A Sound and Complete Axiomatization of Majority-n Logic

Luca Amarú, *Student Member, IEEE*, Pierre-Emmanuel Gaillardon, *Member, IEEE*, Anupam Chattopadhyay, *Senior Member, IEEE*, Giovanni De Micheli, *Fellow, IEEE*

***Abstract*— Manipulating logic functions via majority operators recently drew the attention of researchers in computer science. For example, circuit optimization based on majority operators enables superior results as compared to traditional synthesis tools. Also, the Boolean satisfiability problem finds new solution approaches when described in terms of majority decisions. To support computer logic applications based on majority, a sound and complete set of axioms is required. Most of the recent advances in majority logic deal only with ternary majority (MAJ-3) operators because the axiomatization with solely MAJ-3 and complementation operators is well understood. However, it is of interest extending such axiomatization to n-ary majority operators (MAJ-n) from both the theoretical and practical perspective. In this work, we address this issue by introducing a sound and complete axiomatization of MAJ-n logic. Our axiomatization naturally includes existing MAJ-3 and MAJ-5 axiomatic systems. Based on this general set of axioms, computer applications can now fully exploit the expressive power of majority logic.**

***Index Terms*— Majority Logic, Boolean Algebra, Axiomatization, Soundness, Completeness.**

I. Introduction

BOOLEAN logic and its axiomatization is fundamental to the whole field of computer science. Traditionally, Boolean logic is axiomatized in terms of conjunction (AND), disjunction (OR) and complementation (INV) operators. Virtually, all of today's digital computation is performed by using these operators with their associated laws. Recently, it was shown that more efficient logic computation is possible by using a majority operator in place of conjunction and disjunction operators [1]–[4]. Moreover, the properties of majority operators, such as stability, have been proved to be the best fit for solving important problems in computer science [5]–[8]. Regarding emerging technologies, majority operators are the natural logic primitives for several beyond-CMOS candidates [9]–[23]. In order to exploit the unique opportunity led by majority in computer applications, a sound and complete set of manipulation rules is required. Most of the recent studies on majority logic based computation consider ternary majority (MAJ-3) operators because the axiomatization in this context is well understood. To unlock the real expressive power of majority logic, it is of interest to extend such axiomatization to n-ary (n odd) majority operators (MAJ-n).

Luca Amarú, Pierre-Emmanuel Gaillardon and Giovanni De Micheli are with the Integrated Systems Laboratory, Swiss Federal Institute of Technology, Lausanne, EPFL, 1015 Lausanne, Switzerland (e-mail: luca.amaru@epfl.ch; pierre-emmanuel.gaillardon@epfl.ch; giovanni.demicheli@epfl.ch).

Anupam Chattopadhyay is with Nanyang Technological University, 639798, Singapore (e-mail: anupam@ntu.edu.sg).

We introduce in this paper a sound and complete axiomatization of MAJ-n logic. Our axiomatization is the natural extension of existing majority logic systems with fixed number of inputs. Based on the majority axioms introduced in this work, computing systems can use at its best the expressive power of majority logic.

The remainder of this paper is organized as follows. Section II gives background and notations useful for the rest of this paper. Section III introduces our sound and complete axiomatization for MAJ-n logic. Section IV discusses relevant applications of our majority logic system in logic optimization, Boolean satisfiability, repetition codes and emerging technologies. Section V concludes the paper.

II. Background and Notations

We provide hereafter terms and notions useful in the rest of the paper. We start by introducing basic notation and symbols for logic operators and we continue by presenting special properties of Boolean functions. We define a compact vector notation for Boolean variables and discuss Boolean algebras with a particular emphasis on MAJ-3/INV Boolean algebra.

A. Notations

In the binary Boolean domain, the symbol $\mathbb{B}$ indicates the set of binary values $\{0, 1\}$; the symbols $\wedge$ and $\vee$ represent the conjunction (AND) and disjunction (OR) operators; the symbol $\neg$ represents the complementation (INV) operator; and $\mathtt{0}/\mathtt{1}$ represent the false/true logic values. Alternative symbols for $\wedge$, $\vee$ and $\neg$ are $\cdot$, $+$, and $'$, respectively.

B. Self-Dual Function

A logic function $f(x, y, .., z)$ is said to be *self-dual* if $f(x, y, .., z) = \neg f(\neg x, \neg y, .., \neg z)$ [7]. By complementation, an equivalent *self-dual* formulation is $\neg f(x, y, .., z) = f(\neg x, \neg y, .., \neg z)$.

C. Majority Function

An n-input (n being odd) majority function M_n is defined on reaching a threshold $\lceil n/2 \rceil$ of true inputs [7]. For example, the three input majority function $M_3(x, y, z)$ can be expressed as $\wedge, \vee$ by $(x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$. Also $(x \vee y) \wedge (x \vee z) \wedge (y \vee z)$

is a valid representation for $M_3(x, y, z)$. The majority function is *self-dual* [7]. Note that an M_n operator filled with $\lfloor n/2 \rfloor$ $\mathtt{0/1}$ collapses into a AND/OR operator [7].

D. Vector Notation for Boolean Variables

For the sake of compactness, we denote a container (vector) of $n-m+1$ Boolean variables by x_m^n, where the notation starts from index m and ends at index n. When the actual length of the vector is not important, a simpler notation for x_m^n is boldface $\boldsymbol{x}$. The element at index i in vector x_m^n is denoted by x_i. The complementation of a vector x_m^n is denoted by $\neg x_m^n$ which means $\neg x_i \; \forall i \in [m, m+1, .., n-1, n]$. With this notation, the aforementioned self-dual property becomes $\neg f(x_m^n) = f(\neg x_m^n)$. For the sake of clarity, we give an example about the vector notation. Let (a, b, c, d, e) be 5 Boolean variables to be represented in vector notation. Here, the start/end indeces are $m = 1$ / $n = 5$, respectively, and the vector itself is x_1^5. The elements of x_1^5 are $x_1 = a$, $x_2 = b$, $x_3 = c$, $x_4 = d$ and $x_5 = e$.

