Three Dimensional Multi-Valued Design in Nanoscale Integrated Circuits

Sergey Edward Lyshevski
Department of Electrical Engineering
Rochester Institute of Technology
Rochester, New York 14623-5603, USA
E-mail: Sergey.Lyshevski@rit.edu

Abstract

Novel three-dimensional (3D) nanoscale integrated circuits (nanoICs) are examined in this paper. These nanoICs are synthesized utilizing aggregated 3D neuronal-hypercells (ℵ-hypercells) with multi-terminal electronic nanodevices. The proposed nanodevices ensure multi-valued input-output characteristic that lead to a direct technological solution of multi-valued logic synthesis problem. Super-high-performance computing architectures and memories can be devised (synthesized), designed and optimized. At the system-level, we examine nanoICs as networked aggregated 3D ℵ-hypercells. In particular, scalable 3D ℵ-hypercell topologies are under consideration. These ℵ-hypercells integrate interconnected functional multi-terminal electronic nanodevices that implement logic functions. The proposed nanoICs platform suits the envisioned cognizant computing ensuring preeminent information processing and immense memory.

1. Introduction

Two-dimensional computing architectures have been examined extensively and exceptional solutions have been proposed. However, two-dimensional paradigm approaches fundamental limits. Optimal topologies and structures of the super-high-performance computer architecture are three-dimensional [1-3]. New areas in design of nanoICs have been emerged including *nanoarchitectronics* [1]. It has been shown that functional multi-terminal electronic nanodevices exhibit multi-valued input-output characteristics [2]. Thus, multi-valued logic becomes a reality that can be integrated within advanced networked 3D ℵ-hypercell topologies. These hypercells are mapped in 3D space performing logic design, optimization and analysis tasks. This paper reports technology-dependent synthesis and design concepts for nanoICs. A library of 3D ℵ-hypercells can be developed and utilized. Three-dimensional design methods are based on embedding decision diagrams [3, 4] extending results reported in [5-14]. In this paper we generalize approaches in design of 3D ℵ-hypercell topologies. We report methods of manipulation, representation and optimization of multi-valued nanoICs with the ultimate goal to perform the logic design and synthesize nanoICs.

2. Synthesis and Design Taxonomy

We propose a novel 3D computing architecture focusing on the unified technology-dependent top-down and bottom-up synthesis taxonomy. As reported in Figure 1.a, by utilizing this synthesis and design taxonomy, one can coherently perform integrated top-down and bottom-up syntheses.

Top-down synthesis: devise novel super-high-performance computing platforms implemented utilizing aggregated 3D ℵ-hypercells with electronic nanodevices (see Figures 1.b and 1.c);

Bottom-up synthesis: use functional molecules (interconnected multi-terminal electronic nanodevices) aggregated within 3D ℵ-hypercells, and utilize these aggregated (networked) ℵ-hypercells within computing architectures.

Three-dimensional computing architectures can be synthesized using ℵ-hypercells D_{ijk} that are analogous to multi-terminal neurons in the superb brain bioarchitecture shown in Figure 1.b. The reported concurrent synthesis taxonomy results in a radically new computing platform that utilize novel phenomena, advanced design, as well as new technologies at system-, subsystem- (aggregated ℵ-hypercells) and device-levels, see Figure 1.c. Synthesis and optimization of computing architectures is formulated using the Bayesian probability theory that results in a viable decision-theoretic informative analysis [1, 3]. The topological entropy E_T is a function of 3D computing platform performance as well as characteristics of ℵ-hypercells and ℵ-hypertopologies.

Proceedings of the 35th International Symposium on Multiple-Valued Logic (ISMVL'05)
0195-623X/05 $ 20.00 **IEEE**

The entropy can be maximized ensuring superior achievable performance [1]. Specifically, efficiency, bandwidth, reliability, compactness, simplicity, robustness and other parameters can be maximized while minimizing power and losses. Special attention can be focused to ensure distinct priorities, for example, robust massive parallel computing.

