# Mapping Monotone Boolean Functions into Majority

Eleonora Testa, *Student Member, IEEE,* Mathias Soeken, *Member, IEEE,* Luca G. Amarù, *Member, IEEE,* Winston Haaswijk, *Student Member, IEEE,* and Giovanni De Micheli, *Fellow, IEEE*

***Abstract*—We consider the problem of decomposing monotone Boolean functions into majority-of-three operations, with a particular focus on decomposing the majority-$n$ function. When targeting monotone Boolean functions, Shannon's expansion can be expressed by a single majority-of-three operation. We exploit this property to transform binary decision diagrams (BDDs) for monotone functions into majority-inverter graphs (MIGs), using a simple one-to-one mapping. This process highlights desirable properties for further majority graph optimization, e.g., symmetries between the inputs of primitive operations, which are not apparent from BDDs. Although our construction yields a quadratic upper bound on the number of majority-3 operations required to realize majority-$n$, for small $n$ the concrete values are much smaller compared to those obtained from previous constructions which have linear and quasi-linear asymptotic upper bounds. Further, we demonstrate that minimum size MIGs, for the monotone functions majority-5 and majority-7, can be obtained applying a small number of algebraic transformations to the BDD.**

***Index Terms*—Binary Decision Diagrams, Majority Logic, Majority-Inverter Graphs, Function Decomposition**

## I. Introduction

THE majority-of-three function $\langle xyz \rangle$, which is true if and only if at least two of its inputs are true, plays an important role in the digital design of circuits using emerging nanotechnologies. Technologies such as *Spin-Wave Devices* (SWD, [1]), *Quantum-dot Cellular Automata* (QCA, [2]), and *Spin Torque Majority Gates* (STMG, [3]), are inherently majority-based and provide the capability of implementing inexpensive and compact majority-of-three gates.

Several theoretical works from $TC^0$ circuit complexity make use of majority gates with unbounded fan-in that realize the majority-$n$ function, where $n$ is odd. $TC^0$ is the circuit complexity class that contains Boolean circuits with constant depth and polynomial size, built using only unbounded-fanin majority gates and inverters [4]. These works provide results for interesting classes of functions, such as arithmetic operations (e.g., [5], [6]). They also serve as a good starting point for circuit realizations when using majority-based nanotechnologies, provided that one is equipped with a technique to express majority-$n$ functions, i.e., majority operations with $n$ inputs, in terms of majority-of-three.

The problem of expressing majority-$n$ using majority-of-three has already been studied in the 1960s. In [7], Amarel et al. investigated "*how best can the 5-argument majority function be realized with a network of 3-input majority gates?*". The focus was on finding the minimum number of 3-input majorities to build majority-5, but larger $n$ were also considered [7]. In the remainder, we will refer to the minimum number of majority-of-three to build a majority-$n$ as $M(n)$. It is surprising that, as of today, $M(n)$ is known only for 5- and 7-input majorities, while the minimum realization of majority-9 (and larger $n$) in terms of majority-3 is still under investigation.

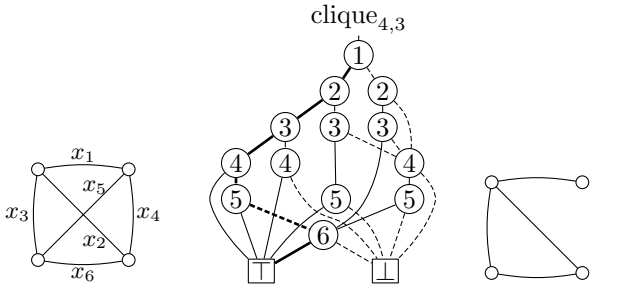
Other works have focused on $M(n)$, but they have only considered its asymptotic bounds [8]. The asymptotic complexity for $M(n)$ is linear, since one can reduce it to median selection [9]. However, when applying the construction to small values for $n$, the resulting majority graphs are still very large. For example, the majority-7 function can be constructed using 42 majority-3 operations according to median selection construction, while it is known that $M(7) = 7$. Sorter networks provide an alternative construction that provides a quasi-linear bound [10]. However, for small $n$ the construction can yield better results compared to median selection. To follow up with the previous example, the majority-7 function can be constructed using 32 majority-3 operations based on the sorter network construction.

In this paper, we focus on finding the minimum number of majority-3 operations to express majority-$n$. In particular, we propose an alternative construction based on *Binary Decision Diagrams* (BDDs, [11]) and we show that for monotone Boolean functions the Shannon decomposition, which is used in the construction of BDDs, can be expressed using the majority function. This leads to a one-to-one translation of BDDs into majority graphs, which are logic networks in which all operations are majority-3 functions. Since majority-$n$ is monotone, we can use the construction as an upper bound on the number of majority-3 operations. This bound is asymptotically quadratic, however, for small $n$ it leads to much smaller values compared to the constructions based on median selection and sorter networks. For instance, for majority-7 the construction leads to a realization with 15 majority-3 operations. Therefore, in order to find small realizations for majority-$n$ networks, for small $n$, the proposed BDD based approach can be a more effective starting point.

We show that we can derive the known optimum results for majority-5 and majority-7 starting from the proposed BDD construction by applying well-known algebraic properties of the majority function and two new identities. We apply the same procedure to majority-9, and we show that significant optimization can be obtained leading to a new best-known solution with 15 majority-3 operations. Majority-7 is the largest

Eleonora Testa, Mathias Soeken, Winston Haaswijk and Giovanni De Micheli are with the Integrated Systems Laboratory – EPFL, 1015 Lausanne, Switzerland (e-mail: eleonora.testa@epfl.ch; mathias.soeken@epfl.ch, winston.haaswijk@epfl.ch, giovanni.demicheli@epfl.ch).

Luca G. Amarù is with Synopsys Inc, 94043 Mountain View CA, United States (e-mail: luca.amaru@synopsys.com).




(a) $K_4$ graph (b) BDD (c) Graph instance with 3-clique

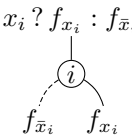
Fig. 1: Diagrammatic notation of a binary decision diagram for the function $\mathrm{clique}_{4,3}$. The '$\bot$' and '$\top$' represent the constant functions 0 and 1 [13]

majority-$n$ function for which a minimum size solution is known. The exact solution was found by using exhaustive search using a SAT-solver (see, e.g., [12]). Here, we explain in detail the entire derivation; this can provide insight into the decomposition of larger majority functions and may help to find values for $M(n)$ where $n \geq 9$.

