Iterative Synthesis Techniques for Multiple-Valued Logic Functions

Mostafa Abd-El-Barr

***Abstract*—A major challenge facing Multiple-Valued Logic (MVL) synthesis techniques is the enormous size of the functional search space. Accordingly, deterministic synthesis techniques of MVL functions are prohibitively expensive and complex in terms of algorithmic running time. Therefore, a number of heuristics for near optimal functional synthesis of MVL have been reported in the literature. Among the well-known heuristics is the Direct Cover algorithm (DCA). We have introduced a number of improved versions of the DCA. These include the Weighted Direct Cover (WDC), the Ordered Direct Cover (ODC), and the Fuzzy Direct Cover (FDC). In this paper, we review and compare the performance of those heuristic iterative techniques using two set of benchmarks. The first consists of 50000 randomly generated 2-varaible 4-valued functions and the second consists of 50000 2-variable 5-valued functions. The average number of product terms required to synthesize a given MVL function is used as the criterion for comparison. The results obtained show that the modified iterative synthesis heuristics outperformed the DCA and that among the modified techniques the FDC produces the best results.**

***Index Terms*—MVL functional synthesis, Direct-Cover algorithms (DCA), Weighted DC (WDC), Ordered DC (ODC), Fuzzified DC (FDC), non-binary digital signal processing (DSP).**

I. Introduction

Digital information processing using Multiple-Valued Logic (MVL) is carried out using more than discrete logic levels. Due to technological reasons and for easy interfacing with the predominantly existing binary digital systems, 4-valued logic has been the most widely used. Recent advances in Very Large Scale Integration (VLSI) technology made it possible to fabricate more efficient 4-valued circuits using binary CMOS (Complementary Metal Oxide Semiconductor) technology. These circuits have shown considerable reduction both in processing time and chip area compared to their binary counterparts [1]-[8].

The MVL functional synthesis problem is however more complex when compared to its binary counterpart. This is because of the massive size of the MVL functional search space. Consider, for example, the case of two-variable functions. While there are only 16 2-variable binary functions there are $4^{16} = 4294967296$ 2-variable 4-valued functions.

Exact minimization of MVL functions is prohibitively expensive [9]. A number of iterative heuristics for near minimal synthesis of MVL functions have been introduced see, for example, [9]-[16]. Fuzzy-based synthesis of MVL functions has also been reported in the literature [17].

An *n*-variable *r*-valued function *f*(*X*) is a mapping $f : R^n \rightarrow R$, where $R = \{0,1,\cdots,r-1\}$ is a set of r-logic values with $r \geq 2$ and $X = \{x_1, x_2, \cdots, x_n\}$ is a set of *r* -valued *n* variables. Table I shows a number of possible MVL logic operators. Fig. 1 shows an example of a 2-varaible 4-valued function. In this figure $1\bullet^0X_1{}^0\bullet^3X_2{}^3$, $2\bullet^0X_1{}^0\bullet^1X_2{}^1$ and $3\bullet^0X_1{}^0\bullet^2X_2{}^2$ are examples for minterms while $2\bullet^0X_1{}^1\bullet^1X_2{}^1$and $2\bullet^0X_1{}^1\bullet^1X_2{}^2$ are examples for *implicants*.

TABLE I: Sample MVL Operators

The operator	The logic equation
The window literal	${}^a x^b = \begin{cases} (r-1) & if\ (a \leq x \leq b) \\ 0 & otherwise \end{cases}$ where $a, b \in R$ and $a \leq b$
The tsum (truncated sum, TS)	$tsum(a_1, a_2, ..., a_n) = a_1 \oplus a_2 \oplus ... \oplus a_n =$ $\begin{cases} a_1 + a_2 + ... + a_n & if \quad a_1 + a_2 + ... + a_n < r-1 \\ r-1 & otherwise \end{cases}$ $a_i \in R$ and $\oplus$ represents the TS.
The maximum (MAX) of two MVL variables	$MAX(x_1, x_2) = \begin{cases} x_1 & \text{if} \quad x_1 \geq x_2 \\ x_2 & \text{otherwise} \end{cases}$
The minimum (MIN) of two MVL variables	$MIN(x_1, x_2) = \begin{cases} x_1 & \text{if} \quad x_1 \leq x_2 \\ x_2 & \text{otherwise} \end{cases}$

