Regular Realization of Symmetric Functions using Reversible Logic

Marek Perkowski, Pawel Kerntopf+, Andrzej Buller*, Malgorzata Chrzanowska-Jeske, Alan Mishchenko, Xiaoyu Song, Anas Al-Rabadi, Lech Jozwiak@, Alan Coppola$ and Bart Massey

PORTLAND QUANTUM LOGIC GROUP

Portland State University, Portland, Oregon 97207-0751.
*+Technical University of Warsaw, Poland, *ATR, Kyoto, Japan, @ Technical University of Eindhoven, $ Cypress Semiconductor Northwest*

Abstract

Reversible logic is of growing importance to many future computer technologies. We introduce a regular structure to realize symmetric functions in binary reversible logic. This structure, called a ***2 * 2 Net Structure***, allows for more efficient realization of symmetric functions than the methods shown by previous authors**.** Our synthesis method allows to realize arbitrary symmetric function in a completely regular structure of reversible gates with smaller "garbage". Because every Boolean function is symmetrizable by repeating input variables, our method is applicable to arbitrary multi-input, multi-output Boolean functions and realizes such arbitrary function in a circuit with a relatively small number of additional gate outputs. The method can be also used in classical logic. Its advantages in terms of numbers of gates and inputs/outputs are especially seen for symmetric or incompletely specified functions with many outputs.

1. Introduction

It is well known that Moore's Law will stop to function around year 2010 and something dramatic will therefore have to happen in microelectronics not later than in the middle of our Century [44],[45].
Optimists believe that quantum computers (QCs will be also *reversible*) will be built and pessimists rely only on other power-saving scalable technologies at the price of essential slowdown of technological improvements in speed and number of gates. In any case, whether the "optimists" or the "pessimists" will prove to be right, the role of reversible computing can only increase in coming years, because, as proved by Landauer [20],[19], it is a ***necessary*** condition for power not be dissipated in the circuit that the circuit be build from reversible gates (observe, *necessary but not sufficient*). *Reversible are circuits (gates) that have the same number of inputs and outputs and are one-to-one mappings between vectors of inputs and outputs; thus the vector of input states can be always uniquely reconstructed from the vector of output states*. *Conservative* are circuits that have the same number of ones in inputs and outputs. Our circuits are both reversible and conservative. Because truly low-power circuits cannot be built without the concepts of reversible logic, various technologies for reversible logic are recently intensively studied. These technologies include; 1) standard CMOS that can use university-available foundry, 2) optical technologies that can be realized with the state of the art materials, 3) quantum logic (QL) technologies, some of which proved to be physically realizable only very recently.

Reversible logic adders [6-12] and complete microprocessors [45] have been built, but using perhaps some "ad hoc" and not published logic design methods. Surprisingly little has been however published on systematic logic synthesis and optimization methods for reversible logic. If something was published, it was usually by people whose background is physics, mathematics, logic, or theoretical computer science, thus the logic synthesis aspects are still underdeveloped. In theory, classical logic synthesis methods can be used, but when adapted to reversible gates, they create unrealistically high numbers of additional gate output signals, making the circuit extremely complex. A good synthesis algorithm for reversible logic (RL) should not create an excessive "garbage" [17] or "waste of outputs". Based on our previous research, we found that there is a very good "match" between the requirements of reversible logic [2,4,9,11,12,14,17,18,33,43,44] and the opportunities given by the regular logic/layout structure [46] and Linearly Independent Logic [47,48].

We believe that regular structures are good for reversible logic, because it is easier to re-use in them the additional outputs of reversible gates, instead of "wasting" them. In addition, we believe that the Linearly Independent Logic [46,47,48] with its reversibility properties should be useful as well. Observe for example, that the Shannon and Davio expansions are used for some outputs in fundamental reversible gates of Fredkin and Toffoli, and their generalizations can be used to create new multivalued and multi-input (more than 3) reversible gates. As just one example of our general methodology based on these principles, we introduce here a regular structure to realize symmetric functions in binary reversible logic. This structure, although somehow similar to Lattice Structures introduced by us previously [50,51] and especially the MOPS structures [52] as well as to realizations from [28], is new and we call it a ***2 * 2 Net Structure.*** By a *regular structure* we understand a logic circuit and its physical layout structure being an array of identical cells regularly connected, or a structure composed of few, regularly connected, structures of this type, called *planes*. For instance, a well-known PLA is a regular structure with AND and OR planes. EXOR PLA is a regular structure with AND and EXOR planes. By *regularly connected*, we understand that every cell (except of boundary cells) is connected to its *k* neighbors. In the proposed structure, the cell, composed of two gates, has two inputs from neighbors, two outputs to neighbors, and two garbage outputs.

