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

M. Sampson (sampson@cslab.ntua.gr), D. Voudouris (dvoudour@cslab.ntua.gr), G. Papakonstantinou (papakon@cslab.ntua.gr)
Dept of Electrical and Computer Engineering
Division of Computer Science
Computing Systems Laboratory
National Technical University of Athens, tel. 2107721529/Fax 2107721533, ZIP 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 multi-output switching function that is not necessarily completely specified. 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.**

keywords: quantum computing, logic design [1]

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.

[1] Preliminary version

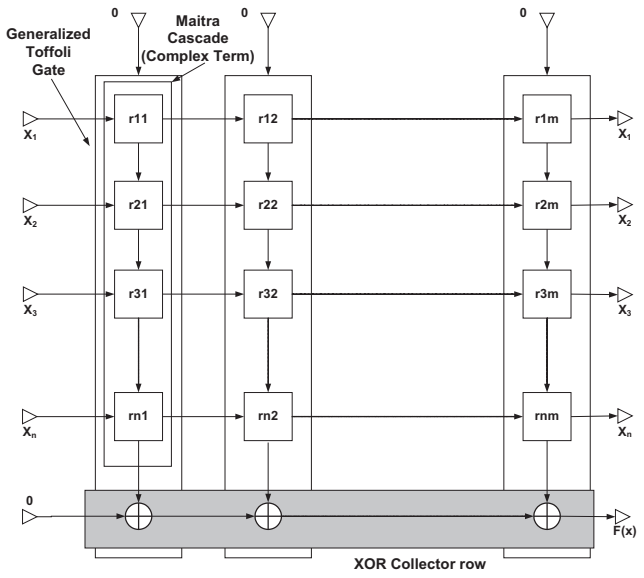

Fig. 1. Reversible wave cascade CA

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 or incompletely specified switching function. Finding minimal ESCT or ESOP expressions for a multi-output 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], [22], [21], [13]. Others have been designed in order to detect almost minimal ESOP or ESCT expressions but for more input variables [4], [9], [10], [21], [13], [20]. Algorithms for finding almost minimal ESCT or ESOP expressions for incompletely specified functions are significantly less [23], [24], [25], [26].

Finding minimal solutions for ESOP and ESCT expressions is presently limited to only 6 input variables, using conventional computers [8], [13], [22]

In [16], [17] and [18] the Grover's algorithm [2] is used as a framework to minimize XOR expressions.

In this work the QMin algorithm [17] is extended in order to detect minimal ESCT or ESOP expressions for multi-output incompletely specified Boolean functions (QMin is designed for completely specified single-output Boolean functions). To the best of the authors' knowledge, this is the first quantum algorithm that detects minimal ESCT or ESOP expressions for a multi-output incompletely specified Boolean function.

II. Theoretical Background

A. Definitions

In this section we provide some background definitions.

Definition 1: A boolean function with n-input variables and m-outputs (multi-output function) is a mapping: $f : \{0,1\}^n \rightarrow \{0,1\}^m$. Variables $x_1, \ldots, x_n$ are called the support of f. If $m = 1$ then f is a single-output boolean function. Generally, a multi-output boolean function can be considered as m different single-output functions.

Definition 2: Let $V_i = \{0,1\}$ and $T = \{0,1,x\}$. A multi output incompletely specified Boolean function with n input variables and m outputs, is a mapping $f : V_1 \times \ldots \times V_n \rightarrow T^m$. The minterms for which $f(X_1, \ldots, X_n) = x$ are the don't care minterms of the function (DC-set). In these cases the value of f (and its specified output) 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. Combining the above, we can produce the minterm representation of f.

More specifically, a Boolean function can be represented in many different ways (these are called expressions). One of them is using the list of its minterms. This representation is called the minterm representation of the function (MT).

The following definitions concern multi-valued Boolean logic.

Definition 3: Let X be variable that takes a value from $V = \{0, \ldots, v-1\}$ and $S \subseteq V$. Then X^S is a literal of X such as $X^S = 1$ when $X \in S$ and $X^S = 0$ when $X \in V \backslash S$.

Definition 4: A multi-valued input, binary output function f is a mapping $f : V_1 \times V_2 \times \cdots \times V_n \rightarrow \{0,1\}$, where $V_i = \{0, \ldots, v_i - 1\}$.