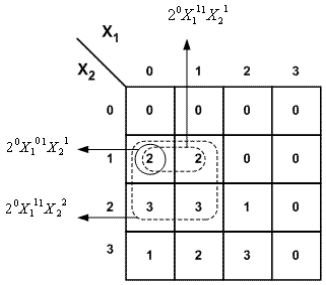


Fig. 1. A tabular representation of *f* (*X*1, *X*2).

II. The Direct Cover Algorithm for Synthesis of MVL Functions

The Direct Cover Algorithm (DCA) for synthesis of MVL functions [10]-[13] consist of the following main steps:

1) Choose a minterm (see Definition 3),
2) Identify a suitable implicant (see Definition 4) that covers the chosen minterm,
3) Obtain a reduced function by removing the identified implicant, and

Manuscript received September 2, 2013; revised December 6, 2013. This work was supported by Kuwait University under Grant #WI 04/10.

Mostafa Abd-El-Barr is with the Department of Information Science, CCSE, Kuwait University, Safat 13060, Kuwait (e-mail: mostafa.abdelbarr@gmail.com).

4) Repeat steps 1 to 3 until no more minterms remain uncovered.

The selection of appropriate minterms and the implicants covering them play an important role in obtaining less number of product terms to cover a given function. The DCAs reported in the literature differ in the way appropriate minterms are chosen. They also differ in the way appropriate implicants are identified. Different metrics to select minterms have been proposed in the literature. We have presented a comprehensive analysis of the DCAs in [14]. The outcome of this analysis has indicated that improvements to the DCA are still possible. We have introduced three iterative-based heuristics as improvements to the DCA. These are explained below.

III. Weighted Direct Cover (WDC)

It is observed in [14] that all criteria used for selection of minterms and/or implicants have linear and monotonic function. All criteria, except for RBC and NRC, assume values greater than or equal to 0. We use the term *weight pattern* to specify the weight for each selection criterion for both minterm and implicant and that the weight should be in the set {0, 1, 2, 3}. A minterm weight pattern 112 means that w_{CF} =1, w_{CFN} =1, and w_{IW} =2. Combining the weight patterns for minterms with the weight patterns for implicants, we will have 4096 different patterns (P-1, P-2, …). Representatives of different weight scenarios are examined (see Table II). These weight patterns will be used for both minterm and implicant. In total, the number of different algorithms (because of different patterns) tested in this paper is $(24)^2$ = 576.

TABLE II: Different Weighting Scenario

Time	WM-1	WM-2	WM-3		WM-1	WM-2	WM-3
P1	0	0	1	P13	1	2	2
P2	0	1	0	P14	1	2	3
P3	0	1	1	P15	2	0	1
P4	0	1	2	P16	2	1	0
P5	0	2	1	P17	2	1	1
P6	1	0	0	P18	2	1	2
P7	1	0	1	P19	2	1	3
P8	1	0	2	P20	2	2	1
P9	1	1	0	P21	2	3	1
P10	1	1	1	P22	3	1	2
P11	1	1	2	P23	3	2	1
P12	1	2	1	P24	3	3	1

All of the 576 patterns will be tested using our benchmark. Since presenting all results from the 576 different algorithms is not practical, we only present the best 10 results (in terms of average PTs) for both 2-variable 4-valued and 2-variable 5-valued functions among the benchmarks used. Table III and Table IV show these results, respectively. The results presented in these tables show that the best result is obtained using the following pattern: for minterm, *CF* = 0, *CFN* = 1 and *IW* = 2 and for implicant, *RBC* = 1, *NRC* = 0 and *LRZ* = 0. Using this pattern (written shortly as 012-100), the average number of product terms (PTs) required to cover a given 2-variable 4-valed function is 7.24914 while it is 12.1658 in the case of a 2-variable 5-valed function.