The method presented below allows to realize arbitrary symmetric function in a completely regular structure of reversible gates with little "garbage". By a *(totally) symmetric function* [49] we understand a Boolean function which is invariant to permuting any of its input variables. *Partially symmetric function* is invariant only to some input permutations. As discussed in [50] and the literature cited there, every Boolean function can be made symmetric, or *"is symmetrizable*". Functions are made symmetric by repeating their input variables, for instance a function F(a,b,c) that is not symmetric becomes symmetric when transformed to (incompletely specified) function F2(a1,b,a2,c) such that a1 = a2 =a and for every input vector (a,b,c) F(a,b,c) = F2(a1,b,a2,c). This function has an output don't care for every combination of inputs when a_1 != a_2. Because every (multi-output) Boolean function is symmetrizable, our method presented here is applicable to arbitrary multi-input, multi-output Boolean function. Recently efficient methods for symmetrization have been developed in our group, making application of the proposed here methods practical for at least some percent of non-symmetrical functions. These methods make use of more general definitions of symmetry than presented here [52]. We found that for high percent of practical function benchmarks the number of variable repetitions is small.

The goal of our Portland Quantum Logic research group is to create efficient logic synthesis methodologies for RL and QL, based on new gates, new structures (usually regular) and new design algorithms. The technique presented here is technology independent and can be thus used in association with any known or future reversible technology (also with different gates used in the structure). We believe, however, that the regularity of our networks will constitute an additional asset for the forthcoming technologies, especially nano-technologies.

The remaining of the paper is as follows. In section 2 basic principles of creating logic synthesis algorithms for reversible logic are reviewed to make this paper self-contained. Binary and Multi-Valued Fredkin gates are introduced in section 3. MV Fredkin gate is the basic gate of our regular structures. Section 4 introduces our idea and illustrates it with examples. Section 5 presents future research and concludes the paper. We also provide a comprehensive literature for our readers to help them in their own research.

2. Principles of creating logic synthesis algorithms for Reversible Logic

Let us recall that:

- Every Boolean function can be build from (binary) *Fredkin gates. (FG).* Such gate has three inputs A, B, C and three outputs, P, Q , R. It is specified by the gate equations:

  P = A
  Q = if A then C else B
  R = if A then B else C

- In addition, it is convenient (although not necessary) to use *Feynman gates ("controlled NOT"* or "*quantum XOR")* gates. Such gates have two inputs, A and B, and two outputs, P and Q. They are called linear, and are described by the following equations:

  P = A
  Q = A EXOR B

- In reversible logic wires can cross one another, so there is no requirement of circuit's planarity. But in

Quantum Logic the wires cannot cross [14-16]. Thus structures with no wire crossing will have additional advantages for realization, especially in truly quantum logic – this advocates developing logic synthesis methods to synthesize "planar" circuits (planar circuit is one in which no wires cross).

- In both reversible and Quantum Logic it is not possible to have a fanout of a gate or a primary input larger than 1. On the other hand, reversible gates have a very high fanout, but of different output functions. This makes the synthesis problem quite different from classical logic, because if we want to have fanout on a "wire" higher than 1 we have to pay for it with special "fanout gates, and we want to use all output signals rather than waste them. Usually the Feynman gate is used to implement the fanout gate; it has its input B set to 0.
- The inputs can be set to constants in reversible gates. The outputs of gates can also be constants. Thus the constant wires can be laid out in a regular way (like a snake pattern) through the entire circuit with very small performance penalty.
- Feedback in gate is not allowed, which limits the number of ways a gate can be used in a circuit.

A trivial synthesis method in reversible logic is to build the multi-output function, using any known logic method, from classical gates CG_i and next replace every gate CG_i with a reversible gate that includes the function realized by CG_i as one of the reversible gate outputs. For instance, every circuit can be build from multiplexers, and next every multiplexer can be extended to Fredkin gate with outputs P and R possibly not used. But with this method a problem arises: "what to do with the remaining outputs of the reversible gates?". This method is practically nonsensical as a general synthesis procedure; it generates the "garbage" - a lot of wires that are congesting the layout without any use other than the reversibility requirement. This is a very bad approach to synthesis, especially for future technologies - remember the "curse of wiring" which will dominate future technologies, with most of chip area occupied by connections, unless cellular-like logic and layouts are used.

Thus, efforts in reversible logic design so far were mostly on designing practical reversible circuits, but there are some publications that attempt at creating general synthesis methods [9,11,17,18,31-34,39,43,45]. The weaknesses of these methods fall to one or more of the following categories:

- they assume that all gates are the same
- they assume cascades of gates or two-dimensional circuits based on cascades, which leads often to complex realizations with large garbage.
- they assume complex gates and set high percent of gate inputs to contacts and sink high percent of gate outputs to garbage.

Heuristics for smart method of synthesis with reversible gates are in our opinion the following:

- Do not create many outputs of gates and subcircuits,
- Re-use these outputs as inputs to other gates,
- Apply re-usability properties of these common sub-functions - we believe that symmetry introduced here is only one of such properties and we will find more of them,
- The method must be generally applicable,
- We believe also in the use of regularity and group/field/linear algebra properties that are so useful in binary logic.

