A Quantum Algorithm for Finding Minimum Exclusive-Or Expressions for Incompletely Specified Boolean Functions. PRELIMINARY VERSION

M. Sampson, D. Voudouris, M. Kalathas, G. Papakonstantinou
Dept of Electrical and Computer Engineering
Division of Computer Science
Computing Systems Laboratory
National Technical University of Athens
157 80, Zografou Campus, Greece

Abstract

This paper presents a quantum algorithm for finding minimal ESCT (Exclusive-or Sum of Complex Terms) or ESOP (Exclusive-or Sum Of Products) expressions for any arbitrary incompletely specified switching function. The proposed algorithm takes advantage of the inherent massive parallelism of quantum circuits in order to achieve better complexity than the conventional ones. The proposed Exclusive-Or (xor) expressions such as ESCT can be used to implement an arbitrary Boolean function into a reversible or even a quantum circuit.

I. Introduction

Quantum circuits have already begun to establish themselves as the future in the computer design technology. Despite the technical difficulties in implementing a complete quantum computer, quantum circuits are already being studied thoroughly. Many algorithms, specifically designed for quantum computers, have been proposed and some of them prove that, in certain kinds of problems, a quantum computer can achieve far better complexity than a conventional one. The three algorithms that constitute, so far, the foundations of quantum algorithms are the Shor's [1], the Grover's [2] and the Quantum Fourier Transformation [3] algorithms. In particular, the Shor's algorithm can find the periodicity of a function in polynomial time, providing exponential speedup which, in principle, renders RSA and related cryptography algorithms obsolete. Grover's algorithm is the optimal quantum searching algorithm even though it doesn't achieve the spectacular speedup of the previous one. Finally, the quantum Fourier Transform is actually the implementation of the Discrete Fourier Transform as a quantum circuit and has many applications in quantum algorithms as it provides the theoretical basis to the phase estimation procedure and is a key feature for many important quantum algorithms. All these quantum algorithms achieve complexities far better than their conventional counterparts. It seems that quantum computers are the answer in solving complexity intensive classical problems.

A very interesting and very difficult problem for a conventional computer is to find minimal ESCT (Exclusive-or Sum of Complex Terms) or ESOP (Exclusive-or Sum Of Products) expressions for an arbitrary completely specified switching function. Finding minimal ESCT or ESOP expressions for an incompletely specified function is even more difficult.

An interesting property of the ESCT and ESOP expressions is that they can be directly mapped to cellular architectures like the Maitra Cellular Architecture (Fig. 1) which has been proved to be reversible [4]. Therefore expressing a Boolean function in ESCT or ESOP form results in a reversible circuit and may help in designing quantum circuits [4].

Some algorithms for finding minimal ESOP or ESCT expressions for an arbitrary completely specified switching function, but with limitations on its number of input variables or the number of terms in its minimal forms, have been presented in the past [5], [6], [7], [8], [20], [19], [12]. Others have been designed in order to detect almost minimal ESOP or ESCT expressions but for more input variables [4], [9], [10], [19], [12], [18]. Algorithms for finding almost minimal ESCT or ESOP expressions for incompletely specified functions are significantly less [21], [22], [23], [24].

In [15] a quantum algorithm for finding FPRM (Fixed Polarity Reed Muller) expressions with number of terms less than a specified threshold is described. It proposes the construction of a specialized quantum operator (oracle) which evaluates FPRM expressions. It then uses this oracle in conjunction with Grover's algorithm [2] in order to find the FPRM expressions with the desired characteristics. In [16] a quantum algorithm (QMin1) for finding ESCT or ESOP expressions with number of terms less than a specified threshold is described (and if the threshold is appropriately selected, it finds minimal expressions). QMin algorithm uses a modified oracle for Grover's algorithm and detects minimal ESCT or ESOP expressions for an arbitrary completely specified Boolean function.