E. Boolean Algebra

The standard binary Boolean algebra (originally axiomatized by Huntington [24]) is a non-empty set $(\mathbb{B}, \wedge, \vee, \neg, 0, 1)$ subject to *identity, commutativity, distributivity, associativity, and complement* axioms over $\wedge, \vee$ and $\neg$ [7], [26]. For the sake of completeness, we report these basic axioms in Eq. 1.

$$\boldsymbol{\Delta} \begin{cases} \textbf{Identity : } \boldsymbol{\Delta.I} \\ x \vee 0 = x \\ x \wedge 1 = x \\ \textbf{Commutativity : } \boldsymbol{\Delta.C} \\ x \wedge y = y \wedge x \\ x \vee y = y \vee x \\ \textbf{Distributivity : } \boldsymbol{\Delta.D} \\ x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z) \\ x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z) \\ \textbf{Associativity : } \boldsymbol{\Delta.A} \\ x \wedge (y \wedge z) = (x \wedge y) \wedge z \\ x \vee (y \vee z) = (x \vee y) \vee z \\ \textbf{Complement : } \boldsymbol{\Delta.Co} \\ x \vee \neg x = 1 \\ x \wedge \neg x = 0 \end{cases} \tag{1}$$

This axiomatization for Boolean algebra is sound and complete [25], [26]. Informally, it means that, logic arguments or formulas, proved by axioms in Δ are valid (soundness) and all true logic arguments are provable (completeness). More precisely, it means that, in the induced logic system, all theorems are tautologies (soundness) and all tautologies are theorems (completeness). We refer the reader to [25] for a more formal discussion on mathematical logic. In computer logic applications, only sound axiomatizations are of interest [26]. Complete and sound axiomatizations are desirable [26].

Other Boolean algebras exist, with different operators and axiomatizations, such as Robbins algebra, Freges algebra, Nicods algebra, MAJ-3/INV algebra, etc. [25]. In the immediate following, we give details on the MAJ-3/INV Boolean algebra.

F. MAJ-3/INV Boolean Algebra

The MAJ-3/INV Boolean algebra introduced in [1] is defined over the set $(\mathbb{B}, M_3, \neg, 0, 1)$, where M_3 is the ternary majority operator and $\neg$ is the unary complementation operator. The following set of five primitive transformation rules, referred to as Ω_3, is an *axiomatic system* for $(\mathbb{B}, M_3, \neg, 0, 1)$. All variables belong to $\mathbb{B}$.

$$\boldsymbol{\Omega_3} \begin{cases} \textbf{Commutativity : } \boldsymbol{\Omega_3.C} \\ M_3(x, y, z) = M_3(y, x, z) = M_3(z, y, x) \\ \textbf{Majority : } \boldsymbol{\Omega_3.M} \\ \begin{cases} \text{if}(x = y)\text{: } M_3(x, y, z) = x = y \\ \text{if}(x = \neg y)\text{: } M_3(x, y, z) = z \end{cases} \\ \textbf{Associativity : } \boldsymbol{\Omega_3.A} \\ M_3(x, u, M_3(y, u, z)) = M_3(z, u, M_3(y, u, x)) \\ \textbf{Distributivity : } \boldsymbol{\Omega_3.D} \\ M_3(x, y, M_3(u, v, z)) = \\ M_3(M_3(x, y, u), M_3(x, y, v), z) \\ \textbf{Inverter Propagation : } \boldsymbol{\Omega_3.I} \\ \neg M_3(x, y, z) = M_3(\neg x, \neg y, \neg z) \end{cases} \tag{2}$$

It has been shown that this axiomatization is sound and complete with respect to $(\mathbb{B}, M_3, \neg, 0, 1)$ [1]. The MAJ-3/INV Boolean algebra finds application in circuit optimization and has already showed some promising results [1].

Note that early attempts to majority logic have already been reported in the 60's [31]–[36] but they mostly focused on three input majority operators. Also, derived logic manipulation methods failed to gain momentum due to their inherent complexity.

While traditional Boolean algebras can be naturally extended from 2 to n variables, it is currently unclear how such a majority axiomatization extends to an arbitrary number of variables n (odd). In the following, we address this question by proposing a natural axiomatization of MAJ-n/INV logic.

III. Axiomatization of MAJ-n Logic

In this section, we present the generic axiomatization of MAJ-n logic. We first extend the set of five axioms presented in [1] to n-variables, with n being an odd integer. Then, we show their validity in the Boolean domain. Finally, we demonstrate their completeness by inclusion of other complete Boolean axiomatizations.

A. Generic MAJ-n/INV Axioms

The five axioms for MAJ-3/INV logic in [1] deal with *commutativity*, *majority*, *associativity*, *distributivity*, and *inverter propagation* laws. The following set of equations extends their domain to an arbitrary odd number n of variables. Note that all axioms, hold with $n \geq 3$.

$$
\boldsymbol{\Omega_n}
\begin{cases}
\textbf{Commutativity : } \boldsymbol{\Omega_n.C} \\
M_n(x_1^{i-1}, x_i, x_{i+1}^{j-1}, x_j, x_{j+1}^n) = \\
M_n(x_1^{i-1}, x_j, x_{i+1}^{j-1}, x_i, x_{j+1}^n) \\
\textbf{Majority : } \boldsymbol{\Omega_n.M} \\
\text{If}(\lceil \frac{n}{2} \rceil \text{ elements of } x_1^n \text{ are equal to } y): \\
\quad M_n(x_1^n) = y \\
\text{If}(x_i \neq x_j): \\
\quad M_n(x_1^n) = M_{n-2}(y_1^{n-2}) \\
\quad\ \text{where } y_1^{n-2} = x_1^n \text{ removing } \{x_i, x_j\} \\
\textbf{Associativity : } \boldsymbol{\Omega_n.A} \\
M_n(z_1^{n-2}, y, M_n(z_1^{n-2}, x, w)) = \\
M_n(z_1^{n-2}, x, M_n(z_1^{n-2}, y, w)) \\
\textbf{Distributivity : } \boldsymbol{\Omega_n.D} \\
M_n(x_1^{n-1}, M_n(y_1^n)) = \\
M_n(M_n(x_1^{n-1}, y_1), M_n(x_1^{n-1}, y_2), ..., \\
\quad M_n(x_1^{n-1}, y_{\lceil \frac{n}{2} \rceil}), y_{\lceil \frac{n}{2} \rceil + 1}, ..., y_n) = \\
M_n(M_n(x_1^{n-1}, y_1), M_n(x_1^{n-1}, y_2), ..., \\
\quad M_n(x_1^{n-1}, y_{\lceil \frac{n}{2} \rceil + 1}), y_{\lceil \frac{n}{2} \rceil + 2}, ..., y_n) = \\
M_n(M_n(x_1^{n-1}, y_1), M_n(x_1^{n-1}, y_2), ..., \\
\quad M_n(x_1^{n-1}, y_{n-1}), y_n) \\
\textbf{Inverter Propagation : } \boldsymbol{\Omega_n.I} \\
\neg M_n(x_1^n) = M_n(\neg x_1^n)
\end{cases}
\tag{3}
$$