In this paper it is sufficient to examine mappings of the form $f : \{0,1\} \times \cdots \times \{0,1\} \times \{0, \ldots, v-1\} \rightarrow \{0,1\}$.

In order to find minimal ESOP or ESCT expressions for a multi-output Boolean functions we will transform it to a single-output Boolean function having its less significant input variable multi-valued.

In accordance to the MT representation for single-output Boolean functions depending on binary input variables, we can define the MVMT [21] formulation for single-output Boolean functions which have their less significant input variable multi-valued. Instead of giving a formal definition for this representation we will demonstrate it with the following example.

Let $f : \{0,1\} \times \{0,1\} \times \{0, \ldots, v-1\} \rightarrow \{0,1\}$ ($v = 3$). Intuitively, the MVMT of f can be seen as the interleaving of v MTs of 2-variable functions f^i, where $f^i = f(x_1, x_2, i)$. Let $f^i = [d], [1], [c]$ for $i = 0, 1, 2$ respectively. Then the MVMT of f is [101101000111]. More specifically (in terms of bits): $f^0 = [d] = [1101]$, $f^1 = [1] = [0001]$, $f^2 = [c] = [1101]$.

The last digit of the MVMT (3) is produced by using the last digits of the MT formulations of f^2, f^1, f^0. The rest of the digits of MVMT are produced in similar way from the corresponding digits of the MT formulations of f^2, f^1, f^0. This is demonstrated in Fig. 3.

In the rest of this paper we will declare a MVMT representation of a function by enclosing it in brackets.

Definition 5: A subfunction $f_i, i = 0, 1, 2$ of a boolean function $f(x_1, \ldots, x_n)$, regarding the Boolean variable x_1 of f's support is defined as:

- $f_0 = f(0, x_2, \ldots, x_n)$
- $f_1 = f(1, x_2, \ldots, x_n)$
- $f_2 = f_0 \oplus f_1$

It is very easy to prove that the upper half of the MT (or MVMT) representation of a switching function corresponds to its f_1 subfunction, while the lower half corresponds to its f_0 subfunction. The f_2 subfunction can, obviously, be produced as the XOR sum of f_0, f_1.

Definition 6: Let x_i be binary variable literals, y a binary value (constant input) and G_i arbitrary 2-input 1-output boolean functions ($1 \leq i \leq n$). Then $U = G_n(x_n, G_{n-1}(x_{n-1}, G_{n-2}(x_{n-2}, \ldots, G_1(x_1, y))))$ is an n-variable complex term (or Maitra term) that depends on variables $x_1, \ldots, x_n$. Functions G_i will be called the **ESCT cell functions** of the term.

The G_i ESCT cell function can be any single-output two-input function. It has been proved in [12] that it is sufficient for G_i to be any of the six functions $x + y$ (cell 1), $\overline{x} + y$ (cell 2), $\overline{x}y$ (cell 3), xy (cell 4), $x \oplus y$ (cell 5), y (cell 6) which we will call cell set for the rest of the paper.

A product term is a special case of a complex term where the $G_i(x, y)$ function may be of the form: $xy, \bar{x}y, x\bar{y}, \bar{x}\bar{y}, x, y, 0, 1$. If the last four cases are not allowed then the product term is actually a minterm.

Definition 7: 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:

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

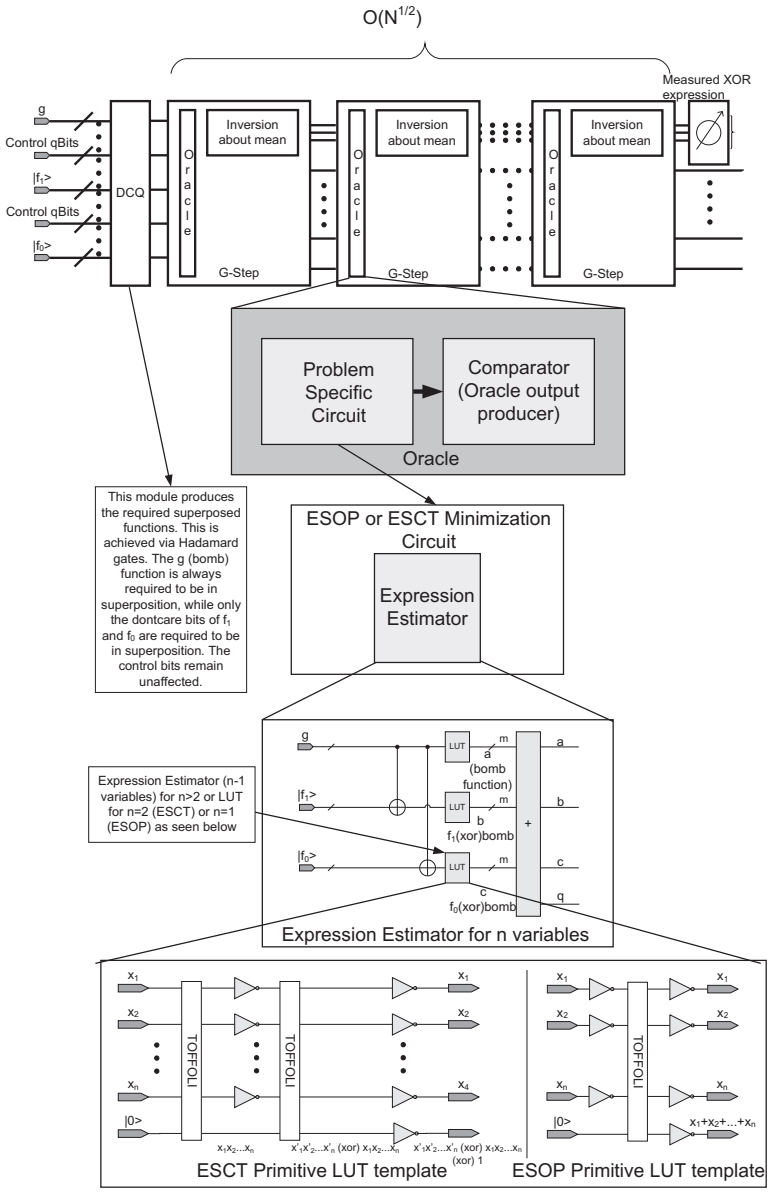

Fig. 2. MOQmin Algorithm hierarchy

where M_i are complex terms and m is their number in 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.

In the previous definition, if we use only product terms (cells 3,4,6), instead of complex terms, then the produced expression is called an *Exclusive or Sum Of Products* (ESOP) expression.

An ESCT expression can be directly mapped to a special cellular architecture, called the reversible wave cascade cellular architecture which can be seen in Fig. 1.

***Definition** 8:* A minimal (or exact) ESCT (ESOP) expression of a single-output switching function $f(x_1, \ldots, x_n)$ of n variables, is defined as the ESCT (ESOP) expression which has the fewest number of complex terms comparing to any other ESCT (ESOP) expression for this function.

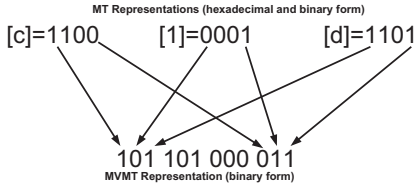

Fig. 3. MVMT example

The same definition applies for multi-output boolean functions, but in this case different outputs may share common terms, in order to reduce the overall weight.

***Definition** 9:* The ESCT (ESOP) weight $w(f)$ of a switching function $f(x_1, \ldots, x_n)$ of n variables is defined as the number of complex terms in a minimal ESCT (ESOP) expression of f.

Since the complement of a complex term is also a complex term [6], it holds: $w(f) = w(\bar{f})$.

The ESCT and ESOP weight of a multi-valued switching function is defined in accordance to Definition 9.

***Definition** 10:* Let x_i be binary literals, Y takes a value from $\{0, \ldots, v-1\}$, G_i $(2 \leq i \leq n)$ is a boolean function $\{0,1\} \times \{0,1\} \rightarrow \{0,1\}$ from the cell set, and G_1 is an arbitrary mapping $\{0,1\} \times \{0, \ldots, v-1\} \rightarrow \{0,1\}$. Then $U = G_n(x_n, G_{n-1}(x_{n-1}, G_{n-2}(x_{n-2}, \ldots, G_1(x_1, Y))))$ is an mv-term.

