MVL-MIN: A New Heuristic Minimization Tool For Multiple-valued Logic Functions

Ibrahim Savran

***Abstract*—**In this paper, a new heuristic method to minimize multiple-valued logic (MVL) functions and its implementation results will be introduced. The motivation for this work has been to develop a light weight MVL minimization algorithm which is simpler than existing algorithms so that it can serve as a basis for the future work. MVL-MIN is designed and implemented from scratch and it is compared with MVSIS. MVSIS is a MVL minimization program developed by the Electronic Systems Design group at Berkeley. The advantages of the new algorithm include a new cube calculus based technique for detecting and eliminating the totally redundant prime implicants. A comparison with MVSIS approved that the proposed algorithm is able to solve all test files within a fixed allocation of computer resources. Since conventional testbenches are fairly small, we generated 3 sets of testbench in blif-mv format. Each set includes 8 multiple-valued logic functions. MVL-MIN is able to solve all test-benches about 5 times faster than MVSIS on average.

***Keywords*—**Multiple-valued logic, multiple-valued logic minimization, cube calculus, heuristic minimization.

I. Introduction

THE performance of binary logic is limited due to high number of interconnections, which occupy a large area on an integrated circuit. More than half of the chip area is devoted to wires [1, 2]. As a result, researchers are looking for a way to reduce it. One can achieve a more cost-effective way of utilizing interconnections by using a larger set of signals over the same area in MVL. Using fewer wires to transmit multi discrete values allows denser information content per interconnection and thus results in a circuit with fewer conductors than the binary-valued counterpart [3, 4]. For example, in the case of 4-valued logic, half of the wire space is saved [5].

In modern VLSI technology, the chip size and performance are closely related to the number of wires, pins, etc. In principle, MVL can provide high data processing capability per unit chip area, and decrease the connection between circuit units.

Since 1970s, the Complementary Metal-Oxide Semi-conductor (CMOS) has been the main technology for implementing dense and energy-efficient VLSI circuits. However, the general trend of reducing the size of CMOS (Complementary Metal-Oxide-Semiconductor) technology in nano-scale has confronted many difficulties. The main obstacles include very high leakage currents, high power density, parasitic effects, and restriction of routing and placement processes [6, 7, 8]. Thus, many nano-scale technologies such as Quantum-dot Cellular Automata (QCA) [9], Single Electron Transistor (SET) [10], and CNTFET [11] have been introduced to overcome these challenges. Among them, CNTFET is the most promising candidate to be a successor to the CMOS technology in the near future [12]. Several arithmetic MVL circuits have been proposed [13, 14]. In 2006, IBM demonstrated the first IC built using SWCNTs [15]. Then, a decoder, a sensor interface circuit, stand-alone circuit elements such as half-adder sum generators, D-latches and memory (SRAM) cells have been fabricated [16, 17, 18, 19, 20, 21, 22]. In 2008, Cao et al. [23] announced that they made medium size IC using CNTFETs. Recently, Shulaker et al. [24] fabricated first CNT computer entirely using CNTFETs.

The minimization of logic expressions is an important step in modern circuit design. Unfortunately, since the MVL functions have more literals to deal with, the designing MVL devices is more complicated than those of their binary counterparts [25].

In logic design, the most common optimization metric is the number of terms in the logic expression, which is easily calculated during the minimization phase. The correlation between the number of terms and chip area is very high in the binary case, i.e. it is a good estimate.

Due to the exponential nature of finding the optimal cover, the state-of-the-art optimization algorithms can only handle functions that have a limited number of terms. Therefore, most of the tools rely on heuristic minimization methods such as MVSIS [26]. In the literature, there are several methodologies reported for the synthesis of MVL functions, such as direct-cover-based approaches [27, 28], network learning via local search methods [29, 30], genetic algorithms [31, 32, 33], and artificial intelligence methods [34].

