METHOD 4. Composition method, from inputs to outputs of an inverse function. As method 3, but METHOD 2 is used after completion to a reversible function and finding the inverse function.

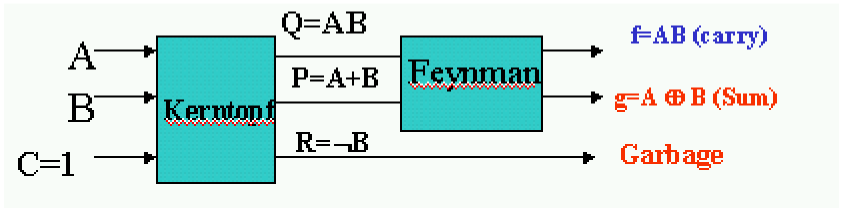
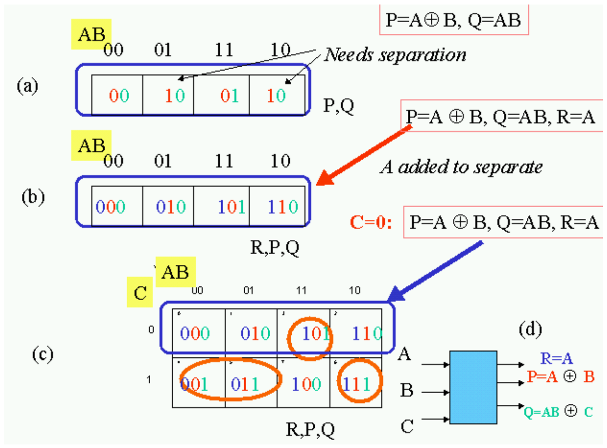


Figure 13. Illustration to METHOD 1 and METHOD 2. Realization of half-adder using Kerntopf and Feynman gates.

Figure 14. Illustration of converting a non-reversible function <P(A,B), Q(A,B)> to a reversible function <R(A,B,C), P(A,B,C), Q(A,B,C)>; (a) original function that needs separation of cells, (b) function after separation done by adding variable A as an output, (c) fully reversible function of A, B, and new variable C, (d) symbol of reversible function



Example 7. A Full adder composed of two half adders is presented in Figure 15. ***C-in = C*** is the carry-in and ***C-out*** is the carry-out. It can be observed in a Kmap of variables ***A,B,C*** that two garbage functions are the minimum number to separate all repeated values of outputs Sum and C-out, because there are for instance 3 occurrences of combination 10. Thus, a total of outputs is 2 +2 = 4. Since there are three inputs, at least one constant input is necessary. Therefore, the circuit from Figure 15 is optimal in a sense of the minimum number of additional inputs and outputs. This kind of analysis of separations with minimal number of functions is fundamental to reversible logic design and leads to some new combinatorial optimization problems.

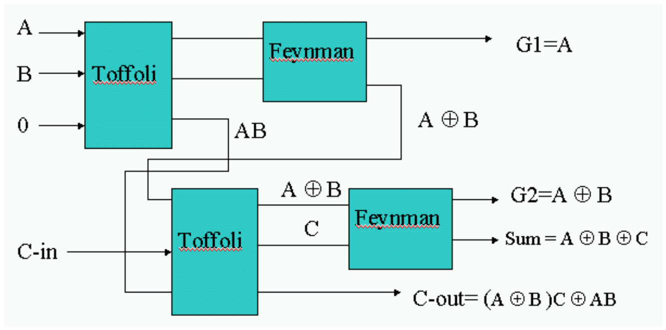


Figure 15. A full adder composed from two half adders (the version with Toffoli's gate). Observe two garbage outputs.

7. A Levelized Variable Decomposition Method

Sometimes it is difficult for a composer/decomposer from previous sections to find a good choice of input or output gate. In such case some kind of brute-force method is needed to simplify the synthesis problem, at the cost of introducing more constant inputs and garbage outputs. This method, presented in this section, creates, however, always simpler subfunctions, usually ones that have a smaller number of input variables. For instance, this method can be used for a 4-variable function by decomposing it to two 3-variable reversible functions that can be next synthesized by NPN matching to library cells. The method described in this section, called levelized variable decomposition, is based on basic expansions and recursive ideas of binary logic synthesis. It is used in conjunction with other methods.

The fundamental role of Shannon Expansion in binary logic is well known. Shannon Expansion is used in recursive algorithms, Binary Decision Diagrams and minimization of SOP expressions. Shannon Expansion corresponds to a multiplexer gate. In reversible logic, an equivalent of multiplexer is the Fredkin gate. In this paper an ***equivalent of Shannon Expansion for reversible logic*** is investigated. The standard Shannon Expansion, which we will call ***Forward Shannon Expansion*** is used to create decision diagrams and thus also to design multilevel logic structures that use multiplexers, in particular multiplexers with two data inputs and one control input. Let us observe that this expansion can be treated as a way to determine the two functions on data inputs of a multiplexer, knowing the output function $f(x_1, x_2, \ldots x_i, \ldots, x_n)$ of the multiplexer and the selected control variable x_i connected to the control input of the multiplexer. Shannon Expansion can be visualized by creating two Kmaps of the data functions being the ***positive cofactor*** $f(x_1, x_2, \ldots x_i = 1, \ldots, x_n)$ and the ***negative cofactor*** $f(x_1, x_2, \ldots x_i = 0, \ldots, x_n)$ of function $f(x_1, x_2, \ldots, x_n)$, respectively, from the Kmap of this function and a choice of variable x_i . The Kmap of the positive cofactor is created by rewriting the Kmap of f for its half where $x_i = 1$ and filling all cells with don't cares for its half where $x_i = 0$. The map of the negative cofactor is created by rewriting the map of f for $x_i = 0$ and filling all cells with don't cares for $x_i = 1$. This graphical method can be easily extended to MV logic; for instance for the ternary logic the Kmap is splitted into three Kmaps; for $x_i = 0$, for $x_i = 1$, and for $x_i = 2$, respectively. (Each of these maps has one-third cares and two-thirds don't cares).