The $G_i, i = 2, \ldots, n$ are the ESCT cell functions as previously defined. The G_1 mapping is defined in accordance to these functions and its MVMT representation is produced as the interleaving of the MT representations of each bit in the Y multi-valued variable when applied to the corresponding ESCT cell function of the cell set. The term

$G_n(x_n, G_{n-1}(x_{n-1}, G_{n-2}(x_{n-2}, \ldots, G_2(x_2, z))))$, where $z = 0, 1$, is called the 2v-term of mv-term U and the Y multi-valued variable is called the mv-var of U.

The complement of an mv-term (like the complement of a complex term) is also one mv-term and has complemented 2v-term and complemented mv-var.

A XOR sum of mv-terms is a perfect way to represent an arbitrary single-output Boolean function having its less significant variable multi-valued.

***Definition** 11:* 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:

$$f = \overline{x}_n f_0 \oplus x_n f_1 \quad (1)$$

$$f = f_0 \oplus x_n f_2 \quad (2)$$

$$f = f_1 \oplus \overline{x}_n f_2 \quad (3)$$

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

It must be noted that the above decompositions are valid even for multi-valued single output boolean functions as long as the x variable is binary.

B. ESOP and ESCT Minimization for multi-output Boolean functions

In [8] algorithm XMin6 was proposed that was able to produce minimal ESCT expressions for single-output Boolean functions of up to six input binary variables. A generalization of XMin6 (regarding the number of input variables of its input functions), for functions having their less significant variable multi-valued, can be summarized to the pseudo-algorithm XMin. This algorithm has a flag (output_expression) which distinguishes whether the output expression will be ESOP or ESCT. In [8] all the necessary theoretical background for this algorithm is presented in detail.

procedure XMin(func, output_expression_isESCT: Boolean)
Begin
$w(func) = \infty$;
If func == constant 0
    w(func) = 0;
If func == constant 1 **And** output_expression_isESCT
    w(func) = 0;
Else If func.number_of_variables == 1
    w(func) = 1;
Else
    foreach possible g
    Begin
        $w(g) = XMin(g)$;
        $w(f_0 \oplus g) = XMin(f_0 \oplus g)$
        $w(f_1 \oplus g) = XMin(f_1 \oplus g)$
        $w(func) = MIN(w(func), w(f_0 \oplus g) + w(g_1 \oplus g) + w(g))$;
    End
EndIf
End

C. Grover's Algorithm

In [2], L. Grover presented in detail 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 ($O(N)$).

In our case, the Oracle of the original algorithm has to be designed to find itself the appropriate marked states. Taking this under consideration, we have designed a different Oracle which will be described in detail in the following sections.

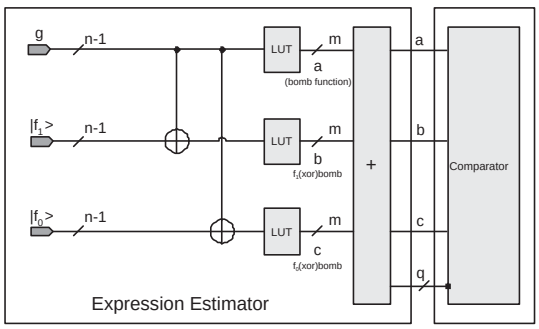

Fig. 4. QMin-Oracle Circuit

III. ALGORITHMS

A. Previous Work

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

In [16] 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 [17] and [18], different Oracle implementations are proposed in order to find minimal ESOP or ESCT expressions for an arbitrary completely or incompletely specified Boolean function. In those cases, 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 [16]. This algorithm also detects the expressions that have number of terms (product or complex) less or equal to a certain threshold. In the second case there is also an extra initialization step in order to address the incompletely specified functions. The proposed quantum algorithms were named QMin and DCQMin respectively.

B. MOQMin

All the previously mentioned algorithms are designed to minimize completely or incompletely specified single output Boolean functions. Our proposed algorithm, called MOQMin, addresses a more generalized aspect of the ESOP and ESCT minimization problem, the minimization of incompletely specified multi-output Boolean functions. In this way, we have a complete framework for the ESCT and ESOP minimization problem, which can be easily customized during initialization in order to address every variation of it. This problem is particularly difficult to solve in a conventional computer not only due to combinatorial explosion introduced by the single output minimization problem itself, but also by the fact that minimality of a multi-output Boolean function is not guaranteed if the corresponding single-output functions that compose the multi-output function are minimized independently. This is very obvious if taken under consideration the fact that even the individual minimal single output functions can share common terms. In this way, the problem has to be approached in a more generalized way, a fact that adds more complexity to the algorithm. The framework of the Grover's algorithm has proved to be very useful in such cases.

