SOFTWARE-HARDWARE CODESIGN APPROACH TO GENERALIZED ZAKREVSKIJ STAIRCASE METHOD FOR EXACT SOLUTIONS OF ARBITRARY CANONICAL AND NON-CANONICAL EXPRESSIONS IN GALOIS LOGIC

M. A. Perkowski, P. Lech, Y. Khateeb, R. Yazdi, and K. Regupathy,
Dept. Electr. Engn., Portland State Univ., Portland, OR 97207

Staircase Method of Zakrevskij has been used for exact solution to incomplete Exclusive-Or Sum of Products (ESOP) and Fixed Polarity Reed-Muller (FPRM) forms. It can be observed, that the method of Zakrevskij can be generalized to all problems where the function sought is the canonical form of an EXOR of arbitrary linearly independent (LI) generating functions. Next it can be observed that the EXOR expression can be not necessarily canonical, so that arbitrary functions are used instead of LI functions. Finally, the method can be extended to a non-canonical Galois Field (GF(k)) sum of arbitrary multivalued generating functions. We call this the "Generalized Zakrevskij Staircase Method". This paper introduces a software-hardware approach using the DEC PERLE-1 FPGA-based board. The Boolean decision function to be satisfied with minimum non-zero arguments is a generalization of the Helliwell Function and is first minimized in software taking into account the predicted cost of the solution. Next, the simplified incomplete function is realized in hardware and an exhaustive breadth-first search with use of a special "m out of N" counter is executed, where m is an expected solution cost and N is the number of classes of equivalent generating functions. The method is efficient because for incomplete functions "m" is small even for large "N".

A Xilinx 3090 FPGA-based accelerator board to DECstation 5000 executes a computation intensive part of the algorithm and achieves performance superior to supercomputers with a fraction of cost, while the rest of the algorithm is done by software. The joint Reconfigurable Computing Project of Portland State Univesity and Technical University of Eindhoven aims at using these concepts in a new application domain - **solving of NP-hard search combinatorial problems** [4].

Generalized Zakrevskij Staircase Method for EXOR logic minimization.

In 1988, Martin Helliwell introduced the Helliwell Function for exact ESOP minimization [3] and implemented a GAL-based circuit (courtesy Lattice Corp.) for hardware minimization of exact ESOPs for single-output 5-variable functions. The **generating functions** were all possible products of terms of n-variable function F, there existed thus $N = 3^5 = 243$ of such generating functions. There were $2^5 = 32$ flip-flops corresponding to every minterm of the function, set initially to the value of a function to be minimized. The problem was to find, by a hardware search, such choice of the generating functions that the EXOR of them would make the states of all flip-flops equal to zero ($F = \sum_g \; iff \; F \oplus \sum_g = 0$). A **243-bit binary counter in natural code** was used to exhaustively search all combinations of the generating functions, so the search was generating worse solutions after already finding a solution with a smaller cost (such as generating candidate 00...01110 after already finding that combination 00...00011 was a solution). This was the first hardware accelerator for EXOR logic problem and its performance was much superior to IBM PC AT, but the limited size of functions discouraged us at this time to continue this research. Searching with a binary counter is not a **depth-first**, nor a **breadth-first** method and its only advantage is the simplicity and regularity of hardware.

In 1990 Perkowski and Jeske found several generalizations of the Helliwell's Method: to multi-output, multi-valued functions, to Positive Polarity Reed-Muller Forms, to Fixed Polarity Reed-Muller Forms, GRM forms and other [2, 12]. The method was implemented in software using depth-first search, but unfortunately the limit of 5 variables was not exceeded. However, we observed that the search algorithms can be made much more efficient for strongly unspecified functions, and by using more sophisticated tree search strategies. A tree is pruned by finding equivalent operators on each level. A better search strategy was subsequently implemented which allows to realize a higher percent of functions with more than 5 variables, and these improved results will be soon published. We observed also that very good upper bounds can be found by EXORCISM-MV-2, which are next used to limit the first backtracks in our search. In 1993 Sasao solved Helliwell's Function for only small functions using BDDs, but his method allowed to realize more functions of more than 5 variables than our approach at this time. Similar BDD-based results were reported by Somenzi and Escobar and other authors (unpublished). In 1993 Perkowski generalized the Helliwell's function for arbitrary canonical family in Linearly Independent Logic and for Galois Logic.