The performance of the WDC algorithm is assessed through the results obtained using a benchmark consisting of 50000 randomly generated 2-variable 4-valued functions and a benchmark consisting of 50000 randomly generated 2-varaiable 5-valued functions. This assessment is presented in Section VI.

TABLE III: The Best 10 Weight Patterns for 2-Variable 4-Valued Functions

	Minterm			Implicant			# PTs
Weight Pattern							
	CF	CFN	IW	RBC	NRC	LRZ	
P1	0	1	2	1	0	0	7.2491
P2	1	2	3	1	0	0	7.2502
P3	0	0	1	2	1	3	7.2623
P4	0	1	1	1	0	0	7.2625
P5	1	2	2	1	0	0	7.2629
P6	0	0	1	2	1	2	7.2636
P7	0	0	1	3	1	2	7.2636
P8	1	1	2	1	0	0	7.2636
P9	0	0	1	2	1	1	7.2650
P10	1	2	3	2	1	0	7.2707

TABLE IV: The Best 10 Weight Patterns for 2-Variable 5-Valued Functions

	Minterm			Implicant			# PTs
Weight Pattern							
	CF	CFN	IW	RBC	NRC	LRZ	
P1	0	1	2	1	0	0	12.166
P2	1	2	3	1	0	0	12.175
P3	1	1	2	1	0	0	12.189
P4	1	2	2	1	0	0	12.196
P5	0	1	1	1	0	0	12.197
P6	1	1	1	1	0	0	12.208
P7	0	2	1	1	0	0	12.213
P8	1	2	1	1	0	0	12.222
P9	2	2	1	1	0	0	12.224
P10	2	3	1	1	0	0	12.229

IV. Ordered Direct Cover (ODC)

According to this approach selection criteria are ordered based on a given priority [15]. There are three criteria for minterm selection. These are Smallest CF, Smallest CFN, and Smallest IW. There are three criteria for implicant selection. These are Smallest RBC, Smallest NRC, and Largest LRZ. Assume that for minterm selection we set Criterion 1 as ‘Samllest CF”, Criterion 2 as “Smallest CFN” and Criterion 3 as “Smallest IW” and for implicant selection we set Criterion 1 as “Samllest RBC”, Criterion 2 as “Smallest NRC, and Criterion 3 as “Largest LRZ” (See Table V). In Table VI, we summarize the different scenarios for ordering (O-1, O-2, …) the above mentioned criteria.

TABLE V: Criteria Used for ODC Algorithm

	Criterion 1	Criterion 2	Criterion 3
Minterm	Smallest CF	Smallest CFN	Smallest IW
Implicant	Smallest RBC	Smallest NRC	Largest LRZ

We assess the performance of the ODC through the results obtained using the same benchmark used in the case of the WDC.

TABLE VI: DIFFERENT ORDER SCENARIO

	Order-1	Order-2	Order-3		Order-1	Order-2	Order-3
P1	CR1	~	~	P9	CR2	CR1	CR3
P2	CR1	CR2	~	P10	CR2	CR3	CR1
P3	CR1	CR3	~	P11	CR3	~	~
P4	CR1	CR2	CR3	P12	CR3	CR1	~
P5	CR1	CR3	CR2	P13	CR3	CR2	~
P6	CR2	~	~	P14	CR3	CR1	CR2
P7	CR2	CR1	~	P15	CR3	CR2	CR1
P8	CR2	CR3	~				

Table VII and Table VIII show ten orderings that give the best results for 2-variable 4-valued and 2-variable 5-valued functions, respectively.

TABLE VII: THE BEST 10 RESULTS OF ORDERED DC ALGORITHM FOR 2-VARIABLE 4-VALUED FUNCTIONS

	Patter Order (Minterm; Implicant)	#PTs
1	(IW,x,x; RBC, LRZ, x)	7.20234
2	(IW,x,x; RBC, LRZ, NRC)	7.20234
3	(IW, CF,x; RBC, LRZ, x)	7.20248
4	(IW, CF, x; RBC, LRZ, NRC)	7.20248
5	(IW, CFN, x; RBC, LRZ, x)	7.20248
6	(IW, CFN, x; RBC, LRZ, NRC)	7.20248
7	(IW, CF, CFN; RBC, LRZ, x)	7.20248
8	IW, CF, CFN; RBC, LRZ, NRC)	7.20248
9	IW, CFN,CF; RBC, LRZ, x)	7.20248
10	(IW, CFN, CF; RBC, LRZ, NRC)	7.20248