## II. Preliminaries

### A. Binary decision diagrams

*Binary Decision Diagrams* (BDDs, [11]) are connected directed acyclic graphs, in which each node represents a Boolean function. Two terminal nodes labeled '$\bot$' and '$\top$' represent the constant functions 0 and 1, and all nonterminal nodes are labeled '$i$' for some $1 \leq i \leq n$ and connect their two successor nodes $f_{x_i}$ and $f_{\bar{x}_i}$




(1)

using Shannon's decomposition:

$$f = x_i \,?\, f_{x_i} : f_{\bar{x}_i} = x_i f_{x_i} \oplus \bar{x}_i f_{\bar{x}_i} \tag{2}$$

Here, the cofactors $f_{x_i}$ and $f_{\bar{x}_i}$ are the functions that are obtained by replacing $x_i$ with 1 and 0 in $f$, respectively. Each BDD has one node without parents representing the function, called the root. A BDD is called *ordered* if on each path from the root vertex to a terminal node, all variables, i.e., node labels, appear at most once and in the same order; variables may be skipped. A BDD is called *reduced*, if no two nodes represent the same function and if no node's successors are the same function. It is well known, that a reduced and ordered BDD is canonical—it has a unique representation—with respect to a given variable ordering [11]. In the following, we use the term BDD to refer to an ordered and reduced BDD.

We will use the Boolean function $\mathrm{clique}_{n,k}$ (see for example [14]) as a running example throughout the paper. The function has $\binom{n}{k}$ variables, each of which representing an edge in the complete graph $K_n$ over $n$ variables. For example, the 6 variables representing edges in $K_4$ are shown in Fig. 1(a). A variable assignment corresponds to a particular graph instance, in which an edge exists if the corresponding variable is set to true, and is absent otherwise. We have $\mathrm{clique}_{n,k}(x) = 1$, if and only if the graph instance corresponding to the assignment $x$ contains a $k$-clique, which is a fully connected sub-graph of $k$ vertices.

**Example 1.** *Fig. 1(b) shows a BDD for* $\mathrm{clique}_{4,3}$ *with the variable ordering* $x_1 < x_2 < x_3 < x_4 < x_5 < x_6$. *The highlighted path represents the graph instance which assigns* $x_1 = x_2 = x_3 = x_6 = 1$ *and* $x_4 = x_5 = 0$ *(see Fig. 1(c)). This graph instance contains a* 3*-clique on edges* $x_2$, $x_3$, *and* $x_6$, *thus the function is equal to 1, i.e., the BDD path terminates in node '*$\top$*'.*

When working with BDDs, it is often more convenient to represent a BDD of a Boolean function over $n$ variables as a sequential list of branch instructions $I_{s-1}, I_{s-2}, \ldots, I_1, I_0$, where each $I_k$ has the form $(v_k \,?\, h_k : l_k)$ [13]. In this notation, $v_k$ is the label of the node, and $h_k < k$ and $l_k < k$ are indexes to other branch instructions, called *high* and *low*, respectively. Instructions $I_1 = (n+1 \,?\, 1 : 1)$ and $I_0 = (n+1 \,?\, 0 : 0)$ are special instructions to represent the terminal nodes. They have as vertex labels the "*impossible*" value $n+1$, which is not used in any of the other steps.

**Example 2.** *The BDD in Fig. 1(b) has the following sequential list of branch instructions:*

$$\begin{array}{lll} I_{14} = (1 \,?\, 13 : 12), & I_{13} = (2 \,?\, 11 : 10), & I_{12} = (2 \,?\, 9 : 6), \\ I_{11} = (3 \,?\, 8 : 7), & I_{10} = (3 \,?\, 4 : 6), & I_9 = (3 \,?\, 2 : 6), \\ I_8 = (4 \,?\, 1 : 5), & I_7 = (4 \,?\, 1 : 0), & I_6 = (4 \,?\, 3 : 0), \\ I_5 = (5 \,?\, 1 : 2), & I_4 = (5 \,?\, 1 : 0), & I_3 = (5 \,?\, 2 : 0), \\ I_2 = (6 \,?\, 1 : 0), & I_1 = (7 \,?\, 1 : 1), & I_0 = (7 \,?\, 0 : 0). \end{array}$$

*For example,* $I_{14}$ *refers to the root node labeled 1, and* $I_2$ *refers to the only node labeled 6.*

### B. Majority logic

In this work, we are concerned with Boolean functions $f : \mathbb{B}^n \to \mathbb{B}$ that map $n$ input truth values to one output truth value. The central function in this paper is the *majority-of-three function*. The majority function of three Boolean variables $x$, $y$, and $z$, denoted $\langle xyz \rangle$, evaluates to true if and only if at least two of the three inputs are true. The majority function is monotone and self-dual [13] and can be expressed in disjunctive and conjunctive normal form as

$$\langle xyz \rangle = xy \vee xz \vee yz = (x \vee y)(x \vee z)(y \vee z). \tag{3}$$

Setting any variable to 0 gives the conjunction of the other two variables, and analogously one obtains the disjunction by setting any variable to 1, i.e.,

$$\langle x0y \rangle = x \wedge y \quad \text{and} \quad \langle x1y \rangle = x \vee y. \tag{4}$$

We like to emphasize some basic identities of the majority function which are essential for the forthcoming definition of majority graphs. First, all three arguments to the majority function are *commutative*, i.e.,

$$\langle xyz \rangle = \langle yxz \rangle = \langle zxy \rangle. \tag{5}$$

Also, the majority function evaluates to a single argument if two arguments are equal or inverse to each other, i.e.,

$$\langle xxy\rangle = x \qquad \langle x\bar{x}y\rangle = y. \tag{6}$$

The *associativity rule* on the majority function allows to exchange variables if two operations are nested and share a common variable, i.e.,

$$\langle xu\langle yuz\rangle\rangle = \langle\langle xuy\rangle uz\rangle. \tag{7}$$

The associativity rule becomes obvious, when replacing $u$ by some operator symbol '$\circ$' and the angular brackets by parentheses: $(x \circ (y \circ z)) = ((x \circ y) \circ z)$; as pointed out by Schensted [15]. An alternative way to think about its validity, is by setting $u$ to 0 and 1, and noting that $\wedge$ and $\vee$ are associative [13]. Due to these properties, the set $M = \mathbb{B} = \{0, 1\}$ and the ternary operation $\langle xyz\rangle$ defined according to (3) are a *median algebra* [16]. A median algebra is defined as a set and a majority operator satisfying commutativity, associativity, and the first identity in (6).

Also a *distributivity rule* can be derived from these three rules [17]:

$$\langle xu\langle yvz\rangle\rangle = \langle\langle xuy\rangle v\langle xuz\rangle\rangle \tag{8}$$

To easily memorize this rule, it's handy to replace $u$ by a symbol '$\circ$' and $v$ by a symbol '$\times$'.