Zakrevskij [15] introduced the so-called Staircase Method which he used for exact solution to (non-canonical) ESOP expressions and (canonical) FPRM forms, and reported good results for multi-output strongly incomplete functions, specified by minterms. In essence, his method is based on partitioning a set of generating functions (products of variables or products of literals in his case), to classes of equivalency with respect to a set of care minterms. Two generating functions are equivalent if they cover exactly the same care minterms. If a function is completely specified, there are no equivalent generating functions for it. (The more don't cares exist in the function, the larger are the equivalency classes, so Zakrevskij method has the best advantage for very strongly unspecified functions). Only one representative function is selected from a class for subsequent search, which limits substantially the search. It can be observed that from a programming point of view the depth-search software search method is used with cost-based backtracking. Zakrevskij's method was next generalized, improved and extended by Shmerko, Zajtseva, and Yanushkevitch and other researchers in Europe, including logics more general than binary. The idea of dividing to classes of equivalency is the reason of the success of this method for incomplete functions, but with the introduction of more cares the method becomes not efficient, and is very similar to the one from [3]. On the other hand, when the exact ESOP or FPRM minimization is not required, there exist many efficient algorithms, such as EXORCISM-MV-3, so Zakrevskij Method loses its merits when approximate solutions are sought.

There are, however, the following two ideas in Zakrevskij Method that remain very powerful to generate exact solutions, and should be in our opinion further investigated: **(1)** Partitioning the set of generating functions to equivalence classes. Although used by Zakrevskij only to ESOP and PPRM solutions, (3^n and 2^n generating functions corresponding to terms of literals, and variables, respectively), it can be used to arbitrary Linearly Independent family of functions. Moreover, this idea can be used to **any** functions, even linearly dependent, such as generating functions for all possible polarities for AND/OR/EXOR expansions based on generalized Maitra cascades [6, 5, 7, 13, 11, 9, 8] and AND/OR cascades. **(2)** Using the breadth-first search at level m, next at level $m-1$, $m-2$, etc, which leads to the staircase pattern.

The following observations can be made:

(a1) The method of Zakrevskij can be generalized to all problems where the function sought is the canonical form of an EXOR of arbitrary linearly independent (LI) generating functions. The only difference is that instead of all products of literals, all generating functions of certain Linearly Independent form are created [11], which can be of AND/OR, AND/OR/EXOR, or another type [6, 5, 7, 13, 11, 9, 8]. Next, for these generating functions, groups of functions are compared and all those that cover exactly the same care minterms are put into one equivalence class. The equivalence classes are thus generated as in the original staircase method, and one representative from each class is selected. Next the search is performed as in Zakrevskij Method. This generalized method can be applied to arbitrary Linearly Independent forms, which means, in particular, to all AND/EXOR forms such as Fixed Polarity, Generalized Reed-Muller, Kronecker, Pseudo-Kronecker, Generalized-Kronecker, and all AND/OR/EXOR forms such as those from [2, 12, 6, 5, 7, 13, 11, 9, 8].

(a2) The original method of Zakrevskij was created for Positive Polarity Reed-Muller forms (PPRM). It was next extended for FPRM and ESOP, but it is our understanding that the generalization for ESOP was not exact. It can be observed that the EXOR expression of generating functions used in both ESOP and PPRM can be not necessarily canonical, so that arbitrary functions can be used instead of LI functions. Thus, if for some reason given is an arbitrary set of generating functions (specified by a singular matrix M, [6]), then a minimal solution is generated for a non-canonical expression type. This way, the method can be also applied to an arbitrary class of non-canonical expressions that includes the class of canonical Linearly Independent forms defined in point (a1). These expressions can be ESOPs (when all 3^n product terms are used as generating functions), and all AND/EXOR subsets of ESOPs that are created by selecting smallest sets of generating functions from the sets being unions of sets of some generating functions of some canonical forms. More interestingly, all AND/OR/EXOR expressions created from the generating functions which are selected from unions of sets of generating functions for any canonical AND/OR/EXOR forms can be also minimized this way. Maitra cascades [6, 5, 7, 8]. We call this the Generalized Zakrevskij Staircase Method.