Below we will illustrate one way how these heuristics can be used.

3. Fredkin Gates and their uses.

Fredkin gate has a very important place in history of Reversible and Quantum computing and is still the most often used 3-input, 3-output gate. Several generalizations of this gate have been also proposed, one of which will be the base of our design here.

3.1. Fredkin Gate

Fredkin Gate (FG) is a fundamental concept in reversible and quantum computing, the base of many both theoretical and "realization-related" papers. It was introduced by Ed Fredkin and Tomasso Toffoli in 1982 [17].

Fredkin Gate has been realized or proposed to be realized in various technologies:

- **optical:** [5], [37], [31], [33],
- **electrical** (CMOS): [5,6,,7,8,10,11,13,26],
- **mechanical** (nano-technology): [27].
- **quantum**: [38,42,30,53].

Fredkin gate together with Toffoli and Feynman gates belong to the most often discussed in reversible and quantum literature, so it is very probably that future realization efforts will concentrate on realizations based on these gates and their derivations.

3.2. Multi-Valued Fredkin Gate.

Multi-valued Fredkin Gate (MVFG) was introduced by Picton [33]. He showed how it can be used to build MIN/MAX based Sum-of-Product kind of circuits realizations in reversible logic. His designs are however often quite inefficient, especially when applied to quantum logic.

MVFG, see Figure 1, is described by equations:

P = A
Q = B
R = C if A < B else R = D
S = D if A < B else S = C

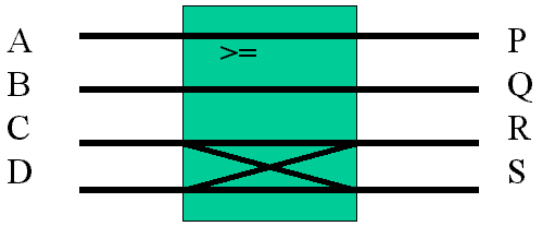


Figure 1. Multi-Valued Fredkin Gate

Observe, that the definition of the gate does not specify the type of signals. They can be thus binary, multi-valued, fuzzy, continuous or complex. The only requirement is that the relation of order (<) can be defined on them. Below, only the binary case of signals A, B, C and D is discussed. We introduced also other generalizations of Fredkin gate using multi-valued logic [55] that have additional advantages and are simpler. In fact, the whole families of such gates exist. It is interesting that a single reversible gate in binary logic has so many reversible generalizations in multiple-valued logic.

3.3. Use of Multi-valued Fredkin Gate to create MIN/MAX gate

Multi-Valued Fredkin gate can be used to create not only many interesting structures, but also many interesting gates of general use. In particular, as shown by Picton [33], the MIN/MAX gate can be built from two Multi-Valued Fredkin gates (see Figure 2). In our subsequent papers we will demonstrate various uses of Multi-Valued Fredkin gates, especially multi-level iterative and regular structures. Here we consider only the special case of MIN/MAX gate that can be build from two MV Fredkin gates. Observe that in case of binary logic for signals A, B, C and D, the MIN/MAX gate becomes the AND/OR gate. The reader is invited to analyze the behavior of the circuit from Figure 2 for A<B and for A>B in order to prove its correctness. This can be done by propagating the binary values through the circuit according to the equations of each MVFG.

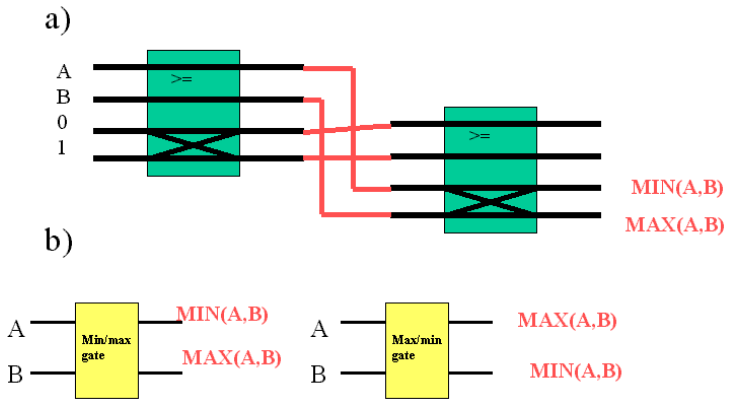


Figure 2. Realization of MIN/MAX gates: (a) Realization of MIN/MAX reversible gate from two MV Fredkin

gates, (b) schematics of MIN/MAX gate, (c) MAX/MIN gate

4. Regular structure of MAX/MIN and Feynman gates to realize an arbitrary symmetric function.

4.1. Preliminaries

Let our area of interest be Boolean expressions that are specified as sum-of-products. There is a subset of such expressions in which every variable is either negated or not negated, but not both.