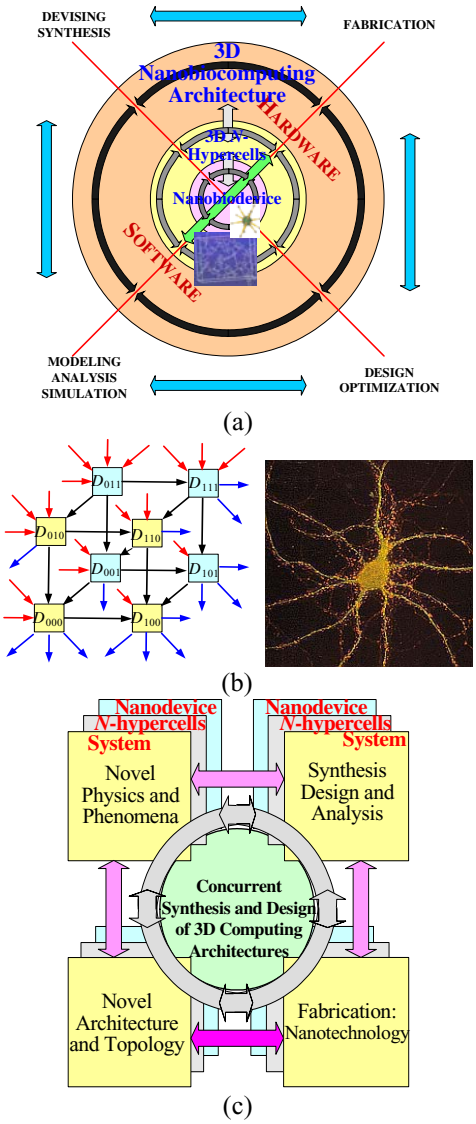

Figure 1.
(a) Top-down and bottom-up taxonomy;
(b) Three-dimensional architectures synthesized using ℵ-hypercells D_{ijk} (neurons);
(c) Concurrent synthesis and design at system-, subsystem- and device- levels.

Synthesis of novel computing architectures is a process to discover and study novel hardware topological evolutions based upon fundamental theory and synergetic integration of nanodevices in the unified functional core. High-level hierarchy, abstraction, adaptability, functionality, integrity, compliance, configurability, flexibility and prototypeability are integrated within proposed 3D ℵ-hypercell platform. Coherent quantitative synthesis and symbolic descriptions are used searching and evaluating possible architectures and topologies examining ℵ-hypercells aggregated in 3D ℵ-hypercells topologies [1]. Massive parallelization is due to the fact that data structures are embeddable to ℵ-hypercells and $3D^A$ ℵ-hypertopologies. Logic functions can be represented by word-level decision trees and diagrams [3]. This is a very important feature to ensure massive parallel computing. The node implements the processing of a set of functions reduced into a word. There are two unique parallelization sources that result in massive parallel computing: (1) *natural* parallelism of 3D ℵ-hypercells and $3D^A$ ℵ-hypertopologies is utilized (decision trees that represent logic functions are embedded into ℵ-hypercells); (2) *enhanced* parallelism due to the word-level representation of logic functions (each node performs logic computations on the bits in the words in parallel).

3. Synthesis and Design of ℵ-Hypercells

We start with useful definitions.

Definition 1. A multi-valued variable X_i can take values from $P_i = \{a_0, a_1, \ldots, a_{|P_i|-2}, a_{|P_i|-1}\}$. If each symbolic value a_i is associated with a unique integer i, we have $P_i = \{0, 1, \ldots, |P_i|-2, |P_i|-1\}$.

Definition 2. A multi-valued function F is a function which maps vertices in $P_1 \times P_2 \times \ldots \times P_{n-1} \times P_n$ to P_F, e.g., $F : P_1 \times P_2 \times \ldots \times P_{n-1} \times P_n \rightarrow P_F$.

Definition 3. A cube $c = c_1 \times c_2 \times \ldots \times c_{n-1} \times c_n$ can be written as a product of multi-valued literals $X_i^{c_i}$ as $X_1^{c_1} X_2^{c_2} \ldots X_{n-1}^{c_{n-1}} X_n^{c_n}$. Here, $X_i^{c_i}$ is the logic function as $X_i^{c_i} = (X_i = b_1) + \ldots + (X_i = b_k), b_j \in c_i \subseteq P_i$.