In conclusion, Zakrevskij Method in this generalized formulation becomes already **the** most general method of solving **all exact problems** for all Linearly Independent and non-canonical multi-output expressions that have EXOR output gates. Unfortunately, as it is now, the method is computationally not efficient. To release its full potential, more sophisticated search methods, and a hardware realization of search should be employed. This paper introduces a software-hardware approach using the DECPeRLe-1 FPGA-based board.

Our approach to minimize arbitrary single-output expression with EXOR output gate and with given set of generating functions has several stages as follows:

Given:

(d1) the set of care minterms of a multi-output function F, with the corresponding binary output values of a single-output function for each care minterm. **(d2)** the set of generating functions.

Find: The minimum solution, i.e. the expression being an EXOR of generating functions with the minimum number of inputs to the output EXOR gate. (i.e., in other words, the minimum number of EXOR-ed functions selected from the set of generating functions from (d2)).

(s1) **Step executed in software.** For function F of n variables create an arbitrary number C of all generating functions G_i stored in hypothetical registers ($C = 2^n$ for any canonical AND/EXOR form, 3^n for ESOP, $C = 2^n$ for any LI form, $C = 3^n$ for non-canonical expressions being generalizations of canonical Maitra LI forms, $C = v \cdot 2^n$ for a combination of generating functions from various canonical forms, etc.). Create a binary matrix with generating functions as columns and care minterms as rows. Thus there exist $R \leq 2^n$ rows. If function covers this minterm, there is a 1 on their intersection, otherwise - 0. (s1a) Find equivalence classes of generating functions with respect to care minterms. Two functions are equivalent if they have the same columns. If a function is completely specified, this step should be omitted. (s1b) Select one representative from each class (exactly as in Zakrevskij method). Thus the new matrix has $N \leq C$ columns.

(s2) **Step executed in software.** Create a function $\bigoplus_{i=1}^{i=N} c_i\ G_i$, where G_i are vectors of *minterm-variables* G_{i_j}. Minterm-variables correspond to values of function G_i on all care minterms. Each such vector corresponds then to a column of the matrix. All these vectors have length R. For any given function G_i, all the minterm-variables G_{i_j} are constants 0 or 1. Variable c_i=1 means selection of function G_i for the EXOR combination. Variable c_i=0 means no selection of function G_i for the EXOR combination.

Satisfaction of formula

$$\bigoplus_{i=1}^{i=N} c_i\ G_i \oplus F = \underline{0} \qquad (F1),$$

where $\underline{0} = (00...0)$,

means that function F is realized by the selected generating functions for which c_i =1, with cost being the number of c_i that are equal 1. Thus, the minimum number of c_i = 1 for which formula $(F1)$ is satisfied, is the exact minimum solution to the generalized minimization problem. In yet another words, exoring all selected groups equals the original function F. The decision function from formula F1 is a generalization of the Helliwell Function. Its generalization for multi-output case is trivial, the cares of each output must be separately repeated in the vectors. Figure 1 explains the principle of our approach.

(s3) **Step Executed in software.**

(s3a) Substitute constant values of minterm-variables from registers G_i to function $\bigoplus_{i=1}^{i=N} c_i\ G_i$. Thus, function $\bigoplus_{i=1}^{i=N} c_i\ G_i$ becomes now function $H(c_1, c_2, ..., c_N)$ of only c_i variables.

(s3b) Having value m+1 of an upper bound of the solution, found in software by a heuristic search depth-first program, expect the solution with cost m and modify function H accordingly to this expectation. First, create function S^m with arguments c_i. (By S^m we denote the (single-index) symmetric function on variables c_i, that equals 1 when exactly m of its input variables are equal 1.) Create a new decision function: $H \cap S^m$ = (ON,OFF), and replace all minterms in $H \cap \overline{S^m}$ by don't cares.

(s3c) Using fast Boolean minimization methods, simplify the incomplete function H_2 = (ON,OFF), where ON = $ON(H) \cap S^m$, and OFF = $OFF(H) \cap S^m$ to some completely specified function H'_2.

(s3e) Download the simplified, completely specified, decision function $H'_2(c_1, .., c_N)$ as a netlist to hardware. If this function is zero, there is no solution of cost m. Otherwise, there exists at least one solution of cost m.

(s4) **Step executed in hardware.** Use the special Search Counter (address generator) circuit that generates all binary vectors $(c_1, .., c_N)$ in "m out of N code". If $H'_2(c_1, .., c_N) = 1$ for some vector of c_i-s then a new solution is found. If it is known that a better solution cannot be found, return and stop (see below), else continue. This way, a single level on depth m is searched in the solution tree for one downloading of a hardware configuration.

