 Reversible decision diagrams (RDDs)

A decision diagram (DD) is a fundamental data structure that is used extensively to represent and manipulate logic functions [27]. Reversible decision diagrams have been created using two methods [1]: (1) direct synthesis from classical decision diagrams by using (a) reversible gates and (b) the careful and optimal use of outputs from a horizontal level to the next and a vertical level to the next in order to minimize the number of gates and garbage lines used, and (2) the use of the classical rules for obtaining the corresponding reversible decision diagrams from their corresponding reversible decision trees (e.g., Figure 3) [1]: (a) *join all of the isomorphic nodes*, and (b) *remove all of the redundant nodes*. The following is a simple example that illustrates the creation of a two-valued RDDs using the first method.

Example 5. This example shows the realization of the Boolean function $F = ab \oplus bc \oplus ac$ using a reversible binary decision diagram (RBDD), as shown in Figure 8. Note that, in Figure 8b, the binary reversible Shannon primitives [1], also known as Fredkin gates [1], are used in the internal nodes.

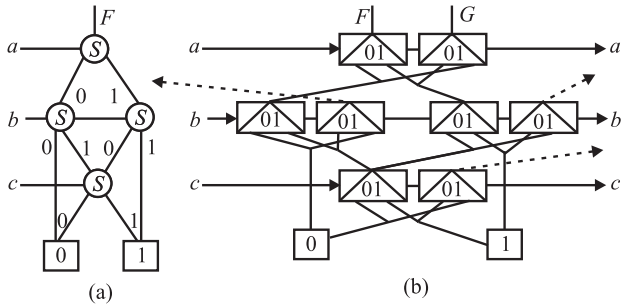


Fig. 8. Reversible DD: (a) BDD for the Boolean function $F = ab \oplus bc \oplus ac$, and (b) reversible BDD (RBDD) for F.

2.7 Reversible cascade circuits

The reversible expansions, from Section 2.1, have an important circuit realization using cascades [1]. One important motivation for the use of such topology is that one of the most efficient ways to realize quantum circuits, that are *intrinsically* reversible, is by using cascade-based circuit topology of intercon-

nected reversible quantum primitives [1]. Figure 9 shows two important multiple-valued reversible and quantum primitives: C-Not (C(NOT); Feynman) and C-C-Not (C^2NOT; C^2(NOT); Toffoli) gates [1, 10].

(a) (b)

Fig. 9. Ternary GF reversible quantum primitives: (a) (2,2) C-Not gate, and (b) (3,3) C-C-Not (C^2NOT) gate.

Table 2 proves the reversibility for the ternary (3, 3) Toffoli gate where the inputs are in order $\{a,b,c\}$ and the outputs are in order $\{a,b,d = c +_3 a *_3 b\}$.

Table 2. Reversibility proof for the ternary (3, 3) Toffoli gate where the inputs are in the order $\{a,b,c\}$ and the outputs are in the order : $\{a,b,d = c +_3 a *_3 b\}$.

000	000	100	100	200	200
001	001	101	101	201	201
002	002	102	102	202	202
010	010	110	111	210	212
011	011	111	112	211	210
012	012	112	110	212	211
020	020	120	122	220	221
021	021	121	120	221	222
022	022	122	121	222	220

As an example, by using the following reversible Davio_0 expansion (cf. Eqns. (30)-(32)):

$$\vec{f}_{D0} = \begin{bmatrix} 1 & 1+c & 1+2c+c^2 \\ 1 & c & c^2 \\ 1 & 2+c & 1+c+c^2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 2 & 1 & 0 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} f_{r0} \\ f_{r1} \\ f_{r2} \end{bmatrix} \tag{59}$$

One obtains the ternary reversible Davio_0 cascade shown in Fig. 10, where the quantum notation for ternary reversible C-Not and C^2NOT primitives is used [1].

One can observe that, by using Equation (59), other cascade forms can be created. Therefore, an important issue becomes apparent, when analyzing Figure 10 and Equation (59) as an example, is that one would like to find *optimal* methods for the cascade circuit synthesis in terms of: (1) minimizing the number of garbage outputs (i.e., minimizing outputs that are needed only for the purpose of reversibility), and (2) minimizing the number of gates that are internally used.

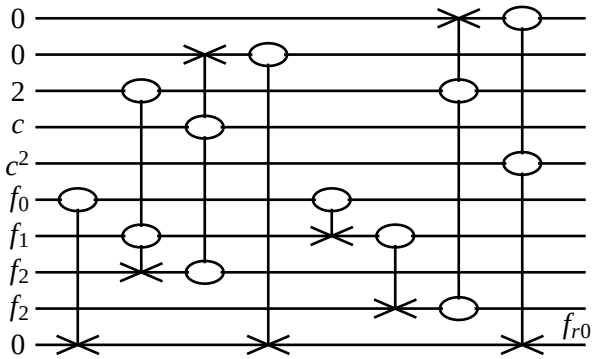


Fig. 10. Ternary GF quantum cascade for the realization of the output f_{r0} in the reversible expansion in Equation (59).

A second important issue is the design of a reversible cascade that is *quantum-realizable* [1, 10], i.e., the reversible circuit can be directly implemented (*mapped*) into a functioning quantum system. (This issue of technology mapping is important because while each quantum primitive is reversible, the opposite is not necessarily true, i.e., *not each reversible primitive is quantum*.) An important theorem that addresses this issue of quantum-realizable circuits is the theorem that states that any $C^2(U)$ gate can be built from $C(NOT)$ gates, $C(V)$ gates, and $C(V^+)$ gates, where $V^2 = U$ (or $VV = U$) and $VV^+ = I$ (cf. Barenco et al. Theorem [10].) For example, if $U = NOT = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ (i.e., $C^2(U)$ is the Toffoli gate) then $V = \frac{1}{2}\begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix}$ (i.e., square root of NOT gate) and $V^+ = \frac{1}{2}\begin{bmatrix} 1-i & 1+i \\ 1+i & 1-i \end{bmatrix}$ (i.e., square root of NOT Hermitian gate).

Other spectral transforms that transform functions in the quantum domain $\boldsymbol{Q}$ (which is the linear complex Hilbert vector space), such as the quantum Fourier transform, the quantum Walsh transform, and the quantum Chrestenson transform, have been also addressed [1] as methods that can be used to design quantum circuits of quantum discrete functions in the quantum domain (space) $\boldsymbol{Q}$.

2.8 Reversible systolic arrays

It has been shown in [28, 29] a new design method to implement *m*-ary logic functions bijectively through the realization of *m*-ary reversible functional expansions using *m*-ary reversible systolic arrays and the corresponding quantum systolic arrays as illustrated in Figure 11.