Note that in the Boolean case, in which each element has its inverse, the median algebra is a complemented distributive lattice [16] and therefore a Boolean algebra (see, e.g., [18]).

Since the majority function is self-dual, inverters can be *propagated* from the inputs to the outputs, i.e.,

$$\langle\bar{x}\bar{y}\bar{z}\rangle = \overline{\langle xyz\rangle}. \tag{9}$$

The majority function can be generalized for an odd number of $n$ variables to $\langle x_1 \dots x_n\rangle = [x_1 + \dots + x_n > \frac{n}{2}]$. Many of the contributions in this work can be generalized to this general majority function, although the description is restricted to 3-input majority functions in this paper. More details about general majority functions and their properties can be found in the literature [19], [20].

### C. Majority graphs

The mapping approach, which we present in Section III, changes BDDs into majority graphs. In order to keep our notation simple, here we introduce majority graphs; we discuss how our approach can be extended for inverters in Section III-A.

A majority graph is a directed acyclic graph in which each terminal node is a primary input or a constant and each nonterminal node represents a majority operation with three incoming edges. Formally, given a Boolean function $f(x_1, \dots, x_n)$, it is most convenient to represent a majority graph with $r$ majority gates as a chain $x_{n+1}, \dots, x_{n+r}$, where

$$x_i = \langle x_{j(i)} x_{k(i)} x_{l(i)}\rangle \qquad \text{for } n < i \le n + r, \tag{10}$$

with $-1 \le j(i) < i$, $-1 \le k(i) < i$, and $-1 \le l(i) < i$. We also define $x_0 = \bot$ and $x_{-1} = \top$.

We call a majority graph *leafy* if $j(i) \le n$, $k(i) \le n$, or $l(i) \le n$ for all $n < i \le n+r$. In other words, in each majority operation at least one operand is an input variable.

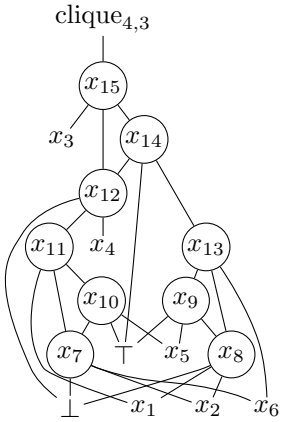



Fig. 2: Majority graph for the function $\mathrm{clique}_{4,3}$

**Example 3.** *Fig. 2 shows the majority graph for the Boolean function* $\mathrm{clique}_{4,3}$. $x_1, \dots, x_6$ *are the primary inputs, while each node* $x_7, \dots, x_{15}$ *represents the majority-of-three function. The majority nodes are given by:*

$$\begin{aligned} x_7 &= \langle x_0 x_2 x_6\rangle, & x_8 &= \langle x_0 x_1 x_2\rangle, & x_9 &= \langle x_{-1} x_5 x_8\rangle, \\ x_{10} &= \langle x_{-1} x_5 x_7\rangle, & x_{11} &= \langle x_1 x_7 x_{10}\rangle, & x_{12} &= \langle x_0 x_4 x_{11}\rangle, \\ x_{13} &= \langle x_6 x_8 x_9\rangle, & x_{14} &= \langle x_{-1} x_{12} x_{13}\rangle, & x_{15} &= \langle x_3 x_{12} x_{14}\rangle. \end{aligned}$$

### D. Related works

Majority logic was intensively studied in the 1960s [21], [22], [7]. In 1962, Akers [21] proposed a method based on truth tables to build both 3-input majority and $n$-input majority networks; Miller and Winder [22] illustrated a geometric process based on Karnaugh maps. In 1964, Amarel et al. [7] proposed algorithms to rewrite majority-$n$ into majority-3. At that time, many majority-based algorithms were proposed but, due to the limited available computational resources, they were not followed by implementations. Recently, majority logic has obtained a renewed interest in the computer science community thanks to its properties that ensure efficient Boolean function optimization and manipulation, and thanks to the many and diversified nanotechnologies that use majority as their building block, see e.g., QCA [23]. In [24], the authors present an algorithm and tool called MALS for majority-based logic synthesis. This algorithm maps the network into 3-input subgraphs, and then uses the geometric method [22] to synthesize them into majority operations. Nanotechnology applications are addressed by mapping the resulting majority networks into majority based devices. In [25], the authors consider majority logic decomposition based on BDDs. In [25], majority dominator nodes are used in order to guide the decomposition process; these nodes allow the identification of candidate functions that can be used to build the majority decomposition. Although, BDDs are used in the implementation of their algorithm, it has a very different nature compared to our work, in which we equate BDDs to majority graphs in the case of monotone Boolean functions. In [26], the authors show how the algebraic properties of 3-input majority logic can be used to optimize Boolean function representations, and has been experimentally demonstrated to optimize area and delay

of CMOS circuits [26]. A generalization to majority operation of $n$ inputs is presented in [19].

## III. Transforming BDDs into Majority Graphs

As discussed in Section II, one can express any Boolean function $f : \mathbb{B}^n \to \mathbb{B}$ in terms of its cofactors using (2). If $f$ is monotone, i.e., $f_{\bar{x}_i} f_{x_i} = 0$ for all $i$, it is possible to use the majority function for decomposition [27]:

$$f = \langle x_i f_{x_i} f_{\bar{x}_i} \rangle = x_i f_{x_i} \oplus x_i f_{\bar{x}_i} \oplus f_{x_i} f_{\bar{x}_i} \tag{11}$$

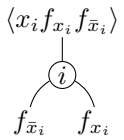
**Remark 1.** *The cofactors $f_{x_i}$ and $f_{\bar{x}_i}$ commute in* (11)*, but they do not in* (2)*.*

**Remark 2.** *The cofactor operation preserves monotonicity in* (2)*. Thus,* (11) *can iteratively be applied to the whole BDD.*

Due to these properties,

$$f = x_i \,?\, f_{x_i} : f_{\bar{x}_i} = \langle x_i f_{x_i} f_{\bar{x}_i} \rangle \tag{12}$$

and one can replace each "Shannon node" by a "majority node" in the BDD of monotone functions:




(13)

**Remark 3.** *Since $f_{x_i}$ and $f_{\bar{x}_i}$ commute, it is no further necessary to distinguish between the two successors in the BDD node. Also, since the resulting majority graph is leafy, we can use a notation analogous to the BDD notation, in which the node is labeled by the variable operand $x_i$. Note that the nodes in BDD and majority graphs use the same notation and have the same shape, but they differ in the operation they implement (i.e., if-then-else and majority, respectively). They also differ in how terminal nodes are drawn.*