TABLE VIII: THE BEST 10 RESULTS OF ORDERED DC ALGORITHM FOR 2-VARIABLE 5-VALUED FUNCTIONS

	Patter Order (Minterm; Implicant)	#PTs
1	(IW,CF,x; RBC, LRZ, x)	12.0682
2	(IW,CF,x; RBC, LRZ, NRC)	12.0682
3	(IW, CFN,x; RBC, LRZ, x)	12.0682
4	(IW, CFN,x; RBC, LRZ, NRC)	12.0682
5	(IW, CF, CFN; RBC, LRZ, x)	12.0682
6	(IW, CF,CFN; RBC, LRZ, NRC)	12.0682
7	(IW, CFN, CF; RBC, LRZ, x)	12.0682
8	IW, CFN, CF; RBC, LRZ, NRC)	12.0682
9	IW, x,x; RBC, LRZ, x)	12.0685
10	(IW, x, x; RBC, LRZ, NRC)	12.0685

The performance of the ODC algorithm is assessed through the results obtained using a benchmark consisting of 50000 randomly generated 2-variable 4-valued functions and a benchmark consisting of 50000 randomly generated 2-varaiable 5-valued functions. This assessment is presented in Section VI.

V. FUZZY-BASED DIRECT COVER (FDC)

The Fuzzy Direct Cover (FDC) algorithm employs fuzzy rules to select the best minterm and the best implicant covering it [16]. A fuzzy logic rule expresses the way in which linguistic variables (objectives) are interrelated, the relationship between these objectives, and the overall function value [17]. The goal is to find a high quality solution, represented by a linguistic variable. See the next two illustrative examples.

1) **IF** a minterm has a good CF **OR** good CFN **OR** good IW, **THEN** it is a good minterm for selection.
2) **IF** an implicant has a good RBC **OR** a good LRZ **OR** a good NRC, **THEN** it is a good implicant for selection.

Our proposed Fuzzy-based Direct Cover (FZDC) algorithm employs fuzzy rules (along with preferences) to select the best set of minterm and the most appropriate implicant covering each such that the whole function is covered. The goodness of a minterm (implicant) is examined using the abovementioned fuzzy rules and preferences. Looking at these rules, it is easy to deduce that we can use the 'OR-'like operator to aggregate all decision criteria. Table IX shows the mathematical formulae we introduced for each membership function in the minterm selection.

TABLE IX: MEMBERSHIP FUNCTIONS IN MIN TERMS SELECTION

Minterm Selection		
Technique	Criterion	Formulated Membership Function
DM [10]	CF	$\mu_{CF} = \begin{cases} 1 & if\ CF = 0 \\ \frac{Max_{CF} - CF}{Max_{CF}} & if\ 0 < CF < Max_{CF} \\ 0 & if\ otherwise \end{cases}$
ND [8]	CFN	$\mu_{CFN} = \begin{cases} 1 & if\ CFN = 0 \\ \frac{Max_{CFN} - CFN}{Max_{CFN}} & if\ 0 < CFN < Max_{CFN} \\ 0 & otherwise \end{cases}$
BS [3]	IW	$\mu_{IW} = \begin{cases} 1 & if\ IW = Min_{IW} \\ \frac{Max_{IW} - IW}{Max_{IW} - Min_{IW}} & if\ Min_{IW} < IW < Max_{IW} \\ 0 & otherwise \end{cases}$

