Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 30(9): 3320–3336.
DOI: 10.3934/era.2022168
Received: 26 May 2022
Revised: 28 June 2022
Accepted: 01 July 2022
Published: 11 July 2022

Research article

Simplification of logical functions with application to circuits

Jun-e Feng[1], **Rong Zhao**[1,*] **and Yanjun Cui**[2]

[1] School of Mathematics, Shandong University, Jinan 250100, China

[2] College of Science and Engineering, University of Minnesota Twin Cities, Minneapolis 55455, USA

* **Correspondence:** Email: zhaorongjy1126@163.com.

Abstract: The simplification problem of logical functions is investigated via the matrix method. First, necessary and sufficient conditions are put forward for the decomposition of logical matrices. Based on this, several criteria are proposed for the simplification of logical functions. Furthermore, an algorithm, which can derive simpler logical forms, is developed, and illustrative examples are given to verify the effectiveness. Finally, the obtained theoretical results are applied to the simplification of electric circuits.

Keywords: circuits; logical functions; simplification; semi-tensor product

1. Introduction

Logic algebra was first given by George Boole in 1854 to describe the laws of human thought [1], and that is why we call it Boolean algebra. Then, Boolean logic was used to design circuits by Claude Shannon in 1938 [2]. The functions used in circuits of computers and other electronic devices (e.g., switch devices and optical devices) are all logical (Boolean) ones, in which the input and output variables all belong to $\{0, 1\}$ with 0 and 1 representing "off" and "on", respectively. In all Boolean operations, there are three important ones, which are negation (complementation), disjunction (Boolean sum) and conjunction (Boolean product). Other operations in Boolean algebra can be expressed by these three ones, so the set of these three operations is called an adequate set [3]. Although there are some smaller adequate sets, the adequate set with these three operations is most commonly used. In practical circuits of a computer, these three operations are implemented by the Invertor logic gate, OR logic gate and AND logic gate, the number of which directly affects the efficiency of a combinational circuit. The circuits are usually implemented by the disjunctive normal form (also called the sum-of-products expansion) [2, 5, 6]. There are often some terms unnecessary in the disjunctive normal form. For example, $(x \wedge y \wedge z) \vee (x \wedge \bar{y} \wedge z)$ can be simplified to $x \wedge z$. Here, $x, y, z \in \{0, 1\}$ are logical variables,

$\bar{y}$ represents the complementation of logical variable y, and $\vee, \wedge$ are the operations disjunction and conjunction, respectively. It is obvious that the later one implemented by only one logic gate is simpler than the former one, which requires four logic gates, as shown in Figure 1. Therefore, the minimization or simplification of the disjunctive normal form is very important both to the cost and the efficiency of computers and other electronic devises.

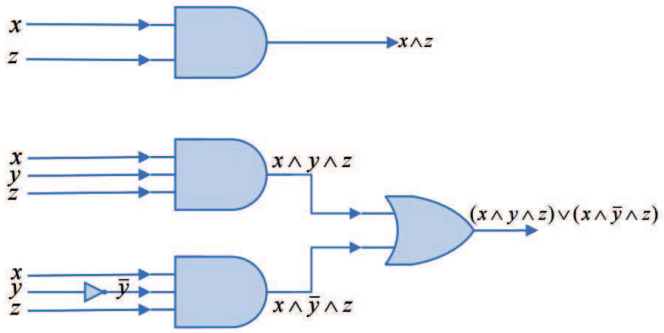

Figure 1. Two circuits with the same output.

Simplification of the disjunctive normal form means not only reduction of the number of the disjunction terms, but also reduction of the number of conjunctions in every disjunction term. The following is a review of the research status of minimizing logical functions. In 1953, Maurice Karnaugh proposed the Karnaugh map method to minimize the disjunctive normal form by hand [4]. However, the Karnaugh map method can simplify circuits with up to six variables. In fact, the method becomes rather complex even for five or six variables. Then, in the 1960s, the Quine-McCluskey method was invented to minimize circuits, which can be automatically done by a computer. Circuits with ten variables can be simplified via the Quine-McCluskey method. Since then, newer algorithms have been devised for minimizing circuits ([5] and [6]), and some of them can handle circuits with up to 25 variables. Some other new methods, such as by binary decision diagrams [7] and cube algebra [8], for the simplification of logical functions have also been proposed, but they are all graph-based approaches. It has been proved that the problem of minimizing circuits is an NP-complete problem [4]. In practice, it is enough to simplify circuits with a larger number of literals, and not necessary to minimize them.

On the other hand, Boolean functions can be expressed in algebraic forms via the semi-tensor product of matrices, which was first proposed by Prof. Cheng and his colleagues [9]. Based on the algebraic forms, investigations of Boolean networks have arrived at a new level. The earliest analysis and control on Boolean networks can be found in the literature [10]. Since then, a variety of research results on Boolean networks have been obtained. Here, we give some recent ones. Pinning controllability and set controllability of BNs were considered in [11] and [12], respectively. Via matrix equations, necessary and sufficient conditions for observability [13] were given, and then the minimum-time control for observability of BNs was studied in [14]. Note that observability aims to determine the initial state by the knowledge of input-output data. Correspondingly, detectability or reconstructibility of BNs, aiming to determine the current state by the knowledge of input-output data, was also discussed in [15, 16]. Furthermore, state feedback control [17, 18], output feedback control [19] and aperiodic sampled-data control [20] for stabilization of BNs were taken into consideration. In addition, the optimal control problem [21–23], the disturbance decoupling problem [24, 25] and other control problems [26–30] have also achieved important research results one after another.

Since the Boolean network is studied via its algebraic form, the corresponding controllers or realizations are all obtained in terms of algebraic expressions. The algebraic expression should be transformed into its logical form, which only can be implemented in practice. [31] proposed a general method for the transformation from the algebraic form to the logical one. If the algebraic expression has some special structure, then its logical form often is complex only using the general transformation method of [31]. As analysis above, it is difficult to implement the complex logical function in practice, and it will take more cost. Therefore, in the process of transformation, we should simply the algebraic expressions if they have some special forms.

In this paper, we propose a matrix-based method to simplify logical functions. The main contributions of the paper are summarized as follows.

1) The decomposition conditions of a logical matrix into the Kronecker product of some (logical) matrices are first proposed.
2) Several criteria under which the algebraic expressions of logical functions can be simplified are presented. Combining with the method of [31], an algorithm is devised for the simplification of logical functions.
3) Moreover, the results of this paper are applicable to the design and simplification of circuits.

The rest of this paper is organized as follows: Some notations and necessary preliminaries are given in Section 2. In Section 3, the decomposition of logical matrices is addressed. Section 4 discusses the simplification of logical functions, and at the same time, several criteria and corresponding algorithm are presented. In Section 5, the results of this paper are applied to the design of circuits. Finally, Section 6 makes the conclusion to this paper.