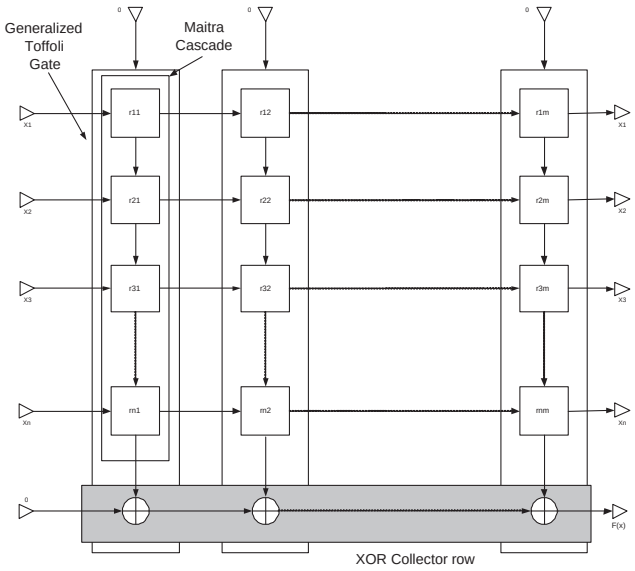

Fig. 1. Reversible wave cascade CA

In this work the previously mentioned QMin algorithm [16] is extended in order to detect minimal ESCT or ESOP expressions for incompletely specified Boolean functions (QMin is designed for completely specified Boolean functions). To the best of the authors' knowledge, this is the first quantum algorithm that detects minimal ESCT or ESOP expressions for an arbitrary incompletely specified Boolean function.

II. Theoretical Background

A. Definitions

In this section we provide some background definitions.

***Definition** 1:* Let $V_i = \{0,1\}, i = 1,\ldots,n$ and $B = \{0,1\}$. A single output Boolean function $f(X_1,\ldots,X_n)$ is a mapping $f : V_1 \times \ldots \times V_n \rightarrow B$. Variables $X_1,\ldots,X_n$ are called the support of f.

***Definition** 2:* Let $X_1^*,\ldots,X_n^*$ be literals ($X_i^* = 0,1$, $i = 1,\ldots,n$) of the Boolean variables $X_1,\ldots,X_n$, which belong to the support of Boolean function f. Then $C = X_1^* \cdots X_n^*$ is a minterm of f.

***Definition** 3:* Let $V_i = \{0,1\}$ and $T = \{0,1,d\}$. A single output incompletely specified Boolean function $f(X_1,\ldots,X_n)$ is a mapping $f : V_1 \times \ldots \times V_n \rightarrow T$. The minterms for which $f(X_1,\ldots,X_n) = d$ are the don't care minterms of the function (DC-set). In these cases the value of f is unspecified. The set of minterms for which $f = 0$ is defined as the OFF-set of f. Likewise the set of minterms for which $f = 1$ is the ON-set of f.

***Definition** 4:* A complex Maitra term [11](complex term or Maitra term for simplicity) is recursively defined as follows:

1) Constant 0 (1) Boolean function is a Maitra term.
2) A literal is a Maitra term.
3) If M_i is a Maitra term, a is a literal, and G is an arbitrary two-variable Boolean function (Maitra cell), then $M_{i+1} = G(a, M_i)$ is a Maitra term.

Additionally, it is required that each variable appears in each Maitra term only once.

As it was presented in [6], every complex term can be written as: $F_p(x_n, y)$, where y is a complex term, not depending on x_n, and $F_p(x_n, y)$ denotes a boolean function between x_n, y. In particular, $F_p(x_n, y)$ can be one of the following: $F_p(x_n, y) = \{F_1(x_n, y) = x_n + y, F_2(x_n, y) = \bar{x_n} + y, F_3(x_n, y) = \bar{x_n}y, F_4(x_n, y) = x_n y, F_5(x_n, y) = x_n \oplus y, F_6(x_n, y) = y\}$. Hence, $p = 1, 2, ..., 6$.