Definition 4. The cofactor of a function f with respect to a multi-valued literal X^s, denoted as f_{X^s}, is obtained by eliminating all cubes of f that are disjoint to s, and expanding the remaining cubes by unioning into the X position all values not in s.

Multi-Valued Shannon Expansion Theorem. Let f be a function, and $\{c^1 \times c^2 \times \ldots \times c^{i-1} \times c^i\}$ be a set of multi-valued cubes such that $\sum_{i=1}^{N} c^i = 1$. Then, $f = \sum_{i=1}^{N} c^i f_{c_i}$. Furthermore, $f \equiv 1$ iff $f_{c^i} = 1, \forall i$.

There are several methods for representing logic functions in the multi-valued domain. ℵ-hypercell is a core technology-defined subsystem in computing architecture. Each ℵ-hypercell can be considered as a homogeneous aggregated assembly for massive super-high-performance parallel computing. In [3, 4], we applied the switching theory in logic design of 3D nanoICs in [3]. To perform logic design, the graph-based data structures and 3D circuit topology was utilized. The ℵ-hypercell is a topological representation of a switching function by *n*-dimensional graph, and the switching function *f* is given as

Switching Function

$$\Rightarrow \underset{\text{Operation}}{\overset{2^n-1}{\underset{i=0}{\mathbf{L}}}} \overset{\text{Coefficient}}{\mathbf{K}_i} \left(x_1^{i_1} \ldots x_n^{i_n}\right) \Rightarrow \text{Form of Switching Function}$$

The data structure is described in matrix form using the truth vector ***F*** of a given switching function *f* as well as the vector of coefficients **K**. The logic operations are represented by **L**.

ℵ-hypercells are used to compute switching functions. To illustrate the concept, in [3] distinct switching functions *f* (for example, $f = \bar{x}_1 x_2 \vee x_1 \bar{x}_2 \vee x_1 x_2 x_3$) were implemented by the *N*-hypercube, see Figure 2.a. From the technology-dependent implementation viewpoint, we proposed the ℵ-hypercell as documented in Figure 2.b. This topology maps the device-level consideration. Utilizing the root and intermediate nodes at the edges, as shown in Figure 2, the reported ℵ-hypecell implements *f*.

The logic design in spatial dimensions is based on advanced methods and data structures that fit 3D topology. The appropriate data structure of logic function and methods of embedding this structure in the ℵ-hypercells must be found. The three-step-solution in logic function manipulation to change the carrier of information from the algebraic form (logic equation) to the hypercube structure is reported [3]. In particular, we proceed as follows.

Step 1: Logic function (switching or multi-valued) is transformed to the appropriate algebraic form (Reed-Muller, arithmetic or word-level in matrix or algebraic representation).

Step 2: Derived algebraic form is converted to the graphical form (decision tree, decision diagram or logic network).

Step 3: Obtained graphical form is embedded into ℵ-hypercell.

The design is expressed as

Logic Function ⇔ Graph ⇔ *N* - Hypercell Structure

Step 1 Step 2 Step 3

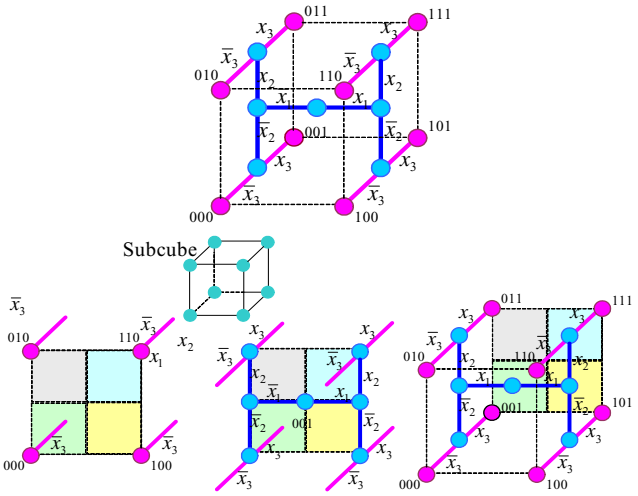

Figure 2. *N*- and ℵ-hypercubes that implement function *f*

The proposed procedure results in:

- Algebraic representations and rules of manipulations with switching and multi-valued logic functions;
- Matrix representations and rules of manipulations with switching and multi-valued logic functions (matrix representations provide a consistent understanding of logic relationships for variables and functions from the viewpoint of spectral theory);
- Graph-based representations are found using decision trees, decision diagrams and logical networks;
- Data structures are embedded into ℵ-hypercells.