The organization of the paper is as follows: Section 2 covers the background of the study and the terminology. Then, section 3 briefly explains the “Cube Calculus Operators” which are used to find the prime cover. Section 4 describes our MVL heuristic minimization algorithm MVL-MIN, and presents an example. Finally, Section 5 concludes the paper and gives future applications.

II. Background

To be consistent with the literature we want to give definitions and algebraic procedures in this section. For

Ibrahim Savran is with the Karadeniz Technical University, Trabzon, 61080, Turkey

https://doi.org/10.17758/DIRPUB1.E0318012

simplicity and formality of the explanations in this study the following notation is used.

A. Definitions

- Let S_{on}, S_{off}, S_{dc}, S_{now}, S_{pc} be the set of on-cubes, the set of off-cubes, the set of "don't-care" cubes, uncovered part of S_{on} set, and the prime cover set respectively [35].
- Let X_1, X_2, ..., X_n be multiple-valued variables such that each variable X_i can take values from a certain finite discrete set V_i (V_i ={0,1,...,r_{i-1}}). A literal $X_i^{S_i}$ of variable X_i represents a characteristic function of subset S_i of V_i, that is, the literals value is '1' for symbols from this subset. A multiple-valued dont'care is depicted as $X_i^{V_i}$ or X_i^{*}.
- Let X, be an n-dimensional cube which X = $X_{n-1}X_{n-2}...X_0$ where X_i's are multiple-valued variables.
- Prime cube: cube X is prime if there is no other cube Z such that $Z \supset X$.
- Let f be an n-input 1-output *multiple-valued function*. We can show the MV function as a mapping: $f(X_0, X_1,..., X_{n-1})$: $\times_{i=0}^{n-1} V_i \rightarrow \{0, 1, *\}$ where V_i is the radix of i_{th} variable. When all radixes are same ($V = V_i$; *for all i*),
multiple-valued mapping will be simplified, $f: \; V_i \rightarrow \{0, 1, *\}$.
- Cover: a set of cubes which together cover every element of the set S_{on}.
- Prime Cover: a cover in which all cubes are prime.

The goal of multiple-valued logic minimization is to find an optimal prime cover for a given function

III. CUBE CALCULUS

In this section, we are going to explain the cube operations which are needed for the minimization algorithm [36]. The cube calculus operators in the study of Marek et. al which use positional notation. Thus handling these operators are more complex than what we introduce here.

A. Expansion Operator -$\hat{E}$

This operator is the first operator to find local primes of a given on-set cube. The expansion operator requires two operands: The first parameter is a cube from the set S_{now}, A 2 S_{now}, which is used as the *expander* for the second operand which comes from S_{off} set.

Let $A = A_{n-1}^{\alpha_{n-1}} A_{n-2}^{\alpha_{n-2}} ... A_0^{\alpha_0}$ be an n-coordinate expander cube, and let $B = B_{n-1}^{\beta_{n-1}} B_{n-2}^{\beta_{n-2}} ... B_0^{\beta_0}$ be the expandee cube where $A \in S_{now}$; $B \in S_{off}$ respectively. The expansion operator produces a new cube $C = C_{n-1}^{\gamma_{n-1}} C_{n-2}^{\gamma_{n-2}} ... C_0^{\gamma_0}$, where $C = \hat{E}(A, B)$ as follows:

$$\hat{E}(A,B) = \begin{cases} \text{if } \alpha_i = * \vee \alpha_i = \beta_i \text{ then } \gamma_i = * \\ \text{else if } \alpha_i \neq \beta_i \text{ then } \gamma_i = s \end{cases} \quad (1)$$

The special character s is for coordinates which we need to use in the 'non-disjoint sharp operator.' The byproduct set holds the cubes that are produced by the expansion operators. Here we can express this set as $S_B = \{C | C = \hat{E}(A, B), A \in S_{now}, B \in S_{off}\}$.

B. Elimination Operator –$\hat{X}$

This operator processes the byproduct set S_B which is produced by the expansion operator. Elimination is used to remove the non-maximal cubes from the byproduct set, because these are not necessary for finding primes.