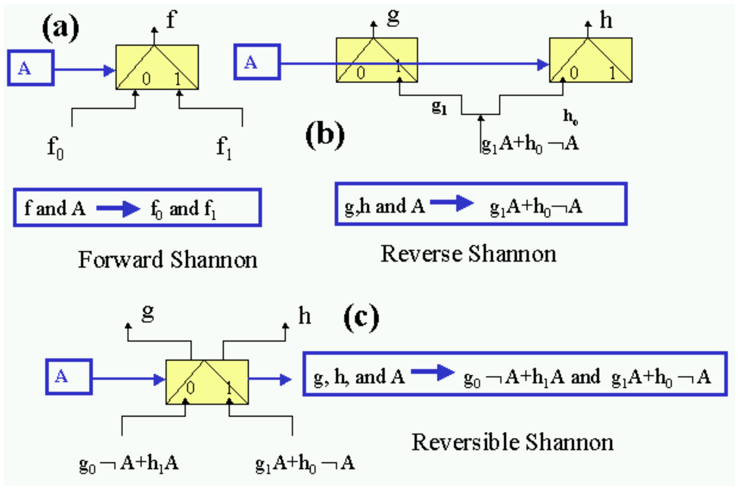


*Figure 16. Comparison of expansions Expansion for binary logic and standard 3*3 Fredkin gate, (a) Forward Shannon, (b) Reverse Shannon, (c) Reversible Shannon.*

In [50,51,52] we introduced the concept of ***Reverse Shannon Expansion***. While in binary logic the Forward Shannon Expansion splits the Kmap into two Kmaps, the Reverse Shannon Expansion does the opposite – it joins two Kmaps into a single Kmap. When one of the corresponding cells is a care and another a don't care, the resultant value is always that of a care. In case of two don't cares the result will be the don't care. Because in our methodology the Reverse Shannon Expansion is applied always to two Kmaps that have disjoint sets of cares, it never leads to a conflict. Figure 16 compares the Forward, Reverse and ***Reversible Expansions*** for binary logic and ***3 * 3*** gates (the arrows are related to the flow of information in the circuit, for the purpose of illustrating functional expansions the direction of arrows should be reversed). While Forward Shannon is used to create BDDs, both the Forward and Reverse Shannon expansions are used to create classical Lattice Diagrams [53]. Interestingly, only the ***Reversible Shannon Expansion*** is necessary to create ***Reversible Shannon Decision Diagrams*** introduced below. Observe that the combined operations of splitting and joining cofactors in the Reversible Shannon Expansion can be also called ***"permutting"*** or ***"shuffling".*** Treating the Reversible Shannon Expansion as the ***cofactor permuter*** is a useful heuristic to create algorithms.

The Levelized Variable Decomposition for ***k*k*** Fredkin Gate is performed as follows:

1. select any ***expansion variable*** (a primary input) and select the number ***k-1*** of arbitrary data signals (by data signals we understand output functions or intermediate functions, but not primary inputs).
2. calculate positive and negative cofactors for each of data functions, as in the Shannon expansion from left to right with respect to the selected control variable.
3. join the cofactors shifting them by one, with wrapping around.

For ***3*3*** binary Fredkin Gate the rules from points 2 and 3 above simplify. The shift becomes just a flipping of two pairs of cofactors, because there are only four cofactors (see the formalism and countings for more general families of gates in [1]). The Reversible Shannon Expansion for standard Fredkin gate is shown in Figure 16 and the procedure of repetitive using this expansion for 3*3 Fredkin gates is illustrated in Figure 17. When the Reversible Shannon Expansion is used repeatedly for subsequent input variables as controls in multiplexers, a Reversible Shannon Decision Diagram ***(ORSDD)*** is created. When the variables are ordered, this diagram is called Ordered RSDD. Otherwise, it is a free diagram, ***FRSDD***. Observe, that the ORSDD is not canonical, because arbitrary signals can be selected and permuted for any expansion. Thus, a general purpose ORSDD and FRSDD are useful for synthesis but not for representation. When one imposes, however, some order of selecting signals for permuting, for instance by ordering output functions and intermediate signals, the ORSDD can become canonical, but such diagrams will not be considered below. For instance, the Lattice RSDDs introduced below are not canonical, because incompletely specified garbage functions are used in their creation. However, one has to keep in mind, that only a small number of possible types of RSDD diagrams are presented here.

Example 8. Figures 17 and 18 illustrate the ***systematic procedure*** of using the binary 3*3 Reversible Shannon Expansion to create a ***Reversible Fredkin Lattice Diagram*** for a *single-output* function. Similar procedure exists for multi-output functions, in which all output functions are ordered from left to right. When one creates an RSDD, a method to select data output signals for permutting is needed. For example, these can be the most similar signals, which means those, that corresponding functions on primary input variables differ in the smallest number of minterms. This is a generalization of standard BDDs, in which only *identical function* nodes are combined (as isomorphic nodes correspond to tautological functions). The principle of selecting pairs of next level data signals for RSE can be also based on some other property. For instance, symmetry, in order to reduce the next level functions to simpler functions (As examples, one can select symmetric, unate, reversible, or simple functions such as for instance variables). Finally, the pair of signals to be selected can come from some geometrical neighborhood, as in standard lattices and their recent generalizations. Below the simplest method of selecting signals from a neighborhood is discussed, the same as in standard Lattices. (Similar rules exist for selecting larger groups of data signals for k>3).

