Ant Colony Direct Cover Technique for Multi-Level Synthesis of Multiple-Valued Logic Functions

Mostafa Abd-El-Barr, *Senior Member, IEEE*

***Abstract*—** A number of heuristic algorithms for producing near minimal sum-of products realization of multiple valued logic (MVL) functions have been introduced in the literature. In particular, the direct cover (DC) algorithms have been used effectively for synthesis of MVL functions for implementation using 2-level programmable logic arrays (PLAs). In this paper, a hybrid ant colony (ACO) algorithm and DC technique to synthesize MVL functions is introduced. The results are compared to other techniques found in the literature. A benchmark set of 50000 randomly generated 2-varaible 4-valued functions is used to test the results obtained using the proposed algorithm. It is shown that the results obtained using the introduced hybrid ACO-DC technique are superior to those produced by existing techniques in terms of the average number of product terms needed for synthesis of a given MVL function.

***Index Terms*—** Ant Colony, Direct Cover Algorithms, Multiple-Valued Logic, Functional Synthesis.

I. INTRODUCTION

Application of multi-valued (non-binary) digital signals can provide considerable relief for a number of problems faced using the binary systems. Increased information density and processing efficiency of circuits could theoretically be substantially increased without any drastic increase in the cost of the underlying fabrication technology through the use of Multiple-Valued Logic (MVL). The use of non-binary data storage (ROM, RAM, Flash Memory) has led to reduction in on-chip physical space as compared to the use of binary data storage.

It should however be noted that the MVL synthesis problem is more involved compared to its binary counterpart. Consider, for example, synthesis of 2-variable 4-valued functions. There are $r^{(r^n)} = 4^{4^2} = 2^{32}$ such functions. A number of heuristic algorithms for producing near-minimal sum-of products (PLA) realization of MVL functions have been introduced [1-8]. In addition, a number of proposed MVL PLA realizations have also been proposed [9-12]. Iterative heuristics offer the possibility of exploring larger solution space in arriving at near-optimal solutions. A number of these techniques have been reported in the literature [13-17].
In this paper, a hybrid of ACO-DC algorithm to synthesize MVL functions is introduced. To achieve further reduction to the number of gates needed to represent the function, an additional logic level is added at the output of circuits' structure. The proposed technique works by decomposing a given MVL function using ACO and synthesizing a simpler circuit using a selected DC algorithm. A benchmark set of randomly generated 2-varaibale 4-valued function has been used to test the results obtained using the proposed hybrid ACO-DC algorithm and o compare the obtained results with those obtained using existing techniques in terms of the average number of product terms needed for synthesis of a given MVL function.

Manuscript received May 13, 2009. This work is financially supported by Kuwait University under Research Grant # WI 02/07. The author gratefully acknowledge such support.

Mostafa Abd-El-Barr is with the Department of Information Science, CFW, Kuwait University, Mostafa@cfw.kuniv.edu

This paper is organized as follows. Some background material on MVL, and direct cover techniques are presented in Section 2. The proposed technique is introduced in Section 3. Section 4 describes the experiments, results and comparison with other techniques. Section 5 concludes the paper.

II. BACKGROUND MATERIAL

An n-variable r-valued function, $f(X)$, is defined as a mapping $f{:}R^n \rightarrow R$ where $R=\{0,1,\ldots,r-1\}$ is a set of r logic values with $r \geq 2$ and $X=\{x_1,x_2,\ldots,x_n\}$ is a set of n r-valued variables.

Definition 1: A *tsum* (truncated sum) operator is defined as $\text{tsum}(a_1,\ldots,a_n) = a_1 \oplus \cdots \oplus a_n, = \min(a_1,+\ldots+ a_n, r\text{-}1)$, where $a_i \in R$.□

Definition 2: *A window literal* ${}^a x^b$ of an MVL variable x is equal to r-1 if $a \leq x \leq b$ & 0 otherwise, $a,b \in R$ and $a \leq b$. □

Definition 3: A *product term* (PT), $P(x_1,\ldots,x_n)$ is defined as the minimum of a set of *window literals* on variables $x_1,\ldots,x_n$, i.e. $P(x_1,\ldots,x_n) = c \bullet {}^{a1}x_1^{\,b1} \bullet\cdots\bullet {}^{an}x_n^{\,bn} = \min(c, {}^{a1}x_1^{\,b1}, {}^{an}x_n^{\,bn})$, where $a_i, b_i \in R$ and $a_i \leq b_i$.and $c \in \{1,2,\ldots, r\text{-}1\}$. □

In the above definition, c is called the value of the PT.