Definition 5. A ternary Shannon expansion $f = x_i^0 f_0 \vee x_i^1 f_1 \vee x_i^2 f_2$ is performed in every node of the ternary decision tree. Here, for example, $f_0 = f(x_i = 0)$, $f_1 = f(x_i = 1)$ and $f_2 = f(x_i = 2)$. The complete *n*-level ternary decision tree C_n, is a tree with 3^k nodes at the *k*th level, $k = 0, 1, \ldots, n - 1$.

Definition 6. The ℵ-hypercell is an extended classical hypercube that consists of terminal nodes, intermediate nodes, root node and interconnect.

Figure 3 documents the ternary function *f* of two variables (x_1 and x_2) and its implementation. Letting $x_1^0 = 0, x_1^1 = 1, x_1^2 = 2$ and $x_2^0 = 0, x_2^1 = 1, x_2^2 = 2$, we have corresponding $f_{x_1^{0,1,2} x_2^{0,1,2}}$, e.g., 00, 01 and 02.

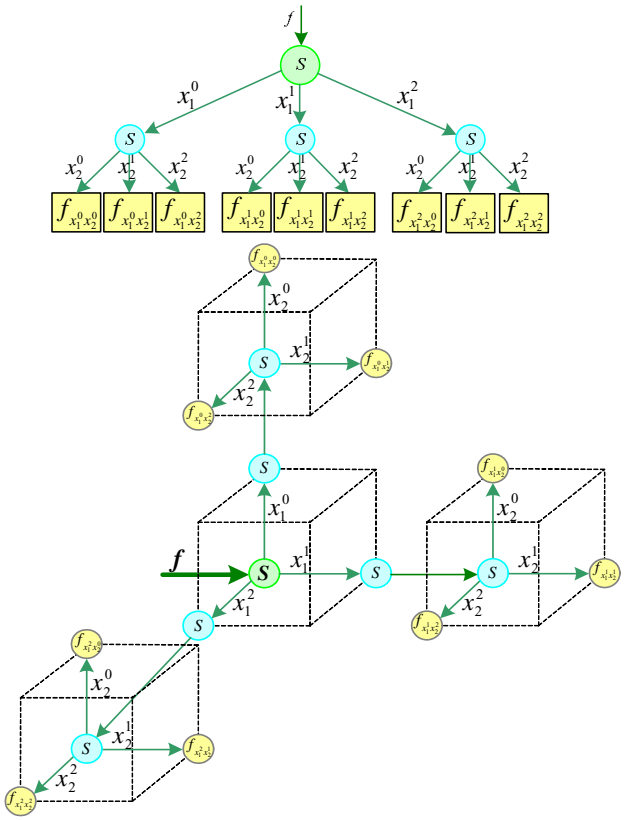

Figure 3. Ternary function f of two variables (x_1 and x_2) and its implementation using N-hypercubes

The ternary as well as higher order decision trees can be embedded in ℵ-hypercells. The algorithm embedding a ternary and quaternary decision trees is illustrated by Figure 4. Letting q be the number of *variables* of a ternary function, the numbers of terminal and intermediate nodes in a ℵ-hypercell that represents a ternary decision trees are $3q$ and $3q$-1. The number of terminal and intermediate nodes in a ℵ-hypercell that represents a quaternary decision trees are $4q$ and $4q$-1.