A hierarchical diagram of MOQMin algorithm is presented in Fig. 2. As it can be observed, MOQMin is essentially Grover's algorithm, with modified Oracle. The input of the algorithm is the characteristic function of the multi-output function in minterm representation.

The MOQMin-Oracle is composed of two distinct components. The first one called the Expr-Estimator is the one that implements the weight estimation process and produces ESOP or ESCT expressions for our input function (the for-loop in the XMin pseudo-code). The second one, is the Comparator [16], 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 than Threshold and 0 otherwise, thus denoting a non marked state.

The Expr-Estimator component can be seen in Fig. 4. It is composed of three main buses. The first is initialized by the Walsh-Hadamard gates that are deployed by the DCQ module of Fig. 2 and stands for the g function used in the expressions of XMin pseudo-code. Please note that the DCQ module will always deploy Walsh-Hadamard gates for every qbit of g. The second and the third one, are initialized as the minterm representation of subfunctions f_1 and f_0 of our input characteristic function, respectively. The DCQ module enables us to apply the algorithm for incompletely specified Boolean functions in the same fashion as in [18], that is by applying Walsh-Hadamard gates in the qbits of the input functions that correspond to don't care minterms. The CNOT gates produce the bitwise XOR sum of the g function (first bus) with f_1 and f_0, respectively, thus producing the required $g \oplus f_1$ and $g \oplus f_0$ functions.

It is important to note that the input characteristic function to the MOQMin quantum circuit is in the MVMT representation. Therefore it is very easy to derive the minterm representations of its subfunctions by taking the leftmost (f_1) and the rightmost (f_0) part.

The LUT operators that follow, give the number of terms of their input function and are in fact, recursive snapshots of the bigger Expr-Estimator circuit. As we decompose the characteristic input function, the LUT subcircuits are composed of the entire Expression Estimator of the next level of decomposition. This recursive process ends at the level of the multivalued variable of the characteristic function. At this point, the LUT subcircuits (primitive LUTs) are explicitly defined with primitive quantum gates.

The explicit implementation of these subcircuits depends on the number of outputs of the initial Boolean function, but in principle follows a certain pattern. For the case of ESOP expressions, the logical OR functions needs to be implemented because the weight always equals to 1 unless the input minterm equals to 0. This can be easily implemented using a Toffoli gate with the control qbit equal to $|0>$ and the necessary NOT gates. For the case of ESCT expressions the subcircuit becomes a little bit more complicated. The only two cases where the weight is 0 is when the input minterm is either all ones or zeros, otherwise the weight is 1. In this case, the LUT can be implemented by two consecutive Toffoli gates an the appropriate NOT gates in between. The number of qbits of such LUT circuits depends on the number of outputs of the initial circuit.

For ESOP expressions, the primitive LUT circuit can be seen in Fig. 2 (bottom, rightmost part). For ESCT expressions, the primitive LUT circuit can be seen in Fig. 2 (bottom, leftmost part). For all other cases the LUT circuit is actually the Expr-Estimator circuit defined recursively.

When the LUT operators are applied, the number of terms

for each possible ESCT or ESOP expression of functions $g, g \oplus f_1, g \oplus f_0$ is calculated. 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 [14]. This sum is produced in the same single step for all the possible g functions due to superposition.

The other component of MOQMin-Oracle is the Comparator. The Comparator used, has been presented in [16]. A 2-qubit Comparator can be seen in Fig. 5. It should be noted that the Comparator is used only once and not within the recursion performed to construct the Expression-Estimator for the initial expression.

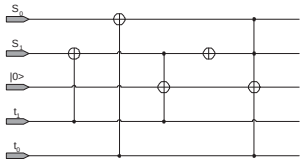

Fig. 5. A 2-qubit Comparator, comparing $s_0 s_1$ to $t_0 t_1$, implementing function: $(s_1 \oplus t_1)t_1 \oplus \overline{(s_1 \oplus t_1)}(s_0 \oplus t_0)t_0$

The q output of the MOQMin Oracle operator in Fig. 4 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 (see XMin pseudo-code) and are considered outputs of our quantum minimization circuit (Fig. 2). These outputs are simply propagated through the comparator.