(s5) **Steps executed in software and hardware.** Repeate above steps (s1)-(s4) for $m-1$, $m-2$,... and smaller values of m. This means, for every value of m, a new hardware configuration is created and downloaded.

We developed several variants of this algorithm which speed-up the operation in some special cases. **(case 1).** In some problems, for instance in ESOP minimization, it can be proved that for every m, if a solution with cost m exists, then also a solution with cost $m + 1$ exists. In such case, if no solution is found for certain m, then our algorithm can stop and return $m + 1$ as the minimum solution cost. Its corresponding combination of c_i satisfying the equation $H'_2 = 1$ is returned as the solution.

(case 2). When no upper bound is known, the algorithm with increasing the value of m can be used, instead of the above algorithm with decreasing the value of m. In the increasing variant, the first solution is the minimum one, but usually more iterations are needed. In this variant, it speeds the algorithm when we calculate a lower bound of the cost as the starting value of m.

(case 3). When function is complete and exact ESOP is looked for, a simplified algorithm is executed. It starts from a upper bound found by EXORCISM-MV-2 and EXORCISM-MV-3. When it finds in (s3c) that $H \neq 0$ it is not downloading the configuration to hardware but keeps decreasing m by one and iterating. When it finally finds for value m' in (s3c) that $H = 0$ which means, no solution of cost m', it downloades the configuration for $m' + 1$ to hardware and finds the solution in hardware.

The problem of designing the "m out of N" counter is an interesting design problem in itself which will be discussed in our oral presentation. It is designed as a one-dimensional cellular automaton. The method is (relatively) efficient because for incomplete functions the value of "m" is small even for large "N". At the time of this writing we do not know how big functions would be possible to minimize.

Approach to the Minimization of Multivalued Logic Functions with Galois-Addition Output Gates.

Zakrevskij Method and its hardware realization presented above can be further extended to canonical and non-canonical Galois(k) sum of arbitrary multivalued generating functions. The only differences are the following: (a) Instead of EXOR gates, the Galois(k) addition gates are used in the Generalized Helliwell Function H. (b) Instead of storing values 0 and 1 as for GF(2), the flip-flops in registers of function F and of generating functions store values from 0 to $k-1$ for k-valued logic. Because the coding for $k \neq 2^r$ is difficult, we concentrated on the cases when $k = 2^r$. Thus, a 2^r-valued signal is realized with r

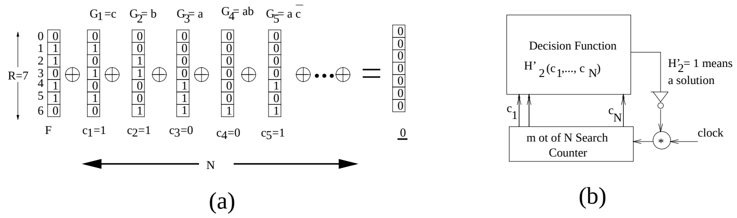

Figure 1: *Principle of the presented method for ESOP minimization: (a) EXOR-ing groups G_i selected by signals c_i with F. Groups c, b and $a\bar{c}$ are selected to realize function F = (ON,OFF) where ON = (1,2,4,5), OFF = (3,6). The result of EXORing is vector $\underline{0}$, so $F = c \oplus b \oplus a\bar{c}$. (b) the schematic diagram explaining the configuration downloaded to hardware.*

binary singals. For instance, this way, only two two-input EXOR gates are needed to realize a Galois(4) addition and only 3 two-input EXOR gates for Galois(8) addition. The construction of the "m out of N" Search Counter and the rest of the circuit remain the same. Similarly, the software preprocessing of the function is basically the same. This way, the expressions that are GF(k) additions of arbitrary k-valued generating functions can be exactly minimized. The classes of expressions include all Galois Logic canonical forms, all Linearly Independent forms with GF(k) Addition, all Galois Logic Linearly Independent GF(k) forms, and all their non-canonical extensions and generalizations [6, 1, 10]. Thus, similarly as in the binary case, **the Generalized Zakrevskij Method allows to find exact solutions for arbitrary canonical and non-canonical expressions.**

Conclusion.