Figure 17 illustrates the use of garbage function ***g*** in the first expansion from top. From Kmaps of the primary output function ***f*** and garbage function ***g*** as data signals, two new functions***, fg,*** and ***gf,*** represented by the two lowest two Kmaps, are created. Observe the permutation of cofactors that occurred in Kmaps. To fully understand the method, the careful reader should analyze concurrently Figures 16, 17 and 18. The latter shows only functions realized by signals in respective points of the reversible lattice diagram.

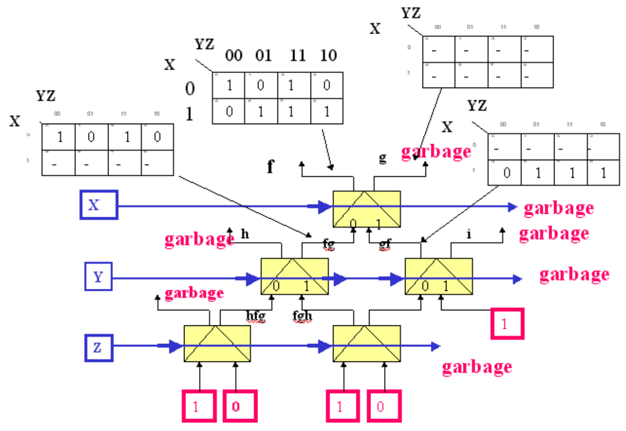


Figure 17. Using the Levelized Variable Decomposition Method for Realization of function F in a binary Reversible Fredkin Lattice Diagram. Observe the constants in inputs at the bottom and garbage outputs

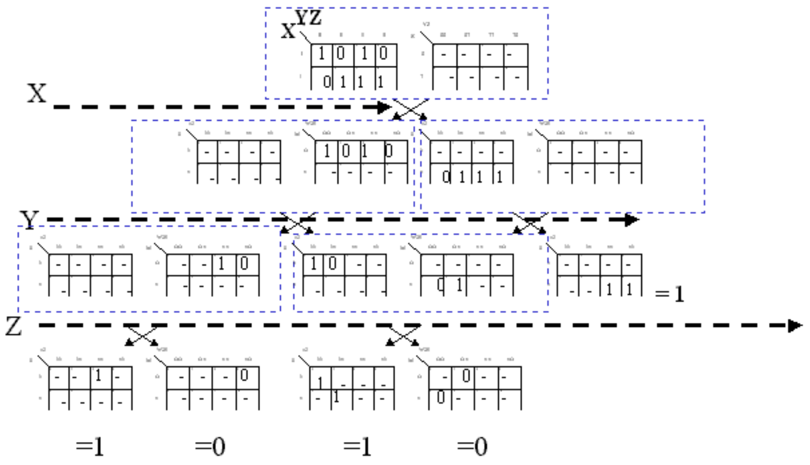


Figure 18. All stages of level-by-level application of Reversible Shannon Expansion to function f. The leaf with no ones is converted to 0 and the leaf with no zeros to 1.

The procedure is as follows:

- We start from the function ***f(X,Y,Z).*** Because a 3*3 gate is selected, and one of its inputs and one of its outputs is the selected input variable ***X***, we have to perform the 3*3 Reversible Shannon Expansion with 2 argument functions, as shown in Figure 16c. Thus, one more garbage function ***g*** is allocated to the expansion.
- Reversible Shannon expansion for the pair of data functions ***(f,g)*** and control variable ***X*** is executed which leads to new functions ***fg*** and ***gf*** of the lower level. Analysis of example of this expansion helps to understand what we mean by "permuting cofactors". The upper cofactor (for $\neg X$) from the left upper map goes to the upper part of the lower left map, and the lower cofactor of the right upper map goes to the lower part of the left lower map. Similarly, the upper cofactor of the upper right map goes to the upper part of the lower right map and the lower cofactor of the upper left map goes to the lower part of the right lower map (see Figure 18).
- Garbage function ***h*** is added to make a pair ***(h, fg)*** for next Reversible Shannon Expansion, and one more garbage function ***i*** is added to create a pair of data functions ***(gf, i).***
- The next level expansion variable ***Y*** is selected (randomly, based on a heuristic, or using look-ahead strategy).
- Expansions with respect to this variable are executed taking the argument data functions from left to right, and so on.

As a result, by selecting at every level such data signals (inputs of gates) as to create the given type of lattice structure (in this case from left to right, omitting constants), the Reversible Fredkin Lattice from Figure 17 has been obtained. There remains of course the problem in what order to select the variables at the successive levels of the reversible lattice – this problem is known from BDDs and lattice diagrams (we do not have yet any solution better than those proposed in [12,13,53]). If at the bottom of the lattice one obtains linear functions, it is cheaper to realize them using 2*2 Feynman rather than Fredkin gates. Look again at Figure 18 that illustrates all Kmaps created during the successive Reversible Shannon Expansions. When certain input is a constant, the expansion process is locally terminated. The total number of primary output signals is 8 and the minimum number of outputs for 3 inputs in reversible logic is 3, so the additional garbage is 8 - 3 = 5. Of course, assuming other reversible gates than Fredkin, a better result can be found.