Definition 1. The variable that stands non-negated (positive) throughout the expression is called a ***positive polarity variable***. Variable that stands always in negative (negated) form (as a negated literal) is called a ***negative polarity variable***.
Note: If function has ***n*** variables, and each variable is either positive or negative, there are 2^n different combinations of polarities of input variables, which are called ***polarities of the functional expression***. If all variables have positive polarity, the polarity of expression is called ***positive.*** If all variables have negative polarity, the polarity of expression is ***negative***. There are then 2^n various ***polarities of expressions***.

Definition 2. *Unate function* is a function expressed only using AND and OR operators (for instance Sum-of-Products) in which every variable has either positive or negative polarity, but not both.

Example 1: function ***f1 = ab + b c'*** is unate and has polarity: a=positive, b=positive, c=negative. In short, it has the polarity (a,b,c) = (1,1,0), when positive polarity of variable is denoted by 1 and negative polarity by 0. Function f2 = a b' + a' b is not unate. This is EXOR gate, and it is a linear gate. All functions and gates can be characterized as: unate, linear and other. *Linear function* is an EXOR operator of literals and constants. Function majority of three variables: f3 = ab + ac + bc is both symmetric and positive unate. Functions that are both positive unate and totally symmetric will be of our interest in this paper.

Definition 3. Totally Symmetric function that has value 1 when exactly ***k*** of its ***n*** inputs are equal one and exactly ***n-k*** remaining inputs are equal 0, is called a ***single-index symmetric function*** and denoted by $S^k(x_1, x_2, \ldots, x_n)$. Analogously, by $S^{\{i,j,k\}}$ we denote the function that is one when I, j, or k of its variables are equal one. Obviously, this notation can be extended to any number of indices, so every symmetric function can be written as $S^I(x_1, x_2, \ldots, x_n)$, where ***I*** is any subset of the set of indices $\{0,1,2,\ldots n\}$. It can be also easily checked that:

$$S^{I1}(x_1, x_2, \ldots, x_n)\ AND\ S^{I2}(x_1, x_2, \ldots, x_n) = S^{I1\ intersection\ I2}(x_1, x_2, \ldots, x_n) \quad (1)$$

$$S^{I1}(x_1, x_2, \ldots, x_n)\ OR\ S^{I2}(x_1, x_2, \ldots, x_n) = S^{I1\ union\ I2}(x_1, x_2, \ldots, x_n) \quad (2)$$

$$S^{I1}(x_1, x_2, \ldots, x_n)\ XOR\ S^{I2}(x_1, x_2, \ldots, x_n) = S^{I1\ symmetric\_difference\ I2}(x_1, x_2, \ldots, x_n) \quad (3)$$

$$S^i(x_1, x_2, \ldots, x_n)\ XOR\ S^j(x_1, x_2, \ldots, x_n) = S^i(x_1, x_2, \ldots, x_n)\ OR\ S^j(x_1, x_2, \ldots, x_n) = S^{i,j}(x_1, x_2, \ldots, x_n) \quad (4)$$

4.2. The basic idea of 2* 2 Net Structures

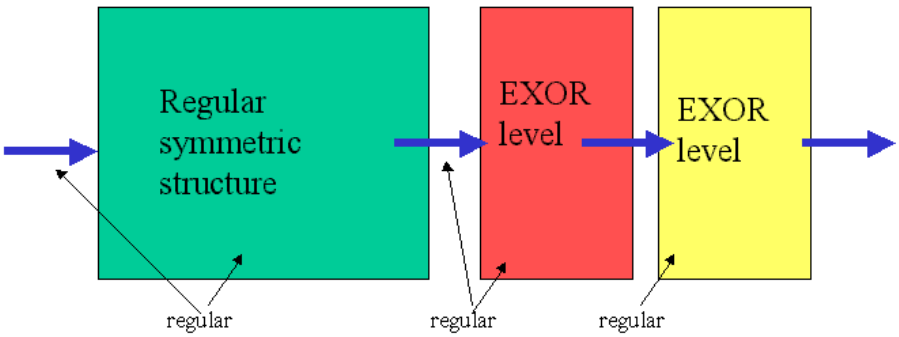


Figure 3. Three Plane Regular structure to realize arbitrary multi-input, multi-output Boolean function using MVL Fredkin gates