Let A and B be two cubes where A, B $\in$ S_B. This operator works based on the following rules:

$$\hat{X}(A,B) = \begin{cases} \text{if } \alpha_i = * \vee \alpha_i = \beta_i, \\ \quad \forall i = 0, 1, \ldots, n-1 \\ \quad\quad \text{then the cube } B \text{ is eliminated} \\ \text{else if } \beta_i = * \vee \alpha_i = \beta_i, \\ \quad \forall i = 0, 1 \ldots, n-1 \\ \quad\quad \text{then the cube } A \text{ is removed} \\ \text{else both of the cubes remain.} \end{cases} \quad (2)$$

The work of this procedure can be expressed as follows. This operator takes two elements –*A, B*- as parameters and does the following:

- If A is non-maximal cube then $\hat{X}(A, B) = \{A\}$,
- If B eliminates A then $\hat{X}(A, B) = \{B\}$,
- If neither A nor B is eliminated then $\hat{X}(A, B) = \{A, B\}$.

C. Non-disjoint Sharp Operator –$\hat{S}$

This MVL operator is introduced in [36] as well. For computing local primes, this process is the final procedure. After eliminating the weak elements by using the X operator, the cubes remained in S_B set are going to be trimmed. The sharp procedure starts with using a full-cube which is a cube where all coordinates are assigned to "don't care", $F = X_{n-1}^{*} X_{n-2}^{*} ... X_0^{*}$. A trimmig can be done on any coordinate of the cube F, if that coordinate is not assigned "don't care".

$$\hat{S}(A,B) = A\#B = \begin{cases} \emptyset & \text{when } A \subseteq B \\ A & \text{when } A \cap B = \emptyset \\ A\#_{basic}B & \text{otherwise} \end{cases} \quad (3)$$

https://doi.org/10.17758/DIRPUB1.E0318012

Algorithm 1 MVL-MIN Algorithm

$S_{now} = S_{on}$
while $S_{now} \neq \emptyset$ **do**
  $A \in S_{now}$ // Choose a cube from the set
  $S_B = \emptyset$
  for $\forall B \in S_{off}$ **do**
    $S_B = S_B \cup \widehat{E}(A, B)$
  end for
  for $\forall C, D \in S_B$ where $C \neq D$ **do**
    $S_B = S_B - \{C, D\} \cup \widehat{X}(C, D)$
  end for
  $S_P = \{X^*_{n-1} X^*_{n-2} \ldots X^*_0\}$
  for $\forall C \in S_B$ **do**
    $S_{Pb} = \emptyset$
    for $\forall D \in S_P$ **do**
      $S_{Pb} = S_{Pb} \cup \widehat{S}(D, C)$
    end for
    $S_P = S_{Pb}$
  end for
end while

IV. MVL-MIN MINIMIZATION ALGORITHM

Due to high complexity of the existing direct-cover minimization methods may be avoided by using the above-mentioned proposition that suggests to expand of cubes *Soff* instead of all possible expanding of the on-set cube, for which it is necessary to obtain the complete sets of prime implicants. This can be done by applying the algorithm 1 for a single output logic function as follows.

V. DATA SETS AND TESTING

A. Blif-MV Format

An MV function can be expressed as a netlist of MV-nodes. An MV-network is a network of nodes; each node represents an MV-relation with a single multiple-valued output. Here we use a subset of the "blif-MV" format that is used to specify MVL functions [37]. *blif-MV* format is a standart by the Verilog to *blif-MV* (vl2mv) tool. After a design specification is read, it is converted into an MV-network, a design representation used within MVL-MIN.

blif-MV is a file format designed for describing hierarchical symbolic sequential MV systems. A system can be composed of interacting sequential subsystems.