It is known from the research on standard binary lattices [12,13,50,51,52,53] that arbitrary symmetric function can be realized in a lattice without repeated variables. It is also known that an arbitrary (non-symmetric) function can be realized in a lattice with ***repeated variables*** (using so-called ***symmetrization***). Similar property exists for the presented method. This method terminates for arbitrary function, assuming that the variables are repeated in levels. Thus, if the leafs of the lattice are not all constants after expanding for all input variables, some of these variables are used again in new levels of expansions, which we call *"variable repetition"*. Interestingly, the functions that do not require variable repetition in the Reversible Shannon Lattices ***are not necessarily the symmetric functions***. The characterization of the functions realizable in these structures without repetitions and the respective exact synthesis algorithms are interesting open problems.

The levelized algorithm can be used as part of another decomposition method, and it can map to any selected regular or non-regular structure, not only lattice. Finally, the Reversible Shannon Expansion can be used on equal terms with any other output gate decomposition rule in output decomposition.

Example 9. We will design a Toffoli gate from Fredkin gate using output decomposition. This is a design of a reversible function, so the synthesis without garbage is possible based on results in [71]. We synthesize from outputs using Feynman gate applied to functions $c \oplus ab$ and b. This creates new functions $c \oplus ab$ and $c \oplus ab \oplus b = c \oplus \neg a b$. Now, because function ***a*** is an output of Toffoli gate, RSE expansion with respect to variable ***a*** is selected in order not to create a garbage output of its through variable (in this case, ***a***.). This creates new functions $x = b \oplus c$ and $y = c$. After applying Feynman again to ***y*** and ***x***, the primary inputs ***c*** and ***b*** are created. All created signals are not duplicated so the procedure is complete, see Figure 19.

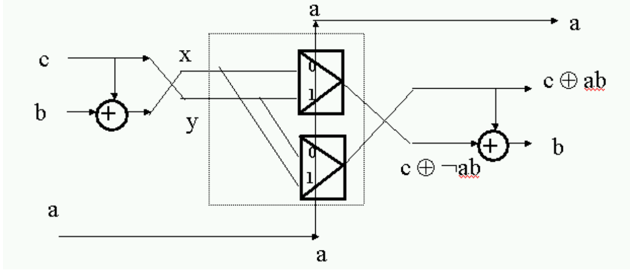


Figure 19. Illustration of using Reversible Shannon Expansion (RSE) in a general structure. Synthesis is done from outputs to inputs and the expansion is applied in the second step.

The general levelized method can assume ***any structure of the layout***, thus any order and choice of input signals of successive Reversible Shannon expansions. Assuming other type of structure, ***cascade*** or ***non-planar lattice with intersecting signals***, this other type of structure would be created if sufficient number of garbage outputs is assumed. For arbitrary structures, however, the method requires small modification: if the structure is too constrained (by assuming no garbage added), the structural equations may have no solutions or the algorithm loops. This happens, for instance, when a Maitra–Cascade-like structure is assumed for a function that is not Maitra-realizable. It happens also when we assume a levelized circuit of a too small width). Thus the algorithm must be modified to deal with these special cases. The methods like this can be applied to realize the cascades [7,25,38,39,40,41,67,71]. Finally, our general approach will work also for irregular structures. In such case, any pair of signals can be the inputs to the Reversible Shannon Expansion, regardless of their order. The signals are paired in a way to obtain the smallest total complexity for the level. We analyzed also various planar and non-planar regular structures to realize arbitrary functions in binary and multiple-valued reversible and quantum logic. Interestingly, because in quantum realization every classical reversible logic gate can be composed of Feynman gates and 1*1 unitary quantum gates, all quantum circuits can be built from regular 2*2 structures, since 1*1 gates, similarly to inverters in classical regular structure, do not change the regularity patterns and can be inserted anywhere "for free". The 2*2 regular structures are easier to analyze and synthesize than the 3*3 structures.

8. Reversible Curtis Decomposition

Curtis Decomposition Mode is an adaptation of the well-known Ashenhurst/Curtis decompositions of Boolean functions. However, because of the properties of reversible logic, these decompositions have been modified by us. For instance, standard Curtis decomposition assumes that the number of outputs from the predecessor block ***G*** is smaller than the number of its inputs. In our case, following [28], the number of outputs and inputs in a block ***G*** can be the same. Curtis decompositions are illustrated in Figure 20.

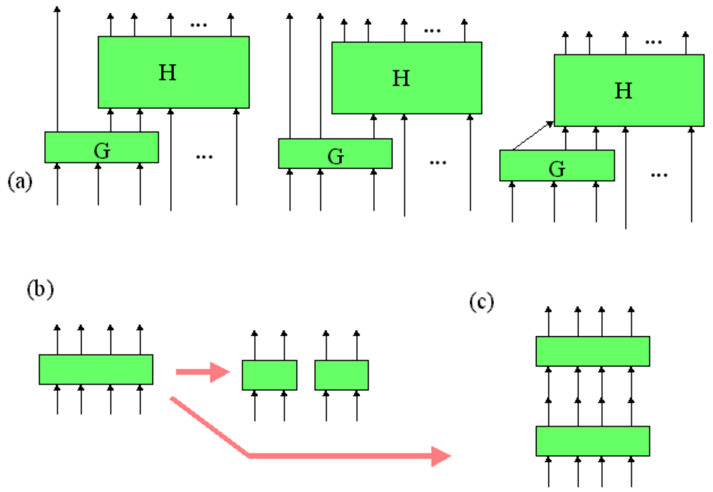


*Figure 20. Ashenhurst/Curtis Decompositions modified to reversible logic: (a) Decompositions with arbitrary reversible 3*3 gate as block G, (b) Parallel Decomposition to reversible blocks, (c) Serial Decomposition to reversible blocks*