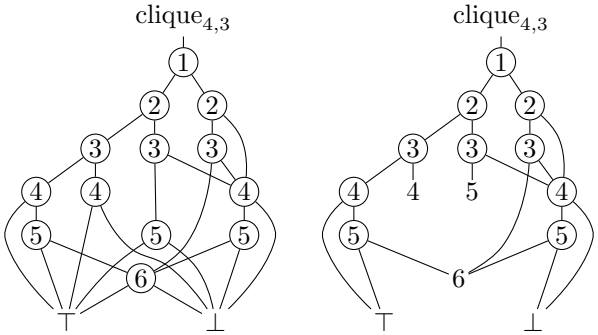
The replacement allows us to generate majority graphs for monotone functions $f(x_1, \ldots, x_n)$ directly from their BDDs using the following simple transformation rule. Let the BDD for $f$ be represented using a sequential list of branch instructions $I_{s-1}, \ldots, I_0$ as described in Example 2. Then a majority graph for $f$ is

$$x_{n+k-1} = \langle x_{v_k} x_{t(h_k)} x_{t(l_k)} \rangle \quad \text{where } I_k = (v_k \,?\, h_k : l_k) \tag{14}$$

for $2 \leq k < s$. In (14) the index transformation function $t : [0, s-1] \mapsto [-1, n+s-2]$ is defined as

$$t(i) = \begin{cases} 0 & \text{if } i = 0, \\ -1 & \text{if } i = 1, \\ n+i-1 & \text{otherwise.} \end{cases} \tag{15}$$

**Example 4.** *We show how to apply the transformation to the BDD in Fig. 1(b). By just translating every BDD node into a MAJ node as in* (13)*, one obtains the majority graph depicted in Fig. 3(a). Note that we removed the boxes around the terminal nodes, mainly to further help distinguishing the two representations.*




(a) Majority graph (b) Majority graph (compact)

Fig. 3: Diagrammatic notation of a leafy majority graph for the function $\mathrm{clique}_{4,3}$

*Since $\langle x_i 01 \rangle = x_i$, we write nodes that have two constant successors also as terminal nodes. This applies to three nodes in the graph. Fig. 3(b) shows the compact version.*

We prove some properties of the transformation.

**Theorem 1.** *The sequence of majority operations derived from* (14) *is a leafy majority graph.*

*Proof.* We have $v_k \leq n < n + k - 1$. This implies that the graph is leafy. Also, $n + h_k - 1 < n + k - 1$, since $h_k < k$. The same applies to $l_k$. □

**Theorem 2.** *The majority graph represents $f$.*

*Proof.* The last step in the majority graph is $x_{n+s-2}$ which corresponds to branch instruction $I_{s-1}$ of the BDD. Therefore, due to (12),

$$f = x_{n+s-2} \overset{(12)}{=} \langle x_{v_{s-1}} x_{t(h_{s-1})} x_{t(l_{s-1})} \rangle$$

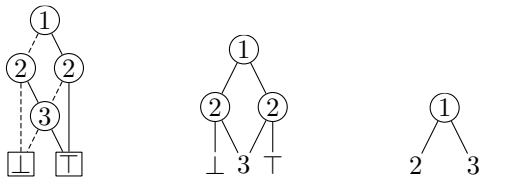
where

$$I_{s-1} = (v_{s-1} \,?\, h_{s-1} : l_{s-1})$$

The rest follows from induction. When $i = 1, 0$, the instructions are $I_1 = (n+1 \,?\, 1 : 1)$ and $I_0 = (n+1 \,?\, 0 : 0)$, respectively, which hold by construction. Let us suppose it holds for all $I_i$, with $2 \leq i \leq k$. $I_{k+1}$ has the form $(v_{k+1} \,?\, h_{k+1} : l_{k+1})$. $v_{k+1}$ is the label of the node, and $h_{k+1} < k+1$ and $l_{k+1} < k+1$ are indexes to other branch instructions, which can only be instructions $I_i$ with $2 \leq i \leq k$. Then it follows that also $I_{k+1}$ holds, and this concludes the proof. □

### A. Discussion on inversions

When reducing the size of the resulting majority graph (which has not complemented edges), some transformation rules may introduce complemented edges in the majority graph and transform it into a *Majority-Inverter Graph* (MIG) [28]. MIGs are majority graphs which make use of complemented edges to represent inversions. In the next section, which deals with deriving optimum majority graphs for majority-5 and majority-7, we do make use of complemented edges to primary inputs and we call them optimum MIGs. However, we also present solutions of the same size that do not require complemented edges.




(a) BDD (b) not optimized majority-3 (c) optimum result

Fig. 4: Majority-3 from its optimal BDD

### B. Discussion on optimality

In this section, we answer the question whether a size-optimum BDD produces a size-optimum MIG. Using a simple counter-example, the majority-3 function, we show that this is not the case. In general, the size of a BDD depends solely on the variable order. Therefore, there are one or more variable orders that lead to the smallest BDD. The next section shows that the optimum BDD for the majority-3 function requires 4 nodes, and therefore the direct transformation into a majority graph requires 3 majority gates. This simple example shows that this MIG is far from the optimal; and since the majority-3 function is symmetric (see, e.g., [13]), it is not affected by the variable order. Optimal MIG representations can be achieved by making use of transformation rules and identities presented in Section II. It has been proven in [28] that any other functionally equivalent MIG can be reached using a finite sequence of transformation rules. It is worth noting that these identities and rules reshape the MIG in a way such that the inverse transformation (from MIG to BDD) is not always possible (see, e.g., majority-5 optimum result given in the next section). We can conclude that the BDD representations (for all variable orderings) are a subset of all MIG representations.

## IV. Mapping Majority-$n$

In this section, we illustrate how to use the proposed approach to map the majority-$n$ function into MIGs with majority-3 operations. For example, for the majority-3 function $\langle x_1 x_2 x_3 \rangle$, the BDD and its corresponding majority graph are shown in Fig. 4(a) and (b), respectively. But the majority expression that Fig. 4(b) represents is $\langle x_1 \langle x_2 0 x_3 \rangle \langle x_2 1 x_3 \rangle \rangle$, which is of course far from optimal. The distributivity rule applies to this expression, giving $\langle \langle x_1 01 \rangle x_2 x_3 \rangle = \langle x_1 x_2 x_3 \rangle$; its diagrammatic notation is shown in Fig 4(c).