Commutativity means that changing the order of the variables in M_n does not change the result. Majority defines a logic decision threshold (over $n \geq 3$ variables) and a hierarchical reduction of majority operators with complementary variables. Note that $M_3(x, y, \neg y) = x$ as boundary condition. Associativity says that swapping pairs of variables between cascaded M_n sharing $n-2$ variables does not change the result. In this context, it is important to recall that $n-2$ is an odd number if n is an odd number. Distributivity delimits the re-arrangement freedom of variables over cascaded M_n operators. Inverter propagation moves complementation freely from the outputs to the inputs of a M_n operator, and *viceversa*.

For the sake of clarity, we give an example for each axiom over a finite n-arity.

Commutativity with $n = 5$:
$M_5(a, b, c, d, e) = M_5(b, a, c, d, e) = M_5(a, b, c, e, d)$.

Majority with $n = 7$:
$M_7(a, b, c, d, e, g, g') = M_5(a, b, c, d, e)$.

Associativity with $n = 5$:
$M_5(a, b, c, d, M_5(a, b, c, g, h)) =$
$M_5(a, b, c, g, M_5(a, b, c, d, h))$.

Distributivity with $n = 7$:
$M_7(a, b, c, d, e, g, M_7(x, y, z, w, k, t, v)) =$
$M_7(M_7(a, b, c, d, e, g, x), M_7(a, b, c, d, e, g, y),$
$M_7(a, b, c, d, e, g, z), M_7(a, b, c, d, e, g, w), k, t, v)$.

Inverter propagation with $n = 9$:
$\neg M_9(a, b, c, d, e, g, h, x, y) =$
$M_9(\neg a, \neg b, \neg c, \neg d, \neg e, \neg g, \neg h, \neg x, \neg y)$.

B. Soundness

To demonstrate the validity of these laws, and thus the validity of the MAJ-n axiomatization, we need to show that each equation in Ω_n is sound with respect to the original domain, i.e., $(\mathbb{B}, M_n, \neg, 0, 1)$ [1]. The following theorem addresses this requirement.

Theorem 3.1: Each axiom in Ω_n is sound (valid) w.r.t. $(\mathbb{B}, M_n, \neg, 0, 1)$.

Proof:

Commutativity $\boldsymbol{\Omega_n.C}$ Since majority is defined on reaching a threshold $\lceil n/2 \rceil$ of true inputs then it is independent of the order of its inputs. This means that changing the order of operands in M_n does not change the output value. Thus, this axioms is valid in $(\mathbb{B}, M_n, \neg, 0, 1)$.

Majority $\boldsymbol{\Omega_n.M}$ Majority first defines the output behavior of M_n in the Boolean domain. Being a definition, it does not need particular proof for soundness. Consider then the second part of the majority axiom. The recursive inclusion of M_{n-2} derives from the mutual cancellation of complementary variables. In a binary majority voting system of n electors, two electors voting to opposite values annihilate themselves. The final decision is then just depending on the votes from the remaining $n-2$ electors. Therefore, this axiom is valid in $(\mathbb{B}, M_n, \neg, 0, 1)$.

Associativity $\boldsymbol{\Omega_n.A}$ We split this proof in three parts that cover the whole Boolean space. Thus, it is sufficient to prove the validity of the associativity axiom for each of these parts. **(1) the vector z_1^{n-2} contains at least one logic 1 and one logic 0.** In this case, it is possible to apply $\Omega_n.M$ and reduce M_n to M_{n-2}. If we remain in case (1), we can keep applying $\Omega_n.M$. At some point, we will end up in case (2) or (3). **(2) the vector z_1^{n-2} contains all logic 1.** For $n > 3$, the final voting decision is 1 for both equations, so the equality holds. In case $n = 3$ and the the vector z_1^{n-2} contains all logic 1, the majority operator collapses into a disjunction operator. For example, $M_3(1, a, M_3(1, c, d)) = \vee_2(a, \vee_2(c, d))$. Here, the validity of the associativity axiom follows then from traditional disjunction associativity. **(3) the vector z_1^{n-2} contains all logic 0.** For $n > 3$, the final voting decision is 0 for both equations, so the equality holds. In case $n = 3$ and the vector z_1^{n-2} contains all logic 0, the majority operator collapses into a conjunction operator. For example, $M_3(0, a, M_3(0, c, d)) = \wedge_2(a, \wedge_2(c, d))$. Here, the validity of the associativity axiom follows then from traditional conjunction associativity.

Distributivity $\boldsymbol{\Omega_n.D}$ We split this proof in three parts that cover the whole Boolean space. Thus, it is sufficient to prove the validity of the distributivity axiom for each of these parts. Note that the distributivity axiom deals with a majority operator M_n where one inner variable is actually another independent majority operator M_n. Distributivity rearranges the computation in M_n moving up the variables at the bottom level and down the variables at the top level. In this part of the proof we show that such rearrangement does not change the functionality of M_n, i.e., the final voting decision in $\Omega_n.D$. Recall that n is an odd integer greater than 1 so $n-1$ must be an even integer. **(1) half of x_1^{n-1} values are logic 0 and the remaining half are logic 1.** In this case, the final voting decision in axiom $\Omega_n.D$ only depends on y_1^n. Indeed,

[1]By M_n, it is intended any M_i with $i \leq n$. Indeed, any M_i operator with $i \leq n$ can be emulated by a fully-fed M_n operator with pairs of regular/complemented variables, e.g., $M_5(a, b, c, d, \neg d) = M_3(a, b, c)$.

all elements in x_1^{n-1} annihilate due to axiom $\Omega_n.M$. In the two identities of $\Omega_n.D$, we see that when x_1^{n-1} annihilate the equations simplify to $M_n(y_1^n)$, according to the predicted behavior. **(2) at least $\lceil n/2 \rceil$ of x_1^{n-1} values are logic 0.** Owing to $\Omega_n.M$, the final voting decision in this case is logic 0. This is because more than half of the variables are logic 0 matching the prefixed voting threshold. In the two identities of $\Omega_n.D$, we see that more than half of the inner M_n evaluate to logic 0 by direct application of $\Omega_n.M$. In the subsequent phase, also the outer M_n evaluates to logic 0, as more than half of the variables are logic 0, according to the predicted behavior. **(3) at least $\lceil n/2 \rceil$ of x_1^{n-1} values are logic 1.** This case is symmetric to the previous one.