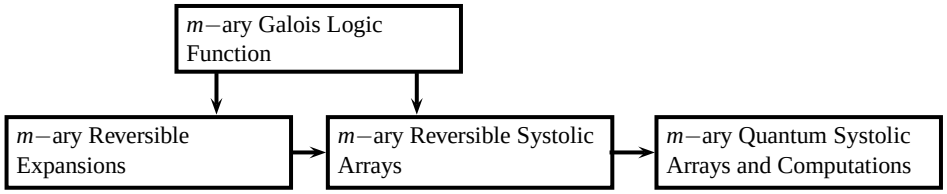


Fig. 11. The implementation of an *m*-ary Galois logic function bijectively using *m*-ary reversible systolic arrays and their corresponding *m*-ary quantum systolic arrays.

Since the architecture of conventional computers suffers from two inherent difficulties of (1) long communication paths and (2) the fact that the single CPU sequentially fetches and executes instructions, systolic architecture speeds up the computation through the following: (1) cost-effectiveness through *simplicity* and *regularity*, (2) *concurrency* (parallelism) through pipelining and multiprocessing, (3) regular communication wiring occurs only between neighboring processing elements (PEs) (i.e., the *elimination of global broadcasting*), and (4) I/O *bandwidth - throughput* improvements, where in a systolic architecture three factors are fundamental: (1) the type of the processing element (PE), (2) the systolic topology, and (3) the input/output ordering of data items in the I/O streams. Figure 12 shows an example of a 2D reversible systolic array [28,29], where each pulsation in the array consists of the following operations: (1) *shift* and (2) *multiply* and *add*.

Basic processing cells that are used in the construction of reversible systolic arithmetic arrays are the add-multiply cells. This kind of cells has the three inputs $\{a,b,c\}$ and the three outputs are $\{a=a,b=b,d=c+a*b\}$. One can assume *six interface registers* are attached at the I/O ports of a processing cell. All registers are clocked for synchronous transfer of data among adjacent cells. Hexagonally connected processors (i.e., processing elements (PEs)) can optimally perform matrix multiplication, where three data streams flow through the array in a pipelined fashion. One can follow the operation of the 2D reversible hexagonal systolic array by studying the data flow by moving transparencies of the band matrices over the network (cf. Figure 12). The reversible systolic array can finish the band matrix multiplication in T time units, where: $T=3n+\min(w1,w2)$. Therefore the computation time is linearly proportional to the dimension n of the matrix, and when the matrix bandwidths increase to $w1=w2=n$ (for dense matrices $[\boldsymbol{A}]$ and $[\boldsymbol{B}]$), the time becomes $O(4n)$, neglecting the I/O time delays. If one used a single additive-multiply processor to perform the same matrix multiplication, $O(n^3)$ computation time would be needed. The reversible systolic array thus has a speed gain of $O(n^2)$, and this becomes more apparent for large n. Multiplication of band matrices $[\boldsymbol{A}]$ and $[\boldsymbol{B}]$, $[\boldsymbol{A}]\cdot[\boldsymbol{B}]=[\boldsymbol{C}]$, and the associated definition of bandwidth is shown as follows:

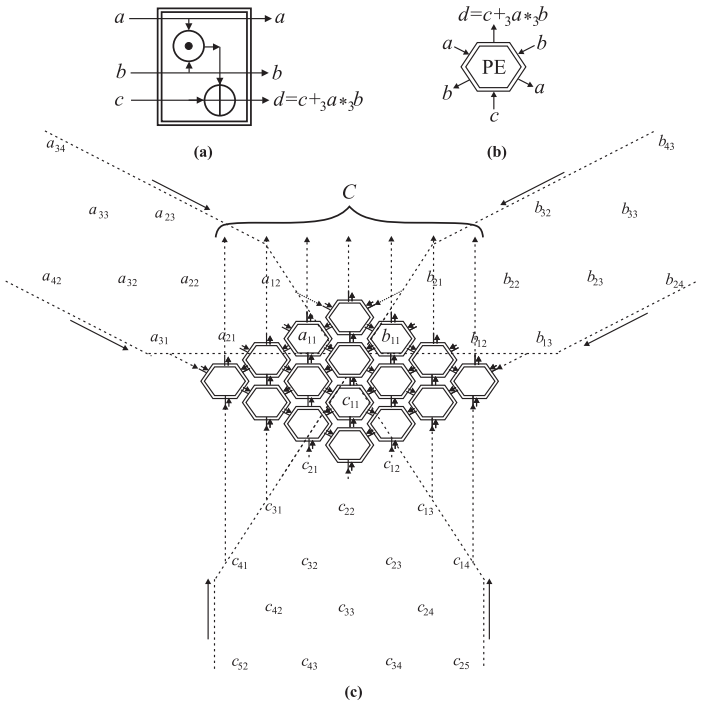


Fig. 12. Reversible GF(3) Kung systolic array: (a) reversible (3, 3) Toffoli gate, (b) reversible (3, 3) Kung cell, and (c) reversible Kung systolic architecture.

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & 0 \\ 0 & a_{42} & a_{43} & a_{44} & a_{45} \\ 0 & 0 & a_{53} & a_{54} & a_{55} \end{bmatrix}}_{\text{Band}_1} \cdot \underbrace{\begin{bmatrix} b_{11} & b_{12} & b_{13} & 0 & 0 \\ b_{21} & b_{22} & b_{23} & b_{24} & 0 \\ 0 & b_{32} & b_{33} & b_{34} & b_{35} \\ 0 & 0 & b_{43} & b_{44} & b_{45} \\ 0 & 0 & 0 & b_{54} & b_{55} \end{bmatrix}}_{\text{Band}_2} = \underbrace{\begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & 0 \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \\ 0 & c_{52} & c_{53} & c_{54} & c_{55} \end{bmatrix}}_{\text{Band}_3}$$

where: bandwidth$_1$: $w1 = 3+2-1 = 4$, bandwidth$_2$: $w2 = 2+3-1 = 4$, and bandwidth$_3$: $w3 = w1+w2-1 = 4+4-1 = 7$. Figure 12 shows the two-dimensional (2D) reversible hexagonal systolic array [28, 29] that implements the operation of multiplying two band matrices [$\boldsymbol{A}$] and [$\boldsymbol{B}$].