This example was rather trivial. Next, we show that we can rewrite majority-5 and majority-7 into their optimum MIGs using the identities presented in Section II and new identities which will be described first. We present majority-5 and majority-7 optimization in detail in order to (i) demonstrate that our method leads to the optimum known results, and (ii) to illustrate the complete optimization procedure. Then, we show that the optimization procedure can be generalized for larger $n$ by showing the optimized majority-9, leading to a new best solution with 15 majority-3 operations.

### A. Replacement rule and swapping rule

The replacement rule describes under which condition one operand in a majority expression can be replaced by another one.

**Theorem 3** (Replacement rule)**.** *We have*

$$\langle xyz \rangle = \langle wyz \rangle \quad \textit{if and only if } (y \oplus z)(w \oplus x) = 0,$$

*or in other words* $y \neq z \Rightarrow w = x$.

*Proof.* First note that $\langle xyz \rangle = x(y \oplus z) \oplus yz$. Then

$$\begin{aligned} 0 &= \langle xyz \rangle \oplus \langle wyz \rangle \\ &= x(y \oplus z) \oplus yz \oplus w(y \oplus z) \oplus yz \\ &= x(y \oplus z) \oplus w(y \oplus z) = (w \oplus x)(y \oplus z), \end{aligned}$$

which concludes the proof. An alternative way to see that the theorem is true, is by applying the majority law to $y$ and $z$. □

One can readily verify that the relevance rule presented in [26] is a special case of the replacement rule.

**Corollary 1** (Relevance rule)**.** *We have* $\langle xyz \rangle = \langle x_{y/\bar{z}} yz \rangle$*, where* $x_{y/\bar{z}}$ *is obtained by replacing all occurrences of* $y$ *with* $\bar{z}$ *in* $x$.

Relevance rule is a special case of the replacement rule when $w = x_{y/\bar{z}}$. It can be easily shown that the condition $(y \oplus z)(w \oplus x) = 0$ is always met. In fact, when $y = z$, $(y \oplus z) = 0$; when $y \neq z = \bar{z}$, $(w \oplus x) = (x_{y/\bar{z}} \oplus x) = (x \oplus x) = 0$.

The swapping rule describes when two operands in a majority expression can be swapped between them.

**Theorem 4** (Swapping rule)**.** *Let* $v_1, v_2, w_1, w_2$ *not depend on* $x$ *and* $y$*. We have* $\langle x \langle y v_1 w_1 \rangle \langle y v_2 w_2 \rangle \rangle = \langle x \langle y v_2 w_1 \rangle \langle y v_1 w_2 \rangle \rangle$*, if* $(v_1 \oplus v_2)(w_1 \oplus w_2) = 0$.

In other words, the swapping rule describes a condition in which the subfunctions $v_1$ and $v_2$ can be swapped. Due to commutativity, one can also swap $w_1$ with $w_2$, or both.

*Proof.* From the condition, it follows that we have either $(v_1 = v_2)$ or $(w_1 = w_2)$. Therefore, one of the following cases is true.

Case $(v_1 = v_2)$: Then $\langle x \langle y v_1 w_1 \rangle \langle y v_1 w_2 \rangle \rangle = \langle x \langle y v_1 w_1 \rangle \langle y v_1 w_2 \rangle \rangle$ is trivially true.

Case $(w_1 = w_2)$: Then $\langle x \langle y v_1 w_1 \rangle \langle y v_2 w_1 \rangle \rangle = \langle x \langle y v_2 w_1 \rangle \langle y v_1 w_1 \rangle \rangle$ due to commutativity. □

These new rules will be used in the next section to derive the optimum solution for majority-5 and majority-7. However, they can be employed in more general majority-based optimization.

### B. Mapping majority-5

We now show how to use these diagrams for the majority decomposition of the majority-5 expression $\langle x_1 x_2 x_3 x_4 x_5 \rangle$. The starting point derived from a BDD similar to the majority-3 case is shown in Fig. 5(a). First, distributivity is applied to the gray nodes to change $\langle x_3 \langle x_4 0 x_5 \rangle \langle x_4 1 x_5 \rangle \rangle$ into $\langle x_3 x_4 \langle x_5 01 \rangle \rangle = \langle x_3 x_4 x_5 \rangle$. However, since the nodes labeled '4' have other ingoing edges, these nodes need to be preserved. The resulting network is shown in Fig. 5(b). Relevance rule is applied on the gray nodes: $\langle x_3 0 \langle x_4 0 x_5 \rangle \rangle = \langle x_3 0 \langle \bar{x}_3 x_4 x_5 \rangle \rangle$ and $\langle x_3 1 \langle x_4 1 x_5 \rangle \rangle = \langle x_3 1 \langle \bar{x}_3 x_4 x_5 \rangle \rangle$. This allows us to replace $\perp$ and $\top$ by $\bar{x}_3$ for the gray nodes, thereby making the two

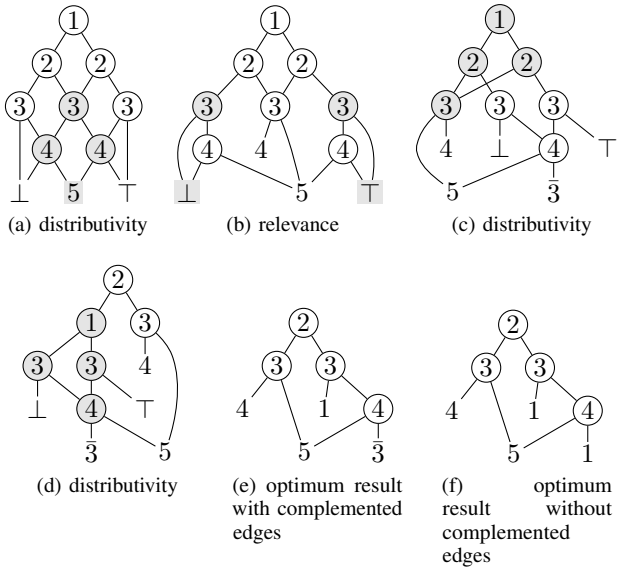



Fig. 5: Decomposing majority-5 into majority-3

nodes labeled '4' structurally equal, as shown in Fig. 5(c). After again applying the distributivity law twice, one obtains the final network in Fig. 5(e). This network has 4 nodes, which is optimum. The final expression is

$$\langle x_2 \langle x_3 x_4 x_5 \rangle \langle x_1 x_3 \langle \bar{x}_3 x_4 x_5 \rangle \rangle \rangle. \tag{16}$$

Note that an alternative expression without complemented edges can be obtained by applying the relevance rule on Fig. 5(e). The final expression without complemented edges is

$$\langle x_2 \langle x_3 x_4 x_5 \rangle \langle x_1 x_3 \langle x_1 x_4 x_5 \rangle \rangle \rangle. \tag{17}$$

This is shown in Fig. 5(f).