Our main idea can be summarized as follows: every (multi-output, incomplete) k-nary input k-nary output function realized in the form of a GF(k)-addition of arbitrary functions from a well-defined set of functions over GF(k), can be minimized (exactly or approximately) in a system, that realizes a generalized Helliwell function in a hardware data-path, and implements the staircase algorithm of Zakrevskij with a sequence of down-loaded "m out of N" Address Generators realized as Cellular Automata (Finite State Machines). In addition, a general software technique of pre-processing the function H was shown, that can be used for any kind of tree-search problems, realized in hardware. We created very similar approaches to solving arbitrary Boolean equations, Generalized Satisfiability Functions, Graph Coloring, Maximum Clique, Set Covering, Petrick Functions, and Clique Partitioning using FPGA technology. In each of them, the essence is to perform the enumeration of all subsets and checking some logical conditions, using various Address Generators that correspond to Depth-First, Breadth-First and other Tree Search methods. Most generally, the main contribution of this note is to propose a very general method to perform arbitrary tree search for NP-complete problems using hardware Address Generators.

References

[1] K.M. Dill, K. Ganguly, R. Safranek, and M.A. Perkowski, "A New Linearly Independent, Galois-Field Reed-Muller Logic," *submitted to Reed-Muller'97.*

[2] D.H. Green, "Families of Reed-Muller Canonical Forms," *Intern. J. of Electr.*, pp. 259-280, February 1991, No 2.

[3] M. Perkowski, M. Chrzanowska-Jeske, "An Exact Algorithm to Minimize Mixed-Radix Exclusive Sums of Products for Incompletely Specified Boolean Functions," *Proc. of the IEEE ISCAS'90 Symp.*, pp. 1143-1146, New Orleans, 1-3 May, 1990.

[4] M. Perkowski, L. Jozwiak, and D. Foote, "Architecture of a Programmable FPGA Coprocessor for Constructive Induction Approach to Machine Learning and other Discrete Optimization Problems," *Proc. 4th Reconfigurable Architectures Workshop,* Geneva, Svitzerland, April 1-5, 1997.

[5] M. Perkowski, "A Fundamental Theorem for Exor Circuits," *Proc. Reed-Muller'93*, pp. 52-60.

[6] M. Perkowski, A.Sarabi, F. Beyl, "XOR Canonical Forms of Switching Functions," *Proc. of Reed-Muller'93*, pp. 27-32.

[7] M. Perkowski, A.Sarabi, and F. Beyl, "Fundamental Theorems and Families of Forms for Binary and Multiple-Valued Linearly Independent Logic," *Proc. Reed-Muller'95*, pp. 288-299.

[8] M. Perkowski, L. Jozwiak, and R. Drechsler, "A Canonical AND/EXOR Form that includes both the Generalized Reed-Muller Forms and Kronecker Reed-Muller Forms," *submitted to Reed-Muller'97.*

[9] M.A. Perkowski, L. Jozwiak, and R. Drechsler, "Two hierarchies of Generalized Kronecker Trees, Forms, Decision Diagrams, and Regular Layouts," *submitted to Reed-Muller'97.*

[10] M. Perkowski, L. Jozwiak, R. Drechsler, and B. Falkowski, "Ordered and Shared, Linearly Independent, Variable-Pair Decision Diagrams for Incompletely Specified Functions," *submitted,*1997.

[11] M. Perkowski, B. Steinbach, "Efficient Exact Minimization of Incomplete AND/EXOR Forms," *in preparation*, 1997.

[12] T. Sasao (editor), "Logic Synthesis and Optimisation," *Kluwer Academic Publishers,* 1993.

[13] N. Song, M. A. Perkowski, M. Chrzanowska-Jeske, A. Sarabi, "A New Design Methodology for Two-Dimensional Logic Arrays," *VLSI Design,* special issue on Decomposition in VLSI Design, (L. Jozwiak ed)., Vol. 3, No. 3-4, 1995.

[14] J. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. Touati, and Ph. Boucard, "Programmable Active Memories: Reconfigurable Systems Come of Age," *IEEE Trans. on VLSI Systems,* Vol. 4, No. 1., pp. 56 - 69, March 1996.

[15] A. Zakrevskij, "Minimum polynomial implementation of systems of incompletely specified Boolean functions," *Proc. Reed-Muller'95,* pp.250-256.