A wave cascade expression can be directly mapped to a special celllular architecture, called the reversible wave cascade cellular architecture.

***Definition** 5:* An ESCT (Exclusive-or Sum of Complex Terms) expression (some times also called reversible wave cascade or Maitra expression) for a switching function is an exlusive-OR sum of complex terms:

$$Q = \sum_{i=1}^{m} \oplus M_i,$$

where M_i are complex terms and m is their number inside the expression. The same variable ordering is used for every M_i. The size, $s(Q)$, of the expression Q is defined as the number of complex terms inside the expression.

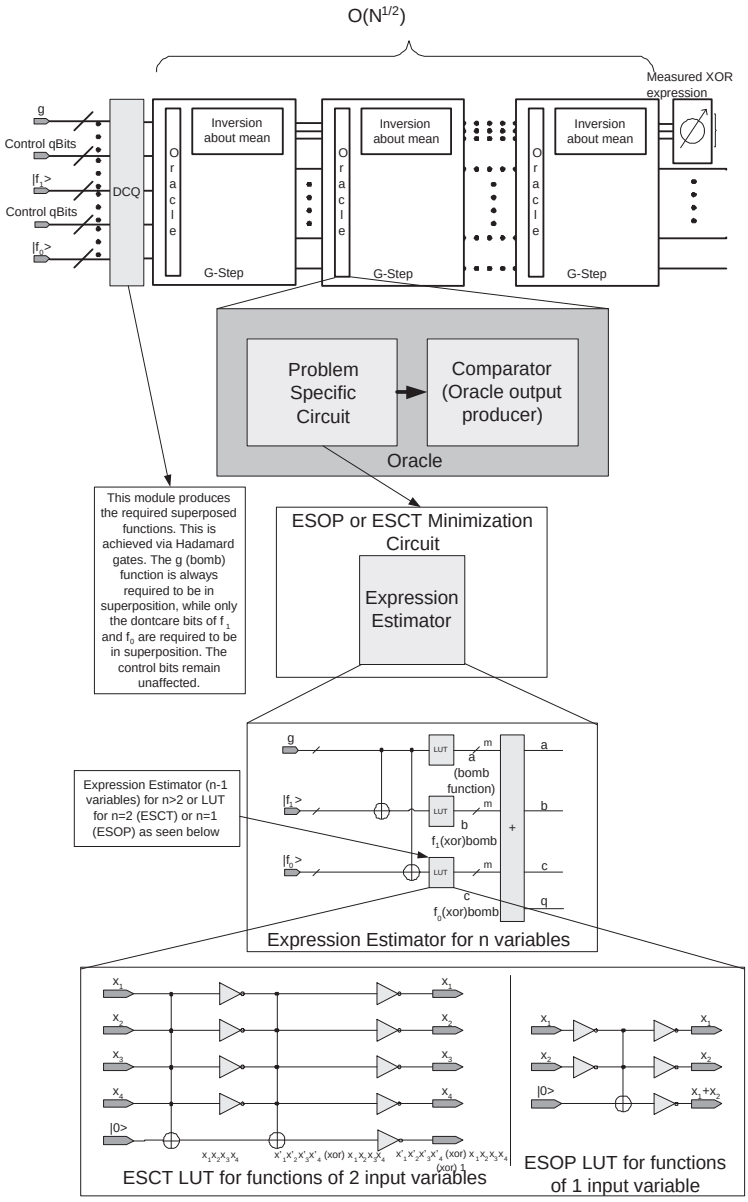

Fig. 2. DCQmin Algorithm hierarchy

In the case where the Maitra cells of a complex term do not implement the logical OR and XOR functions, the complex term is also called a product term and an ESCT expression composed by product terms is reduced to an ESOP expression.

***Definition** 6:* A minimal (or exact) ESCT (ESOP) expression of a switching function $f(x_1, \ldots, x_n)$ of n variables is defined as the one which has fewer number of terms comparing to every other ESCT (or ESOP) expression for this function.