Each propagating data level in the three data flow streams in Figure 12 is called a *wave front*, the initial values of the input [$\boldsymbol{C}$] array elements (from the lower side in Figure 12) are all zeros, and the final (resulting) values of the output [$\boldsymbol{C}$] array elements (from the upper side in Figure 12) are obtained over GF(3) as follows: $c_{11} = a_{11}b_{11} + a_{12}b_{21}, c_{21} = a_{21}b_{11} + a_{22}b_{21}, c_{31} = a_{31}b_{11} + a_{32}b_{21}, c_{41} = a_{42}b_{21}$, $c_{51} = 0, c_{12} = a_{11}b_{12} + a_{12}b_{22}, c_{22} = a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32}, c_{32} = a_{31}b_{12} +$

$a_{32}b_{22}+a_{33}b_{32}, c_{42}=a_{42}b_{22}+a_{43}b_{32}, c_{52}=a_{53}b_{32}, c_{13}=a_{11}b_{13}+a_{12}b_{23}, c_{23}=a_{21}b_{13}+a_{22}b_{23}+a_{23}b_{33}, c_{33}=a_{31}b_{13}+a_{32}b_{23}+a_{33}b_{33}+a_{34}b_{43}, c_{43}=a_{42}b_{23}+a_{43}b_{33}+a_{44}b_{43}, c_{53}=a_{53}b_{33}+a_{54}b_{43}, c_{14}=a_{12}b_{24}, c_{24}=a_{22}b_{24}+a_{23}b_{34}, c_{34}=a_{32}b_{24}+a_{33}b_{34}+a_{34}b_{44}, c_{44}=a_{42}b_{24}+a_{43}b_{34}+a_{44}b_{44}+a_{45}b_{54}, c_{54}=a_{53}b_{34}+a_{54}b_{44}+a_{55}b_{54}, c_{15}=0, c_{25}=a_{23}b_{35}, c_{35}=a_{33}b_{35}+a_{34}b_{45}, c_{45}=a_{43}b_{35}+a_{44}b_{45}+a_{45}b_{55}, c_{55}=a_{53}b_{35}+a_{54}b_{45}+a_{55}b_{55}$.

The topological distribution of the processing elements (PEs) in the systolic structure, shown in Figure 12, is obtained as follows: # PEs in top-left = w_1 = 4, # PEs in top-right = w_2 = 4, # PEs in bottom = w_3 = 7, and the total # PEs = $4 \cdot 4$ = 16.

For the systolic implementation of *m*-ary functions using the corresponding GF(3), as an example of an *m*-ary GF logic, the matrix-based elements that are obtained using the GF(3) reversible Shannon and Davio functional expansions are obtained as follows, respectively:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ {}^0e & {}^1e & {}^2e & 0 & 0 \\ {}^1e & {}^2e & {}^0e & 0 & 0 \\ 0 & {}^0e & {}^1e & {}^2e & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & f_0 & 0 & 0 \\ 0 & 0 & f_1 & 0 & 0 \\ 0 & 0 & f_2 & 0 & 0 \\ 0 & 0 & f_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & {}^0e f_0 + {}^1e f_1 + {}^2e f_2 & 0 & 0 \\ 0 & 0 & {}^1e f_0 + {}^2e f_1 + {}^0e f_2 & 0 & 0 \\ 0 & 0 & {}^0e f_1 + {}^1e f_2 + {}^2e f_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{60}$$

where: $a_{21}={}^0e$, $a_{22}={}^1e$, $a_{23}={}^2e$, $a_{31}={}^1e$, $a_{32}={}^2e$, $a_{33}={}^0e$, $a_{42}={}^0e$, $a_{43}={}^1e$, $a_{44}={}^2e$, $b_{13}=f_0$, $b_{23}=f_1$, $b_{33}=f_2$, $b_{43}=f_0$, $c_{23}={}^0e f_0+{}^1e f_1+{}^2e f_2$, $c_{33}={}^1e f_0+{}^2e f_1+{}^0e f_2$, $c_{43}={}^2e f_0+{}^0e f_1+{}^1e f_2$.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1+e & 1+2e+e^2 & 0 & 0 \\ 1 & e & e^2 & 0 & 0 \\ 0 & 2+e & 1+e+e^2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & f_2 & 0 & 0 \\ 0 & 0 & 2f_0+f_1 & 0 & 0 \\ 0 & 0 & 2f_0+2f_1+2f_2 & 0 & 0 \\ 0 & 0 & f_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \tag{61}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & f_2+(1+e)(2f_0+f_1)+(1+2e+e^2)(2f_0+2f_1+2f_2) & 0 & 0 \\ 0 & 0 & f_2+e(2f_0+f_1)+e^2(2f_0+2f_1+2f_2) & 0 & 0 \\ 0 & 0 & f_2+(2+e)(2f_0+f_1)+(1+e+e^2)(2f_0+2f_1+2f_2) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

where:

$a_{21}=1, a_{22}=1+e, a_{23}=1+e+e^2, a_{31}=1, a_{32}=e, a_{33}=e^2, a_{42}=2+e, a_{43}=1+e+e^2, a_{44}=1, b_{13}=f_2, b_{23}=2f_0+f_1, b_{33}=2f_0+2f_1+2f_2, b_{43}=f_2, c_{23}=f_2+(1+e)(2f_0+f_1)+(1+2e+e^2)(2f_0+2f_1+2f_2), c_{33}=f_2+e(2f_0+f_1)+e^2(2f_0+2f_1+2f_2), c_{43}=f_2+(2+e)(2f_0+f_1)+(1+e+e^2)(2f_0+2f_1+2f_2)$.

Example 6 shows the reversible implementation, using the reversible Kung systolic array from Figure 12, of the two-digit ternary multiplication which is performed utilizing the mod-multiplication operator.

Example 6. Table 3 shows the maps for the ternary multiplication (M) and the output carry (C_{out}).

Table 3. Ternary mod-multiplication for ternary 2-digit multiplier: (a) multiplication (M), and (b) carry out (C_{out}).

(**a**)

b \ **a**	**0**	**1**	**2**
0	0	0	0
1	0	1	2
2	0	2	1

(**b**)

b \ **a**	**0**	**1**	**2**
0	0	0	0
1	0	0	0
2	0	0	1

The following is a bijective implementation, using GF(3) reversible 2D hexagonal Kung systolic array from Figure 12, of the two-digit ternary multiplication which is performed using the mod-multiplication operator.

$M =^1 a^1b + 2 \cdot^2 a^1b + 2 \cdot^1 a^2b +^2 a^2b$
$C_{out} =^2 a^2b$
$\rightarrow c_{33} = M,\ a_{31} =^1 a,\ b_{13} =^1 b,\ a_{32} = 2 \cdot^2 a,\ b_{23} =^1 b,\ a_{33} = 2 \cdot^1 a,\ b_{33} =^2 b,$
$a_{34} =^2 a,\ b_{43} =^2 b,$
$\rightarrow c_{11} = C_{out},\ a_{11} =^2 a,\ b_{11} =^2 b,\ a_{12} = 0,\ b_{21} = 0.$