2. Preliminaries

In this section, we will introduce some notations and some necessary preliminaries, which will be used throughout this paper. First, we list the notations, most of which are from the literature [10].

- $\mathcal{M}_{m\times n}$: the set of $m \times n$ real matrices. Denote $\mathcal{M}_m = \mathcal{M}_{m\times 1}$.
- $\mathcal{D} := \{0, 1\}$, where $1 \sim T$ means "true" and $0 \sim F$ means "false". A logical variable A takes value from $\mathcal{D}$, expressed as $A \in \mathcal{D}$. Identify $T = 1 \sim \delta_2^1$, $F = 0 \sim \delta_2^2$.
- δ_n^i: the i-th column of the identity matrix I_n. Denote $\Delta_n := \{\delta_n^i | i = 1, \cdots, n\}$ and $\Delta := \Delta_2$.
- $\mathbf{1}_n := \delta_n^1 + \delta_n^2 + \cdots + \delta_n^n$.
- For a matrix $L \in \mathcal{M}_{m\times n}$, denote by $\mathrm{Col}_i(L)$ and $\mathrm{Col}(L)$ the i-th column of L and the set of all columns of L, respectively.
- Matrix $L \in \mathcal{M}_{m\times n}$ is called a logical matrix if $\mathrm{Col}(L) \subset \Delta_n$. In particular, L is called a logical vector when $n = 1$. Denote by $\mathcal{L}_{m\times n}$ the set of $m \times n$ logical matrices.
- If $L \in \mathcal{L}_{n\times r}$, by definition it can be expressed as $L = [\delta_n^{i_1}, \delta_n^{i_2}, \cdots, \delta_n^{i_r}]$. For the sake of compactness, it is briefly denoted by $L = \delta_n[i_1, i_2, \cdots, i_r]$.
- For $A = (a_{ij}) \in \mathcal{M}_{m\times n}$, $B = (b_{ij}) \in \mathcal{M}_{p\times q}$, the Kronecker product of matrices A and B is defined as

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{pmatrix} \in M_{mp\times nq}.$$

- Let $A \in \mathcal{M}_{m\times n}, B \in \mathcal{M}_{p\times q}, t = lcm(n, p)$. Then, the semi-tensor product (STP) of A and B is

$$A \ltimes B := (A \otimes I_{t/n})(B \otimes I_{t/p}),$$

  where $lcm(n, p)$ represents the least common multiple of n and p.

When $n = p$, $A \ltimes B$ is the traditional matrix product, which means that STP is a generalization of the traditional matrix product. It is easy to check that STP keeps all the properties of hte traditional matrix product available.

Moreover, STP satisfies a pseudo commutative law. For any column vector $x \in \mathcal{M}_t$ and any matrix $A \in \mathcal{M}_{m\times n}$, we have

$$x \ltimes A = (I_t \otimes A) \ltimes x. \tag{2.1}$$

In this paper, the symbol "$\ltimes$" represents the semi-tensor product of matrices, which is often omitted in most places. We refer to [9] for the details.

Definition 1. *Define a swap matrix as*

$$W_{[m,n]} = \left[I_m \otimes \delta_n^1, I_m \otimes \delta_n^2, \cdots, I_m \otimes \delta_n^n\right].$$

The following properties are fundamental for swap matrices.

Lemma 1. *Given two column vectors $x \in \mathcal{M}_n, y \in \mathcal{M}_m$,*

$$W_{[m,n]} \ltimes x \ltimes y = y \ltimes x.$$

Definition 2. *A power-reducing matrix is defined as*

$$P_k^r = diag\{\delta_k^1,\ \delta_k^2,\ \cdots,\ \delta_k^k\}.$$

Then, we have the following lemma.

Lemma 2. *Given any column vector $x \in \Delta_k$,*

$$x^2 = P_k^r x.$$

Using STP and vector expression of logical variables, any logical function can be equivalently converted into its algebraic form [9].

Lemma 3. *If $y = f(x_1, x_2, \ldots, x_n) : \mathcal{D}^n \to \mathcal{D}$ is a Boolean function, then there exists a unique matrix $M_f \in \mathcal{L}_{2\times 2^n}$, called the structure matrix of f, such that in vector form $y = M_f \ltimes_{i=1}^n x_i$, with $y = \delta_2^{2-y} \in \Delta_2$ and $x_i = \delta_2^{2-x_i} \in \Delta_2$.*

To illustrate the lemma above, we give the structure matrices of the three commonly used logical operators.

Example 1. *(i) Complementation:*

$$\bar{X} := 1 - X, \quad X \in \mathcal{D}. \tag{2.2}$$

Its structure matrix is

$$M_n = \delta_2[2,\ 1].$$

(ii) Conjunction:

$$X \wedge Y := \min(X, Y), \quad X, Y \in \mathcal{D}. \tag{2.3}$$

Its structure matrix is

$$M_c = \delta_2[1,\ 2,\ 2,\ 2].$$

(iii) Disjunction:

$$X \vee Y := \max(X, Y), \quad X, Y \in \mathcal{D}. \tag{2.4}$$

Its structure matrix is

$$M_d = \delta_2[1,\ 1,\ 1,\ 2].$$

Remark 1. *It is obvious that* $M_0 = \delta_2[2,\ 2]$ *and* $M_0 = \delta_2[1,\ 1]$ *are the structure matrices of constants* 0 *and* 1*, respectively.*

Example 2. *Given two logical functions* $F(X, Y) = (X \wedge Y) \vee (\bar{X} \wedge Y) \vee (X \wedge \bar{Y})$ *and* $G(X, Y) = X \vee (\bar{X} \wedge Y)$*, compute their structure matrices.*

Denote by $x, y \in \Delta$ *and* $f, g : \Delta^2 \to \Delta$ *the vector forms of logical variables* X, Y *and logical functions* F, G*. Then, we have*

$$\begin{aligned}
f &= M_d^2 M_c xy M_c M_n xy M_c x M_n y \\
&= M_d^2 M_c (I_4 \otimes M_c M_n) P_4^r (I_4 \otimes M_c) xyx M_n y \\
&= M_d^2 M_c (I_4 \otimes M_c M_n) P_4^r (I_4 \otimes M_c)(I_8 \otimes M_n) P_4^r xy \\
&= \delta_2[1,\ 1,\ 1,\ 2] xy, \\
g &= M_d x M_c M_n xy \\
&= M_d (I_2 \otimes M_c M_n) x^2 y \\
&= M_d (I_2 \otimes M_c M_n) P_2^r xy \\
&= \delta_2[1,\ 1,\ 1,\ 2] xy.
\end{aligned}$$

It is noticed that these three logical functions F, G and $W = X \vee Y$ do have the same structure matrix. In fact, they are different expressions of the same logical function. In the three logical functions, the simplest one is $W = X \vee Y$, and the most complex one is F. The simpler one is easier to implement in practical circuits. However, it is often difficult to find the simplest one, since the computation complex. Hence, we hope to propose a general method to simplify logical functions.