Let us discuss, for instance, the Curtis-like reversible decomposition from the left scheme in Figure 20a. The bound set are input variables of a reversible block ***G.*** The decomposition of this structure exists when two conditions are concurrently satisfied: (1) the incompatibility graph created for the bound set of input variables [28] has $2^2 = 4$ colors, and (2) every color corresponds to exactly two triplets of bound variable values in the Kmap of the block ***G***. This formulation leads to a new combinational problem of balanced and minimal graph coloring that generalizes the graph coloring used traditionally in functional decomposition [28]. If the above condition is satisfied, then block ***G*** is a 3*3 reversible function with its left output being a garbage, selected in a way to separate all pairs of cells of the Kmap of block ***G.*** (This garbage function can be next re-used in composition mode of synthesis). In addition, the reversible function ***G*** depends on the binary encoding of colors A, B, C and D from graph coloring with pairs of bits, which affects the realization cost of block G using cascade of 3*3 gates. In any case, this function is realizable without garbage if the respective cells exist in our cell library. As we have seen, the main design task of decomposition is to decompose to such functions that at least one of them (in our case, ***G***) is reversible – this is the general principle of reversible Curtis and other functional decompositions.

9. MP-Decomposition Algorithm

1. *Create a PKDD or PKDDCE of the multi-output function.*
2. *Map it to Toffoli, Fredkin ,and other gates and inverters as in section 3.*

3. *Based on groups of gates that have small garbage outputs (as presented in section 3), find natural partitioning to smaller blocks.*
4. *For each block separately apply each of the decomposition types in both directions and select the best solution. The algorithm is recursive, so at every level any of the types and modes can be called to find the solution. The algorithm creates new subfunctions performing search either forwards or backwards and using the NPN matching to standard library of reversible cells. The search is greedy, at any stage the function evaluated as having the smallest cost function is realized. If a solution with predicted cost or smaller than the cost of previous solution cannot be found, the program backtracks.*

This procedure is non-deterministic. It creates in certain order new subfunctions and can remove the functions considered to be bad choices. Its software realization requires efficient problem representation, applications of information theory measures [31,32,33] to select best subfunction candidates in look-ahead searches, additional heuristics for special functions and technology matching, and the so-called AI method of "intelligent non-chronological backtracking" based on costs and constraints.

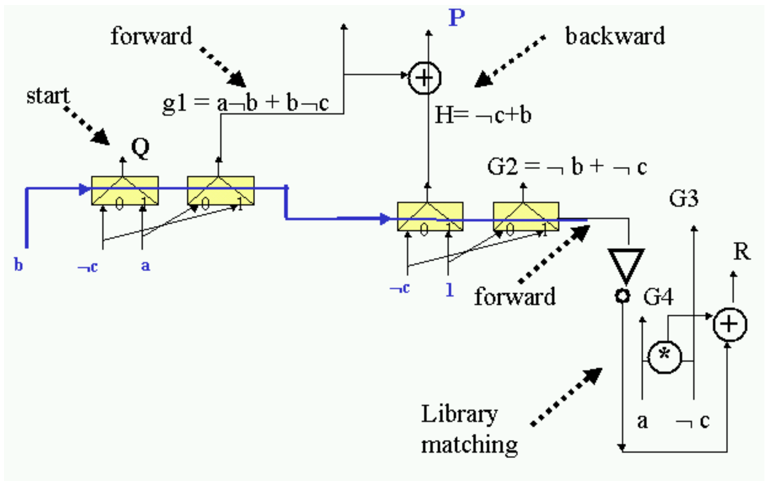


Figure 21. Illustration of cooperation of various methods to design the Kerntopf gate from Fredkin, Toffoli, Feynman gates and inverters. One fan-out gate for input a, an inverter for c and two fan-out gates for ¬ c are not shown.

Example 10. *Forward versus backward search.* We will design a Kerntopf gate from Fredkin, Toffoli, Feynman gates and inverters. Because of NPN-equivalence-based similarity of function ***P*** of Kerntopf gate and mux output of a Fredkin gate, the algorithm starts from output ***Q***, realizing it with one Fredkin gate. As byproducts, subfunctions ***b*** and $g1 = a \neg b + b \neg c$ are created on other outputs of this Fredkin gate (see Figure 21, left upper corner). This is a forward direction of search. Assuming Feyman gate with output ***P***, new signal $H = \neg c + b$ is generated in backward direction. It is realized by next Fredkin gate with inputs $\neg c$ and ***1***, and byproduct outputs ***b*** and $G2 = \neg b + \neg c$. Propagating forwards, and using library matching, output function ***R*** is realized with garbage ***G3*** and ***G4.*** Next the circuit is completed by adding fan-out gate for signal ***a*** and two fan-out gates for signal $\neg c$. Observe, that ***G3*** and ***G4*** cannot be used as source of signals ***a*** and $\neg c$, since it would lead to a circuit with a loop, which is not allowed for reversible logic. Thus realization of Kerntopf gate required four constant inputs and four garbage outputs.

10. Conclusions and Future Work

The paper introduces the concept of multi purpose decomposition of reversible logic that includes several types and modes. It is applicable to multi-output binary functions and it has been hand-simulated on several small functions from [70] and functions from other papers on reversible logic. The program IRMA2FPGAS can be run with no modification as its input mode [37]; in two ways, from inputs to outputs and from outputs to inputs after finding an inreverse function in a preprocessing stage. This program can be further improved by better adjusting its evaluation functions to reversible logic properties. The generalization of presented method to multiple-valued logic is relatively simple and it follows the lines of generalizing the Curtis and bi-decompositions to multiple-valued logic [49]. New reversible MV gates (other than Picton's gates and gates from [1]) can be realized using techniques that adapt methods from [15,62]. Not presented here are the methods for synthesis of reversible logic to generalized Maitra cascades [46] and Tandem networks of Butler [6,7,8,9], as well as bi-decomposition [48,49] and new decompositions which we plan also to incorporate into the decomposer. Both modes and all types of our decomposition, as well as the preprocessing stage, have been also formulated by us for multiple-valued logic.