TABLE X: MEMBERSHIP FUNCTIONS IN IMPLICANT SELECTION

Implicant Selection		
Technique	Criterion	Formulated Membership Function
DM [10]	RBC	$\mu_{RBC} = \begin{cases} 1 & if\ RBC = Min_{RBC} \\ \frac{MAX_{RBC} - RBC}{Max_{RBC} - Min_{RBC}} & if\ MIN_{RBC} < RBC < Max_{RBC} \\ 0 & otherwise \end{cases}$
ND [8]	NRC	$\mu_{NRC} = \begin{cases} 1 & if\ NRC = Min_{NRC} \\ \frac{MAX_{NRC} - NRC}{Max_{NRC} - Min_{NRC}} & if\ MIN_{NRC} < NRC < Max_{NRC} \\ 0 & otherwise \end{cases}$
BS [3]	LRZ	$\mu_{LRZ} = \begin{cases} 1 & if\ LRZ = Max_{LRZ} \\ \frac{LRZ}{Max_{LRZ}} & if\ 0 < LRZ < Max_{LRZ} \\ 0 & otherwise \end{cases}$

Table X shows the same for implicant selection.

Efficiency of the proposed fuzzy selection process is influenced by the parameter used in fuzzy operators, i.e. the value of β in OWA operator. In addition to that, fuzzy preference rules will also impact the performance of the proposed algorithm. Since there are three criteria for each of the minterm and implicant selection, there will be additional 6 parameters to fine-tune in order to get the best performance of the algorithm. In order to obtain the best result using the proposed fuzzy-based selection criteria, the following set of experiments are conducted:

1. Experiments with different fuzzy operators
2. Experiments with parameters in fuzzy operators

Table XI shows the fuzzy preference used for the first experiment.

TABLE XI: FUZZY PREFERENCE FOR MINTERM AND IMPLICANT

Minterm		Implicant	
Criteria	Fuzzy Preference	Criteria	Fuzzy Preference
IW	0.9	RBC	0.9
CF	0.2	LRZ	0.2
CFN	0.1	NRC	0.1

Using the above mentioned fuzzy preferences, we tested the proposed fuzzy selection criteria against the 50000 randomly generated 2-variables 4-valued MVL functions. Five different fuzzy operators are used for this purpose. Table XII shows the results of the experiment. It should be noted that we list the results obtained in two cases: not considering minterm values in any order (No CMV) and taking minterm values in ascending orders; lower to higher values (With CMV). We set β = 0.5 while collecting the results reported in Table XII.

TABLE XII: PERFORMANCE OF DIFFERENT FUZZY OPERATOR IN FDC

	# PT No CMV	# PT With CMV
Max	8.5513	8.0301
Max with pref.	7.30344	7.21964
OWA	7.38226	7.28898
OWA with pref.	7.27186	7.19450
Weighted Average	7.30646	7.19784

VI. COMPARISON

In this section we provide a comparison among the iterative heuristics presented above. The results shown in Table XIII reveal that the three heuristic algorithms outperform other existing DC-based techniques, regardless of the number of minterms in the given MVL function. Among the three introduced algorithms, it is clear that in the worst case, the FDC produces results that are as good as those produced by the other two algorithms. However, in vast majority of the cases, the FDC produces results that are better than those produced by the other two algorithms in terms of the average number of product terms needed to synthesize a given function.

TABLE XIII: AVERAGE #PT WITH RESPECT TO DIFFERENT NUMBER OF MINTERMS OF MVL FUNCTIONS USED

# minterm	# functions	Promper	Besslich	Dueck	ODC	WDC	FDC
16	500	7.594	7.562	7.002	7.086	7.01	6.946
15	2679	8.295	8.307	7.51	7.481	7.501	7.423
14	6589	8.355	8.405	7.569	7.516	7.559	7.500
13	10585	8.275	8.352	7.541	7.491	7.545	7.484
12	11230	8.049	8.098	7.382	7.33	7.383	7.320
11	9003	7.707	7.757	7.129	7.086	7.134	7.087
10	5434	7.323	7.366	6.831	6.787	6.837	6.794
9	2575	6.871	6.879	6.473	6.436	6.479	6.444
8	1038	6.309	6.322	6.023	5.978	6.022	5.981
7	277	5.726	5.751	5.527	5.484	5.523	5.505
6	75	5.133	5.147	4.973	4.96	4.987	4.960
5	13	4	4	4	3.923	4	4
4	1	4	4	4	4	4	4
3	1	3	3	3	3	3	3
Total	50000						

The following Tables XIV to XVI provides tabular forms of the obtained percentage improvements achieved using the three iterative heuristics as compared to the results obtained using the conventional DC heuristics.