On the other hand, from a given algebraic form of a logical function, we can get the corresponding logical form by the following lemma [31], which comes from Shannon's decomposition [32].

Lemma 4. *Let* $M_f \in \mathcal{L}_{2\times 2^n}$ *be the structure matrix of logical function* $\mathsf{y} = f(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n)$*. Then,*

$$\mathsf{y} = [\mathsf{x}_1 \wedge f_1(\mathsf{x}_2, \ldots, \mathsf{x}_n)] \vee [\bar{\mathsf{x}}_1 \wedge f_2(\mathsf{x}_2, \ldots, \mathsf{x}_n)], \tag{2.5}$$

where the structure matrix of $f_i(\cdot)$ *is* $M_f \delta_2^i$*.*

Finally, to simplify the logical function, we also need the following lemma, which will be used in the decomposition of logical matrices.

Lemma 5.

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD).$$

3. Decomposition of logical matrices

In this section, we will propose some decomposition results of logical matrices using Lemma 5.

Proposition 1. *Consider logical matrix* $L = [L_1\ L_2\ \dots\ L_k]$ *with* $L_i \in \mathcal{L}_{m\times l}$.

1) $L = \bar{L} \otimes \mathbf{1}_l^T$ *with* $\bar{L} \in \mathcal{L}_{m\times k}$ *if and only if*

$$Col_1(L_i) = Col_2(L_i) = \dots = Col_l(L_i),\ \ i = 1, 2, \dots, k. \tag{3.1}$$

Furthermore, if $L = \bar{L} \otimes \mathbf{1}_l^T$ *then* $Col_i(\bar{L}) = Col_{j_i}(L_i)$ *for every* $1 \le j_i \le l$ *and* $i = 1, 2, \dots, k$.

2) $L = \mathbf{1}_k^T \otimes \bar{L}$ *with* $\bar{L} \in \mathcal{L}_{m\times l}$ *if and only if*

$$L_1 = L_2 = \dots = L_k. \tag{3.2}$$

Furthermore, if $L = \mathbf{1}_k^T \otimes \bar{L}$ *then* $\bar{L} = L_i$ *for every* $i = 1, 2, \dots, k$.

Proof. We only prove the first item, and the other one can be proven via a similar method.

(Necessity) Since $L_i \in \mathcal{L}_{m\times l}$ and $L = [L_1\ L_2\ \dots\ L_k]$, we have $L \in \mathcal{L}_{m\times kl}$. If $L = \bar{L} \otimes \mathbf{1}_l^T$ then $\bar{L}$ has k columns. Denote $\bar{L} \otimes \mathbf{1}_l^T = [\bar{L}_1\ \bar{L}_2\ \dots\ \bar{L}_k]$, where

$$\bar{L}_i = Col_i(\bar{L}) \otimes \mathbf{1}_l^T,\ \ i = 1, 2, \dots, k.$$

From $L = \bar{L} \otimes \mathbf{1}_l^T$, we derive $L_i = \bar{L}_i$, which means that all columns of L_i are equal for every $i = 1, 2, \dots, k$.

(Sufficiency) If $L = [L_1\ L_2\ \dots\ L_k]$ satisfying

$$Col_1(L_i) = Col_2(L_i) = \dots = Col_l(L_i),\ \ i = 1, 2, \dots, k,$$

then

$$L = [Col_{j_1}(L_1) \otimes \mathbf{1}_l^T\ Col_{j_2}(L_2) \otimes \mathbf{1}_l^T\ \dots\ Col_{j_k}(L_k) \otimes \mathbf{1}_l^T],\ 1 \le j_i \le l,\ i = 1, 2, \dots, k,$$

which means that L has the form $L = \bar{L} \otimes \mathbf{1}_l^T$ with

$$\bar{L} = [Col_{j_1}(L_1)\ Col_{j_2}(L_2)\ \dots\ Col_{j_k}(L_k)],$$

with $1 \le j_i \le l,\ i = 1, 2, \dots, k$. The proof is completed.

Example 3. *i) Consider a logical matrix*

$$L = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}.$$

Take $m = k = l = 2$. *It is easy to see* $L = [L_1\ L_2]$ *and* $Col_1(L_i) = Col_2(L_i), i = 1, 2$. *Thus, from item 1) of Proposition 1, we have* $L = \bar{L} \otimes \mathbf{1}_2^T$ *with*

$$\bar{L} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

ii) Consider a logical matrix

$$L = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

Take $m = k$ and $l = 3$. It is easy to see $L = [L_1\ L_2]$ and $L_1 = L_2$. Thus, from item 2) of Proposition 1, we have $L = \mathbf{1}_2^T \otimes \bar{L}$ with

$$\bar{L} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

Combining the two items of Proposition 1, we have the following corollary easily.

Corollary 1. *Consider logical matrix $L = [L_1\ L_2\ \dots\ L_k]$ with $L_i \in \mathcal{L}_{m\times lh}$. Then, $L = \mathbf{1}_k^T \otimes \bar{L} \otimes \mathbf{1}_h^T$ with $\bar{L} \in \mathcal{L}_{m\times l}$ if and only if*

$$L_1 = L_2 = \dots = L_k,$$

and

$$Col_{\alpha h+1}(L_i) = Col_{\alpha h+2}(L_i) = \dots = Col_{\alpha h+h}(L_i), \tag{3.3}$$

where $\alpha = 0, 1 \dots, l-1$ and $i = 1, 2, \dots, k$. Furthermore, if $L = \mathbf{1}_k^T \otimes \bar{L} \otimes \mathbf{1}_h^T$ then

$$\bar{L} = [Col_1(L_i), Col_{h+1}(L_i), \cdots, Col_{h(l-1)+1}(L_i)]$$

for every $i = 1, 2, \dots, k$.

Proposition 1 and Corollary 1 decompose a logic matrix into the Kronecker product of one logic matrix and row vector $\mathbf{1}_k^T$. $\mathbf{1}_k^T$ can be seen as a special logical matrix which belongs to $\mathcal{L}_{1\times k}$. Next, one more general decomposition case is proposed using a similar idea.