Inverter Propagation $\Omega_n.I$ Inverter propagation moves complementation from output to inputs, and *viceversa*. This axiom is a special case of the self-duality property previously presented. It holds for all majority operators in $(\mathbb{B}, M_n, \neg, 0, 1)$.

■

The soundness of Ω_n in $(\mathbb{B}, M_n, \neg, 0, 1)$ guarantees that repeatedly applying Ω_n axioms to a Boolean formula we do not corrupt its original functionality. This property is of interest in logic manipulation systems where functional correctness is an absolute requirement.

C. Completeness

While soundness speaks of the correctness of a logic systems, completeness speaks of its manipulation capabilities. For an axiomatization to be complete, all possible manipulations of a Boolean formula must be attainable by a sequence, possibly long, of primitive axioms.

We study the completeness of Ω_n axiomatization by comparison to other complete axiomatizations of Boolean logic. The following theorem shows our main result.

Theorem 3.2: The set of five axioms in Ω_n is complete w.r.t. $(\mathbb{B}, M_n, \neg, 0, 1)$.

Proof: We first consider Ω_3 and we show that it is complete w.r.t. $(\mathbb{B}, M_3, \neg, 0, 1)$. We need to prove that every valid argument, i.e., $(\mathbb{B}, M_3, \neg, 0, 1)$-formula, has a proof in the system Ω_3. By contradiction, suppose that a true $(\mathbb{B}, M_3, \neg, 0, 1)$-formula, say α, cannot be proven true using Ω_3 rules. Such $(\mathbb{B}, M_3, \neg, 0, 1)$-formula α can always be reduced into a $(\mathbb{B}, \wedge, \vee, \neg, 0, 1)$-formula. Indeed, recall that $M(x, y, z) = (x \vee y) \wedge (x \vee z) \wedge (y \vee z)$. Using Δ, all $(\mathbb{B}, \wedge, \vee, \neg, 0, 1)$-formulas can be proven, including α. However, every $(\mathbb{B}, \wedge, \vee, \neg, 0, 1)$-formula is also contained by $(\mathbb{B}, M_3, \neg, 0, 1)$, where $\wedge$ and $\vee$ are emulated by majority operators. Moreover, rules in Ω_3 with one input fixed to 0 and 1 behaves as Δ rules (Eq. 1). For example, $\Omega_3.A$ with variable u fixed to logic 1 (0) behaves as $\Delta.A$ for disjunction (conjunction). The other axioms follow analogously. This means that also Ω_3 is capable to prove the reduced $(\mathbb{B}, M, \neg, 0, 1)$-formula α, contradicting our assumption. Thus Ω_3 is complete w.r.t. $(\mathbb{B}, M_3, \neg, 0, 1)$.

We consider now Ω_n. First note that $(\mathbb{B}, M_n, \neg, 0, 1)$ naturally includes $(\mathbb{B}, M_3, \neg, 0, 1)$. Similarly, Ω_n axioms inherently extend the ones in Ω_3. Thus, the completeness property is inherited provided that Ω_n axioms are sound. However, Ω_n soundness is already proven in Theorem 3.1. Thus, Ω_n axiomatization is also complete. ■

Being sound and complete, the axiomatization Ω_n defines a consistent framework to operate on Boolean logic via n-ary majority operators and inverters. In the following section, we discuss some promising applications in computer science of such majority logic system.

IV. Discussion

In this section, we discuss relevant application of Ω_n axiomatization. We first present the potential of logic optimization performed via MAJ-n operators and inverters. Then, we show how Boolean satisfiability can be described in terms of majority operators and solved using Ω_n. Successively, we demonstrate the manipulation of repetition codes via Ω_n under a majority logic decoding scheme. Finally, we discuss the application of majority logic to several emerging technologies, such as quantum-dot cellular automata, spin-wave devices, threshold logic and others.

A. Logic Optimization

Logic optimization is the process of manipulating a logic data structure, such as a logic circuit, in order to minimize some target metric [27]. Usual optimization targets are size (number of nodes/elements), depth (maximum number of levels) and interconnections (number of edges/nets). More elaborated targets use a combination of size/depth/interconnections metrics, such as nodes×interconnections and others.

Theoretical results from computer science show that majority logic circuits are much more compact than traditional ones based on conjunction and disjunction operators [6]. For example, majority logic circuits of depth 2 and 3 possess the expressive power to represent arithmetic functions, such as powering, multiplication, division, addition etc., in polynomial size [6]. On the other hand, the traditional AND/OR-based counterparts are exponentially sized [6].

Given the existence of very compact majority logic circuits, we need an efficient set of manipulation laws to reach those circuits automatically. In this context, the axiomatic system previously introduced is the natural set of tools addressing this need. For example, consider a logic circuit (or Boolean function) $f = M_5(M_3(a, b, c), M_3(a, b, d), M_3(a, b, e), M_3(a, b, g), h)$. In circuit optimization, a common problem is to minimize the number of elements while keeping short some input-output paths. Suppose we want to minimize the number of majority operators while keeping the path h to f as short as possible, i.e., one majority operator. The original circuit cost is 5 majority operators. To manipulate this formula, we first equalize the n-arity of the majority operators using axiom $\Omega_n.M$, i.e., by adding a fake annihilated variable x, as:

$f = M_5(M_5(a, b, c, x, \neg x), M_5(a, b, d, x, \neg x),$
$M_5(a, b, e, x, \neg x), M_5(a, b, g, x, \neg x), h)$

At this point, we can apply $\Omega_n.D$ and save one majority operator as:

$f = M_5(M_5(a, b, c, x, \neg x), M_5(a, b, d, x, \neg x),$
$M_5(a, b, e, x, \neg x), g, h)$.

Finally, we can reduce the majority n-arity to its minimum via $\Omega_n.M$ as:

$f = M_5(M_3(a,b,c), M_3(a,b,d), M_3(a,b,e), g, h)$.

The resulting circuit cost is 4 majority operators.