TABLE XIV: THE PERCENTAGE OF IMPROVEMENT ACHIEVED BY FDC

Type of functions	Number of functions	% functions
$\geq$ 12% Functions improvement	6589	$\cong$ 13.2%
10% $\leq$ improvement <12%	(2679+10585+11230) = 24494	$\cong$ 49%
< 10% improvement $\leq$ 5%	(500+9003+5434+2575+1038) = 18550	$\cong$ 37.1%
< 5% improvement	(277+75)= 352	$\cong$ 0.7%
No improvement	15	$\cong$ 0.03%

TABLE XV: THE PERCENTAGE OF IMPROVEMENT ACHIEVED BY WDC

Type of functions	Number of functions	% Functions
$\geq$ 10% Functions improvement	(2679+6589+10585)=19853	$\cong$ 39.7%
5% $\leq$ improvement <10%	(500+11230+9003+5434+2575) = 28742	$\cong$ 57.48%
< %5	(1038+277+75) = 1390	$\cong$ 2.78%
No improvement	15	$\cong$ 0.03%

TABLE XVI: THE PERCENTAGE OF IMPROVEMENT ACHIEVED BY ODC

Type of functions	Number of functions	% Functions
$\geq$ 10% improvement	(2679+6589+10585+11230) = 31083	$\cong$ 62.1%
5% $\leq$ improvement <10%	(500+9003+5434+2575+1038) = 18550	$\cong$ 37.1%
< %5 improvement	(277+75+13) = 365	$\cong$ 0.73%
No improvement	2	$\cong$ 0.004%

From Table XIV, we can see that the maximum percentage of improvement achieved using the FDC heuristic over all DC-based techniques is 12.067% and this is achieved in 6589 functions out of the 50000 (about 13.2% of the benchmark functions). From Table XV, we can see that the maximum percentage of improvement achieved using the WDC heuristic over all DC-based techniques is 11.192%. This has been achieved in 6589 functions (about 13.2% of the 50000 benchmark functions). From Table XVI, we can see that the maximum percentage of improvement achieved using the ODC heuristic over all DC-based techniques is 11.828%. This has been achieved in 6589 functions (about 13.2% of the 50000 benchmark functions). A summary of the results obtained using the three heuristic algorithms is provided in Table XVII in the form of an overall comparison among the heuristic algorithms and the existing DC-based algorithms in terms of the average number of PTs used using the randomly generated 50000 2-variable 4-valued and the 50000 2-variable 5-valued benchmarks.

TABLE XVII: COMPARISON AMONG THREE DIFFERENT HEURISTIC ALGORITHMS

	#PT (2-variable 5-valued)	#PT (2-variable 4-valued)
Promper	13.4404	7.89012
Besslich	13.6507	7.93882
Dueck	12.1525	7.24786
WDC	12.1658	7.24914
ODC	12.0682	7.20234
FDC	12.0694	7.19422

VII. CONCLUDING REMARKS

In this paper, we have presented three iterative heuristics

for synthesis of MVL functions: the WDC, ODC, and FDC. We have also compared the results obtained using these algorithms through simulating the three algorithms using two benchmarks: 50000 2-variable 4-valued and 2-variable 5-valued functions. The results obtained showed that the three heuristics outperform the conventional DCs. Among the three heuristics the FDC achieved the best improvements.

ACKNOWLEDGMENT

The author would like to acknowledge the support of Kuwait University provided in the form of the funded research project WI 04/10.