Proposition 2. *Assume logical matrix $L = [L_1\ L_2\ \dots\ L_k]$ with $L_i \in \mathcal{L}_{m\times hl}$, and $L_i = [L_1^i\ L_2^i\ \dots\ L_h^i]$ with $L_j^i \in \mathcal{L}_{m\times l}$, $j = 1, 2, \dots, h$ $i = 1, 2, \dots, k$. Then, $L = \bar{L}(I_k \otimes \mathbf{1}_h^T \otimes I_l)$ with $\bar{L} \in \mathcal{L}_{m\times kl}$ if and only if*

$$L_1^i = L_2^i = \dots = L_h^i,\ i = 1, 2, \dots, k. \tag{3.4}$$

Furthermore, if $L = \bar{L}(I_k \otimes \mathbf{1}_h^T \otimes I_l)$, then

$$\bar{L} = [L_{j_1}^1\ L_{j_2}^2\ \dots\ L_{j_k}^k],\ 1 \le j_i \le h,\ i = 1, 2, \dots, k. \tag{3.5}$$

Proof. Via computation, we have

$$I_k \otimes \mathbf{1}_h^T \otimes I_l = diag\{\overbrace{N\ N \cdots N}^{k}\} \in \mathcal{L}_{kl\times khl}$$

with $N = [\overbrace{I_l\ I_l\ \cdots\ I_l}^{h}]$. Then, it follows that

$$\bar{L}(I_k \otimes \mathbf{1}_h^T \otimes I_l) = [\bar{L}_1\ \bar{L}_2\ \dots\ \bar{L}_k]$$

where $\bar{L}_i = [\bar{L}_1^i\ \bar{L}_2^i\ \dots\ \bar{L}_h^i], i = 1, 2, \dots, k$, and

$$\bar{L}_1^i = \bar{L}_2^i = \dots = \bar{L}_h^i = [Col_{\bar{i}+1}(\bar{L})\ Col_{\bar{i}+2}(\bar{L})\ \cdots\ Col_{\bar{i}+l}(\bar{L})]$$

with $\bar{i} = (i-1)\cdot l, i = 1, 2, \ldots, k$.

Hence, it is obvious that $L = \bar{L}(I_k \otimes \mathbf{1}_h^T \otimes I_l)$ with $\bar{L} \in \mathcal{L}_{m\times kl}$ if and only if

$$L_1^i = L_2^i = \ldots = L_h^i,\ i = 1, 2, \ldots, k.$$

Furthermore, it is easy to have $\bar{L} = [L_{j_1}^1\ L_{j_2}^2\ \ldots\ L_{j_k}^k]$. The proof is completed.

Example 4. *Consider the logical matrix*

$$L = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Take $m = k = h = l = 2$. It is easy to see $L = [L_1\ L_2]$, $L_1 = [L_1^1\ L_2^1]$, $L_2 = [L_1^2\ L_2^2]$, and

$$L_1^1 = L_2^1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ L_1^2 = L_2^2 = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}.$$

It is obvious that L satisfies that condition of Proposition 2. Thus, we have $L = \bar{L}(I_2 \otimes \mathbf{1}_2^T \otimes I_2)$ with

$$\bar{L} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}.$$

Propositions 1 and 2 will take a key role in the simplification of logical functions.

4. Simplification of logical functions

From Lemma 5 of Section 3, it is easy to get the following results, which will be useful to the subsequent simplification.

Proposition 3. *For $x \in \Delta_{n_1}$, $y \in \Delta_{n_2}$, $z \in \Delta_{n_3}$, $L_1 \in \mathcal{L}_{m\times n_1}$, $L_2 \in \mathcal{L}_{m\times n_2}$ and $L_3 \in \mathcal{L}_{m\times n_1 n_3}$, we have the following*

1)

$$L_1 x = (L_1 \otimes \mathbf{1}_{n_2}^T)xy.$$

2)

$$L_2 y = (\mathbf{1}_{n_1}^T \otimes L_2)xy.$$

3)

$$L_2 yz = (\mathbf{1}_{n_1}^T \otimes L_2 \otimes I_{n_3})xyz.$$

4)

$$L_3 xz = L_3(I_{n_1} \otimes \mathbf{1}_{n_2}^T \otimes I_{n_3})xyz.$$

Combining Lemma 5 with $xy = x \otimes y$ and $xyz = x \otimes y \otimes z$, the results will be obtained, so it is omitted here for the brevity.

The function of Proposition 3 is to delete the redundant variables in a logic function. For a given algebraic form $y = L \ltimes_{i=1}^n x_i$ with $x_i \in \Delta$ and $L \in \mathcal{L}_{2\times 2^n}$, we could get its corresponding logical function

using Lemma 4 repeatedly. However, we hope to get a simpler form of it. In order to derive a simpler logical function, we simplify the algebraic form in the process of using Lemma 4. The following corollary is obtained directly from Propositions 1 to 3 and Corollary 1.

Corollary 2. *Considering a given algebraic form $y = L \ltimes_{i=1}^{n} x_i$ with $x_i \in \Delta$ and $L \in \mathcal{L}_{2\times 2^n}$, we have the following results.*

1) If there exists positive integer $l < n$, satisfying $L = \bar{L} \otimes \mathbf{1}_{2^l}^T$, then

$$y = \bar{L} \ltimes_{i=1}^{n-l} x_i.$$

2) If there exist. positive integer $k < n$, satisfying $L = \mathbf{1}_{2^k}^T \otimes \bar{L}$, then

$$y = \bar{L} \ltimes_{i=k+1}^{n} x_i.$$

3) If there exist positive integers k and l with $k + l < n$, satisfying $L = \mathbf{1}_{2^k}^T \otimes \bar{L} \otimes \mathbf{1}_{2^l}^T$, then

$$y = \bar{L} \ltimes_{i=k+1}^{n-l} x_i.$$

4) If there exist positive integers k, h, l with $k + h + l = n$, satisfying $L = \bar{L}(I_{2^k} \otimes \mathbf{1}_{2^h}^T \otimes I_{2^l})$, then

$$y = \bar{L} \ltimes_{i=1}^{k} x_i \ltimes_{i=k+h+1}^{n} x_i.$$

Based on the analysis above, we could derive a simpler logical form if we use repeatedly Lemma 4 together with Corollary 2. Next, we give the simplification algorithm of a given logical function $\mathsf{y} = f(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n)$.

Algorithm 1 Simplification of Logical Functions

Input: A logical function f.

Output: The simplified logical function f.

Step 1: According to Lemma 3, write the structure matrix of logical function f.

Step 2: Check whether L satisfies the conditions of Propositions 1–2 and Corollary 1. If yes, then simplify the algebraic form $y = L \ltimes_{i=1}^{n} x_i$ according to Corollary 2, and delete the redundant variables. If not, then go to the next step.

Step 3: Use Lemma 4 to have

$$\mathsf{y} = [\mathsf{x}_1 \wedge f_1(\mathsf{x}_2, \ldots, \mathsf{x}_n)] \vee [\bar{\mathsf{x}}_1 \wedge f_2(\mathsf{x}_2, \ldots, \mathsf{x}_n)],$$

where $L_i := L\delta_2^i$ is the structure matrix $f_i(\cdot)$, $i = 1, 2$.