### C. Mapping majority-7

Here, we decompose the majority-7 expression $\langle x_1 x_2 x_3 x_4 x_5 x_6 x_7 \rangle$ using an approach similar to the one employed for majority-5, and we demonstrate that the optimum known result can be obtained with our methodology. Fig. 6(a) shows the starting point, derived from its BDD as done for the majority-3 in Fig. 4. First, the majority-5 graph is identified (gray in Fig. 6(a)) and written as node $M_5 = \langle x_3 x_4 x_5 x_6 x_7 \rangle$; its expression will be used again later. From Fig. 6(b) to 6(e), only the left branch of node labeled '1' is considered, but it is worth noting that all steps are applied in the same way also to its right branch. First, the majority-3 highlighted in Fig. 6(c) is changed into its optimum result, then the relevance rule is applied on the gray colored nodes of Fig. 6(d), allowing the replacement of 0 with $\bar{x}_5$. Further, the distributivity rule is applied on Fig. 6(e). The same procedure (Fig. 6(b) to 6(e)) works for the right branch and the complete graph is shown in Fig. 6(f). Here, the distributivity rule is applied on the topmost nodes. In Fig. 6(g), the two nodes labeled '4' are almost identical; they only differ in $\perp$ and $\top$. In this scenario, $\perp$ and $\top$ are interchangeable, and this allows us to replace $\perp$ and $\top$ with two signals of opposite polarities. Fig. 6(h) shows the resulting network, where $\perp$ is replaced by the input $x_3$, and $\top$ by the input $\bar{x}_3$. Further, the replacement rule is applied on node labeled '1'. The highlighted branch (being $x$ here) is substituted with a new sub-graph $w$, resulting in Fig. 6(i). The replacement rule can also be applied in a similar way on the left branch of node labeled '1'; the resulting graph is Fig. 6(j). The replacement rule allows us to apply the distributivity rule on the graph (highlighted in gray). The new graph is shown in Fig. 6(k); the node $M_5$ has been changed into its optimum expression from Fig. 5(e). The swapping rule is next applied on the graph shown in Fig. 6(k). Since $(v_1 \oplus v_2)(w_1 \oplus w_2) = 0$, then $\langle x \langle y v_1 w_1 \rangle \langle y v_2 w_2 \rangle \rangle = \langle x \langle y v_2 w_1 \rangle \langle y v_1 w_2 \rangle \rangle$ and branches $w_1$ and $w_2$ can be exchanged between them, resulting in Fig. 6(l). Distributivity and relevance are applied (highlighted in Fig 6(l)) to obtain the network of Fig. 6(m). This is the final network, which has 7 nodes and which corresponds to the optimum solution. The final resulting expression is

$$\langle x_2 \langle x_5 \langle x_1 x_3 x_4 \rangle \langle \bar{x}_5 x_6 x_7 \rangle \rangle \langle x_4 \langle x_5 x_6 x_7 \rangle \langle x_1 x_3 \bar{x}_4 \rangle \rangle \rangle. \tag{18}$$

Note that an alternative expression without complemented edges is

$$\langle x_2 \langle x_5 \langle x_1 x_3 x_4 \rangle \langle x_6 x_7 \langle x_1 x_3 x_4 \rangle \rangle \rangle \langle x_4 \langle x_5 x_6 x_7 \rangle \langle x_1 x_3 \langle x_5 x_6 x_7 \rangle \rangle \rangle \rangle. \tag{19}$$

This expression can be obtained by applying the relevance rule on the highlighted nodes in Fig. 7(a). The optimum result without complemented edges is shown in Fig. 7(b). Further, it is worth noting that to find the same solution, exhaustive search using a SAT-solver (see, e.g., [12]) takes around 0.5 seconds; here, the entire derivation is provided.

We have applied the same rules for the decomposition of the majority-9 expression $\langle x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 \rangle$, using an approach similar to the one employed for majority-5 and majority-7, and we have demonstrated the realization of majority-9 using only 15 nodes. The initial (not optimized) majority graph is the one shown in Fig. 8(a). As of today, $M(9)$ is still unknown. We applied similar reduction and rewriting techniques as in the case of majority-7 and were able to reduce the number of majority-3 operations to 15 (Fig. 8(b)). To the best of our knowledge, this is the smallest realization of majority-9 in terms of majority-3. State-of-the-art exact synthesis is not able to find an optimum representation for majority-9. The best-known lower bound is 10; this was derived by showing that no majority graph can be found with 9 nodes or less.