***Definition** 7:* The ESCT (ESOP) weight $w(f)$ (or simply weight) of a switching function $f(x_1, \ldots, x_n)$ of n variables is defined as the number of complex terms (product terms) in a minimal ESCT (ESOP) expression of f.

Every two-variable switching function has ESCT weight equal to 1 [11].

***Definition** 8:* The subfunctions f_1, f_0, f_2 of a switching function $f(\mathbf{x})$ ($\mathbf{x}$ is the vector of its variables) are defined as $f_1 = f(x_1, x_2, \ldots, x_{n-1}, 1)$, $f_0 = f(x_1, x_2, \ldots, x_{n-1}, 0)$, $f_2 = f_1 \oplus f_0$, regarding variable x_n.

***Definition** 9:* The minterm representation (MT) of a switching function f with n variables is a bitvector of size 2^n where the i-th bit is 1 if the i-th minterm of f is 1.

For the rest of this paper, the minterm representation of a switching function will be enclosed in brackets. It is very easy to prove that the upper half of the MT representation of a switching function corresponds to its f_1 subfunction, while the lower half corresponds to its f_0 subfunction.

***Definition** 10:* Let f be a Boolean function, x_n a Boolean variable that belongs to f's support and f_0, f_1, f_2 are the

subfunctions of f regarding x_i. The function f can be expressed as:

$$f = \overline{x}_n f_0 \oplus x_n f_1 \tag{1}$$

$$f = f_0 \oplus x_n f_2 \tag{2}$$

$$f = f_1 \oplus \overline{x}_n f_2 \tag{3}$$

These expansions are known as Shannon (Boole), positive Davio and negative Davio respectively.

B. ESOP and ESCT Minimization

As it was stated in the introduction, finding minimal ESOP or ESCT expressions for an arbitrary Boolean function is a very difficult problem.

In this section, we present the theoretical formulation which will help us construct algorithms that find minimal ESOP or ESCT expressions for an arbitrary Boolean function.

An arbitrary completely specified Boolean function can always be written as:

$$f(x_1, \ldots, x_n) = \overline{x_n} g_1 \oplus x_n g_2 \oplus g_3 \tag{4}$$

This is because we can express subfunctions f_0, f_1 of f as following: $f_0 = g_1 \oplus g_3, f_1 = g_2 \oplus g_3$ and g_1, g_2, g_3 are Boolean functions of $x_1, x_2, \ldots, x_{n-1}$ variables. From Shannon expansion it holds: $f = x_n f_1 \oplus \overline{x_n} f_0 = x_n(g_2 \oplus g_3) \oplus \overline{x_n}(g_1 \oplus g_3) = x_n g_2 \oplus \overline{x_n} g_1 \oplus (x_n \oplus \overline{x_n}) g_3 = x_n g_2 \oplus \overline{x_n} g_1 \oplus g_3$.

In [12], [20] it is proved that each minimal ESOP expression is written in the form of equation 4. In [16] it is also proved that each minimal ESCT expression is written in the same form. The only difference is that in the first case functions g_1, g_2, g_3 must be expressed as minimal ESOP expressions, while in the second one they must be minimal ESCT expressions.

In both cases, if we find the appropriate g_1, g_2, g_3 functions (and then their minimal ESOP or ESCT expressions) we can find minimal ESOP or ESCT expressions for our input function. Since $g_1 = f_0 \oplus g_3$ and $g_2 = f_1 \oplus g_3$, the difficulty is to find the g_3 function for which the f is minimal. In order to find it we must check every possible g_3 function.

This previous observation gives us the following Theorem.

Theorem 1: Every minimal ESOP or ESCT expression for a completely specified Boolean function f of n variables can be found by merging every possible function g of $n-1$ variables with subfunctions f_0, f_1 of f.