Definition 4: For an MVL function $f(x_1,\ldots,x_n)$, an assignment of values to variables $x_1=a_1,\ldots, x_n=a_n$ is called a *minterm*, iff: $f(a_1,\ldots,a_n) \neq 0$, where $a_i \in \{0,1,\ldots,r\text{-}1\}$. □

A *minterm* is a special case of a *product term* consisting of *literal* and *min* operators where the PT is dependent on all variables and $a_1 = b_1, \ldots, a_n = b_n$. Consider, for example, the 4-valued 2-variable function shown in Fig. 1. Some of the minterms are $1\bullet{}^3\mathbf{X}_1{}^3\bullet{}^0\mathbf{X}_2{}^0$, $2\bullet{}^1\mathbf{X}_1{}^1\bullet{}^0\mathbf{X}_2{}^0$ and $3\bullet{}^2\mathbf{X}_1{}^2\bullet{}^1\mathbf{X}_2{}^1$.

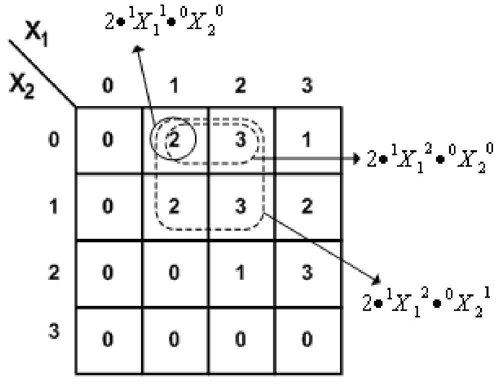

X_1
X_2
$2\bullet{}^1X_1{}^1\bullet{}^0X_2{}^0$
$2\bullet{}^1X_1{}^2\bullet{}^0X_2{}^0$
$2\bullet{}^1X_1{}^2\bullet{}^0X_2{}^1$
0 1 2 3
0: 0 2 3 1
1: 0 2 3 2
2: 0 0 1 3
3: 0 0 0 0

Fig. 1: A tabular representation of f(x$_1$,x$_2$).

Definition 5: An *implicant* of a function $f(x_1,\dots,x_n)$, is a PT, $I(x_1,\dots,x_n)$, such that $f(x_1,\dots,x_n) \geq I(x_1,\dots,x_n) \geq$ for all assignments of x_i's. □

In Fig. 1, $3\bullet{}^2\mathbf{X}_1{}^2\bullet{}^0\mathbf{X}_2{}^1$ and $2\bullet{}^1\mathbf{X}_1{}^2\bullet{}^0\mathbf{X}_2{}^1$ are examples for *implicants*.

Direct cover (DC) approaches for synthesis of MVL functions consist of the following main steps:

1) choose a minterm,
2) identify a suitable implicant that covers that minterm,
3) obtain a reduced function by removing the identified implicant, and
4) repeat steps 1 to 3 until no more minterms remain uncovered.

The DC approaches reported in the literature differ in the way step 1 and 2 are achieved. For example, the algorithm due to Armstrong [3] selects minterms randomly and selects the implicant which results in the largest number of zero minterms (LRZ). The algorithm due to Besslich [1] uses what is known as the *isolation weight* (IW) for selecting minterms and selects the implicant that leads to minimizing the cost of the resulting function. The algorithm due to Dueck & Miller [2] uses what is called the *isolation factor* (IF) for selecting minterms and selects implicant having minimum Relative Break Count (RBC).

In a joint publication, the author of this paper [5, 6] has achieved improvement in the performance of the DC techniques by aggregating and/or ordering the above mentioned criteria. A new way of synthesizing a given MVL function by injecting pseudo minterm(s) in the representation of a given function has also been proposed in [7, 8]. In these techniques the authors showed significant improvement compared to existing techniques in terms of the average number of product terms required to synthesize a given MVL function at the expense of an additional MVL MUX at the circuit's output.

III. PROPOSED APPROACH

A number of iterative heuristics algorithms have been used for synthesis of MVL functions [13-17]. Although these techniques showed some encouraging results, the execution times for synthesizing a given MVL function will be much longer compared to any DC techniques. In this paper, we try to get some benefit of the randomness nature of these iterative heuristics and the faster execution times of the deterministic techniques. This can be achieved by a calculated blend of these techniques.