Step 4: If $L_i \in \mathcal{L}_{2\times 2}$, end the algorithm. Otherwise, for $i = 1, 2$, taking $L = L_i$ with logical functions f_i, return to Step 2.

Remark 2. *In Algorithm 1, the time complexity of Step 1 is $O(2^n)$. The number of iterations of Steps 2–4 is $n-1$. Then, the time complexity of Steps 2–4 is $O(2^n)+O(2^{n-1})+\cdots+O(2^2) = O(2^n)$. Consequently, the time complexity of Algorithm 1 is $O(2^n) + O(2^n) = O(2^n)$.*

Remark 3. *1) In [33], two methods, including the K-maps and the Quine-McCluskey method, are introduced. However, K-maps are awkward to use when there are more than four variables. On the other hand, the K-maps is a graphical method, and the Quine-McCluskey method is based on tabular representation.*

2) In this paper, a matrix-based method is proposed to simplify logical functions. It provides a new point of view for simplifying logical functions and is more convenient for programming by computers. On the other hand, our method is not limited to the number of variables and is only subject to the computer configuration. These are the main differences and innovations of this paper, compared with the K-maps and the Quine-McCluskey method.

The following example used to interpret the simplification of a logical function is from [33] (see Example 10, Chapter 12.4).

Example 5. *Simplify the following logical function:*

$$\begin{aligned} f =&(w \wedge x \wedge y \wedge \bar{z}) \vee (w \wedge \bar{x} \wedge y \wedge z) \vee (w \wedge \bar{x} \wedge y \wedge \bar{z}) \vee (\bar{w} \wedge x \wedge y \wedge z) \\ &\vee(\bar{w} \wedge x \wedge \bar{y} \wedge z) \vee (\bar{w} \wedge \bar{x} \wedge y \wedge z) \vee (\bar{w} \wedge \bar{x} \wedge \bar{y} \wedge z). \end{aligned} \tag{4.1}$$

Take the variable order w, x, y and z. Denote by F, W, X, Y and Z the corresponding vectors of f, w, x, y and z, respectively. Then, we have the algebraic form $F = LWXYZ$ of logical function f with

$$L = \delta_2 [2\ 1\ 2\ 2\ 1\ 1\ 2\ 2\ 1\ 2\ 1\ 2\ 1\ 2\ 1\ 2].$$

Using Lemma 4, we have

$$f = (w \wedge f_1(x, y, z)) \vee (\bar{w} \wedge f_2(x, y, z)) \tag{4.2}$$

with $L_1 = \delta_2 [2\ 1\ 2\ 2\ 1\ 1\ 2\ 2]$ and $L_2 = \delta_2 [1\ 2\ 1\ 2\ 1\ 2\ 1\ 2]$ being structure matrices of logical functions f_1 and f_2, respectively.

Using Lemma 4 again, it follows that

$$f_1 = (x \wedge f_{11}(y, z)) \vee (\bar{x} \wedge f_{12}(y, z)) \tag{4.3}$$

with $L_{11} = \delta_2 [2\ 1\ 2\ 2]$ and $L_{12} = \delta_2 [1\ 1\ 2\ 2]$ being structure matrices of logical functions f_{11} and f_{12}, respectively. Then,

$$f_{11} = (y \wedge f_{111}(z)) \vee (\bar{y} \wedge f_{112}(z)) \tag{4.4}$$

where the structure matrices of f_{111} and f_{112} are $L_{111} = \delta_2 [2\ 1]$ and $L_{112} = \delta_2 [2\ 2]$. From L_{111} and L_{112}, we have $f_{111} = \bar{z}$ and $f_{112} \equiv 0$, which implies that $f_{11} = y \wedge \bar{z}$.

Notice $L_{12} = I_2 \otimes \mathbf{1}_2^T$ and $f_{12} = y$. Thus, we derive that

$$f_1 = (x \wedge y \wedge \bar{z})) \vee (\bar{x} \wedge y). \tag{4.5}$$

On the other hand, since $L_2 = \mathbf{1}_4^T \otimes I_2$, we have $L_2XYZ = Z$, which implies that $f_2 = z$.

Therefore, we obtain that

$$f = (w \wedge x \wedge y \wedge \bar{z})) \vee (w \wedge \bar{x} \wedge y) \vee (\bar{w} \wedge z). \tag{4.6}$$

Using the Quine-McCluskey method, [33] derived that simplified functions are

$$f = (w \wedge y \wedge \bar{z}) \vee (w \wedge \bar{x} \wedge y) \vee (\bar{w} \wedge z)$$

and

$$f = (w \wedge y \wedge \bar{z}) \vee (\bar{x} \wedge y \wedge z) \vee (\bar{w} \wedge z).$$

It is easy to see that both (4.6) and [33] require seven logical gates to implement the logical function. Furthermore, our method is simpler than [33].

Remark 4. *It should be noticed that the order of variables often affects the simplification of logical functions. Different orders often result in different forms, since different algebraic forms correspond to different orders.*

We present a simple example to depict this point.

Example 6. *Consider the following logical function:*

$$f = (x \wedge y \vee \bar{z}) \vee (x \wedge \bar{y} \wedge \bar{z}). \tag{4.7}$$

Denote by F, X, Y and Z the corresponding vectors of f, x, y and z, respectively.

If we take the variable order x, y, z, then we can get the algebraic form $F = L_1XYZ$ of logical function f with

$$L_1 = \delta_2[1\ 1\ 2\ 1\ 2\ 1\ 2\ 1].$$

Using Algorithm 1, it is easy to simplify the logical function f as

$$f = (x \wedge y) \vee (x \wedge \bar{y} \wedge \bar{z}) \vee (\bar{x} \wedge \bar{z}). \tag{4.8}$$

meanwhile, if we take the variable order y, z, x, then the algebraic form of f is $F = L_2YZX$ with

$$L_2 = \delta_2[1\ 2\ 2\ 2\ 1\ 1\ 1\ 1].$$

Using Algorithm 1, we obtain that

$$f = (y \wedge z \wedge x) \vee \bar{y}. \tag{4.9}$$

It is clear that both (4.7) *and* (4.8) *require six logic gates, while* (4.9) *requires three logic gates.*

On the other hand, the method for simplifying logical functions is useful for constructing the logic forms of state feedback controllers. The following example is employed to give an explanation.

Example 7. *Consider the example in [34], where three state feedback controllers were designed to make set stabilization for Markovian jump Boolean control networks. However, [34] only gave the algebraic forms of the controllers: $u_i(t) = K_i x(t)$, $i = 0, 1, 2$ with*

$$K_0 = K_1 = \delta_2[1\ 1\ 1\ 1\ 1\ 1\ 1\ 1],$$

$$K_2 = \delta_2[1\ 1\ 2\ 2\ 1\ 1\ 2\ 2].$$

It is easy to see that $u_0 = u_1 = 1$ is the logical form of the former two. As for the third one u_2, in light of Algorithm 1, its logical form is derived as $u_2 = x_2$.