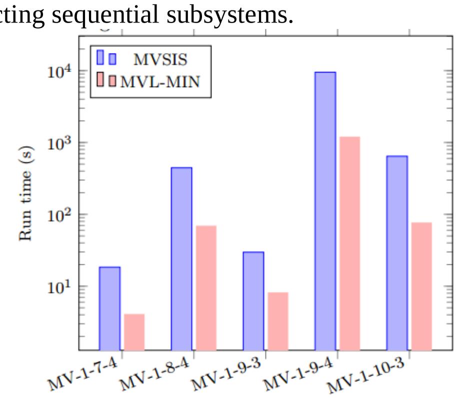

Fig. 1: Run time comparison of MVSIS and MVL-MIN on the first dataset

A multiple-valued logic can be expressed with four basic primitives: multiple-valued variables, tables, wires and latches. A MV-variable takes values from some finite domain. A relation defined over a set of variables is represented by a table. The variables of a table are divided into inputs and outputs. Wires are used connection among tables. A latch is a memory unit saves the value of the input wire, and transfers the values to the output [2].

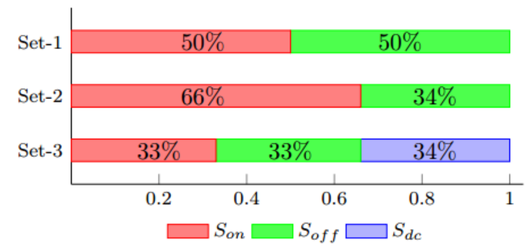

Fig. 2: The Structure of Data Sets

Due to small size of conventional testbenches, we generated 3 sets, each includes 8 multiple-valued logic functions in MV-blif format. The numbers of input variable in Datasets are ranging from 4 to 12. Nomenclature of data file names are based on input variable count and domain of a variable. For example the file names have the form MV-g-v-r, where g is the group number, v is the MV variable count and r is the domain of a variable. MVL functions in the first benchmark group consists of %50 S_{on} cubes and %50 S_{off} cubes. The distribution of the second group test files are %66 S_{on}, %34 S_{off}.

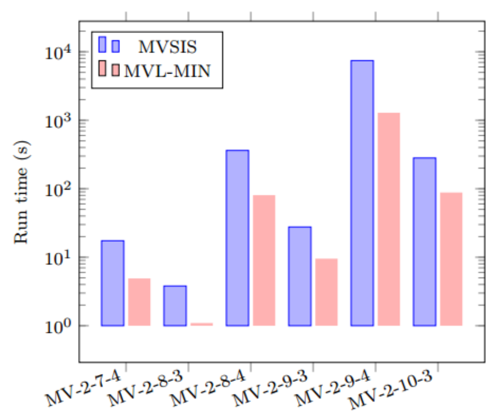

Fig. 3: Run time comparison of MVSIS and MVL-MIN on the second dataset

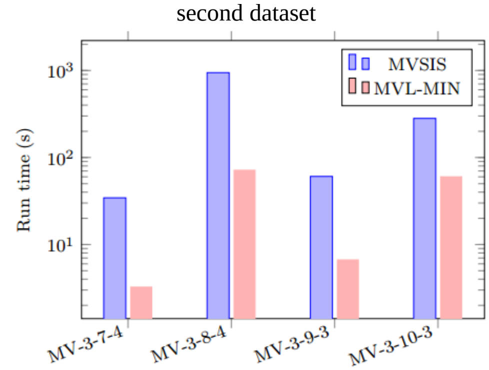

Fig. 4: Run time comparison of MVSIS and MVL-MIN on the third dataset

The third group is constructed evenly su%33 S_{on}, %33 S_{off} and %34 S_{dc}. The fig. 1 depicts the features of the data sets.

https://doi.org/10.17758/DIRPUB1.E0318012

MVL-MIN achieved around 5x speedup over MVSIS. A comparison with MVSIS shows that the proposed algorithm is able to solve all test files within a fixed allocation of computer resources.

VI. Results and Future Applications

A heuristic Multiple-valued logic function simplification algorithm has been introduced. This algorithm uses MV cube calculus operations which are "expansion," "elimination," and "Non-Disjoint Sharp."