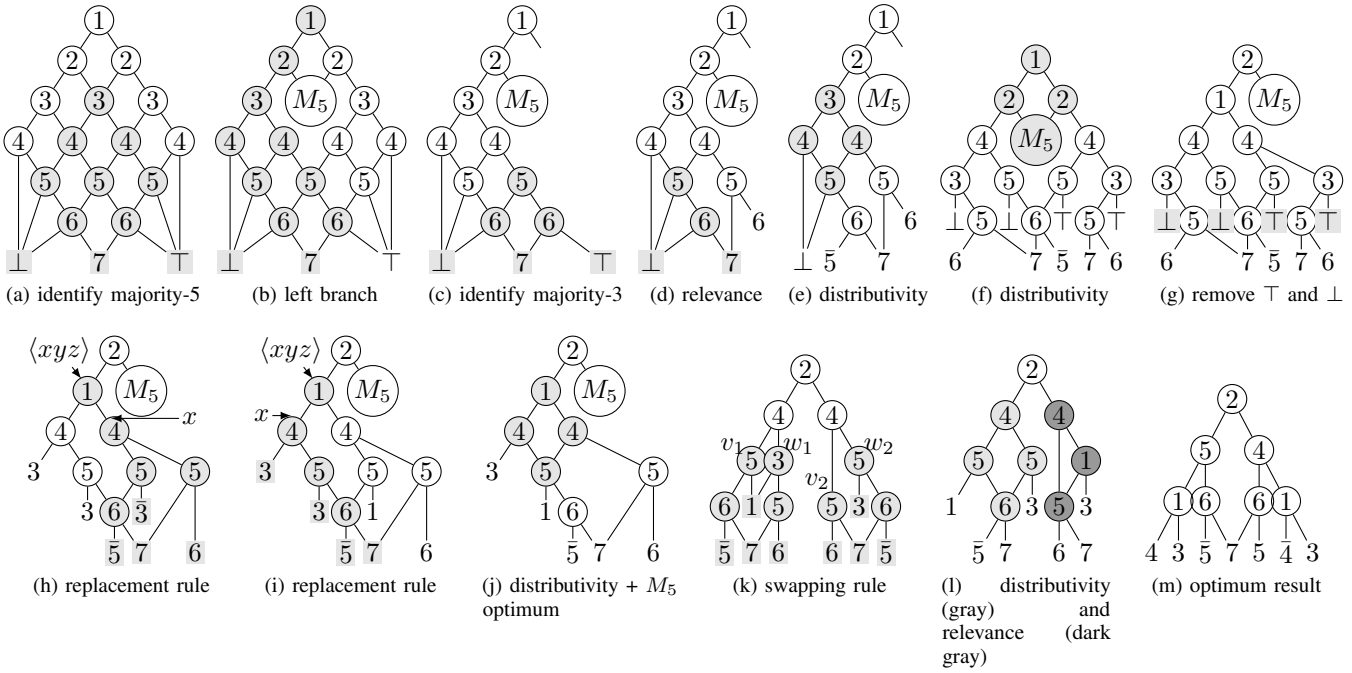
## V. Upper bounds

In this section, we compare upper bounds of the proposed method with the state-of-the-art. Further, we show that our method leads to a tighter upper bound for majority-9.

Our proposed synthesis method from a BDD suggests an upper bound $u_{\mathrm{B}}(n)$ for the majority-$n$ function.

**Theorem 5.** *The majority-$n$ function $\langle x_1 \dots x_n \rangle$ can be realized using a majority graph with at most $u_{\mathrm{B}}(n) \leq \left(\left\lceil \frac{n}{2} \right\rceil\right)^2 - 1$ majority operations.*

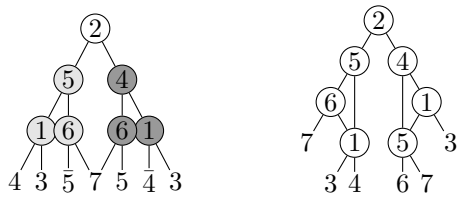
*Proof.* Let $n = 2k + 1$, i.e., $k = \left\lfloor \frac{n}{2} \right\rfloor$. The BDD to represent $\langle x_1 \dots x_n \rangle$ has a diamond shape with 1 node at the first level, 2 nodes at the second level, until $k + 1$ nodes at level $k + 1$.



(a) identify majority-5 (b) left branch (c) identify majority-3 (d) relevance (e) distributivity (f) distributivity (g) remove $\top$ and $\bot$

(h) replacement rule (i) replacement rule (j) distributivity + $M_5$ optimum (k) swapping rule (l) distributivity (gray) and relevance (dark gray) (m) optimum result

Fig. 6: Decomposing majority-7 into majority-3



(a) optimum majority-7, with complemented edges (b) optimum majority-7, without complemented edges

Fig. 7: Optimum majority-7 with (a) and without (b) complemented edges, respectively

Then, the number of nodes per level decreases: it has $k$ nodes on level $k+2$, $k-1$ nodes on level $k+3$, until 1 node on the last level. Further, this node in the last level will be a leaf in the majority graph (thus the -1). In summary, this leads to:

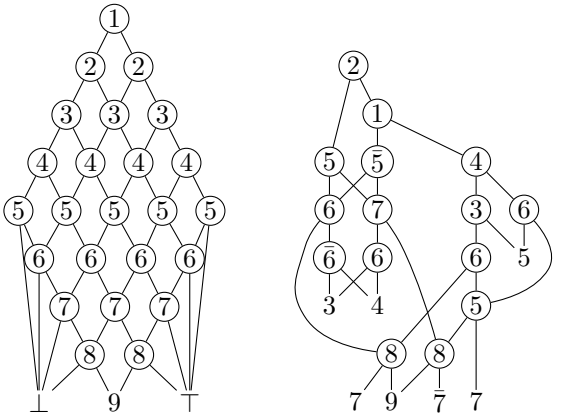
$$u_{\mathrm{B}} \leq \sum_{i=1}^{k+1} i + \sum_{i=1}^{k} i - 1.$$

From this, we can derive

$$\sum_{i=1}^{k+1} i + \sum_{i=1}^{k} i - 1 = \frac{(k+1)(2k+2)}{2} - 1 = (k+1)^2 - 1$$
$$= \left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right)^2 - 1 = \left(\left\lceil \frac{n}{2} \right\rceil\right)^2 - 1.$$

□

Theorem 5 yields a quadratic upper bound, but as discussed in the introduction, it is possible to do much better. A quasi-linear construction follows from sorter networks [29].



(a) not optimized majority-9: 24 nodes (b) optimized majority-9: 15 nodes

Fig. 8: Decomposing majority-9 into majority-3

We simply sort the $n$ bits and pick the one that ends up in the middle position. Sorter networks consist of comparators, which are functions that map a pair of numbers $x, y \mapsto \min(x, y), \max(x, y)$. For Boolean numbers we have $\min(x, y) = x \wedge y$ and $\max(x, y) = x \vee y$, i.e., each comparator in a sorter network can be composed with 2 majority-3 operations. Let $S(n)$ be the optimum number of comparators in a sorter network that sorts $n$ elements. Then an upper bound on the number of majority-3 operations is $u_{\mathrm{S}}(n) \leq 2S(n)$. From [10], it is known that $S(10) = 29$. Using, e.g., the systematic construction from Batcher [30], one can derive upper bounds for larger $n$.

Dor and Zwick showed that less than $2.942n$ comparisons are necessary to select the median value from a set of $n$ numbers, without the need to sort them [9]. Applying it directly, it

TABLE I: Upper bounds on the number of majority-3 operations to realize majority-$n$

| $n$ | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 |
|---|---|---|---|---|---|---|---|---|
| Optimum ($M(n)$) | 1 | 4 | 7 | | | | | |
| Optimized BDDs | 1 | 4 | 7 | **15** | | | | |
| BDDs | 3 | 8 | 15 | 24 | 35 | 48 | 63 | 80 |
| Sorter networks | 6 | 18 | 32 | 50 | 70 | 90 | 112 | 142 |
| Median selection* | 18 | 30 | 42 | 53 | 65 | 77 | 89 | 101 |

* These numbers are based on the number of comparators in the construction of [9], but do not take other operations into account.

would lead to an upper bound of $5.884n$ majority-3 operations to decompose majority-$n$. However, this number needs to be treated more carefully, since their analysis is based on the *comparison model* in which *only* the number of comparators are counted and all other operations are considered free.