1) Optimization Script: As emerged from the previous optimization example, an intuitive heuristic to optimize majority logic circuits consists of majority inflation rules (from Ω_n) followed by majority reduction rules (from Ω_n). Alg. 1 depicts a simple optimization script and a brief description follows. First, the n-arity of all majority operators in the logic circuit is temporarily increased by using $\Omega_n.M$ rule from right to left, for example $M_3(a,b,c) = M_5(a,b,c,\neg c,c)$. This operation unlocks new simplification opportunities. Then, redundant majority operators are identified and deleted through $\Omega_n.A, \Omega_n.D, \Omega_n.M$ rules. Finally, the n-arity of all majority operators in the logic circuit is decreased to the minimum via $\Omega_n.M$ rule from left to right.

Algorithm 1 Majority Logic Optimization Heuristic

INPUT: Majority Logic Network.
OUTPUT: Optimized Majority Logic Network.

```
Majority Operator Increase n-arity(Ω_n.M);
// increase n-arity of the majority operator
Majority Operator Simplifcation(Ω_n.A, Ω_n.D, Ω_n.M);
// deleting redundant majority operators
Majority Operator Reduce n-arity(Ω_n.M);
// decrease n-arity of the majority operator
```

This approach naturally targets depth and size reductions in the majority logic network. However, it can be extended to target more elaborated metrics, such as $\sum_{i=1}^{M} fanin(node_i)$ or $M \times N_{inv}$, where M is the total number of nodes and N_{inv} is the number of inverters. The best metric depends on the considered technology for final implementation.

2) Full-Adder Case Study: In order to prove the efficacy of the majority optimization heuristic in Alg. 1, we consider as case study the full-adder logic circuit. The full-adder logic circuit is fundamental to most arithmetic circuits. Consequently, the effective optimization of full-adders is of paramount importance.

A full-adder represents a three-input and two-output Boolean function:

$sum = a \oplus b \oplus c_{in}$

$c_{out} = M_3(a,b,c)$

Using just majority operators with n-arity equal to three, the best full-adder implementation counts 3 majority nodes, inverters apart, as depicted by Fig. 1. However, a more compact majority logic network is possible by exploiting higher n-arity degrees and manipulating such majority logic circuit via Ω_n. In particular, the critical operation is sum because c_{out} is naturally represented by a single M_3 operator. So, for sum our optimization heuristic first expands the top majority operator from an n-arity of three

$sum = M_3(a, \neg M_3(a,b,c_{in}), M_3(\neg a,b,c_{in}))$

to an n-arity of 5 as

$sum = M_5(a, \neg M_3(a,b,c_{in}), \neg M_3(a,b,c_{in}), M_3(a,b,c_{in}), M_3(\neg a,b,c_{in}))$.

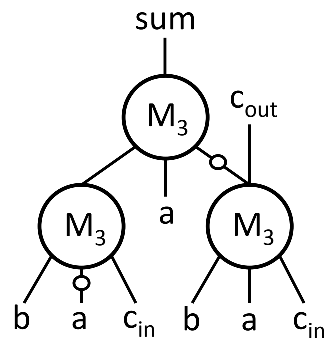

Fig. 1. Majority logic circuit for the full-adder with operator n-arity equal to 3. Complementation is represented by bubbles on the edges.

After that, derived simplification rules from Ω_n, called relevance rules in [1], reduce the number of majority operators to 2 as

$sum = M_5(a, \neg M_3(a,b,c_{in}), \neg M_3(a,b,c_{in}), b, c_{in})$.

In its graph representation, depicted by Fig. 2, this representation of *sum* just consists of two majority operators as the internal $M_3(a,b,c_{in})$, is shared. Moreover, $M_3(a,b,c_{in})$ is also generating the c_{out} function which can be further shared. This means that the optimized logic circuit in Fig. 2, counting just two majority operators, is a minimal implementation for the full-adder in terms of majority logic. To provide a reference, an optimized AND-inverter graph representation for the full-adder is depicted by Fig. 3. It counts 8 nodes and has been optimized using the state-of-the-art academic ABC optimizer [39] which manipulates AND-inverter graphs. We can see that the majority logic circuit produced by our optimization heuristic is much more compact thanks to the majority logic expressiveness and to the properties of our axiomatic system, Ω_n.

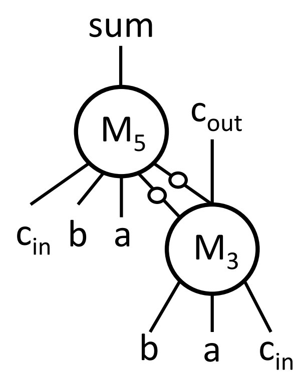

Fig. 2. Majority logic circuit for the full-adder with unbounded operator n-arity. Complementation is represented by bubbles on the edges.

The minimality of the majority logic circuit in Fig. 2 is formally proved in the following theorem.

Theorem 4.1: The majority logic circuit in Fig. 2 for the full-adder has the minimum number of majority operators.

Proof: The full-adder consists of two distinct functions. Being distinct, they require at least two separate majority operators fed with different signals. The majority logic circuit

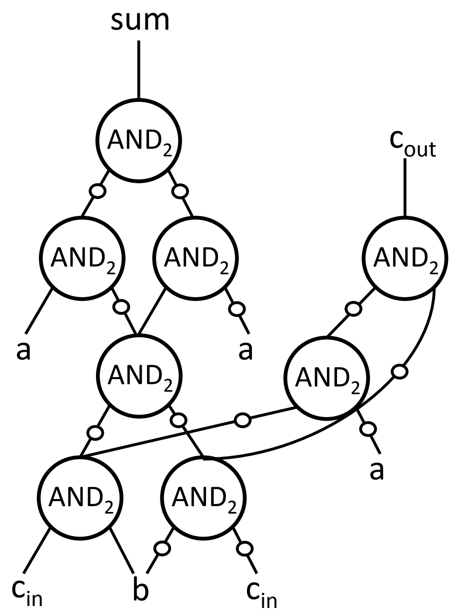

Fig. 3. AND-inverter logic circuit for the full-adder optimized via ABC academic tool. Complementation is represented by bubbles on the edges.

in Fig. 2 actually consists of two majority operators thus being minimal. ■

On top of having the minimum number of operators, the majority network in Fig. 2 has lower $\sum_{i=1}^{M} fanin(node_i)$ metric (equal to 8) as compared to the majority network in Fig. 1 (equal to 9). The number of inverters is 2 in both cases.