Figure 3 presents a regular structure based on three regular planes. The first plane from left is a levelized triangular structure in which the input variables correspond to the columns (we will call it also the ***triangular plane***). In contrast to Lattices, however, this structure, when realizing arbitrary multi-output function with geometrically adjacent output signals does not require variable repetition. (For some multi-output functions such as "counter of ones in a binary string" the Lattice without repeated variables required to push the output signals by few spaces apart. Or, these output signals were adjacent but the input variables had to be repeated. This was a disadvantage of Lattices for large multi-output symmetric functions, which was next attempted to I mprove in MOPS circuits [52]). The structure of the first plane is planar, regular and algorithmically created. It realizes all positive unate symmetric functions of its input variables. The second structure from left is just a pair of columns of Feynman gates that converts these positive unate symmetric functions to single-index symmetric functions for which arbitrary symmetric function can be easily created. Finally, the plane from the right is a plane of Feynman gates that uses their internal EXOR gates to realize every output function as an EXOR of single-index symmetric functions from plane two. (Formula **(4)** above is used). This plane can be compared in its functionality to the OR plane in a standard AND/OR PLA that is used to realize a Sum-of-Product (SOP, DNF) expression. Because the functions on the output of second plane are disjoint as single-index functions, the OR of them is the same as the EXOR of them (This is based on Boolean Law: A OR B = A EXOR B EXOR AB. Thus when functions A and B are disjoint, AB=0 and A OR B = A EXOR B.)

The whole idea is thus based on the well known fact that every symmetric function of variables $x_1, x_2, \ldots, x_n$ can be realized as OR or EXOR of its single-index symmetric functions $S^i(x_1, x_2, \ldots, x_n)$. Figure 4 illustrates how positive polarity unate symmetric functions can be created systematically in a regular planar arrangement of MAX/MIN modules. Observe that each output function, from top to bottom, includes the next function and are all positive polarity and unate. The sets of indices of the adjacent functions differ by one.

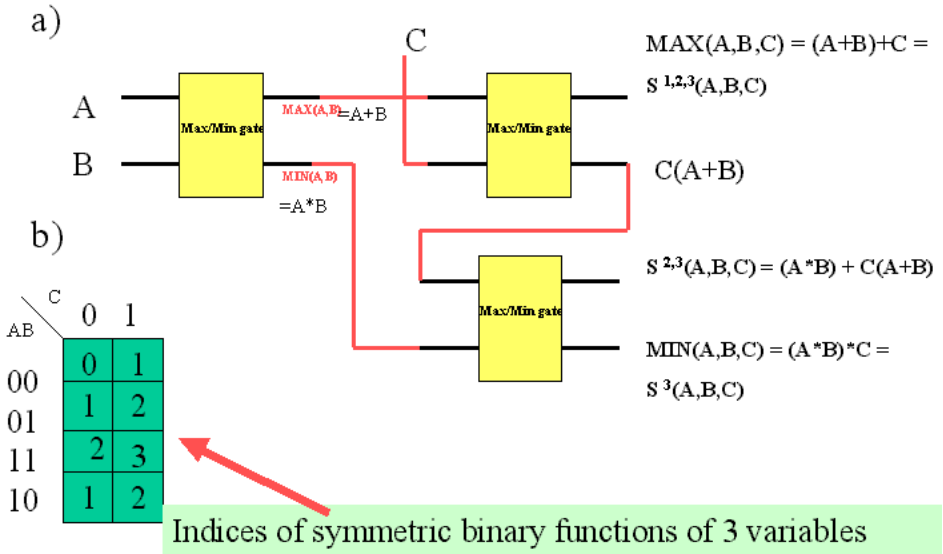
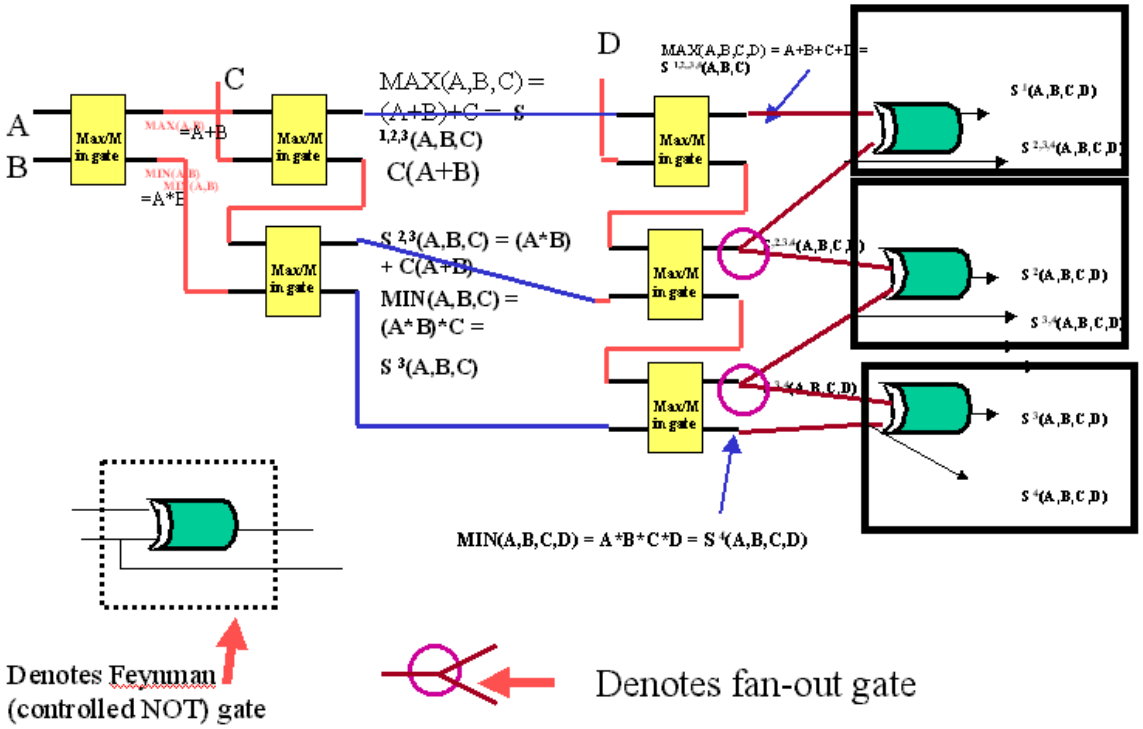