This can be adapted into the Verification Interacting with Synthesis (VIS), which is a research tool. This application operates on MVL networks as well [38]. Otherwise a new MVL synthesis tool can be compiled by integrating our tool with "vl2mv" and a place and route program, such as ODIN [39].

Marek et. al. accelerated MVL operations on GPUs and on FPGAs [namely cube calculus machine]. We want to compare our algorithm with others but since most of them are using AI, they can only handle limited MVL functions (usually 2-3 MV variable input). The columns labeled as $s(S_{on})$ and $s(S_{off})$ depicts the number of the cubes in S_{on} and S_{off} respectively.

TABLE I
Timing Results of MVSIS and MVL-MIN on Benchmark Set 1

Test Bench	n*	r**	$s(S_{on})$	$s(S_{off})$	Time (s) MVSIS	Time (s) MVL-MIN	Speedup
MV-1-4-3	4	3	40	41	0.030	0.035	0.9
MV-1-4-4	4	4	134	122	0.330	0.192	1.7
MV-1-5-3	5	3	112	131	0.460	0.310	1.5
MV-1-7-3	7	3	1105	1082	3.270	0.704	4.6
MV-1-7-4	7	4	8250	8134	18.410	4.109	4.5
MV-1-7-5	7	5	39152	38973		79.675	
MV-1-8-3	8	3	3362	3199	9.080	0.934	9.7
MV-1-8-4	8	4	32678	32858	445.250	69.054	6.4
MV-1-8-5	8	5	195151	195474		2204.622	
MV-1-9-3	9	3	9732	9951	29.740	8.213	3.6
MV-1-9-4	9	4	131059	131085	9513.500	1207.837	7.9
MV-1-10-3	10	3	29250	29799	644.380	77.657	8.3
MV-1-10-4	10	4	523670	524906		19089.376	
MV-1-11-3	11	3	88391	88756		738.118	
MV-1-12-3	12	4	265355	266086		6714.434	

* # of Input variable
** Radix

TABLE II
Timing Results of MVSIS and MVL-MIN on Benchmark Set 2

Test Bench	n	r	$s(S_{on})$	$s(S_{off})$	Time (s) MVSIS	Time (s) MVL-MIN	Speedup
MV-2-4-3	4	3	57	24	0.100	0.125	0.8
MV-2-4-4	4	4	167	89	0.120	0.087	1.4
MV-2-5-3	5	3	165	78	0.180	0.114	1.5
MV-2-7-3	7	3	1436	751	1.270	0.746	1.7
MV-2-7-4	7	4	10991	5393	17.310	4.918	3.5
MV-2-7-5	7	5	51958	26167		92.744	
MV-2-8-3	8	3	4421	2140	3.870	1.068	3.6
MV-2-8-4	8	4	43781	21755	362.130	80.271	4.5
MV-2-8-5	8	5	260026	130599		2461.869	
MV-2-9-3	9	3	13020	6663	27.660	9.558	2.9
MV-2-9-4	9	4	174746	87398	7436.280	1286.171	5.8
MV-2-10-3	10	3	39391	19658	281.290	88.300	3.1
MV-2-10-4	10	4	699151	349425		22029.749	
MV-2-11-3	11	3	118656	58491		895.367	
MV-2-12-3	12	4	354591	176850		7815.372	