Similarly, the software will be written for multiple-valued logic; for instance, a separate BDD is used for every logic value of any function (Using one BDD for one value of a function allows also for the representation of incompletely specified functions and multiple-valued relations as the input data to our decomposer [39]). However, for simplification, only the binary case was presented in this paper. Some aspects of MV extensions can be found in [1,41,54,55]. Finally, we found strong links of all these methods to realizations of quantum logic circuits from "gates" [16]. Of course, creating theories of designing quantum circuits with many gates may seem premature when only few gates are possible in recent quantum computers. We believe, however, that because of recent breakthroughs in technology, quantum computers will be built in coming 30 years. Observe that in the history of classical computing the results of logic synthesis theory were not immediately applicable for designing computers, but ultimately were found very useful to develop next generation EDA tools. We work on reversible logic synthesis theory early enough to be ready with complete theory and efficient tools when their time will come.

The authors acknowledge help of Professor Tsutomu Sasao from Kyushu University of Technology and Dr. Antoni Michalski from Informatics Department of Finance Ministry of Poland.

Literature

[1] A. Al-Rabadi, M. Perkowski, " New Classes of Invariant Multi-Valued Spectral Transforms for Three-Dimensional Layout Realization of Reversible Logic," *Proc. RM'2001 Workshop.*
[2] A. Al-Rabadi, M. Perkowski, "Shannon Families of Lattice Structures for Logic Synthesis in Three-Dimensional Space," *Proc. RM'2001 Workshop.*
[3] A. Al-Rabadi, M. Perkowski, "New Multi-valued Reed-Muller-based Families of Spectral Transforms**,"** *Proc. RM'2001 Workshop.*
[4] C. Bennett, "Logical reversibility of computation", *IBM Journal of Research and Development*, **17** (1973), pp. 525-532.
[5] Bennett, C., Landauer, R.: "The fundamental physical limits of computation"; *Scientific American* **253** (July 1985), pp.38-46.
[6] J. Butler, K.J. Breeding, "Some characteristics of universal cell nets," *IEEE Trans. on Comp.,* pp. 897-903, Oct.1973.
[7] J.T. Butler, "On the Number of functions realized by cascades and disjunctive networks," *IEEE Trans. on Comp.,*Vol. **24**, No. 7, pp. 681-690, 1975.
[8] J.T. Butler, "Tandem Networks of Universal Cells," *IEEE Trans. on Comp.,* Vol. **27**, No. 9, pp. 785-799, 1978.
[9] J.T. Butler, "Analysis and Design of Fanout-free Networks of Positive Symmetric Gates*," J. of the Associ. for Comp. Mach.,* **25**, No. 3, pp. 481-498, 1978.
[10] A. Chojnacki, L. Jozwiak, "Multi-Valued Sub-function Encoding in Functional Decomposition Based on Information Relationships Measures", pp. 83 – 90, *Proc. ISMVL '2000,* Portland, Oregon.
[11] A. Chojnacki, L. Józwiak, "High-quality FPGA Designs through Functional Decomposition with Sub-function Input Support Selection Based on Information Relationship Measures," *Proc. IEEE International Symposium on Quality Electronic Design, ISQED'2001,* San Jose, California, USA, March 26-28, 2001.
[12] M. Chrzanowska-Jeske, Z. Wang, Y. Xu, "A Regular Representation for Mapping to Fine-Grain, Locally-Connected FPGAs," *Proc. Intern. Symp. On Circuits and Systems ISCAS'97*, pp. 2749-2752, June 1997.
[13] M. Chrzanowska-Jeske, Y. Xu, M. Perkowski, ``Logic Synthesis for a Regular Layout," *VLSI Design,* Vol. **10**, No. 1, pp. 35 - 55, 1999.
[14] H.A. Curtis, "A Generalized Tree Circuit", *JACM*, Vol. **8**, pp. 484-496, 1961.
[15] X. Deng, T. Hanyu, and M. Kameyama, "Quantum Device Model Based Super Pass Gate for Multiple-Valued Digital Systems," *Proc. ISMVL 95.*
[16] D. Deutsch, "Quantum computational networks", *Proc. Roy. Soc. Lond.* A 425 , 73 (1989).
[17] De Vos, A.: "Introduction to r-MOS systems"; *Proc. 4 th Workshop on Physics and Computation*, Boston, 1996, pp. 92-96.
[18] De Vos, A.: "Towards reversible digital computers"; *Proc. European Conference on Circuit Theory and Design*, Budapest (1997), pp. 923 - 931.
[19] De Vos, A.: "Reversible computing"; *Progress in Quantum Electronics* **23** (1999), pp. 1 - 49
[20] D.L. Dietmeyer, P.R. Schneider, "Identification of symmetry, redundancy and equivalence of Boolean functions," *IEEE Trans. Electron. Comp.,* Vol. **16**, pp. 804-817, Dec. 1967.
[21] D.L. Dietmeyer, "Logic design of digital systems," *Boston: Allyn and Bacon,* 1971.
[22] D.P. DiVincenzo, J. Smolin, "Results on two-bit gate design for quantum computers," *Proc. of the Workshop on Physics and Computation, PhysComp '94*, p. 14.
[23] R. Drechsler, "Pseudo-Kronecker Expressions for Symmetric Functions," IEEE Trans. on Computers, 48, no 9, 1999, pp. 987 – 990