We see that the axiomatic system Ω_n can be used to optimize majority logic circuits and produces excellent results. As the Ω_n rules are simple enough to be programmed on a computer, MAJ-n logic optimization can be automated and applied to large systems.

B. Boolean Satisfiability

Boolean *satisfiability* (SAT) is the first known NP-complete problem [28]. Traditionally, SAT is formulated in *Conjunctive Normal Form* (CNF) [29]. Recently, majority logic has been considered as an alternative to CNF to speed-up SAT [4]. In [4], a *Majority Normal Form* (MNF) has been introduced, which is a majority of majorities, where majorities are fed with literals, `0` or `1`. The MNF-SAT problem is NP-complete in its most general definition [4]. However, there are interesting restrictions of MNF whose satisfiability can instead be decided in polynomial time. For example, when there are no mixed logic constants appearing in the MNF, the MNF-SAT problem can be solved in polynomial time. This result is valid not just for MNF but for majority logic circuits in general [4].

In order to solve the general problem of majority logic satisfiability, and thus of MNF-SAT, a set of manipulation rules is needed. Indeed, the core of most modern SAT solving tools make extensive use of Boolean logic axioms. When dealing with majority logic, our proposed axiomatic system Ω_n is the natural tool to operate on MNF forms, or alike, and prove their satisfiability.

For the sake of clarity, we give an example of majority SAT solving via Ω_n laws. We consider not just an MNF, which is a two level logic representation form, but a general formula in $(\mathbb{B}, M_n, \neg, 0, 1)$. Our example is the *unSAT* function $f = M_5(M_3(a,b,c), M_5(M_5(a,b,c,0,0), \neg b, c, 0, 0), \neg a, \neg b, 0)$. In oder to check the satisfiability of f, a majority SAT solver first tries to enforce at least 3 over 5 logic `1` in the top M_5 [4]. Otherwise, a conflict in the input assignment appears. If all possible input assignments lead to a conflict the function is declared unsatisfiable [4].

Let us first focus on the element $M_5(M_5(a,b,c,0,0), \neg b, c, 0, 0)$. Here, even before looking for possible assignments, our axiom $\Omega_n.A$ re-arranges the variables as $M_5(M_5(\neg b, b, c, 0, 0), a, c, 0, 0)$. In this formula, our axiom $\Omega_n.M$ directly annihilates b and $\neg b$ leading to $M_5(M_3(c,0,0), a, c, 0, 0)$. Furthermore, $\Omega_n.M$ still applies twice corresponding to $M_5(0, a, c, 0, 0)$ and then 0. We can substitute this to the original formula as $f = M_5(M_3(a,b,c), 0, \neg a, \neg b, 0)$ which symplifies the SAT problem. Now, we need both $\neg a$ and $\neg b$ to be `1` in order to do avoid an immediate conflict. This means $a = 0$ and $b = 0$. However, this assigment evaluates always to `0` the term $M_3(a,b,c)$ generating a conflict for all input patterns. Thus, the original formula is declared unsatisfiable.

As we can see, our majority logic axiomatic system Ω_n is the ground for proving the satisfiability of formula in $(\mathbb{B}, M_n, \neg, 0, 1)$. Without Ω_n, SAT tools would need to decompose all majority operators in AND/ORs because with conjunctions and disjunctions the classic set of Boolean manipulation rules apply. However, such decomposition would nullify the competitive advantage enabled by the majority logic expressiveness. In this scenario, our Ω_n rules fill the gap for manipulating majority operators natively.

C. Decoding of Repetition Codes

Repetition codes are basic error-correcting codes. The main rationale in using repetition codes is to transmit a message several times over a noisy channel hoping that the channel corrupts only a minority of the bits [30]. In this scenario, decoding the received message via majority logic is the natural way to correct transmission errors.

Consider safety-critical communication systems. It is common to have hierarchical levels of coding to decrease the chance of error and thus resulting in system malfunction. When applied on several levels, majority logic decoding is nothing but a majority logic circuit. The maximum number of cascaded majority operators determines the decoding performance. We want to maximize the decoding performance while keeping the error probability low. In this scenario, we can use our axiomatic system Ω_n to explore different trade-offs in depth/size manipulation of the corresponding majority decoding scheme.

For the sake of clarity, we give an example of the optimization for majority logic decoding via Ω_n. Consider a safety-critical communication system sending the same binary message a over 5 different channels C_1, C_2, C_3, C_4 and C_5. Each channel is affected by different levels of noise requiring just 1 repetition for C_1, C_2, C_3, and C_4 but 5 repetitions for

C_5. Suppose also the communication over channel 5 is much slower than in the other channels. The final decoded message is the majority of the each decoded message per channel. If we name x_i the decoded message a for i-th channel and y the final decoded message, the system can be represented in majority logic as $y = M_5(x_1, x_2, x_3, x_4, x_5)$. Note that for x_1, x_2, x_3, x_4 the decoded message is actually identical to the received message because only 1 repetition is sent over the channels. The element x_5 is the only one needing further majority decoding, namely $x_5 = M_5(z_1, z_2, z_3, z_4, z_5)$ where z_i are the received a messages over channel C_5. The final system is then expressable as $y = M_5(x_1, x_2, x_3, x_4, M_5(z_1, z_2, z_3, z_4, z_5))$. To decode the final message y, the critical element for perfomance is $M_5(z_1, z_2, z_3, z_4, z_5)$, with z_5 being the latest arriving message to be processed. In this context, we can use $\Omega_n.D$ axiom to redistribute the decoding operations and obtain an improvement in performance, which is not a trivial process. The idea is to push to the top majority level z_i variables, with the highest possible i index. For this purpose, axioms $\Omega_n.D$ transforms $y = M_5(x_1, x_2, x_3, x_4, M_5(z_1, z_2, z_3, z_4, z_5))$ into $y = M_5(M_5(x_1, x_2, x_3, x_4, z_1), M_5(x_1, x_2, x_3, x_4, z_2),$ $M_5(x_1, x_2, x_3, x_4, z_3), z_4, z_5)$. In this latter model of majority decoding, most of the computation is performed in advance before the late messages z_4 and z_5 arrive. This means that, when the late z_5 arrives, there is need for just one level of majority computation and not two as in the initial model.