5. Application to design and simplification of circuits

The efficiency and the cost of designing a combinational circuit depend on the number and arrangement of its logic gates. A circuit in computers or other electric devices is often implemented by three logic gates: "OR" logic gate, "And" logic gate and "Inverter" logic gate. The process of designing a combinational circuit requires the table specifying the output for the combination of inputs. In computers, we always use binary expansions to code decimal expansions. For instance, decimal number 7 is represented by 0111, and 256 is encoded as 100, 000, 000.

To illustrate the applications of our method on the design of circuits, we build a circuit about decimal numbers, and the following example is from [33] (Example 8 in Chapter 12.4).

Example 8. *Using OR logic gates, AND logic gates and Inverter logic gates, build a circuit that produces an output* 1 *if the decimal digit is* 5 *or greater than* 5 *and an output* 0 *if the decimal digit is less than* 5*.*

Table 1. The values of f corresponding to different digits.

Digit	w	x	y	z	f
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1

Since there are 16 *combinations of four bits and only* 10 *decimal digits, there are* 6 *ones that are not used to encode digits. This gives us freedom in producing a simpler circuit with the desired output because the output values for all those combinations that never occur can be arbitrarily chosen.*

Let $f(w, x, y, z)$ denote the output of the circuit, where wxyz is a binary expansion of a decimal digit. The values of f are shown in Table 1. Take the variable order w, x, y and z. Denote by F, W, X, Y and Z the corresponding vectors of f, w, x, y and z, respectively. From TABLE 1, we have the algebraic form $F = LWXYZ$ of logical function f with

$$L = \delta_2 [* \ * \ * \ * \ * \ * \ 1 \ 1 \ 1 \ 1 \ 1 \ 2 \ 2 \ 2 \ 2 \ 2],$$

where $*$ is freedom, which can either take 1 *or* 2*.*

We take all $*$ by 1*, and then*

$$f = (w \wedge f_1(x, y, z)) \vee (\bar{w} \wedge f_2(x, y, z))$$

with $L_1 = \delta_2 [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1]$ and $L_2 = \delta_2 [1 \ 1 \ 1 \ 2 \ 2 \ 2 \ 2 \ 2]$ being structure matrices of logical functions f_1 and f_2, respectively.

It is obvious that $f_1 \equiv 1$ *since* $L_1 = \delta_2[1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$. *Using Lemma 4 again, it follows that*

$$f_2 = (x \wedge f_{21}(y,z)) \vee (\bar{x} \wedge f_{22}(y,z))$$

with $L_{21} = \delta_2[1\ 1\ 1\ 2]$ *and* $L_{22} = \delta_2[2\ 2\ 2\ 2]$ *being structure matrices of logical functions* f_{21} *and* f_{22}, *respectively. Then,* $f_{22} \equiv 0$, *and*

$$f_{21} = (y \wedge f_{211}(z)) \vee (\bar{y} \wedge f_{212}(z)).$$

where the structure matrices of f_{211} *and* f_{212} *are* $L_{211} = \delta_2[1\ 1]$ *and* $L_{212} = \delta_2[1\ 2]$. *From* L_{211} *and* L_{212}, *we have* $f_{211} \equiv 1$ *and* $f_{212} = z$. *Thus, we derive that*

$$f_2 = (x \wedge y) \vee (x \wedge \bar{y} \wedge z).$$

Hence, we derive that $f = w \vee (\bar{w} \wedge x \wedge y) \vee (\bar{w} \wedge x \wedge \bar{y} \wedge z)$, *which can be used to implement the circuit using* 5 *logic gates.*

In fact, noticing the detailed form of logical function f, *we can still simplify it slightly. If* $w = 1$, *then* $f = w \vee (x \wedge y) \vee (x \wedge \bar{y} \wedge z) = w = 1$; *if* $w = 0$, *then* $f = w \vee (x \wedge y) \vee (x \wedge \bar{y} \wedge z) = (x \wedge y) \vee (x \wedge \bar{y} \wedge z)$, *which derives the simpler form*

$$f = w \vee (x \wedge y) \vee (x \wedge \bar{y} \wedge z). \tag{5.1}$$

Then, logical function (5.1) *can be implemented by a circuit using only* 4 *logic gates, which is shown in Figure 2.*

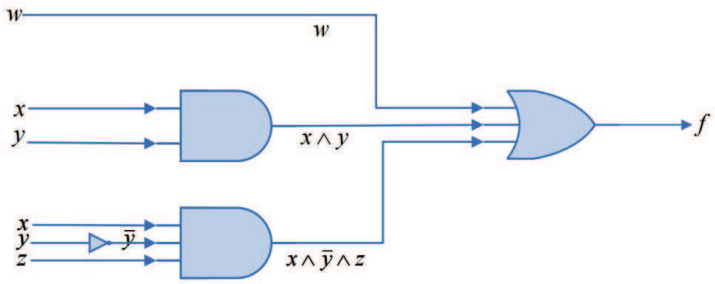

Figure 2. The circuit with output $f = w \vee (x \wedge y) \vee (x \wedge \bar{y} \wedge z)$.

In [33], using the freedom of 6 *combinations of four bits, [33] obtained three implementations. Two of them require* 7 *logic gates, and one needs* 3 *logic gates. Compared to the K-map method, our method could be more systematic. On the other hand, if we take other orders of variables* w, x, y, z, *or take value* 2 *for some* $*$ *in the structure matrix* L, *we can get other implementations.*

Example 9. *Consider a circuit which was given in [33] (see Example 4, Chapter 12), and the diagram of this circuit is shown in Figure 3. As we can see from Figure 3, to implement this circuit, 26 logic gates are needed. Next, we use Algorithm 1 to simplify this circuit.*

First, we can convert $f = (w \wedge x \wedge \bar{y} \wedge \bar{z}) \vee (w \wedge \bar{x} \wedge y \wedge z) \vee (w \wedge \bar{x} \wedge y \wedge \bar{z}) \vee (w \wedge \bar{x} \wedge \bar{y} \wedge \bar{z}) \vee (\bar{w} \wedge x \wedge \bar{y} \wedge \bar{z}) \vee (\bar{w} \wedge \bar{x} \wedge y \wedge \bar{z}) \vee (\bar{w} \wedge \bar{x} \wedge \bar{y} \wedge \bar{z})$ *into its algebraic form:*

$$\begin{aligned} F &= LWXYZ \\ &= \delta_2[2\ 2\ 2\ 1\ 1\ 1\ 2\ 1\ 2\ 2\ 2\ 1\ 2\ 1\ 2\ 1]WXYZ, \end{aligned} \tag{5.2}$$

where $L = \delta_2[2\ 2\ 2\ 1\ 1\ 1\ 2\ 1\ 2\ 2\ 2\ 1\ 2\ 1\ 2\ 1]$.