Figure 4. Example of realization of some symmetric functions of three variables in the left plane from Fig. 3: (a) regular structure from reversible MAX/MIN gates that realizes positive polarity unate symmetric functions, (b) indices of a symmetric function of variables A,B,C; each cell includes an index of a symmetric function corresponding to it, for instance, function $S^{2,3}(A,B,C)$ will have ones in cells



with indices 2 and 3 and zeros in cells with indices 0 and 1.

Figure 5. Realization of all single-index symmetric function using only reversible gates as EXORs

Let us observe that positive unate symmetric functions generated on the outputs of the triangular plane have a very nice property, the EXOR of the neighbor functions creates a single-index symmetric function. This is illustrated in Figure 5. However, because the EXOR gate is not reversible, we have to complete it to Feynman gate by repeating one of its inputs to the output. Because our structure is regular, this does not complicate the structure. In result, we obtain a structure such as shown in Figure 5 (this function has four variables for better illustration). As we see, in this regular structure, we obtain not only the single-index symmetric functions, but also some interval symmetric functions whose parameters are highly correlated to the neighboring single-index functions. We have also to add fan-out gates to the plane 2 (as illustrated in Figure 5).

4.3. Complexity of proposed realization

Theorem 1: Every positive unate symmetric function of n variables can be realized in
1+2+.. n-1 = n(n-1)/2 MAX/MIN gates

Proof. Every positive unate (symmetric) function of 2 variables can be realized in 1 gate.Every positive unate function of 3 variables can be realized in 1+2 gates. Every positive unate function of 4 variables can be realized in 1+2+3 gates, etc.

Theorem 2: Every single index totally symmetric function of n variables can be realized in
n(n-1)/2 MAX/MIN gates, n-2 fan-out gates and n-1 Feynman gates.

Theorem 3: Every single-output totally symmetric function of n variables can be realized with n(n-1)/2 MAX/MIN gates, n-2 fan-out gates, n-1 Feynman gates in 2nd plane and at most n-1 Feynman gates in the third plane.

Theorem 4: Every m-output totally symmetric function of n variables can be realized in n(n-1)/2 MAX/MIN gates, n-2 fan-out gates, n-1 Feynman gates in 2nd plane and at most m * (n-1) gates in the third plane.

Observe that these are upper bounds, since EXORing the single-index functions is a wasteful method of creating symmetric functions.

5. Conclusion.

Symmetric functions belong to the most difficult to be realized in reversible logic. Although our circuits may look excessive at the first glance to people unfamiliar with reversible logic, their comparisons with realizations obtained using other methods [6-12,17,31-33] illustrate the true advantages of using regular structures. The known

methods create truly excessive numbers of "garbage" outputs for larger benchmarks. Moreover, the circuits shown by us can be significantly further improved, we have no space here to present these improvements. The improvements are in the following directions: (1) for binary logic simpler cells for the first plane can be designed: they have AND, OR, A and A' outputs and use 4*4 binary Fredkin gates, or use AND, OR, and A' outputs, and use 3*3 Kerntopf gates [18]; (2) the circuit can be generalized to take into account more powerful definitions of symmetry (for instance, the definition of symmetric function with 2^n polarities), and also generalized to non-symmetric functions[52]; (3)Because the output functions are linear combinations of single-index functions, the middle plane can be totally eliminated and the number of Feynman gates in the output plane can be significantly reduced; (4) it can be generalized for multi-valued logic; (5) General-Purpose regular structures for asymptotically garbageless realizations of arbitrary functions have been designed; (6) Reversible Logic FPGAs (RLFPGAs) have been conceptually designed, in which these structures, among many others, can be embedded; (7) the methods can be extended to Quantum Logic.