TABLE III
Timing Results of MVSIS and MVL-MIN on Benchmark Set 3

Test Bench	n	r	$s(S_{on})$	$s(S_{off})$	Time (s) MVSIS	Time (s) MVL-MIN	Speedup
MV-3-4-3	4	3	30	33	0.050	0.094	0.5
MV-3-4-4	4	4	105	114	0.090	0.113	0.7
MV-3-5-3	5	3	106	111	0.680	0.461	1.4
MV-3-7-3	7	3	966	977	0.690	0.083	8.3
MV-3-7-4	7	4	6822	6769	34.550	3.209	10.8
MV-3-7-5	7	5	31016	31373		54.173	
MV-3-8-3	8	3	2917	2891	6.720	0.775	8.7
MV-3-8-4	8	4	27245	27361	944.370	72.447	13.0
MV-3-8-5	8	5	155985	156462		1409.629	
MV-3-9-3	9	3	8666	8844	60.850	6.788	8.9
MV-3-9-4	9	4	109345	109129		815.900	
MV-3-10-3	10	3	26339	26217	766.950	60.972	12.6
MV-3-10-4	10	4	436750	437043		13591.574	
MV-3-11-3	11	3	78795	78489		570.188	
MV-3-12-3	12	4	236031	236300		5290.462	

References

[1] J.T. Butler, "Multiple-Valued Logic in VLSI," IEEE Computer Society Press Technology Series, Los Alamitos, CA, 1991.

[2] T. Furusho, T. Nishi and M. Konishi, "Distributed Optimization Method for Simultaneous Production Scheduling and Transportation Routing in Semiconductor fabrication bays," Int. J. of Inn. Com., Inf. and Con., vol.4, no.3, pp.559-578, 2008.

[3] S. L. Hurst. "Multiple-valued logic - its Status and Future." IEEE Trans. Comput., vol. 33, pp. 1160-1179, 1984. https://doi.org/10.1109/TC.1984.1676392

[4] E. Ozer, R. Sendag, and D. Gregg, "Multiple-Valued Logic Buses for Reducing Bus Size, Transitions and Power in Deep Submicron Technologies," Adv. Net. and Comm. Hard. Work. (ANCHOR), June 2005.

[5] E. Dubrova, "Multiple-Valued Logic Synthesis and Optimization," Hassoun S. and Sasao T., editors, Logic Synthesis and Verification, Kluwer Acad. Pub., pp. 89-114, 2002. https://doi.org/10.1007/978-1-4615-0817-5_4

[6] S. Lin, Y.B. Kim, F. Lombardi, and Y.J. Lee, "A new SRAM cell design using CNTFETs," Proceedings of the International SoC Design Conference, pp. 168-171, Nov. 2008. https://doi.org/10.1109/SOCDC.2008.4815599

[7] E. Ozer, R. Sendag, and D. Gregg, "Multiple-valued logic buses for reducing bus energy in low-power systems," IEE Proceedings of Computers and Digital Techniques, vol. 153, pp. 270-282, Jul. 2006. https://doi.org/10.1049/ip-cdt:20050160

[8] D. M. Miller and M. A. Thornton, "Multiple valued logic: Concepts and representations," Synthesis lectures on digital circuits and systems, vol. 2, pp. 1-127, 2007. https://doi.org/10.2200/S00065ED1V01Y200709DCS012

[9] C.S. Lent, P.D. Tougaw, W. Porod, and G.H. Bernstein, "Quantum cellular automata," Nanotechnology, vol. 4, pp. 49-57, Jan. 1993. https://doi.org/10.1088/0957-4484/4/1/004

[10] K. Matsumoto, M. Ishii, K. Segawa, Y. Oka, B.J. Vartanian, and J.S. Harris, "Room temperature operation of a single electron transistor made by the scanning tunneling microscope nanooxidation process for the TiOx/Ti system," App. Phy. Lett., vol. 68, pp.34-36, Jan. 1996. https://doi.org/10.1063/1.116747

[11] R. Martel, T. Schmidt, H.R. Shea, T. Hertel, and P. Avouris, "Single- and multi-wall carbon nanotube field-effect transistors," App. Phy. Lett., vol. 73, pp. 2447-2449, Oct. 1998. https://doi.org/10.1063/1.122477

[12] R. F. Mirzaee, M.H. Moaiyeri, M. Maleknejad, K. Navi, and O. Hashemipour, "Dramatically low-transistor-count high-speed ternary adders," IEEE 43rd International Symposium on Multiple-Valued Logic, pp. 170-175, May 2013. https://doi.org/10.1109/ISMVL.2013.24

[13] S. L. Murotiya, "Low-Power High-Speed and Compact Ternary VLSI Circuit Designs using Carbon Nanotube Field Effect Transistors." PhD Dissertation 2015.