The Ant Colony Optimization (ACO) algorithm [18] is a meta-heuristic that has a combination of distributed computation, autocatalysis (positive feedback) and constructive greediness to find an optimal solution for a number of combinatorial optimization problems. This algorithm tries to mimic the ant's behavior in the real world. We believe that the constructive nature of ACO algorithm is suitable with the technique that we are targeting.

The basic idea of our approach is to use the ant to decompose the given function to a number of levels and then synthesize a hopefully simpler circuit using the best DC techniques found in the literature [6]. Working from the circuit's output, the proposed algorithm proceeds as follows:

1. place a certain gate type at this level,
2. decompose using ACO, and
3. synthesize the (sub)-functions

Steps 2 and 3 can be performed repeatedly to create a multi-level structure. However, only 3-level synthesis is performed in this paper. In addition, we limit the application of the proposed algorithm to the case of 2-input *tsum* gate (see Definition 1) at the output of the circuit's last level. We opted to use the DC algorithm proposed in [6] since it represents the best baseline DC technique available in the literature.

From definition 4, we know that a minterm with value 0 can only be decomposed into two minterms each having 0 values. In this paper, this is written as $D(0) \rightarrow \{(0,0)\}$. However, there are different possible decompositions for minterm with value 2: $D(2) \rightarrow \{(0,2), (1,1), (2,0)\}$. For 4-valued functions, table that summarizes the different possible decomposition of values is shown in Table 1.

In the proposed algorithm an ant will travel through the truth table of the given function and select one combination out of the possible decompositions shown in Table 1 for each position in the truth table.

The selection process itself is a stochastic process influenced by the pheromone dropped by the previous ants. The probability of selecting a possible decomposition is calculated as $p = \tau_d / \Sigma\, \tau_d$, where $\tau_{d.}$ is the pheromone value of dth possible decomposition. After the ant finishes selecting a possible decomposition for all positions in the table, the best ant will update the pheromone on the selected combination. The amount of pheromone dropped is proportional to its fitness and calculated as follow: $\Delta\tau = PW \bullet F_f$, where PW is the pheromone weight and $F_f = (100\text{-}N_g)/100$ is the functional fitness. Note that in the last formulae, N_g is the number of gates used. Using such fitness function calculation; the representation that has least number of gates will have the highest F_f.

TABLE 1: Decomposition table

Val.	Possible decompositions
0	(0,0)
1	(0,1), (1,0)
2	(0,2), (1,1), (2,0)
3	(0,3), (1,2), (1,3), (2,1), (2,2), (2,3), (3,0), (3,1), (3,2), (3,3)

In addition to F_f, we introduce an additional criterion in selecting the best representation called *balance*, ***B***. Balance is calculated as the different in the number of gates between the two sub-functions generated by the ant's decomposition process. The lesser the different is the better the selection. We believe that having a balanced circuit is desirable. The proposed algorithm will try to find the circuits representation that uses the least number of gates. Out of those representations, the one that has the best balance will be selected. Thus, we can say that having the highest value of F_f is necessary but not sufficient to have a good representation while it is sufficient but not necessary to have a good B.

Example: Consider the example shown in Fig. 1, if we scan through the truth table row-wise and enumerate them, the possible path for ants to travel through the example shown in Fig. 1 is shown in Table 2.

Let us assume that an ant selects the following path: ((0,0), (2,0), (2,1), (0,1), (0,0), (2,0), (2,1), (1,1), (0,0), (0,0), (0,1), (2,1), (0,0), (0,0), (0,0), (0,0)). This is shown in Fig. 2. From this figure, it is easy to see that F1 can be synthesized using 3 literal gates while F2 requires only one literal gate (see Definition 2). This makes the total number of gates needed to realize the function in Fig. 1 to be 5 gates, including the tsum combining both F1 and F2.