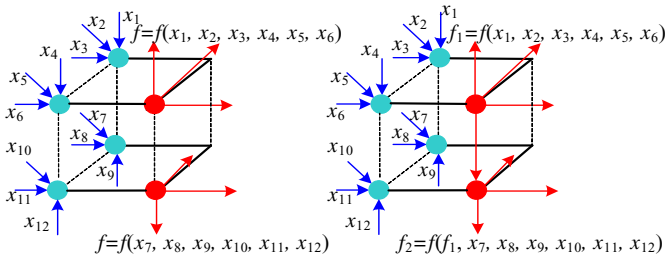

Figure 4. Embedding a ternary (quaternary) decision trees of two-variable ternary function into ℵ-hypercells

The N-hypercube primitives for Boolean ternary, quartery as well as higher order functions are easy to derive and apply in the logic design. The library of 3D elementary gates, such as MIN, MAX, SUM, TSUM and TPRODUCT, was developed in [3]. It is documented that distinct decision trees are used to design 3D MIN and MAX ℵ-hypercell models. Each gate consists of three ℵ-hypercell primitives. The number of activated terminal nodes is defined by a logic function. In particular, the MIN gate has four active nodes, while the MAX gate generates 8 active nodes. In general, an arbitrary k-valued network can be represented in ℵ-hypercell space over the library of 3D primitives (e.g., gate primitives) because k-valued gate can be represented by a k-ary decision tree that is mapped to a ℵ-hypercell. Correspondingly, a network of gates can be represented by a set of a k-ary trees. These trees are mapped into a ℵ-hypercell topology formed by connecting the ℵ-hypercell primitives. The inner and outer ℵ-hypercells are three dimensional [3]. Each 3D ℵ-hypercell carries limited information because the number of nodes and links is limited. The 3D ℵ-hypercells are aggregated in the 3D ℵ-hypercells topologies. This technology-dependent approach allows one to design novel architectures from far-reaching physical, implementation, and technological viewpoints.

4. Synthesis of Multi-Valued Logic Networks Using Multi-Valued Decision Diagrams

We examine a method for the synthesis of large multi-valued logic networks with ℵ-hypercell (MVℵ) using multi-valued decision diagrams. These decision diagrams can be mapped to netlists without reversing the information flow. The size of the resulting MVℵ is linear in the size of the decision diagram.

In general, a MVℵ can be modeled as a directed acyclic graph $C = (V, E)$ such that each vertex $v \in V$ is labeled with the names of (1) basic cell, (2) *input* or (3) *output*. The collection of basic cells that are possible in the MVℵ is given by a fixed library that contains MIN, MAX, INV and LITERAL gates. It is possible to include basic ℵ-hypercell with arbitrary complexity and with a varying number of inputs. There is an edge (u, v) in E from vertex u to v if and only if an output pin of the cell associated with u is connected to an input pin of the cell associated with v. Edges also contain additional information to specify the pins of the source and sink nodes that they are connected to. Vertices have exactly one incoming edge per input pin. The nodes labeled as *input/output* may not have

incoming (outgoing) edges. To simulate a MVℵ, each *input* may assume the values of a given ordered finite set $P = \{0, 1, \ldots, k–1\}$, where k denotes the number of logic levels. The complement (INV gate) of a signal x is defined as $\bar{x} = (k-1) - x$. A LITERAL gate $(a; b)$ $(a, b \in P, 0 \le a \le b < k)$ has one input and one output. For a given input x, the behavior of this gate is $f(x) = \begin{cases} k-1 : a \le x \le b \\ 0 : \text{ otherwise} \end{cases}$. We assume that the characteristic functions are available for each of the *inputs*. The set of $J_j(x_i)$ values as $J_j(x_i) = k - 1$ if $x_i = j$, and $J_j(x_i) = 0$ otherwise.

The binary decision diagram can be extended to represent functions

$$f : \mathbf{B}^n \to \{0, 1, \ldots, k-1\}.$$

The resulting graphs represent a *multi-terminal binary decision diagram* that can be extended to *multi-valued decision diagrams* that represent function $f : \{0, 1, \ldots, k-1\}^n \to \{0, 1, \ldots, k-1\}$.

When the MVℵ is represented as a directed acyclic graph, a corresponding decision diagram is designed as: (1) terminal nodes for the k constant functions are synthesized; (2) for each *input* of the MVℵ, a graph vertex (variable) in the decision diagram is designed, where the ith outgoing edge points to the terminal node labeled $i, i \in \{0, 1, \ldots, k-1\}$; (3) the gates of the MVℵ are checked in the topological order, and the corresponding decision diagram operation is designed.