Finally, the DCQ module at the beginning of the circuit is used to initialize the quantum registers that contain the MT of g, f_1 and f_0 appropriately. More accurately, Hadamard gates are applied to all the qbits representing g so that the required superposition of all possible states to be acquired. On the other hand, Hadamard gates are applied only to the qbits of the f_1, f_0 that correspond to the DC-Set minterms (the don't-care minterms). In that way, a superposition of all possible minterm combinations is produced and the circuit has the ability to handle incompletely specified functions [18].

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

All the above steps of the algorithm can be summarized by the following pseudo-code:

procedure MOQMin(func)

Begin

for (each step of the Grover's Algorithm)

Begin

do in parallel for every possible g function

//one step

$w(g) = Expr\_Estimator(g);$

$w(f_0 \oplus g) = Expr\_Estimator(f_0 \oplus g)$

$w(f_1 \oplus g) = Expr\_Estimator(f_1 \oplus g)$

End

$mark(func) = threshold > (w(f_0 \oplus g) + w(g_1 \oplus g) + w(g));$

$invertmarkedstates()$

$invertaboutmean()$

End

End

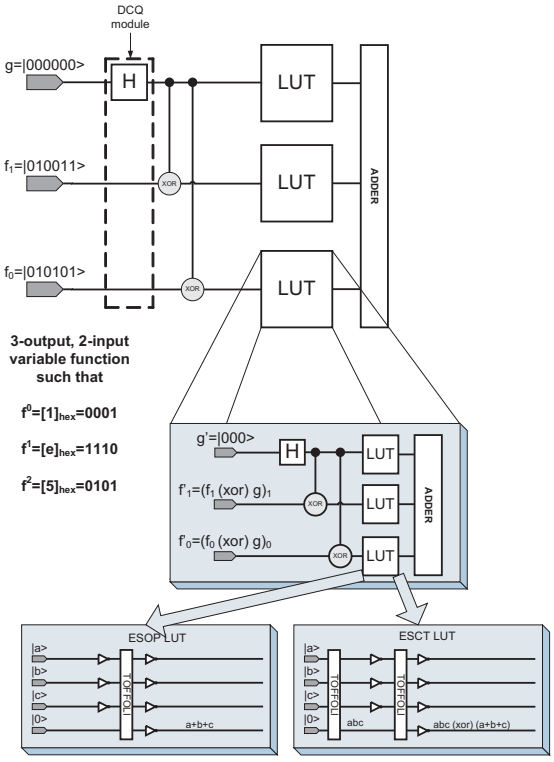

Fig. 6. Oracle example for both the ESCT and ESOP case

For example, let $f(x_1, x_2) \rightarrow \{f^0, f^1, f^2\}$ be a 2 input, 3 output completely specified Boolean function such that $f^0 = [1] = 0001, f^1 = [e] = 1110, f^2 = [5] = 0101$ (in hex and binary notation respectively). The characteristic function is $[010011010101]$ in binary notation. After the application of the Hadamard operator, the g register will contain a superposition of all 2^6 possible functions. Consequently, right before the LUTs, the other two registers for f_1 and f_0 will contain the superposed xor-sum of the the initial f_1 and f_0 with all possible g functions. In other words, in this example we have:

$f_1 = 010011$

$f_0 = 010101$

$g =$a superposition of all 2^6 possible functions.

As it can be confirmed by conventional algorithms [21] and [9], the combination $g = |010101>$, $f_1 \oplus g = |000110>$, $f_0 \oplus g = |000000>$ provides a minimal solution of ESCT with weight equal to 2 and for the ESOP case the minimum weight is 3, and can be obtained by the combination $g = |010011>$, $f_1 \oplus g = |000110>$, $f_0 \oplus g = |000000>$.The diagram in Fig. 6 depicts the MOQMin-Oracle that can be used to minimize this particular function for the ESCT or ESOP case.

The MOQMin Oracle circuit has been simulated successfully using the Fraunhofer Quantum Computing Simulator [15].