Similar to the iteration in Example 5 or 8, it follows from Algorithm 1 that the simplified logical function f *is*

$$f = (x \wedge \bar{y} \wedge \bar{z}) \vee (w \wedge \bar{x} \wedge y) \vee (\bar{x} \wedge \bar{z}). \tag{5.3}$$

Thus, the circuit can be simplified as Figure 4, which only needs 10 logic gates. In [33], using the K-maps method, f *was simplified as* $f = (\bar{y} \wedge \bar{z}) \vee (w \wedge \bar{x} \wedge y) \vee (\bar{x} \wedge \bar{z})$*, which also needs 10 logic gates. This shows that our method is effective for the simplification of circuits. Compared with the K-maps that is a graphical method, our method is matrix-based, which provides a new perspective to simplify logical functions.*

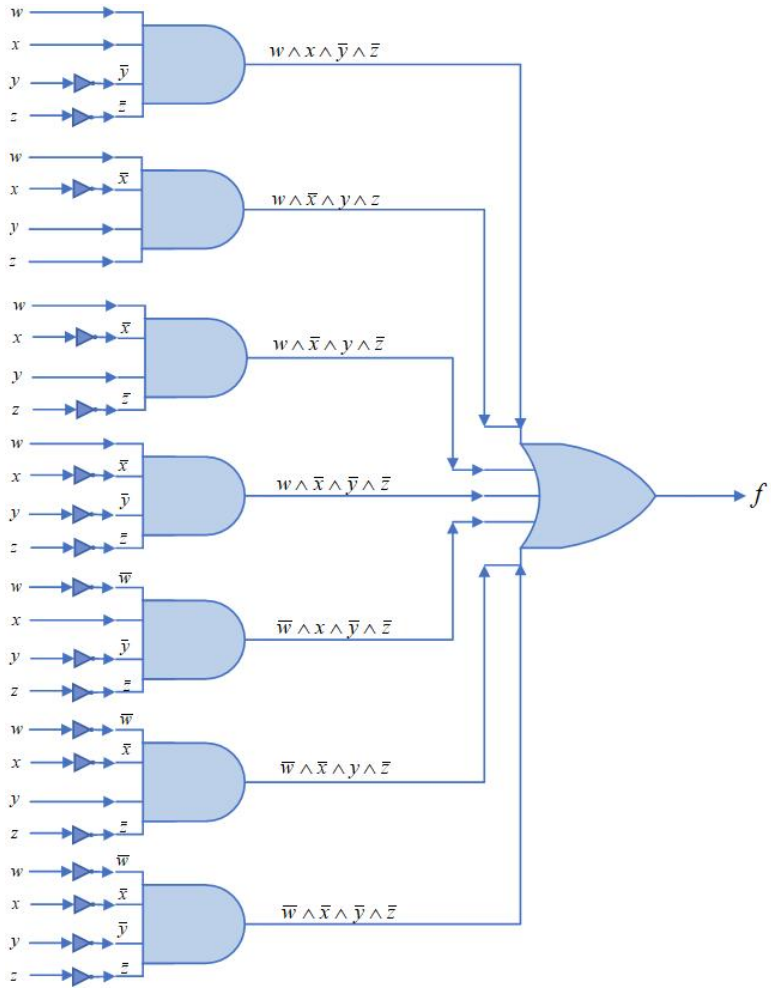

Figure 3. The circuit with output $f = (w \wedge x \wedge \bar{y} \wedge z) \vee (w \wedge \bar{x} \wedge y \wedge z) \vee (w \wedge \bar{x} \wedge y \wedge \bar{z}) \vee (w \wedge \bar{x} \wedge \bar{y} \wedge \bar{z}) \vee (\bar{w} \wedge x \wedge \bar{y} \wedge \bar{z}) \vee (\bar{w} \wedge \bar{x} \wedge y \wedge \bar{z}) \vee (\bar{w} \wedge \bar{x} \wedge \bar{y} \wedge \bar{z})$.

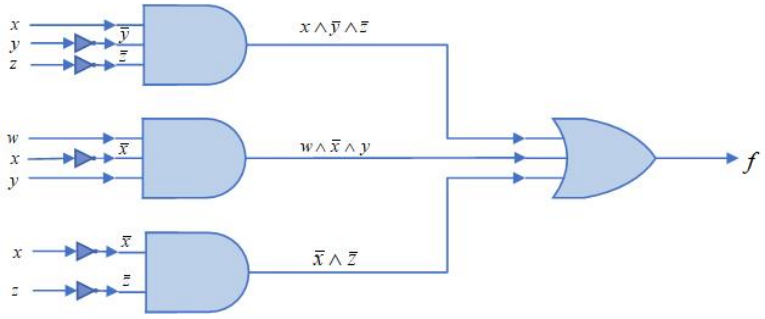

Figure 4. The circuit with output $f = (x \wedge \bar{y} \wedge \bar{z}) \vee (w \wedge \bar{x} \wedge y) \vee (\bar{x} \wedge \bar{z})$.

6. Conclusions

In this paper, the simplification problem of logical functions has been addressed. First, the decomposition conditions for logical matrices have been derived. Based on this, the simplification of logical functions has been investigated. Several criteria and an algorithm for simplifying logical functions have been proposed. Furthermore, the method in this paper has been applied to the design and simplification of circuits.

It has been mentioned in Remark 4 that the order of variables often affects the simplification of logical functions, which has been shown via Example 6. Therefore, in order to get a simpler form of a given logic function, we should try more different variable orders, even try all different orders to get the simplest one. However, it may lead to more computation complexity. Thus, how to get a simpler form with less computation complexity is of great significance in practice and worthy of further investigation in the future works.

Acknowledgments

This work was supported by the National Natural Science Foundation (NNSF) of China under Grant 61877036.

Conflict of interest

The authors declare that there are no conflicts of interest.