Both the construction on sorter networks and median selection have asymptotically better upper bounds compared to the quadradic bound from the BDD construction. However, when actually calculating the numbers for small instances with $n \leq 17$, the BDD approach yields the smallest values (see Table I, which also shows the known optimum results up to $n = 7$ that were confirmed using exhaustive enumeration [12]). The approach is therefore a good starting point for finding compact majority-$n$ realizations for small $n$. The results obtained using BDD + optimization rules are listed in Table I as "Optimized BDD". Our method is able to obtain (i) the optimum known results for majority-5 and -7, and (ii) the best result for majority-9. One may be able to derive a general derivation procedure to obtain optimum or close to optimum majority-$n$ realizations for $n \geq 9$.

## VI. Conclusions

We have described a mapping method for monotone Boolean functions, based on the transformation of BDDs into majority graphs. We have used the proposed method to map the majority-$n$ function into MIGs with majority-3 operations. Due to the more compact initial representation, the approach is favorable compared to methods based on median selection and sorter networks, for small $n$. In particular, we were able to derive the optimum MIG for majority-5 and majority-7 functions. For their optimization, we introduced two useful identities for majority-based function optimization.

## Acknowledgment

This research has been supported by the Swiss National Science Foundation (200021-169084 MAJesty) and by H2020-ERC-2014-ADG 669354 CyberCare.

## References

[1] A. Khitun and K. L. Wang, "Non-volatile magnonic logic circuits engineering," *Journal of Applied Physics*, vol. 110, no. 034306, 2011.
[2] C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernstein, "Quantum cellular automata," *Nanotechnology*, vol. 4, no. 1, pp. 49–57, 1993.
[3] D. E. Nikonov, G. I. Bourianoff, and T. Ghani, "Proposal of a spin torque majority gate logic," *IEEE Electron Device Letters*, vol. 32, no. 8, pp. 1128–1130, 2011.
[4] M. Agrawal, E. Allender, and S. Datta, "On TC0, AC0, and arithmetic circuits," *Journal of Computer and System Sciences*, vol. 60, no. 2, pp. 395 – 421, 2000.
[5] W. Hesse, E. Allender, and D. A. M. Barrington, "Uniform constant-depth threshold circuits for division and iterated multiplication," *Journal of Computer and System Sciences*, vol. 65, no. 4, pp. 695–716, 2002.
[6] J. H. Reif and S. R. Tate, "On threshold circuits and polynomial computation," *SIAM Journal on Computing*, vol. 21, no. 5, pp. 896–908, 1992.
[7] S. Amarel, G. Cooke, and R. O. Winder, "Majority gate networks," *IEEE Transactions on Electronic Computers*, no. 1, pp. 4–13, 1964.
[8] A. S. Kulikov and V. V. Podolskii, "Computing majority by constant depth majority circuits with low fan-in gates," in *Symposium on Theoretical Aspects of Computer Science*, 2017, pp. 49:1–49:14.
[9] D. Dor and U. Zwick, "Selecting the median," *SIAM Journal on Computing*, vol. 28, no. 5, pp. 1722–1758, 1999.
[10] M. Codish, L. Cruz-Filipe, M. Frank, and P. Schneider-Kamp, "Twenty-five comparators is optimal when sorting nine inputs (and twenty-nine for ten)," in *Int'l Conf. on Tools with Artificial Intelligence*, 2014, pp. 186–193.
[11] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. on Computers*, vol. 35, no. 8, pp. 677–691, 1986.
[12] M. Soeken, L. G. Amarù, P.-E. Gaillardon, and G. De Micheli, "Exact synthesis of majority-inverter graphs and its applications," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2017, accepted.
[13] D. E. Knuth, *The Art of Computer Programming, Volume 4A*. Addison-Wesley, 2011.
[14] S. Jukna, *Boolean Function Complexity*. Springer, 2012.
[15] C. Schensted, 1978, a letter to Martin Gartner from December 9, 1978.
[16] G. Birkhoff and S. A. Kiss, "A ternary operation in distributive lattices," *Bulletin of the American Mathematical Society*, vol. 53, no. 8, pp. 749–752, 1947.
[17] M. Sholander, "Medians and betweenness," *Proceedings of the American Mathematical Society*, vol. 5, pp. 801–807, 1954.
[18] T. Sasao, *Switching Theory for Logic Synthesis*. Springer, 1999.
[19] L. G. Amarù, P.-E. Gaillardon, A. Chattopadhyay, and G. De Micheli, "A sound and complete axiomatization of majority-$n$ logic," *IEEE Trans. on Computers*, vol. 65, no. 9, pp. 2889–2895, 2016.
[20] A. Chattopadhyay, L. G. Amarù, M. Soeken, P.-E. Gaillardon, and G. De Micheli, "Notes on majority Boolean algebra," in *Int'l Symp. on Multiple-Valued Logic*, 2016, pp. 50–55.
[21] S. B. Akers Jr., "Synthesis of combinational logic using three-input majority gates," in *Symp. on Switching Circuit Theory and Logical Design*, 1962, pp. 149–157.
[22] H. S. Miller and R. O. Winder, "Majority-logic synthesis by geometric methods," *IRE Trans. Electronic Computers*, vol. 11, no. 1, pp. 89–90, 1962.
[23] P. D. Tougaw and C. S. Lent, "Logical devices implemented using quantum cellular automata," *Journal of Applied Physics*, vol. 75, no. 3, pp. 1818–1825, 1993.
[24] R. Zhang, P. Gupta, and N. K. Jha, "Majority and minority network synthesis with application to QCA-, SET-, and TPL-Based nanotechnologies," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 26, no. 7, pp. 1233–1245, 2007.
[25] L. G. Amarù, P. Gaillardon, and G. D. Micheli, "BDS-MAJ: a BDD-based logic synthesis tool exploiting majority logic decomposition," in *Design Automation Conference*, 2013, pp. 47:1–47:6.
[26] L. G. Amarù, P.-E. Gaillardon, and G. De Micheli, "Majority-inverter graph: A novel data-structure and algorithms for efficient logic optimization," in *Design Automation Conference*, 2014, pp. 194:1–194:6.
[27] S. B. Akers Jr., "A truth table method for the synthesis of combinational logic," *IEEE Trans. Electronic Computers*, vol. 10, no. 4, pp. 604–615, 1961.
[28] L. G. Amarù, P.-E. Gaillardon, and G. De Micheli, "Majority-inverter graph: A new paradigm for logic optimization," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 35, no. 5, pp. 806–819, 2016.
[29] D. E. Knuth, *The Art of Computer Programming, Volume 3, Second Edition*. Addison-Wesley, 1998.
[30] K. E. Batcher, "Sorting networks and their applications," in *AFIPS Sprint Joint Computing Conference*, 1968, pp. 307–314.