The upper implementation follows directly by the substitution of the array's values using the corresponding m-ary functional literals. A second way to implement the upper multiplier is by using the many-variable ternary Kronecker-based reversible Shannon expansion (cf. Equation (36)) and the following Davio expansion:

$$\vec{f} = \begin{bmatrix} f_{r0} \\ f_{r1} \\ f_{r2} \\ f_{r3} \\ f_{r4} \\ f_{r5} \\ f_{r6} \\ f_{r7} \\ f_{r8} \end{bmatrix} = \left[\begin{bmatrix} 1 & 1+a & 1+2a+a^2 \\ 1 & a & a^2 \\ 1 & 2+a & 1+a+a^2 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1+b & 1+2b+b^2 \\ 1 & b & b^2 \\ 1 & 2+b & 1+b+b^2 \end{bmatrix} \right] \left[\begin{bmatrix} 0 & 0 & 1 \\ 2 & 1 & 0 \\ 2 & 2 & 2 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 & 1 \\ 2 & 1 & 0 \\ 2 & 2 & 2 \end{bmatrix} \right] \left[\begin{bmatrix} f_{a0} \\ f_{a1} \\ f_{a2} \end{bmatrix} \otimes \begin{bmatrix} f_{b0} \\ f_{b1} \\ f_{b2} \end{bmatrix} \right]$$

$$
= \begin{bmatrix}
1 & 1+b & 1+2b+b^2 & 1+a & (1+a)(1+b) & (1+a)(1+2b+b^2) & 1+2a+a^2 & (1+2a+a^2)(1+b) & (1+2a+a^2)(1+2b+b^2) \\
1 & b & b^2 & 1+a & (1+a)b & (1+a)b^2 & 1+2a+a^2 & (1+2a+a^2)b & (1+2a+a^2)b^2 \\
1 & 2+b & 1+b+b^2 & 1+a & (1+a)(2+b) & (1+a)(1+b+b^2) & 1+2a+a^2 & (1+2a+a^2)(2+b) & (1+2a+a^2)(1+b+b^2) \\
1 & 1+b & 1+2b+b^2 & a & a(1+b) & a(1+2b+b^2) & a^2 & a^2(1+b) & a^2(1+2b+b^2) \\
1 & b & b^2 & a & ab & ab^2 & a^2 & a^2b & a^2b^2 \\
1 & 2+b & 1+b+b^2 & a & a(2+b) & a(1+b+b^2) & a^2 & a^2(2+b) & a^2(1+b+b^2) \\
1 & 1+b & 1+2b+b^2 & 2+a & (2+a)(1+b) & (2+a)(1+2b+b^2) & 1+a+a^2 & (1+a+a^2)(1+b) & (1+a+a^2)(1+2b+b^2) \\
1 & b & b^2 & 2+a & (2+a)b & (2+a)b^2 & 1+a+a^2 & (1+a+a^2)b & (1+a+a^2)b^2 \\
1 & 2+b & 1+b+b^2 & 2+a & (2+a)(2+b) & (2+a)(1+b+b^2) & 1+a+a^2 & (1+a+a^2)(2+b) & (1+a+a^2)(1+b+b^2)
\end{bmatrix}
$$

$$
\times \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 2 \\
0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 2 & 0 & 2 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 2 & 2 & 2 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 \\
1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
f_{a0b0} \\ f_{a0b1} \\ f_{a0b2} \\ f_{a1b0} \\ f_{a1b1} \\ f_{a1b2} \\ f_{a2b0} \\ f_{a2b1} \\ f_{a2b2}
\end{bmatrix} \tag{62}
$$

where the values of the arrays $[\boldsymbol{A}]$ and $[\boldsymbol{B}]$ (in Figure 12) are given the corresponding values from the reversible Shannon and Davio expansions in a manner similar to the method used in Equations (60) and (61).

In Example 6, the implementation (utilizing Figure 12) can be done in two ways: (1) using a *single* matrix - matrix multiplication systolic array that multiplies the first basis vector matrix by the second vector of weighted sum of cofactors, and (2) using *two* chained matrix-matrix multiplication systolic arrays where the first systolic array multiplies the spectral transform matrix $[\boldsymbol{S}]$ and the cofactors vector and the second systolic array multiplies the basis vector matrix and the output vector that results from the first systolic array.

Although the Toffoli processing element (TPE) - based results that are presented in this section are illustrated for the case of the 2D hexagonal reversible systolic array, and since the add-multiply cell is the basic PE in these circuits and the TPE cell is the reversible GF counterpart of this fundamental add-multiply PE, other ternary add-multiply-based systolic arrays can be constructed reversibly using the interconnection between TPEs as well [28, 29]. Figure 13 shows the quantum circuit realization of the reversible systolic array from Figure 12 [28, 29], where the quantum data called qubits (quantum bits) are used in the input data streams as shown.

3 Group-Theoretic Methods

A finite group G is a set of finite number of elements together with a binary operation (called the group operation) satisfying the properties of (1) closure, (2) associativity, (3) identity, and (4) inverse. Group theory has been used in the design of digital circuits [13, 30]. Since several of the primitives used in reversible logic perform specific permutations of inputs, and since group theory is a mathematical

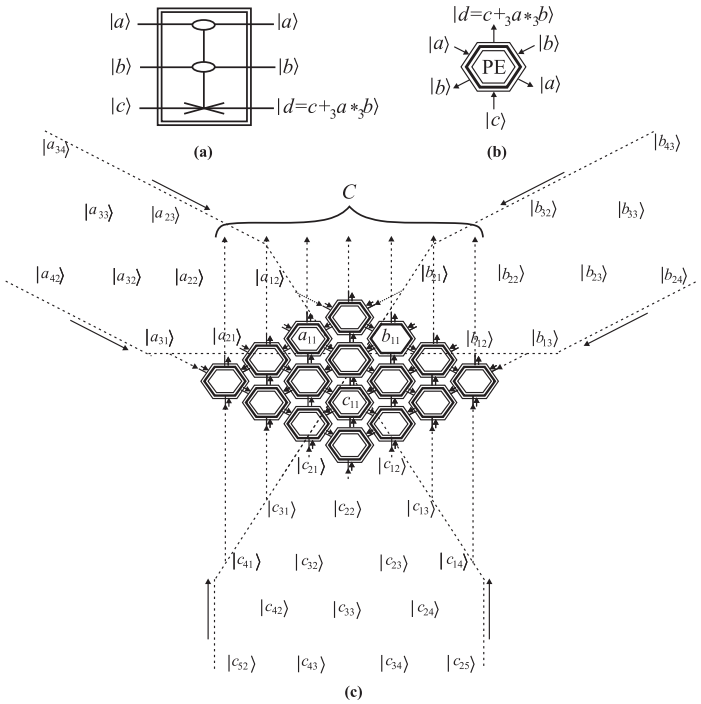


Fig. 13. Quantum GF(3) systolic array: (a) quantum (3, 3) Toffoli PE (TPE), (b) quantum (3, 3) Kung cell, and (c) quantum Kung systolic array.

tool that offers a powerful representation of systems that exhibit internal *symmetries* (e.g., permutations), it becomes natural to think of group-theoretic *compact* representation of the Galois reversible primitives.