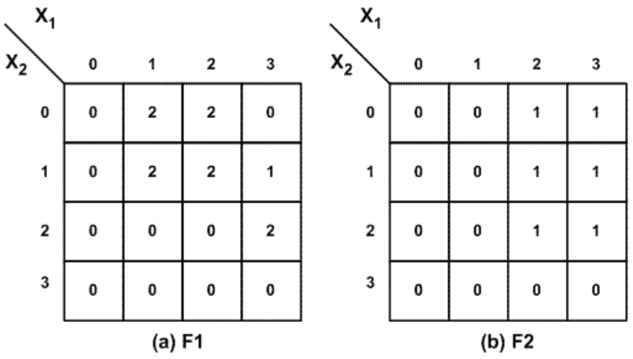
The F_f value of this representation is equal to (100-5)/100 = 0.95 while the balance is equal to 2. Suppose that any other ant managed to get a representation with higher F_f, then the representation of the later will be used. The algorithm will iterate until a certain stopping criteria such as the number of iterations is met.

IV. EXPERIMENTAL RESULTS

The proposed approach is tested against 50000 randomly generated 2-variables 4-valued functions. This set of benchmark functions is used evaluate the performance of the proposed algorithm as well as other existing techniques found in literature. Comparison is made based on the results obtained in terms of the average number of product terms needed to realize a given function.

TABLE 2: Possible Ants paths in Fig. 1

Minterm		Possible path
Pos.	Val.	
0	0	(0,0)
1	2	(0,2), (1,1), (2,0)
2	3	(0,3), (1,2), (1,3), (2,1), (2,2), (2,3), (3,0), (3,1), (3,2), (3,3)
3	1	(0,1), (1,0)
4	0	(0,0)
5	2	(0,2), (1,1), (2,0)
6	3	(0,3), (1,2), (1,3), (2,1), (2,2), (2,3), (3,0), (3,1), (3,2), (3,3)
7	2	(0,2), (1,1), (2,0)
8	0	(0,0)
9	0	(0,0)
10	1	(0,1), (1,0)
11	3	(0,3), (1,2), (1,3), (2,1), (2,2), (2,3), (3,0), (3,1), (3,2), (3,3)
12	0	(0,0)
13	0	(0,0)
14	0	(0,0)
15	0	(0,0)



$$\begin{array}{c|cccc} X_2 \backslash X_1 & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 2 & 2 & 0 \\ 1 & 0 & 2 & 2 & 1 \\ 2 & 0 & 0 & 0 & 2 \\ 3 & 0 & 0 & 0 & 0 \end{array} \qquad \begin{array}{c|cccc} X_2 \backslash X_1 & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 2 & 0 & 0 & 1 & 1 \\ 3 & 0 & 0 & 0 & 0 \end{array}$$

Fig. 2: Decomposed function of example 1

The ACO parameters used in the experiments are as follows:
(a) number of runs = 10,
(b) number of iterations = 200,
(c) number of ants = 30.

The MAX-MIN ant system is used to maintain the range of pheromone values in any possible path within certain limit. In addition to that, if there is any stagnancy occurring during the iteration, the pheromone initial value, which is equal to 1, will be applied to all possible paths. This will hopefully force the ants to try to find solution in new areas in the search space.

In our first few experiments, we tried to find out the best value for PW, pheromone evaporation rate (ρ) and pheromone range. Using the above parameters, we can see that the best performance (in terms of quality of solution and stability of exploration) of the algorithm can be achieved when $2 \leq PW \leq 3$ and ρ is equal to 0.05. Then, throughout our experiments, we use $PW = 2.5$.

Table 3 shows a comparison of the proposed technique with the techniques proposed in [7, 8]. We choose to compare the proposed technique with these techniques because they also add an additional gate at the circuit output which is the MVL MUX. In our proposed technique we add a TSUM gat at the output.

TABLE 3: Comparison with techniques proposed in [15]

Algorithm	# gates
Minterm Injection [7]	7.09408
2 Minterm Injection [7]	7.09276
MCPM (PI_SM method) [8]	7.09064
MCPM (PI_CM method) [8]	7.0705
The proposed approach	7.02906

From Table 3, we can see that the proposed technique outperforms the techniques reported in [7] and [8] in terms of the average number of gates used to realize a given 2-variable 4-valued function.

V. CONCLUDING REMARKS

A hybrid ACO and DC algorithm for multi-level synthesis of MVL functions is proposed in this paper. The main idea of the technique is to decompose a given MVL function into simpler (sub)-functions using ACO. The resulted (sub)-functions is then synthesized using DC technique. The proposed technique is tested against 50000 randomly generated 2-variable 4-valued functions and compared against existing DC techniques. The results show that the proposed technique outperforms other existing techniques in terms of the average number of product terms needed to realize a given MVL function.