Theorem 1 denotes that we can merge any possible g function (including the constant 0 and 1 functions) with f_0, f_1 and we will acquire at least one minimal ESCT (ESOP) expression for function f in the form: $x(f_1 \oplus g) \oplus \bar{x}(f_0 \oplus g) \oplus g$. The only difference is that in the case of ESCT expressions we must stop the recursion at the 2-variable level (since every non constant 2-variable function has ESCT weight equal to 1), instead of the 1-variable level in the ESOP case. Moreover, the constant function 1 does not add to the ESCT weight, although it does so in the ESOP case [8].

In the case of incompletely specified functions, we can take advantage of this previous theoretical formulation to find minimal ESOP or ESCT expressions, although the problem becomes much more difficult. In essence, an incompletely specified function is a set of completely specified functions (by placing don't care terms either in the ON or the OFF set of the function with all possible combinations). Therefore a naive algorithm for finding minimal ESOP or ESCT expressions would be to produce every possible completely specified Boolean function derived from the incompletely specified one and then minimize each one of them. At the end we would keep the best results. This algorithm is of course intractable for a conventional computer (especially if our input function contains many don't care terms), but as we will see in the next sections it is tractable for a quantum computer.

C. Grover's Algorithm

In [2], L. Grover presented a quantum algorithm for finding a specific element in an unsorted database in $O(\sqrt{(N)})$ steps (N is the number of elements in the database). This result is much better than its conventional analogue ($O(N)$). The initial state of the database is the superposition of all possible N elements. This is accomplished using the Walsh-Hadamard gate. In each step, Grover's algorithm increases the amplitude of the marked states (the states of the elements we are searching for) and decreases those of the unmarked ones by $O(1/\sqrt{(N)})$. After $O(\sqrt{(N)})$ steps the probability of the marked states will be almost 1 and those of the unmarked states will be almost 0. At that point, we perform a measurement and find one of the elements we are searching for. The actual behavior of Grover's algorithm depends on a specific quantum operator called the Oracle. This operator decides which states are the marked ones and which are not.

III. ALGORITHMS

A. Previous Work

Grover's algorithm can be seen as a generic framework for solving many difficult problems, for a conventional computer. This can be done by constructing an appropriate oracle operator.

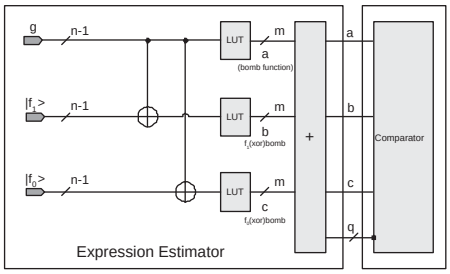

Fig. 3. QMin-Oracle Circuit

In [15] Li, Thornton and Perkowski presented a quantum algorithm based on Grover's, that could find FPRM (Fixed Polarity Reed Muller) expressions for a specific completely specified Boolean function, with number of terms less than a specific threshold. Their algorithm is actually Grover's algorithm using a special Oracle operator. This oracle operator calculates all possible FPRM expressions for the input Boolean function and then favors those expressions that have number of FPRM coefficients less than a specified threshold. At the end, Grover's algorithm will result in one of these expressions. By constantly readjusting the threshold with the new minimum number of FPRM coefficients and rerunning the Grover's algorithm, one can find minimal FPRM expressions for an arbitrary completely specified Boolean function.

In [16], a different oracle implementation is proposed in order to find minimal ESOP or ESCT expressions for an arbitrary completely specified Boolean function. In this case, the oracle is modified in the same principle as before. It is composed of two distinct parts: The Expression-Estimator and the Comparator. The first part calculates in parallel all possible ESOP or ESCT expressions and determines the number of terms (complex or product) in each one (cost function). The Comparator is the same as in [15]. This algorithm also detects the expressions that have number of terms (product or complex) less or equal to a certain threshold. The proposed quantum algorithm was named QMin.

B. DCQMin

All the previously mentioned algorithms are designed to minimize completely specified Boolean functions. Our proposed algorithm, called DCQMin, is essentially the QMin algorithm but redesigned for minimizing incompletely specified Boolean functions.