[14] L.S. Phanindra, M N Rajath, V Rakesh, K S V. Patel "A novel design and implementation of multi-valued logic arithmetic full adder circuit using CNTFET," 2016 IEEE Int. Conf. on Rec. Trends in Elect., Inf. & Comm. Tech. (RTEICT). 20-21 May 2016 Bangalore, India. https://doi.org/10.1109/RTEICT.2016.7807885

[15] IBM, "Milestone Advances Effort to Enhance Semiconductors through Nanotechnology." Yorktown Heights, NY, USA, 2006, Mar 24.

[16] S. J. Tans, A. R. M. Verschueren, and C. Dekker, "Room temperature transistor based on a single carbon nanotube," Nature, vol. 393, no. 6680, pp. 49-52, May.1998.

[17] V. Derycke, R. Martel, J. Appenzeller, and P. Avouris, "Carbon Nanotube Inter- and Intramolecular Logic Gates," Nano Lett., vol. 1, no. 9, pp. 453-456, 2001. https://doi.org/10.1021/nl015606f

[18] A. Bachtold, P. Hadley, T. Nakanishi, and C. Dekker, "Logic Circuits with Carbon Nanotube Transistors," Science, vol. 294, no. 5545, pp. 1317-1320, Nov. 2001. https://doi.org/10.1126/science.1065824

[19] Z. Chen, J. Appenzeller, Y. M. Lin, J. S. Oakley, A. G. Rinzler, J. Tang, S. J. Wind, P. M. Solomon, P. Avouris, "An integrated logic circuit assembled on a single carbon nanotube," Science, vol. 311, no. 5768, pp. 1735, Mar. 2006. https://doi.org/10.1126/science.1122797

[20] N. Patil, A. Lin, E. Myers, H. S. P.Wong andS. P. Mitra, "Integrated wafer-scale growth and transfer of directional carbon nanotubes and misaligned-carbonnanotube-immune logic," in Proc. Symp. VLSI Tech., Honolulu, HE, pp. 205 206, 2008.

[21] N. Patil, A.Lin, Z. Jie, W. Hai,K. Anderson, H.-S.P. Wong, S. Mitra, "Scalable carbon nanotube computational and storage circuits immune to metallic and mispositioned carbon nanotubes," IEEE Trans. Nano Tech., vol.10, no. 4, pp. 744 750, July 2011.

[22] M. Shulaker, J. V. Rethy, G. Hills, H. Chen, G. Gielen, H.P. Wong, S. Mitra, "Experimental demonstration of a fully digital capacitive sensor interface built entirely using carbon-nanotube FETs," in Proc. IEEE Int. Solid-State Circ. Conf. Dig. of Tech. papers (ISSCC), Feb. 2013,pp. 112113. https://doi.org/10.1109/ISSCC.2013.6487660

[23] Q. Cao, H.-S. Kim, N. Pimparkar, J. P. Kulkarni, C. Wang, M. Shim, K. Roy, M. A. Alam, J. A. Rogers, "Medium-scale carbon nanotube thin-film integrated circuits on flexible plastic substrates," Nature, vol. 454, no. 7203, pp. 495-500, Jul. 2008. https://doi.org/10.1038/nature07110

[24] M. Shulaker, G. Hills, N. Patil, H. Wei, H. Y. Chen, H. S. P. Wong and S. Mitra, "Carbon nanotube computer," Nature, vol. 501, no. 7468, pp. 526-530, Sep. 2013. https://doi.org/10.1038/nature12502