ACKNOWLEDGMENT

The author would like to acknowledge the financial support received from Kuwait University through funded Research Project # WI 02/07.

REFERENCES

[1] P. W. Besslich, "Heuristic Minimization of MVL functions: A Direct Cover Approach", *IEEE Transactions on Computers, 35*(2), 1986, pp. 134-144.

[2] G. W. Dueck and D. M. Miller, "A Direct Cover MVL Minimization Using the Truncated Sum", in *Proceeding of the 17th international symposium on multi-valued logic*, 1987, pp. 221-227.

[3] G. Promper and J. A. Armstrong, "Representation of Multi-valued Functions Using Direct Cover Method", *IEEE Transactions on Computers, 30*(9), 1981, pp. 674-679.

[4] C. Yang and Y.-M. Wang, "A neighborhood decoupling algorithm for truncated sum minimization", in *Proceedings of the 20th International Symposium on Multiple Valued Logic,* 1990, pp. 153-160.

[5] M. Abd-El-Barr and B. Sarif," Weighted and Ordered Direct Cover Algorithms for Minimization of MVL Functions", in *Proceedings of the 37th International Symposium on Multiple Valued Logic*, 2007, pp. 48-53.

[6] B. Sarif and M. Abd-El-Barr, "Fuzzy-based Direct Cover Algorithm for synthesis of Multi-Valued Logic Functions", in *Proceedings of IASTED International Conference on Circuits and Systems*, 2008.

[7] B. Sarif and M. Abd-El-Barr, "Minterm Injection Technique for Synthesis of Multiple- Valued Logic Functions", in *Proceedings of IASTED International Conference on Circuits and Systems*, 2008.

[8] B. Sarif and M. Abd-El-Barr, "The Use of Multiple Connected Pseudo Minterms in The Synthesis of MVL Functions", in *Proceedings of the 39th International Symposium on Multiple Valued Logic*, 2009, pp. 145-150.

[9] P. Tirumalai and J. T. Butler, "On the Realization of Multiple-valued Logic Functions Using CCD PLA's", *Proceeding 14th International Symposium, Multiple-Valued Logic*, 1984, pp. 33-42.

[10] T. Sasao, "On the Optimal Design of Multiple-Valued PLAs", *IEEE Transactions on Computers, 38*(4), 1989, pp. 582-592.

[11] M. Abd-El-Barr and M. Hasan," New MVL-PLA Structures Based on Current-Mode CMOS Technology", in *Proceedings of the 26th International Symposium on Multiple Valued Logic,* 1996, pp. 98-103.

[12] F. Pelayo, A. Prieto, A. Lloris, and J. Ortega, "CMOS Current-Mode Multi-valued PLA's", *IEEE Transactions on Circuits and Systems, 38*(4), 1991, pp. 434-441.

[13] T. Kalganova, J. Miller and N. Lipnitskaya, "Multiple-Valued Combinational Circuits Synthesized Using Evolvable Hardware Approach", in *Proc. 7th-Workshop on Post-Binary Ultra Large Scale Integration Systems in Association with ISMVL'98, 1998,* pp.52-54.

[14] W. Wang and C. Moraga, "Evolutionary Methods in the Design of Quaternary Digital Circuits", *IEEE Proc. 28th-Int. Symposium On Multiple-Valued Logic, 1998,* pp.89-94.

[15] B. Sarif, and M. Abd-El-Barr, "Synthesis of MVL Functions - Part I: The Genetic Algorithm Approach", *International Conference on Microelectronics, 2006. ICM '06. 16-19 Dec. 2006,* pp. 154 – 157.

[16] M. Abd-El-Barr and B. Sarif, "Synthesis of MVL Functions - Part II: The Ant Colony Optimization Approach*", International Conference on Microelectronics, 2006. ICM '06. 16-19 Dec. 2006,* pp. 158 – 161.

[17] B. Sarif, and M. Abd-El-Barr, "Synthesis of MVL Functions Using Discrete Particle Swarm Optimization"*, in the 2008 IEEE Swarm Intelligence Symposium, St. Louis, USA, September 21-23, 2008.*

[18] M. Dorigo and G. Di Caro, "New Ideas in Optimization", McGraw Hill, London, UK, 1999.