D. Emerging Technologies

Majority gates with more than 3 inputs have been simulated and implemented for a variety of non-CMOS technologies. A further generalization of majority gates is threshold logic gate [6], which performs weighted sum of multiple inputs and once the sum is more than a pre-determined threshold, the output is true. As such, a threshold logic gate can be configured to function as a majority logic gate. In the following, we describe a few published works that describes majority or threshold gates with more than 3 inputs.

Majority logic gates were experimentally demonstrated with *Quantum-dot Cellular Automata* (QCA) in [12] and [13]. For facilitating QCA circuit design, a tool named QCADesigner is developed [15]. Simulation of M_5 gate using QCADesigner is presented in several papers, including [14]. Fig. 4 depicts two possible QCA implementations for a M_5 gate. Applications of large majority gates towards efficient adder construction were also discussed. For example, a M_7 has also been proposed. Fig. 5 depicts a possible QCA implementation for a M_5 gate.

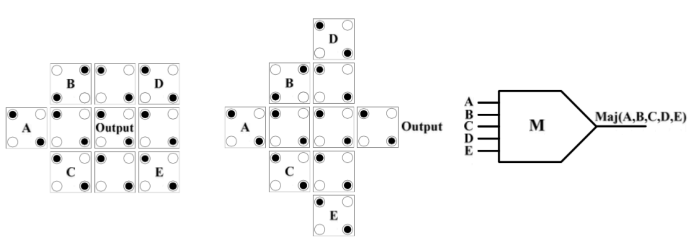

Fig. 4. Two different implementations of a M_5 gate in QCA technology [14].

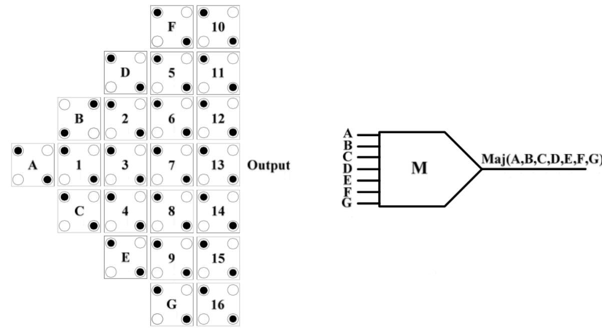

Fig. 5. Physical implementation of a M_7 gate in QCA technology [14].

Note that a M_5 gate, a M_3 gate and an inverter gate are sufficient to build a full-adder, as highlighted by the theoretical case study in Section IV-A. In this scenario, the proposed Ω_n axiomatic system is key to unveil such efficient circuit implementations in QCA nanotechnology, where majority gates are the logic primitives for computation.

Very recently, a majority logic circuit based on domain-wall nanowires has been proposed in [17]. The circuit is used for computing binary additions efficiently and can be shown to scale for majority gates with arbitrary number of inputs.

All-spin logic gates are originally proposed in [11]. Majority logic gates using all-spin logic is proposed in [10]. There, layout of M_3 gate using all-spin logic is shown and it is noted that majority gates with larger number of inputs can also be implemented. Indeed, a high fan-in majority gate is realizable by a simple superposition of spin-waves with same amplitude but different phases [20]. Fig. 6 depicts a sketch of a high fan-in majority gate in spin-wave technology.

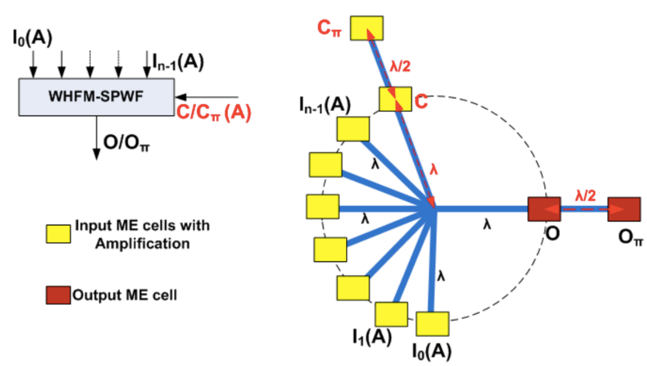

Fig. 6. Block diagram and schematic representation of a high fan-in majority gate in spin-wave technology [20].

In [9], a *Spin-Memeristor Threshold Logic* (SMTL) gate using memristive crossbar array is proposed. There, an array of SMTL gates is designed and simulated with experimentally validated device model characteristics. By varying the threshold input count, different possible mappings are demonstrated with good performance improvement over CMOS FPGA structures.

A programmable CMOS/memristor threshold logic is proposed in [16]. A 4-input threshold logic gate is experimentally demonstrated using Ag/a-Si/Pt memristive devices. They also propose a threshold logic network similar to [9] with programmable fan-in.

It is to be noted that none of the aforementioned implementations employed any automated synthesis flow to exploit majority gates with larger than 3 inputs. Thus, the potential of compact realization of diverse applications, even if feasible with these technologies, is hardly experimented due to the lack of an efficient synthesis flow. Our proposed sound and complete axiomatization aims at filling this gap.

Note that the aforementioned examples are just few of the possible applications of n-ary majority logic and of its sound and complete axiomatization. More opportunities exist in other fields of computer science but their discussion is out of the scope of this paper.

V. Conclusions

In this paper, we proposed a sound and complete axiomatization of majority logic. Stemming from previous work on MAJ-3/INV logic, we extended fundamental axioms to arbitrary n-ary majority operators. Based on this general set of axioms, computer applications can now fully exploit the expressive power of majority logic. We discussed the potential impact in the fields of logic optimization, Boolean satisfiability, repetition codes and emerging technologies. From a general standpoint, the possibility of manipulating logic in terms of majority operators paves the way for more efficient computer applications where the core reasoning tasks are performed in the Boolean domain. In particular, possible directions for future work include the development of (i) a complete majority satisfiability solver and (ii) a majority synthesis tool targeting nanotechnologies.

Acknowledgements

The authors would like to thank Prof. Maciej Ciesielski for valuable discussions. This research was supported by ERC-2009-AdG-246810.

References

[1] L. Amarú, P.-E. Gaillardon, G. De Micheli, *Majority-Inverter Graph: A Novel Data-Structure and Algorithms for Efficient Logic Optimization*, Proc. DAC'14.

[2] L. Amarú, P.-E. Gaillardon, G. De Micheli, *Boolean Logic Optimization in Majority-Inverter Graphs*, Proc. DAC'15.