A symmetric group S_n (i.e., of degree n and is the group of all permutations on n objects) is a permutation group of order $n!$, where each of the $n!$ group operators (permutations) is a group element. As an example, Figure 14 shows the 2-*cycle* (transposition) group representations of several important classes of GF(2) (i.e., binary) reversible gates [21]. General k-*cycle* group representations for reversible circuits made of serial-interconnected and parallel-interconnected reversible primitives are done by performing the appropriate step-by-step permutations of each stage of the reversible circuit.

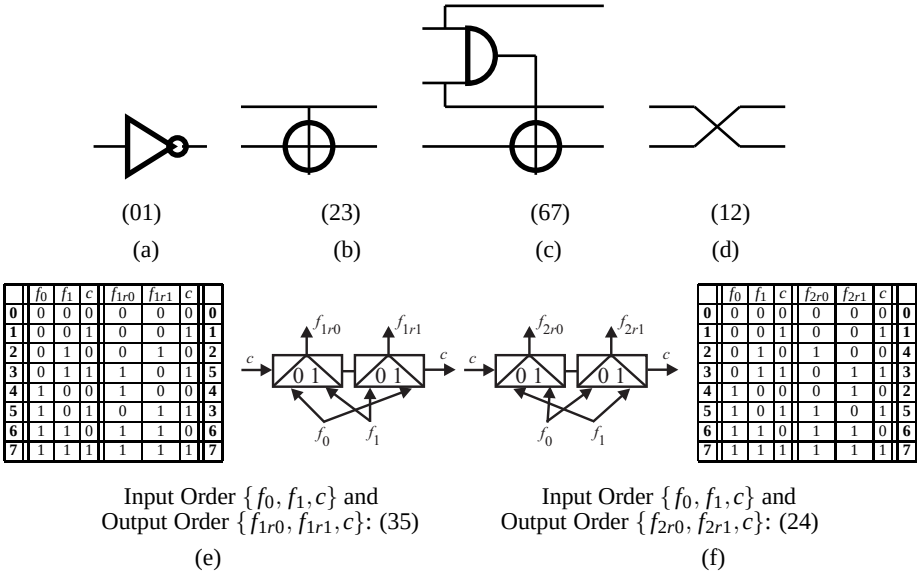
Example 7. Using the 2-cycle (i.e., transposition) group representations in Figure 14, the following are group representations of various reversible circuits:

7a. Stage1: Feynman gate (23), Stage2: Swap gate (12), Interconnect: Serial (○)→ Group representation: 3-cycle group element (123).

7b. Stage1: Swap gate (12), Stage2: Feynman gate (23), Interconnect: Serial (○) → Group representation: 3-cycle group element (132).

7c. Stage1: Operator (map) (123), Stage2: Operator (map) (24), Interconnect: Serial (○) → Group representation: 4-cycle group element (1423).

7d. Stage1: Not gate (01) in parallel with Feynman gate (23), Stage2: Toffoli gate (67), Interconnect: Serial (○) → Group representation: (04)(15)(2637).



	f_0	f_1	c	f_{1r0}	f_{1r1}	c	
0	0	0	0	0	0	0	**0**
1	0	0	1	0	0	1	**1**
2	0	1	0	0	1	0	**2**
3	0	1	1	1	0	1	**5**
4	1	0	0	1	0	0	**4**
5	1	0	1	0	1	1	**3**
6	1	1	0	1	1	0	**6**
7	1	1	1	1	1	1	**7**

	f_0	f_1	c	f_{2r0}	f_{2r1}	c	
0	0	0	0	0	0	0	**0**
1	0	0	1	0	0	1	**1**
2	0	1	0	1	0	0	**4**
3	0	1	1	0	1	1	**3**
4	1	0	0	0	1	0	**2**
5	1	0	1	1	0	1	**5**
6	1	1	0	1	1	0	**6**
7	1	1	1	1	1	1	**7**

Fig. 14. Group-theoretic representations for important reversible primitives: (a) Not gate (Inverter), (b) Feynman gate, (c) Toffoli gate, (d) Swap gate, (e) reversible GF(2) Shannon1 (Fredkin1) gate, and (f) reversible GF(2) Shannon2 (Fredkin2) gate.

The ternary reversible expansions in Equations (20) - (25) result in ternary (4,4) gates that can be represented using the symmetric group S that corresponds to specific permutations of 4 input objects (i.e., $\{f_0, f_1, f_2, c\}$ in Equations (20) - (25)), and by using ternary group-theoretic representation, Table 4 shows the group representations of Equations (20) - (25), respectively, for the input order $\{f_0, f_1, f_2, c\}$ and output order $\{f_{r0}, f_{r1}, f_{r2}, c\}$.

Note that each 2-digit decimal number within each of the number strings between parentheses in Table 4 is produced using the ternary number expansion $\sum_{n=0}^{3} c_n 3^n$ for the corresponding group-represented input row (vector $\{f_0, f_1, f_2, c\}$) and output row (vector $\{f_{r0}, f_{r1}, f_{r2}, c\}$) for each ternary truth table of Equations (20) - (25).

One can observe that, by using the group-theoretic representations in Table 4, the spectral-based (i.e., spectral generated) reversible expansions in Equations (20) - (25) can be further classified into two families [21]: (1) 2-cycle permutations

family1 that contains Equations (20), (23), and (24), and (2) 3-cycle permutations family2 that contains Equations (21), (22), and (25).

Table 4. Group-theoretic representations of the ternary reversible expansions (gates) in Equations (20) - (25).

Equation	**Group-Theoretic Representation**
20	(0309)(0529)(0618)(0856)(1028)(1331)(1438)(1521)(1634) (1765)(1955)(2258)(2347)(2561)(2674) (3036)(3345)(3559) (4248)(4468)(4664)(4967)(5270)(5377)(5763)(6072)(6975)
21	(041028)(052911)(071955)(085620)(133731)(143238) (164658)(175947)(226434)(233565)(257361) (266274) (434967)(446850)(527670)(537177)
22	(032709)(051129)(065418)(082056)(123036)(143832) (155745)(174759)(213363)(236535)(246072) (267462) (426648)(445068)(516975)(537771)
23	(0428)(0511)(0755)(0820)(0927)(1230)(1337)(1533)(1664) (1723)(1854)(2157)(2246)(2460)(2573) (3238)(3458)(3547) (4367)(4450)(4563)(4866)(5169)(5276)(5965)(6274)(7177)
24	(0327)(0410)(0654)(0719)(1129)(1236)(1432)(1563)(1622) (1735)(2056)(2145)(2359)(2472)(2662) (3137)(3357)(3446) (4266)(4349)(4765)(5068)(5175)(5371)(5864)(6173)(7076)
25	(030927)(042810)(061854)(075519)(123630)(133137) (154557)(165846)(216333)(223464)(247260) (256173) (424866)(436749)(517569)(527076)

As another example of group-based representation, we obtain (121314)(151716)(212322)(242526) as the group-theoretic representation of the ternary Toffoli gate. The (121314)(151716)(212322)(242526) group-theoretic representation of the ternary Toffoli gate stems from the fact that the ternary Toffoli primitive transforms the inputs as follows: 12→13→14→12, 15→17→16→15, 21→23→22→21, 24→25→26→24, while preserving the domain-range mapping for the rest (cf. Table 2).