References

1. K. Hilary, The laws of thought, *Philoso. Phenomen. Res.*, **52** (1992), 895–911. https://doi.org/10.2307/2107916
2. C. E. Shannon, A symbolic analysis of relay and switching circuits, *Trans. Am. Inst. Electr. Eng.*, **57** (1938), 713–723. https://doi.org/10.1109/t-aiee.1938.5057767
3. D. Cheng, J. Feng, J. Zhao, S. Fu, On adequate sets of multi-valued logic, *J. Franklin Inst.*, **358** (2021), 6705–6722. https://doi.org/10.1016/j.jfranklin.2021.07.003
4. M. Karnaugh, The map method for synthesis of combinational logic circuits, *Trans. Am. Inst. Electr. Eng.*, **72** (1953), 593–599. https://doi.org/10.1109/TCE.1953.6371932

5. J. P. Hayes, Introduction to digital logic design, *Prent. Hall*, 1993. https://doi.org/978-0-201-15461-0
6. R. H. Katz, G. Borriello, Contemporary logic design, *Prent. Hall*, 2004. https://doi.org/10.1016/0026-2692(95)90052-7
7. X. L. Wang, X. Y. Zhang, W. L. Wang, A new representation and simplification method of logic function, in *International Conference on Computer & Automation Engineering*, 2010. https://doi.org/10.1109/ICCAE.2010.5451556
8. S. Kahramanli, S. Guenes, S. Sahan, F. Basciftci, A new method based on cube algebra for the simplification of logic functions, *Arab. J. Sci. Eng.*, **32** (2007), 101–114. https://doi.org/10.1016/j.agee.2006.06.020
9. D. Cheng, Semi-tensor product of matrices and its applications-a survey, *Methods Appl. Anal.*, **3** (2007), 641–668. https://doi.org/10.1007/10984413_5
10. D. Cheng, H. Qi, Z. Li, Analysis and control of boolean networks: A semi-tensor product approach, *London: Springer-Verlag*, 2011. https://doi.org/10.3724/SP.J.1004.2011.00529
11. F. Li, Y. Tang, Pinning controllability for a Boolean network with arbitrary disturbance inputs, *IEEE Trans. Cybern.*, **51** (2019), 3338–3347. https://doi.org/10.1109/TCYB.2019.2930734
12. Y. Li, H. Li, G. Xiao, Set controllability of Markov jump switching Boolean control networks and its applications, *Nonlinear Anal. Hybri.*, **45** (2022), 101179. https://doi.org/10.1016/j.nahs.2022.101179
13. Y. Yu, M. Meng, J. Feng, Observability of Boolean networks via matrix equations, *Automatica*, **111** (2020), 108621. https://doi.org/10.1016/j.automatica.2019.108621
14. S. Zhu, J. Lu, L. Lin, Y. Liu, Minimum-time and minimum-triggering control for the observability of stochastic Boolean networks, *IEEE Trans. Autom. Control*, **67** (2022), 1558–1565. https://doi.org/10.1109/TAC.2021.3069739
15. B. Wang, J. Feng, H. Li, Y. Yu, On detectability of Boolean control networks, *Nonlinear Anal. Hybri.*, **36** (2020), 100859. https://doi.org/10.1016/j.nahs.2020.100859
16. Z. Gao, B. Wang, J. Feng, T. Li, Finite automata approach to reconstructibility of switched Boolean control networks, *Neurocomputing*, **454** (2021), 34–44. https://doi.org/10.1016/j.neucom.2021.05.019
17. C. V. A. Yerudkar, L. Glielmo, Feedback stabilization control design for switched Boolean control networks, *Automatica*, **115** (2020), 108934. https://doi.org/10.1016/j.automatica.2020.108934
18. M. Meng, J. Lam, J. Feng, K. Cheung, Stability and stabilization of Boolean networks with stochastic delays, *IEEE Trans. Autom. Control*, **64** (2019), 790–796. https://doi.org/10.1109/TAC.2018.2835366
19. N. Bof, E. Fornasini, M. Valcher, Output feedback stabilization of Boolean control networks, *Automatica*, **57** (2015), 21–28. https://doi.org/10.1016/j.automatica.2015.03.032
20. J. Lu, L. Sun, D. W. C. Liu, Y. Ho, J. Cao, Stabilization of Boolean control networks under aperiodic sampled-data control, *SIAM J. Control Optim.*, **56** (2018), 4385–4404. https://doi.org/10.1137/18M1169308

21. Y. Wu, T. Shen, A finite convergence criterion for the discounted optimal control of stochastic logical networks, *IEEE Trans. Autom. Control*, **63** (2018), 262–268. https://doi.org/10.1109/TAC.2017.2720730
22. Y. Wu, X. Sun, X. Zhao, T. Shen, Optimal control of Boolean control networks with average cost: a policy iteration approach, *Automatica*, **100** (2019), 378–387. https://doi.org/10.1016/j.automatica.2018.11.036
23. Y. Wu, Y. Guo, M. Toyoda, Policy iteration approach to the infinite horizon average optimal control of probabilistic Boolean networks, *IEEE Trans. Neural Netw. Learn. Syst.*, **32** (2021), 2910–2924. https://doi.org/10.1109/TNNLS.2020.3008960
24. S. Wang, H. Li, New results on the disturbance decoupling of Boolean control networks, *IEEE Control Syst. Lett.*, **5** (2021), 1157–1162. https://doi.org/10.1109/LCSYS.2020.3017645
25. Y. Li, J. Zhu, B. Li, Y. Liu, J. Lu, A necessary and sufficient graphic condition for the original disturbance decoupling of Boolean networks, *IEEE Trans. Autom. Control*, **66** (2021), 3765–3772. https://doi.org/10.1109/TAC.2020.3025507
26. J. Zhang, J. Sun, Exponential synchronization of complex networks with continuous dynamics and Boolean mechanism, *Neurocomputing*, **307** (2018), 146–152. https://doi.org/10.1016/j.neucom.2018.03.061
27. R. Li, T. Chu, X. Wang, Bisimulations of Boolean control networks, *SIAM J. Control Optim.*, **56** (2018), 388–416. https://doi.org/10.1137/17M1117331
28. Q. Zhang, J. Feng, The solution and stability of continuous-time cross-dimensional linear systems, *Front. Inf. Tech. El.*, **22** (2021), 210-221. https://doi.org/10.1631/FITEE.1900504
29. Y. Zheng, J. Feng, Output tracking of delayed logical control networks with multi-constraints, *Front. Inf. Tech. El.*, **21** (2020), 316–323. https://doi.org/10.1631/FITEE.1900376
30. J. Yue, Y. Yan, Z. Chen, X. Jin, Identification of predictors of Boolean networks from observed attractor states, *Math. Methods Appl. Sci.*, **42** (2019), 3848–3864. https://doi.org/10.1002/mma.5616
31. D. Cheng, H. Qi, Controllability and observability of Boolean control networks, *Automatica*, **45** (2009), 1659–1667. https://doi.org/10.1007/s00034-014-9900-8
32. M. D. Ciletti, Advanced digital design with the verilog HDL, *Prent. Hall Upper Saddle River*, 2003. https://doi.org/978-0-13-089161-7
33. K. H. Rosen, Discrete mathematics and its applications, *New York: McGraw-Hill*, 2002. https://doi.org/978-0-07-242434-8
34. S. Zhu, J. Feng, The set stabilization problem for Markovian jump Boolean control networks: An average optimal control approach, *Appl. Math. Comput.*, **402** (2021), 126133. https://doi.org/10.1016/j.amc.2021.126133

AIMS Press