For the synthesis process, assume that a multi-valued decision diagram, representing a k-valued function of k-valued variables, is given. The outgoing edges per node of a decision diagram are mapped to small sets of logic gates, producing a k-output circuit. If the function to being computed is $f(X)$, then the k outputs of the resulting circuit correspond to the characteristic functions of f, e.g., $J_0(f(X))$, …, $J_{k-1}(f(X))$. The circuit outputs thus form a 1 of k code, where the ith output of the circuit is logically true if and only if the multi-valued decision diagram would evaluate to logic value i. This represents a general procedure for mapping a multi-valued decision diagram to a MVℵ. It is assumed that the logic gates compute the MIN and MAX functions.

5. Ternary Field Logic

Ternary field consists of the set of elements $T = \{0, 1, 2\}$ and two basic binary operations, e.g., *addition* (+) and *multiplication* (· or absence of any operator) as defined in Table 1. These addition and multiplication are: (1) closed, e.g., for $x, y \in T$, $x + y \in T$, while $xy \in T$; (2) commutative and associative, e.g., $x + y = y + x$ and $xy = yx$ (commutative), $x + (y + z) = (z + y) + z = z + y + z$ and $x(yz) = (xy)z = xyz$ (associative); (3) multiplication is distributive over addition, e. g., $x(y + z) = xy + xz$.

Table 1. Ternary field operations

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

.	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

There are six reversible ternary unary operations corresponding to six possible permutations of 0, 1, and 2. These operations are called *reversible ternary shift operations* [16-19]. Six shift operations, their operator symbols and equations are reported in [16, 17]. All these six shift operators can be built as reversible ternary gates. The gate symbols for these shift gates are documented in Figure 5. A ternary signal can be converted to one of the six forms using one of the reversible ternary shift gates.

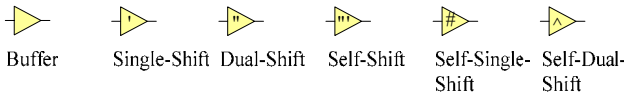

Figure 5. Gate symbols for shift gates

Literals of a ternary variable x can be defined. Any ternary function can be expanded using the following Shannon ternary field expansion theorem.

Theorem. A ternary function f can be expanded with respect to the variable x as $f = {}^0x f_0 + {}^1x f_1 + {}^2x f_2$.

An optimized ternary decision diagrams for reversible logic design for an n-variable function f can be synthesized. The number of possible variable ordering is $n!$. One has 16^n possible choices of ternary field expansions for n levels. Therefore, the total number of possible Kronecker ternary field decision trees for the function f is $n!16^n$. For $n = 3$, we have 24,576. The number of Kronecker decision diagrams is the same resulting in the NP-hard optimization problem. For a pseudo-Kronecker ternary field decision tree for an n-variable function f, the number of possible variable orderings is $n!$. However, the total number of possible choices of ternary field expansions is $16^{\frac{1}{2}(3^n-1)}$ for $\frac{1}{2}(3^n - 1)$ nodes. Therefore, the total number of possible decision trees for the function f is $n!16^{\frac{1}{2}(3^n-1)}$ =27,021,597,764,222,976 for $n = 3$.

6. Conclusions

This paper focuses on the development (synthesis), integration and demonstration of a novel technology-dependent 3DnanoICs architecture that will ensure superior processing capabilities. It is envisioned that the integration of the technology-dependent 3D logic design with molecular nanotechnology (that allows one to synthesize complex functional molecules) will result in revolutionary performance evolvements. Quantitative and qualitative performance indexes, such as intrinsic data-intensive processing, robust adaptive computing, enhanced functionality, reliability, redundancy, fault- and defect-tolerance and other, under specific criteria to be examined. Superior performance are due to massive parallelism implemented by 3D $\aleph$-hypercells in terms of logic design and nanotechnology implementation. Though we may possess a limited knowledge in the molecular electronics design and its technological implementation execution, from nanobiocomputing standpoints and *nanobioarchitectronics*, we enable to mimic in some extent superb 3D networked biomolecular architectures. Novel methods under the developments to design novel computing and memory systems.