4 Elementary Cellular Automata (ECA) - Based Synthesis

A cellular automaton is a decentralized universal computing model of *time-based* (i.e., *temporal*) digital circuits and systems that provide an excellent platform for performing complex computation with the help of only local information. A CA consists of a spatial *lattice* of cells, each of which, at time t, can be in one m states. The lattice starts out with some initial configuration of local states (where configuration means the pattern of states over the entire lattice) and at each time step, the states of all cells in the lattice are synchronously updated. The communica-

tion in CA between constituent cells is limited to local interaction, and the overall structure can be viewed as a parallel processing device. Elementary Cellular Automata (ECA) as a special type of CA has been proven to be a powerful computing paradigm for the following properties: (1) universality: an ECA can model any discrete system, and thus it is a powerful logically *complete* system from which all functions can be obtained and can be used to model complex systems; (2) simplicity: an elementary cellular automaton is the building block of the elementary cellular automata, which is a simple structure that evolves over time using specific evolution rules, that leads to modeling complex discrete systems using such simple structures; and (3) regularity: the evolution of ECA to generate discrete time systems consists of geometrically evolving cellular grids. Set-theoretically, a regular ECA rule is a mapping of $(s_t(i-1) \otimes s_t(i) \otimes s_t(i+1))$ onto $s_{t+1}(i)$ (where $\otimes$ here means the Cartesian product).

An ECA consists of an array of cells in one dimension (1-D). In a Boolean ECA, each cell can take on one of two states $\{0,1\}$ where the binary string representing the array changes at discrete time steps (intervals). The Boolean ECA dynamics is commonly represented: (1) *spatially*: by a horizontal sequence of 0's and 1's or of white and black cells, and (2) *temporally*: time, in successive rows, is the vertical axis. The next state of any cell in ECA depends only upon its present "neighborhood", which includes (a) the state of the cell itself and (b) those of its immediate neighbors to the left and right. That is, if $s_t(i)$ is the state of cell i at time t, the dynamic law governing the Boolean ECA is described by the Boolean function (mapping):

$$s_{t+1}(i) = f(s_t(i-1), s_t(i), s_t(i+1)) \tag{63}$$

Since there are $2^3 = 8$ possible neighborhoods and since each neighborhood can map into either of the two states of $s_t(i+1)$, then there are $2^8 = 256$ mappings (or ECA rules).

The evolution that results from Equation (63) is an irreversible evolution [31, 32]; the evolution of ECA according to Equation (63) in general leads to *irreversible dynamics*, i.e., result is not a one-to-one mappings between vectors of inputs and outputs, and thus the vector of input states cannot be always uniquely reconstructed from the vector of output states. To achieve reversible discrete dynamics, one may use the following alternative definition of ECA evolution [31,32]:

$$s_{cr,t+1}(i-1) = f_1(s_t(i-1), s_t(i), s_t(i+1)) \tag{64}$$

$$s_{cr,t+1}(i) = f_2(s_t(i-1), s_t(i), s_t(i+1)) \tag{65}$$

$$s_{cr,t+1}(i+1) = f_3(s_t(i-1), s_t(i), s_t(i+1)) \tag{66}$$

where subscript c means conservative and subscript r means reversible, such that

the (3,3) mappings from one step to the next are conservative and reversible and thus producing a CRECA.

Example 8. Using the algorithm CRBF [31], Table 5 shows a a (3,3) conservative reversible map for ECA Rule # 170.

Table 5. An example of CRECA for $s_{t+1}(i-1)$ Rule #170.

$s_t(i-1)$	$s_t(i)$	$s_t(i+1)$	$s_{t+1}(i-1)$	$s_{t+1}(i)$	$s_{t+1}(i+1)$
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	0	1	1
1	1	1	1	1	1

The (3,3) conservative reversible mapping in Table 5 for Rule #170 can be represented by the set of three binary strings as follows $\{10101010, 11110000, 11001100\}_2$, which produces the set of Rule $\#\{170, 240, 204\}$.

Figure 15 shows an example of ECA discrete system dynamics for irreversible Rule #240 (Figure 15a) versus reversible Rule $\#\{170, 240, 204\}$ (cf. Table 5) using the same initial condition $\{110010101\}$ (Figure 15b). While the evolution in Figure 15a is a non-overlapping neighbor evolution, the evolution in Figure 15c is an overlapping neighbor evolution. One notes that the irreversible overlapping-based neighbor ECA evolution for Rule #240 in Figure 15c leads to a more complex evolution than the irreversible non-overlapping-based neighbor ECA evolution for Rule #240 in Figure 15a. One can also observe that if one conducts a 3-block reversible evolution with overlapping neighbor (Figure 15d) then several cell(s) will result with conflict values inside it, and this case will be forbidden (avoided) since the value and its "opposite" cannot possess the same spatial location (address) at the same time. Therefore, 3-block reversible non-overlapping neighbor ECA evolutions (such as in Figure 15b) will be only used.

Given: (1a) 1-D array length l, (1b) time steps n, (2) initial condition (distribution), and (3) reversible maps $\{f_1, f_2, f_3\}$, the *analysis* of CRECA refers to the finding of the CRECA evolution at step $t+1$ from step t. While analysis is useful to explore the whole space of all potential reversible system dynamics, the opposite problem of *synthesis* is the more interesting problem. The synthesis problem can be stated as follows: Given (1a) 1-D array length l, (1b) discrete time steps n, (2) a priori *known or assigned* (i.e., a priori unknown) initial condition, and (3) the result of

a *total* spatial evolution over time, produce the reversible maps $\{f_1, f_2, f_3\}$. It turns out that, while CRECA analysis is relatively an easy problem, CRECA synthesis is a more difficult one.

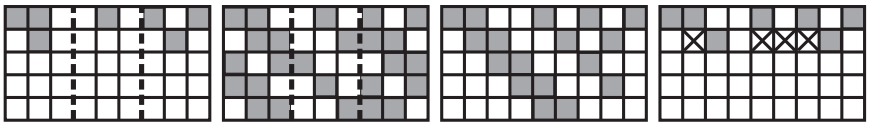


Fig. 15. Discrete system dynamics for: (a) irreversible non-overlapping neighbor ECA evolution for Rule #240, (b) reversible non-overlapping neighbor ECA evolution for Rule #{170,240,204}, (c) irreversible overlapping neighbor ECA evolution for Rule #240, and (d) reversible overlapping neighbor ECA evolution for Rule #{170,240,204} where x means a cell with contradictory values (i.e., a cell with more than one value at the same spatial location.)