IV. Complexity estimation

As it can be easily derived from the pseudo-code for XMin, the time complexity of the conventional algorithm lies in the main loop. If $func$ is an n-input variables function then the time complexity is $O(3 \cdot 2^{2^{n-1}} \cdot 3 \cdot 2^{2^{n-2}} \ldots 3 \cdot 2^{2^0}) = O(3^n \cdot 2^{\sum_{i=0}^{n-1} 2^i}) = O(3^n \cdot 2^{2^n - 1}) = O(3^n \cdot 2^{2^n})$. The factor 3 is derived by the 3 recursive calls of XMin in the pseudo-code, while the $2^{2^{n-i}}$ factors, where $i = 0 \ldots n-1$, are derived from the number of all possible g functions in every level of the decomposition. On the other hand the estimated time complexity of MOQMin is $O(\sqrt{2^K}(1 + (1 + (\ldots + (1 + 3 + K_0 log(K_0)) + \ldots) + K_{n-1} log(K_{n-1})) + K_n log(K_n)) + C) = O(2^{K/2} \cdot (n + \sum_{i=0}^{n} K_i log(K_i)))$ where K_i and K are the number of qbits required to represent the weight of the function at each level of the recursion and C the number of steps required by the comparator. In particular, the $\sqrt{2^K}$ factor is derived from Grover's algorithm complexity. Moreover, the other factor of the initial complexity formula can easily be derived by the recursive definition of the LUT operators. For example, $O(1+3+K_0 log(K_0))$ is the complexity of the final LUT (1 step for the CNOT operators, 3 steps on average for either the ESOP or ESCT LUT and $K_0 log(K_0)$ steps for the adder). Using the Shannon expansion [8] we can easily derive the worst upper bound for K or K_i for functions of more than 1 input variables. It holds that $K \leq log(2^n) \leq n$. As a result, MOQMin's time complexity is $O(2^n)$. Of course, there is a significant speed-up (the complexity drops from double exponential to exponential), but the complexity of the problem remains exponential.

V. Conclusions and future work

In this work quantum algorithms QMin [17] and DC-QMin [18] are extended for minimizing multi-output Boolean functions. MOQMin is a quantum algorithm that receives as input the characteristic of an arbitrary multi-output function and detects its ESOP or ESCT expressions with number of terms (product or complex respectively) less than a specified threshold. It is obvious that by repeatedly executing MOQMin 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 [20], QuiXor [21] or QuickDCMIN [23] (ESOP case) or EMin1 [7] (ESCT case).

Future work will deal with other types of expressions such as and-or, and-or-xor. Another issue that needs to be addressed is whether there is a more efficient way to perform the above minimization.

VI. 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", Elsevier Integr. VLSI J. 41, 1 (Jan. 2008), 87-105. DOI= http://dx.doi.org/10.1016/j.vlsi.2007.01.003.

[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] R. C. Minnick, "Cutpoint cellular logic" IEEE Trans. Electron. Comput., vol. EC-13, Dec, 1964, pp. 685-698.

[13] A. Gaidukov, "Algorithm to derive minimum esop for 6variable function", 5th IWSBP, September 2002.

[14] Phil Gossett, "Quantum Carry Save Arithmetic", quant-ph/980861 (1998)

[15] http://www.qc.fraunhofer.de/

[16] 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.

[17] 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.

[18] M. Sampson, D. Voudouris, M. Kalathas, G. Papakonstantinou, "A Quantum Algorithm for finding Minimum Exclusive-Or Expressions for incompletely specified Boolean Functions", HERCMA 2007, Athens, 2007.

[19] 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.

[20] A. Mishchenko, M. Perkowski "Fast Heuristic Minimization of Exclusive-Sums-of-Products", 5th International Reed-Muller Workshop, Starkville, Mississippi, August, 2001

[21] 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.

[22] 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.

[23] M. Kalathas, D. Voudouris, G. Papakonstantinou A heuristic algorithm to minimize ESOPs for multiple output incompletely specified functions, GLSVLSI 2006, Philadelphia, USA, 2006.

[24] 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.
[25] 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.
[26] G. Lee, R. Drechsler ETDD-based Synthesis of term-based FPGAs for incompletely specified boolean functions, ASP-DAC 1998.