[24] S. Elspas, “The Theory of Multirail Cascades,” in “Recent Developments in Switching Theory,” (ed. A.Mukhopadhyay), Academic Press, New York and London, 1971, pp. 315 – 367.
[25] K.Y. Fang, A.S. Wojcik, "Modular Decomposition of Combinational Multiple-valued Circuits*," IEEE Trans. on Comp.,* **37**, No. 10, pp. 1293-1301, 1988.
[26] R. Feynman, “Quantum Mechanical Computers,” *Optics News,* Vol. 11, 1985, pp. 11 – 20.
[27] Feynman, R.: "Feynman lectures on computation" (A. Hey and R. Allen, eds); *Addison-Wesley,* Reading (1996).
[28] C. Files, “A New Functional Decomposition Method as Applied to Machine Learning and VLSI Layout,” *Ph.D. Thesis,* Portland State University, June 2000.
[29] E. Fredkin, T. Toffoli, "Conservative Logic", *Int. Journal of Theor. Phys.,* **21** (1982), pp. 219-253.
[30] L. Józwiak, “General Decomposition and Its Use in Digital Circuit Synthesis,” *VLSI Design: An International Journal of Custom Chip Design Simulation and Testing,* Vol. **3**, Nos. 3-4, 1995.
[31] L. Józwiak, “Information Relationships and Measures: An Analysis Apparatus for Efficient Information System Synthesis,” *Proc. EUROMICRO-97, 23rd Conference "New Frontiers of Information Technology",* Budapest, Hungary, Sept. 1-4, 1997, pp. 13-23.
[32] L. Józwiak, “Efficient Logic Synthesis for FPGAs and PLDs with Information Relationships and Measures*,” IWLS'98 - The IEEE/ACM International Workshop on Logic Synthesis,* Tahoe City, California, USA, June 7-10, 1998, pp. 248-254.
[33] L. Józwiak, “Information Relationships and Measures in Application to Logic Design,” *IEEE International Symposium on Multiple-Valued Logic,* Freiburg Im Breisgau, Germany, May 20-22, 1999.
[34] L. Józwiak, A. Chojnacki, F. Volf, M. Perkowski, “A Bottom-Up Approach to Multiple-Level Logic Synthesis,” *Proc. of the Second International Workshop on Design and Diagnostics of Electronic Circuits and Systems,* Szczyrk, Poland, September 2 - 4, 1998, pp. 39-45.
[35] L. Józwiak, A. Chojnacki, “Functional Decomposition Based on Information Relationship Measures Extremely Effective for Symmetric Functions,” *Proc. 25th EUROMICRO Conference,* Milan, Italy, September 8-10,1999, pp. 150-160.
[36] L. Józwiak, A. Chojnacki, “High-quality Sub-function Construction in Functional Decomposition Based on Information Relationship Measures. DATE'2001, March, 2001. Munich, Germany, IEEE Computer Society Press, Los Alamitos, CA, USA
[37] L. Józwiak, A. Chojnacki, “Effective and Efficient FPGA Synthesis through Functional Decomposition Based on Information Relationship Measures,” *Proc. EUROMICRO Symp. on Digital System Design,* Sept. 4-6, 2001, Warsaw, Poland.
[38] P. Kerntopf, “A Comparison of Logical Efficiency of Reversible and Conventional Gates,” *9th IEEE Workshop on Logic Synthesis*, 2000, pp. 261 – 269.
[39] P. Kerntopf, “Non-linear Transformations of Decision Diagrams”, *Proc. IWLS 2001,* pp. 173-178.
[40] P. Kerntopf, “An Approach to Minimization of Decision Diagrams,” *Proc. EUROMICRO Symposium on Digital Systems Design, Warsaw, Poland, Sept. 2001.*
[41] P. Kerntopf, “Maximally Efficient Binary and Multi-Valued Reversible Gates,” *ULSI Workshop, Warsaw, Poland, May 2001.*
[42] B.G. Kim, D.L. Dietmeyer, "Multilevel Logic Synthesis of Symmetric Switching Functions," *IEEE Trans. on CAD*, Vol. 10, No. 4, pp. 436-446, Apr. 1991.
[43] V. N. Kravets, K. A. Sakallah, “M32: A Constructive Multilevel Logic Synthesis System”, *Proc. of DAC ‘98,* pp. 336-341.
[44] V.N. Kravets, K.A. Sakallah, “Constructive Library-Aware Synthesis Using Symmetries”. *Proc. of D&T in Europe*, 2000, pp.208-213.
[45] R. Landauer, “Irreversibility and heat generation in the computational process”, IBM Journal of Research and Development, **5,** pp. 183-191, 1961.
[46] K.K. Maitra, "Cascaded switching networks of two-input flexible cells," *IRE Trans.* 1962, EC-11, pp. 136-146.
[47] A. Michalski, “On synthesis of of combinational circuits with modifiable interconnection patterns,” *Polish Scientific Publishers,* Lodz, Poland, 1980, (in Polish).
[48] A. Mishchenko, B. Steinbach, M. Perkowski, “An Algorithm for Bi-Decomposition of Logic Functions,” *Proc. DAC’2001,* pp. 103 – 108.
[49] A. Mishchenko, B. Steinbach, M. Perkowski, “Bi-Decomposition of Multi-Valued Relations, “ *Proc. IWLS‘2001,* pp. 35-40.
[50] M. Perkowski, E. Pierzchala, ``New Canonical Forms for Four-Valued Logic," *Report, Electrical Engineering Department,* PSU, 1993, http://www.ece.pdx.edu/~mperkows/=PUBLICATIONS/=publications-1993.html
[51] M. Perkowski, E. Pierzchala, R. Drechsler, ``Ternary and Quaternary Lattice Diagrams for Linearly-Independent Logic, Multiple-Valued Logic and Analog Synthesis,” *Proc. ISIC-97,* Singapur, 10-12 Sept.1997.
[52] M. Perkowski, E. Pierzchala, R. Drechsler, ``Layout-Driven Synthesis for Submicron Technology: Mapping Expansions to Regular Lattices," *Proc. First International Conference on Information, Communications and Signal Processing, ICICS'97,* Singapur, 9-12 Sept. 1997.
[53] M. Perkowski, M. Chrzanowska-Jeske, Y. Xu, ``Lattice Diagrams Using Reed-Muller Logic," *Proc. RM'97 Conference,* Oxford Univ., U.K., Sept. 1997, pp. 85 - 102.