DCQMin stages are presented in Fig. 2. As it can be observed DCQMin is essentially Grover's algorithm, with modified oracle.

The DCQMin-Oracle is composed of two distinct components. The first one called the Expr-Estimator is the one that implements Theorem 1 and produces ESOP or ESCT expressions for our input function. The second one is the Comparator which compares the number of terms (complex or product) of every produced expression with a given Threshold. It returns 1, thus denoting a marked state, if this number is less or equal to Threshold and 0 otherwise, thus denoting a non marked state.

The Expr-Estimator component can be seen in Fig. 3. It is composed of three main buses. The first is initialized by the Walsh-Hadamard gates of Fig. 2 (they reside outside the Oracle operator and inside the DCQ module Fig. 5) and stands for the g function of Theorem 1. The second and the third one are initialized as the minterm representation of subfunctions f_1 and f_0 of our input function, respectively. The CNOT gates produce the bitwise XOR sum of the g function (first bus) with f_1 and f_0, respectively, thus producing the $g \oplus f_1$ and $g \oplus f_0$ functions of Theorem 1.

It is important to note that the input function to the QMin quantum circuit is in the minterm representation. Therefore it is easy to derive the minterm representations of its subfunctions.

The LUT operators that follow, give the weight of their input function and are really recursive snapshots of the bigger Expr-Estimator circuit. For ESOP expressions and functions of 1 variable, the LUT circuit can be seen in Fig. 2 (bottom level rightmost part). For ESCT expressions and functions of 2 variable, the LUT circuit can be seen in Fig. 2 (bottom level leftmost part). For all other cases the LUT circuit is really the Expr-Estimator circuit defined recursively.

At this point, the number of terms for each possible ESCT or ESOP expression of functions $g, g \oplus f_1, g \oplus f_0$ is calculated. According to Theorem 1, the number of terms in an expression of our initial function is their sum. Therefore, we use a quantum adder to perform this task. Such a quantum adder is presented in [13].

The other component of DCQMin-Oracle is the Comparator. The Comparator used, has been presented in [15]. A 2-qubit Comparator can be seen in Fig. 4. It should be noted that the Comparator is used only once and not within the recursion performed by the Expression-Estimator.

It must be noted that the q output of the DCQMin Oracle operator in Fig. 3 is the input to the Comparator operator. The a, b, c outputs correspond to the $g, g \oplus f_1, g \oplus f_0$ functions of Theorem 1 and are considered outputs of our quantum minimization circuit (Fig. 2). These outputs are simply propagated through the comparator.

The actual quantum minimization algorithm is, of course, Grover's algorithm, using the previously described DCQMin Oracle circuit as the specialized oracle operator. It takes as input (Fig. 2) an arbitrary switching function f and a Threshold. Its outputs

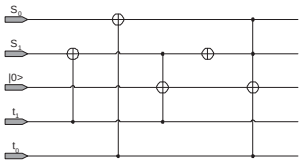

Fig. 4. A 2-qubit Comparator, comparing $s_0 s_1$ to $t_0 t_1$, implementing function: $(s_1 \oplus t_1)t_1 \oplus \overline{(s_1 \oplus t_1)}(s_0 \oplus t_0)t_0$

are the a, b, c, q outputs of the Oracle circuit. At the end of the execution, those expressions of f that have number of terms (either complex or product, depending on the type of expressions we are searching for) less or equal to the Threshold, will have, all together, probability almost 1 (marked states), while all the others will have probability almost 0 (unmarked states). Upon measuring one of its outputs (for instance the a output), all the outputs (and therefore the propagated corresponding inputs) will collapse to those corresponding to one of the marked states. The weight of the output expression will be given by q, while the actual expression can be reconstructed from outputs a, b, c according to Theorem 1.