[3] L. Amarú, P.-E. Gaillardon, G. De Micheli, *Majority-Inverter Graph: A New Paradigm for Logic Optimization*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015.

[4] L. Amarú, P.-E. Gaillardon, G. De Micheli, *Majority Logic Representation and Satisfiability*, Proc. IWLS'14.

[5] E. Mossel, R. O'Donnell, K. Oleszkiewicz, *Noise stability of functions with low influences: invariance and optimality*, IEEE Symposium on Foundations of Computer Science, 2005.

[6] M. Krause, P. Pudlak, *On the computational power of depth-2 circuits with threshold and modulo gates*, Theor. Comput. Sci., 174, pp. 137-156, 1997.

[7] T. Sasao, *Switching Theory for Logic Synthesis*, Springer, 1999.

[8] P. Wohl, J.A. Waicukauski, *ATPG and compression by using majority gates*, http://www.google.com/patents/US8549372, 2013, October, Google Patents, US Patent 8,549,372.

[9] D. Fan, M. Sharad, K. Roy, *Design and synthesis of ultralow energy spin-memristor threshold logic* IEEE Transactions on Nanotechnology,, 13(3), 574-583, 2014.

[10] C. Augustine, et al. *Low-power functionality enhanced computation architecture using spin-based devices*, IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), 2011.

[11] B. Behin-Aein, et al. *Proposal for an all-spin logic device with built-in memory*, Nature nanotechnology 5.4 (2010): 266-270.

[12] A. Imre, et al. *Majority logic gate for magnetic quantum-dot cellular automata*, Science 311.5758 (2006): 205-208.

[13] G.L. Snider, et al. *Quantum-dot cellular automata: Line and majority logic gate*, Japanese Journal of Applied Physics 38.12S (1999): 7227.

[14] R. Arman, et al. *A symmetric quantum-dot cellular automata design for 5-input majority gate*, Journal of Computational Electronics 13.3 (2014): 701-708.

[15] K. Walus, et al. *QCADesigner: A rapid design and simulation tool for quantum-dot cellular automata*, IEEE Transactions on Nanotechnology, 3.1 (2004): 26-31.

[16] L. Gao, et al. *Programmable CMOS/memristor threshold logic*, IEEE Transactions on Nanotechnology, 12.2 (2013): 115-119.

[17] Y. Hao, et al. *Energy efficient in-memory machine learning for data intensive image-processing by non-volatile domain-wall memory*, IEEE Asia and South Pacific Design Automation Conference (ASP-DAC), 2014.

[18] W. Li, Y. Yang, H. Yan and Y. Liu, "Three-Input Majority Logic Gate and Multiple Input Logic Circuit Based on DNA Strand Displacement," in *Nano Letters*, vol. 13, no. 6, pp. 2980–2988, May 2013, doi: 10.1021/nl4016107.

[19] G. Yang, W. N.N. Hung, X. Song and M. Perkowski, "Majority-based reversible logic gates," in *Elsevier Theoretical Computer Science*, vol. 334, no. 1–3, pp. 259–274, April 2005, doi:10.1016/j.tcs.2004.12.026.

[20] P. Shabadi, *"Towards Logic Functions as the Device using Spin Wave Functions Nanofabric"*, Masters Theses 1896 - February 2014. Paper 850.

[21] S. Srivastava, S. Bhanja. "Hierarchical probabilistic macromodeling for QCA circuits." IEEE Transactions on Computers 56.2 (2007): 174-190.

[22] H. Cho, E. E. Swartzlander, "Adder and multiplier design in quantum-dot cellular automata." IEEE Transactions on Computers 58.6 (2009): 721-727.

[23] R. Zhang, P. Gupta, N. K. Zhong, Jha, *Threshold network synthesis and optimization and its application to nanotechnologies*. IEEE Transactions on Computer-Aided De-sign of Integrated Circuits and Systems 24, 107-118 (2005).

[24] E. V. Huntington, *Sets of Independent Postulates for the Algebra of Logic*, Transactions of the American Mathematical Society, 5:3 (1904), 288-309.

[25] B. Jonsson, Bjarni, *Boolean algebras with operators. Part I.*, American journal of mathematics (1951): 891-939.

[26] F. M. Brown, *Boolean reasoning: the logic of Boolean equations*, Courier Corporation, 2003.

[27] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, New York, 1994.

[28] M. R. Garey, D. S. Johnson, *Computers and Intractability– A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[29] A. Biere, M. Heule, H. van Maaren, *Handbook of satisfiability* Vol. 185. ios press, 2009.

[30] J. L. Massey, *Threshold Decoding*, M.I.T. Press, 1963.

[31] S. B. Akers, Jr., "On the Algebraic Manipulation of Majority Logic," in *IRE Transactions on Electronic Computers*, vol. EC-10, no. 4, pp. 779, 1961, doi=10.1109/TEC.1961.5219289.

[32] M. Cohn and R. Lindaman, "Axiomatic Majority-Decision Logic," in *IRE Transactions on Electronic Computers*, vol. EC-10, no. 1, pp. 17–21, March 1961, doi: 10.1109/TEC.1961.5219147.

[33] R. Lindaman, "A Theorem for Deriving Majority-Logic Networks Within an Augmented Boolean Algebra," in *IRE Transactions on Electronic Computers*, vol. EC-9, no. 3, pp. 338–342, Sept. 1960, doi: 10.1109/TEC.1960.5219856.

[34] H.S. Miller, R. O. Winder. *Majority-logic synthesis by geometric methods* IRE Transactions on Electronic Computers, (1962): 89-90.

[35] Y. Tohma, *Decompositions of Logical Functions Using Majority Decision Elements*, IEEE Trans. on Electronic Computers, pp. 698-705, 1964.

[36] F. Miyata, *Realization of arbitrary logical functions using majority elements*, IEEE Transactions on Electronic Computers, (1963): 183-191.

[37] L. G. Valiant, "Short monotone formulae for the majority function," in *Journal of Algorithms*, vol. 5, no. 3, pp. 363–366, September 1984, doi: 10.1016/0196-6774(84)90016-6.

[38] I. Wegener, "The Complexity of Boolean functions," in *Wiley-Teubner Series in Computer Science*, ISBN: 3-519-02107-2, 1987.

[39] ABC synthesis tool - available online at http://www.eecs.berkeley.edu/∼alanmi/abc/.