7. References

[1] S. E. Lyshevski, *Nanocomputers and Nanoarchitectronics*, *Handbook of Nanoscience, Engineering and Technology*, Ed. W. Goddard, D. Brenner, S. Lyshevski and G. Iafrate, pp. 6.1-6.39, CRC Press, Boca Raton, FL, 2002.

[2] S. E. Lyshevski, *NEMS and MEMS: Fundamentals of Nano- and Microengineering*, CRC Press, Boca Raton, FL, 2005.

[3] S. Yanushkevich, V. Shmerko and S. E. Lyshevski, *Logic Design of Nano-ICs*, CRC Press, Boca Raton, FL, 2005.

[4] S. N. Yanushkevich, V. P. Shmerko, L. Guy and D. C. Lu, "Three dimensional multiple valued circuit design based on single-electron logic," *Proc. Int. Symposium on Multi-Valued Logic*, pp. 1-6, 2004

[5] R. Drechsler and D.M. Miller, "Decision diagrams in multi-valued logic," *Int. Journal Multi-Valued Logic*, vol. 4, pp. 1-8, 1998.

[6] B. Harking and C. Moraga, "Efficient derivation of Reed-Muller expansions in multiple-valued logic systems," *Proc. Int. Symp. Multi-Valued Logic,* Sendai, Japan, pp. 436-441, 1992.

[7] D. Jankovic, R.S. Stankovic and R. Drechsler, "Efficient calculation of fixed-polarity polynomial expressions for multiple-valued logic functions," *Proc. Int. Symp. Multi-Valued Logic*, Boston, Massachusetts, pp. 76-82, 2002.

[8] P. Kerntopf, "Multiple-valued decision diagrams based on generalized Shannon expansion," *Workshop on Post-Binary Ultra-Large-Scale Integration Systems (ULSI)*, Tokyo, pp. 9-16, 2003.

[9] K. L. Kodandapani and R.V. Setur, "Multi-valued algebraic generalization of Reed-Muller canonical forms," *Proc. Symp. Multi-Valued Logic,* pp. 505-544, 1974.

[10] D. M. Miller and R. Drechsler, "On the construction of multi-valued decision diagrams," *Proc. Int. Symp. Multi-Valued Logic*, Boston, MA, pp. 245-253, 2002.

[11] D. M. Miller and R. Drechsler, "Implementing a multiple-valued decision diagram package", *Proc. Int. Symp. Multi-Valued Logic*, Fukuoka, Japan, pp. 52-57, 1998.

[12] D.M. Miller and R. Drechsler, "Augmented sifting of multiple-valued decision diagrams," *Proc. 33rd IEEE Int. Symp. Multiple-Valued Logic*, Tokyo, pp. 375-382, 2003.

[13] T. Sasao, "Ternary decision diagrams: Survey," *Proc. Int. Symp. Multi-Valued Logic*, Antigonish, Nova Scotia, Canada, pp. 241-250, 1997.

[14] R. S. Stankovic, "Functional decision diagrams for multiple-valued functions," *Proc. Int. Symp. Multi-Valued Logic*, Bloomington, IN, pp. 284-289, 1995.

[15] L. Macchiarulo and P. Civera, "Ternary decision diagrams with inverted edges and cofactors – an application to discrete neural networks synthesis," *Proc. Int. Symp. Multi-Valued Logic*, pp. 58-63, 1998.

[16] A. Al-Rabadi and M. Perkowski, "Multiple-valued galois field S/D trees for GFSOP minimization and their complexity," *Proc. IEEE Int. Symp. Multi-Valued Logic,* Warsaw, Poland, pp. 159-166, 2001.

[17] K. M. Dill, K. Ganguly, R. J. Safranek, and M. A. Perkowski, "A new Zhegalkin galois logic," *Proc. Int. Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, Oxford University, UK, pp. 247-257, 1997.

[18] U. Kalay, D.V. Hall, and M. Perkowski, "Easily testable multiple-valued galois field sum-of-products circuits," *Int. Journal Multi-Valued Logic,* vol. 5, pp. 507-528, 2000.

[19] P. Kerntopf, "Maximally efficient binary and multi-valued reversible gates," *Workshop on Post-Binary Ultra-Large-Scale Integration Systems (ULSI),* Warsaw, Poland, pp. 55-58, 2001.