[25] A. Srivastava and H.N. Venkata, "Quaternary to Binary Bit Conversion CMOS Integrated Circuit Design Using Multiple Input Foating Gate MOSFETs," Integration VLSI J. vol.36 no.3, pp. 87-101, 2003. https://doi.org/10.1016/S0167-9260(03)00049-X

[26] M. Gao, J.H. Jiang, Y. Jiang, Y. Li, S. Sinha and R. Brayton "MVSIS," In the Notes of the International Workshop on Logic Synthesis, Tahoe City, June 2001

[27] G.W. Dueck and D.M. Miller, "A direct cover MVL minimization Using the Truncated Sum," Proc. ofthe 17th IEEE Int. Symp. on Multiple-Valued Logic, pp. 221-226, 1987.

[28] G.W. Dueck, "Direct cover MVL Minimization with CostTables," Proc. of the 22nd IEEE Int. Symp. on MultipleValued Logic, pp.58-65, 1992.

[29] Z. Tang, Q. Cao, and O. Ishizuka, "A Learning Multiplevalued Logic Network: Algorithm and applications," IEEE Trans. on Computers, vol.47, no.2, pp. 247-250, 1998. https://doi.org/10.1109/12.663773

[30] Q. Cao, Z. Tang, R. Wang, and W. Wang, "Local Search Based Learning Method for Multiple-valued Logic Networks," IEICE Trans. Fundamentals, vol.E86-A, no.7, pp. 1876-1884, 2003.

[31] W. Wang and C. Moraga, "Design of Multivalued Circuits Using Genetic Algorithms," Proc. of the 26th IEEE Int. Symp. on Multiple-valued Logic, pp. 216-221, 1996. https://doi.org/10.1109/ISMVL.1996.508361

[32] Y. Hata, K. Hayase, and T. Hozumi, "Multiple-valued Logic Minimization by Genetic Algorithms," Proc. of the 27th IEEE Int. Symp. on Multiple-valued Logic, pp. 97- 102, 1997. https://doi.org/10.1109/ISMVL.1997.601380

[33] B. Sarif and M. Abd-El-Barr, "Synthesis of MVL functions - part I: The Genetic Algorithm Approach," Int. Conference on Microelectronics, pp. 154-157, 2006. https://doi.org/10.1109/ICM.2006.373290

[34] S. Gao, Z. Tang, C. Vairappan "An Effective Immune Algorithm for Multiple-Valued Logic Minimization Problems," Int. Jour. of Innov. Com., Inf. and Con. 5, 11-A, pp. 3961-3969, 2009.

[35] S. Kahramanli, S. Tosun "A Novel Essential Prime Implicant Identification Method for Exact Direct Cover Logic Minimization," The Int. Con. on Com. Des.,CDES'06, June 26-29,2006, Las Vegas, USA.

[36] M. Perkowski, D. Foote, Q. Chen, A. Al-Rabadi, Jozwiak, L. "Learning Hardware Using Multiple-valued Logic, Part 2: Cube Calculus and Architecture," 2002.

[37] R. K. Brayton, M. Chiodo, R. Hojati, T. Kam, K. Kodandapani, R. P. Kurshan, S. Malik, A. L. SangiovanniVincentelli, E. M. Sentovich, T. Shiple, K. J. Singh, and H.Y. Wang. BLIF-MV: An Interchange Format for Design Verification and Synthesis. Technical Report UCB/ERL M91/97, Electronics Research Lab, Univ. of California, Berkeley, CA 94720, November 1991.

[38] R. Alur and T. Henzinger, "VIS: A system for Verification and Synthesis", The VIS Group, In the Proceedings of the 8th International Conference on Computer Aided Verification, Springer Lecture Notes in Computer Science, 1102, Edited by New Brunswick, NJ, pp. 428-432, July 1996.

[39] P. Jamieson, K.B. Kent, F. Gharibian and L. Shannon,"Odin II - An Open-source Verilog HDL Synthesis tool for CAD Research," Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines, pp. 149-156, 2010. https://doi.org/10.1109/FCCM.2010.31