Literature

[1] W.C. Athas & L."J." Svensson , "Reversible Logic Issues in Adiabatic CMOS", Exploratory Design Group, University of Southern California - Information Sciences Institute, Marina del Rey, CA 90292-6695, {athas,svensson}@isi.edu

[2] Ch.H. Bennett , "Notes on the History of Reversible Computation", *IBM J. Res. Develop.,* Vol. 32, 1988, pp. 16-23.

[3] Ch. H. Bennett and R. Landauer, "The Fundamental Limits of Computation", *Scientific American,* July 1985, pp. 38-46.

[4] Bennett, C., "Logical reversibility of computation", *I.B.M. Journal of Research and Development*, **17** (1973), pp. 525-532.

[5] R. Cuykendall, and D. McMillin, "Control-Specific Optical Fredkin Circuits", *Applied Optics,* **26**, pp. 1959-1963, 1987.

[6] A. De Vos, "Proposal for an Implementation of Reversible Gates in c-MOS", *Int. Journal of Electronics*, Vol. 76, 1994, pp. 293-302.

[7] A. De Vos, "Reversible Computing in c-MOS", *Proc. Advanced Training Course on Mixed Design of VLSI Circuits,* 1994, pp. 36-41.

[8] A. De Vos, "A 12-Transistor c-MOS Building-Block for Reversible Computers*", Int. Journal of Electronics,* Vol. 79, 1995, pp. 171-182.

[9] A. De Vos, "Reversible and Endoreversible Computing", *Int. Journal of Theor. Phys.,* Vol. 34, 1995, pp. 2251-2266.

[10] De Vos, A.: "Introduction to r-MOS systems"; *Proc. 4 th Workshop on Physics and Computation,* Boston, 1996, pp. 92-96.

[11] De Vos, A.: "Towards reversible digital computers"; *Proc. European Conference on Circuit Theory and Design,* Budapest, 1997, pp. 923-931.

[12] De Vos, A.: "Reversible computing"; *Progress in Quantum Electronics*, **23** (1999), pp. 1-49.

[13] B. Desoete, A. De Vos, M. Sibinski, T. Widerski, "Feynman's Reversible Logic Gates Implemented in Silicon", *Proc. 6 th Intern. Conf. MIXDES*, 1999, pp. 497-502.

[14] R.Feynman, "Quantum Mechanical Computers", *Optics News,* **11** (1985), pp. 11-20.

[15] R. Feynman, "There's plenty of space at the bottom: an invitation To Enter a New Field of Physics," *Nanotechnology,* Ed BC Crandal and J.Lewis, the MIT Press 1992, pp. 347-363

[16] Feynman, R.: "Feynman lectures on computation" (A. Hey and R. Allen, eds); *Addison-Wesley,* Reading (1996).

[17] E. Fredkin, T. Toffoli, "Conservative Logic", *Int. Journal of Theor. Phys.,* **21** (1982), pp. 219-253.

[18] P. Kerntopf, "Logic Synthesis Using Reversible Gates," *Proc. 3rd Symposium on Logic, Design and Learning,* Portland, Oregon, May 31, 2000. P. Kerntopf, "A Comparison of Logical Efficiency of Reversible and Conventional Gates," *9th IEEE Workshop on Logic Synthesis*, P. Kerntopf, "On Efficiency of Reversible Logic (3,3) – Gates. *" Proc. 7th Intl. Conf. MIXDES, 2000*, pp. 185-190.

[19] R. Keyes, and R. Landauer, "Minimal energy dissipation in logic"; *I.B.M. Journal of Research and Development,* **14** (1970), pp. 153-157.

[20] R. Landauer, "Irreversibility and heat generation in the computational process"; *I.B.M. Journal of Research and Development,* **5** (1961), pp. 183-191.

[21] J. Lim, D. Kim, and S. Chae, " Reversible Energy Recovery Logic Circuits and Its