[54] M. Perkowski, A. Al-Rabadi, P. Kerntopf, A. Mishchenko, M. Chrzanowska-Jeske, “Three-Dimensional Realization of Multiple-Valued Functions using Reversible Logic”, *Proceedings of ULSI 2001 Workshop, Warsaw, Poland, May 2001.*
[55] M. Perkowski, P. Kerntopf, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, X. Song, A. Al-Rabadi, L. Jozwiak, A. Coppola, B. Massey, “Regular Realization of Symmetric Functions Using Reversible Logic”, *accepted to Euro-Micro,* 2001
[56] M. Perkowski, P. Kerntopf, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, X. Song, A. Al-Rabadi, L. Jozwiak, Alan Coppola, B. Massey, “ Regularity and Symmetry as a Base for Efficient Realization of Reversible Logic Circuits”, *Proceedings of IWLS 2001, pp.* 90 - 95.
[57] P. Picton, "Optoelectronic, Multivalued, Conservative Logic”, *Int. Journal of Optical Computing,* **2**., 1991, pp. 19-29.
[58] P. Picton, “Modified Fredkin Gates in Logic Design,” *Microelectronics Journal*, **25,** 1994, pp. 437-441.
[59] P. Picton, “Multi-valued Sequential Logic Design Using Fredkin Gates”, *MVL Journal,* **1**, 1996, pp. 241-251.
[60], P. Picton, “A Universal Architecture for Multiple-valued Reversible Logic,” *MVL Journal,* **5,** 2000, pp.27-37.
[61] A. Peres, "Reversible Logic and Quantum Computers", *Physical Review A,* Vol. 32, 1985, pp. 3266-3276.
[62] 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.
[63] M. Rawski, L. Józwiak, T. Luba, A. Chojnacki, “Efficient Logic Synthesis for FPGAs with Functional Decomposition Based on Information Relationship Measures,” *Proc. EUROMICRO-98 Conference,* Vasteras, Sweden, August 25-27, 1998, pp. 8-15.
[64] M. Rawski, L. Józwiak, A. Chojnacki, “Application of the Information Measures to Input Support Selection in Functional Decomposition,” *Proc. RSCTC'98 - International Conference on Rough Sets and Current Trends in Computing,* Warsaw, Poland, June 22 - 26, 1998, ISBN 3-540-64655-8, Springer Verlag, Berlin, 1998, pp. 573 - 580.
[65] M. Rawski, L. Józwiak, T. Luba, “Efficient Input Support Selection for Sub-functions in Functional Decomposition Based on Information Relationship Measures,” *Proc. 25th EUROMICRO Conference,* Milan, Italy, September 8-10,1999, pp. 94-101.
[66] M.R. Rayner, D.J. Newton, "On the Symmetry of Logic ", *Journal of Physics A: Mathematical and General,* Vol. 28, 1995, pp. 5623-5631.
[67] T. Sasao, K. Kinoshita, “Cascade realization of 3-input 3-Output Conservative Logic Circuits*”, IEEE Trans. on Computers*, **27**, 1978, pp. 214-221.
[68] T. Sasao, K. Kinoshita, “Conservative Logic Elements and Their Universality*,” IEEE Trans. on Computers*, **28**, 1979, pp. 682-685.
[69] T. Sasao, K. Kinoshita, “A Catalog of Magnetic Bubble Logical Circuits for Three-Variable Logical Functions,” Technology Reports of the Osaka University, 24, No.1169, pp. 133-140, 1974.
[70] P.R. Schneider, D.L. Dietmeyer, "An algorithm for synthesis of multiple-output combinational logic*," IEEE Trans. on Comp.,* pp. 117-128, Febr. 1968.
[71] 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.
[72] T. Toffoli, "Reversible computing"; in: "Automata, languages and programming" (J. De Bakker and J. Van Leeuwen, eds); *Springer,* New York (1980), pp. 632 - 644.
[73] F.A.M. Volf, L. Józwiak, M.P.J. Stevens, “Division-Based versus General Decomposition-Based Multiple-Level Logic Synthesis,” *VLSI Design: An International Journal of Custom Chip Design Simulation and Testing,* **3**, No 3-4, 1995.
[74] F.A.M. Volf, L. Józwiak, “Decompositional Logic Synthesis Approach for Look Up Table Based FPGAs, *Proc. 8th IEEE International ASIC Conference,* Austin, Texas, 18-22 Sept., 1995.
[75] F.A.M. Volf, “A bottom-up approach to multiple-level logic synthesis for look-up table based FPGAs*”, Ph.D. Thesis, Technical University of Eindhoven,* 29 September 1997.