The DCQ module Fig. 2, receives as input the incompletely specified Boolean function and produces its associated set of completely specified Boolean functions as described in subsection II-B. All these functions are in superposition and they are fed to the main quantum circuit. DCQMin uses Grover's algorithm to find which of these functions has a solution with the least possible number of product (ESOP case) or complex (ESCT case) terms and returns one of them.

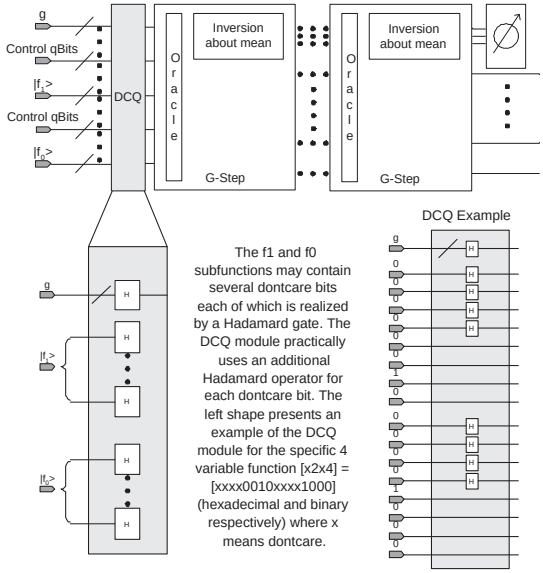

Fig. 5. DCQmin-Oracle Circuit with example

An incompletely specified Boolean function can be described as a Boolean function where some minterms are either 0 or 1. This means that we cannot, initially, decide the actual value for these minterms. In quantum circuits this is equivalent as having a qubit to be in either the $|0>$ or the $|1>$ state, in equal probability. The input to the DCQ is the minterm formulation of the incompletely specified Boolean function where each qubit represents the presence or not of its corresponding minterm. Some of these qubits represent don't care minterms. Putting those qubits in superposition of their basic states is equivalent as declaring their corresponding minterms as don't care minterms. This is what the DCQ module really does. It adds a Walsh-Hadamard gate for every such qubit, thus putting those qubits in superposition of their basic states.

An example of the DCQ module for the 4 variable function $[x2x4]$ (where x represent a don't care minterm) is shown in Fig. 5.

The DCQMin oracle circuit has been simulated successfully using the Fraunhofer Quantum Computing Simulator [14].

IV. Conclusions and future work

In this work quantum algorithm QMin [16] is extended for minimizing incompletely specified Boolean functions. DCQMin is a quantum algorithm that receives as input an arbitrary incompletely specified Boolean function and detects its ESOP or ESCT expressions with number of terms (complex or product) less than a specified threshold. It is obvious that by repeativelly executing DCQMin and updating the Threshold as necessary we can find minimal expressions for a specific function. An initial

estimation for the Threshold can be obtained from conventional heuristic minimizers such as exorcism-4 [18], QuiXor [19] or QuickDCMIN [21] (ESOP case) or EMin1 [7] (ESCT case).

Future work will focus on extending the algorithm in order to address the multi-output switching function minimization problem.

V. Acknowledgments

This work has been funded by the project PENED 2003. This project is part of the OPERATIONAL PROGRAMME ”COMPETITIVENESS” and is co-funded by the European Social Fund (75%) and National Resources (25%).