REFERENCES

[1] E. Dubrova, "Multiple-valued logic in VLSI," *International Journal on Multiple-Valued Logic and Soft Computing*, pp. 1-17, 2002.
[2] K. Naiff, D. Rich, and K. Smalley, "A four-state ROM using multi-level process technology," *IEEE Journal of Solid-State Circuits*, vol. 19, no. 2, pp. 174-179, April 1984.
[3] H. Razavi and S. Bou-Ghazale, "Design of a Fast CMOS ternary adder," in *Proc. IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, May 1987, pp. 20-23.
[4] T. Hanyu and M. Kameyama, "A 200 MHz pipelined multiplier using 1.5 V-supply multiple-valued MOS current-mode circuits with dual-rail source-coupled logic," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 11, pp. 1239-1245, November 1995.
[5] V. Patel and K. Gurmurthy, "Arithmetic operations in multi-valued logic," *International Journal of VLSI & Communication Systems (VLSICS)*, vol. 1, no. 1, pp. 21-32, March 2010.
[6] T. Manikas and D. Teeters, "Multiple-valued logic memory system design using Nano-scale electro-chemical cells," in *Proc. ISMVL-2008*, May 2008, pp. 197-201.
[7] M. Hosseinzadeh, S. Jassbi, and K. Navi, "A novel multiple-valued logic OHRNS modulo adder circuit," in *Proc. World Academy of Science, Engineering, and Technology*, vol. 25, ISSN 1307-6884, pp. 128-133, November 2007.
[8] I. Kouretus and V. Puliourus, "High-radix redundant circuits for RNS modulo r^n—1, r^n, OR r^n+1," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS-2003)*, vol. 5, pp. V-229-V-232, 2003.
[9] E. Dubrova, Y. Jiang, and R. Brayton, "Minimization of multi-valued functions in post algebra," in *Proc. International Workshop on Logic and Synthesis*, 2001, pp. 132-137.
[10] G. Promper and A. Armstrong, "Representation of multiple valued functions using the direct cover method," *IEEE Transactions on Computers (TC)*, pp. 674-679, September 1981.
[11] W. Besslich, "Heuristic minimization of MVL functions: A direct cover approach," *IEEE Transactions on Computers (TC)*, vol. C-35, no. 2, pp. 134-144, February 1986.
[12] G. Dueck and M. Miller, "A direct cover MVL minimization using the truncated sum," in *Proc. ISMVL-87*, May 1987, pp. 221-227.
[13] C. Yang and Y. Wang, "A neighbourhood decoupling algorithm for truncated sum minimization," in *Proc. IEEE Twentieth International Symposium on Multiple-Valued Logic*, 1990, pp. 153-160.
[14] M. Abd-El-Barr and L. Al-Awami, "Analysis of direct cover algorithms for minimization of MVL functions," in *Proc. 15th IEEE International Conference on Microelectronics*, 2003, pp. 308-312.
[15] M. Abd-El-Barr and B. Sarif, "Weighted and ordered direct cover algorithms for minimization of MVL functions," in *Proc. 37th IEEE International Symposium on Multiple-Valued Logic*, 2007, pp. 48-53.
[16] B. Sarif and M. Abd-El-Barr, "Fuzzy-based direct cover algorithm for synthesis of multiple-valued logic functions," in *Proc. IASTED Circuits and Systems*, Hawaii, 2008, pp. 625-630.
[17] L. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 8, no. 3, pp. 338-353, 1965.



Mostafa Abd-El-Barr received his PhD degree in Computer Engineering, University of Toronto, Canada in 1986. He is with the Department of Information Science, College of Computing Sciences and Engineering (CCSE), Kuwait University since 2003 and an Adjunct Professor with the ECE Department, University of Victoria (UVic), BC, Canada. His research interests include Design and Analysis of Reliable & Fault-Tolerant Computer Systems, Computer Networks Optimization, Parallel Processing/Algorithms, Information Security, Multiple-Valued Logic (MVL) Design & Analysis, VLSI System Design, and Digital Systems Testing. He is the author and/or co-author of more than 160 scientific papers published in journals and conference proceedings/symposia. He has three books published (two are also translated to the Chinese Language). He is also an official IEEE/EAC/ABET. Professor Abd-El-Barr is a Senior IEEE Member and a member of the International Association of Engineers (IAENG). He is a Senior International Associate Editor of the International Journal of Information Technology & Web Engineering and a member of the Editorial of the International Journal of Computer Science and Security (IJCSS). Dr. Abd-El-Barr is a Certified Professional Engineer in Ontario, Canada.