Modeling and processing using Swap-based circuits is important since many of the two-valued and many-valued quantum circuit implementations use two-valued and multiple-valued quantum *Swap*-based and *Not*-based gates [1, 10]. This can be important, since the Swap and Not gates are basic primitives in quantum computing, from which many other *m*-valued fundamental gates are built, such as [1]: (1) *m*-valued Not gate, (2) *m*-valued Controlled-Not gate (*m*-valued C-Not gate or *m*-valued Feynman gate), (3) *m*-valued Controlled-Controlled-Not gate (m-valued C^2-Not gate or *m*-valued Toffoli gate), (4) *m*-valued Swap gate, and (5) *m*-valued Controlled-Swap gate (*m*-valued C-Swap gate or *m*-valued Fredkin gate). The following is an algorithm to generate one possible Swap-based CRECA (SCRECA) [31].

Algorithm SCRECA

---

1. For a BECA (Boolean ECA or binary-input binary-output ECA) map specified as follows: (1) spatial length (i.e., array length) is l, and (2) temporal length (i.e., number of steps) is n.
2. **If** the length l is not a multiplication of 3, **Then** add minimum number of columns with zero values to make the length $l/3$ is an integer, **Else** goto 3.
3. For temporal (row) index $K = 1, 2, 3, \ldots, n$, spatial automaton a_k with index $k = 0, 1, 2, \ldots, (l/3) - 1$, and evolution matrix from an arbitrary time $(z-1)$ to time z: $[M_{(z-1)z}]$, find transformation matrix from time (row) p (i.e., $t+p$) to time (row) q (i.e., $t+q$) as follows:

$$\vec{a}^T_{k,t+q} = \vec{a}^T_{k,t+p}[[M_{(q-1)q}] \ldots [M_{(p+1)(p+2)}][M_{p(p+1)}]]^T \text{ or}$$
$$\vec{a}_{k,t+q} = [[M_{(q-1)q}] \ldots [M_{(p+1)(p+2)}][M_{p(p+1)}]]\vec{a}_{k,t+p}$$

where t, p, and q are positive integers.

4. For top-to-down and left-to-right evolution, **goto** 5.
5. Since the CRECA is *conservative*, then by using the Swap-based reversible primitives, synthesize a reversible circuit for the CRECA map as follows:

   5a. **For** $(m = 0; m < n; m++)$
   For $(k = 0; k \leq (l/3) - 1; k++)$

   5b. Perform one-to-one spatial mapping of permuted automaton cell a_k between levels m and $(m+1)$ into (3,3) Swap-based reversible primitive (cf. Figure 16) between stages m and $(m+1)$ in the corresponding reversible circuit.

6. **End**

The SCRECA Algorithm [31] can be used for modeling (representation) and processing (operation) reversibly on the final resulting lattice of the ECA evolution, whether that evolution was conducted using a non-overlapping neighbor ECA evolution or an overlapping neighbor ECA evolution.

Since the elementary cellular automaton in its fundamental form is a 3-cell block, then the SCRECA algorithm is based on mapping a partition of the spatial state of the automata in blocks of three cells and then finding a suitable permutation matrix reflecting the behavior of the conservative reversible system. This simplicity of the Swap-based SCRECA algorithm is also the reason for its ability to model complex evolutions.

As SCRECA is: (1) reversible and (2) conservative, the output of each automaton $\vec{a}^T_{k,t+q}$ can be always obtained as *permutation* of input $\vec{a}^T_{k,t+p}$. Therefore, the matrix $[[M_{(q-1)q}]\dots[M_{(p+1)(p+2)}][M_{p(p+1)}]]^T$(i.e., the total matrix that results from multiplying the matrices:$\{[M_{(q-1)q}],\dots,[M_{(p+1)(p+2)}],[M_{p(p+1)}]\}$) is always a *permutation* matrix. For example, for a 3-input 3-output permutation, Figure 16 shows all possible reversible (3,3) Swap-based permutations using the Wire (Buffer) and Swap reversible logic primitives. The matrix representation in Figure 16 is obtained by solving for the output spatial state (vector) permutation from the input state (vector). This is shown for Figure 16d as an example as follows: $\begin{pmatrix}1&0&0\\0&0&1\\0&1&0\end{pmatrix}\begin{pmatrix}u\\v\\w\end{pmatrix}=\begin{pmatrix}u\\w\\v\end{pmatrix}$, where $\begin{pmatrix}u\\v\\w\end{pmatrix}$ is the input vector and $\begin{pmatrix}u\\w\\v\end{pmatrix}$ is the output vector in Figure 16d, respectively.

One notes that the above algorithm SCRECA can be used *spatially* for any of the following three cases: (a1) evolving a single automaton, (b1) evolving any set

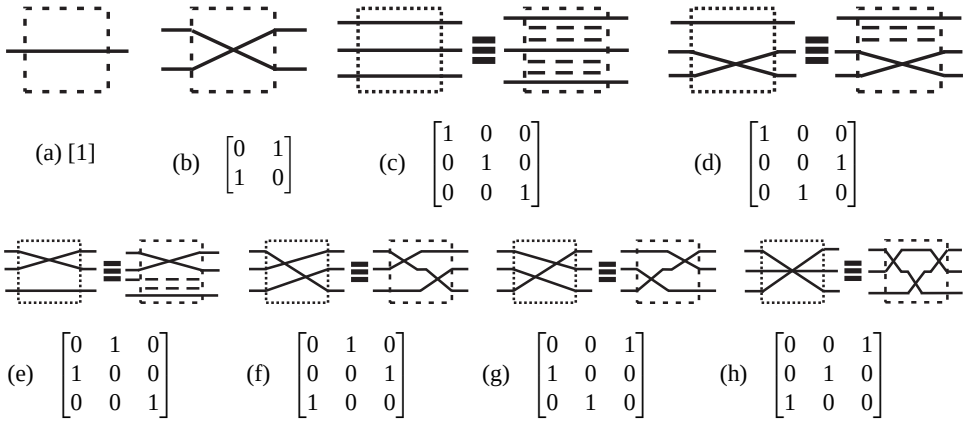


Fig. 16. Two-valued (binary) reversible and conservative permutation-based circuits: (a) (1,1) Wire (Buffer), (b) (2,2) Swap, and (c) - (h) all possible reversible (3,3) Swap-based primitives.

of automata at the same time, and (c1) evolving all of automata at the same time. Also, one notes that the above algorithm SCRECA can be used *temporally* for any of the following three cases: (a2) step-by-step automata evolution between two consecutive times (steps) $(k-1)$ and k, (b2) automata evolution between any two times (steps) p and q, and (c2) total automata evolution for all of steps n. While temporal complexity in methods (a2) through (c2) result in step-by-step evolution matrix transformations (i.e., matrix multiplications), methods (a1) through (c1) result in spatial complexity that determines the number of input-output permutation primitive, e.g., (3,3), (6,6), (9,9), etc. The determining factor for using methods (a1) - (c2) can be, for example, the complexity of possible circuit implementation of the resulting reversible circuit models.