8-Phase Clocked Power Generator for Ultra-Low-Power Applications," *IEICE Trans. Electron,* OL.E82 -C, No. 4 April 1999.
[23] N. Margolus, "Physics and Computation", *Ph. D. Thesis*, Massachussets Institute of Technology, USA 1988.
[24] L.J. Micheel, A.H. Taddiken and A.C. Seabaugh, "Multiple-Valued Logic Computation Using Micro- and Nanoelectronic Devices," *Proc. ISMVL,* IEEE, 1993, pp. 164-169
[25] R C. Merkle and K. Eric Drexler; "Helical Logic", WWW.
[26] R C. Merkle, "Reversible electronic logic using switches," *Nanotechnology,* **4** (1993) pp. 21-40
[27] R. C. Merkle, "Two types of mechanical reversible logic," *Nanotechnology,* **4** (1993), pp. 114-131.
[28] O.N. Muzychenko, "Uniform and Regular Structures for Realization of Symmetric Functions of the Algebra of Logic", Automation and Remote Control, Vol. 59, No.4, 1998, pp. 581-592
[29] Y.N. Patt , "A Complex Logic Module for the Synthesis of Combinational Switching Circuits", *Proc. 30 th AFIPS Spring Joint Computer Conf.,* 1967, pp. 699-706.
[30] A. Peres, "Reversible Logic and Quantum Computers", *Physical Review A*, **32** (1985), pp. 3266-3276.
[31] P. Picton, "Optoelectronic, Multivalued, Conservative Logic", *Int. Journal of Optical Computing,* Vol. 2, 1991, pp. 19-29.
[32] P. Picton, "Multi-valued Sequential Logic Design Using Fredkin Gates", *MVL Journal,* vol.1, 1996, pp.241-251.
[33], P. Picton, "A Universal Architecture for Multiple-valued Reversible Logic," *MVL Journal,* **5** (2000), pp.27-37.
[34] P. Picton, "Modified Fredkin Gates in Logic Design," *Microelectronics Journal*, **25** (1994), pp. 437-441.
[35] M.R. Rayner, D.J. Newton, "On the Symmetry of Logic ", *Journal of Physics A: Mathematical and General,* **28** (1995), pp. 5623-5631.
[36] G. Stix, "Riding the back of electrons"; *Scientific American*, 279 (September 1998), pp. 20-21.
[37] J. Shamir, H. J. Caulfield, W. Micelli, W., and R.I. Seymour, "Optical Computing and the Fredkin Gates", *Applied Optics,* **25,** pp. 1604-1607, 1986.
[38] J.A. Smolin, and D.P. DiVincenzo, "Five Two-Bit Quantum Gates are sufficient to Implement the Quantum Fredkin Gate", *Physical Review A*, 53, pp. 2855-2856.
[39] L. Storme, A. De Vos, G. Jacobs, "Group Theoretical Aspects of Reversible Logic Gates", *Journal of Universal Computer Science,* Vol. 5, 1999, pp. 307-321.
[40] T. Toffoli, "Reversible Computing", in Automata, Languages and Programming, *Springer Verlag,* 1980, pp. 632- 644.
[41] P. Wayner, " Silicon in Reverse", *Byte Magazine*, August 1994, page 67.
[42] A workshop on the Physics of Computation was held at MIT in 1981; the papers were printed in the April, June and December issues of the 1982 International Journal for Theoretical Physics, Volume 21.
[43] V. I. Varshavsky, "Logic Design and Quantum Challenge". *Preprint from the author*.
[44] J. Birnbaum, "Computing Alternatives", Talk given on March 3, 1997 atACM97, March 3, 1997, San Jose, California, by Director of Hewlett-Packard Laboratories, Senior Vice-President of Research and Development.
[45] M. Frank, "Physical Limits of Computing," CIS 4930.1194X / 6930.1078X, Spr. '00.
http://www.cise.ufl.edu/~mpf/course.html
[46] A. Sarabi, N. Song, M. Chrzanowska-Jeske, M. A. Perkowski, "A Comprehensive Approach to Logic Synthesis and Physical Design for Two-Dimensional Logic Arrays," *Proc. DAC'94,* San Diego, June 1994, pp. 321 - 326.
[47] M. A. Perkowski, "A Fundamental Theorem for EXOR Circuits," *Proc. of IFIP W.G. 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design,"* Hamburg, Germany, September 16-17, pp. 52 - 60, 1993.
[48] M. Perkowski, B. Falkowski, M. Chrzanowska-Jeske, and R. Drechlser, ``Efficient Algorithms for Creation of Linearly-Independent Decision Diagrams and their Mapping to Regular Layouts". *VLSI Design*. In print
[49] Z. Kohavi, "Switching Functions and Finite Automata Theory, *Prentice Hall.*
[50] E. Pierzchala, M. A. Perkowski, S. Grygiel, "A Field Programmable Analog Arrray for Continuous, Fuzzy and Multi-Valued Logic Applications," *Proc. ISMVL'94,* pp. 148 - 155, Boston, MA, May 25-27, 1994.
[51] M. Chrzanowska-Jeske, Y. Xu, and M. Perkowski, ``Logic Synthesis for a Regular Layout," VLSI Design, Vol. 10, No. 1, pp. 35 - 55, 1999.
[52] M.A. Perkowski, M. Chrzanowska-Jeske, and Y. Xu, ``Multi-Level Programmable Arrays for Sub-Micron Technology based on Symmetries," *Proc. ICCIMA'98 Conference,* pp. 707-720, February 1998, Australia, published by *World Scientific.*

[53] H.F. Chau and F.Wilczek, “Realization of the Fredkin gate using a series of one- and two-body operators”, Report IASSNS-HEP-95/15, (quant-ph/9503005).
[54] Dallas Morning News, 19 April 1993.
[55] M.A. Perkowski. A. Al-Rabadi, P. Kerntopf, M. Chrzanowska-Jeske and A.Mishchenko, “Three-Dimensional Realization of Multi-Valued Symmetric Functions using Reversible Logic”. To be submitted.