References

[1] P. W. Shor, ”Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer”, SIAM J. Computing 26, pp. 1484-1509 (1997).
[2] L.K. Grover, ”A fast quantum mechanical algorithm for database search”, Proc. 28th Ann. ACM Symp. on Theory of Comput., 212219, 1996.
[3] D. E. Knuth ”The Art of Computer Programming”, Vol. 2: Seminumerical Algorithms, Second ed., Addison-Wesley, 1981.
[4] A. Mishchenko, M. Perkowski, ”Logic Synthesis of Reversible Wave Cascades”,International Workshop on Logic And Synthesis 2002, New Orleans, Louisiana, June 4-7, 2002.
[5] G. Papakonstantinou, ”Synthesis of cutpoint cellular arrays with exclusive-OR collector row”, Electronic Letters, 13(1977).
[6] D. Voudouris, S. Stergiou, G. Papakonstantinou ”Minimization of reversible wave cascades”, IEICE Trans. on Fund., Vol E88-A, No. 4, pp. 1015-1023, 2005/04.
[7] D. Voudouris, G. Papakonstantinou ”Maitra Cascade Minimization”, 6th IWSBP, 2005, Freiberg (Sachsen), Germany.
[8] D. Voudouris, M. Sampson, G. Papakonstantinou ”Exact ESCT Minimization for functions of up to six input variables”, submitted for publication to Integration, The VLSI Journal, Elsevier (under minor revision).
[9] D. Voudouris, M. Kalathas, G. Papakonstantinou ”Decomposition of Multi-output Boolean Functions”, HERCMA 2005, Athens, Hellas.
[10] G. Lee ”Logic synthesis for celullar architecture FPGA using BDD”, ASP-DAC 97, pp 253-258 Jan 1997.
[11] K.K. Maitra ”Cascaded switching networks of two-input flexible cells” IRE Trans. Electron. Comput., pp, 136-143, 1962.
[12] A. Gaidukov, ”Algorithm to derive minimum esop for 6variable function”, 5th IWSBP, September 2002.
[13] Phil Gossett, ”Quantum Carry Save Arithmetic”, quant-ph/980861 (1998)
[14] http://www.qc.fraunhofer.de/
[15] Lun Li, Mitch Thornton and Marek Perkowski ”A Quantum CAD Accelerator based on Grover’s algorithm for finding the minimum Fixed Polarity Reed-Muller form”, ISMVL’06, Proc. of the ISMVL’06 vol. 00, pp. 33- 33, 17-20 May 2006.
[16] M. Sampson, D. Voudouris, G. Papakonstantinou, ”A Quantum Algorithm for Finding Minimum Exclusive-Or Expressions,” ISVLSI, pp. 416-421, IEEE Computer Society Annual Symposium on VLSI (ISVLSI ’07), 2007.
[17] Song, N., Perkowski, M.A., ”EXORCISM-MV-2: minimization of exclusive sum of products expressions for multiple-valued input incompletely specified functions”, ISMVL1993, Proc. of ISMVL93, pp.132-137, 24-27 May 1993.
[18] A. Mishchenko, M. Perkowski ”Fast Heuristic Minimization of Exclusive-Sums-of-Products”, 5th International Reed-Muller Workshop, Starkville, Mississippi, August, 2001
[19] Stergiou S., Voudouris D., Papakonstantinou G., ”Multiple-Value Exclusive-Or Sum-Of-Products Minimization Algorithms”, IEICE Trans. on Fund., 2004, vol 87, part 5, pp. 1226-1234.
[20] T. Hirayama, Y. Nishitani, T. Sato ”A Faster Algorithm of Minimizing AND-EXOR Expressions”, IEICE Trans. on Fund., Vol E85-A, No. 12, pp. 2708-2714, 2002/12.
[21] M. Kalathas, D. Voudouris, G. Papakonstantinou A heuristic algorithm to minimize ESOPs for multiple output incompletely specified functions, GLSVLSI 2006, Philadelphia, USA, 2006.
[22] T. Kozlowski, E. L. Dagless, J. M. Saul An enhanced algorithm for the minimization of exclusive-or sum-of-products for incompletely specified functions, 1995 IEEE Intrn. Conf. on Computer Design (ICDD95), pp. 244, 1995.
[23] N. Song, M. A. Perkowski Minimization of exclusive sum-of-products expressions for multiple-valued input, incompletely specified functions, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 15, Nr. 4, April 1996.
[24] G. Lee, R. Drechsler ETDD-based Synthesis of term-based FPGAs for incompletely specified boolean functions, ASP-DAC 1998.