5 Conclusions and Future Work

This paper presents several recent developments in the reversible and quantum logic circuit synthesis of binary and multiple-valued switching circuits. This review includes the synthesis methods and representations using: (1) spectral transforms and functional expansions, (2) group theory, and (3) cellular automata. Future work will involve more detailed investigation into cost / benefit based analysis (evaluation) of such methods. Future work will also involve items such as new group-based synthesis methods for the design of reversible and quantum circuits, and the investigation of the implementation of reversible cellular automata for the test of reversible and quantum circuits.

References

[1] A. N. Al-Rabadi, *Reversible Logic Synthesis: From Fundamentals to Quantum Computing*. New York: Springer-Verlag, 2004.

[2] C. Bennett, "Logical reversibility of computation," *IBM J. of Research and Development*, vol. 17, pp. 525–532, 1973.

[3] A. D. Vos, "Reversible computing," *Progress in Quantum Electronics*, vol. 23, pp. 1–49, 1999.

[4] R. Feynman, "Quantum mechanical computers," *Optics News*, vol. 11, pp. 11–20, 1985.

[5] E. Fredkin and T. Toffoli, "Conservative logic," *Int. J. of Theoretical Physics*, vol. 21, no. 3/4, pp. 219–253, 1982.

[6] P. Kerntopf, "On efficiency of reversible logic (3,3)-gates," in *Proc. Int. Conf. on Mixed Design of Integrated Circuits and Systems (MIXDES)*, June 2000, pp. 185–190.

[7] R. Landauer, "Irreversibility and heat generation in the computational process," *IBM J. of Research and Development*, vol. 5, pp. 183–191, 1961.

[8] N. Margolus, "Physics and computation," Ph.D. dissertation, Massachusetts Institute of Technology, 1988.

[9] D. Maslov and G. Dueck, "Garbage in reversible designs of multiple-output functions," in *Proc. Int. Symp. Represent. Methodol. and Future Comput. Tech.*, 2003, pp. 162–170.

[10] M. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[11] P. Picton, "Optoelectronic multi-valued conservative logic," *Int. J. of Optical Computing*, vol. 2, pp. 19–29, 1991.

[12] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Reversible logic circuit synthesis," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, 2002, pp. 125–132.

[13] G. Yang, X. Song, W. N. Hung, F. Xie, and M. A. Perkowski, "Group theory based synthesis of binary reversible circuits," in *The 3rd Annual Conference on Theory and Applications of Models of Computation (TAMC2006)*, 2006, pp. 365–374.

[14] K. Roy and S. Prasad, *Low-Power CMOS VLSI Circuit Design*. John Wiley & Sons Inc., 2000.

[15] C. Mead and L. Conway, *Introduction to VLSI Systems*. Addison-Wesley, 1979.

[16] K. R. Rao and D. F. Elliott, *Fast Transforms Algorithms, Analyses, Applications*. Academic Press Inc., 1982.

[17] S. Agaian, J. Astola, and K. Egiazarian, *Binary Polynomial Transforms and Nonlinear Digital Filters*. New York: Marcel Dekker Inc., 1995.

[18] B. J. Falkowski and C. Fu, "Family of fast transforms over GF(3) logic," in *Proc. of the IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, 2003, pp. 323–328.

[19] R. S. Stankovic, *Spectral Transform Decision Diagrams in Simple Questions and Simple Answers*. Nauka, Belgrade, 1998.

[20] S. M. Reddy, "Easily testable realizations of logic functions," *IEEE Trans. on Comp.*, vol. C-21, pp. 1183–1188, 1972.

[21] A. N. Al-Rabadi, "New classes of kronecker-based reversible decision trees and their group-theoretic representations," in *Proc. of the International Workshop on Spectral*

Methods and Multirate Signal Processing (SMMSP), Vienna, Austria, Sept. 11–12, 2004, pp. 233–243.

[22] A. N. Al-Rabadi and M. Perkowski, "New families of reversible expansions and their regular lattice circuits," *Journal of Multiple-Valued Logic and Soft Computing (MVLSC)*, vol. 11, no. 3-4, pp. 213–238, 2005.

[23] A. N. Al-Rabadi, "Reversible fast permutation transforms for quantum circuit synthesis," in *Proc. of the IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, Toronto, Canada, May 19–22, 2004, pp. 81–86.

[24] ——, "Quantum circuit synthesis using classes of GF(3) reversible fast spectral transforms," in *Proc. of the IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, Toronto, Canada, May 19–22, 2004, pp. 87–93.

[25] ——, "Spectral techniques in the reversible logic circuit synthesis of switching functions," in *Proc. of the International Workshop on Spectral Methods and Multirate Signal Processing (SMMSP)*, Vienna, Austria, Sept. 11–12, 2004, pp. 271–279.

[26] ——, "New classes of reversible butterfly diagrams and their quantum circuits," in *Proc. of the International Workshop on Spectral Methods and Multirate Signal Processing (SMMSP)*, Riga, Latvia, June 20–22, 2005, pp. 117–121.

[27] R. E. Bryant, "Graph-based algorithms for boolean functions manipulation," *IEEE Trans. on Comp.*, vol. C-35, no. 8, pp. 667–691, 1986.

[28] A. N. Al-Rabadi, "Reversible systolic arrays, Part I: Two-valued bijective single-instruction multiple-data (SIMD) architectures and their quantum extensions," in *Proc. of the International Workshop on Spectral Methods and Multirate Signal Processing (SMMSP)*, Moscow, Russia, Sept. 1–2, 2007.

[29] ——, "Reversible systolic arrays, Part II: m-ary equipollent SIMD circuits and their quantum realizations," in *Proc. of the International Workshop on Spectral Methods and Multirate Signal Processing (SMMSP)*, Moscow, Russia, Sept. 1–2, 2007.

[30] T. Sasao, "Cascade realizations of two-valued input multiple-valued output functions using decomposition of group functions," in *Proc. of the IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, 2003, pp. 125–132.

[31] A. N. Al-Rabadi and W. Feyerherm, "Reversible conservative noisy elementary cellular automata (ECA) circuits and their quantum computation," in *Proc. of the IEEE/ACM International Workshop on Logic and Synthesis (IWLS)*, Temecula, California, June 2–4, 2004, pp. 273–279.

[32] A. N. Al-Rabadi, "m-valued quantum representations and operations of elementary cellular automata," in *Proc. of the 11th International Symposium on Robotics and Applications (ISORA) in the World Automation Congress (WAC)*, Budapest, Hungary, July 2006.